OpenTISim Documentation

Release v0.6.2

Mark van Koningsveld

Mar 09, 2020

Contents:

OpenTISim is a python package for the evaluation of investment decisions for terminals.

Welcome to OpenTISim documentation! Please check the contents below for information on installation, getting started and actual example code. If you want to dive straight into the code you can check out our GitHub page or the working examples presented in Jupyter Notebooks.

CHAPTER 1

Installation

1.1 Stable release

To install OpenTISim, run this command in your terminal:

```
# Use pip to install OpenTISim
pip install opentisim
```

This is the preferred method to install OpenTISim, as it will always install the most recent stable release.

If you do not pip installed, this Python installation guide can guide you through the process.

1.2 From sources

The sources for OpenTISim can be downloaded from the Github repo.

You can either clone the public repository:

```
# Use git to clone OpenTISim
git clone git://github.com/TUDelft-CITG/OpenTISim
```

Or download the tarball:

```
# Use curl to obtain the tarball
curl  -OL https://github.com/TUDelft-CITG/OpenTISim/tarball/master
```

Once you have a copy of the source, you can install it with:

```
# Use python to install
python setup.py install
```

CHAPTER 2

OpenTISim

This page lists all functions and classes available in the OpenTISim.model and OpenTISim.core modules. For examples on how to use these submodules please check out the Examples page, information on installing OpenTISim can be found on the Installation page.

2.1 Submodules

The main components are the Model module and the Core module. All of their components are listed below.

2.2 opentisim.agribulk_defaults module

Defaults for following objects:

- 1. Quay_wall
- 2. Berth
- 3. Cyclic_Unloader
  - Gantry crane
  - Harbour crane
  - Mobile crane

    Continuous_Unloader
  - Continuous screw
- 4. Conveyor
  - Hinterland conveyor
  - Quay conveyor
- 5. Storage

  - Silo
  - Warehouse
- 6. Unloading_station
  - Hinterland station
- 7. Commodity
  - Maize
  - Soybean
  - Wheat
- 8. Vessel
  - Handysize
  - Handymax
  - Panamax
- 9. Labour

Default values are based on Claes 2018; Corbeau 2018; Daas 2018; Juha 2018; Kranendonk 2018; Schutz 2018; Schuurmans 2018 and Verstegen 2018

2.3 opentisim.agribulk_mixins module

Basic properties mixins:

- identifiable_properties_mixin
- history_properties_mixin
- hascapex_properties_mixin
- hasopex_properties_mixin
- hasrevenue_properties_mixin
- hastriggers_properties_mixin
- quay_wall_properties_mixin
- berth_properties_mixin
- cyclic_properties_mixin
- continuous_properties_mixin
- conveyor_properties_mixin
- storage_properties_mixin
- unloading_station_properties_mixin
- commodity_properties_mixin
- vessel_properties_mixin
- labour_properties_mixin
- hasscenario_properties_mixin

class opentisim.agribulk_mixins.**berth_properties_mixin**(*crane_type*, *max_cranes*, *delivery_time*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**commodity_properties_mixin**(*handling_fee*, *handysize_perc*, *handymax_perc*, *panamax_perc*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**continuous_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation_perc*, *maintenance_perc*, *consumption*, *insurance_perc*, *crew*, *crane_type*, *peak_capacity*, *eff_fact*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**conveyor_properties_mixin**(*type*, *length*, *ownership*, *delivery_time*, *lifespan*, *unit_rate_factor*, *mobilisation*, *maintenance_perc*, *insurance_perc*, *consumption_constant*, *consumption_coefficient*, *crew*, *utilisation*, *capacity_steps*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**cyclic_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation_perc*, *maintenance_perc*, *consumption*, *insurance_perc*, *crew*, *crane_type*, *lifting_capacity*, *hourly_cycles*, *eff_fact*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**energy_properties_mixin**(*price*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**hascapex_properties_mixin**(*capex=[]*, **args*, ***kwargs*)

Bases: object

Something has CAPEX

capex: list with cost to be applied from investment year

class opentisim.agribulk_mixins.**hasopex_properties_mixin**(*labour=[]*, *maintenance=[]*, *energy=[]*, *insurance=[]*, *lease=[]*, *demurrage=[]*, *residual=[]*, **args*, ***kwargs*)

Bases: object

Something has OPEX

opex: list with cost to be applied from investment year

class opentisim.agribulk_mixins.**hasrevenue_properties_mixin**(*renevue=[]*, **args*, ***kwargs*)

Bases: object

Something has Revenue

revenue: list with revenues to be applied from investment year

class opentisim.agribulk_mixins.**hasscenario_properties_mixin**(*historic_data=[]*, *scenario_data=[]*, **args*, ***kwargs*)

Bases: object

Something has a scenario

historic_data: observed demand scenario_data: generated estimates of future demand

plot_demand(*width=0.1*, *alpha=0.6*, *fontsize=20*)
generate a histogram of the demand data

scenario_random(*startyear=2019*, *lifecycle=20*, *rate=1.02*, *mu=0.01*, *sigma=0.065*)
trend generated from random growth rate increments

class opentisim.agribulk_mixins.**hastriggers_properties_mixin**(*triggers=[]*, **args*, ***kwargs*)

Bases: object

Something has InvestmentTriggers

triggers: list with revenues to be applied from investment year

class opentisim.agribulk_mixins.**history_properties_mixin**(*year_purchase=[]*, *year_online=[]*, **args*, ***kwargs*)

Bases: object

Something that has a purchase history

purchase_date: year in which the decision was made to add another element online_date: year by which the elements starts to perform

class opentisim.agribulk_mixins.**identifiable_properties_mixin**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: object

Something that has a name and id

name: a name id: a unique id generated with uuid

class opentisim.agribulk_mixins.**labour_properties_mixin**(*international_salary*, *international_staff*, *local_salary*, *local_staff*, *operational_salary*, *shift_length*, *annual_shifts*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**quay_wall_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *mobilisation_min*, *mobilisation_perc*, *maintenance_perc*, *insurance_perc*, *freeboard*, *Gijt_constant_2*, *Gijt_constant*, *Gijt_coefficient*, *max_sinkage*, *wave_motion*, *safety_margin*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**storage_properties_mixin**(*type*, *ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation_min*, *mobilisation_perc*, *maintenance_perc*, *crew*, *insurance_perc*, *storage_type*, *consumption*, *capacity*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**train_properties_mixin**(*wagon_payload*, *number_of_wagons*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**unloading_station_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation*, *maintenance_perc*, *insurance_perc*, *consumption*, *crew*, *production*, *wagon_payload*, *number_of_wagons*, *prep_time*, **args*, ***kwargs*)

Bases: object

class opentisim.agribulk_mixins.**vessel_properties_mixin**(*type*, *call_size*, *LOA*, *draft*, *beam*, *max_cranes*, *all_turn_time*, *mooring_time*, *demurrage_rate*, **args*, ***kwargs*)

Bases: object

2.4 opentisim.agribulk_objects module

Main generic object classes:

- 1. Quay_wall
- 2. Berth
- 3. Cyclic_Unloader
  - Gantry crane
  - Harbour crane
  - Mobile crane

    Continuous_Unloader
  - Continuous screw
- 4. Conveyor
  - Hinterland conveyor
  - Quay conveyor
- 5. Storage
  - Silo

  - Warehouse
- 6. Unloading_station
  - Hinterland station
- 7. Commodity
  - Maize
  - Soybean
  - Wheat
- 8. Vessel
  - Handysize
  - Handymax
  - Panamax
- 9. Labour

class opentisim.agribulk_objects.**Berth**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.history_properties_mixin*, *opentisim.agribulk_mixins.berth_properties_mixin*, *opentisim.agribulk_mixins.hascapex_properties_mixin*, *opentisim.agribulk_mixins.hasopex_properties_mixin*, *opentisim.agribulk_mixins.hasrevenue_properties_mixin*, *opentisim.agribulk_mixins.hastriggers_properties_mixin*

class opentisim.agribulk_objects.**Commodity**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.commodity_properties_mixin*, *opentisim.agribulk_mixins.hasscenario_properties_mixin*

class opentisim.agribulk_objects.**Continuous_Unloader**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.history_properties_mixin*, *opentisim.agribulk_mixins.continuous_properties_mixin*, *opentisim.agribulk_mixins.hascapex_properties_mixin*, *opentisim.agribulk_mixins.hasopex_properties_mixin*, *opentisim.agribulk_mixins.hasrevenue_properties_mixin*, *opentisim.agribulk_mixins.hastriggers_properties_mixin*

class opentisim.agribulk_objects.**Conveyor_Hinter**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.history_properties_mixin*, *opentisim.agribulk_mixins.conveyor_properties_mixin*, *opentisim.agribulk_mixins.hascapex_properties_mixin*, *opentisim.agribulk_mixins.hasopex_properties_mixin*, *opentisim.agribulk_mixins.hasrevenue_properties_mixin*, *opentisim.agribulk_mixins.hastriggers_properties_mixin*

class opentisim.agribulk_objects.**Conveyor_Quay**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.history_properties_mixin*, *opentisim.agribulk_mixins.conveyor_properties_mixin*, *opentisim.agribulk_mixins.hascapex_properties_mixin*, *opentisim.*

agribulk_mixins.hasopex_properties_mixin, *opentisim.agribulk_mixins.hasrevenue_properties_mixin*, *opentisim.agribulk_mixins.hastriggers_properties_mixin*

class opentisim.agribulk_objects.**Cyclic_Unloader**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.history_properties_mixin*, *opentisim.agribulk_mixins.cyclic_properties_mixin*, *opentisim.agribulk_mixins.hascapex_properties_mixin*, *opentisim.agribulk_mixins.hasopex_properties_mixin*, *opentisim.agribulk_mixins.hasrevenue_properties_mixin*, *opentisim.agribulk_mixins.hastriggers_properties_mixin*

class opentisim.agribulk_objects.**Energy**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.energy_properties_mixin*

class opentisim.agribulk_objects.**Labour**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.labour_properties_mixin*

class opentisim.agribulk_objects.**Quay_wall**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.quay_wall_properties_mixin*, *opentisim.agribulk_mixins.history_properties_mixin*, *opentisim.agribulk_mixins.hascapex_properties_mixin*, *opentisim.agribulk_mixins.hasopex_properties_mixin*, *opentisim.agribulk_mixins.hasrevenue_properties_mixin*, *opentisim.agribulk_mixins.hastriggers_properties_mixin*

class opentisim.agribulk_objects.**Storage**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.history_properties_mixin*, *opentisim.agribulk_mixins.storage_properties_mixin*, *opentisim.agribulk_mixins.hascapex_properties_mixin*, *opentisim.agribulk_mixins.hasopex_properties_mixin*, *opentisim.agribulk_mixins.hasrevenue_properties_mixin*, *opentisim.agribulk_mixins.hastriggers_properties_mixin*

class opentisim.agribulk_objects.**Train**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.train_properties_mixin*

class opentisim.agribulk_objects.**Unloading_station**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.unloading_station_properties_mixin*

class opentisim.agribulk_objects.**Vessel**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.agribulk_mixins.identifiable_properties_mixin*, *opentisim.agribulk_mixins.vessel_properties_mixin*

2.5 opentisim.agribulk_system module

class opentisim.agribulk_system.**System**(*startyear=2019*, *lifecycle=20*, *operational_hours=5840*, *debug=False*, *elements=[]*, *crane_type_defaults={'consumption': 485, 'crane_type': 'Mobile crane', 'crew': 3, 'delivery_time': 1, 'eff_fact': 0.35, 'hourly_cycles': 25, 'insurance_perc': 0.01, 'lifespan': 40, 'lifting_capacity': 30, 'maintenance_perc': 0.02, 'mobilisation_perc': 0.15, 'name': 'Mobile_crane_01', 'ownership': 'Terminal operator', 'unit_rate': 3325000}*, *storage_type_defaults={'capacity': 6000, 'consumption': 0.002, 'crew': 1, 'delivery_time': 1, 'insurance_perc': 0.01, 'lifespan': 30, 'maintenance_perc': 0.02, 'mobilisation_min': 200000, 'mobilisation_perc': 0.003, 'name': 'Silo_01', 'ownership': 'Terminal operator', 'storage_type': 'Silos', 'type': 'silo', 'unit_rate': 60}*, *allowable_waiting_service_time_ratio_berth=0.3*, *allowable_berth_occupancy=0.4*, *allowable_dwelltime=0.049315068493150684*, *allowable_waiting_service_time_ratio_station=0.5*, *allowable_station_occupancy=0.4*)

Bases: object

This class implements the 'complete supply chain' concept (Van Koningsveld et al, 2020) for agribulk terminals.

The module allows variation of the type of quay crane used and the type of storage used.

Terminal development is governed by the following triggers: - the allowable waiting time as a factor of service time at the berth - the allowable dwell time of cargo in the storage area, and - the allowable waiting time as a factor of service time at the station.

berth_invest(*year*, *handysize*, *handymax*, *panamax*)

Given the overall objectives for the terminal apply the following decision recipe (Van Koningsveld and Mulder, 2004) for the berth investments.

Decision recipe Berth: QSC: berth_occupancy & allowable_waiting_service_time_ratio Benchmarking procedure: there is a problem if the estimated berth_occupancy triggers a waiting time over service time ratio that is larger than the allowed waiting time over service time ratio

- allowable_waiting_service_time_ratio = .30 # 30% (see PIANC (2014))
- **a berth needs:**
  - a quay, and
  - cranes (min:1 and max: max_cranes)
- **berth occupancy depends on:**
  - total_calls, total_vol and time needed for mooring, unmooring
  - total_service_capacity as delivered by the cranes
- berth occupancy in combination with nr of berths is used to lookup the waiting over service time ratio

Intervention procedure: invest enough to make the planned waiting service time ratio < allowable waiting

service time ratio - adding berths, quays and cranes decreases berth_occupancy_rate (and increases the number of servers)

which will yield a smaller waiting time over service time ratio

calculate_berth_occupancy (*year*, *handysize_calls*, *handymax_calls*, *panamax_calls*)

- Find all cranes and sum their effective_capacity to get service_capacity
- Divide callsize_per_vessel by service_capacity and add mooring time to get total time at berth
- Occupancy is total_time_at_berth divided by operational hours

calculate_demurrage_cost (*year*)

Find the demurrage cost per type of vessel and sum all demurrage cost

calculate_energy_cost (*year*)

1. calculate the value of the total demand in year (demand * handling fee)

2. calculate the maximum amount that can be handled (service capacity * operational hours) Terminal.revenues is the minimum of 1. and 2.

calculate_revenue (*year*)

1. calculate the value of the total demand in year (demand * handling fee)

2. calculate the maximum amount that can be handled (service capacity * operational hours) Terminal.revenues is the minimum of 1. and 2.

calculate_station_occupancy (*year*)

The station occupancy is calculated based on the service rate for the throughput of the online quay unloaders (effective capacity * occupancy). The unloading station should at least be able to handle the throughput by the online quay unloaders at a level that the station occupancy planned remains below the target occupancy level.

calculate_vessel_calls (*year=2019*)

Calculate volumes to be transported and the number of vessel calls (both per vessel type and in total)

check_crane_slot_available ()

conveyor_hinter_invest (*year*, *agribulk_defaults_hinterland_conveyor_data*)

Given the overall objectives for the terminal apply the following decision recipe (Van Koningsveld and Mulder, 2004) for the hinter conveyor investments.

Operational objective: maintain a hinter conveyor capacity that at least matches the station unloading capacity (so basically the hinter conveyors follow what happens on the station)

Decision recipe quay conveyor: QSC: hinter_conveyor_capacity planned Benchmarking procedure: there is a problem when the hinter_conveyor_capacity_planned is smaller than the station_service_rate_planned

For the hinter conveyor investments the strategy is to at least match the unloading station capacity

Intervention procedure: the intervention strategy is to add hinter conveyors until the trigger is achieved

- find out how much hinter_conveyor_capacity is planned
- find out how much station_service_rate_planned is planned
- add hinter_conveyor_capacity until it matches station_service_rate_planned

conveyor_quay_invest (*year*, *agribulk_defaults_quay_conveyor_data*)

Given the overall objectives for the terminal apply the following decision recipe (Van Koningsveld and Mulder, 2004) for the quay conveyor investments.

Operational objective: maintain a quay conveyor capacity that at least matches the quay crane capacity (so basically the quay conveyors follow what happens on the berth)

Decision recipe quay conveyor: QSC: quay_conveyor_capacity planned Benchmarking procedure: there is a problem when the quay_conveyor_capacity_planned is smaller than the quay_crane_service_rate_planned

For the quay conveyor investments the strategy is to at least match the quay crane processing capacity

Intervention procedure: the intervention strategy is to add quay conveyors until the trigger is achieved

- find out how much quay_conveyor_capacity is planned
- find out how much quay_crane_service_rate is planned
- add quay_conveyor_capacity until it matches quay_crane_service_rate

crane_invest (*year*)

Given the overall objectives for the terminal apply the following decision recipe (Van Koningsveld and Mulder, 2004) for the crane investments.

Decision recipe Crane: QSC: planned waiting over service time ratio Benchmarking procedure (triggered in self.berth_invest): there is a problem when the planned planned waiting over service time ratio is larger than the max allowable waiting over service time ratio Intervention procedure: invest until planned waiting over service time ratio is below the max allowable waiting over service time ratio

quay_invest (*year*, *length*, *depth*)

Given the overall objectives for the terminal apply the following decision recipe (Van Koningsveld and Mulder, 2004) for the quay investments.

Decision recipe Quay: QSC: quay_per_berth Benchmarking procedure (triggered in self.berth_invest): there is a problem when

the number of berths > the number of quays, but also while the planned waiting over service time ratio is too large

Intervention procedure: invest enough to make sure that each quay has a berth and the planned waiting over service time ratio is below the max allowable waiting over service time ratio

- adding quay will increase quay_per_berth
- quay_wall.length must be long enough to accommodate largest expected vessel
- quay_wall.depth must be deep enough to accommodate largest expected vessel
- quay_wall.freeboard must be high enough to accommodate largest expected vessel

simulate ()

The 'simulate' method implements the terminal investment strategy for this terminal class.

This method automatically generates investment decisions, parametrically derived from overall demand trends and a number of investment triggers.

Generic approaches based on: - PIANC. 2014. Master plans for the development of existing ports. MarCom - Report 158, PIANC - Van Koningsveld, M. (Ed.), Verheij, H., Taneja, P. and De Vriend, H.J. (in preparation). Ports and Waterways.

Navigating the changing world. TU Delft, Delft, The Netherlands.

- Van Koningsveld, M. and J. P. M. Mulder. 2004. Sustainable Coastal Policy Developments in the Netherlands. A Systematic Approach Revealed. Journal of Coastal Research 20(2), pp. 375-385

Specific application based on (modifications have been applied where deemed an improvement): - Ijzermans, W., 2019. Terminal design optimization. Adaptive agribulk terminal planning

in light of an uncertain future. Master's thesis. Delft University of Technology, Netherlands. URL: http://resolver.tudelft.nl/uuid:7ad9be30-7d0a-4ece-a7dc-eb861ae5df24.

The simulate method applies Frame of Reference style decisions while stepping through each year of the terminal lifecycle and checks if investments are needed (in light of strategic objective, operational objective, QSC, decision recipe and intervention method):

1. for each year estimate the anticipated vessel arrivals based on the expected demand
2. for each year evaluate which investments are needed given the strategic and operational objectives
3. for each year calculate the energy costs (requires insight in realized demands)
4. for each year calculate the demurrage costs (requires insight in realized demands)
5. for each year calculate terminal revenues (requires insight in realized demands)
6. collect all cash flows (capex, opex, revenues)
7. calculate PV's and aggregate to NPV

storage_invest (*year*, *agribulk_defaults_storage_data*)

Given the overall objectives for the terminal apply the following decision recipe (Van Koningsveld and Mulder, 2004) for the storage investments.

Operational objective: maintain a storage capacity that is large enough to at least contain one time the largest vessel call size or that is large enough to accommodate a maximum allowable dwell time plus 10 percent.

Decision recipe storage: QSC: storage_capacity Benchmarking procedure: there is a problem when the storage_capacity is too small to store one time the largest call size or when it is too small to allow for a predetermined max allowable dwell time

The max allowable dwell time is here determined as 5% of the annual demand, increased by 10% (PIANC, 2014)

Intervention procedure: the intervention strategy is to add storage until the benchmarking trigger is achieved. The trigger is the max of one call size, or the volume derived from the dwell time requirement.

terminal_capacity_plot (*width=0.25*, *alpha=0.6*, *fontsize=20*)

Gather data from Terminal and plot which elements come online when

terminal_elements_plot (*width=0.1*, *alpha=0.6*, *fontsize=20*, *demand_step=100000*)

Gather data from Terminal and plot which elements come online when

train_call (*year*)

Calculation of the train calls per year, this is calculated from: - find out how much throughput there is - find out how much cargo the train can transport - calculate the numbers of train calls

unloading_station_invest (*year*)

The operational objective for the investment strategy for unloading stations is to have sufficient planned unloading stations to keep the station occupancy below a given threshold for the quay crane capacity planned.

current strategy is to add unloading stations as soon as a service trigger is achieved - find out how much service capacity is online - find out how much service capacity is planned - find out how much service capacity is needed - add service capacity until service_trigger is no longer exceeded

2.6 opentisim.container_defaults module

Main generic object classes: - 1. Quay_wall - 2. Berth - 3. Cyclic_Unloader

- STS crane
- 4. Horizontal transport
  - Tractor trailer
- 5. Commodity
  - TEU
- 6. Containers
  - Laden
  - Reefer
  - Empty
  - OOG
- 7. Laden and reefer stack
- 8. Stack equipment
- 9. Empty stack
- 10. OOG stack
- 11. Gates
- 12. Empty handler
- 13. Vessel
- 14. Labour
- 15. Energy
- 16. General
- 17. Indirect Costs

2.7 opentisim.container_mixins module

Basic properties mixins:

- identifiable_properties_mixin
- history_properties_mixin
- hascapex_properties_mixin
- hasopex_properties_mixin
- hasrevenue_properties_mixin

- hastriggers_properties_mixin
- quay_wall_properties_mixin
- berth_properties_mixin
- cyclic_properties_mixin
- transport_properties_mixin
- container_properties_mixin
- laden_stack_properties_mixin
- empty_stack_properties_mixin
- oog_stack_properties_mixin
- stack_equipment_properties_mixin
- gate_properties_mixin
- empty_handler_properties_mixin
- commodity_properties_mixin
- vessel_properties_mixin
- labour_properties_mixin
- hasscenario_properties_mixin

class opentisim.container_mixins.**berth_properties_mixin**(*crane_type*, *max_cranes*, *delivery_time*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**commodity_properties_mixin**(*handling_fee*, *fully_cellular_perc*, *panamax_perc*, *panamax_max_perc*, *post_panamax_I_perc*, *post_panamax_II_perc*, *new_panamax_perc*, *VLCS_perc*, *ULCS_perc*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**container_properties_mixin**(*type*, *teu_factor*, *dwell_time*, *peak_factor*, *stack_occupancy*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**cyclic_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation_perc*, *maintenance_perc*, *consumption*, *insurance_perc*, *crew*, *crane_type*, *lifting_capacity*, *hourly_cycles*, *eff_fact*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**empty_handler_properties_mixin**(*type*, *ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation*, *maintenance_perc*, *crew*, *salary*, *fuel_consumption*, *required*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**empty_stack_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *mobilisation*, *maintenance_perc*, *width*, *height*, *length*, *capacity*, *gross_tgs*, *area_factor*, *pavement*, *drainage*, *household*, *digout*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**energy_properties_mixin**(*price*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**gate_properties_mixin**(*type*, *ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation*, *maintenance_perc*, *crew*, *salary*, *canopy_costs*, *area*, *staff_gates*, *service_gates*, *design_capacity*, *exit_inspection_time*, *entry_inspection_time*, *peak_hour*, *peak_day*, *peak_factor*, *truck_moves*, *operating_days*, *capacity*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**general_services_mixin**(*type*, *office*, *office_cost*, *workshop*, *workshop_cost*, *fuel_station_cost*, *scanning_inspection_area*, *scanning_inspection_area_cost*, *lighting_mast_required*, *lighting_mast_cost*, *firefight_cost*, *maintenance_tools_cost*, *terminal_operating_software_cost*, *electrical_station_cost*, *repair_building*, *repair_building_cost*, *ceo*, *secretary*, *administration*, *hr*, *commercial*, *operations*, *engineering*, *security*, *general_maintenance*, *crew_required*, *delivery_time*, *lighting_consumption*, *general_consumption*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**hascapex_properties_mixin**(*capex=[]*, **args*, ***kwargs*)

Bases: object

Something has CAPEX

capex: list with cost to be applied from investment year

class opentisim.container_mixins.**hasland_properties_mixin**(*land_use=[]*, **args*, ***kwargs*)

Bases: object

Something has land use [m^2]

land_use: list with land use to be applied from investment year

class opentisim.container_mixins.**hasopex_properties_mixin**(*labour=[]*, *maintenance=[]*, *energy=[]*, *insurance=[]*, *lease=[]*, *demurrage=[]*, *residual=[]*, *fuel=[]*, **args*, ***kwargs*)

Bases: object

Something has OPEX

opex: list with cost to be applied from investment year

class opentisim.container_mixins.**hasrevenue_properties_mixin**(*renevue=[]*, **args*, ***kwargs*)

Bases: object

Something has Revenue

revenue: list with revenues to be applied from investment year

class opentisim.container_mixins.**hasscenario_properties_mixin**(*historic_data=[]*, *scenario_data=[]*, **args*, ***kwargs*)

Bases: object

Something has a scenario

historic_data: observed demand scenario_data: generated estimates of future demand

plot_demand(*width=0.1*, *alpha=0.6*, *fontsize=20*)
generate a histogram of the demand data

scenario_random(*startyear=2019*, *lifecycle=20*, *rate=1.02*, *mu=0.01*, *sigma=0.065*)
trend generated from random growth rate increments

class opentisim.container_mixins.**hastriggers_properties_mixin**(*triggers=[]*, **args*, ***kwargs*)

Bases: object

Something has InvestmentTriggers

triggers: list with revenues to be applied from investment year

class opentisim.container_mixins.**history_properties_mixin**(*year_purchase=[]*, *year_online=[]*, **args*, ***kwargs*)

Bases: object

Something that has a purchase history

purchase_date: year in which the decision was made to add another element online_date: year by which the elements starts to perform

class opentisim.container_mixins.**identifiable_properties_mixin**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: object

Something that has a name and id

name: a name id: a unique id generated with uuid

class opentisim.container_mixins.**indirect_costs_mixin**(*preliminaries*, *engineering*, *miscellaneous*, *electrical_works_fuel_terminal*, *electrical_works_power_terminal*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**labour_properties_mixin**(*international_salary*, *international_staff*, *local_salary*, *local_staff*, *operational_salary*, *shift_length*, *annual_shifts*, *daily_shifts*, *blue_collar_salary*, *white_collar_salary*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**laden_stack_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *mobilisation*, *maintenance_perc*, *width*, *height*, *length*, *capacity*, *gross_tgs*, *area_factor*, *pavement*, *drainage*, *household*, *digout_margin*, *reefer_factor*, *consumption*, *reefer_rack*, *reefers_present*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**land_price_mixin**(*price*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**oog_stack_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *mobilisation*, *maintenance_perc*, *width*, *height*, *length*, *capacity*, *gross_tgs*, *area_factor*, *pavement*, *drainage*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**quay_wall_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *mobilisation_min*, *mobilisation_perc*, *maintenance_perc*, *insurance_perc*, *berthing_gap*, *freeboard*, *Gijt_constant*, *Gijt_coefficient*, *max_sinkage*, *wave_motion*, *safety_margin*, *apron_width*, *apron_pavement*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**stack_equipment_properties_mixin**(*type*, *ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation*, *maintenance_perc*, *insurance_perc*, *crew*, *salary*, *required*, *fuel_consumption*, *power_consumption*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**transport_properties_mixin**(*type*, *ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation*, *maintenance_perc*, *insurance_perc*, *crew*, *salary*, *utilisation*, *fuel_consumption*, *productivity*, *required*, *non_essential_moves*, **args*, ***kwargs*)

Bases: object

class opentisim.container_mixins.**vessel_properties_mixin**(*type*, *delivery_time*, *call_size*, *LOA*, *draught*, *beam*, *max_cranes*, *all_turn_time*, *mooring_time*, *demurrage_rate*, *transport_costs*, *all_in_transport_costs*, **args*, ***kwargs*)

Bases: object

2.8 opentisim.container_objects module

Main generic object classes:

1. Quay_wall
2. Berth
3. Cyclic_Unloader
   - STS crane
4. Horizontal transport
   - Tractor trailer
5. Commodity
   - TEU
6. Containers
   - Laden
   - Reefer
   - Empty
   - OOG
7. Laden and reefer stack
   - RTG stack
   - RMG stack
   - SC stack
   - RS stack
8. Stack equipment
   - RTG
   - RMG
   - SC
   - RS
9. Empty stack
10. OOG stack

- 11. Gates
- 12. Empty handler
- 13. Vessel
- 14. Labour
- 15. Energy
- 16. General
- 17. Indirect Costs

class opentisim.container_objects.**Berth**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.berth_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hasrevenue_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*

class opentisim.container_objects.**Commodity**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.commodity_properties_mixin*, *opentisim.container_mixins.hasscenario_properties_mixin*

class opentisim.container_objects.**Container**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.container_properties_mixin*

class opentisim.container_objects.**Cyclic_Unloader**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.cyclic_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hasrevenue_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*

class opentisim.container_objects.**Empty_Handler**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.empty_handler_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*

class opentisim.container_objects.**Empty_Stack**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.empty_stack_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*, *opentisim.container_mixins.hasland_properties_mixin*

class opentisim.container_objects.**Energy**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.energy_properties_mixin*

class opentisim.container_objects.**Gate**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.gate_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*, *opentisim.container_mixins.hasland_properties_mixin*

class opentisim.container_objects.**General_Services**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.hasland_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.general_services_mixin*, *opentisim.container_mixins.history_properties_mixin*

class opentisim.container_objects.**Horizontal_Transport**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.transport_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*

opentisim.container_objects.**Indirect_Costs**
alias of opentisim.container_objects.Indirect Costs

class opentisim.container_objects.**Labour**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.labour_properties_mixin*

class opentisim.container_objects.**Laden_Stack**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.laden_stack_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*, *opentisim.container_mixins.hasland_properties_mixin*

opentisim.container_objects.**Land_Price**
alias of opentisim.container_objects.Land Price

class opentisim.container_objects.**OOG_Stack**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.oog_stack_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*, *opentisim.container_mixins.hasland_properties_mixin*

class opentisim.container_objects.**Quay_wall**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.quay_wall_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hasrevenue_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*, *opentisim.container_mixins.hasland_properties_mixin*

class opentisim.container_objects.**Stack_Equipment**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.history_properties_mixin*, *opentisim.container_mixins.stack_equipment_properties_mixin*, *opentisim.container_mixins.hascapex_properties_mixin*, *opentisim.container_mixins.hasopex_properties_mixin*, *opentisim.container_mixins.hastriggers_properties_mixin*

class opentisim.container_objects.**Vessel**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.container_mixins.identifiable_properties_mixin*, *opentisim.container_mixins.vessel_properties_mixin*

2.9 opentisim.container_system module

class opentisim.container_system.**System**(*terminal_name='Terminal'*, *startyear=2019*, *lifecycle=20*, *operational_hours=7500*, *debug=False*, *elements=[]*, *crane_type_defaults={'consumption': 8*, *'crane_type': 'STS crane'*, *'crew': 5.5*, *'delivery_time': 1*, *'eff_fact': 0.75*, *'hourly_cycles': 25*, *'insurance_perc': 0.01*, *'lifespan': 40*, *'lifting_capacity': 2.13*, *'maintenance_perc': 0.02*, *'mobilisation_perc': 0.15*, *'name': 'STS_crane'*, *'ownership': 'Terminal operator'*, *'unit_rate': 10000000}*, *stack_equipment='rs'*, *laden_stack='rs'*, *allowable_waiting_service_time_ratio_berth=0.1*, *allowable_berth_occupancy=0.6*, *laden_perc=0.8*, *reefer_perc=0.1*, *empty_perc=0.05*, *oog_perc=0.05*, *transhipment_ratio=0.69*, *energy_price=0.17*, *fuel_price=1*, *land_price=0*)

Bases: `object`

This class implements the 'complete supply chain' concept (Van Koningsveld et al, 2020) for container terminals.

The module allows variation of the type of quay crane used and the type of quay crane used and the type of stack equipment used.

Terminal development is governed by the following triggers: - the allowable waiting time as a factor of service time at the berth, and - the distribution ratios (adding up to 1) for:

- ladens
- empties

- reefers
- out of gauges
- the transhipment ratio

berth_invest (*year*, *fully_cellular*, *panamax*, *panamax_max*, *post_panamax_I*, *post_panamax_II*, *new_panamax*, *VLCS*, *ULCS*)

Given the overall objectives for the terminal apply the following decision recipe (Van Koningsveld and Mulder, 2004) for the berth investments.

Decision recipe Berth: QSC: berth_occupancy & allowable_waiting_service_time_ratio Benchmarking procedure: there is a problem if the estimated berth_occupancy triggers a waiting time over service time ratio that is larger than the allowed waiting time over service time ratio

- allowable_waiting_service_time_ratio = .10 # 10% (see PIANC (2014, 2014b))
- **a berth needs:**
  - a quay
  - cranes (min:1 and max: max_cranes)
- **berth occupancy depends on:**
  - total_calls, total_vol and time needed for mooring, unmooring
  - total_service_capacity as delivered by the cranes
- berth occupancy in combination with nr of berths is used to lookup the waiting over service time ratio

Intervention procedure: invest enough to make the planned waiting service time ratio < allowable waiting service time ratio - adding berths, quays and cranes decreases berth_occupancy_rate (and increases the number of servers)

which will yield a smaller waiting time over service time ratio

box_moves (*year*)

Calculate the box moves as input for the power and fuel consumption

calculate_berth_occupancy (*year*, *fully_cellular_calls*, *panamax_calls*, *panamax_max_calls*, *post_panamax_I_calls*, *post_panamax_II_calls*, *new_panamax_calls*, *VLCS_calls*, *ULCS_calls*)

- Find all cranes and sum their effective_capacity to get service_capacity
- Divide callsize_per_vessel by service_capacity and add mooring time to get total time at berth
- Occupancy is total_time_at_berth divided by operational hours

calculate_demurrage_cost (*year*)

Find the demurrage cost per type of vessel and sum all demurrage cost

calculate_energy_cost (*year*)

todo voeg energy toe voor nieuwe elementen

calculate_fuel_cost (*year*)

Fuel cost

calculate_gate_minutes (*year*)

- Find all gates and sum their effective_capacity to get service_capacity
- Calculate average entry and exit time to get total time at gate

- Occupancy is total_minutes_at_gate per hour divided by 1 hour

calculate_general_labour_cost(*year*)
General labour

calculate_indirect_costs()
Indirect costs are a function of overall CAPEX.

calculate_land_use(*year*)
Calculate total land use by summing all land_use values of the physical terminal elements

calculate_throughput(*year*)
Find throughput (minimum of crane capacity and demand)

calculate_vessel_calls(*year*)
Calculate volumes to be transported and the number of vessel calls (both per vessel type and in total)

check_crane_slot_available()

crane_invest(*year*)
Given the overall objectives for the terminal apply the following decision recipe (Van Koningsveld and Mulder, 2004) for the crane investments.

Decision recipe Crane: QSC: planned waiting over service time ratio Benchmarking procedure (triggered in self.berth_invest): there is a problem when the planned planned waiting over service time ratio is larger than the max allowable waiting over service time ratio Intervention procedure: invest until planned waiting over service time ratio is below the max allowable waiting over service time ratio

empty_handler_invest(*year*)
current strategy is to add empty hanlders as soon as a service trigger is achieved - find out how many empty handlers are online - find out how many empty handlers areplanned - find out how many empty handlers are needed - add empty handlers until service_trigger is no longer exceeded

empty_stack_capacity(*year*)
Calculate the stack capacity for empty containers

empty_stack_invest(*year*)
current strategy is to add stacks as soon as trigger is achieved - find out how much stack capacity is online - find out how much stack capacity is planned - find out how much stack capacity is needed - add stack capacity until service_trigger is no longer exceeded

gate_invest(*year*)
current strategy is to add gates as soon as trigger is achieved - find out how much gate capacity is online - find out how much gate capacity is planned - find out how much gate capacity is needed - add gate capacity until service_trigger is no longer exceeded

general_services_invest(*year*)

horizontal_transport_invest(*year*)
current strategy is to add horizontal transport (tractors) as soon as a service trigger is achieved - find out how many cranes are online and planned - find out how many tractors trailers are online and planned (each STS needs a pre-set number of tractors trailers) - add tractor trailers until the required amount (given by the cranes) is achieved

laden_reefer_stack_capacity(*year*)
Calculate the stack capacity for laden and reefer containers

laden_reefer_stack_invest(*year*)

current strategy is to add stacks as soon as trigger is achieved

- find out how much stack capacity is planned

- find out how much stack capacity is required
- add stack capacity until service_trigger is no longer exceeded

The laden stack has a number of positions for laden containers and a number of positions for reefer containers

laden_stack_area_plot (*width=0.25*, *alpha=0.6*)
Gather data from laden stack area and plot it against demand

land_use_plot (*width=0.25*, *alpha=0.6*, *fontsize=20*)
Gather data from Terminal and plot which elements come online when

oog_stack_capacity (*year*)
Calculate the stack capacity for OOG containers

oog_stack_invest (*year*)
Current strategy is to add stacks as soon as trigger is achieved - find out how much stack capacity is planned - find out how much stack capacity is needed - add stack capacity until service_trigger is no longer exceeded

opex_plot (*cash_flows*)
Gather data from Terminal elements and combine into a cash flow plot

quay_invest (*year*, *length*, *depth*)
Given the overall objectives for the terminal apply the following decision recipe (Van Koningsveld and Mulder, 2004) for the quay investments.

Decision recipe Quay: QSC: quay_per_berth Benchmarking procedure (triggered in self.berth_invest): there is a problem when

> the number of berths > the number of quays, but also while the planned waiting over service time ratio is too large

Intervention procedure: invest enough to make sure that each quay has a berth and the planned waiting over service time ratio is below the max allowable waiting over service time ratio

- adding quay will increase quay_per_berth
- quay_wall.length must be long enough to accommodate largest expected vessel
- quay_wall.depth must be deep enough to accommodate largest expected vessel
- quay_wall.freeboard must be high enough to accommodate largest expected vessel

simulate ()
The 'simulate' method implements the terminal investment strategy for this terminal class.

This method automatically generates investment decisions, parametrically derived from overall demand trends and a number of investment triggers.

Generic approaches based on: - Quist, P. and Wijdeven, B., 2014. Ports & Terminals Hand-out. Chapter 7 Container terminals. CIE4330/CIE5306 - PIANC. 2014. Master plans for the development of existing ports. MarCom - Report 158, PIANC - PIANC. 2014b. Design principles for small and medium marine containter terminals. MarCom - Report 135, PIANC - Van Koningsveld, M. (Ed.), Verheij, H., Taneja, P. and De Vriend, H.J. (2020). Ports and Waterways.

Navigating the changing world. TU Delft, Delft, The Netherlands.

- Van Koningsveld, M. and J. P. M. Mulder. 2004. Sustainable Coastal Policy Developments in the Netherlands. A Systematic Approach Revealed. Journal of Coastal Research 20(2), pp. 375-385

Specific application based on (modifications have been applied where deemed an improvement): - Koster, P.H.F., 2019. Concept level container terminal design. Investigating the consequences of accelerating

the concept design phase by modelling the automatable tasks. Master's thesis. Delft University of Technology, Netherlands. URL: http://resolver.tudelft.nl/uuid:131133bf-9021-4d67-afcb-233bd8302ce0.

- Stam, H.W.B., 2020. Offshore-Onshore Port Systems. A framework for the financial evaluation of offshore container terminals. Master's thesis. Delft University of Technology, Netherlands.

The simulate method applies frame of reference style decisions while stepping through each year of the terminal lifecycle and check if investment is needed (in light of strategic objective, operational objective, QSC, decision recipe, intervention method):

1. for each year estimate the anticipated vessel arrivals based on the expected demand
2. for each year evaluate which investment are needed given the strategic and operational objectives
3. for each year calculate the energy costs (requires insight in realized demands)
4. for each year calculate the demurrage costs (requires insight in realized demands)
5. for each year calculate terminal revenues (requires insight in realized demands)
6. collect all cash flows (capex, opex, [revenues])
7. calculate PV's and aggregate to NPV

stack_equipment_invest (*year*)
current strategy is to add stack equipment as soon as a service trigger is achieved - find out how much stack equipment is online - find out how much stack equipment is planned - find out how much stack equipment is needed - add equipment until service_trigger is no longer exceeded

terminal_capacity_plot (*width=0.25*, *alpha=0.6*)
Gather data from Terminal and plot which elements come online when

terminal_elements_plot (*width=0.08*, *alpha=0.6*, *fontsize=20*, *demand_step=50000*)
Gather data from Terminal and plot which elements come online when

throughput_box (*year*)

- Find the total TEU/year for every throughput type (types: ladens, empties, reefers, oogs)
- Translate the total TEU/year to number of boxes for every throughput type

throughput_characteristics (*year*)

- Find commodity volume
- Find the on terminal modal split (types: ladens, empties, reefers, oogs)
- Return the total TEU/year for every throughput type

2.10 opentisim.hydrogen_defaults module

Defaults for following objects:

- 1. Jetty
- 2. Berth
- 3. Unloader

  - Liquid hydrogen
  - Ammonia
  - MCH
- 4. Pipelines
  - jetty
  - hinterland
- 5. Storage
  - Liquid hydrogen
  - Ammonia
  - MCH
- 6. H2 retrieval
  - Ammonia
  - MCH
- 6. Commodity
  - Liquid hydrogen
  - Ammonia
  - MCH
- 7. Vessel
  - smallhydrogen
  - largehydrogen
  - smallammonia
  - largeammonia
  - Handysize
  - Panamax
  - VLCC
- 8. Labour

Default values are based on Claes 2018; Corbeau 2018; Daas 2018; Juha 2018; Kranendonk 2018; Schutz 2018; Schuurmans 2018 and Verstegen 2018

opentisim.hydrogen_defaults.**commodity_MCH_data = {'handling_fee': 1000, 'handysize_perc':**
Liquid hydrogen:

opentisim.hydrogen_defaults.**h2retrieval_lh2_data = {'capacity': 171, 'consumption': 600,**
Ammonia

opentisim.hydrogen_defaults.**h2retrieval_nh3_data = {'capacity': 55, 'consumption': 5889,**
MCH

opentisim.hydrogen_defaults.**hinterland_pipeline_data = {'capacity': 4000, 'consumption_co**
Liquid hydrogen

opentisim.hydrogen_defaults.**largeammonia_data = {'LOA': 230, 'all_turn_time': 24, 'beam':**
MCH:

opentisim.hydrogen_defaults.**largehydrogen_data = {'LOA': 300, 'all_turn_time': 30, 'beam'**
Ammonia:

opentisim.hydrogen_defaults.**storage_MCH_data = {'capacity': 38500, 'consumption': 10, 'c**
Liquid hydrogen

opentisim.hydrogen_defaults.**storage_lh2_data = {'capacity': 3540, 'consumption': 610, 'c**
Ammonia

opentisim.hydrogen_defaults.**storage_nh3_data = {'capacity': 34130, 'consumption': 100, '**
MCH

2.11 opentisim.hydrogen_mixins module

Basic properties mixins:

- identifiable_properties_mixin
- history_properties_mixin
- hascapex_properties_mixin
- hasopex_properties_mixin
- hasrevenue_properties_mixin
- hastriggers_properties_mixin
- jetty_properties_mixin
- berth_properties_mixin
- cyclic_properties_mixin
- continuous_properties_mixin
- pipeline_properties_mixin
- storage_properties_mixin
- commodity_properties_mixin
- vessel_properties_mixin
- labour_properties_mixin
- hasscenario_properties_mixin

class opentisim.hydrogen_mixins.**berth_properties_mixin**(*crane_type*, *delivery_time*, **args*, ***kwargs*)
Bases: object

class opentisim.hydrogen_mixins.**commodity_properties_mixin**(*type*, *handling_fee*, *smallhydrogen_perc*, *largehydrogen_perc*, *smallammonia_perc*, *largeammonia_perc*, *handysize_perc*, *panamax_perc*, *vlcc_perc*, **args*, ***kwargs*)
Bases: object

class opentisim.hydrogen_mixins.**energy_properties_mixin**(*price*, **args*, ***kwargs*)
Bases: object

class opentisim.hydrogen_mixins.**h2retrieval_properties_mixin**(*type*, *ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation_min*, *mobilisation_perc*, *maintenance_perc*, *crew_min*, *crew_for5*, *insurance_perc*, *h2retrieval_type*, *consumption*, *capacity*, **args*, ***kwargs*)

Bases: object

class opentisim.hydrogen_mixins.**hascapex_properties_mixin**(*capex=[]*, **args*, ***kwargs*)

Bases: object

Something has CAPEX

capex: list with cost to be applied from investment year

class opentisim.hydrogen_mixins.**hasopex_properties_mixin**(*labour=[]*, *maintenance=[]*, *energy=[]*, *insurance=[]*, *lease=[]*, *demurrage=[]*, **args*, ***kwargs*)

Bases: object

Something has OPEX

opex: list with cost to be applied from investment year

class opentisim.hydrogen_mixins.**hasrevenue_properties_mixin**(*renevue=[]*, *residual=[]*, **args*, ***kwargs*)

Bases: object

Something has Revenue

revenue: list with revenues to be applied from investment year

class opentisim.hydrogen_mixins.**hasscenario_properties_mixin**(*historic_data=[]*, *scenario_data=[]*, **args*, ***kwargs*)

Bases: object

Something has a scenario

historic_data: observed demand scenario_data: generated estimates of future demand

plot_demand(*width=0.1*, *alpha=0.6*, *fontsize=20*)
generate a histogram of the demand data

scenario_random(*startyear=2019*, *lifecycle=20*, *rate=1.02*, *mu=0.01*, *sigma=0.065*)
trend generated from random growth rate increments

class opentisim.hydrogen_mixins.**hastriggers_properties_mixin**(*triggers=[]*, **args*, ***kwargs*)

Bases: object

Something has InvestmentTriggers

triggers: list with revenues to be applied from investment year

class opentisim.hydrogen_mixins.**history_properties_mixin**(*year_purchase=[]*, *year_online=[]*, **args*, ***kwargs*)

Bases: object

Something that has a purchase history

purchase_date: year in which the decision was made to add another element online_date: year by which the elements starts to perform

class opentisim.hydrogen_mixins.**identifiable_properties_mixin**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: object

Something that has a name and id

name: a name id: a unique id generated with uuid

class opentisim.hydrogen_mixins.**jetty_properties_mixin**(*ownership*, *delivery_time*, *lifespan*, *mobilisation_min*, *mobilisation_perc*, *maintenance_perc*, *insurance_perc*, *Gijt_constant_jetty*, *jettywidth*, *jettylength*, *mooring_dolphins*, *catwalkwidth*, *catwalklength*, *Catwalk_rate*, **args*, ***kwargs*)

Bases: object

class opentisim.hydrogen_mixins.**labour_properties_mixin**(*international_salary*, *international_staff*, *local_salary*, *local_staff*, *operational_salary*, *shift_length*, *annual_shifts*, **args*, ***kwargs*)

Bases: object

class opentisim.hydrogen_mixins.**pipeline_properties_mixin**(*type*, *length*, *ownership*, *delivery_time*, *lifespan*, *unit_rate_factor*, *mobilisation*, *maintenance_perc*, *insurance_perc*, *consumption_coefficient*, *crew*, *utilisation*, *capacity*, **args*, ***kwargs*)

Bases: object

class opentisim.hydrogen_mixins.**storage_properties_mixin**(*type*, *ownership*, *delivery_time*, *lifespan*, *unit_rate*, *mobilisation_min*, *mobilisation_perc*, *maintenance_perc*, *crew_min*, *crew_for5*, *insurance_perc*, *storage_type*, *consumption*, *capacity*, **args*, ***kwargs*)

Bases: object

class opentisim.hydrogen_mixins.**train_properties_mixin**(*wagon_payload*, *number_of_wagons*, **args*, ***kwargs*)

Bases: object

class opentisim.hydrogen_mixins.**vessel_properties_mixin**(*type*, *call_size*, *LOA*, *draft*, *beam*, *max_cranes*, *all_turn_time*, *pump_capacity*, *mooring_time*, *demurrage_rate*, **args*, ***kwargs*)

Bases: object

2.12 opentisim.hydrogen_objects module

Main generic object classes:

Defaults for following objects: - 1. Jetty - 2. Berth - 3. Unloader

- Liquid hydrogen
- Ammonia
- MCH
- 4. Pipelines
  - jetty
  - hinterland
- 5. Storage
  - Liquid hydrogen
  - Ammonia
  - MCH
- 6. H2 retrieval
  - Ammonia
  - MCH
- 6. Commodity
  - Liquid hydrogen
  - Ammonia

  - MCH
- 7. Vessel
  - smallhydrogen
  - largehydrogen
  - smallammonia
  - largeammonia
  - Handysize
  - Panamax
  - VLCC
- 8. Labour

class opentisim.hydrogen_objects.**Berth**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.history_properties_mixin*, *opentisim.hydrogen_mixins.berth_properties_mixin*, *opentisim.hydrogen_mixins.hascapex_properties_mixin*, *opentisim.hydrogen_mixins.hasopex_properties_mixin*, *opentisim.hydrogen_mixins.hasrevenue_properties_mixin*, *opentisim.hydrogen_mixins.hastriggers_properties_mixin*

class opentisim.hydrogen_objects.**Commodity**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.commodity_properties_mixin*, *opentisim.hydrogen_mixins.hasscenario_properties_mixin*

class opentisim.hydrogen_objects.**Energy**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.energy_properties_mixin*

class opentisim.hydrogen_objects.**H2retrieval**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.history_properties_mixin*, *opentisim.hydrogen_mixins.h2retrieval_properties_mixin*, *opentisim.hydrogen_mixins.hascapex_properties_mixin*, *opentisim.hydrogen_mixins.hasopex_properties_mixin*, *opentisim.hydrogen_mixins.hasrevenue_properties_mixin*, *opentisim.hydrogen_mixins.hastriggers_properties_mixin*

class opentisim.hydrogen_objects.**Jetty**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.jetty_properties_mixin*, *opentisim.hydrogen_mixins.history_properties_mixin*, *opentisim.hydrogen_mixins.hascapex_properties_mixin*, *opentisim.hydrogen_mixins.hasopex_properties_mixin*, *opentisim.hydrogen_mixins.hasrevenue_properties_mixin*, *opentisim.hydrogen_mixins.hastriggers_properties_mixin*

class opentisim.hydrogen_objects.**Labour**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.labour_properties_mixin*

class opentisim.hydrogen_objects.**Pipeline_Hinter**(*name=[]*, *id=None*, **args*, ***kwargs*)
Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*,

opentisim.hydrogen_mixins.history_properties_mixin, *opentisim.hydrogen_mixins.pipeline_properties_mixin*, *opentisim.hydrogen_mixins.hascapex_properties_mixin*, *opentisim.hydrogen_mixins.hasopex_properties_mixin*, *opentisim.hydrogen_mixins.hasrevenue_properties_mixin*, *opentisim.hydrogen_mixins.hastriggers_properties_mixin*

class opentisim.hydrogen_objects.**Pipeline_Jetty**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.history_properties_mixin*, *opentisim.hydrogen_mixins.pipeline_properties_mixin*, *opentisim.hydrogen_mixins.hascapex_properties_mixin*, *opentisim.hydrogen_mixins.hasopex_properties_mixin*, *opentisim.hydrogen_mixins.hasrevenue_properties_mixin*, *opentisim.hydrogen_mixins.hastriggers_properties_mixin*

class opentisim.hydrogen_objects.**Storage**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.history_properties_mixin*, *opentisim.hydrogen_mixins.storage_properties_mixin*, *opentisim.hydrogen_mixins.hascapex_properties_mixin*, *opentisim.hydrogen_mixins.hasopex_properties_mixin*, *opentisim.hydrogen_mixins.hasrevenue_properties_mixin*, *opentisim.hydrogen_mixins.hastriggers_properties_mixin*

class opentisim.hydrogen_objects.**Train**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.train_properties_mixin*

class opentisim.hydrogen_objects.**Vessel**(*name=[]*, *id=None*, **args*, ***kwargs*)

Bases: *opentisim.hydrogen_mixins.identifiable_properties_mixin*, *opentisim.hydrogen_mixins.vessel_properties_mixin*

2.13 opentisim.hydrogen_system module

class opentisim.hydrogen_system.**System**(*startyear=2019*, *lifecycle=20*, *operational_hours=5840*, *debug=False*, *elements=[]*, *commodity_type_defaults={'handling_fee': 150, 'handysize_perc': 0, 'historic_data': year volume 0 2014 1000000 1 2015 1100000 2 2016 1250000 3 2017 1400000 4 2018 1500000, 'largeammonia_perc': 60, 'largehydrogen_perc': 0, 'name': 'Ammonia', 'panamax_perc': 0, 'smallammonia_perc': 40, 'smallhydrogen_perc': 0, 'type': 'Ammonia', 'vlcc_perc': 0}*, *storage_type_defaults={'capacity': 34130, 'consumption': 100, 'crew_for5': 1, 'crew_min': 3, 'delivery_time': 1, 'insurance_perc': 0.01, 'lifespan': 30, 'maintenance_perc': 0.01, 'mobilisation_min': 200000, 'mobilisation_perc': 0.003, 'name': 'ATank_01', 'ownership': 'Terminal operator', 'storage_type': 'tank', 'type': 'AmmoniaTank', 'unit_rate': 60000000}*, *h2retrieval_type_defaults={'capacity': 55, 'consumption': 5889, 'crew_for5': 1, 'crew_min': 3, 'delivery_time': 2, 'h2retrieval_type': 'tank', 'insurance_perc': 0.01, 'lifespan': 20, 'maintenance_perc': 0.015, 'mobilisation_min': 200000, 'mobilisation_perc': 0.003, 'name': 'H2retrieval_NH3_01', 'ownership': 'Terminal operator', 'type': 'AmmoniaTank', 'unit_rate': 100000000}*, *allowable_berth_occupancy=0.5*, *allowable_dwelltime=0.038356164383561646*, *h2retrieval_trigger=1*)

Bases: object

This class implements the 'complete supply chain' concept (Van Koningsveld et al, 2020) for hydrogen terminals. The module allows variation of the commodity type, the storage type and the h2retrieval type. Terminal development is governed by three triggers: the allowable berth occupancy, the allowable dwell time and an h2retrieval trigger.

H2retrieval_capacity_plot(*width=0.25*, *alpha=0.6*)
Gather data from Terminal and plot which elements come online when

Jetty_capacity_plot(*width=0.3*, *alpha=0.6*)
Gather data from Terminal and plot which elements come online when

Pipeline1_capacity_plot(*width=0.2*, *alpha=0.6*)
Gather data from Terminal and plot which elements come online when

Pipeline2_capacity_plot(*width=0.2*, *alpha=0.6*)
Gather data from Terminal and plot which elements come online when

Storage_capacity_plot(*width=0.25*, *alpha=0.6*)
Gather data from Terminal and plot which elements come online when

berth_invest(*year*)
Given the overall objectives of the terminal

Decision recipe Berth: QSC: berth_occupancy Problem evaluation: there is a problem if the

berth_occupancy > allowable_berth_occupancy

- allowable_berth_occupancy = .50 # 50%
- **a berth needs:**
  - a jetty
- **berth occupancy depends on:**
  - total_calls and total_vol
  - total_service_capacity as delivered by the vessels

Investment decisions: invest enough to make the berth_occupancy < allowable_berth_occupancy

- adding jettys decreases berth_occupancy_rate

calculate_berth_occupancy (*year*, *smallhydrogen_calls*, *largehydrogen_calls*, *smallammonia_calls*, *largeammonia_calls*, *handysize_calls*, *panamax_calls*, *vlcc_calls*, *smallhydrogen_calls_planned*, *largehydrogen_calls_planned*, *smallammonia_calls_planned*, *largeammonia_calls_planned*, *handysize_calls_planned*, *panamax_calls_planned*, *vlcc_calls_planned*)

- Find all cranes and sum their effective_capacity to get service_capacity
- Divide callsize_per_vessel by service_capacity and add mooring time to get total time at berth
- Occupancy is total_time_at_berth divided by operational hours

calculate_demurrage_cost (*year*)
Find the demurrage cost per type of vessel and sum all demurrage cost

calculate_energy_cost (*year*)
The energy cost of all different element are calculated. 1. At first find the consumption, capacity and working hours per element 2. Find the total energy price to multiply the consumption with the energy price

calculate_h2retrieval_occupancy (*year*, *hydrogen_defaults_h2retrieval_data*)

- Divide the throughput by the service rate to get the total hours in a year
- Occupancy is total_time_at_h2retrieval divided by operational hours

calculate_revenue (*year*, *hydrogen_defaults_commodity_data*)

1. calculate the value of the total throughput in year (throughput * handling fee)

calculate_vessel_calls (*year=2019*)
Calculate volumes to be transported and the number of vessel calls (both per vessel type and in total)

check_throughput_available (*year*)

demand_terminal_plot (*width=0.1*, *alpha=0.6*)

h2retrieval_invest (*year*, *hydrogen_defaults_h2retrieval_data*)
current strategy is to add h2 retrieval as long as target h2 retrieval is not yet achieved - find out how much h2 retrieval is online - find out how much h2 retrieval is planned - find out how much h2 retrieval is needed - add h2 retrieval until target is reached

jetty_invest (*year*, *nrofdolphins*)
*** Decision recipe jetty: *** QSC: jetty_per_berth problem evaluation: there is a problem if the jetty_per_berth < 1 investment decisions: invest enough to make the jetty_per_berth = 1

- adding jetty will increase jetty_per_berth

- jetty_wall.length must be long enough to accommodate largest expected vessel
- jetty_wall.depth must be deep enough to accommodate largest expected vessel
- jetty_wall.freeboard must be high enough to accommodate largest expected vessel

pipeline_hinter_invest(*year*)
current strategy is to add pipeline as soon as a service trigger is achieved - find out how much service capacity is online - find out how much service capacity is planned - find out how much service capacity is needed - add service capacity until service_trigger is no longer exceeded

pipeline_jetty_invest(*year*)
current strategy is to add pipeline as soon as a service trigger is achieved - find out how much service capacity is online - find out how much service capacity is planned - find out how much service capacity is needed - add service capacity until service_trigger is no longer exceeded

plant_occupancy_plot(*width=0.3*, *alpha=0.6*)
Gather data from Terminal and plot which elements come online when

simulate()
The 'simulate' method implements the terminal investment strategy for this terminal class.

This method automatically generates investment decisions, parametrically derived from overall demand trends and a number of investment triggers.

Generic approaches based on: - Van Koningsveld, M. (Ed.), Verheij, H., Taneja, P. and De Vriend, H.J. (2020). Ports and Waterways.

Navigating the changing world. TU Delft, Delft, The Netherlands.

- Van Koningsveld, M. and J. P. M. Mulder. 2004. Sustainable Coastal Policy Developments in the Netherlands. A Systematic Approach Revealed. Journal of Coastal Research 20(2), pp. 375-385

Specific application based on: - Ijzermans, W., 2019. Terminal design optimization. Adaptive agribulk terminal planning

in light of an uncertain future. Master's thesis. Delft University of Technology, Netherlands. URL: http://resolver.tudelft.nl/uuid:7ad9be30-7d0a-4ece-a7dc-eb861ae5df24.

The simulate method applies frame of reference style decisions while stepping through each year of the terminal lifecycle and check if investment is needed (in light of strategic objective, operational objective, QSC, decision recipe, intervention method):

1. for each year estimate the anticipated vessel arrivals based on the expected demand
2. for each year evaluate which investment are needed given the strategic and operational objectives
3. for each year calculate the energy costs (requires insight in realized demands)
4. for each year calculate the demurrage costs (requires insight in realized demands)
5. for each year calculate terminal revenues (requires insight in realized demands)
6. for each year calculate the throughput (requires insight in realized demands) 6. for each year calculate terminal throughputequires insight in realized demands)
7. collect all cash flows (capex, opex, revenues)
8. calculate PV's and aggregate to NPV

storage_invest(*year*, *hydrogen_defaults_storage_data*)
current strategy is to add storage as long as target storage is not yet achieved - find out how much storage is online - find out how much storage is planned - find out how much storage is needed - add storage until target is reached

terminal_elements_plot (*width=0.1*, *alpha=0.6*, *fontsize=20*)
Gather data from Terminal and plot which elements come online when

terminal_occupancy_plot (*width=0.3*, *alpha=0.6*)
Gather data from Terminal and plot which elements come online when

throughput_elements (*year*)

- Find which elements are important and needs to be included
- Find from each element the online capacity
- Find where the lowest value is present, in the capacity or in the demand

2.14 Module contents

Top-level package for OpenTISim.

CHAPTER 3

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

3.1 Types of Contributions

3.1.1 Report Bugs

Report bugs at https://github.com/TUDelft-CITG/OpenTISim/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

3.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

3.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

3.1.4 Write Documentation

OpenClim could always use more documentation, whether as part of the official OpenCLSim docs, in docstrings, or even on the web in blog posts, articles, and such.

3.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/TUDelft-CITG/OpenTISim/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

3.2 Get Started!

Ready to contribute? Here's how to set up *OpenTISim* for local development.

1. Fork the *OpenTISim* repository on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/OpenTISim.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv opentisim
$ cd opentisim/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 opentisim tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. The style of OpenTISim is according to Black. Format your code using Black witht the following lines of code:

```
$ black opentisim
$ black tests
```

   You can install black using pip.

7. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

8. Submit a pull request through the GitHub website.

3.3 Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.4, 3.5 and 3.6, and for PyPy. Check CircleCI and make sure that the tests pass for all supported Python versions.

3.4 Tips

To run a subset of tests:

```
$ py.test tests.test_opentisim
```

To make the documentation pages $ make docs # for linux/osx

For windows $ del docsopentisim.rst $ del docsmodules.rst $ sphinx-apidoc -o docs/ opentisim $ cd docs $ make html $ start explorer _buildhtmlindex.html

3.5 Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

CHAPTER 4

Credits

4.1 Development Lead

- Mark van Koningsveld

4.2 Contributors

Various MSc projects

- Wijnand Ijzermans, 2019. **Terminal design optimization. Adaptive agribulk terminal planning in light of an uncertain future.** MSc thesis. Delft University of Technology, Civil Engineering and Geosciences, Hydraulic Engineering - Ports and Waterways. Delft, the Netherlands.
- Stephanie Lanphen, 2019. **Hydrogen import terminal. Elaborating the supply chains of a hydrogen import terminal, and its corresponding investment decisions.** MSc thesis. Delft University of Technology, Civil Engineering and Geosciences, Hydraulic Engineering - Ports and Waterways. Delft, the Netherlands.
- Piebe Koster, 2019. **Optimisation of concept level container terminal design. Accelerate the generation and visualisation of the terminal design to reduce the probability of a sub-optimal solution.** MSc thesis. Delft University of Technology, Civil Engineering and Geosciences, Hydraulic Engineering - Ports and Waterways. Delft, the Netherlands.

Ongoing MSc work

- Hugo Stam, 2019. **Logistical optimisation of offshore onshore port systems from a economic perspective.** MSc thesis. Delft University of Technology, Civil Engineering and Geosciences, Environmental Fluid Mechanics. Delft, the Netherlands.

CHAPTER 5

History

5.1 v0.6.2 (2020-03-09)

- Updated the container code (minor bugs fixed)

5.2 v0.6.1 (2020-03-09)

- Updated the container code with vessel classes

5.3 v0.6.0 (2020-02-14)

- Updated the container code and example

5.4 v0.5.0 (2020-01-24)

- Updated the agribulk code and example

5.5 v0.4.0 (2019-07-18)

- Renamed and first release to PyPi

5.6 v0.3.0 (2019-07-10)

- Merged multiple terminal types to the master

5.7 v0.2.0 (2019-04-06)

- Working version of redesigned code

5.8 v0.1.0 (2019-02-18)

- Final version MSc project Wijnand IJzermans

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

o

Index

B

C

D

E

G

H

I

J

L

O

P

Q

S

T

U

V