

---

# **transport Documentation**

***Release v2.0.0-alpha-3***

**Milan Lovric**

**May 18, 2020**



---

## Contents

---

<b>1</b>	<b>Contents</b>	<b>3</b>
1.1	Documentation . . . . .	3
1.2	‘transport’ . . . . .	11
1.3	Authors . . . . .	240
1.4	License . . . . .	240
<b>2</b>	<b>Indices and tables</b>	<b>241</b>
	<b>Index</b>	<b>243</b>



**NISMOD v2 Transport Model** is a national-scale (*Great Britain*) transport model developed to support policy making regarding future infrastructure.



## 1.1 Documentation

**NISMOD v2 Transport Model** is a national-scale (*Great Britain*) transport model developed to support policy making regarding future infrastructure. It forecasts the impact of various endogenous and exogenous factors on transport demand and capacity utilisation, following an elasticity-based simulation methodology. The model consists of three submodels covering the following modes of transport: **road** (passenger and freight vehicle flows), **rail** (total station usage), and **air** (domestic and international passenger movements).

### 1.1.1 1. Key Features

#### 1.1 Road Model

- NISMOD v2 Transport Model predicts vehicle demand (inter-zonal flows) for passenger and freight vehicles, and stochastically simulates road traffic on all major UK roads including A-roads and motorways.
- It is currently the only national-scale road traffic model capable of routing-based network assignment and provisioning a national-scale origin-destination matrix (on TEMPRo & LAD spatial zoning levels), while achieving a respectable match with AADF traffic counts, total vehicle kilometres, expected number of car trips, and the observed trip length distribution from the National Travel Survey.
- The freight model has been modelled after the DfT's 2006 Base-Year Freight Matrices model, which includes traffic flows for freight vehicles (vans, rigid HGVs, and articulated HGVs) between local authority districts (LADs), sea ports, selected airports, and major distribution centres. The accuracy of the freight model is mostly limited by the spatial zoning system (LAD).
- The demand prediction model is elasticity-based and it can predict future vehicle flows from exogenous (scenario-based) changes in population and GVA, and endogenously calculated changes in inter-zonal travel time and travel cost (but also dependent on exogenous interventions such as new road development and congestion charging policies).

- Congested travel times on individual road links have been modelled separately for each hour of the day, using the speed-flow curves estimated on English roads (DfT's 2005 FORGE model), the overcapacity formula from WebTAG, and the passenger car unit (PCU) concept to capture different vehicle sizes.
- The number of lanes on each road segment has been estimated by map-matching AADF count point locations to the OpenRoads major road network. This has allowed a distinction between single and dual carriageway A-roads, which are then assumed to have 1 and 2 lanes per direction, respectively.
- The network assignment exists in two version and it has been implemented using state-of-the-art routing algorithms. The routing version uses a heuristic search algorithm A\* to find the fastest path between two locations using congested link travel times, while the route-choice version uses an advanced discrete-choice model (path-size logit) to choose the optimal path based on distance, travel time, travel cost (fuel and road tolls), and the number of intersections.
- The route-choice version of the network assignment uses a route set pre-generated on the IRIDIS cluster of the *University of Southampton*. This pre-generated route set consists of more than 90 million different route options which enables the national-scale assignment to run within minutes, despite each individual vehicle trip being simulated separately (including time of day choice, engine type choice, route choice).
- The model can also incorporate scenarios for changes in vehicle fuel efficiency and changes in market shares of different engine types, including internal combustion engines on petrol, diesel, LPG, hydrogen or CNG; hybrid EVs on petrol or diesel; plug-in hybrid EVs on petrol or diesel; fuel cell EVs on hydrogen, and battery EV. This can be used to test policies such as the fossil fuel phase-out.
- Electricity and fuel consumptions are calculated using the four-parameter formula from WebTAG. Behavioural assumptions are made for plug-in hybrid EVs (electricity on urban, fuel on rural road links).
- Interventions such as new road development, road expansion with new lanes, and congestion charging zones can be dynamically implemented in each simulated year.
- The model can output various metrics on the road link level (e.g. road capacity utilisation, peak hour travel times), zonal level (e.g. vehicle kilometres, EV electricity consumption), inter-zonal level (e.g. predicted vehicle flows, average travel times, average travel costs) and national level (e.g. total CO2 emissions, total energy consumptions). The outputs are in csv and shapefile format, allowing them to be visualised with a software of choice.

## 1.2 Rail Model

- NISMOD v2 Transport Model includes a national-scale rail model for predicting future station usage demand.
- It currently uses station usage data for 3054 stations covering National Rail, London Underground, Docklands Light Railway, London Trams (previously Croydon Tramlink), Manchester Metrolink, and Tyne & Wear (Newcastle) Metro.
- Elasticity-based demand model predicts station usage (entry + exit) from changes in exogenous inputs including: population, GVA, rail fare index, generalised journey time (GJT) index and car trip costs.
- Car trip costs can be provided as an input or calculated from the outputs of the NISMOD road model.
- Elasticities of rail fares and GJT vary per elasticity zone (London Travelcard, South-East, PTE, other).
- The model implements a policy intervention for building new rail stations in future years.

## 1.2 Air Model

- NISMOD v2 Transport Model also includes an air model that predicts domestic and international passenger movements.

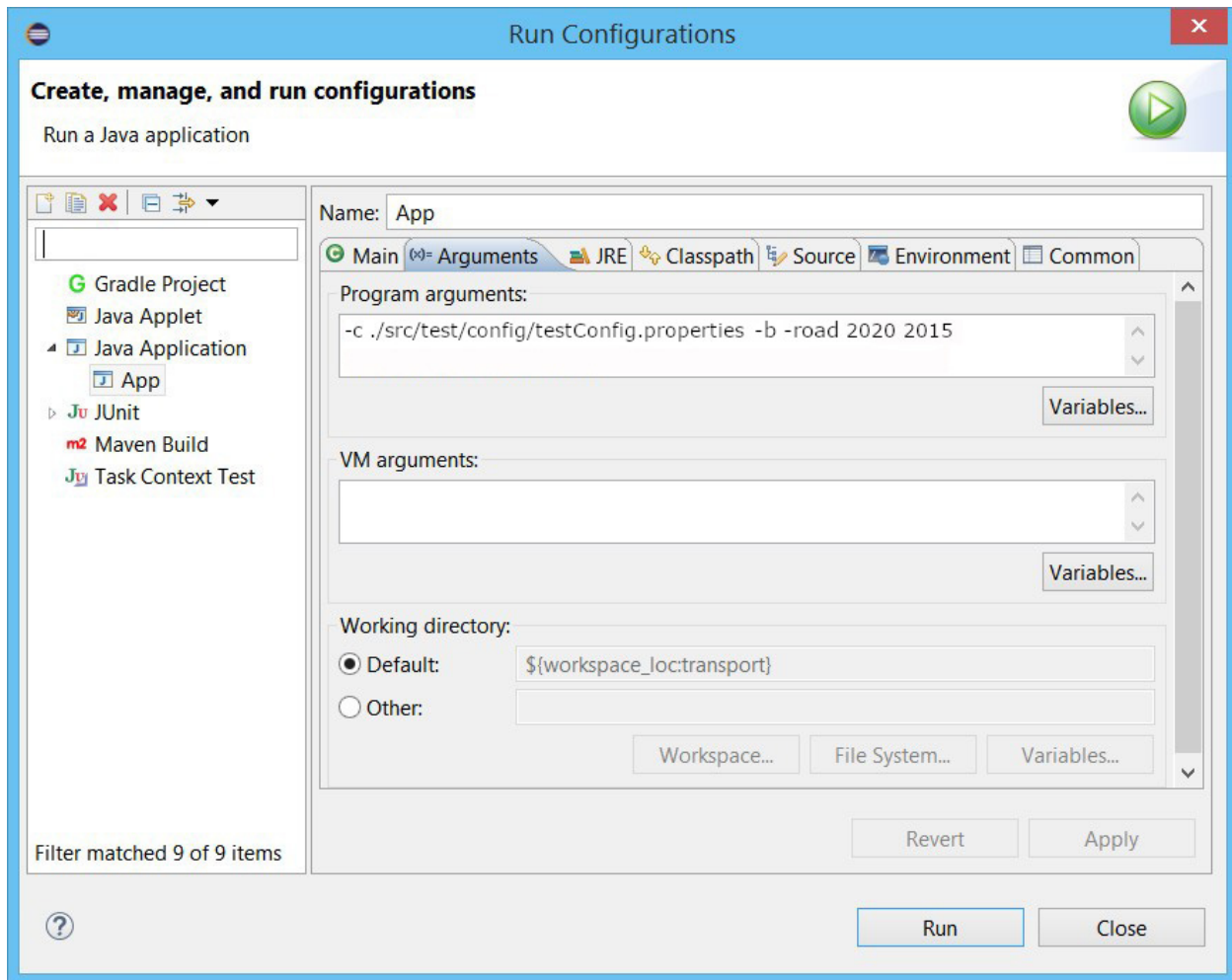


- Air demand data is inter-nodal, i.e. between individual airports (domestic - between two UK airports, and international - between a UK airport and an international airport).
- Base-year (2015) demand data is obtain from the Civil Aviation Authority (CAA), while information about airports is obtained from CAA, NaPTAN, and ourairports.com.
- Demand files use IATA codes (or ICAO where IATA is unavailable) to identify airports and ISO 3166 Alpha-2 codes to identify countries.
- Elasticity-based demand model predicts passenger movements from changes in exogenous inputs including: population, GVA, domestic and international fare incides, and trip rates.

## 1.1.2 2. How to run the model

### 2.1 Using Eclipse IDE

- Install *Java Development Kit* version 8 from: <http://www.oracle.com>.
- Install *Eclipse IDE for Java Developers*: <https://eclipse.org/downloads/>.
- Run Eclipse and choose the workspace folder.
- Import the existing Maven project from the local git folder where the code has been cloned. In Eclipse: *File -> Import -> Maven -> Existing Maven Projects*.
- Wait until all Maven dependencies (specified in the *pom.xml* file) are downloaded. If the *pom.xml* file has been changed, the Maven project should be first updated (*Alt+F5*).
- The classes containing the *main* method can be run as a Java application. The classes containing the methods annotated with *@Test* can be run as *JUnit* tests.
- To run the main model in Eclipse, open the *Run Configuration* for *nismod.transport.App.java* and pass the path to the config file as an argument:



## 2.2 Using Command Prompt (Console)

- Install *Java Development Kit* version 8 from: <http://www.oracle.com>.
- Make sure the Java home environment variable is set for the operating system and pointing to the directory where *Java Development Kit* has been installed.
- Download maven, install it and set the environment variables: <http://maven.apache.org/>.
- To build the project type:

```
mvn clean install
```

- To run the base-year **road** model (2015) type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/
↪config.properties -b
```

- To predict and run a future year (e.g. 2020) using the results of a previously run year (e.g. 2015), for the **road** model type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/
↪config.properties -road 2020 2015
```

- To predict and run a future year (e.g. 2020) using the results of a previously run year (e.g. 2015), for the **rail** model type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/  
↪config.properties -rail 2020 2015
```

- To predict and run a future year (e.g. 2020) using the results of a previously run year (e.g. 2015), for the **air** model type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/  
↪config.properties -air 2020 2015
```

---

**Note:** Rail and air model need not be run for the base-year as 2015 demand data is given as an input.

---

- Options:
  - To increase the max heap size, run with `java -XX:MaxHeapSize=120g ...`
  - To enable debug messages, run with `java -Dlog4j2.debug ...`

## 2.3 Showcase Demo

The model provides an interactive showcase demo with three policy interventions in the case study area of South-East England. The interventions are:

- *Road expansion* - expanding existing road links with additional lanes.
- *Road development* - building new road links between two existing intersections.
- *Congestion charging* - time-based (peak and off-peak) congestion charging in the policy area.

To run the showcase demo type:

```
java -cp target/transport-0.0.1-SNAPSHOT.jar nismod.transport.App -c ./path/to/config.  
↪properties -d
```

What is the impact of traffic policy interventions in South East England?

Click to explore how three policy interventions would influence road capacity utilisation, vehicle demand and travel times on major roads and motorways.

Intervention 1:  
Road Expansion

What happens when we increase road capacity by adding lanes?

Intervention 2:  
Road Development

What happens when we build completely new roads?

Intervention 3:  
Congestion Charging

What happens when we implement a congestion charging zone?

ITRC MISTRAL Transport Model | To learn more, please contact Milan Lovric, University of Southampton (M.Lovric@soton.ac.uk)

Intervention 1:  
Road Expansion

What we asked:  
What happens when we increase road capacity by expanding the road network?

What we found:

- Lower road capacity utilisation on expanded links.
- Slight increase in vehicle flows.
- Slight decrease in travel times.

Try it yourself!

Expand a road link by first selecting two nodes on the "before" map:

Node A  
23

Node B  
22

How many lanes to add?  
1 2 3 4 5

How many directions?  
☐ 1-way ☒ 2-way

RUN

Observe the change in capacity utilisation in the "after" map

Before Policy Intervention

After Policy Intervention

Capacity Utilisation: Very High High Medium Low Very Low

Before Policy Intervention

After Policy Intervention

OD Matrix [trips]

ORIG \ DEST	Southampton	Isle of Wight	Eastleigh	New Forest
Southampton	15000	450	16500	8250
Isle of Wight	600	3000	450	270
Eastleigh	119500	340	16500	2700
New Forest	13680	300	4200	18000

Demand (Total Number of Trips)

119760

Travel Time [min]

ORIG \ DEST	Southampton	Isle of Wight	Eastleigh	New Forest
Southampton	8.30	100.07	14.29	69.04
Isle of Wight	82.02	17.17	92.70	67.17
Eastleigh	14.33	109.42	10.84	81.14
New Forest	34.67	97.35	44.34	59.58

After Policy Intervention

OD Matrix [trips]

ORIG \ DEST	Southampton	Isle of Wight	Eastleigh	New Forest
Southampton	15036	450	16472	8237
Isle of Wight	597	2987	452	267
Eastleigh	119464	358	16440	2711
New Forest	13782	307	4222	18064

Demand (Total Number of Trips)

119866

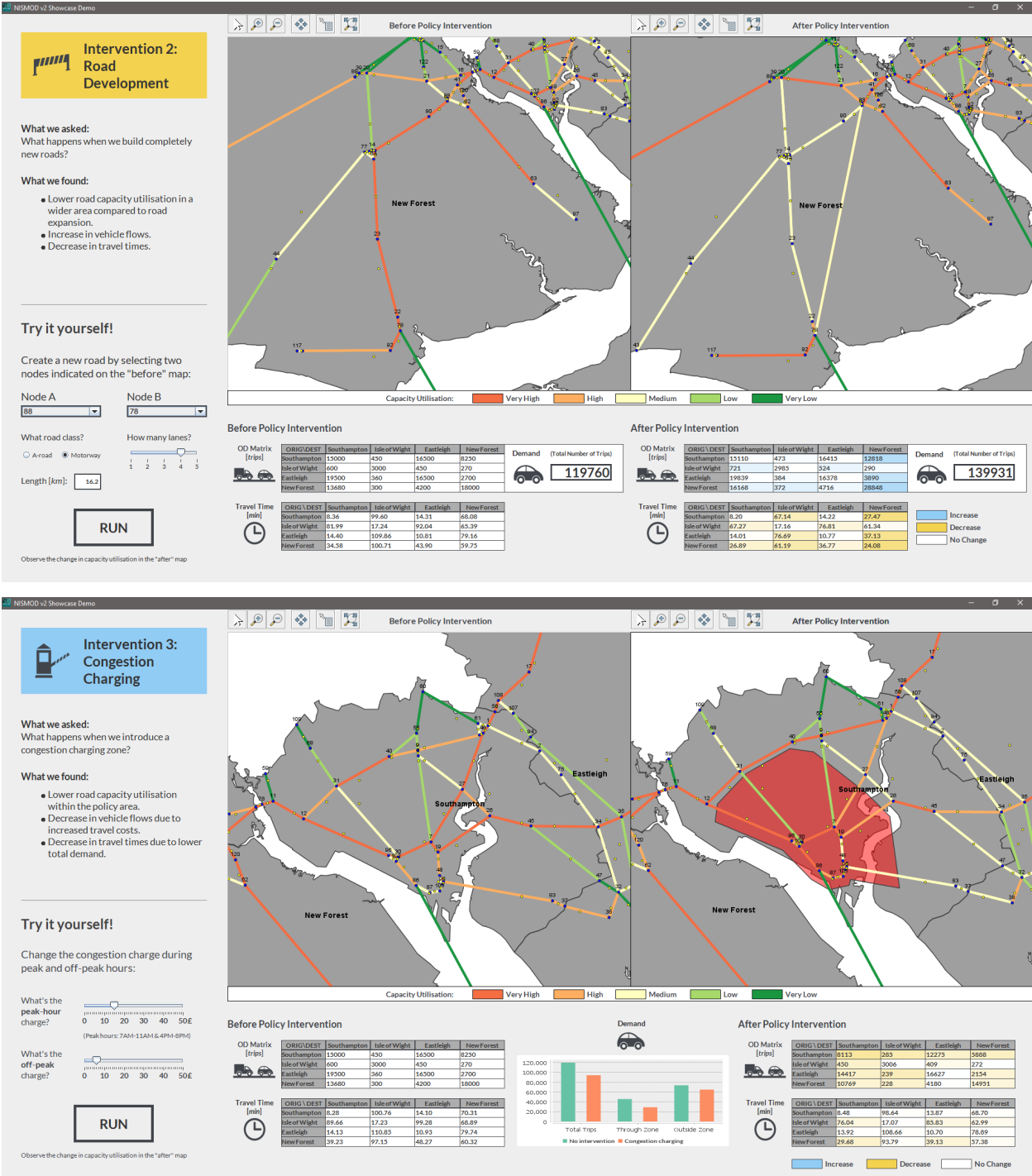
Travel Time [min]

ORIG \ DEST	Southampton	Isle of Wight	Eastleigh	New Forest
Southampton	8.39	98.98	14.38	70.09
Isle of Wight	81.82	16.96	91.70	65.89
Eastleigh	14.26	110.16	10.88	79.51
New Forest	34.58	96.32	44.24	60.42

Legend: Increase Decrease No Change

8

Chapter 1. Contents



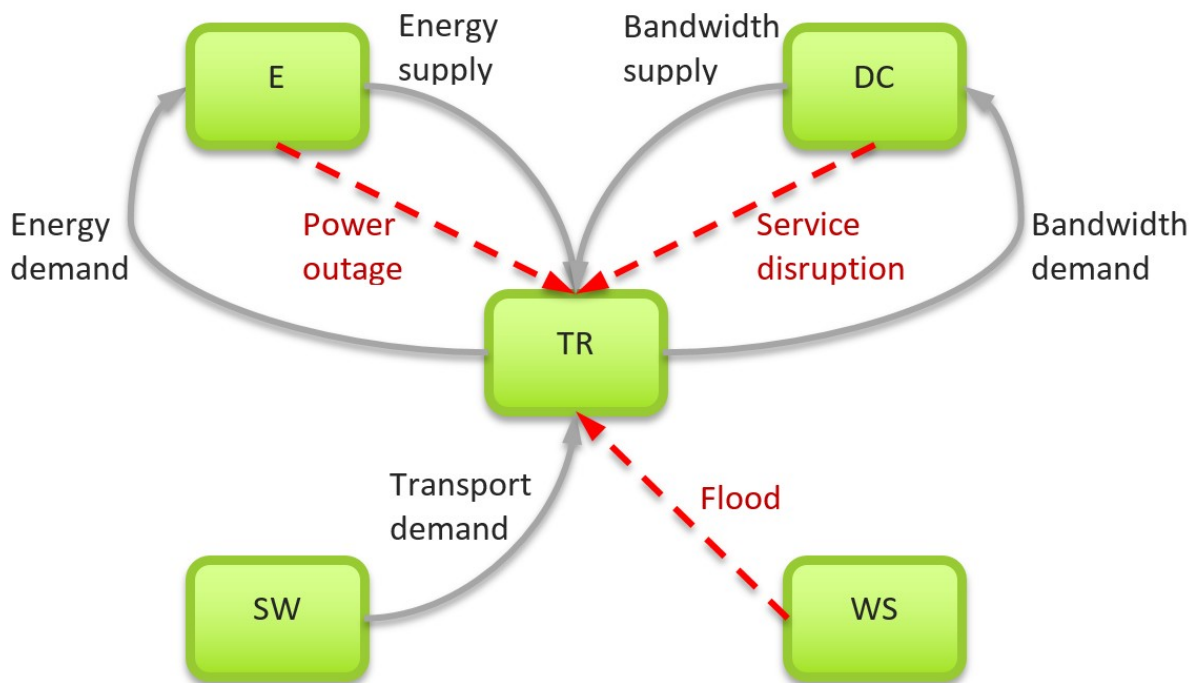
**Note:** Showcase demo requires a display with a 1920 x 1080 resolution.

### 1.1.3 3. Cross-sectoral Dependencies

The UK transport sector has various links with other infrastructure sectors:

- *Energy*: energy consumption, fuel price, electrification of vehicles, fuel transport, power outage (rail and air disruption).
- *Digital Communications*: supporting smart mobility (e.g. mobility as a service, autonomous mobility on demand), coverage and service disruptions.
- *Water*: floods causing road and rail disruptions.
- *Solid Waste*: waste transport (e.g. waste exports through seaports).

To enable studies of some of those cross-sectoral interdependencies, the transport model has been integrated into a wider *Simulation Modelling Integration Framework (smif)*: <https://github.com/nismod/smif>



## 1.1.4 4. Acknowledgments

This work has been undertaken at the *University of Southampton*, as part of the ITRC consortium, under grant EP/N017064/1 (MISTRAL: Multi-scale InfraSTRucture systems AnaLytics) of the UK *Engineering and Physical Science Research Council (EPSRC)*. <https://www.itrc.org.uk/>

The test resources contain a sample of data and shapefiles that come with the following licencing and copyright statements:

- *Open Government Licence* <http://www.nationalarchives.gov.uk/doc/open-government-licence/version/3/>
- Contains *National Statistics* data © *Crown* copyright and database right 2012.
- Contains *Ordnance Survey* data © *Crown* copyright and database right 2012.

The authors acknowledge the use of the *IRIDIS High Performance Computing Facility*, and associated support services at the *University of Southampton*, in the completion of this work.

The implementation uses an open source library *GeoTools* for geospatial processing. <http://www.geotools.org/about.html>

## 1.2 ‘transport’

### 1.2.1 nismod.transport

#### App

public class **App**  
NISMOD V2.0.0 Transport Model.  
**Author** Milan Lovric

#### Methods

##### main

public static void **main** (*String*[] *args*)

### 1.2.2 nismod.transport.air

#### AirDemandModel

public class **AirDemandModel**  
Main air demand prediction model (elasticity-based).  
**Author** Milan Lovric

#### Fields

##### baseYear

public static int **baseYear**

##### domesticAirports

public static *Map*<*String*, *Airport*> **domesticAirports**

##### internationalAirports

public static *Map*<*String*, *Airport*> **internationalAirports**

#### Constructors

## AirDemandModel

```
public AirDemandModel (String domesticAirportsFile, String internationalAirportsFile, String baseYearDomesticPassengerFile, String baseYearInternationalPassengerFile, String populationFile, String GVAFile, String elasticitiesFile, String domesticAirportFareIndexFile, String internationalAirportFareIndexFile, String domesticTripRatesFile, String internationalTripRatesFile, List<Intervention> interventions, Properties props)
```

Constructor for the air demand model.

### Parameters

- **domesticAirportsFile** – List of domestic airports.
- **internationalAirportsFile** – List of international airports.
- **baseYearDomesticPassengerFile** – Base year domestic air passenger file (demand).
- **baseYearInternationalPassengerFile** – Base year international air passenger file (demand).
- **populationFile** – Population file.
- **GVAFile** – GVA file.
- **elasticitiesFile** – Elasticities file.
- **domesticAirportFareIndexFile** – Domestic airport fare index.
- **internationalAirportFareIndexFile** – International airport fare index.
- **domesticTripRatesFile** – Domestic trip rates file.
- **internationalTripRatesFile** – International trip rates file.
- **interventions** – List of interventions.
- **props** – Properties.

### Throws

- **IOException** –
- **FileNotFoundException** –

## Methods

### getDomesticAirPassengerDemand

```
public InternodalPassengerDemand getDomesticAirPassengerDemand (int year)
```

Getter method for the air passenger demand in a given year.

### Parameters

- **year** – Year for which the demand is requested.

**Returns** Air passenger demand.



### getInternationalAirPassengerDemand

public *InternodalPassengerDemand* **getInternationalAirPassengerDemand** (int *year*)

Getter method for the air passenger demand in a given year.

#### Parameters

- **year** – Year for which the demand is requested.

**Returns** Air passenger demand.

### predictAndSaveAirDemands

public void **predictAndSaveAirDemands** (int *toYear*, int *fromYear*)

Predicts air passenger demands (domestic and international) up to toYear (if flag is true, also intermediate years) and saves results.

#### Parameters

- **toYear** – The final year for which the demand is predicted.
- **fromYear** – The year from which the predictions are made.

### predictDomesticAirDemandUsingResultsOfFromYear

public void **predictDomesticAirDemandUsingResultsOfFromYear** (int *predictedYear*, int *fromYear*)

Predicts domestic air passenger internodal demand. Uses already existing results of the fromYear, from the output folder.

#### Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

### predictInternationalAirDemandUsingResultsOfFromYear

public void **predictInternationalAirDemandUsingResultsOfFromYear** (int *predictedYear*, int *fromYear*)

Predicts international air passenger internodal demand. Uses already existing results of the fromYear, from the output folder.

#### Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

### saveAllResults

public void **saveAllResults** (int *year*)

Saves all results into the output folder.

#### Parameters

- **year** – Year of the data.

## saveDomesticAirDemand

public void **saveDomesticAirDemand** (int *year*, String *outputFile*)  
Saves domestic air demand to an output file.

### Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

## saveInternationalAirDemand

public void **saveInternationalAirDemand** (int *year*, String *outputFile*)  
Saves international air demand to an output file.

### Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

## AirDemandModel.ElasticityTypes

public static enum **ElasticityTypes**

### Enum Constants

#### COST\_DOMESTIC

public static final *AirDemandModel.ElasticityTypes* **COST\_DOMESTIC**

#### COST\_INTERNATIONAL

public static final *AirDemandModel.ElasticityTypes* **COST\_INTERNATIONAL**

#### GVA

public static final *AirDemandModel.ElasticityTypes* **GVA**

#### POPULATION

public static final *AirDemandModel.ElasticityTypes* **POPULATION**

## Airport

public abstract class **Airport**  
This class stores information about an airport.

**Author** Milan Lovric

## Constructors

### Airport

```
public Airport (String iataCode, String caaName, String ourAirportsName, double longitude, double latitude, String countryCode, String continentCode, long terminalCapacity, long runwayCapacity)
```

Constructor for the airport.

#### Parameters

- **iataCode** – Airport IATA code.
- **caaName** – Airport name in CAA datasets.
- **ourAirportsName** – Airport name in ourAirports dataset.
- **longitude** – Longitude coordinate.
- **latitude** – Latitude coordinate.
- **countryCode** – Code of the country in which the airport is located.
- **continentCode** – Code of the continent in which the airport is located.
- **terminalCapacity** – Airport terminal capacity (max number of passengers that can be processed).
- **runwayCapacity** – Airport runway capacity (max number of flights that can be processed).

### Airport

```
public Airport (Airport airport)
```

Constructor for an airport.

#### Parameters

- **airport** – Airport which data is going to be copied.

## Methods

### getCAAName

```
public String getCAAName ()
```

Getter method for the airport CAA name.

**Returns** CAA name.

### getContinent

```
public ContinentCode getContinent ()
```

Getter method for the continent in which the airport is located.

**Returns** ContinentCode continent.

### getCountry

public [Locale](#) **getCountry** ()  
Getter method for the country in which the airport is located.  
**Returns** Country locale.

### getIataCode

public [String](#) **getIataCode** ()  
Getter method for the IATA code of the station.  
**Returns** IATA code.

### getLatitude

public double **getLatitude** ()  
Getter method for Latitude.  
**Returns** Latitude.

### getLongitude

public double **getLongitude** ()  
Getter method for longitude.  
**Returns** Longitude.

### getOurAirportsName

public [String](#) **getOurAirportsName** ()  
Getter method for the ourAirports name.  
**Returns** OurAirports name.

### getRunwayCapacity

public long **getRunwayCapacity** ()  
Getter method for the airport runway capacity.  
**Returns** Runway capacity.

### getTerminalCapacity

public long **getTerminalCapacity** ()  
Getter method for the airport terminal capacity.  
**Returns** Terminal capacity.

## toString

```
public String toString()
```

## Airport.AirportGroup

```
public static enum AirportGroup
    Airports grouped by DfT.
```

### Enum Constants

#### DO

```
public static final Airport.AirportGroup DO
```

#### LH

```
public static final Airport.AirportGroup LH
```

#### SH

```
public static final Airport.AirportGroup SH
```

## Airport.AirportGroupCAA

```
public static enum AirportGroupCAA
    Airport groups by CAA used in the flight movements dataset.
```

### Enum Constants

#### EU

```
public static final Airport.AirportGroupCAA EU
```

#### INT

```
public static final Airport.AirportGroupCAA INT
```

#### UK

```
public static final Airport.AirportGroupCAA UK
```

## Airport.ContinentCode

public static enum **ContinentCode**  
ISO continent code

### Enum Constants

#### AF

public static final *Airport.ContinentCode* **AF**

#### AN

public static final *Airport.ContinentCode* **AN**

#### AS

public static final *Airport.ContinentCode* **AS**

#### EU

public static final *Airport.ContinentCode* **EU**

#### NA

public static final *Airport.ContinentCode* **NA**

#### OC

public static final *Airport.ContinentCode* **OC**

#### SA

public static final *Airport.ContinentCode* **SA**

### Methods

#### getName

public *String* **getName** ()

## Airport.ForeignRegionCAA

public static enum **ForeignRegionCAA**

These are the foreign region groups used by CAA for international internodal passenger demand. There is 1:1 mapping between a country and a region (this is unlike the OurAirports data where one country could map to multiple regions, e.g. some Russian airports are in Asia, while some are in Europe).

### Enum Constants

#### ATLANTIC\_OCEAN\_ISLANDS

public static final *Airport.ForeignRegionCAA* **ATLANTIC\_OCEAN\_ISLANDS**

#### AUSTRALASIA

public static final *Airport.ForeignRegionCAA* **AUSTRALASIA**

#### CANADA

public static final *Airport.ForeignRegionCAA* **CANADA**

#### CARRIBEAN\_AREA

public static final *Airport.ForeignRegionCAA* **CARRIBEAN\_AREA**

#### CENTRAL\_AFRICA

public static final *Airport.ForeignRegionCAA* **CENTRAL\_AFRICA**

#### CENTRAL\_AMERICA

public static final *Airport.ForeignRegionCAA* **CENTRAL\_AMERICA**

#### EASTERN\_EUROPE\_EU

public static final *Airport.ForeignRegionCAA* **EASTERN\_EUROPE\_EU**

#### EASTERN\_EUROPE\_OTHER

public static final *Airport.ForeignRegionCAA* **EASTERN\_EUROPE\_OTHER**

#### EAST\_AFRICA

public static final *Airport.ForeignRegionCAA* **EAST\_AFRICA**

## **FAR\_EAST**

public static final *Airport.ForeignRegionCAA* **FAR\_EAST**

## **INDIAN\_OCEAN\_ISLANDS**

public static final *Airport.ForeignRegionCAA* **INDIAN\_OCEAN\_ISLANDS**

## **INDIAN\_SUB\_CONTINENT**

public static final *Airport.ForeignRegionCAA* **INDIAN\_SUB\_CONTINENT**

## **MIDDLE\_EAST**

public static final *Airport.ForeignRegionCAA* **MIDDLE\_EAST**

## **NEAR\_EAST**

public static final *Airport.ForeignRegionCAA* **NEAR\_EAST**

## **NORTH\_AFRICA**

public static final *Airport.ForeignRegionCAA* **NORTH\_AFRICA**

## **OIL\_RIGS**

public static final *Airport.ForeignRegionCAA* **OIL\_RIGS**

## **PACIFIC\_OCEAN\_ISLANDS**

public static final *Airport.ForeignRegionCAA* **PACIFIC\_OCEAN\_ISLANDS**

## **SOUTHERN\_AFRICA**

public static final *Airport.ForeignRegionCAA* **SOUTHERN\_AFRICA**

## **SOUTH\_AMERICA**

public static final *Airport.ForeignRegionCAA* **SOUTH\_AMERICA**

## **UNITED\_STATES\_OF\_AMERICA**

public static final *Airport.ForeignRegionCAA* **UNITED\_STATES\_OF\_AMERICA**



## WESTERN\_EUROPE\_EU

```
public static final Airport.ForeignRegionCAA WESTERN_EUROPE_EU
```

## WESTERN\_EUROPE\_OTHER

```
public static final Airport.ForeignRegionCAA WESTERN_EUROPE_OTHER
```

## WEST\_AFRICA

```
public static final Airport.ForeignRegionCAA WEST_AFRICA
```

## Methods

### getName

```
public String getName ()
```

## DomesticAirport

```
public class DomesticAirport extends Airport  
    This class stores information about a domestic (UK) airport.
```

**Author** Milan Lovric

## Constructors

### DomesticAirport

```
public DomesticAirport (String iataCode, String atcoCode, String caaName, String naptanName, String  
                        ourAirportsName, int easting, int northing, double longitude, double latitude,  
                        String ladCode, String ladName, long terminalCapacity, long runwayCapacity)
```

Constructor for the airport.

#### Parameters

- **iataCode** – Airport IATA code.
- **atcoCode** – Airport ATCO code.
- **caaName** – Airport name in CCA datasets.
- **naptanName** – Airport name in NaPTAN (for UK airports).
- **ourAirportsName** – Airport name in ourAirports dataset.
- **easting** – Easting coordinate.
- **northing** – Northing coordinate.
- **longitude** – Longitude coordinate.
- **latitude** – Latitude coordinate.

- **ladCode** – LAD code of the zone in which the airport is located (for UK airports).
- **ladName** – LAD name of the zone in which the airport is located (for UK airports).
- **terminalCapacity** – Airport terminal capacity (max number of passengers that can be processed).
- **runwayCapacity** – Airport runway capacity (max number of flights that can be processed).

## DomesticAirport

public **DomesticAirport** (*DomesticAirport* airport)  
Constructor for an airport.

### Parameters

- **airport** – Airport which data is going to be copied.

## Methods

### getAtcoCode

public *String* **getAtcoCode** ()  
Getter method for the airport ATCO code.

**Returns** NaPTAN name.

### getEasting

public int **getEasting** ()  
Getter method for easting.

**Returns** Easting.

### getLADCode

public *String* **getLADCode** ()  
Getter method for the LAD code in which station is located.

**Returns** LAD code.

### getLADName

public *String* **getLADName** ()  
Getter method for the LAD name in which station is located.

**Returns** LAD name.

### getNaPTANName

```
public String getNaPTANName ()
```

Getter method for the airport NaPTAN name.

**Returns** NaPTAN name.

### getNorthing

```
public int getNorthing ()
```

Getter method for Northing.

**Returns** Northing.

### toString

```
public String toString ()
```

## DomesticInternodalPassengerDemand

```
public class DomesticInternodalPassengerDemand extends InternodalPassengerDemand
```

This class encapsulates domestic internodal (domestic airport to domestic airport) passenger data.

**Author** Milan Lovric

## Constructors

### DomesticInternodalPassengerDemand

```
public DomesticInternodalPassengerDemand ()
```

### DomesticInternodalPassengerDemand

```
public DomesticInternodalPassengerDemand (String fileName)
```

Constructor that reads OD matrix from an input csv file.

#### Parameters

- **fileName** – Path to the input file.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### saveAirPassengerDemand

```
public void saveAirPassengerDemand (int year, String outputFile)
```

Saves air passenger demand to an output file.

#### Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

### InternationalAirport

public class **InternationalAirport** extends *Airport*

This class stores information about an international (non-UK) airport.

**Author** Milan Lovric

#### Constructors

### InternationalAirport

public **InternationalAirport** (*String iataCode*, *String caaName*, *String ourAirportsName*, double *longitude*, double *latitude*, *String countryCode*, *String continentCode*, long *terminalCapacity*, long *runwayCapacity*)

Constructor for the airport.

#### Parameters

- **iataCode** – Airport IATA code.
- **caaName** – Airport name in CCA datasets.
- **ourAirportsName** – Airport name in ourAirports dataset.
- **longitude** – Longitude coordinate.
- **latitude** – Latitude coordinate.
- **countryCode** – Code of the country in which the airport is located.
- **continentCode** – Code of the continent in which the airport is located.
- **terminalCapacity** – Airport terminal capacity (max number of passengers that can be processed).
- **runwayCapacity** – Airport runway capacity (max number of flights that can be processed).

### InternationalAirport

public **InternationalAirport** (*InternationalAirport airport*)

Constructor for an airport using an existing airport.

#### Parameters

- **airport** – Airport which data is going to be copied.

## Methods

### toString

```
public String toString ()
```

### InternationalInternodalPassengerDemand

public class **InternationalInternodalPassengerDemand** extends *InternodalPassengerDemand*  
This class encapsulates international internodal (domestic airport to international airport) passenger data.

**Author** Milan Lovric

## Constructors

### InternationalInternodalPassengerDemand

```
public InternationalInternodalPassengerDemand ()
```

### InternationalInternodalPassengerDemand

```
public InternationalInternodalPassengerDemand (String fileName)
```

Constructor that reads OD matrix from an input csv file.

#### Parameters

- **fileName** – Path to the input file.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### saveAirPassengerDemand

```
public void saveAirPassengerDemand (int year, String outputFile)
```

Saves air passenger demand to an output file.

#### Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

### InternodalPassengerDemand

```
public abstract class InternodalPassengerDemand
```

This class encapsulates internodal (airport to airport) passenger data.

**Author** Milan Lovric

## Fields

### data

protected MultiKeyMap **data**

## Constructors

### InternodalPassengerDemand

public **InternodalPassengerDemand** ()

## Methods

### getDemand

public Map<*Passengers*, Long> **getDemand** (*String firstIATA*, *String secondIATA*)

### printDemand

public void **printDemand** ()

### saveAirPassengerDemand

public abstract void **saveAirPassengerDemand** (int *year*, *String file*)

### setDemand

public void **setDemand** (*String firstIATA*, *String secondIATA*, long *totalPax*, long *scheduledPax*, long *charter-Pax*)

### InternodalPassengerDemand.Passengers

public static enum **Passengers**

## Enum Constants

### CHARTER

public static final *InternodalPassengerDemand.Passengers* **CHARTER**

### SCHEDULED

public static final *InternodalPassengerDemand.Passengers* **SCHEDULED**

## TOTAL

public static final *InternodalPassengerDemand.Passengers* **TOTAL**

### 1.2.3 nismod.transport.decision

#### CongestionCharging

public class **CongestionCharging** extends *Intervention*

Intervention that implements link-based congestion charge which depends on the vehicle type and time of day when trip is made.

**Author** Milan Lovric

#### Constructors

##### CongestionCharging

public **CongestionCharging** (*Properties props*)

Constructor.

##### Parameters

- **props** – Properties object.

##### CongestionCharging

public **CongestionCharging** (*String fileName*)

Constructor.

##### Parameters

- **fileName** – Path to the input properties file.

#### Methods

##### install

public void **install** (*Object o*)

##### uninstall

public void **uninstall** (*Object o*)

#### Intervention

public abstract class **Intervention**

Abstract class for policy interventions.

**Author** Milan Lovric

## Fields

### installed

protected boolean **installed**

### props

protected [Properties](#) **props**

## Constructors

### Intervention

protected **Intervention** ([Properties](#) *props*)

### Intervention

protected **Intervention** ([String](#) *fileName*)

## Methods

### getEndYear

public int **getEndYear** ()

**Returns** The last year in which intervention still remains installed.

### getProperty

public [String](#) **getProperty** ([String](#) *key*)

**Parameters**

- **key** – Name of the property

**Returns** Property

### getStartYear

public int **getStartYear** ()

**Returns** The year in which intervention is installed.

### getState

public boolean **getState** ()



## install

public abstract void **install** (*Object o*)

## toString

public *String* **toString** ()

## uninstall

public abstract void **uninstall** (*Object o*)

## Intervention.InterventionType

public static enum **InterventionType**

### Enum Constants

#### CongestionCharging

public static final *Intervention.InterventionType* **CongestionCharging**

#### NewRailStation

public static final *Intervention.InterventionType* **NewRailStation**

#### RoadDevelopment

public static final *Intervention.InterventionType* **RoadDevelopment**

#### RoadExpansion

public static final *Intervention.InterventionType* **RoadExpansion**

#### NewRailStation

public class **NewRailStation** extends *Intervention*  
Intervention that builds a new rail station.

**Author** Milan Lovric

## Constructors

### NewRailStation

public **NewRailStation** (*Properties props*)  
Constructor.

#### Parameters

- **props** – Properties of the intervention.

### NewRailStation

public **NewRailStation** (*String fileName*)  
Constructor.

#### Parameters

- **fileName** – File with the properties.

## Methods

### getNLC

public *Integer* **getNLC** ()

**Returns** NLC code of the new rail station.

### install

public void **install** (*Object o*)

### uninstall

public void **uninstall** (*Object o*)

## PricingPolicy

public class **PricingPolicy**  
A class that encapsulates the pricing policy for congestion charging intervention.  
**Author** Milan Lovric

## Constructors

### PricingPolicy

public **PricingPolicy** (*String policyName*, *String fileName*, *int maxEdgeID*, *List<Integer> edgeIDs*)  
Reads congestion charge file which contains charges that depend on vehicle type and time of day (hour).

**Parameters**

- **policyName** – Name of the policy.
- **fileName** – File name.
- **maxEdgeID** – Maximum edge ID.
- **edgeIDs** – List of edge IDs affected by the policy.

**Throws**

- **IOException** – if any.
- **FileNotFoundException** – if any.

**Return** Map with congestion charges.

**Methods****getLinkCharges**

```
public double[] getLinkCharges (VehicleType vht, TimeOfDay time)
```

Get link charges for a particular combination of vehicle type and time of day.

**Parameters**

- **vht** – Vehicle type.
- **time** – Time of day.

**Returns** Array with link charges.

**getPolicy**

```
public EnumMap<VehicleType, EnumMap<TimeOfDay, double[]>> getPolicy ()
```

Get the entire pricing policy (for combinations of vehicle type and time of day).

**Returns** Pricing policy.

**getPolicyEdges**

```
public List<Integer> getPolicyEdges ()
```

Return policy edges.

**Returns** Policy edges.

**getPolicyName**

```
public String getPolicyName ()
```

Return policy name.

**Returns** Policy name.

## RoadDevelopment

public class **RoadDevelopment** extends *Intervention*  
Intervention that creates a new road link between two existing nodes.

**Author** Milan Lovric

## Constructors

### RoadDevelopment

public **RoadDevelopment** (*Properties props*)  
Constructor.

#### Parameters

- **props** – Properties of the road development intervention.

### RoadDevelopment

public **RoadDevelopment** (*String fileName*)  
Constructor.

#### Parameters

- **fileName** – File with the properties.

## Methods

### getDevelopedEdgeID

public *Integer* **getDevelopedEdgeID** ()  
**Returns** Edge ID of the developed road link.

### getDevelopedEdgeID2

public *Integer* **getDevelopedEdgeID2** ()  
**Returns** Edge ID of the developed road link (in other direction)

### install

public void **install** (*Object o*)

### uninstall

public void **uninstall** (*Object o*)

## RoadExpansion

public class **RoadExpansion** extends *Intervention*

Intervention that expands a road link with a number of lanes.

**Author** Milan Lovric

## Constructors

### RoadExpansion

public **RoadExpansion** (*Properties props*)

Constructor.

#### Parameters

- **props** – Properties of the road expansion intervention.

### RoadExpansion

public **RoadExpansion** (*String fileName*)

Constructor.

#### Parameters

- **fileName** – File with the properties.

## Methods

### getExpandedEdgeID

public *Integer* **getExpandedEdgeID** (*RoadNetwork roadNetwork*)

#### Parameters

- **roadNetwork** – Road network

**Returns** Edge ID which should be expanded.

### install

public void **install** (*Object o*)

### uninstall

public void **uninstall** (*Object o*)

## 1.2.4 nismod.transport.demand

### AssignableODMatrix

public interface **AssignableODMatrix**

Origin-destination matrix for passenger vehicles.

**Author** Milan Lovric

### Methods

#### getIntFlow

public int **getIntFlow** (*String* originZone, *String* destinationZone)

Gets the flow for a given origin-destination pair as a whole number.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

#### getSortedDestinations

public *List*<*String*> **getSortedDestinations** ()

Gets the sorted list of destinations.

**Returns** List of destination zones.

#### getSortedOrigins

public *List*<*String*> **getSortedOrigins** ()

Gets the sorted list of origins.

**Returns** List of origin zones.

#### getTotalIntFlow

public int **getTotalIntFlow** ()

Gets sum of all the flows in the matrix.

**Returns** Sum of all the flows in the matrix (i.e. number of trips).

#### getUnsortedDestinations

public *List*<*String*> **getUnsortedDestinations** ()

Gets the unsorted list of destinations.

**Returns** List of destination zones.

### getUnsortedOrigins

```
public List<String> getUnsortedOrigins ()
```

Gets the unsorted list of origins.

**Returns** List of origin zones.

### DemandModel

```
public class DemandModel
```

Main demand prediction model (elasticity-based).

**Author** Milan Lovric

### Fields

#### assignmentIterations

```
public final int assignmentIterations
```

#### baseYear

```
public final int baseYear
```

#### baseYearFreight

```
public final int baseYearFreight
```

#### freightScalingFactor

```
public final double freightScalingFactor
```

#### linkTravelTimeAveragingWeight

```
public final double linkTravelTimeAveragingWeight
```

#### predictionIterations

```
public final int predictionIterations
```

### Constructors

## DemandModel

```
public DemandModel (RoadNetwork roadNetwork, String baseYearODMatrixFile, String baseYearFreight-
MatrixFile, String populationFile, String GVAFile, String elasticitiesFile, String
elasticitiesFreightFile, String energyUnitCostsFile, String unitCO2EmissionsFile,
String engineTypeFractionsFile, String autonomousVehiclesFractionsFile,
List<Intervention> interventions, RouteSetGenerator rsg, Zoning zoning, Prop-
erties props)
```

The constructor for the demand prediction model.

### Parameters

- **roadNetwork** – Road network for assignment.
- **baseYearODMatrixFile** – Base-year origin-destination matrix file name.
- **baseYearFreightMatrixFile** – Base-year freight matrix file name.
- **populationFile** – Population file name.
- **GVAFile** – GVA file name.
- **elasticitiesFile** – Elasticities file name.
- **elasticitiesFreightFile** – Elasticities freight file name.
- **energyUnitCostsFile** – Energy unit costs file name.
- **unitCO2EmissionsFile** – Unit CO2 emissions file name.
- **engineTypeFractionsFile** – Engine type fractions file.
- **autonomousVehiclesFractionsFile** – Autonomous vehicles fractions file.
- **interventions** – List of interventions.
- **rsg** – Route Set Generator with routes for both cars and freight.
- **zoning** – Zoning system (used for ‘tempro’ and ‘combined’ assignment type).
- **props** – Properties file.

### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### addCongestionCharges

```
public void addCongestionCharges (int year, PricingPolicy congestionCharges)
```

Adds congestion charges to the list of the existing ones.

### Parameters

- **year** – Year of the policy.
- **congestionCharges** – Congestion charges.



### assignBaseYear

public void **assignBaseYear** ()  
Assigned base year demand.

### getCongestionCharges

public *List<PricingPolicy>* **getCongestionCharges** (int *year*)  
Getter method for congestion charges.

#### Parameters

- **year** – Year of the congestion charges.

**Returns** Congestion charges.

### getCostSkimMatrix

public *SkimMatrix* **getCostSkimMatrix** (int *year*)  
Getter method for cost skim matrix in a given year.

#### Parameters

- **year** – Year for which the the skim matrix is requested.

**Returns** Cost skim matrix.

### getCostSkimMatrixFreight

public *SkimMatrixFreight* **getCostSkimMatrixFreight** (int *year*)  
Getter method for freight cost skim matrix in a given year.

#### Parameters

- **year** – Year for which the the skim matrix is requested.

**Returns** Cost skim matrix.

### getEngineTypeFractions

public *Map<VehicleType, Map<EngineType, Double>>* **getEngineTypeFractions** (int *year*)  
Getter method for engine type fractions in a given year.

#### Parameters

- **year** – Year of the data.

**Returns** Map with engine type fractions.

### getFreightDemand

public *FreightMatrix* **getFreightDemand** (int *year*)  
Getter method for the freight demand in a given year.

#### Parameters

- **year** – Year for which the demand is requested.

**Returns** Origin-destination matrix with freight vehicle flows.

### getListsOfLADsForNewRouteGeneration

public *HashMap<Integer, List<List<String>>>* **getListsOfLADsForNewRouteGeneration** ()

Getter method for the list of LADs.

**Returns** Lists of LADs for new route generation.

### getPassengerDemand

public *ODMatrixMultiKey* **getPassengerDemand** (int *year*)

Getter method for the passenger demand in a given year.

#### Parameters

- **year** – Year for which the demand is requested.

**Returns** Origin-destination matrix with passenger vehicle flows.

### getRoadNetwork

public *RoadNetwork* **getRoadNetwork** ()

Getter method for the road network.

**Returns** Road network.

### getRoadNetworkAssignment

public *RoadNetworkAssignment* **getRoadNetworkAssignment** (int *year*)

Getter method for the road network assignment in a given year.

#### Parameters

- **year** – Year for which the road network assignment is requested.

**Returns** Road network assignment.

### getTimeSkimMatrix

public *SkimMatrix* **getTimeSkimMatrix** (int *year*)

Getter method for time skim matrix in a given year.

#### Parameters

- **year** – Year for which the the skim matrix is requested.

**Returns** Time skim matrix.

### getTimeSkimMatrixFreight

public *SkimMatrixFreight* **getTimeSkimMatrixFreight** (int *year*)

Getter method for freight time skim matrix in a given year.

#### Parameters

- **year** – Year for which the the skim matrix is requested.

**Returns** Time skim matrix.

### predictHighwayDemand

public void **predictHighwayDemand** (int *predictedYear*, int *fromYear*)

Predicts (passenger and freight) highway demand (origin-destination vehicle flows).

#### Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

### predictHighwayDemandUsingResultsOfFromYear

public void **predictHighwayDemandUsingResultsOfFromYear** (int *predictedYear*, int *fromYear*)

Predicts (passenger and freight) highway demand (origin-destination vehicle flows). Uses already existing results of the fromYear.

#### Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

### predictHighwayDemands

public void **predictHighwayDemands** (int *toYear*, int *baseYear*)

Predicts (passenger and freight) highway demand (origin-destination vehicle flows) for all years from baseYear to toYear

#### Parameters

- **toYear** – The final year for which the demand is predicted.
- **baseYear** – The base year from which the predictions are made.

### removeCongestionCharges

public void **removeCongestionCharges** (int *year*, *PricingPolicy* *congestionCharges*)

Removes congestion charges from the list of the existing ones.

#### Parameters

- **year** – Year of the congestion charges.
- **congestionCharges** – Congestion charges.

## removeCongestionCharges

public void **removeCongestionCharges** (int *year*, String *policyName*)

Removes congestion charges from the list of the existing ones using the policy name.

### Parameters

- **year** – Year of the congestion charges.
- **policyName** – Name of the policy.

## saveAllResults

public void **saveAllResults** (int *toYear*, int *baseYear*)

Saves all results from baseYear to toYear (including intermediate if flat is set)

### Parameters

- **toYear** – The final year for which the demand is predicted.
- **baseYear** – The base year from which the predictions are made.

## saveAllResults

public void **saveAllResults** (int *year*)

Saves all results into the output folder.

### Parameters

- **year** – Year of the data.

## saveAssignmentResults

public void **saveAssignmentResults** (int *year*, String *outputFile*)

Saves road network assignment results into a csv file.

### Parameters

- **year** – Year of the data.
- **outputFile** – Output file name.

## saveEnergyConsumptions

public void **saveEnergyConsumptions** (int *year*, String *outputFile*)

Saves energy consumptions into a csv file.

### Parameters

- **year** – Year of the data.
- **outputFile** – Output file name.

## setCongestionCharges

public void **setCongestionCharges** (int year, *List<PricingPolicy>* congestionCharges)  
Setter method for congestion charges (overrides them completely).

### Parameters

- **year** – Year of the congestion charges.
- **congestionCharges** – Congestion charges.

## setEngineTypeFractions

public void **setEngineTypeFractions** (int year, *Map<VehicleType, Map<EngineType, Double>>* engine-  
TypeFractions)  
Setter method for engine type fractions in a given year.

### Parameters

- **year** – Year of the data.
- **engineTypeFractions** – Map with engine type fractions.

## setEngineTypeFractions

public void **setEngineTypeFractions** (int year, *VehicleType* vht, *Map<EngineType, Double>* engine-  
TypeFractions)  
Setter method for engine type fractions in a given year for a specific vehicle type.

### Parameters

- **year** – Year of the data.
- **vht** – Vehicle type.
- **engineTypeFractions** – Map with engine type fractions.

## DemandModel.ElasticityTypes

public static enum **ElasticityTypes**

### Enum Constants

#### COST

public static final *DemandModel.ElasticityTypes* **COST**

#### GVA

public static final *DemandModel.ElasticityTypes* **GVA**

## POPULATION

public static final *DemandModel.ElasticityTypes* **POPULATION**

## TIME

public static final *DemandModel.ElasticityTypes* **TIME**

## EstimatedODMatrix

public class **EstimatedODMatrix** extends *RealODMatrix*

Origin-destination matrix created from productions, attractions and observed trip length distribution.

**Author** Milan Lovric

## Fields

### BIN\_LIMITS\_KM

public static final double[] **BIN\_LIMITS\_KM**

### BIN\_LIMITS\_MILES

public static final int[] **BIN\_LIMITS\_MILES**

## OTLD

public static final double[] **OTLD**

## Constructors

### EstimatedODMatrix

public **EstimatedODMatrix** (*HashMap<String, Integer>* productions, *HashMap<String, Integer>* attractions, *SkimMatrix* distanceSkimMatrix, double[] binLimitsKm, double[] observedTripLengthDistribution)

Constructor for estimated OD matrix.

#### Parameters

- **productions** – Productions
- **attractions** – Attractions
- **distanceSkimMatrix** – Distance skim matrix
- **binLimitsKm** – Bin limits in km
- **observedTripLengthDistribution** – Observed trip length distribution (normalised).

## EstimatedODMatrix

```
public EstimatedODMatrix (String fileName, SkimMatrix distanceSkimMatrix, double[] binLimitsKm,
                           double[] observedTripLengthDistribution)
```

Constructor for estimated OD matrix that reads productions and attractions from an input csv file.

### Parameters

- **fileName** – Path to the input file with productions and attractions
- **distanceSkimMatrix** – Distance skim matrix
- **binLimitsKm** – Bin limits in km
- **observedTripLengthDistribution** – Observed trip length distribution (normalised).

### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### createUnitMatrix

```
public void createUnitMatrix ()
```

Creates a unit OD matrix (all ones).

### deleteInterzonalFlows

```
public void deleteInterzonalFlows (String zone)
```

Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

### Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

### getAttractions

```
public HashMap<String, Integer> getAttractions ()
```

Getter method for the attractions.

**Returns** Attractions

### getBinIndexMatrix

```
public ODMatrixMultiKey getBinIndexMatrix ()
```

Getter method for the bin index matrix.

**Returns** Bin index matrix

### getObservedTripLengthDistribution

public double[] **getObservedTripLengthDistribution** ()  
Getter method for the observed trip length distribution.

**Returns** Observed trip length distribution

### getProductions

public [HashMap](#)<[String](#), [Integer](#)> **getProductions** ()  
Getter method for the productions.

**Returns** Productions

### getTripLengthDistribution

public double[] **getTripLengthDistribution** ()  
Getter method for the trip length distribution.

**Returns** Trip length distribution

### iterate

public void **iterate** ()  
Iterates scaling to productions, scaling to attractions, rounding and scaling to observed trip length distribution.

### printMatrixFormatted

public void **printMatrixFormatted** (int *precision*)  
Prints the matrix as a formatted table.

### printMatrixFormatted

public void **printMatrixFormatted** ([String](#) *message*, int *precision*)  
Prints the message and the matrix as a formatted table.

### scaleToAttractions

public void **scaleToAttractions** ()  
Scales OD matrix to attractions.

### scaleToObservedTripLengthDistribution

public void **scaleToObservedTripLengthDistribution** ()  
Scales OD matrix to observed trip length distribution.



### scaleToProductions

```
public void scaleToProductions ()  
    Scales OD matrix to productions.
```

### updateTripLengthDistribution

```
public void updateTripLengthDistribution ()  
    Updates trip length distribution (using the current values of the OD matrix).
```

### FreightMatrix

```
public class FreightMatrix  
    Origin-destination matrix for freight vehicles (following the format of DfT's BYFM 2006 study).  
    Author Milan Lovric
```

### Fields

#### MAX\_FREIGHT\_ZONE\_ID

```
public static final int MAX_FREIGHT_ZONE_ID
```

#### MAX\_VEHICLE\_ID

```
public static final int MAX_VEHICLE_ID
```

### Constructors

#### FreightMatrix

```
public FreightMatrix ()
```

#### FreightMatrix

```
public FreightMatrix (String fileName)  
    Constructor that reads OD matrix from an input csv file.
```

##### Parameters

- **fileName** – Path to the input file.

##### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### createUnitBYFMMatrix

public static *FreightMatrix* **createUnitBYFMMatrix** ()  
Creates a unit freight matrix for the specific DfT BYFM 2006 zoning system.  
**Returns** Unit BYFM freight matrix.

### createUnitMatrix

public static *FreightMatrix* **createUnitMatrix** (*List<Integer> origins*, *List<Integer> destinations*)  
Creates a unit freight matrix for given lists of origin and destination zones.  
**Parameters**

- **origins** – List of origin zones.
- **destinations** – List of destination zones.

**Returns** Unit freight matrix.

### deleteInterzonalFlows

public void **deleteInterzonalFlows** (int *zone*)  
Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)  
**Parameters**

- **zone** – Zone for which inter-zonal flows need to be deleted from the freight matrix.

### getAbsoluteDifference

public double **getAbsoluteDifference** (*FreightMatrix other*)  
Gets sum of absolute differences between elements of two matrices.  
**Parameters**

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getFlow

public int **getFlow** (int *origin*, int *destination*, int *vehicleType*)  
Gets the flow for a given origin-destination pair.  
**Parameters**

- **origin** – Freight origin.
- **destination** – Freight destination.
- **vehicleType** – Vehicle type.

**Returns** Origin-destination flow.

### getKeySet

public [Set](#)<[MultiKey](#)> **getKeySet** ()  
Gets the keyset of the multimap.

**Returns** Key set.

### getScaledMatrix

public [FreightMatrix](#) **getScaledMatrix** (double *scale*)  
Multiplies each value of the matrix with a scaling factor.

**Parameters**

- **scale** – Scaling factor.

**Returns** Scaled freight matrix.

### getSortedDestinations

public [List](#)<[Integer](#)> **getSortedDestinations** ()  
Gets the sorted list of destinations.

**Returns** List of destinations.

### getSortedOrigins

public [List](#)<[Integer](#)> **getSortedOrigins** ()  
Gets the sorted list of origins.

**Returns** List of origins.

### getTotalIntFlow

public int **getTotalIntFlow** ()  
Gets sum of all the flows in the matrix.

**Returns** Sum of all the flows in the matrix (i.e. number of trips).

### getUnsortedDestinations

public [List](#)<[Integer](#)> **getUnsortedDestinations** ()  
Gets the unsorted list of destinations.

**Returns** List of destinations.

### getUnsortedOrigins

public [List](#)<[Integer](#)> **getUnsortedOrigins** ()  
Gets the unsorted list of origins.

**Returns** List of origins.

## getVehicleTypes

public `List<Integer>` **getVehicleTypes** ()

Gets the sorted list of vehicle types.

**Returns** List of vehicle types.

## printMatrix

public void **printMatrix** ()

Prints the full matrix.

## printMatrixFormatted

public void **printMatrixFormatted** ()

Prints the matrix as a formatted table.

## printMatrixFormatted

public void **printMatrixFormatted** (`String s`)

Prints the matrix as a formatted table, with a print message.

### Parameters

- **s** – Print message

## saveMatrixFormatted

public void **saveMatrixFormatted** (`String outputFile`)

Saves the matrix into a csv file.

### Parameters

- **outputFile** – Path to the output file.

## scaleMatrix

public void **scaleMatrix** (`SkimMatrixFreight scale`)

Scales matrix flows using a real-valued scaling matrix.

### Parameters

- **scale** – Scaling factors.

## setFlow

public void **setFlow** (int *origin*, int *destination*, int *vehicleType*, int *flow*)

Sets the flow for a given origin-destination pair.

### Parameters

- **origin** – Freight origin.

- **destination** – Freight destination.
- **vehicleType** – Vehicle type.
- **flow** – Origin-destination flow.

## ODMatrixArray

public class **ODMatrixArray** implements *AssignableODMatrix*  
Origin-destination matrix for passenger vehicles.

**Author** Milan Lovric

## Constructors

### ODMatrixArray

public **ODMatrixArray** (*Zoning* zoning)  
Constructor.

#### Parameters

- **zoning** – Zoning system.

### ODMatrixArray

public **ODMatrixArray** (*RealODMatrix* realMatrix, *Zoning* zoning)  
Constructor that rounds the flows of a real-valued OD matrix.

#### Parameters

- **realMatrix** – Origin-destination matrix with real-valued flows.
- **zoning** – Zoning system.

### ODMatrixArray

public **ODMatrixArray** (*String* fileName, *Zoning* zoning)  
Constructor that reads OD matrix from an input csv file.

#### Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### calculateTripEnds

```
public HashMap<String, Integer> calculateTripEnds ()
```

Calculates the number of trips ending in each destination zone

**Returns** Number of trip ends.

### calculateTripStarts

```
public HashMap<String, Integer> calculateTripStarts ()
```

Calculates the number of trips starting in each origin zone

**Returns** Number of trip starts.

### createLadMatrixFromTEMProMatrix

```
public static ODMatrixArray createLadMatrixFromTEMProMatrix (ODMatrixArrayTempro tempro-  
Matrix, Zoning zoning)
```

Creates LAD OD matrix from TEMPro OD matrix.

**Parameters**

- **temproMatrix** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

**Returns** LAD based OD matrix.

### createTEMProFromLadMatrix

```
public static ODMatrixArrayTempro createTEMProFromLadMatrix (ODMatrixArray ladODMatrix,  
ODMatrixArrayTempro baseTem-  
pro, Zoning zoning)
```

Creates tempPro OD matrix from LAD OD matrix.

**Parameters**

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempPro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

**Returns** TEMPro based OD matrix.

### createUnitMatrix

```
public static ODMatrixArray createUnitMatrix (List<String> origins, List<String> destinations, Zoning  
zoning)
```

Creates a unit OD matrix for given lists of origin and destination zones.

**Parameters**

- **origins** – List of origin zones.

- **destinations** – List of destination zones.
- **zoning** – Zoning system.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *ODMatrixArray* **createUnitMatrix** (*List<String>* zones, *Zoning* zoning)  
Creates a quadratic unit OD matrix for a given lists of zones.

#### Parameters

- **zones** – List of origin zones.
- **zoning** – Zoning system.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *ODMatrixArray* **createUnitMatrix** (*Set<String>* zones, *Zoning* zoning)  
Creates a quadratic unit OD matrix for a given lists of zones.

#### Parameters

- **zones** – Set of origin zones.
- **zoning** – Zoning system.

**Returns** Unit OD matrix.

### deleteInterzonalFlows

public void **deleteInterzonalFlows** (*String* zone)  
Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

#### Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

### getAbsoluteDifference

public int **getAbsoluteDifference** (*ODMatrixArray* other)  
Gets sum of absolute differences between elements of two matrices.

#### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

## getFlow

public int **getFlow** (*String originZone*, *String destinationZone*)

Gets the flow for a given origin-destination pair.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

## getFlow

public int **getFlow** (int *originZoneID*, int *destinationZoneID*)

Gets the flow for a given origin-destination pair.

### Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.

**Returns** Origin-destination flow.

## getIntFlow

public int **getIntFlow** (*String originZone*, *String destinationZone*)

Gets the flow for a given origin-destination pair as a whole number.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

## getIntFlow

public int **getIntFlow** (int *originZoneID*, int *destinationZoneID*)

Gets the flow for a given origin-destination pair as a whole number.

### Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.

**Returns** Origin-destination flow.

## getSortedDestinations

public *List<String>* **getSortedDestinations** ()

Gets the sorted list of destinations.

**Returns** List of destinations.



### getSortedOrigins

```
public List<String> getSortedOrigins ()
```

Gets the sorted list of origins.

**Returns** List of origins.

### getTotalFlow

```
public int getTotalFlow ()
```

Gets sum of all the flows in the matrix.

**Returns** Sum of all the flows in the matrix (i.e. number of trips).

### getTotalIntFlow

```
public int getTotalIntFlow ()
```

Gets sum of all the flows in the matrix.

**Returns** Sum of all the flows in the matrix (i.e. number of trips).

### getUnsortedDestinations

```
public List<String> getUnsortedDestinations ()
```

Gets the unsorted list of destinations.

**Returns** List of destinations.

### getUnsortedOrigins

```
public List<String> getUnsortedOrigins ()
```

Gets the unsorted list of origins.

**Returns** List of origins.

### printMatrix

```
public void printMatrix ()
```

Prints the full matrix.

### printMatrixFormatted

```
public void printMatrixFormatted ()
```

Prints the matrix as a formatted table.

### printMatrixFormatted

public void **printMatrixFormatted** ([String](#) *s*)  
Prints the matrix as a formatted table, with a print message.

#### Parameters

- **s** – Print message

### saveMatrixFormatted

public void **saveMatrixFormatted** ([String](#) *outputFile*)  
Saves the matrix into a csv file.

#### Parameters

- **outputFile** – Path to the output file.

### saveMatrixFormattedList

public void **saveMatrixFormattedList** ([String](#) *outputFile*)  
Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

#### Parameters

- **outputFile** – Path to the output file.

### scaleMatrixValue

public void **scaleMatrixValue** (double *factor*)  
Scales (and rounds) matrix values with a scaling factor.

#### Parameters

- **factor** – Scaling factor.

### setFlow

public void **setFlow** (int *originZoneID*, int *destinationZoneID*, int *flow*)  
Sets the flow for a given origin-destination pair.

#### Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.
- **flow** – Origin-destination flow.

### setFlow

public void **setFlow** ([String](#) *originZone*, [String](#) *destinationZone*, int *flow*)  
Sets the flow for a given origin-destination pair.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

## ODMatrixArrayTempro

public class **ODMatrixArrayTempro** implements *AssignableODMatrix*  
Origin-destination matrix with real values, memory use optimised for Tempro.

**Author** Milan Lovric

## Constructors

### ODMatrixArrayTempro

public **ODMatrixArrayTempro** (*Zoning* zoning)  
Constructor for an empty OD matrix. Uses the maximum Tempro zone ID which will create a rather large matrix.

#### Parameters

- **zoning** – Zoning system.

### ODMatrixArrayTempro

public **ODMatrixArrayTempro** (*String* fileName, *Zoning* zoning)  
Constructor that reads OD matrix from an input csv file. Can use both matrix and list format.

#### Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### calculateTripEnds

public *HashMap*<*String*, *Integer*> **calculateTripEnds** ()  
Calculates the number of trips ending in each destination zone

**Returns** Number of trip ends.

## calculateTripStarts

```
public HashMap<String, Integer> calculateTripStarts ()
```

Calculates the number of trips starting in each origin zone

**Returns** Number of trip starts.

## clone

```
public ODMatrixArrayTempro clone ()
```

## createLadMatrixFromTEMProMatrix

```
public static RealODMatrix createLadMatrixFromTEMProMatrix (ODMatrixArrayTempro tempromatrix, Zoning zoning)
```

Creates real-valued LAD OD matrix from real-valued TEMPPro OD matrix.

### Parameters

- **tempromatrix** – TEMPPro ODMatrix which should be aggregated to LAD matrix.
- **zoning** – Zoning system with mapping between TEMPPro and LAD zones.

**Returns** LAD based real-valued OD matrix.

## createTEMPProFromLadMatrix

```
public static ODMatrixArrayTempro createTEMPProFromLadMatrix (ODMatrixArray ladODMatrix, ODMatrixArrayTempro baseTempromatrix, Zoning zoning)
```

Creates tempromatrix OD matrix from LAD OD matrix.

### Parameters

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempromatrix** – TEMPPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPPro and LAD zones.

**Returns** TEMPPro based OD matrix.

## createTEMPProFromLadMatrix

```
public static ODMatrixArrayTempro createTEMPProFromLadMatrix (ODMatrixMultiKey ladODMatrix, ODMatrixArrayTempro baseTempromatrix, Zoning zoning)
```

Creates tempromatrix OD matrix from LAD OD matrix.

### Parameters

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempromatrix** – TEMPPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPPro and LAD zones.

**Returns** TEMPPro based OD matrix.

### createUnitMatrix

public static *ODMatrixArrayTempro* **createUnitMatrix** (*List<String>* origins, *List<String>* destinations, *Zoning* zoning)

Creates a unit OD matrix for given lists of origin and destination zones.

#### Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **zoning** – Zoning system.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *ODMatrixArrayTempro* **createUnitMatrix** (*List<String>* zones, *Zoning* zoning)

Creates a quadratic unit OD matrix for a given lists of zones.

#### Parameters

- **zones** – List of zones.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *ODMatrixArrayTempro* **createUnitMatrix** (*Set<String>* zones, *Zoning* zoning)

Creates a quadratic unit OD matrix for a given lists of zones.

#### Parameters

- **zones** – Set of zones.
- **zoning** – Zoning system.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *ODMatrixArrayTempro* **createUnitMatrix** (*Zoning* zoning)

Creates a quadratic unit OD matrix.

#### Parameters

- **zoning** – Zoning system.

**Returns** Unit OD matrix.

### deleteInterzonalFlows

public void **deleteInterzonalFlows** (*String* zone)

Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

#### Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

### getAbsoluteDifference

public int **getAbsoluteDifference** (*ODMatrixArrayTempro other*)

Gets sum of absolute differences between elements of two matrices.

#### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getFlow

public int **getFlow** (*String originZone, String destinationZone*)

Gets the flow for a given origin-destination pair.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

### getIntFlow

public int **getIntFlow** (*String originZone, String destinationZone*)

Gets the flow for a given origin-destination pair, rounded to a whole number.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

### getSortedDestinations

public *List<String>* **getSortedDestinations** ()

Gets the sorted list of destinations.

**Returns** List of destinations.

### getSortedOrigins

public *List<String>* **getSortedOrigins** ()

Gets the sorted list of origins.

**Returns** List of origins.

### getTotalIntFlow

public int **getTotalIntFlow** ()

Gets sum of all the (rounded) flows in the matrix.

**Returns** Sum of all the (rounded) flows in the matrix (i.e. number of trips).

### getUnsortedDestinations

public [List<String>](#) **getUnsortedDestinations** ()

Gets the unsorted list of destinations.

**Returns** List of destinations.

### getUnsortedOrigins

public [List<String>](#) **getUnsortedOrigins** ()

Gets the unsorted list of origins.

**Returns** List of origins.

### printMatrix

public void **printMatrix** ()

Prints the full matrix.

### printMatrixFormatted

public void **printMatrixFormatted** ()

Prints the matrix as a formatted table.

### printMatrixFormatted

public void **printMatrixFormatted** ([String](#) s)

Prints message followed by the formatted matrix.

**Parameters**

- **s** – Message.

### saveMatrixFormatted

public void **saveMatrixFormatted** ([String](#) outputFile)

Saves the matrix into a csv file. Uses a rectangular/matrix format.

**Parameters**

- **outputFile** – Path to the output file.

### saveMatrixFormatted2

public void **saveMatrixFormatted2** (*String outputFile*)

Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

#### Parameters

- **outputFile** – Path to the output file.

### saveMatrixFormatted3

public void **saveMatrixFormatted3** (*String outputFile*)

Saves the matrix into a csv file. Uses a list format (origin, destination, flow) and number codes for zones.

#### Parameters

- **outputFile** – Path to the output file.

### scaleMatrixValue

public void **scaleMatrixValue** (double *factor*)

Scales matrix values with a scaling factor.

#### Parameters

- **factor** – Scaling factor.

### scaleMatrixValue

public void **scaleMatrixValue** (*ODMatrixArrayTempro scalingMatrix*)

Scales matrix values with another matrix (element-wise multiplication).

#### Parameters

- **scalingMatrix** – Scaling matrix.

### setFlow

public void **setFlow** (*String originZone, String destinationZone, int flow*)

Sets the flow for a given origin-destination pair.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

### setFlow

public void **setFlow** (int *originCode, int destinationCode, int flow*)

Sets the flow for a given origin-destination pair.

#### Parameters



- **originCode** – Origin zone integer code.
- **destinationCode** – Destination zone integer code.
- **flow** – Origin-destination flow.

### sumMatrixSubset

public int **sumMatrixSubset** (*List<String> origins*, *List<String> destinations*)

Sums the elements of a matrix subset (provided as two lists of origins and destinations).

#### Parameters

- **origins** – List of origin zones (a subset).
- **destinations** – List of destination zones (a subset).

**Returns** Sum of the subset.

### ODMatrixMultiKey

public class **ODMatrixMultiKey** implements *AssignableODMatrix*

Origin-destination matrix for passenger vehicles.

**Author** Milan Lovric

### Constructors

#### ODMatrixMultiKey

public **ODMatrixMultiKey** ()

#### ODMatrixMultiKey

public **ODMatrixMultiKey** (*String fileName*)

Constructor that reads OD matrix from an input csv file.

#### Parameters

- **fileName** – Path to the input file.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

#### ODMatrixMultiKey

public **ODMatrixMultiKey** (*RealODMatrix realMatrix*)

Constructor that rounds the flows of a real-valued OD matrix.

#### Parameters

- **realMatrix** – Origin-destination matrix with real-valued flows.

## Methods

### calculateTripEnds

public `HashMap<String, Integer>` **calculateTripEnds** ()  
Calculates the number of trips ending in each destination zone  
**Returns** Number of trip ends.

### calculateTripStarts

public `HashMap<String, Integer>` **calculateTripStarts** ()  
Calculates the number of trips starting in each origin zone  
**Returns** Number of trip starts.

### clone

public `ODMatrixMultiKey` **clone** ()

### createLadMatrixFromTEMProMatrix

public static `ODMatrixMultiKey` **createLadMatrixFromTEMProMatrix** (`ODMatrixMultiKey` *temproMatrix*, *Zoning* *zoning*)  
Creates LAD OD matrix from TEMPro OD matrix.  
**Parameters**

- **temproMatrix** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

**Returns** LAD based OD matrix.

### createSparseUnitMatrix

public static `ODMatrixMultiKey` **createSparseUnitMatrix** (`Set<String>` *zones*, `HashMap<String, Point>` *centroids*, double *threshold*)  
Creates a unit OD matrix for a given lists of zones with a distance threshold. If straight line distance between origin and destination zone centroids is larger than threshold that flow is zero.  
**Parameters**

- **zones** – Set of origin zones.
- **centroids** – List of zone centroids.
- **threshold** – Distance threshold in [m].

**Returns** Unit OD matrix.

**createTEMProFromLadMatrix**

```
public static ODMatrixMultiKey createTEMProFromLadMatrix (ODMatrixMultiKey ladODMatrix,
                                                         ODMatrixMultiKey baseTempro,
                                                         Zoning zoning)
```

Creates temprow OD matrix from LAD OD matrix.

**Parameters**

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTemprow** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

**Returns** TEMPro based OD matrix.

**createUnitMatrix**

```
public static ODMatrixMultiKey createUnitMatrix (List<String> origins, List<String> destinations)
```

Creates a unit OD matrix for given lists of origin and destination zones.

**Parameters**

- **origins** – List of origin zones.
- **destinations** – List of destination zones.

**Returns** Unit OD matrix.

**createUnitMatrix**

```
public static ODMatrixMultiKey createUnitMatrix (List<String> zones)
```

Creates a quadratic unit OD matrix for a given lists of zones.

**Parameters**

- **zones** – List of origin zones.

**Returns** Unit OD matrix.

**createUnitMatrix**

```
public static ODMatrixMultiKey createUnitMatrix (Set<String> zones)
```

Creates a quadratic unit OD matrix for a given lists of zones.

**Parameters**

- **zones** – Set of origin zones.

**Returns** Unit OD matrix.

**deleteInterzonalFlows**

```
public void deleteInterzonalFlows (String zone)
```

Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

**Parameters**

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

### getAbsoluteDifference

public double **getAbsoluteDifference** (*ODMatrixMultiKey other*)  
Gets sum of absolute differences between elements of two matrices.

#### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getFlow

public int **getFlow** (*String originZone, String destinationZone*)  
Gets the flow for a given origin-destination pair.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

### getIntFlow

public int **getIntFlow** (*String originZone, String destinationZone*)  
Gets the flow for a given origin-destination pair as a whole number.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

### getKeySet

public *Set<MultiKey>* **getKeySet** ()  
Gets the keyset of the multimap.

**Returns** Key set.

### getMatrixSubset

public *ODMatrixMultiKey* **getMatrixSubset** (*List<String> origins, List<String> destinations*)  
Creates a new OD matrix (a matrix subset) for given lists of origin and destination zones.

#### Parameters

- **origins** – List of origin zones.

- **destinations** – List of destination zones.

**Returns** Matrix subset.

### getSortedDestinations

```
public List<String> getSortedDestinations ()
```

Gets the sorted list of destinations.

**Returns** List of destinations.

### getSortedOrigins

```
public List<String> getSortedOrigins ()
```

Gets the sorted list of origins.

**Returns** List of origins.

### getSumOfFlows

```
public int getSumOfFlows ()
```

Gets sum of all the flows.

**Returns** Sum of flows.

### getTotalFlow

```
public int getTotalFlow ()
```

Gets sum of all the flows in the matrix.

**Returns** Sum of all the flows in the matrix (i.e. number of trips).

### getTotalIntFlow

```
public int getTotalIntFlow ()
```

Gets sum of all the flows in the matrix.

**Returns** Sum of all the flows in the matrix (i.e. number of trips).

### getUnsortedDestinations

```
public List<String> getUnsortedDestinations ()
```

Gets the unsorted list of destinations.

**Returns** List of destinations.

### getUnsortedOrigins

public `List<String>` **getUnsortedOrigins** ()

Gets the unsorted list of origins.

**Returns** List of origins.

### printMatrix

public void **printMatrix** ()

Prints the full matrix.

### printMatrixFormatted

public void **printMatrixFormatted** ()

Prints the matrix as a formatted table.

### printMatrixFormatted

public void **printMatrixFormatted** (`String s`)

Prints the matrix as a formatted table, with a print message.

#### Parameters

- **s** – Print message

### saveMatrixFormatted

public void **saveMatrixFormatted** (`String outputFile`)

Saves the matrix into a csv file.

#### Parameters

- **outputFile** – Path to the output file.

### saveMatrixFormatted2

public void **saveMatrixFormatted2** (`String outputFile`)

Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

#### Parameters

- **outputFile** – Path to the output file.

### scaleMatrixValue

public void **scaleMatrixValue** (double *factor*)

Scales (and rounds) matrix values with a scaling factor.

#### Parameters

- **factor** – Scaling factor.

## setFlow

public void **setFlow** ([String](#) *originZone*, [String](#) *destinationZone*, int *flow*)  
Sets the flow for a given origin-destination pair.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

## sumMatrixSubset

public int **sumMatrixSubset** ([List](#)<[String](#)> *origins*, [List](#)<[String](#)> *destinations*)  
Sums the elements of a matrix subset (provided as two lists of origins and destinations).

### Parameters

- **origins** – List of origin zones (a subset).
- **destinations** – List of destination zones (a subset).

**Returns** Sum of the subset.

## RealODMatrix

public class **RealODMatrix** implements [AssignableODMatrix](#)  
Origin-destination matrix with real values.

**Author** Milan Lovric

## Constructors

### RealODMatrix

public **RealODMatrix** ()

### RealODMatrix

public **RealODMatrix** ([String](#) *fileName*)  
Constructor that reads OD matrix from an input csv file.

### Parameters

- **fileName** – Path to the input file.

### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### calculateTripEnds

```
public HashMap<String, Integer> calculateTripEnds ()
```

Calculates the number of trips ending in each destination zone

**Returns** Number of trip ends.

### calculateTripStarts

```
public HashMap<String, Integer> calculateTripStarts ()
```

Calculates the number of trips starting in each origin zone

**Returns** Number of trip starts.

### clone

```
public RealODMatrix clone ()
```

### createLadMatrixFromTEMProMatrix

```
public static RealODMatrix createLadMatrixFromTEMProMatrix (RealODMatrix tempMatrix,
                                                             Zoning zoning)
```

Creates LAD OD matrix from TempPro OD matrix.

**Parameters**

- **tempMatrix** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

**Returns** LAD based OD matrix.

### createTEMProFromLadMatrix

```
public static RealODMatrix createTEMProFromLadMatrix (RealODMatrix ladMatrix, RealODMatrix baseTempPro, Zoning zoning)
```

Creates tempPro OD matrix from LAD OD matrix.

**Parameters**

- **ladMatrix** – LAD to LAD OD matrix.
- **baseTempPro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

**Returns** TEMPro based OD matrix.



### createUnitMatrix

public static *RealODMatrix* **createUnitMatrix** (*List<String> origins*, *List<String> destinations*)

Creates a unit OD matrix for given lists of origin and destination zones.

#### Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *RealODMatrix* **createUnitMatrix** (*List<String> zones*)

Creates a quadratic unit OD matrix for a given lists of zones.

#### Parameters

- **zones** – List of zones.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *RealODMatrix* **createUnitMatrix** (*Set<String> zones*)

Creates a quadratic unit OD matrix for a given lists of zones.

#### Parameters

- **zones** – Set of zones.

**Returns** Unit OD matrix.

### deleteInterzonalFlows

public void **deleteInterzonalFlows** (*String zone*)

Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

#### Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

### getAbsoluteDifference

public double **getAbsoluteDifference** (*RealODMatrix other*)

Gets sum of absolute differences between elements of two matrices.

#### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

## getFlow

public double **getFlow** (*String* originZone, *String* destinationZone)

Gets the flow for a given origin-destination pair.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

## getIntFlow

public int **getIntFlow** (*String* originZone, *String* destinationZone)

Gets the flow for a given origin-destination pair, rounded to a whole number.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

## getKeySet

public *Set*<MultiKey> **getKeySet** ()

Gets the keyset of the multimap.

**Returns** Key set.

## getSortedDestinations

public *List*<*String*> **getSortedDestinations** ()

Gets the sorted list of destinations.

**Returns** List of destinations.

## getSortedOrigins

public *List*<*String*> **getSortedOrigins** ()

Gets the sorted list of origins.

**Returns** List of origins.

## getSumOfFlows

public double **getSumOfFlows** ()

Gets sum of all the flows.

**Returns** Sum of flows.

### getTotalIntFlow

```
public int getTotalIntFlow ()
```

Gets sum of all the (rounded) flows in the matrix.

**Returns** Sum of all the (rounded) flows in the matrix (i.e. number of trips).

### getUnsortedDestinations

```
public List<String> getUnsortedDestinations ()
```

Gets the unsorted list of destinations.

**Returns** List of destinations.

### getUnsortedOrigins

```
public List<String> getUnsortedOrigins ()
```

Gets the unsorted list of origins.

**Returns** List of origins.

### printMatrix

```
public void printMatrix ()
```

Prints the full matrix.

### printMatrixFormatted

```
public void printMatrixFormatted (int precision)
```

Prints the matrix as a formatted table.

**Parameters**

- **precision** – Number of decimal places for the matrix value.

### printMatrixFormatted

```
public void printMatrixFormatted (String s, int precision)
```

Prints message followed by the formatted matrix.

**Parameters**

- **s** – Message.
- **precision** – Number of decimal places.

### roundMatrixValues

```
public void roundMatrixValues ()
```

Rounds OD matrix values.

### saveMatrixFormatted

public void **saveMatrixFormatted** (*String outputFile*)  
Saves the matrix into a csv file.

#### Parameters

- **outputFile** – Path to the output file.

### saveMatrixFormatted2

public void **saveMatrixFormatted2** (*String outputFile*)  
Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

#### Parameters

- **outputFile** – Path to the output file.

### scaleMatrixValue

public void **scaleMatrixValue** (double *factor*)  
Scales matrix values with a scaling factor.

#### Parameters

- **factor** – Scaling factor.

### scaleMatrixValue

public void **scaleMatrixValue** (*RealODMatrix scalingMatrix*)  
Scales matrix values with another matrix (element-wise multiplication).

#### Parameters

- **scalingMatrix** – Scaling matrix.

### setFlow

public void **setFlow** (*String originZone*, *String destinationZone*, double *flow*)  
Sets the flow for a given origin-destination pair.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

### sumMatrixSubset

public double **sumMatrixSubset** (*List<String> origins*, *List<String> destinations*)  
Sums the elements of a matrix subset (provided as two lists of origins and destinations).

#### Parameters

- **origins** – List of origin zones (a subset).
- **destinations** – List of destination zones (a subset).

**Returns** Sum of the subset.

## RealODMatrixTempro

public class **RealODMatrixTempro** implements *AssignableODMatrix*  
Origin-destination matrix with real values, memory use optimised for Tempro.

**Author** Milan Lovric

## Constructors

### RealODMatrixTempro

public **RealODMatrixTempro** (*Zoning* zoning)

Constructor for an empty OD matrix. Uses the maximum Tempro zone ID which will create a rather large matrix.

#### Parameters

- **zoning** – Zoning system.

### RealODMatrixTempro

public **RealODMatrixTempro** (*String* fileName, *Zoning* zoning)

Constructor that reads OD matrix from an input csv file. Can use both matrix and list format.

#### Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### calculateTripEnds

public *HashMap*<*String*, *Integer*> **calculateTripEnds** ()

Calculates the number of trips ending in each destination zone

**Returns** Number of trip ends.

### calculateTripStarts

```
public HashMap<String, Integer> calculateTripStarts ()
```

Calculates the number of trips starting in each origin zone

**Returns** Number of trip starts.

### ceilMatrixValues

```
public void ceilMatrixValues ()
```

Ceil OD matrix values.

### clone

```
public RealODMatrixTempro clone ()
```

### createLadMatrixFromTEMProMatrix

```
public static RealODMatrix createLadMatrixFromTEMProMatrix (RealODMatrixTempro tempromatrix, Zoning zoning)
```

Creates real-valued LAD OD matrix from real-valued TEMPro OD matrix.

**Parameters**

- **tempromatrix** – TEMPro ODMatrix which should be aggregated to LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

**Returns** LAD based real-valued OD matrix.

### createTEMProFromLadMatrix

```
public static RealODMatrixTempro createTEMProFromLadMatrix (RealODMatrix ladODMatrix, RealODMatrixTempro baseTempro, Zoning zoning)
```

Creates tempromatrix OD matrix from LAD OD matrix.

**Parameters**

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempromatrix** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

**Returns** TEMPro based OD matrix.

### createTEMProFromLadMatrix

```
public static RealODMatrixTempro createTEMProFromLadMatrix (ODMatrixMultiKey ladODMatrix, RealODMatrixTempro baseTempromatrix, Zoning zoning)
```

Creates tempromatrix OD matrix from LAD OD matrix.

**Parameters**

- **ladODMatrix** – LAD to LAD OD matrix.
- **baseTempPro** – TEMPro ODMatrix used as weights to disaggregate LAD matrix.
- **zoning** – Zoning system with mapping between TEMPro and LAD zones.

**Returns** TEMPro based OD matrix.

### createUnitMatrix

public static *RealODMatrixTempPro* **createUnitMatrix** (*List<String>* origins, *List<String>* destinations, *Zoning* zoning)

Creates a unit OD matrix for given lists of origin and destination zones.

#### Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **zoning** – Zoning system.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *RealODMatrixTempPro* **createUnitMatrix** (*List<String>* zones, *Zoning* zoning)

Creates a quadratic unit OD matrix for a given lists of zones.

#### Parameters

- **zones** – List of zones.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *RealODMatrixTempPro* **createUnitMatrix** (*Set<String>* zones, *Zoning* zoning)

Creates a quadratic unit OD matrix for a given lists of zones.

#### Parameters

- **zones** – Set of zones.
- **zoning** – Zoning system.

**Returns** Unit OD matrix.

### createUnitMatrix

public static *RealODMatrixTempPro* **createUnitMatrix** (*Zoning* zoning)

Creates a quadratic unit OD matrix.

#### Parameters

- **zoning** – Zoning system.

**Returns** Unit OD matrix.

### deleteInterzonalFlows

public void **deleteInterzonalFlows** (*String zone*)

Deletes all inter-zonal flows to/from a particular zone (leaving only intra-zonal flows)

#### Parameters

- **zone** – Zone for which inter-zonal flows need to be deleted from the origin-destination matrix.

### floorMatrixValues

public void **floorMatrixValues** ()

Floor OD matrix values.

### getAbsoluteDifference

public double **getAbsoluteDifference** (*RealODMatrixTempro other*)

Gets sum of absolute differences between elements of two matrices.

#### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getFlow

public double **getFlow** (*String originZone, String destinationZone*)

Gets the flow for a given origin-destination pair.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.

### getIntFlow

public int **getIntFlow** (*String originZone, String destinationZone*)

Gets the flow for a given origin-destination pair, rounded to a whole number.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination flow.



### getSortedDestinations

```
public List<String> getSortedDestinations ()
```

Gets the sorted list of destinations.

**Returns** List of destinations.

### getSortedOrigins

```
public List<String> getSortedOrigins ()
```

Gets the sorted list of origins.

**Returns** List of origins.

### getSumOfFlows

```
public double getSumOfFlows ()
```

Gets sum of all the flows.

**Returns** Sum of flows.

### getTotalIntFlow

```
public int getTotalIntFlow ()
```

Gets sum of all the (rounded) flows in the matrix.

**Returns** Sum of all the (rounded) flows in the matrix (i.e. number of trips).

### getUnsortedDestinations

```
public List<String> getUnsortedDestinations ()
```

Gets the unsorted list of destinations.

**Returns** List of destinations.

### getUnsortedOrigins

```
public List<String> getUnsortedOrigins ()
```

Gets the unsorted list of origins.

**Returns** List of origins.

### printMatrix

```
public void printMatrix ()
```

Prints the full matrix.

### printMatrixFormatted

public void **printMatrixFormatted** (int *precision*)  
Prints the matrix as a formatted table.

#### Parameters

- **precision** – Number of decimal places for the matrix value.

### printMatrixFormatted

public void **printMatrixFormatted** (String *s*, int *precision*)  
Prints message followed by the formatted matrix.

#### Parameters

- **s** – Message.
- **precision** – Number of decimal places.

### roundMatrixValues

public void **roundMatrixValues** ()  
Rounds OD matrix values.

### saveMatrixFormatted

public void **saveMatrixFormatted** (String *outputFile*)  
Saves the matrix into a csv file. Uses a rectangular/matrix format.

#### Parameters

- **outputFile** – Path to the output file.

### saveMatrixFormatted2

public void **saveMatrixFormatted2** (String *outputFile*)  
Saves the matrix into a csv file. Uses a list format (origin, destination, flow).

#### Parameters

- **outputFile** – Path to the output file.

### saveMatrixFormatted3

public void **saveMatrixFormatted3** (String *outputFile*)  
Saves the matrix into a csv file. Uses a list format (origin, destination, flow) and number codes for zones.

#### Parameters

- **outputFile** – Path to the output file.

### scaleMatrixValue

public void **scaleMatrixValue** (double *factor*)  
Scales matrix values with a scaling factor.

#### Parameters

- **factor** – Scaling factor.

### scaleMatrixValue

public void **scaleMatrixValue** (*RealODMatrixTempo scalingMatrix*)  
Scales matrix values with another matrix (element-wise multiplication).

#### Parameters

- **scalingMatrix** – Scaling matrix.

### setFlow

public void **setFlow** (*String originZone, String destinationZone, double flow*)  
Sets the flow for a given origin-destination pair.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **flow** – Origin-destination flow.

### setFlow

public void **setFlow** (int *originCode, int destinationCode, double flow*)  
Sets the flow for a given origin-destination pair.

#### Parameters

- **originCode** – Origin zone integer code.
- **destinationCode** – Destination zone integer code.
- **flow** – Origin-destination flow.

### sumMatrixSubset

public double **sumMatrixSubset** (*List<String> origins, List<String> destinations*)  
Sums the elements of a matrix subset (provided as two lists of origins and destinations).

#### Parameters

- **origins** – List of origin zones (a subset).
- **destinations** – List of destination zones (a subset).

**Returns** Sum of the subset.

## RebalancedFreightMatrix

public class **RebalancedFreightMatrix** extends *FreightMatrix*

Freight matrix created by directly scaling flows using traffic counts. Base on DfT's BYFM 2006 zoning system (LAD + distribution centres + seaports + airports).

**Author** Milan Lovric

## Constructors

### RebalancedFreightMatrix

public **RebalancedFreightMatrix** (*RoadNetworkAssignment* rna, *RouteSetGenerator* rsg, *Properties* params)

Constructor for a rebalanced freight matrix that uses network assignment and traffic counts for matrix rebalancing.

#### Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **params** – Properties.

### RebalancedFreightMatrix

public **RebalancedFreightMatrix** (*String* fileName, *RoadNetworkAssignment* rna, *RouteSetGenerator* rsg, *Properties* params)

Constructor for a rebalanced freight matrix that uses network assignment and traffic counts for matrix rebalancing.

#### Parameters

- **fileName** – Path to the file with the initial OD matrix.
- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **params** – Properties.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### assignAndCalculateRMSN

public void **assignAndCalculateRMSN** ()  
Assigns OD matrix and calculates RMSN with traffic counts.

### createUnitMatrix

public void **createUnitMatrix** ()  
Creates a unit OD matrix (all ones).

### getRMSNvalues

public [Map](#)<[VehicleType](#), [List](#)<[Double](#)>> **getRMSNvalues** ()  
Gets the list of RMSN values over all performed rebalancing iterations.  
**Returns** List of RMSN values.

### getScalingFactors

public [SkimMatrixFreightArray](#) **getScalingFactors** ()  
Calculates scaling factors for OD pairs.  
**Returns** Scaling factors.

### iterate

public void **iterate** (int *number*)  
Iterates scaling to traffic counts.  
**Parameters**

- **number** – Number of iterations.

### scaleToTrafficCounts

public void **scaleToTrafficCounts** ()  
Scales OD matrix to traffic counts.

### RebalancedODMatrix

public class **RebalancedODMatrix** extends [RealODMatrix](#)  
Origin-destination matrix (LAD based) created by directly scaling flows using traffic counts.  
**Author** Milan Lovric

## Constructors

### RebalancedODMatrix

public **RebalancedODMatrix** (*List<String> origins*, *List<String> destinations*, *RoadNetworkAssignment rna*, *RouteSetGenerator rsg*, *Properties params*)

Constructor for a rebalanced OD matrix that uses network assignment and traffic counts for matrix rebalancing.

#### Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **params** – Properties.

## Methods

### assignAndCalculateRMSN

public void **assignAndCalculateRMSN** ()

Assigns OD matrix and calculates RMSN with traffic counts.

### createUnitMatrix

public void **createUnitMatrix** ()

Creates a unit OD matrix (all ones).

### getRMSNvalues

public *List<Double>* **getRMSNvalues** ()

Gets the list of RMSN values over all performed rebalancing iterations.

**Returns** List of RMSN values.

### getScalingFactors

public *RealODMatrix* **getScalingFactors** ()

Calculates scaling factors for OD pairs.

**Returns** Scaling factors.

### iterate

public void **iterate** (int *number*)

Iterates scaling to traffic counts.

#### Parameters

- **number** – Number of iterations.

## scaleToTrafficCounts

public void **scaleToTrafficCounts** ()  
Scales OD matrix to traffic counts.

## RebalancedTemproODMatrix

public class **RebalancedTemproODMatrix** extends *RealODMatrixTempro*  
Origin-destination matrix (Tempro based) created by directly scaling flows using traffic counts.

**Author** Milan Lovric

## Constructors

### RebalancedTemproODMatrix

public **RebalancedTemproODMatrix** (*List<String> origins, List<String> destinations, RoadNetworkAssignment rna, RouteSetGenerator rsg, Zoning zoning, Properties params*)

Constructor for a rebalanced OD matrix that uses network assignment and traffic counts for matrix rebalancing.

#### Parameters

- **origins** – List of origin zones.
- **destinations** – List of destination zones.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **zoning** – Zoning system.
- **params** – Properties.

### RebalancedTemproODMatrix

public **RebalancedTemproODMatrix** (*String fileName, RoadNetworkAssignment rna, RouteSetGenerator rsg, Zoning zoning, Properties params*)

Constructor for a rebalanced OD matrix that uses network assignment and traffic counts for matrix rebalancing.

#### Parameters

- **fileName** – Path to the file with the initial OD matrix.
- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **zoning** – Zoning system.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### assignAndCalculateRMSN

public void **assignAndCalculateRMSN** ()  
Assigns OD matrix and calculates RMSN with traffic counts.

### createUnitMatrix

public void **createUnitMatrix** ()  
Creates a unit OD matrix (all ones).

### getRMSNvalues

public [List<Double>](#) **getRMSNvalues** ()  
Gets the list of RMSN values over all performed rebalancing iterations.  
**Returns** List of RMSN values.

### getScalingFactors

public [RealODMatrixTempo](#) **getScalingFactors** ()  
Calculates scaling factors for OD pairs.  
**Returns** Scaling factors.

### getSortedDestinations

public [List<String>](#) **getSortedDestinations** ()  
Gets the list of destinations.  
**Returns** List of destinations.

### getSortedOrigins

public [List<String>](#) **getSortedOrigins** ()  
Gets the list of origins.  
**Returns** List of origins.

### iterate

public void **iterate** (int *number*)  
Iterates scaling to traffic counts.

#### Parameters

- **number** – Number of iterations.



## scaleToTrafficCounts

public void **scaleToTrafficCounts** ()  
Scales OD matrix to traffic counts.

## SkimMatrix

public interface **SkimMatrix**  
Skim matrix for storing inter-zonal travel times or costs (for passenger vehicles).  
**Author** Milan Lovric

## Methods

### getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrix other*)  
Gets sum of absolute differences between elements of two matrices.

#### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getCost

public double **getCost** (*String originZone*, *String destinationZone*)  
Gets cost for a given origin-destination pair using ONS codes.

#### Parameters

- **originZone** – Origin zone ONS code.
- **destinationZone** – Destination zone ONS code.

**Returns** Origin-destination cost.

### getCost

public double **getCost** (int *originZoneID*, int *destinationZoneID*)  
Gets cost for a given origin-destination pair using int zone IDs.

#### Parameters

- **originZone** – Origin zone ID.
- **destinationZone** – Destination zone ID.

**Returns** Origin-destination cost.

### getSortedDestinations

public `List<String>` **getSortedDestinations** ()

Gets the sorted list of destinations.

**Returns** List of destination zones.

### getSortedOrigins

public `List<String>` **getSortedOrigins** ()

Gets the sorted list of origins.

**Returns** List of origin zones.

### getUnsortedDestinations

public `List<String>` **getUnsortedDestinations** ()

Gets the unsorted list of destinations.

**Returns** List of destination zones.

### getUnsortedOrigins

public `List<String>` **getUnsortedOrigins** ()

Gets the unsorted list of origins.

**Returns** List of origin zones.

### printMatrix

public void **printMatrix** ()

Prints the matrix.

### printMatrixFormatted

public void **printMatrixFormatted** ()

Prints the matrix as a formatted table.

### printMatrixFormatted

public void **printMatrixFormatted** (`String s`)

Prints the matrix as a formatted table, with a print message.

#### Parameters

- **s** – Print message

### saveMatrixFormatted

public void **saveMatrixFormatted** (*String outputFile*)  
Saves the matrix into a csv file (matrix format).

#### Parameters

- **outputFile** – Path to the output file.

### saveMatrixFormattedList

public void **saveMatrixFormattedList** (*String outputFile*)  
Saves the matrix into a csv file (list format).

#### Parameters

- **outputFile** – Path to the output file.

### setCost

public void **setCost** (*String originZone*, *String destinationZone*, double *cost*)  
Sets cost for a given origin-destination pair using ONS codes.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **cost** – Origin-destination cost.

### setCost

public void **setCost** (int *originZoneID*, int *destinationZoneID*, double *cost*)  
Sets cost for a given origin-destination pair using int zone IDs.

#### Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.
- **cost** – Origin-destination cost.

### SkimMatrixArray

public class **SkimMatrixArray** implements *SkimMatrix*  
Skim matrix for storing inter-zonal travel times or costs (for passenger vehicles).

**Author** Milan Lovric

## Constructors

### SkimMatrixArray

public **SkimMatrixArray** (*Zoning zoning*)

Constructor for an empty skim matrix. Uses the maximum LAD ID.

#### Parameters

- **zoning** – Zoning system.

### SkimMatrixArray

public **SkimMatrixArray** (*String fileName, Zoning zoning*)

Constructor that reads skim matrix from an input csv file. Can use both matrix and list format.

#### Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrix other*)

Gets sum of absolute differences between elements of two matrices.

#### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getAverageCost

public double **getAverageCost** ()

Gets average OD cost.

**Returns** Average cost.

### getAverageCost

public double **getAverageCost** (*ODMatrixMultiKey flows*)

Gets average OD cost weighted by demand.

#### Parameters

- **flows** – The demand as an origin-destination matrix.

**Returns** Average cost.

### getCost

public double **getCost** (*String* originZone, *String* destinationZone)

Gets cost for a given origin-destination pair.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination cost.

### getCost

public double **getCost** (int originZoneID, int destinationZoneID)

Gets cost for a given origin-destination pair.

#### Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.

**Returns** Origin-destination cost.

### getSortedDestinations

public *List*<*String*> **getSortedDestinations** ()

Gets the sorted list of destinations.

**Returns** List of destinations.

### getSortedOrigins

public *List*<*String*> **getSortedOrigins** ()

Gets the sorted list of origins.

**Returns** List of origins.

### getSumOfCosts

public double **getSumOfCosts** ()

Gets sum of OD costs.

**Returns** Sum of costs.

### getSumOfCosts

public double **getSumOfCosts** (*ODMatrixMultiKey* flows)

Gets sum of costs multiplied by demand flows.

#### Parameters

- **flows** – The demand as an origin-destination matrix.

**Returns** Sum of costs.

### getUnsortedDestinations

public *List<String>* **getUnsortedDestinations** ()

Gets the unsorted list of destinations.

**Returns** List of destinations.

### getUnsortedOrigins

public *List<String>* **getUnsortedOrigins** ()

Gets the unsorted list of origins.

**Returns** List of origins.

### printMatrix

public void **printMatrix** ()

Prints the matrix.

### printMatrixFormatted

public void **printMatrixFormatted** ()

Prints the matrix as a formatted table.

### printMatrixFormatted

public void **printMatrixFormatted** (*String* s)

Prints the matrix as a formatted table, with a print message.

#### Parameters

- **s** – Print message

### saveMatrixFormatted

public void **saveMatrixFormatted** (*String* outputFile)

Saves the matrix into a csv file.

#### Parameters

- **outputFile** – Path to the output file.

## saveMatrixFormattedList

public void **saveMatrixFormattedList** (*String* *outputFile*)

Saves the matrix into a csv file. Uses a list format (origin, destination, cost).

### Parameters

- **outputFile** – Path to the output file.

## setCost

public void **setCost** (*String* *originZone*, *String* *destinationZone*, double *cost*)

Sets cost for a given origin-destination pair.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **cost** – Origin-destination cost.

## setCost

public void **setCost** (int *originZoneID*, int *destinationZoneID*, double *cost*)

Sets cost for a given origin-destination pair.

### Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.
- **cost** – Origin-destination cost.

## SkimMatrixArrayTempro

public class **SkimMatrixArrayTempro** implements *SkimMatrix*

Skim matrix for storing inter-zonal travel times or costs (for passenger vehicles).

**Author** Milan Lovric

## Constructors

### SkimMatrixArrayTempro

public **SkimMatrixArrayTempro** (*Zoning* *zoning*)

Constructor for an empty skim matrix. Uses the maximum Tempro ID.

### Parameters

- **zoning** – Zoning system.

## SkimMatrixArrayTempro

public **SkimMatrixArrayTempro** (*String* fileName, *Zoning* zoning)

Constructor that reads skim matrix from an input csv file. Can use both matrix and list format.

### Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrix* other)

Gets sum of absolute differences between elements of two matrices.

### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getAverageCost

public double **getAverageCost** ()

Gets average OD cost.

**Returns** Average cost.

### getAverageCost

public double **getAverageCost** (*ODMatrixMultiKey* flows)

Gets average OD cost weighted by demand.

### Parameters

- **flows** – The demand as an origin-destination matrix.

**Returns** Average cost.

### getCost

public double **getCost** (*String* originZone, *String* destinationZone)

Gets cost for a given origin-destination pair.

### Parameters

- **originZone** – Origin zone.



- **destinationZone** – Destination zone.

**Returns** Origin-destination cost.

## getCost

public double **getCost** (int *originZoneID*, int *destinationZoneID*)  
Gets cost for a given origin-destination pair.

### Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.

**Returns** Origin-destination cost.

## getSortedDestinations

public List<String> **getSortedDestinations** ()  
Gets the sorted list of destinations.

**Returns** List of destinations.

## getSortedOrigins

public List<String> **getSortedOrigins** ()  
Gets the sorted list of origins.

**Returns** List of origins.

## getSumOfCosts

public double **getSumOfCosts** ()  
Gets sum of OD costs.

**Returns** Sum of costs.

## getSumOfCosts

public double **getSumOfCosts** (*ODMatrixMultiKey flows*)  
Gets sum of costs multiplied by demand flows.

### Parameters

- **flows** – The demand as an origin-destination matrix.

**Returns** Sum of costs.

### getUnsortedDestinations

```
public List<String> getUnsortedDestinations ()
```

Gets the unsorted list of destinations.

**Returns** List of destinations.

### getUnsortedOrigins

```
public List<String> getUnsortedOrigins ()
```

Gets the unsorted list of origins.

**Returns** List of origins.

### printMatrix

```
public void printMatrix ()
```

Prints the matrix.

### printMatrixFormatted

```
public void printMatrixFormatted ()
```

Prints the matrix as a formatted table.

### printMatrixFormatted

```
public void printMatrixFormatted (String s)
```

Prints the matrix as a formatted table, with a print message.

#### Parameters

- **s** – Print message

### saveMatrixFormatted

```
public void saveMatrixFormatted (String outputFile)
```

Saves the matrix into a csv file.

#### Parameters

- **outputFile** – Path to the output file.

### saveMatrixFormattedList

```
public void saveMatrixFormattedList (String outputFile)
```

Saves the matrix into a csv file. Uses a list format (origin, destination, cost).

#### Parameters

- **outputFile** – Path to the output file.

## setCost

public void **setCost** (*String originZone*, *String destinationZone*, double *cost*)  
Sets cost for a given origin-destination pair.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **cost** – Origin-destination cost.

## setCost

public void **setCost** (int *originZoneID*, int *destinationZoneID*, double *cost*)  
Sets cost for a given origin-destination pair.

### Parameters

- **originZoneID** – Origin zone ID.
- **destinationZoneID** – Destination zone ID.
- **cost** – Origin-destination cost.

## SkimMatrixFreight

public interface **SkimMatrixFreight**  
Skim matrix for storing inter-zonal travel times or costs (for freight vehicles).

**Author** Milan Lovric

## Fields

### MAX\_FREIGHT\_ZONE\_ID

public static final int **MAX\_FREIGHT\_ZONE\_ID**

### MAX\_VEHICLE\_ID

public static final int **MAX\_VEHICLE\_ID**

## Methods

### getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrixFreight other*)  
Gets sum of absolute differences between elements of two matrices.

### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getAverageCost

public double **getAverageCost** ()  
Gets average cost.

**Returns** Average cost.

### getAverageCost

public double **getAverageCost** (*FreightMatrix* demand)  
Gets average cost weighted by the freight demand.

#### Parameters

- **demand** – Freight OD matrix.

**Returns** Average cost weighted by freight matrix.

### getCost

public double **getCost** (int *originZone*, int *destinationZone*, int *vehicleType*)  
Gets cost for a given origin-destination pair and a vehicle type.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.

**Returns** Origin-destination cost.

### printMatrix

public void **printMatrix** ()  
Prints the matrix.

### printMatrixFormatted

public void **printMatrixFormatted** ()  
Prints the matrix as a formatted table.

### printMatrixFormatted

public void **printMatrixFormatted** (*String* s)  
Prints the matrix as a formatted table, with a print message.

#### Parameters

- **s** – Print message

## saveMatrixFormatted

public void **saveMatrixFormatted** (*String outputFile*)  
Saves the matrix into a csv file (list format for freight).

### Parameters

- **outputFile** – Path to the output file.

## setCost

public void **setCost** (int *originZone*, int *destinationZone*, int *vehicleType*, double *cost*)  
Sets cost for a given origin-destination pair and a vehicle type.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.
- **cost** – Origin-destination cost.

## SkimMatrixFreightArray

public class **SkimMatrixFreightArray** implements *SkimMatrixFreight*  
Skim matrix for storing inter-zonal travel times or costs (for freight vehicles).

**Author** Milan Lovric

## Constructors

### SkimMatrixFreightArray

public **SkimMatrixFreightArray** ()

### SkimMatrixFreightArray

public **SkimMatrixFreightArray** (*String fileName*)  
Constructor that reads freight skim matrix from an input csv file.

### Parameters

- **fileName** – Path to the input file.

### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrixFreight other*)  
Gets sum of absolute differences between elements of two matrices.

#### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getAverageCost

public double **getAverageCost** ()  
Gets average OD cost (ignores empty matrix cells).

**Returns** Average cost.

### getAverageCost

public double **getAverageCost** (*FreightMatrix flows*)  
Gets average OD cost weighted by demand.

#### Parameters

- **flows** – The demand as an origin-destination matrix.

**Returns** Average cost.

### getCost

public double **getCost** (int *originZone*, int *destinationZone*, int *vehicleType*)  
Gets cost for a given origin-destination pair and a vehicle type.

#### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.

**Returns** Origin-destination cost.

### printMatrix

public void **printMatrix** ()  
Prints the matrix.

### printMatrixFormatted

public void **printMatrixFormatted** ()  
Prints the matrix as a formatted table.

## printMatrixFormatted

public void **printMatrixFormatted** ([String](#) s)  
Prints the matrix as a formatted table, with a print message.

### Parameters

- **s** – Print message

## saveMatrixFormatted

public void **saveMatrixFormatted** ([String](#) outputFile)  
Saves the matrix into a csv file.

### Parameters

- **outputFile** – Path to the output file.

## setCost

public void **setCost** (int *originZone*, int *destinationZone*, int *vehicleType*, double *cost*)  
Sets cost for a given origin-destination pair and a vehicle type.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.
- **cost** – Origin-destination cost.

## SkimMatrixFreightMultiKey

public class **SkimMatrixFreightMultiKey** implements [SkimMatrixFreight](#)  
Skim matrix for storing inter-zonal travel times or costs (for freight vehicles).

**Author** Milan Lovric

## Constructors

### SkimMatrixFreightMultiKey

public **SkimMatrixFreightMultiKey** ()

### SkimMatrixFreightMultiKey

public **SkimMatrixFreightMultiKey** ([String](#) fileName)  
Constructor that reads freight skim matrix from an input csv file.

### Parameters

- **fileName** – Path to the input file.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

### Methods

#### getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrixFreight* other)  
Gets sum of absolute differences between elements of two matrices.

##### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

#### getAverageCost

public double **getAverageCost** ()  
Gets average OD cost (ignores empty matrix cells).

**Returns** Average cost.

#### getAverageCost

public double **getAverageCost** (*FreightMatrix* flows)  
Gets average OD cost weighted by demand.

##### Parameters

- **flows** – The demand as an origin-destination matrix.

**Returns** Average cost.

#### getCost

public double **getCost** (int *originZone*, int *destinationZone*, int *vehicleType*)  
Gets cost for a given origin-destination pair and a vehicle type.

##### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.

**Returns** Origin-destination cost.



### getKeySet

```
public Set<MultiKey> getKeySet ()
```

Gets the keyset of the multimap.

**Returns** Keyset.

### printMatrix

```
public void printMatrix ()
```

Prints the matrix.

### printMatrixFormatted

```
public void printMatrixFormatted ()
```

Prints the matrix as a formatted table.

### printMatrixFormatted

```
public void printMatrixFormatted (String s)
```

Prints the matrix as a formatted table, with a print message.

**Parameters**

- **s** – Print message

### saveMatrixFormatted

```
public void saveMatrixFormatted (String outputFile)
```

Saves the matrix into a csv file.

**Parameters**

- **outputFile** – Path to the output file.

### setCost

```
public void setCost (int originZone, int destinationZone, int vehicleType, double cost)
```

Sets cost for a given origin-destination pair and a vehicle type.

**Parameters**

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.
- **vehicleType** – Vehicle type.
- **cost** – Origin-destination cost.

## SkimMatrixMultiKey

public class **SkimMatrixMultiKey** implements *SkimMatrix*  
Skim matrix for storing inter-zonal travel times or costs (for passenger vehicles).

**Author** Milan Lovric

## Constructors

### SkimMatrixMultiKey

public **SkimMatrixMultiKey** (*Zoning* zoning)  
Skim matrix constructors.

#### Parameters

- **zoning** – Zoning system.

### SkimMatrixMultiKey

public **SkimMatrixMultiKey** (*String* fileName, *Zoning* zoning)  
Constructor that reads skim matrix from an input csv file. Can use both matrix and list format.

#### Parameters

- **fileName** – Path to the input file.
- **zoning** – Zoning system.

#### Throws

- **IOException** – if any.
- **FileNotFoundException** – if any.

## Methods

### getAbsoluteDifference

public double **getAbsoluteDifference** (*SkimMatrix* other)  
Gets sum of absolute differences between elements of two matrices.

#### Parameters

- **other** – The other matrix.

**Returns** Sum of absolute differences.

### getAverageCost

public double **getAverageCost** ()  
Gets average OD cost.

**Returns** Average cost.

## getAverageCost

public double **getAverageCost** (*ODMatrixMultiKey* flows)

Gets average OD cost weighted by demand.

### Parameters

- **flows** – The demand as an origin-destination matrix.

**Returns** Average cost.

## getAverageZonalCosts

public *HashMap*<*String*, *Double*> **getAverageZonalCosts** (*List*<*String*> zones)

Gets average zonal cost (used for the rail model).

### Parameters

- **zones** – Zones for which zonal costs are required.

**Returns** Map of average zonal costs.

## getAverageZonalCosts

public *HashMap*<*String*, *Double*> **getAverageZonalCosts** (*List*<*String*> zones, *ODMatrixMultiKey* flows)

Gets average zonal cost weighted by demand (used for the rail model).

### Parameters

- **zones** – Zones for which zonal costs are required.
- **flows** – The demand as an origin-destination matrix.

**Returns** Map of average zonal costs.

## getCost

public double **getCost** (*String* originZone, *String* destinationZone)

Gets cost for a given origin-destination pair.

### Parameters

- **originZone** – Origin zone.
- **destinationZone** – Destination zone.

**Returns** Origin-destination cost.

## getCost

public double **getCost** (int originZoneID, int destinationZoneID)

Gets cost for a given origin-destination pair.

### Parameters

- **originZone** – Origin zone ID.

- **destinationZone** – Destination zone ID.

**Returns** Origin-destination cost.

### getKeySet

public `Set<MultiKey>` **getKeySet** ()

Gets the keyset of the multimap.

**Returns** Keyset.

### getSortedDestinations

public `List<String>` **getSortedDestinations** ()

Gets the sorted list of destinations.

**Returns** List of destinations.

### getSortedOrigins

public `List<String>` **getSortedOrigins** ()

Gets the sorted list of origins.

**Returns** List of origins.

### getSumOfCosts

public double **getSumOfCosts** ()

Gets sum of OD costs.

**Returns** Sum of costs.

### getSumOfCosts

public double **getSumOfCosts** (*ODMatrixMultiKey flows*)

Gets sum of costs multiplied by demand flows.

#### Parameters

- **flows** – The demand as an origin-destination matrix.

**Returns** Sum of costs.

### getUnsortedDestinations

public `List<String>` **getUnsortedDestinations** ()

Gets the unsorted list of destinations.

**Returns** List of destinations.

### getUnsortedOrigins

```
public List<String> getUnsortedOrigins ()
```

Gets the unsorted list of origins.

**Returns** List of origins.

### printMatrix

```
public void printMatrix ()
```

Prints the matrix.

### printMatrixFormatted

```
public void printMatrixFormatted ()
```

Prints the matrix as a formatted table.

### printMatrixFormatted

```
public void printMatrixFormatted (String s)
```

Prints the matrix as a formatted table, with a print message.

**Parameters**

- **s** – Print message

### saveMatrixFormatted

```
public void saveMatrixFormatted (String outputFile)
```

Saves the matrix into a csv file.

**Parameters**

- **outputFile** – Path to the output file.

### saveMatrixFormattedList

```
public void saveMatrixFormattedList (String outputFile)
```

Saves the matrix into a csv file. Uses a list format (origin, destination, cost).

**Parameters**

- **outputFile** – Path to the output file.

### setCost

```
public void setCost (String originZone, String destinationZone, double cost)
```

Sets cost for a given origin-destination pair.

**Parameters**

- **originZone** – Origin zone.

- **destinationZone** – Destination zone.
- **cost** – Origin-destination cost.

### setCost

public void **setCost** (int *originZoneID*, int *destinationZoneID*, double *cost*)  
Sets cost for a given origin-destination pair.

#### Parameters

- **originZone** – Origin zone ID.
- **destinationZone** – Destination zone ID.
- **cost** – Origin-destination cost.

## 1.2.5 nismod.transport.disruption

### Disruption

public abstract class **Disruption**  
Abstract class for a disruption.

**Author** Milan Lovric

### Fields

#### installed

protected boolean **installed**

#### props

protected [Properties](#) **props**

### Constructors

#### Disruption

protected **Disruption** ([Properties](#) *props*)

#### Disruption

protected **Disruption** ([String](#) *fileName*)

## Methods

### getEndYear

```
public int getEndYear ()
```

**Returns** The last year in which intervention still remains installed.

### getProperty

```
public String getProperty (String key)
```

**Parameters**

- **key** – Name of the property

**Returns** Property

### getStartYear

```
public int getStartYear ()
```

**Returns** The year in which intervention is installed.

### getState

```
public boolean getState ()
```

**Returns** The state of the disruption (installed or not).

### install

```
public abstract void install (Object o)
```

### toString

```
public String toString ()
```

### uninstall

```
public abstract void uninstall (Object o)
```

## RoadDisruption

```
public class RoadDisruption extends Disruption
```

Disruption on road links.

**Author** Milan Lovric

## Constructors

### RoadDisruption

public **RoadDisruption** (*Properties props*)  
Constructor.

#### Parameters

- **props** – Properties of the road development intervention.

### RoadDisruption

public **RoadDisruption** (*String fileName*)  
Constructor.

#### Parameters

- **fileName** – File with the properties.

## Methods

### getListOfDisruptedEdgesIDs

public *List*<*Edge*> **getListOfDisruptedEdgesIDs** ()

**Returns** List of disrupted edge IDs.

### getListOfRemovedRoutes

public *List*<*Route*> **getListOfRemovedRoutes** ()

**Returns** List of removed routes

### install

public void **install** (*Object o*)

### uninstall

public void **uninstall** (*Object o*)

## 1.2.6 nismod.transport.network.road

### RoadNetwork

public class **RoadNetwork**  
Routable road network built from the shapefiles.

**Author** Milan Lovric



## Fields

### **averageSpeedFerry**

public double **averageSpeedFerry**

### **freeFlowSpeedARoad**

public double **freeFlowSpeedARoad**

### **freeFlowSpeedMRoad**

public double **freeFlowSpeedMRoad**

### **maximumEdgeID**

public int **maximumEdgeID**

### **maximumNodeID**

public int **maximumNodeID**

### **numberOfLanesARoadCollapsedDualCarriageway**

public int **numberOfLanesARoadCollapsedDualCarriageway**

### **numberOfLanesARoadDualCarriageway**

public int **numberOfLanesARoadDualCarriageway**

### **numberOfLanesARoadRoundabout**

public int **numberOfLanesARoadRoundabout**

### **numberOfLanesARoadSingleCarriageway**

public int **numberOfLanesARoadSingleCarriageway**

### **numberOfLanesARoadSlipRoad**

public int **numberOfLanesARoadSlipRoad**

### numberOfLanesMRoadCollapsedDualCarriageway

```
public int numberOfLanesMRoadCollapsedDualCarriageway
```

### numberOfLanesMRoadDualCarriageway

```
public int numberOfLanesMRoadDualCarriageway
```

### numberOfLanesMRoadSlipRoad

```
public int numberOfLanesMRoadSlipRoad
```

## Constructors

### RoadNetwork

```
public RoadNetwork (URL zonesUrl, URL networkUrl, URL nodesUrl, URL AADFurl, String areaCode-  
FileName, String areaCodeNearestNodeFile, String workplaceZoneFileName, String  
workplaceZoneNearestNodeFile, String freightZoneToLADfile, String freightZoneN-  
earestNodeFile, Properties params)
```

#### Parameters

- **zonesUrl** – Url for the shapefile with zone polygons.
- **networkUrl** – Url for the shapefile with road network.
- **nodesUrl** – Url for the shapefile with nodes.
- **AADFurl** – Url for the shapefile with AADF counts.
- **areaCodeFileName** – Path to the file with census output areas.
- **areaCodeNearestNodeFile** – Path to the file with nearest nodes to output area centroids.
- **workplaceZoneFileName** – Path to the file with workplace zones.
- **workplaceZoneNearestNodeFile** – Path to the file with nearest nodes to workplace zone centroids.
- **freightZoneToLADfile** – Path to the file with freight zone to LAD mapping.
- **freightZoneNearestNodeFile** – Path to the file with nearest nodes to freight zones that are points.
- **params** – Properties with parameters from the config file.

#### Throws

- **IOException** – if any.

## Methods

### addRoadLink

public void **addRoadLink** (Edge *edge*)

This adds edge (including its object) to the network - useful for restoring from a list of removed edges (e.g. during disruption).

#### Parameters

- **edge** – Edge to be added to the network.

### createCustomFeatureType

public static SimpleFeatureType **createCustomFeatureType** (String *linkDataLabel*)

Creates a custom schema for the network.

#### Parameters

- **linkDataLabel** – The label for the link data (e.g. “DayVolume”).

**Returns** SimpleFeature type.

### createNetworkFeatureCollection

public SimpleFeatureCollection **createNetworkFeatureCollection** (Map<Integer, Double> *linkData*, String *linkDataLabel*, String *shapefilePath*)

Creates a custom feature collection for the network.

#### Parameters

- **linkData** – Data assigned to network links.
- **linkDataLabel** – The label of the link data.
- **shapefilePath** – The path to the shapefile into which data will be stored.

#### Throws

- **IOException** – if any.

**Returns** Feature collection.

### createNewRoadLink

public Edge **createNewRoadLink** (Node *fromNode*, Node *toNode*, int *numberOfLanes*, char *roadCategory*, double *length*, int *edgeID*)

Creates a new (unidirectional) road link (edge) between existing intersections (nodes).

#### Parameters

- **fromNode** – Start node of the new road link.
- **toNode** – End node of the new road link.
- **numberOfLanes** – Number of lanes in the road link.
- **roadCategory** – Road category.

- **length** – Length of the road link.

**Returns** Newly created edge.

## exportToShapefile

public void **exportToShapefile** (*String fileName*)

Exports a directed multigraph representation of the network as a shapefile.

### Parameters

- **fileName** – The name of the output shapefile.

### Throws

- **IOException** – if any.

## getAADFCarTrafficCounts

public *Integer*[] **getAADFCarTrafficCounts** ()

Get car traffic counts data for each link (for combined counts return 1/2 of the count per direction).

**Returns** AADF traffic counts per link.

## getAADFFreightTrafficCounts

public *Map*<*VehicleType*, *Integer*[]> **getAADFFreightTrafficCounts** ()

Get car traffic counts data for each link (for combined counts return 1/2 of the count per direction).

**Returns** AADF traffic counts per freight vehicle type and per link.

## getAADFShapefile

public ShapefileDataStore **getAADFShapefile** ()

## getAreaCodeToNearestNodeID

public *HashMap*<*String*, *Integer*> **getAreaCodeToNearestNodeID** ()

Getter method for the area code to the nearest node mapping.

**Returns** Area code to the nearest node mapping.

## getAreaCodeToPopulation

public *HashMap*<*String*, *Integer*> **getAreaCodeToPopulation** ()

Getter method for the area code to population mapping.

**Returns** Area code to population mapping.

### getAstarFunctions

public AStarIterator.AStarFunctions **getAstarFunctions** (Node *destinationNode*)  
Getter method for the AStar functions (edge cost and heuristic function) based on distance.

**Parameters**

- **destinationNode** – Destination node.

**Returns** AStar functions.

### getAstarFunctionsTime

public AStarIterator.AStarFunctions **getAstarFunctionsTime** (Node *destinationNode*, double[] *link-TravelTime*)  
Getter method for the AStar functions (edge cost and heuristic function) based on travel time.

**Parameters**

- **destinationNode** – Destination node.
- **linkTravelTime** – Link travel times to use for edge weighting.

**Returns** AStar functions.

### getAverageAccessEgressDistance

public double **getAverageAccessEgressDistance** (int *node*)  
Average access/egress distance to access a node that has gravitating population.

**Parameters**

- **node** – Node to which

**Returns** Gravitating population.

### getAverageAccessEgressDistanceFreight

public double **getAverageAccessEgressDistanceFreight** (int *node*)  
Average access/egress distance to access a node that has gravitating population.

**Parameters**

- **node** – Node to which

**Returns** Gravitating population.

### getAverageSpeedFerry

public double **getAverageSpeedFerry** ()

### getDijkstraTimeWeighter

public DijkstraIterator.EdgeWeighter **getDijkstraTimeWeighter** (double[] *linkTravelTime*)  
Getter method for the Dijkstra edge weighter with time.

#### Parameters

- **linkTravelTime** – Link travel times to use for edge weighting.

**Returns** Dijkstra edge weighter with time.

### getDijkstraWeighter

public DijkstraIterator.EdgeWeighter **getDijkstraWeighter** ()  
Getter method for the Dijkstra edge weighter.

**Returns** Dijkstra edge weighter.

### getEdgeIDtoEdge

public Edge[] **getEdgeIDtoEdge** ()  
Getter method for edgeID to edge mapping.

**Returns** Edge ID to edge mapping.

### getEdgeIDtoOtherDirectionEdgeID

public Integer[] **getEdgeIDtoOtherDirectionEdgeID** ()  
Getter method for edgeID to other direction edgeID mapping.

**Returns** Edge ID to other direction edge ID mapping.

### getEdgeLength

public double **getEdgeLength** (int *edgeID*)  
Gets edge length for a given edge ID.

#### Parameters

- **edgeID** – Edge ID.

**Returns** Edge length.

### getEdgeToZone

public HashMap<Integer, String> **getEdgeToZone** ()  
Getter method for the edge to zone mapping.

**Returns** Node to zone mapping.

### getEdgesType

public *EdgeType*[] **getEdgesType** ()

Getter method for the array saying if the edge is A-road, motorway, or ferry. Array index is edge ID (without -1 shift).

**Returns** Map between the edge ID and whether the edge is ferry.

### getEndNodeBlacklist

public boolean[] **getEndNodeBlacklist** ()

### getFastestPath

public *RoadPath* **getFastestPath** (DirectedNode *from*, DirectedNode *to*, double[] *linkTravelTime*)

Gets the fastest path between two nodes using astar algorithm and provided link travel times. Links which have no travel time provided will use free flow travel times.

**Parameters**

- **from** – Origin node.
- **to** – Destination node.
- **linkTravelTime** – The map with link travel times.

**Returns** Fastest path.

### getFastestPathDijkstra

public *RoadPath* **getFastestPathDijkstra** (DirectedNode *from*, DirectedNode *to*, double[] *linkTravelTime*)

Gets the fastest path between two nodes using Dijkstra's algorithm and provided link travel times. Links which have no travel time provided will use free flow travel times.

**Parameters**

- **from** – Origin node.
- **to** – Destination node.
- **linkTravelTime** – The map with link travel times.

**Returns** Fastest path.

### getFreeFlowSpeedARoad

public double **getFreeFlowSpeedARoad** ()

### getFreeFlowSpeedMRoad

public double **getFreeFlowSpeedMRoad** ()

### getFreeFlowTravelTime

```
public double[] getFreeFlowTravelTime ()
```

Getter method for free flow travel time.

**Returns** Free flow travel time.

### getFreightZoneToLAD

```
public HashMap<Integer, String> getFreightZoneToLAD ()
```

Getter method for the freight zone to LAD mapping.

**Returns** Area code to the nearest node mapping.

### getFreightZoneToNearestNode

```
public HashMap<Integer, Integer> getFreightZoneToNearestNode ()
```

Getter method for the freight zone to the nearest node mapping.

**Returns** Area code to the nearest node mapping.

### getGravitatingPopulation

```
public int getGravitatingPopulation (int node)
```

Population gravitating to a node.

#### Parameters

- **node** – Node to which the population gravitates.

**Returns** Gravitating population.

### getGravitatingWorkplacePopulation

```
public int getGravitatingWorkplacePopulation (int node)
```

Workplace population gravitating to a node.

#### Parameters

- **node** – Node to which the workplace population gravitates.

**Returns** Gravitating workplace population.

### getIsEdgeUrban

```
public Boolean[] getIsEdgeUrban ()
```

Getter method for the array saying if the edge is urban (true), rural (false), or unkown (null). Array index is edge ID (without -1 shift).

**Returns** Array saying if the edge is urban/rural/unkown.



### **getMaximumEdgeID**

public int **getMaximumEdgeID** ()  
    Getter method for maximum edge ID.  
    **Returns** Maximum edge ID.

### **getMaximumNodeID**

public int **getMaximumNodeID** ()  
    Getter method for maximum node ID.  
    **Returns** Maximum node ID.

### **getNetwork**

public DirectedGraph **getNetwork** ()  
    Getter method for the road network.  
    **Returns** Directed graph representation of the road network.

### **getNetworkShapefile**

public ShapefileDataStore **getNetworkShapefile** ()

### **getNewNetworkShapefile**

public ShapefileDataStore **getNewNetworkShapefile** ()

### **getNodeIDtoNode**

public Node[] **getNodeIDtoNode** ()  
    Getter method for nodeID to node mapping.  
    **Returns** Node ID to node mapping.

### **getNodeToAverageAccessEgressDistance**

public double[] **getNodeToAverageAccessEgressDistance** ()  
    Getter method for the node to average access/egress distance mapping [in metres].  
    **Returns** Node ID to average access/egress distance mapping.

### **getNodeToAverageAccessEgressDistanceFreight**

public double[] **getNodeToAverageAccessEgressDistanceFreight** ()  
    Getter method for the node to average access/egress distance mapping for freight [in metres].  
    **Returns** Node ID to average access/egress distance mapping for freight.

### **getNodeToGravitatingPopulation**

public int[] **getNodeToGravitatingPopulation** ()  
    Getter method for the node to gravitating population mapping.  
    **Returns** Node to gravitating population mapping.

### **getNodeToZone**

public [HashMap](#)<[Integer](#), [String](#)> **getNodeToZone** ()  
    Getter method for the node to zone mapping.  
    **Returns** Node to zone mapping.

### **getNodesShapefile**

public ShapefileDataStore **getNodesShapefile** ()

### **getNumberOfLanes**

public int[] **getNumberOfLanes** ()  
    Getter method for the number of lanes for each link.  
    **Returns** Link id to number of lanes mapping.

### **getNumberOfLanesARoad**

public int **getNumberOfLanesARoad** ([String](#) wayType)

### **getNumberOfLanesMRoad**

public int **getNumberOfLanesMRoad** ([String](#) wayType)

### **getStartNodeBlacklist**

public boolean[] **getStartNodeBlacklist** ()

### **getWorkplaceCodeToPopulation**

public [HashMap](#)<[String](#), [Integer](#)> **getWorkplaceCodeToPopulation** ()  
    Getter method for the workplace zone to population mapping.  
    **Returns** Workplace zone to population mapping.

### getWorkplaceZoneToNearestNode

```
public HashMap<String, Integer> getWorkplaceZoneToNearestNode ()
```

Getter method for the workplace code to the nearest node mapping.

**Returns** Workplace code to the nearest node mapping.

### getZoneToAreaCodes

```
public HashMap<String, List<String>> getZoneToAreaCodes ()
```

Getter method for the zone to area codes mapping.

**Returns** Zone to area codes mapping.

### getZoneToNodes

```
public HashMap<String, List<Integer>> getZoneToNodes ()
```

Getter method for the zone to nodes mapping.

**Returns** Zone to nodes mapping.

### getZoneToWorkplaceCodes

```
public HashMap<String, List<String>> getZoneToWorkplaceCodes ()
```

Getter method for the LAD zone to workplace zones mapping.

**Returns** Zone to workplace code mapping.

### getZonesShapefile

```
public ShapefileDataStore getZonesShapefile ()
```

### isBlacklistedAsEndNode

```
public boolean isBlacklistedAsEndNode (int nodeId)
```

Finds out if the node is blacklisted as a path end node.

**Parameters**

- **nodeId** – Node ID.

**Returns** Whether nodes is blacklisted or not.

### isBlacklistedAsStartNode

```
public boolean isBlacklistedAsStartNode (int nodeId)
```

Finds out if the node is blacklisted as a path start node.

**Parameters**

- **nodeId** – Node ID.

**Returns** Whether nodes is blacklisted or not.

### makeEdgesAdmissible

public void **makeEdgesAdmissible** ()

Overrides actual edge lengths with straight line distances, when they are smaller than straight line distances.

### removeRoadLink

public void **removeRoadLink** (Edge *edge*)

Removes an edge from the road network.

#### Parameters

- **edge** – Edge to remove from the road network.

### replaceNetworkEdgeIDs

public void **replaceNetworkEdgeIDs** ([URL](#) *networkShapeFile*)

Replaces edge IDs in the road network object with fixed edge IDs provided in a shapefile.

#### Parameters

- **networkShapeFile** – Path to the shapefile with the network with edge IDs.

#### Throws

- **IOException** – if any.

### sortGravityNodes

public void **sortGravityNodes** ()

For each zone (LAD) sorts the list of contained nodes based on the gravitating population.

### sortGravityNodesFreight

public void **sortGravityNodesFreight** ()

For each zone (LAD) sorts the list of contained nodes based on the gravitating workplace population.

### toString

public [String](#) **toString** ()

### RoadNetwork.EdgeType

public static enum **EdgeType**

### Enum Constants

#### AROAD

public static final [RoadNetwork.EdgeType](#) **AROAD**

## FERRY

public static final *RoadNetwork.EdgeType* **FERRY**

## MOTORWAY

public static final *RoadNetwork.EdgeType* **MOTORWAY**

## RoadNetworkAssignment

public class **RoadNetworkAssignment**  
Network assignment of origin-destination flows.  
**Author** Milan Lovric

### Fields

#### alpha

public final double **alpha**

#### assignmentFraction

public final double **assignmentFraction**

#### averageAccessEgressSpeedCar

public final double **averageAccessEgressSpeedCar**

#### averageAccessEgressSpeedFreight

public final double **averageAccessEgressSpeedFreight**

#### averageIntersectionDelay

public final double **averageIntersectionDelay**

#### baseYear

public final int **baseYear**

#### betaARoad

public final double **betaARoad**

### **betaMRoad**

public final double **betaMRoad**

### **flagAStarIfEmptyRouteSet**

public final boolean **flagAStarIfEmptyRouteSet**

### **flagIncludeAccessEgress**

public final boolean **flagIncludeAccessEgress**

### **flagIntrazonalAssignmentReplacement**

public final boolean **flagIntrazonalAssignmentReplacement**

### **flagUseRouteChoiceModel**

public final boolean **flagUseRouteChoiceModel**

### **interzonalTopNodes**

public final int **interzonalTopNodes**

### **maximumCapacityARoad**

public final int **maximumCapacityARoad**

### **maximumCapacityMRoad**

public final int **maximumCapacityMRoad**

### **nodesProbabilityWeighting**

public final double **nodesProbabilityWeighting**

### **nodesProbabilityWeightingFreight**

public final double **nodesProbabilityWeightingFreight**

### **peakHourPercentage**

public final double **peakHourPercentage**

## topTemproNodes

public final int **topTemproNodes**

## volumeToFlowFactor

public final double **volumeToFlowFactor**

## Constructors

### RoadNetworkAssignment

```
public RoadNetworkAssignment (RoadNetwork roadNetwork, Zoning zoning, Map<EnergyType, Double> energyUnitCosts, Map<EnergyType, Double> unitCO2Emissions, Map<VehicleType, Map<EngineType, Double>> engineTypeFractions, Map<VehicleType, Double> fractionsAV, Map<VehicleType, Double> vehicleTypeToPCU, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParams, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiencies, Map<TimeOfDay, Double> timeOfDayDistribution, Map<VehicleType, Map<TimeOfDay, Double>> timeOfDayDistributionFreight, Map<TimeOfDay, Map<Integer, Double>> defaultLinkTravelTime, HashMap<String, Double> areaCodeProbabilities, HashMap<String, Double> workplaceZoneProbabilities, List<PricingPolicy> congestionCharges, Properties params)
```

#### Parameters

- **roadNetwork** – Road network.
- **zoning** – Zoning system.
- **energyUnitCosts** – Energy unit costs.
- **unitCO2Emissions** – Unit CO2 emissions.
- **engineTypeFractions** – Market shares of different engine/fuel types.
- **fractionsAV** – Fraction of autonomous vehicles for different vehicle types.
- **vehicleTypeToPCU** – Vehicle to PCU conversion.
- **energyConsumptionParams** – Base fuel consumption rates.
- **relativeFuelEfficiencies** – Relative fuel efficiencies (compared to base year).
- **timeOfDayDistribution** – Time of day distribution.
- **timeOfDayDistributionFreight** – Time of day distribution for freight.
- **defaultLinkTravelTime** – Default link travel times.
- **areaCodeProbabilities** – Probabilities of trips starting/ending in each census output area.
- **workplaceZoneProbabilities** – Probabilities of freight trips starting/ending in each census output area.
- **congestionCharges** – Congestion charges.

- **params** – Assignment parameters.

## Methods

### assignFlowsAndUpdateLinkTravelTimes

```
public void assignFlowsAndUpdateLinkTravelTimes (AssignableODMatrix passengerODM,  
                                                FreightMatrix freightODM, RouteSetGenerator rsg, Properties props, double weight)
```

Assigns passenger and freight origin-destination matrix to the road network using the fastest path based on the current values in the linkTravelTime field. Finally, updates link travel times using weighted averaging.

#### Parameters

- **passengerODM** – Passenger origin-destination matrix.
- **freightODM** – Freight origin-destination matrix.
- **rsg** – Route set generator to store fastest routes generated during the assignment (but could be pregenerated too).
- **props** – Parameters from the config file.
- **weight** – Weighting parameter.

### assignFlowsAndUpdateLinkTravelTimes

```
public void assignFlowsAndUpdateLinkTravelTimes (AssignableODMatrix passengerODM,  
                                                FreightMatrix freightODM, RouteSetGenerator rsg, Zoning zoning, Properties params,  
                                                double weight)
```

Assigns passenger and freight origin-destination matrix to the road network using specification in the config file. Finally, updates link travel times using weighted averaging.

#### Parameters

- **passengerODM** – Passenger origin-destination matrix.
- **freightODM** – Freight origin-destination matrix.
- **rsg** – Route set generator object with routes to be used for the assignment (if route choice used) or an object in which to store routes (if routing used).
- **zoning** – Zoning system (necessary for ‘tempro’ and ‘combined’ assignment types).
- **params** – Parameters from the config file.
- **weight** – Weighting parameter.

### assignFlowsAndUpdateLinkTravelTimesIterated

```
public void assignFlowsAndUpdateLinkTravelTimesIterated (AssignableODMatrix passengerODM,  
                                                         FreightMatrix freightODM, RouteSetGenerator rsg, Properties props, double weight, int iterations)
```

Iterates assignment and travel time update a fixed number of times.



**Parameters**

- **passengerODM** – Passenger origin-destination matrix.
- **freightODM** – Freight origin-destination matrix.
- **rsg** – Route set generator object with routes to be used for the assignment (if route choice used) or an object in which to store routes (if routing used).
- **props** – Properties.
- **weight** – Weighting parameter.
- **iterations** – Number of iterations.

**assignFlowsAndUpdateLinkTravelTimesIterated**

```
public void assignFlowsAndUpdateLinkTravelTimesIterated (AssignableODMatrix passengerODM, FreightMatrix freightODM, RouteSetGenerator rsg, Zoning zoning, Properties params, double weight, int iterations)
```

Iterates assignment and travel time update a fixed number of times.

**Parameters**

- **passengerODM** – Passenger origin-destination matrix.
- **freightODM** – Freight origin-destination matrix.
- **rsg** – Route set generator object with routes to be used for the assignment (if route choice used) or an object in which to store routes (if routing used).
- **zoning** – Zoning system (necessary for ‘tempro’ and ‘combined’ assignment types).
- **params** – Parameters from the config file.
- **weight** – Weighting parameter.
- **iterations** – Number of iterations.

**assignFreightFlowsHourlyRouting**

```
public void assignFreightFlowsHourlyRouting (FreightMatrix freightMatrix, Map<TimeOfDay, RouteSetGenerator> routeStorage, Properties props)
```

Assigns freight origin-destination matrix to the road network using A-star routing. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

**Parameters**

- **freightMatrix** – Freight origin-destination matrix.
- **routeStorage** – Route storage (stores fastest routes separately for each hour of the day).
- **props** – Properties.

**assignFreightFlowsRouteChoice**

public void **assignFreightFlowsRouteChoice** (*FreightMatrix* freightMatrix, *RouteSetGenerator* rsg, *Properties* routeChoiceParameters)

Assigns freight origin-destination matrix to the road network using a route choice model and pre-generated routes. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

**Parameters**

- **freightMatrix** – Freight origin-destination matrix.
- **rsg** – Route set generator containing the routes.
- **routeChoiceParameters** – Route choice parameters.

**assignFreightFlowsRouting**

public void **assignFreightFlowsRouting** (*FreightMatrix* freightMatrix, *RouteSetGenerator* rsg, *Properties* props)

Assigns freight origin-destination matrix to the road network using A-star routing. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

**Parameters**

- **freightMatrix** – Freight origin-destination matrix.
- **rsg** – Route storage (reduces the number of routing calls).
- **props** – Properties.

## assignPassengerFlowsHourlyRouting

```
public void assignPassengerFlowsHourlyRouting (AssignableODMatrix passengerODM,
                                              Map<TimeOfDay, RouteSetGenerator>
                                              routeStorage, Properties props)
```

Assigns passenger origin-destination matrix to the road network using A-star routing algorithm. Calculates the fastest path based on the current values in the `linkTravelTimePerTimeOfDay` instance field, which means different fastest routes may be used in different hours of the day.

### Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **routeStorage** – Stores routes for each hour of the day separately.
- **props** – Properties.

## assignPassengerFlowsRouteChoice

```
public void assignPassengerFlowsRouteChoice (AssignableODMatrix passengerODM, RouteSet-
                                              Generator rsg, Properties routeChoiceParameters)
```

Assigns passenger origin-destination matrix to the road network. Uses the route choice and pre-generated paths.

### Parameters

- **passengerODM** – Passenger origin-destination matrix.
- **rsg** – Route set generator containing the routes.
- **routeChoiceParameters** – Route choice parameters.

## assignPassengerFlowsRouteChoiceTempro

```
public void assignPassengerFlowsRouteChoiceTempro (AssignableODMatrix passengerODM,
                                                    Zoning zoning, RouteSetGenerator rsg,
                                                    Properties routeChoiceParameters)
```

Assigns passenger origin-destination matrix to the road network using the Tempro zoning system. Uses the route choice and pre-generated paths.

### Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **zoning** – Contains Tempro zone information.
- **rsg** – Route set generator containing the routes.
- **routeChoiceParameters** – Route choice parameters.

## assignPassengerFlowsRouteChoiceTemproDistanceBased

```
public void assignPassengerFlowsRouteChoiceTemproDistanceBased (AssignableODMatrix
                                                                    passengerODM,
                                                                    Zoning zoning, Route-
                                                                    SetGenerator rsg,
                                                                    Properties routeChoi-
                                                                    ceParameters)
```

Assigns passenger origin-destination matrix to the road network using the combined Tempro/LAD zoning sys-

tem. When Temprow zones a farther than a distance threshold, it seeks for the nodes within LAD zones that have a route set. Uses the route choice and pre-generated paths (after a distance threshold, there will be less inter-zonal routes).

#### Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **zoning** – Contains Temprow zone information.
- **rsg** – Route set generator containing the routes.
- **routeChoiceParameters** – Route choice parameters.

### assignPassengerFlowsRouting

public void **assignPassengerFlowsRouting** (*AssignableODMatrix* passengerODM, *RouteSetGenerator* rsg, *Properties* props)

Assigns passenger origin-destination matrix to the road network using A-star routing algorithm. Calculates the fastest path based on the current values in the linkTravelTimePerTimeOfDay instance field, however only one route will be used for the same OD pair (the route that was calculated first).

#### Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **rsg** – To store routes during the assignment (reduces the number of routing calls).
- **props** – Routing parameters.

### assignPassengerFlowsTemprow

public void **assignPassengerFlowsTemprow** (*AssignableODMatrix* passengerODM, *Zoning* zoning, *RouteSetGenerator* rsg, *Properties* props)

Assigns passenger origin-destination matrix to the road network using the Temprow zoning system. Calculates the fastest path based on the current values in the linkTravelTime instance field.

#### Parameters

- **passengerODM** – Passenger origin-destination matrix with flows to be assigned.
- **zoning** – Contains Temprow zone information.
- **rsg** – Route set (here new routes will be stored).
- **props** – Properties.

### calculateAbsDifferenceCarCounts

public *HashMap*<*Integer*, *Integer*> **calculateAbsDifferenceCarCounts** ()

Calculates absolute differences between car volumes and traffic counts. For combined counts, takes the average of two absolute differences.

**Returns** Absolute differences between car volumes and traffic counts.

### calculateAssignedFreightMatrix

public *FreightMatrix* **calculateAssignedFreightMatrix** ()

Calculate freight OD matrix from trip list.

**Returns** Freight matrix.

### calculateAssignedODMatrix

public *ODMatrixMultiKey* **calculateAssignedODMatrix** ()

Calculate assigned OD matrix from trip list.

**Returns** ODMatrixMultiKey OD matrix.

### calculateCO2Emissions

public *HashMap<String, Double>* **calculateCO2Emissions** ()

Calculates total CO2 emissions (in kg) for each type of passenger and freight vehicle.

**Returns** Total consumption for each engine type.

### calculateCarEnergyConsumptions

public *Map<EnergyType, Double>* **calculateCarEnergyConsumptions** ()

Calculates total energy consumption for each car/AV energy type (in litres for fuels and in kWh for electricity).

**Returns** Total consumption for each energy type.

### calculateCostSkimMatrix

public *SkimMatrix* **calculateCostSkimMatrix** ()

Calculates cost skim matrix (zone-to-zone financial costs).

**Returns** Inter-zonal skim matrix (cost).

### calculateCostSkimMatrixFreight

public *SkimMatrixFreight* **calculateCostSkimMatrixFreight** ()

Calculates cost skim matrix (zone-to-zone financial costs) for freight.

**Returns** Inter-zonal skim matrix (cost).

### calculateDifferenceCarCounts

public *HashMap<Integer, Integer>* **calculateDifferenceCarCounts** ()

Calculates differences between car volumes and traffic counts. For combined counts, takes the average of the two differences.

**Returns** Differences between car volumes and traffic counts.

### calculateDirectionAveragedAbsoluteDifferenceCarCounts

public [HashMap](#)<[Integer](#), [Double](#)> **calculateDirectionAveragedAbsoluteDifferenceCarCounts** ()  
Calculates absolute differences between car volumes and traffic counts averaged for both directions. For combined counts, takes the average of two absolute differences.

**Returns** Direction averaged absolute differences between car volumes and traffic counts.

### calculateDirectionAveragedPeakLinkCapacityUtilisation

public double[] **calculateDirectionAveragedPeakLinkCapacityUtilisation** ()  
Calculate peak-hour link capacity utilisation (%) averaged by two directions.

**Returns** Peak-hour link capacity utilisation.

### calculateDistanceSkimMatrix

public [SkimMatrix](#) **calculateDistanceSkimMatrix** ()  
Updates cost skim matrix (zone-to-zone distances).

**Returns** Inter-zonal skim matrix (distance).

### calculateDistanceSkimMatrixFreight

public [SkimMatrixFreight](#) **calculateDistanceSkimMatrixFreight** ()  
Updates cost skim matrix (zone-to-zone distances) for freight.

**Returns** Inter-zonal skim matrix (distance).

### calculateDistanceSkimMatrixTempro

public [SkimMatrix](#) **calculateDistanceSkimMatrixTempro** ()  
Updates cost skim matrix (zone-to-zone distances).

**Parameters**

- **zoning** – Zoning system.

**Returns** Inter-zonal skim matrix (distance).

### calculateEnergyConsumptions

public [Map](#)<[EnergyType](#), [Double](#)> **calculateEnergyConsumptions** ()  
Calculates total energy consumption for each energy type of passenger cars and freight vehicles (in litres for fuels and in kWh for electricity).

**Returns** Total consumption for each engine type.

### calculateEnergyConsumptionsPerVehicleType

public `Map<VehicleType, Map<EnergyType, Double>>` **calculateEnergyConsumptionsPerVehicleType** ()  
 Calculates total energy consumptions per vehicle type.

**Returns** Total consumption for each energy type.

### calculateFreightEnergyConsumptions

public `Map<EnergyType, Double>` **calculateFreightEnergyConsumptions** ()  
 Calculates total energy consumption for each freight vehicle engine type (in litres for fuels and in kWh for electricity).

**Returns** Total consumption for each energy type.

### calculateFreightLADTripEnds

public `HashMap<String, Integer>` **calculateFreightLADTripEnds** ()  
 Calculates the number of freight trips ending in a LAD.

**Returns** Number of trips.

### calculateFreightLADTripStarts

public `HashMap<String, Integer>` **calculateFreightLADTripStarts** ()  
 Calculates the number of freight trips starting in a LAD.

**Returns** Number of trips.

### calculateGEHStatisticForCarCounts

public `Double[]` **calculateGEHStatisticForCarCounts** (double *volumeToFlowFactor*)  
 Calculates GEH statistic for simulated and observed hourly car flows. For combined counts, combines the volumes on two road directions. To obtain hourly flows, multiplies daily link volumes (and traffic counts) with volumeToFlowFactor. The formula is taken from WebTAG Unit M3.1.

#### Parameters

- **volumeToFlowFactor** – Converts daily vehicle volume to hourly flow (e.g. 0.1 for peak flow; 1/24.0 for daily average)

**Returns** GEH statistic for simulated and observed hourly car flows.

### calculateGEHStatisticForFreightCounts

public `Map<VehicleType, Double[]>` **calculateGEHStatisticForFreightCounts** (double *volumeToFlowFactor*)

Calculates GEH statistic for simulated and observed hourly freight vehicle flows. For combined counts, combines the volumes on two road directions. To obtain hourly flows, multiplies daily link volumes (and traffic counts) with volumeToFlowFactor. The formula is taken from WebTAG Unit M3.1.

#### Parameters

- **volumeToFlowFactor** – Converts daily vehicle volume to hourly flow (e.g. 0.1 for peak flow; 1/24.0 for daily average)

**Returns** GEH statistic for simulated and observed hourly freight vehicle flows, per vehicle type.

### calculateGEHStatisticPerTimeOfDay

public `Double[]` **calculateGEHStatisticPerTimeOfDay** (*TimeOfDay* hour)

Calculates GEH statistic for simulated and observed hourly flow. It uses `linkVolumesInPCUPerTimeOfDay`, so make sure only car flows have been assigned. For combined counts, takes the average of the two differences. The formula is taken from WebTAG Unit M3.1.

#### Parameters

- **hour** – Hour for which to calculate GEH statistics.

**Returns** GEH statistic for simulated and observed hourly car flows.

### calculateLADTripEnds

public `HashMap<String, Integer>` **calculateLADTripEnds** ()

Calculates the number of passenger (car/AV) trips ending in a LAD.

**Returns** Number of trips.

### calculateLADTripStarts

public `HashMap<String, Integer>` **calculateLADTripStarts** ()

Calculates the number of passenger (car/AV) trips starting in a LAD.

**Returns** Number of trips.

### calculateLinkVolumeInPCU

public `double[]` **calculateLinkVolumeInPCU** (*List<Trip>* tripList)

Calculates daily link volumes in PCU.

#### Parameters

- **tripList** – Trip list.

**Returns** Map of link volumes in PCU.

### calculateLinkVolumeInPCUPerTimeOfDay

public `Map<TimeOfDay, double[]>` **calculateLinkVolumeInPCUPerTimeOfDay** (*List<Trip>* tripList)

Calculates link volumes in PCU per time of day.

#### Parameters

- **tripList** – Trip list.

**Returns** Link volumes in PCU per time of day



### calculateLinkVolumePerVehicleType

```
public Map<VehicleType, int[]> calculateLinkVolumePerVehicleType (List<Trip> tripList)
```

Calculates daily link volumes per vehicle type.

#### Parameters

- **tripList** – Trip list.

**Returns** Map of link volumes per vehicle type.

### calculateMADforExpandedSimulatedVolumes

```
public double calculateMADforExpandedSimulatedVolumes (double expansionFactor)
```

Calculate prediction error (mean absolute deviation for expanded simulated volumes and observed traffic counts).

#### Parameters

- **expansionFactor** – Expansion factor expands simulated volumes.

**Returns** Mean absolute deviation.

### calculateODCarEnergyConsumptions

```
public Map<EnergyType, SkimMatrix> calculateODCarEnergyConsumptions ()
```

Calculates origin-destination energy consumption for car vehicles for each energy type (in litres/kg for fuels and in kWh for electricity).

**Returns** Zonal consumption for each energy type.

### calculatePeakLinkCapacityUtilisation

```
public double[] calculatePeakLinkCapacityUtilisation ()
```

Calculate peak-hour link capacity utilisation (capacity / max. capacity).

**Returns** Peak-hour link capacity utilisation [%].

### calculatePeakLinkDensities

```
public double[] calculatePeakLinkDensities ()
```

Calculate peak-hour link densities (PCU/lane/km/hr).

**Returns** Peak-hour link densities.

### calculatePeakLinkPointCapacities

```
public double[] calculatePeakLinkPointCapacities ()
```

Calculate peak-hour link point capacities (PCU/lane/hr).

**Returns** Peak-hour link point capacities.

### calculateRMSNforExpandedSimulatedVolumes

public double **calculateRMSNforExpandedSimulatedVolumes** (double *expansionFactor*)  
Calculate prediction error (RMSN for expanded simulated volumes and observed traffic counts).

**Parameters**

- **expansionFactor** – Expansion factor expands simulated volumes.

**Returns** Normalised root mean square error.

### calculateRMSNforFreightCounts

public Map<*VehicleType*, Double> **calculateRMSNforFreightCounts** ()  
Calculate prediction error (RMSN for simulated freight volumes and observed traffic counts).

**Returns** Normalised root mean square errors for each freight vehicle separately.

### calculateRMSNforSimulatedVolumes

public double **calculateRMSNforSimulatedVolumes** ()  
Calculate prediction error (RMSN for simulated volumes and observed traffic counts).

**Returns** Normalised root mean square error.

### calculateTimeSkimMatrix

public *SkimMatrix* **calculateTimeSkimMatrix** ()  
Calculated travel time skim matrix (zone-to-zone travel times).

**Returns** Inter-zonal skim matrix (time).

### calculateTimeSkimMatrixFreight

public *SkimMatrixFreight* **calculateTimeSkimMatrixFreight** ()  
Calculated travel time skim matrix (zone-to-zone travel times) for freight.

**Returns** Inter-zonal skim matrix (time).

### calculateZonalCarEnergyConsumptions

public Map<*EnergyType*, HashMap<String, Double>> **calculateZonalCarEnergyConsumptions** (double  
*orig-*  
*in-*  
*ZoneEn-*  
*er-*  
*gy-*  
*Weight*)

Calculates spatial energy consumption for car vehicles for each energy type (in litres/kg for fuels and in kWh for electricity).

**Parameters**

- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).

**Returns** Zonal consumption for each energy type.

### calculateZonalTemporalTripStartsForElectricVehicles

```
public HashMap<String, Map<TimeOfDay, Integer>> calculateZonalTemporalTripStartsForElectricVehicles (VehicleType vht)
```

Calculates the number of electric vehicles (BEV, PHEV) of a given type (CAR, VAN, RIGID, ARTIC), starting in each LAD in each hour.

#### Parameters

- **vht** – Vehicle type (calculation will include the autonomous version of the same vehicle type too).

**Returns** Number of trips.

### calculateZonalTemporalTripStartsForHydrogenVehicles

```
public HashMap<String, Map<TimeOfDay, Integer>> calculateZonalTemporalTripStartsForHydrogenVehicles (VehicleType vht)
```

Calculates the number of hydrogen vehicles (ICE\_H2, FCEV\_H2) of a given type (CAR, VAN, RIGID, ARTIC), starting in each LAD in each hour.

#### Parameters

- **vht** – Vehicle type (calculation will include the autonomous version of the same vehicle type too).

**Returns** Number of trips.

### calculateZonalTemporalVehicleElectricityConsumptions

```
public HashMap<String, Map<TimeOfDay, Double>> calculateZonalTemporalVehicleElectricityConsumptions (VehicleType vht)
```

Calculates zonal (per LAD) and temporal (per hour) electricity consumption for a given vehicle type (in kWh).

#### Parameters

- **vht** – Vehicle type (e.g., if CAR provided, CAR\_AV consumption will be added too).
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).

**Returns** Electricity consumption per zone and time of day.

## calculateZonalTemporalVehicleHydrogenConsumptions

```
public HashMap<String, Map<TimeOfDay, Double>> calculateZonalTemporalVehicleHydrogenConsumptions (VehicleType vht,
```

```
double originZoneEnergyWeight)
```

Calculates zonal (per LAD) and temporal (per hour) hydrogen consumption for a given vehicle type (in kg).

### Parameters

- **vht** – Vehicle type (e.g., if CAR provided, CAR\_AV consumption will be added too).
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).

**Returns** Electricity consumption per zone and time of day.

## calculateZonalVehicleCO2Emissions

```
public Map<VehicleType, HashMap<String, Double>> calculateZonalVehicleCO2Emissions (double originZoneEnergyWeight)
```

Calculates zonal CO2 emissions (in kg) for each vehicle type (sum across all engine types).

### Parameters

- **originZoneEnergyWeight** – Percentage of CO2 emission assigned to origin zone (the rest assigned to destination zone).

**Returns** Zonal CO2 emissions for each vehicle type.

## calculateZonalVehicleKilometresPerVehicleType

```
public Map<String, Map<VehicleType, Double>> calculateZonalVehicleKilometresPerVehicleType ()
```

Calculates vehicle kilometres in each LAD and for each vehicle type. Ignores access and egress to major roads. Ignores minor roads.

**Returns** Vehicle kilometres.

### calculateZonalVehicleKilometresPerVehicleTypeFromTemproTripList

```
public Map<String, Map<VehicleType, Double>> calculateZonalVehicleKilometresPerVehicleTypeFromTemproTripList
```

Calculates vehicle kilometres in each LAD using Tempro-based trips. Optionally includes access and egress (for Tempro-based model). Optionally includes minor trips (Tempro intra-zonal).

#### Parameters

- **includeAccessEgress** – True if access and egress should be included in the calculation.
- **includeMinorTrips** – True if minor trips should be included in the calculation.

**Returns** Vehicle kilometres.

### calculateZonalVehicleKilometresPerVehicleTypeFromTripList

```
public Map<String, Map<VehicleType, Double>> calculateZonalVehicleKilometresPerVehicleTypeFromTripList
```

Calculates vehicle kilometres in each LAD and per vehicle type. Optionally includes access and egress (for LAD-based model).

#### Parameters

- **includeAccessEgress** – True if access and egress should be included in the calculation.

**Returns** Vehicle kilometres.

### getAADFCarTrafficCounts

```
public Integer[] getAADFCarTrafficCounts ()
```

Getter method for AADF car traffic counts.

**Returns** Car traffic counts.

### getAADFFreightTrafficCounts

```
public Map<VehicleType, Integer[]> getAADFFreightTrafficCounts ()
```

Getter method for AADF freight traffic counts.

**Returns** Freight traffic counts.

### getAreaCodeProbabilities

```
public HashMap<String, Double> getAreaCodeProbabilities ()
```

Getter method for output area probabilities.

**Returns** Output area probabilities.

### getCopyOfLinkTravelTimes

```
public Map<TimeOfDay, double[]> getCopyOfLinkTravelTimes ()
```

**Returns** The copy of all link travel times.

### getCopyOfLinkTravelTimesAsMap

```
public Map<TimeOfDay, Map<Integer, Double>> getCopyOfLinkTravelTimesAsMap ()
```

**Returns** The copy of all link travel times as map.

### getEndNodeProbabilities

```
public HashMap<Integer, Double> getEndNodeProbabilities ()
```

Getter method for node probabilities.

**Returns** Node probabilities.

### getEnergyConsumptionParameters

```
public Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> getEnergyConsumptionParameters ()
```

Getter method for energy consumption WebTAG parameters.

**Returns** Energy consumption parameters.

### getEnergyUnitCosts

```
public Map<EnergyType, Double> getEnergyUnitCosts ()
```

Getter method for energy unit costs.

**Returns** Energy unit costs.

### getEngineTypeFractions

```
public Map<VehicleType, Map<EngineType, Double>> getEngineTypeFractions ()
```

Getter method for engine type fractions.

**Returns** Engine type fractions.

### getFlagUseRouteChoiceModel

public boolean **getFlagUseRouteChoiceModel** ()  
 Getter method for the use route choice model flag.

**Returns** Flag.

### getLinkFreeFlowTravelTimes

public double[] **getLinkFreeFlowTravelTimes** ()  
 Getter method for the link free-flow travel times.

**Returns** Link volumes

### getLinkTravelTimes

public Map<TimeOfDay, double[]> **getLinkTravelTimes** ()  
 Getter method for the link travel times per time of day.

**Returns** Link travel times per time of day.

### getLinkVolumeInPCU

public double[] **getLinkVolumeInPCU** ()  
 Getter method for daily link volumes in PCU.

**Returns** Link volumes in PCU.

### getLinkVolumeInPCUPerTimeOfDay

public Map<TimeOfDay, double[]> **getLinkVolumeInPCUPerTimeOfDay** ()  
 Getter method for link volumes in PCU per time of day.

**Returns** Link volumes in PCU per time of day.

### getLinkVolumePerVehicleType

public Map<VehicleType, int[]> **getLinkVolumePerVehicleType** ()  
 Getter method for daily link volumes per vehicle type.

**Returns** Link volumes in PCU per time of day.

### getObservedTripLengthDistribution

public double[] **getObservedTripLengthDistribution** (double[] *binLimitsInKm*, boolean *flagIncludeAccessEgress*, boolean *flagIncludeMinorTrips*)  
 Calculates observed trip length distribution.

**Parameters**

- **binLimitsInKm** – Bin limits in kilometres.

- **flagIncludeAccessEgress** – If true include access and egress into trip distance calculation. \* @param flagIncludeMinorTrips If true include minor road trips into trip distance calculation.

**Returns** Observed trip length distribution.

### getObservedTripLengthFrequencies

```
public double[] getObservedTripLengthFrequencies (double[] binLimitsInKm, boolean flagIncludeAccessEgress, boolean flagIncludeMinorTrips)
```

Calculates observed trip length frequencies.

#### Parameters

- **binLimitsInKm** – Bin limits in kilometres.
- **flagIncludeAccessEgress** – If true include access and egress into trip distance calculation.
- **flagIncludeMinorTrips** – If true include minor road trips into trip distance calculation.

**Returns** Observed trip length distribution.

### getRoadNetwork

```
public RoadNetwork getRoadNetwork ()
```

Getter method for the road network.

**Returns** Road network.

### getStartNodeProbabilities

```
public HashMap<Integer, Double> getStartNodeProbabilities ()
```

Getter method for node probabilities.

**Returns** Node probabilities.

### getTripList

```
public ArrayList<Trip> getTripList ()
```

Getter method for the trip list.

**Returns** Trip list.

### getVolumeToFlowFactor

```
public double getVolumeToFlowFactor ()
```

Getter method for the volume to flow factor.

**Returns** Volume to flow factor.



### getWorkplaceZoneProbabilities

public `HashMap<String, Double>` **getWorkplaceZoneProbabilities** ()

Getter method for workplace zones probabilities.

**Returns** Workplace zones probabilities.

### initialiseTripList

public void **initialiseTripList** (int *initialCapacity*)

Initialise trip list for passengers and freight (e.g. expected total sum of passenger and freight flows).

#### Parameters

- **initialCapacity** – Initial capacity of the trip list.

### loadLinkTravelTimes

public void **loadLinkTravelTimes** (int *year*, `String` *fileName*)

Loads link travel times from a file.

#### Parameters

- **year** – Year of the assignment.
- **fileName** – Input file name (with path).

### printGEHStatistic

public void **printGEHStatistic** ()

Prints GEH statistics for comparison between simulated and observed hourly car flows.

### printGEHStatistic

public void **printGEHStatistic** (double *volumeToFlowFactor*)

Prints GEH statistics for comparison between simulated and observed hourly car flows.

#### Parameters

- **volumeToFlowFactor** – Converts daily vehicle volume to hourly flow (e.g. 0.1 for peak flow; 1/24.0 for daily average)

### printGEHStatisticFreight

public void **printGEHStatisticFreight** ()

Prints GEH statistics for comparison between simulated and observed hourly freight vehicle flows.

### printGEHStatisticFreight

public void **printGEHStatisticFreight** (double *volumeToFlowFactor*)

Prints GEH statistics for comparison between simulated and observed hourly freight vehicle flows.

#### Parameters

- **volumeToFlowFactor** – Converts daily vehicle volume to hourly flow (e.g. 0.1 for peak flow; 1/24.0 for daily average)

### printHourlyGEHstatistic

public void **printHourlyGEHstatistic** ()

Prints GEH statistics for comparison between simulated and observed hourly car flows.

### printRMSNstatistic

public void **printRMSNstatistic** ()

Prints RMSN statistic for comparison between simulated daily car volumes and observed daily traffic counts.

### printRMSNstatisticFreight

public void **printRMSNstatisticFreight** ()

Prints RMSN statistic for comparison between simulated daily freight volumes and observed daily freight traffic counts.

### resetLinkVolumes

public void **resetLinkVolumes** ()

Resets link volumes to zero.

### resetTripList

public void **resetTripList** ()

Reset trip list for passengers and freight.

### saveAssignmentResults

public void **saveAssignmentResults** (int year, String outputFile)

Saves assignment results to output file.

#### Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

### saveEnergyConsumptionsPerVehicleType

public void **saveEnergyConsumptionsPerVehicleType** (int year, String outputFile)

Saves energy consumptions per vehicle type to an output file.

#### Parameters

- **year** – Assignment year.

- **outputFile** – Output file name (with path).

### saveHourlyCarVolumes

public void **saveHourlyCarVolumes** (int *year*, *String outputFile*)

Saves hourly car volumes to output file.

#### Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

### saveLinkTravelTimes

public void **saveLinkTravelTimes** (int *year*, *String outputFile*)

Saves travel times into a file.

#### Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

### saveOriginDestinationCarElectricityConsumption

public void **saveOriginDestinationCarElectricityConsumption** (*String outputFile*)

Saves origin-destination matrix of car electricity consumption.

#### Parameters

- **outputFile** – Output file name (with path).

### savePeakLinkPointCapacities

public void **savePeakLinkPointCapacities** (int *year*, *String outputFile*)

Saves peak link point capacities into a file.

#### Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

### saveTotalCO2Emissions

public void **saveTotalCO2Emissions** (int *year*, *String outputFile*)

Saves total CO2 emissions to an output file.

#### Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

### saveTotalEnergyConsumptions

public void **saveTotalEnergyConsumptions** (int *year*, *String* *outputFile*)

Saves total electricity consumption to an output file.

#### Parameters

- **year** – Year of the assignment.
- **outputFile** – Output file name (with path).

### saveZonalCarEnergyConsumptions

public void **saveZonalCarEnergyConsumptions** (int *year*, double *originZoneEnergyWeight*, *String* *outputFile*)

Saves zonal car energy consumptions to an output file.

#### Parameters

- **year** – Assignment year.
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).
- **outputFile** – Output file name (with path).

### saveZonalTemporalTripStartsForEVs

public void **saveZonalTemporalTripStartsForEVs** (int *year*, *VehicleType* *vht*, *String* *outputFile*)

Saves zonal (LAD) and temporal (hourly) number of EV trips to an output file.

#### Parameters

- **year** – Assignment year.
- **vht** – Vehicle Type.
- **outputFile** – Output file name (with path).

### saveZonalTemporalTripStartsForH2

public void **saveZonalTemporalTripStartsForH2** (int *year*, *VehicleType* *vht*, *String* *outputFile*)

Saves zonal (LAD) and temporal (hourly) number of H2 fuelled trips to an output file.

#### Parameters

- **year** – Assignment year.
- **vht** – Vehicle Type.
- **outputFile** – Output file name (with path).

### saveZonalTemporalVehicleElectricity

public void **saveZonalTemporalVehicleElectricity** (int *year*, *VehicleType* *vht*, double *originZoneEnergyWeight*, *String* *outputFile*)

Saves zonal (LAD) and temporal (hourly) vehicle electricity consumptions to an output file.

**Parameters**

- **year** – Assignment year.
- **vht** – Vehicle Type (it will include consumption of autonomous vehicles too).
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).
- **outputFile** – Output file name (with path).

**saveZonalTemporalVehicleHydrogen**

public void **saveZonalTemporalVehicleHydrogen** (int *year*, *VehicleType* *vht*, double *originZoneEnergyWeight*, *String* *outputFile*)

Saves zonal (LAD) and temporal (hourly) vehicle hydrogen consumptions to an output file.

**Parameters**

- **year** – Assignment year.
- **vht** – Vehicle Type (it will include consumption of autonomous vehicles too).
- **originZoneEnergyWeight** – Percentage of energy consumption assigned to origin zone (the rest assigned to destination zone).
- **outputFile** – Output file name (with path).

**saveZonalVehicleCO2Emissions**

public void **saveZonalVehicleCO2Emissions** (int *year*, double *originZoneEnergyWeight*, *String* *outputFile*)

Saves zonal vehicle CO2 emissions to an output file.

**Parameters**

- **year** – Assignment year.
- **originZoneEnergyWeight** – Percentage of CO2 emission assigned to origin zone (the rest assigned to destination zone).
- **outputFile** – Output file name (with path).

**saveZonalVehicleKilometres**

public void **saveZonalVehicleKilometres** (int *year*, *String* *outputFile*)

Saves zonal vehicle-kilometres.

**Parameters**

- **year** – Assignment year.
- **outputFile** – Output file name (with path).

**saveZonalVehicleKilometresWithAccessEgress**

public void **saveZonalVehicleKilometresWithAccessEgress** (int *year*, *String* *outputFile*)

Saves zonal vehicle-kilometres that include access/egress and minor trips

**Parameters**

- **year** – Assignment year.
- **outputFile** – Output file name (with path).

**setElectricityUnitCost**

public void **setElectricityUnitCost** (double *electricityUnitCost*)  
Setter method for the electricity unit cost.

**Parameters**

- **electricityUnitCost** – The cost of 1 kWh in £.

**setEndNodeProbabilities**

public void **setEndNodeProbabilities** (HashMap<Integer, Double> *endNodeProbabilities*)  
Setter method for node probabilities.

**Parameters**

- **endNodeProbabilities** – Node probabilities.

**setEnergyConsumptionParameters**

public void **setEnergyConsumptionParameters** (*VehicleType* vehicleType, *EngineType* engineType,  
Map<WebTAG, Double> parameters)  
Setter method for the energy consumption parameters.

**Parameters**

- **vehicleType** – Vehicle type
- **engineType** – Engine type
- **parameters** – Energy consumptions parameters (A, B, C, D)

**setEnergyUnitCost**

public void **setEnergyUnitCost** (*EnergyType* energyType, double *energyUnitCost*)  
Setter method for the energy unit cost.

**Parameters**

- **energyType** – The type of a car engine.
- **energyUnitCost** – The cost of 1 L (of fuel) or 1 kWh (of electricity) in £.

**setEngineTypeFractions**

public void **setEngineTypeFractions** (*VehicleType* vht, Map<*EngineType*, Double> *engineTypeFractions*)  
Setter method for energy type fractions.

**Parameters**

- **vht** – Vehicle type
- **engineTypeFractions** – Map with engine type fractions.

### setStartNodeProbabilities

public void **setStartNodeProbabilities** (*HashMap<Integer, Double> startNodeProbabilities*)  
Setter method for node probabilities.

#### Parameters

- **startNodeProbabilities** – Node probabilities.

### updateCostSkimMatrix

public void **updateCostSkimMatrix** (*SkimMatrix costSkimMatrix*)  
Updates cost skim matrix (zone-to-zone financial costs).

#### Parameters

- **costSkimMatrix** – Inter-zonal skim matrix (cost).

### updateCostSkimMatrixFreight

public void **updateCostSkimMatrixFreight** (*SkimMatrixFreight costSkimMatrixFreight*)  
Updates cost skim matrix (zone-to-zone financial costs) for freight.

#### Parameters

- **costSkimMatrixFreight** – Inter-zonal skim matrix (cost) for freight.

### updateLinkTravelTimes

public void **updateLinkTravelTimes** ()  
Updates link travel times per time of day.

### updateLinkTravelTimes

public void **updateLinkTravelTimes** (double *weight*)  
Updates link travel times using weighted averaging between new values (calculated from link volumes) and older values (stored in the instance field).

#### Parameters

- **weight** – Parameter for weighted averaging.

### updateLinkVolumeInPCU

public void **updateLinkVolumeInPCU** ()  
Updates daily link volumes in PCU from the trip list and stores it into instance variable.

### updateLinkVolumeInPCUPerTimeOfDay

public void **updateLinkVolumeInPCUPerTimeOfDay** ()  
Updates link volumes in PCU per time of day from object's trip list and stores into instance variable.

### updateLinkVolumePerVehicleType

public void **updateLinkVolumePerVehicleType** ()  
Updates daily link volumes per vehicle type from trip list and stores into instance variable.

### updateTimeSkimMatrix

public void **updateTimeSkimMatrix** (*SkimMatrix* timeSkimMatrix)  
Updates travel time skim matrix (zone-to-zone travel times).

#### Parameters

- **timeSkimMatrix** – Inter-zonal skim matrix (time).

### updateTimeSkimMatrixFreight

public void **updateTimeSkimMatrixFreight** (*SkimMatrixFreight* timeSkimMatrixFreight)  
Updates travel time skim matrix (zone-to-zone travel times) for freight.

#### Parameters

- **timeSkimMatrixFreight** – Inter-zonal skim matrix (time).

### RoadNetworkAssignment.EnergyType

public static enum **EnergyType**

#### Enum Constants

##### CNG

public static final *RoadNetworkAssignment.EnergyType* **CNG**

##### DIESEL

public static final *RoadNetworkAssignment.EnergyType* **DIESEL**

##### ELECTRICITY

public static final *RoadNetworkAssignment.EnergyType* **ELECTRICITY**



## HYDROGEN

public static final *RoadNetworkAssignment.EnergyType* **HYDROGEN**

## LPG

public static final *RoadNetworkAssignment.EnergyType* **LPG**

## PETROL

public static final *RoadNetworkAssignment.EnergyType* **PETROL**

## RoadNetworkAssignment.EngineType

public static enum **EngineType**

### Enum Constants

## BEV

public static final *RoadNetworkAssignment.EngineType* **BEV**

## FCEV\_H2

public static final *RoadNetworkAssignment.EngineType* **FCEV\_H2**

## HEV\_DIESEL

public static final *RoadNetworkAssignment.EngineType* **HEV\_DIESEL**

## HEV\_PETROL

public static final *RoadNetworkAssignment.EngineType* **HEV\_PETROL**

## ICE\_CNG

public static final *RoadNetworkAssignment.EngineType* **ICE\_CNG**

## ICE\_DIESEL

public static final *RoadNetworkAssignment.EngineType* **ICE\_DIESEL**

## ICE\_H2

public static final *RoadNetworkAssignment.EngineType* **ICE\_H2**

## ICE\_LPG

public static final *RoadNetworkAssignment.EngineType* **ICE\_LPG**

## ICE\_PETROL

public static final *RoadNetworkAssignment.EngineType* **ICE\_PETROL**

## PHEV\_DIESEL

public static final *RoadNetworkAssignment.EngineType* **PHEV\_DIESEL**

## PHEV\_PETROL

public static final *RoadNetworkAssignment.EngineType* **PHEV\_PETROL**

## RoadNetworkAssignment.TimeOfDay

public static enum **TimeOfDay**

## Enum Constants

### EIGHTAM

public static final *RoadNetworkAssignment.TimeOfDay* **EIGHTAM**

### EIGHTPM

public static final *RoadNetworkAssignment.TimeOfDay* **EIGHTPM**

### ELEVENAM

public static final *RoadNetworkAssignment.TimeOfDay* **ELEVENAM**

### ELEVENPM

public static final *RoadNetworkAssignment.TimeOfDay* **ELEVENPM**

**FIVEAM**

public static final *RoadNetworkAssignment.TimeOfDay* **FIVEAM**

**FIVEPM**

public static final *RoadNetworkAssignment.TimeOfDay* **FIVEPM**

**FOURAM**

public static final *RoadNetworkAssignment.TimeOfDay* **FOURAM**

**FOURPM**

public static final *RoadNetworkAssignment.TimeOfDay* **FOURPM**

**MIDNIGHT**

public static final *RoadNetworkAssignment.TimeOfDay* **MIDNIGHT**

**NINEAM**

public static final *RoadNetworkAssignment.TimeOfDay* **NINEAM**

**NINEPM**

public static final *RoadNetworkAssignment.TimeOfDay* **NINEPM**

**NOON**

public static final *RoadNetworkAssignment.TimeOfDay* **NOON**

**ONEAM**

public static final *RoadNetworkAssignment.TimeOfDay* **ONEAM**

**ONEPM**

public static final *RoadNetworkAssignment.TimeOfDay* **ONEPM**

**SEVENAM**

public static final *RoadNetworkAssignment.TimeOfDay* **SEVENAM**

## SEVENPM

public static final *RoadNetworkAssignment.TimeOfDay* **SEVENPM**

## SIXAM

public static final *RoadNetworkAssignment.TimeOfDay* **SIXAM**

## SIXPM

public static final *RoadNetworkAssignment.TimeOfDay* **SIXPM**

## TENAM

public static final *RoadNetworkAssignment.TimeOfDay* **TENAM**

## TENPM

public static final *RoadNetworkAssignment.TimeOfDay* **TENPM**

## THREEAM

public static final *RoadNetworkAssignment.TimeOfDay* **THREEAM**

## THREEPM

public static final *RoadNetworkAssignment.TimeOfDay* **THREEPM**

## TWOAM

public static final *RoadNetworkAssignment.TimeOfDay* **TWOAM**

## TWOPM

public static final *RoadNetworkAssignment.TimeOfDay* **TWOPM**

## RoadNetworkAssignment.VehicleType

public static enum **VehicleType**

## Enum Constants

### ARTIC

public static final *RoadNetworkAssignment.VehicleType* **ARTIC**

### ARTIC\_AV

public static final *RoadNetworkAssignment.VehicleType* **ARTIC\_AV**

### CAR

public static final *RoadNetworkAssignment.VehicleType* **CAR**

### CAR\_AV

public static final *RoadNetworkAssignment.VehicleType* **CAR\_AV**

### RIGID

public static final *RoadNetworkAssignment.VehicleType* **RIGID**

### RIGID\_AV

public static final *RoadNetworkAssignment.VehicleType* **RIGID\_AV**

### VAN

public static final *RoadNetworkAssignment.VehicleType* **VAN**

### VAN\_AV

public static final *RoadNetworkAssignment.VehicleType* **VAN\_AV**

## Fields

### value

int **value**

## Methods

### getValue

```
public int getValue ()
```

## RoadPath

```
public class RoadPath extends Path
    Directed path (a list of directed nodes).
```

**Author** Milan Lovric

## Constructors

### RoadPath

```
public RoadPath ()
```

### RoadPath

```
public RoadPath (Collection nodes)
```

## Methods

### buildEdges

```
protected List buildEdges ()
    Internal method for building the edge set of the walk. This method calculated the edges upon every call.
    Returns The list of edges for the walk, or null if the edge set could not be calculated due to an
        invalid walk.
```

### isValid

```
public boolean isValid ()
```

## Route

```
public class Route
    Route is a sequence of directed edges with a choice utility.
```

**Author** Milan Lovric

## Constructors

### Route

public **Route** (*RoadNetwork* roadNetwork)

### Route

public **Route** (*RoadPath* path, *RoadNetwork* roadNetwork)

Constructor from a given path.

#### Parameters

- **path** – A path from which to construct a route.
- **roadNetwork** – Road network.

## Methods

### addEdge

public boolean **addEdge** (DirectedEdge *edge*)

Adds a directed edge to the end of the current route.

#### Parameters

- **edge** – Directed edge to be added.

**Returns** true if edge addition was successful, false otherwise.

### addEdgeWithoutValidityCheck

public void **addEdgeWithoutValidityCheck** (DirectedEdge *edge*)

Adds a directed edge to the end of the current route.

#### Parameters

- **edge** – Directed edge to be added.

### addEdgeWithoutValidityCheck

public void **addEdgeWithoutValidityCheck** (int *edgeID*)

Adds a directed edge to the end of the current route.

#### Parameters

- **edgeID** – Directed edge to be added.

## calculateConsumption

```
public Map<EnergyType, Double> calculateConsumption (VehicleType vht, EngineType et, double[] linkTravelTime, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>>> relativeFuelEfficiency)
```

Calculates energy consumption of the route.

### Parameters

- **vht** – Vehicle type.
- **et** – Energy type.
- **linkTravelTime** – Link travel time.
- **energyConsumptionParameters** – Base year energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency compared to base year.

**Returns** Consumption for each type.

## calculateCost

```
public void calculateCost (VehicleType vht, EngineType et, TimeOfDay tod, double[] linkTravelTime, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>>> relativeFuelEfficiency, Map<EnergyType, Double> energyUnitCosts, List<PricingPolicy> congestionCharges)
```

Calculates the cost of the route.

### Parameters

- **vht** – Vehicle type.
- **et** – Engine type.
- **tod** – Time of day.
- **linkTravelTime** – Link travel times.
- **energyConsumptionParameters** – Base year energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency (compared to base year).
- **energyUnitCosts** – Energy unit costs.
- **congestionCharges** – Congestion charges.

## calculateLength

```
public void calculateLength ()
```

Calculates the length of the route.



## calculateTravelTime

```
public void calculateTravelTime (double[] linkTravelTime, double avgIntersectionDelay)
```

Calculates the route travel time based on link travel times.

### Parameters

- **linkTravelTime** – Link travel times.
- **avgIntersectionDelay** – Average intersection delay (in minutes).

## calculateUtility

```
public void calculateUtility (VehicleType vht, EngineType et, TimeOfDay tod, double[] linkTravelTime, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, Map<EnergyType, Double> energyUnitCosts, List<PricingPolicy> congestionCharges, Map<RouteChoiceParams, Double> params)
```

Calculates the utility of the route.

### Parameters

- **vht** – Vehicle type.
- **et** – Engine type.
- **tod** – Time of day.
- **linkTravelTime** – Link travel times.
- **energyConsumptionParameters** – Energy consumption parameters (A, B, C, D) for a combination of vehicle type and engine type.
- **relativeFuelEfficiency** – Relative fuel efficiency compared to the base year.
- **energyUnitCosts** – Energy unit costs.
- **congestionCharges** – Congestion charges.
- **params** – Route choice parameters.

## contains

```
public boolean contains (Edge edge)
```

Checks if route contains the edge.

### Parameters

- **edge** – Edge object.

**Returns** True if route contains the edge.

## contains

```
public boolean contains (int edgeID)
```

Checks if route contains the edge.

### Parameters

- **edgeID** – Edge id.

**Returns** True if route contains the edge.

## **equals**

public boolean **equals** (*Object obj*)

## **getCost**

public double **getCost** ()  
Getter method for route cost.

**Returns** Route cost.

## **getDestinationNode**

public DirectedNode **getDestinationNode** ()  
Getter method for destination node.

**Returns** Destination node.

## **getEdges**

public TIntArrayList **getEdges** ()  
Getter method for the list of edges.

**Returns** List of edge IDs.

## **getFormattedString**

public *String* **getFormattedString** ()  
Gets formatted string representation of the route.

**Returns** Route as a string.

## **getFormattedStringEdgeIDsOnly**

public *String* **getFormattedStringEdgeIDsOnly** ()  
Gets formatted string representation of the route using edge IDs only.

**Returns** Route as a string.

## **getLength**

public double **getLength** ()  
Getter method for route length.

**Returns** Route length.

### getNumberOfIntersections

public int **getNumberOfIntersections** ()  
Getter method for number of intersections.

**Returns** Number of intersections.

### getOriginNode

public DirectedNode **getOriginNode** ()  
Getter method for route origin node.

**Returns** Origin node.

### getRoadNetwork

public *RoadNetwork* **getRoadNetwork** ()  
Getter method for the road network.

**Returns** Road network.

### getTime

public double **getTime** ()  
Getter method for route time.

**Returns** Route time.

### getUtility

public double **getUtility** ()  
Getter method for route utility.

**Returns** Route utility.

### hashCode

public int **hashCode** ()

### isEmpty

public boolean **isEmpty** ()  
Checks if route is empty or not.

**Returns** True if route is empty.

## isValid

public boolean **isValid** ()

Checks if route is valid (successive edges in the route are connected in a directional way).

**Returns** True if route is valid.

## setUtility

public void **setUtility** (double *utility*)

Setter method for route utility.

### Parameters

- **utility** – Route utility.

## toString

public **String** **toString** ()

## trimToSize

public void **trimToSize** ()

Trims edges list to size and calculate length (onetime operation).

## Route.WebTAG

public static enum **WebTAG**

### Enum Constants

#### A

public static final *Route.WebTAG* **A**

#### B

public static final *Route.WebTAG* **B**

#### C

public static final *Route.WebTAG* **C**

#### D

public static final *Route.WebTAG* **D**

## RouteSet

public class **RouteSet**

RouteSet is a choice set of possible routes between an origin and a destination node.

**Author** Milan Lovric

## Constructors

### RouteSet

public **RouteSet** (*RoadNetwork* roadNetwork)

Constructor.

#### Parameters

- **roadNetwork** – Road network.

## Methods

### addRoute

public void **addRoute** (*Route* route)

Adds a route to the choice set.

#### Parameters

- **route** – Route to be added.

### addRouteWithoutAnyChecks

public void **addRouteWithoutAnyChecks** (*Route* route)

Adds a route to the choice set.

#### Parameters

- **route** – Route to be added.

### addRouteWithoutValidityAndEndNodesCheck

public void **addRouteWithoutValidityAndEndNodesCheck** (*Route* route)

Adds a route to the choice set.

#### Parameters

- **route** – Route to be added.

### addRouteWithoutValidityCheck

public void **addRouteWithoutValidityCheck** (*Route* route)

Adds a route to the choice set.

#### Parameters

- **route** – Route to be added.

### calculatePathsizes

```
public void calculatePathsizes ()
```

Calculate path sizes (also calculates route lengths if they had not been calculated before).

### calculateProbabilities

```
public void calculateProbabilities ()
```

Calculates choice probabilities using logit formula.

### calculateUtilities

```
public void calculateUtilities (VehicleType vht, EngineType et, TimeOfDay tod, double[] linkTravelTime, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, Map<EngineType, Double> energyUnitCosts, List<PricingPolicy> congestionCharges, Map<RouteChoiceParams, Double> params)
```

Re-calculates utilities for all the routes.

#### Parameters

- **vht** – Vehicle type.
- **et** – Engine type.
- **tod** – Time of day.
- **linkTravelTime** – Link travel times.
- **energyConsumptionParameters** – Base year energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency compared to the base year.
- **energyUnitCosts** – Energy unit costs.
- **congestionCharges** – Congestion charges.
- **params** – Route choice parameters.

### choose

```
public Route choose ()
```

Chooses a route based on the probabilities.

**Returns** Chosen route.

### correctUtilityWithPathSize

```
public void correctUtilityWithPathSize (int routeIndex)
```

Corrects utility with path size for a particular route within the choice set.

**Parameters**

- **routeIndex** – index of the route (list element) within the choice set

**getChoiceSet**

```
public List<Route> getChoiceSet ()
```

Getter method for the choice set.

**Returns** Choice set (list of routes).

**getDestinationNode**

```
public DirectedNode getDestinationNode ()
```

**Returns** Destination node of the choice set.

**getIndexOfRoute**

```
public int getIndexOfRoute (Route route)
```

Gets the index of a route in the choice set.

**Parameters**

- **route** – The route which index is sought for.

**Returns** Route index.

**getOriginNode**

```
public DirectedNode getOriginNode ()
```

**Returns** Origin node of the choice set.

**getPathsizes**

```
public double[] getPathsizes ()
```

Getter method for pathsizes.

**Returns** Choice pathsizes.

**getProbabilities**

```
public double[] getProbabilities ()
```

Getter method for choice probabilities.

**Returns** Choice probabilities.

### **getProbabilitiesAsList**

public `ArrayList<Double>` **getProbabilitiesAsList** ()

Getter method for choice probabilities.

**Returns** Choice probabilities.

### **getSize**

public int **getSize** ()

**Returns** Size of the choice set (number of routes).

### **getUtilities**

public `ArrayList<Double>` **getUtilities** ()

Getter method for choice utilities.

**Returns** Choice utilities.

### **printChoiceSet**

public void **printChoiceSet** ()

Prints the entire choice set.

### **printPathsizes**

public void **printPathsizes** ()

Prints pathsizes for the route set.

### **printProbabilities**

public void **printProbabilities** ()

Prints probabilities for the route set.

### **printStatistics**

public void **printStatistics** ()

Prints statistic for the route set (choice set size for each node pair).

### **printUtilities**

public void **printUtilities** ()

Prints utilities for the route set.

### **RouteSet.RouteChoiceParams**

public static enum **RouteChoiceParams**



## Enum Constants

### COST

public static final *RouteSet.RouteChoiceParams* **COST**

### DELAY

public static final *RouteSet.RouteChoiceParams* **DELAY**

### INTERSEC

public static final *RouteSet.RouteChoiceParams* **INTERSEC**

### LENGTH

public static final *RouteSet.RouteChoiceParams* **LENGTH**

### TIME

public static final *RouteSet.RouteChoiceParams* **TIME**

## RouteSetGenerator

public class **RouteSetGenerator**

RouteSetGenerator can generate, save and read route sets for the route choice.

**Author** Milan Lovric

## Fields

### INITIAL\_ROUTE\_CAPACITY

public static final int **INITIAL\_ROUTE\_CAPACITY**

### INITIAL\_ROUTE\_SET\_CAPACITY

public static final int **INITIAL\_ROUTE\_SET\_CAPACITY**

## Constructors

### RouteSetGenerator

public **RouteSetGenerator** (*RoadNetwork* roadNetwork, *Properties* props)

Constructor for the route set generator.

#### Parameters

- **roadNetwork** – Road network.
- **props** – Parameters from the config file.

### Methods

#### addRoute

public void **addRoute** (*Route route*)  
Adds a route to the route set.

#### Parameters

- **route** – Route to be added.

#### addRouteWithoutValidityCheck

public void **addRouteWithoutValidityCheck** (*Route route*)  
Adds a route to the route set.

#### Parameters

- **route** – Route to be added.

#### calculateAllPathsizes

public void **calculateAllPathsizes** ()  
Calculates all pathsizes for all the route sets (expensive operation).

#### clearRoutes

public void **clearRoutes** ()  
Clears all stored routes.

#### generateRouteSetBetweenFreightZones

public void **generateRouteSetBetweenFreightZones** (int *originFreightZone*, int *destinationFreightZone*)  
Generates routes between two freight zones. A freight zone can be either an LAD ( $\leq 1032$ ) or a point. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

**Parameters**

- **originFreightZone** – Origin freight zone.
- **destinationFreightZone** – Destination freight zone.

**generateRouteSetBetweenFreightZones**

public void **generateRouteSetBetweenFreightZones** (int *originFreightZone*, int *destinationFreightZone*, int *topNodes*)

Generates routes between two freight zones. A freight zone can be either an LAD ( $\leq 1032$ ) or a point. Zone ID ranges from the BYFM DfT model:

- England: 1 - 867
- Wales: 901 - 922
- Scotland: 1001 - 1032
- Freight airports: 1111 - 1115
- Major distribution centres: 1201 - 1256
- Freight ports: 1301 - 1388

**Parameters**

- **originFreightZone** – Origin freight zone.
- **destinationFreightZone** – Destination freight zone.
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

**generateRouteSetForFreightMatrix**

public void **generateRouteSetForFreightMatrix** (*FreightMatrix* *freightMatrix*, int *topNodes*)

Generates routes for all non-zero OD flows in the freight OD matrix. Zone ID ranges from the BYFM DfT model:

**Parameters**

- **freightMatrix** – Freight matrix.
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

**generateRouteSetForFreightMatrix**

public void **generateRouteSetForFreightMatrix** (*FreightMatrix* *freightMatrix*, int *sliceIndex*, int *sliceNumber*)

Generates routes for a slice of the OD matrix (useful for cluster computing), for topNodes only. There might still be some overlap between the slices as some nodes (to which point freight zones are assigned appear again in LAD freight zones).

**Parameters**

- **freightMatrix** – Freight matrix.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).

### generateRouteSetForFreightMatrix

```
public void generateRouteSetForFreightMatrix (FreightMatrix freightMatrix, int sliceIndex, int sliceNumber, int topNodes)
```

Generates routes for a slice of the OD matrix (useful for cluster computing), for topNodes only. There might still be some overlap between the slices as some nodes (to which point freight zones are assigned appear again in LAD freight zones).

#### Parameters

- **freightMatrix** – Freight matrix.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

### generateRouteSetForODMatrix

```
public void generateRouteSetForODMatrix (ODMatrixMultiKey matrix, int topNodes)
```

Generates routes for all non-zero OD flows in the OD matrix. For inter-zonal flows generates routes only between top N nodes.

#### Parameters

- **matrix** – Origin-destination matrix.
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

### generateRouteSetForODMatrix

```
public void generateRouteSetForODMatrix (ODMatrixMultiKey matrix)
```

Generates routes for all non-zero OD flows in the OD matrix.

#### Parameters

- **matrix** – Origin-destination matrix.

### generateRouteSetForODMatrix

```
public void generateRouteSetForODMatrix (ODMatrixMultiKey matrix, int sliceIndex, int sliceNumber, int topNodes)
```

Generates routes for a slice of the OD matrix (useful for cluster computing), for topNodes only

#### Parameters

- **matrix** – Origin-destination matrix.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).
- **topNodes** – Number of topNodes to consider for inter-zonal routes.

## generateRouteSetForODMatrix

public void **generateRouteSetForODMatrix** (*ODMatrixMultiKey* matrix, int sliceIndex, int sliceNumber)

Generates routes for a slice of the Temprow OD matrix (useful for cluster computing).

### Parameters

- **matrix** – Origin-destination matrix.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).

## generateRouteSetForODMatrixTemprow

public void **generateRouteSetForODMatrixTemprow** (*ODMatrixMultiKey* matrix, *Zoning* zoning)

Generates routes for all non-zero OD flows in the OD matrix.

### Parameters

- **matrix** – Origin-destination matrix.
- **zoning** – Temprow zoning system.

## generateRouteSetForODMatrixTemprow

public void **generateRouteSetForODMatrixTemprow** (*RealODMatrixTemprow* matrix, *Zoning* zoning, int sliceIndex, int sliceNumber)

Generates routes for a slice of the OD matrix (useful for cluster computing).

### Parameters

- **matrix** – Origin-destination matrix.
- **zoning** – Temprow zoning system.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).

## generateRouteSetForODMatrixTemprowDistanceBased

public void **generateRouteSetForODMatrixTemprowDistanceBased** (*RealODMatrixTemprow* matrix, *Zoning* zoning, int sliceIndex, int sliceNumber)

Generates routes for a slice of the OD matrix (useful for cluster computing). The number of routes increases the smaller the distance between two Temprow zones.

### Parameters

- **matrix** – Origin-destination matrix.
- **zoning** – Temprow zoning system.
- **sliceIndex** – Index of the OD matrix slice for which to generate routes [1..N].
- **sliceNumber** – Number of slices to divide matrix into (N).

### generateRouteSetNodeToNode

public void **generateRouteSetNodeToNode** (int *origin*, int *destination*)

Generates a route set between two nodes (if it does not already exist in the route set).

#### Parameters

- **origin** – Origin node ID.
- **destination** – Destination node ID.

### generateRouteSetWithLinkElimination

public void **generateRouteSetWithLinkElimination** (int *origin*, int *destination*)

Generates a route set between two nodes using the link elimination method - It first finds the fastest path and then blocks each of its links and tries to find an alternative path.

#### Parameters

- **origin** – Origin node ID.
- **destination** – Destination node ID.

### generateRouteSetWithRandomLinkEliminationRestricted

public void **generateRouteSetWithRandomLinkEliminationRestricted** (int *origin*, int *destination*)

Generates a route set between two nodes using the random link elimination method - It first finds the fastest path and then blocks random links within the fastest path and tries to find an alternative path. The search is limited by the total number of path finding calls and the required number of generated paths.

#### Parameters

- **origin** – Origin node ID.
- **destination** – Destination node ID.

### generateRouteSetWithRandomLinkEliminationRestricted

public void **generateRouteSetWithRandomLinkEliminationRestricted** (int *origin*, int *destination*, int *routeLimit*, int *generationLimit*)

Generates a route set between two nodes using the random link elimination method - It first finds the fastest path and then blocks random links within the fastest path and tries to find an alternative path. The search is limited by the total number of path finding calls and the required number of generated paths.

#### Parameters

- **origin** – Origin node ID.
- **destination** – Destination node ID.
- **routeLimit** – Maximum allowed number of generated routes.
- **generationLimit** – Number of generation trials to get a potentially new route.

## generateRouteSetZoneToZone

public void **generateRouteSetZoneToZone** (*String originLAD*, *String destinationLAD*)

Generates routes between all combinations of nodes from two LAD zones

### Parameters

- **originLAD** – Origin LAD.
- **destinationLAD** – Destination LAD.

## generateRouteSetZoneToZone

public void **generateRouteSetZoneToZone** (*String originLAD*, *String destinationLAD*, int *topNodes*)

Generates routes between top N nodes (sorted by gravitating population) from two LAD zones. If origin and destination LAD are the same (i.e., intra-zonal), then use all the nodes

### Parameters

- **originLAD** – Origin LAD.
- **destinationLAD** – Destination LAD.
- **topNodes** – Number of top nodes within LAD to consider.

## generateRouteSetZoneToZoneTempro

public void **generateRouteSetZoneToZoneTempro** (*String originZone*, *String destinationZone*, *Zoning zoning*)

Generates routes between the nearest nodes of two Tempro zones.

### Parameters

- **originZone** – Origin Tempro zone.
- **destinationZone** – Destination Tempro zone.
- **zoning** – Tempro zoning system.

## generateRouteSetZoneToZoneTemproDistanceBased

public void **generateRouteSetZoneToZoneTemproDistanceBased** (*String originZone*, *String destinationZone*, *Zoning zoning*)

Generates routes between the nearest nodes of two Tempro zones.

### Parameters

- **originZone** – Origin Tempro zone.
- **destinationZone** – Destination Tempro zone.
- **zoning** – Tempro zoning system.

## generateSingleNodeRoutes

public void **generateSingleNodeRoutes** ()

Generates single nodes routes.

### getNumberOfRouteSets

```
public int getNumberOfRouteSets ()
```

Gets the numbers of route sets (OD pairs).

**Returns** Number of route sets.

### getNumberOfRoutes

```
public int getNumberOfRoutes ()
```

Gets the total number of routes.

**Returns** Number of routes.

### getRoadNetwork

```
public RoadNetwork getRoadNetwork ()
```

Getter method for the road network.

**Returns** Road network.

### getRouteSet

```
public RouteSet getRouteSet (int origin, int destination)
```

Getter method for a route set between a specific origin and a destination.

**Parameters**

- **origin** – Origin node ID.
- **destination** – Destination node ID.

**Returns** Route set.

### getStatistics

```
public String getStatistics ()
```

Gets route set statistics in a string.

**Returns** Route set statistics.

### printChoiceSets

```
public void printChoiceSets ()
```

Prints all route sets.

### printStatistics

```
public void printStatistics ()
```

Prints all route set statistics.



## readRoutes

public void **readRoutes** (*String fileName*)  
Reads route sets from a text file.

### Parameters

- **fileName** – File name.

## readRoutesBinary

public void **readRoutesBinary** (*String fileName*)  
Reads route sets from a text file.

### Parameters

- **fileName** – File name.

## readRoutesBinaryGZIPpedWithoutValidityCheck

public void **readRoutesBinaryGZIPpedWithoutValidityCheck** (*String fileName*)  
Reads route sets from a text file.

### Parameters

- **fileName** – File name.

## readRoutesBinaryShortWithoutValidityCheck

public void **readRoutesBinaryShortWithoutValidityCheck** (*String fileName*)  
Reads route sets from a text file.

### Parameters

- **fileName** – File name.

## readRoutesBinaryWithoutValidityCheck

public void **readRoutesBinaryWithoutValidityCheck** (*String fileName*)  
Reads route sets from a text file.

### Parameters

- **fileName** – File name.

## readRoutesWithoutValidityCheck

public void **readRoutesWithoutValidityCheck** (*String fileName*)  
Reads route sets from a text file without checking whether the routes are valid.

### Parameters

- **fileName** – File name.

### removeRoutesWithEdge

public void **removeRoutesWithEdge** (int *edgeID*)

Removes all the routes that contain a given edge (used for disruption).

#### Parameters

- **edgeID** – Edge ID.

### removeRoutesWithEdge

public void **removeRoutesWithEdge** (int *edgeID*, List<*Route*> *removedRoutes*)

Removes all the routes that contain a given edge and store in the list.

#### Parameters

- **edgeID** – Edge ID.
- **removedRoutes** – List of removed routes.

### saveRoutes

public void **saveRoutes** (*String fileName*, boolean *append*)

Saves all route sets into a text file.

#### Parameters

- **fileName** – File name.
- **append** – Whether to append to an existing file.

### saveRoutesBinary

public void **saveRoutesBinary** (*String fileName*, boolean *append*)

Saves all route sets into a binary file.

#### Parameters

- **fileName** – File name.
- **append** – Whether to append to an existing file.

### saveRoutesBinaryGZIPped

public void **saveRoutesBinaryGZIPped** (*String fileName*, boolean *append*)

Saves all route sets into a binary file.

#### Parameters

- **fileName** – File name.
- **append** – Whether to append to an existing file.

## saveRoutesBinaryShort

public void **saveRoutesBinaryShort** (*String fileName*, boolean *append*)

Saves all route sets into a binary file. It also uses unsigned short (2 Bytes, which has a max. value of 65535).

### Parameters

- **fileName** – File name.
- **append** – Whether to append to an existing file.

## Trip

public class **Trip**

This class stores information about a performed trip.

**Author** Milan Lovric

### Fields

#### destination

protected int **destination**

#### engine

protected *EngineType* **engine**

#### hour

protected *TimeOfDay* **hour**

#### multiplier

protected int **multiplier**

#### origin

protected int **origin**

#### route

protected *Route* **route**

#### vehicle

protected *VehicleType* **vehicle**

## Constructors

### Trip

public **Trip** (*VehicleType* vehicle, *EngineType* engine, *Route* route, *TimeOfDay* hour, Integer origin, Integer destination)

Constructor for an LAD-based trip. Origin and destination fields are used for freight trips (according to DfT's BYFM zonal coding). Origin and destination for passenger car/AV trips are 0 as their correct origin and destination LAD zone can be obtained using the first and the last node of the route.

#### Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **origin** – Origin zone for freight trips (null for passenger trips).
- **destination** – Destination zone for freight trips (null for passenger trips).

### Trip

public **Trip** (*VehicleType* vehicle, *EngineType* engine, *Route* route, *TimeOfDay* hour, Integer origin, Integer destination, int multiplier)

Constructor for a trip. Origin and destination are used for freight trips (according to DfT's BYFM zonal coding). Origin and destination for passenger car/AV trips are 0 as their correct origin and destination LAD zone can be obtained using the first and the last node of the route. Multiplier is used to store multiple instances of the same trip (vs creating multiple objects), thus reducing the memory footprint.

#### Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **origin** – Origin zone for freight trips (null for passenger trips).
- **destination** – Destination zone for freight trips (null for passenger trips).
- **multiplier** – Multiplies the same trip.

## Methods

## getAccessEgressConsumption

```
protected Map<EnergyType, Double> getAccessEgressConsumption (double[] linkTravelTime, double[] averageAccessEgressMap, double averageAccessEgressSpeed, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency)
```

Calculate trip consumption only on access and egress.

### Parameters

- **linkTravelTime** –
- **averageAccessEgressMap** –
- **averageAccessEgressSpeed** –
- **energyConsumptions** –
- **relativeFuelEfficiency** –

**Returns** Trip consumptions.

## getCO2emission

```
public Double getCO2emission (double[] linkTravelTime, double[] averageAccessEgressMap, double averageAccessEgressSpeed, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, Map<EnergyType, Double> unitCO2Emissions, boolean flagIncludeAccessEgress)
```

Calculates total CO2 emission for the trip.

### Parameters

- **linkTravelTime** – Link travel time.
- **averageAccessEgressMap** – Average access/egress distance to a node for LAD-based trips.
- **averageAccessEgressSpeed** – Average accces/egress speed.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.
- **unitCO2Emissions** – Unit CO2 emissions.
- **boolean** – flagIncludeAccessEgress Whether to include access/egress.

**Returns** CO2 emissions per energy type.

## getConsumption

```
public Map<EnergyType, Double> getConsumption (double[] linkTravelTime, double[] averageAccessEgressMap, double averageAccessEgressSpeed, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, boolean flagIncludeAccessEgress)
```

Calculate trip consumption including access and egress.

### Parameters

- **linkTravelTime** – Link travel time.
- **averageAccessEgressMap** – Average access/egress distance to a node for LAD-based trips.
- **averageAccessEgressSpeed** – Average access/egress speed.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.
- **boolean** – flagIncludeAccessEgress Whether to include access/egress.

**Returns** Trip consumptions.

## getCost

```
public double getCost (double[] linkTravelTime, double[] averageAccessEgressMap, double averageAccessEgressSpeed, Map<EnergyType, Double> energyUnitCosts, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, List<PricingPolicy> congestionCharges, boolean flagIncludeAccessEgress)
```

Calculate cost of the trip (fuel cost + congestion charge, if any).

### Parameters

- **linkTravelTime** – Link travel time.
- **averageAccessEgressMap** – Average access/egress distance to a node for LAD-based trips.
- **averageAccessEgressSpeed** – Average access/egress speed.
- **energyUnitCosts** – Energy unit costs.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.
- **congestionCharges** – Congestion charges.
- **boolean** – flagIncludeAccessEgress Whether to include access/egress.

**Returns** Total trip cost.

### getDestination

public int **getDestination** ()  
Gets freight trip destination zone (using DfT BYFM zone coding).  
**Returns** Freight trip destination zone.

### getDestinationLAD

public *String* **getDestinationLAD** (*Map*<*Integer*, *String*> *nodeToZoneMap*)  
Gets trip destination zone (LAD).

#### Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

**Returns** Trip destination zone.

### getDestinationLadID

public int **getDestinationLadID** ()  
Gets trip destination LAD zone ID.  
**Returns** Trip destination zone LAD ID.

### getDestinationNode

public DirectedNode **getDestinationNode** ()  
Gets the trip destination node.  
**Returns** Destination node.

### getEngine

public *EngineType* **getEngine** ()  
Getter method for engine type.  
**Returns** Vehicle engine type.

### getLength

public double **getLength** (*double*[] *averageAccessEgressMap*)  
Get trip length including access/egress.

#### Parameters

- **averageAccessEgressMap** – Mapping between nodeID and average access/egress for that node.

**Returns** Trip length including access/egress [in km]

### getMultiplier

```
public int getMultiplier ()
```

Getter method for the multiplier.

**Returns** Multiplier.

### getOrigin

```
public int getOrigin ()
```

Gets freight trip origin zone (using DfT BYFM zone coding).

**Returns** Freight trip origin zone.

### getOriginLAD

```
public String getOriginLAD (Map<Integer, String> nodeToZoneMap)
```

Gets trip origin zone (LAD).

**Parameters**

- **nodeToZoneMap** – Mapping from nodes to zones.

**Returns** Trip origin zone.

### getOriginLadID

```
public int getOriginLadID ()
```

Gets trip origin LAD zone ID.

**Returns** Origin zone LAD ID.

### getOriginNode

```
public DirectedNode getOriginNode ()
```

Gets the trip origin node.

**Returns** Origin node.

### getRoute

```
public Route getRoute ()
```

Getter method for the route.

**Returns** Route.

### getTimeOfDay

```
public TimeOfDay getTimeOfDay ()
```

Getter method for the time of day.

**Returns** Time of day.



## getTravelTime

```
public double getTravelTime (double[] linkTravelTime, double avgIntersectionDelay, double[] averageAccessEgressMap, double averageAccessEgressSpeed, boolean flagIncludeAccessEgress)
```

Calculates travel time including access/egress.

### Parameters

- **linkTravelTime** – Link-based travel time (should be for the same hour as the trip’s time of day).
- **avgIntersectionDelay** – Average intersection delay.
- **averageAccessEgressMap** – Mapping between nodeID and average access/egress for that node.
- **averageAccessEgressSpeed** – Average access/egress speed.
- **flagIncludeAccessEgress** – Whether to include access/egress travel time.

**Returns** Trip travel time including access/egress [in min].

## getVehicle

```
public VehicleType getVehicle ()
```

Getter method for vehicle type.

**Returns** Vehicle type.

## isTripGoingThroughCongestionChargingZone

```
public boolean isTripGoingThroughCongestionChargingZone (String policyName,  
                                                         List<PricingPolicy> congestionCharges)
```

Check whether trip is going through a congestion charging zone for a particular policy.

### Parameters

- **policyName** – Policy name.
- **congestionCharges** – Congestion charges.

**Returns** True if it is going through the congestion charging zone.

## toString

```
public String toString ()
```

## TripMinor

```
public class TripMinor extends Trip
```

This class stores information about a performed trip on minor roads (for which the network is not modelled).

**Author** Milan Lovric

## Fields

### zoning

public static *Zoning* **zoning**

## Constructors

### TripMinor

public **TripMinor** (*VehicleType* vehicle, *EngineType* engine, *TimeOfDay* hour, *Integer* originTemproZoneID, *Integer* destinationTemproZoneID, double length, *Zoning* zoning)

Constructor for a trip. Origin and destination are used for freight trips (according to DfT's BYFM zonal coding). Origin and destination for passenger car/AV trips are 0 as their correct origin and destination zone can be obtained using the first and the last node of the route.

#### Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **originTemproZoneID** – Origin temprow zone ID.
- **destinationTemproZoneID** – Destination temprow zone ID.
- **length** – Trip length;
- **zoning** – Zoning system.

### TripMinor

public **TripMinor** (*VehicleType* vehicle, *EngineType* engine, *TimeOfDay* hour, *Integer* originTemproZoneID, *Integer* destinationTemproZoneID, double length, *Zoning* zoning, int multiplier)

Constructor for a trip. Origin and destination are used for freight trips (according to DfT's BYFM zonal coding). Origin and destination for passenger car/AV trips are 0 as their correct origin and destination zone can be obtained using the first and the last node of the route.

#### Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **originTemproZoneID** – Origin temprow zone ID.
- **destinationTemproZoneID** – Destination temprow zone ID.
- **length** – Trip length;
- **zoning** – Zoning system.
- **multiplier** – Multiplies the same trip.

## Methods

### getAccessEgressConsumption

```
protected Map<EnergyType, Double> getAccessEgressConsumption (double[] linkTravelTime,
double[] distanceFromTempoZoneToNearestNode,
double averageAccessEgressSpeed, Map<VehicleType,
Map<EngineType,
Map<WebTAG, Double>>> energyConsumptionParameters,
Map<VehicleType,
Map<EngineType, Double>> relativeFuelEfficiency)
```

### getCO2emission

```
public double getCO2emission (double averageSpeed, Map<VehicleType, Map<EngineType,
Map<WebTAG, Double>>> energyConsumptionParameters,
Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency,
Map<EnergyType, Double> unitCO2Emissions)
```

Gets CO2 emission for the minor trip.

#### Parameters

- **averageSpeed** – Average speed for a minor trip.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.

**Returns** CO2 emission for the trip.

### getCO2emission

```
public Double getCO2emission (double[] linkTravelTime, double[] distanceFromTempoZoneToNearestNode, double averageAccessEgressSpeed, Map<VehicleType,
Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, Map<EnergyType, Double> unitCO2Emissions, boolean flagIncludeAccessEgress)
```

### getConsumption

```
public Map<EnergyType, Double> getConsumption (double averageSpeed, Map<VehicleType,
Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType,
Map<EngineType, Double>> relativeFuelEfficiency)
```

Gets energy consumptions for the minor trip.

#### Parameters

- **averageSpeed** – Average speed for a minor trip.

- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.

**Returns** Energy consumptions for the trip.

### getConsumption

```
public Map<EnergyType, Double> getConsumption (double[] linkTravelTime, double[] distanceFromTempoZoneToNearestNode, double averageAccessEgressSpeed, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, boolean flagIncludeAccessEgress)
```

### getCost

```
public double getCost (double averageSpeed, Map<EnergyType, Double> energyUnitCosts, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency)
```

Gets fuel cost for the minor trip.

#### Parameters

- **averageSpeed** – Average speed for a minor trip.
- **energyUnitCosts** – Energy unit costs.
- **energyConsumptionParameters** – Energy consumption parameters.
- **relativeFuelEfficiency** – Relative fuel efficiency.

**Returns** Energy consumptions for the trip.

### getCost

```
public double getCost (double[] linkTravelTime, double[] distanceFromTempoZoneToNearestNode, double averageAccessEgressSpeed, Map<EnergyType, Double> energyUnitCosts, Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, List<PricingPolicy> congestionCharges, boolean flagIncludeAccessEgress)
```

### getDestinationLAD

```
public String getDestinationLAD ()
```

Gets trip destination zone (LAD), from tempo to LAD mapping (not from route nodes).

**Returns** Trip destination zone.

### getDestinationLAD

public **String** **getDestinationLAD** (**Map**<**Integer**, **String**> *nodeToZoneMap*)  
Gets trip destination zone (LAD), from temprow to LAD mapping (not from route nodes).

#### Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

**Returns** Trip destination zone.

### getDestinationLadID

public int **getDestinationLadID** ()  
Gets trip destination LAD zone ID.

**Returns** Trip destination zone LAD ID.

### getDestinationTemprowZone

public **String** **getDestinationTemprowZone** ()  
Gets trip destination temprow zone.

**Returns** Trip destination temprow zone.

### getLength

public double **getLength** ()  
Gets trip length (no separate access/egress for minor road trips).

**Returns** Trip length [in km].

### getOriginLAD

public **String** **getOriginLAD** ()  
Gets trip origin zone (LAD), from temprow to LAD mapping (not from route nodes).

**Returns** Trip origin zone.

### getOriginLAD

public **String** **getOriginLAD** (**Map**<**Integer**, **String**> *nodeToZoneMap*)  
Gets trip origin zone (LAD), from temprow to LAD mapping (not from route nodes).

#### Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

**Returns** Trip origin zone.

### getOriginLadID

```
public int getOriginLadID ()  
    Gets trip origin LAD zone ID.  
  
    Returns Origin zone LAD ID.
```

### getOriginTemproZone

```
public String getOriginTemproZone ()  
    Gets trip origin temprow zone code.  
  
    Returns Trip origin temprow zone code.
```

### getTravelTime

```
public double getTravelTime (double averageSpeed)  
    Gets travel time for the minor trip.  
  
    Parameters  
        • averageSpeed – Average speed for a minor trip.  
  
    Returns Travel time.
```

### getTravelTime

```
public double getTravelTime (double[] linkTravelTime, double avgIntersectionDelay, double[] distanceFromTemprowZoneToNearestNode, double averageAccessEgressSpeed, boolean flagIncludeAccessEgress)
```

### getZoning

```
public Zoning getZoning ()  
    Getter for the zoning system.  
  
    Returns Zoning.
```

### toString

```
public String toString ()
```

### TripTemprow

```
public class TripTemprow extends Trip  
    This class stores information about a performed trip (when using the Temprow zoning system).  
  
    Author Milan Lovric
```

## Fields

### zoning

public static *Zoning* **zoning**

## Constructors

### TripTempro

public **TripTempro** (*VehicleType* vehicle, *EngineType* engine, *Route* route, *TimeOfDay* hour, Integer origin-  
TemproZoneID, Integer destinationTemproZoneID, *Zoning* zoning)

Constructor for a passenger car trip using the Tempro zoning system. Origin and destination are integer IDs of Tempro zones.

#### Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **originTemproZoneID** – Origin tempro zone ID.
- **destinationTemproZoneID** – Destination tempro zone ID.
- **zoning** – Zoning system.

### TripTempro

public **TripTempro** (*VehicleType* vehicle, *EngineType* engine, *Route* route, *TimeOfDay* hour, Integer origin-  
TemproZoneID, Integer destinationTemproZoneID, *Zoning* zoning, int multiplier)

Constructor for a passenger car trip using the Tempro zoning system. Origin and destination are integer IDs of Tempro zones. Multiplier is used to store multiple instances of the same trip (vs creating multiple objects), thus reducing the memory footprint.

#### Parameters

- **vehicle** – Vehicle type.
- **engine** – Engine type.
- **route** – Route.
- **hour** – Time of day.
- **originTemproZoneID** – Origin tempro zone ID.
- **destinationTemproZoneID** – Destination tempro zone ID.
- **zoning** – Zoning system.
- **multiplier** – Multiplies the same trip.

## Methods

### getAccessEgressConsumption

```
protected Map<EnergyType, Double> getAccessEgressConsumption (double[] linkTravelTime,
double[] distanceFromTempoZoneToNearestNode,
double averageAccessEgressSpeed, Map<VehicleType,
Map<EngineType,
Map<WebTAG, Double>>> energyConsumptionParameters,
Map<VehicleType,
Map<EngineType, Double>> relativeFuelEfficiency)
```

### getCO2emission

```
public Double getCO2emission (double[] linkTravelTime, double[] distanceFromTempoZoneToNearestNode, double averageAccessEgressSpeed, Map<VehicleType,
Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, Map<EnergyType, Double> unitCO2Emissions, boolean flagIncludeAccessEgress)
```

### getConsumption

```
public Map<EnergyType, Double> getConsumption (double[] linkTravelTime, double[] distanceFromTempoZoneToNearestNode, double averageAccessEgressSpeed, Map<VehicleType, Map<EngineType,
Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, boolean flagIncludeAccessEgress)
```

### getCost

```
public double getCost (double[] linkTravelTime, double[] distanceFromTempoZoneToNearestNode, double averageAccessEgressSpeed, Map<EnergyType, Double> energyUnitCosts,
Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> energyConsumptionParameters, Map<VehicleType, Map<EngineType, Double>> relativeFuelEfficiency, List<PricingPolicy> congestionCharges, boolean flagIncludeAccessEgress)
```

### getDestinationLAD

```
public String getDestinationLAD ()
    Gets trip destination zone (LAD), from Tempo to LAD mapping (not from route nodes).
```

**Returns** Trip destination zone.



### getDestinationLAD

```
public String getDestinationLAD (Map<Integer, String> nodeToZoneMap)
```

Gets trip destination zone (LAD), from tempo to LAD mapping (not from route nodes).

#### Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

**Returns** Trip destination zone.

### getDestinationLadID

```
public int getDestinationLadID ()
```

Gets trip destination LAD zone ID.

**Returns** Trip destination zone LAD ID.

### getDestinationTemproZone

```
public String getDestinationTemproZone ()
```

Gets trip destination tempo zone.

**Returns** Trip destination tempo zone.

### getLength

```
public double getLength ()
```

Gets trip length including access/egress (from Tempo centroid to node).

**Returns** Trip length [in km].

### getOriginLAD

```
public String getOriginLAD ()
```

Gets trip origin zone (LAD), from Tempo to LAD mapping (not from route nodes).

**Returns** Trip origin zone.

### getOriginLAD

```
public String getOriginLAD (Map<Integer, String> nodeToZoneMap)
```

Gets trip origin zone (LAD), from tempo to LAD mapping (not from route nodes).

#### Parameters

- **nodeToZoneMap** – Mapping from nodes to zones.

**Returns** Trip origin zone.

### getOriginLadID

```
public int getOriginLadID ()  
    Gets trip origin LAD zone ID.  
  
    Returns Origin zone LAD ID.
```

### getOriginTemproZone

```
public String getOriginTemproZone ()  
    Gets trip origin temprow zone code.  
  
    Returns Trip origin temprow zone code.
```

### getTravelTime

```
public double getTravelTime (double[] linkTravelTime, double avgIntersectionDelay, double[] distanceFromTemprowZoneToNearestNode, double averageAccessEgressSpeed, boolean flagIncludeAccessEgress)
```

### getZoning

```
public Zoning getZoning ()  
    Getter for the zoning system.  
  
    Returns Zoning.
```

## 1.2.7 nismod.transport.optimisation

### SPSA

```
public class SPSA  
    Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version optimises the cells of the OD matrix. http://www.jhuapl.edu/SPSA/  
  
    Author Milan Lovric
```

### Fields

#### THETA\_MAX

```
public static final double THETA_MAX
```

#### THETA\_MIN

```
public static final double THETA_MIN
```

## Constructors

### SPSA

public **SPSA** ()

## Methods

### getLossFunctionEvaluations

public *List<Double>* **getLossFunctionEvaluations** ()

**Returns** Loss function evaluations for all iterations.

### getThetaEstimate

public *RealODMatrix* **getThetaEstimate** ()

Getter function for the optimisation result (OD matrix).

**Returns** Estimated OD matrix.

### initialise

public void **initialise** (*RoadNetworkAssignment* rna, *Properties* props, *RealODMatrix* initialTheta, double a, double A, double c, double alpha, double gamma)

Initialise the SPSA algorithm with starting values.

#### Parameters

- **rna** – Road network assignment.
- **props** – Parameters from the config file.
- **initialTheta** – Initial OD matrix.
- **a** – SPSA parameter.
- **A** – SPSA parameter.
- **c** – SPSA parameter.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

### lossFunction

public double **lossFunction** ()

Calculate the loss function of the latest theta estimate (OD matrix).

**Returns** RMSN for the difference between volumes and traffic counts.

## runSPSA

public void **runSPSA** (int *maxIterations*)  
Run the algorithm.

### Parameters

- **maxIterations** – Maximum number of iterations.

## SPSA2

public class **SPSA2**

Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version of the algorithm keeps OD matrix constant, but optimises start and end node probabilities (the probability of a trip starting/ending at a particular node within LAD). <http://www.jhuapl.edu/SPSA/>

**Author** Milan Lovric

## Fields

### THETA\_MAX

public static final double **THETA\_MAX**

### THETA\_MIN

public static final double **THETA\_MIN**

## Constructors

### SPSA2

public **SPSA2** ()

## Methods

### getLossFunctionEvaluations

public [List<Double>](#) **getLossFunctionEvaluations** ()

**Returns** Loss function evaluations for all iterations.

## initialise

public void **initialise** ([RoadNetworkAssignment](#) *rna*, [Properties](#) *props*, [ODMatrixMultiKey](#) *odm*,  
[HashMap<Integer, Double>](#) *initialThetaStart*, [HashMap<Integer, Double>](#)  
*initialThetaEnd*, double *a*, double *A*, double *c*, double *alpha*, double *gamma*)

Initialise the SPSA algorithm with starting values.

### Parameters

- **rna** – Road network assignment
- **props** – Parameters from the config file.
- **odm** – Origin-destination matrix.
- **initialThetaStart** – Initial start node probabilities.
- **initialThetaEnd** – Initial end node probabilities.
- **a** – SPSA parameter.
- **A** – SPSA parameter.
- **c** – SPSA parameter.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

### lossFunction

public double **lossFunction** ()

Calculate the loss function of the latest theta estimate (OD matrix).

**Returns** RMSN for the difference between volumes and traffic counts.

### runSPSA

public void **runSPSA** (int *maxIterations*)

Run the algorithm.

**Parameters**

- **maxIterations** – Maximum number of iterations.

### SPSA3

public class **SPSA3**

Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version optimises OD matrix and start/end node probabilities simultaneously. <http://www.jhuapl.edu/SPSA/>

**Author** Milan Lovric

### Fields

#### THETA\_MAX\_FLOW

public static final double **THETA\_MAX\_FLOW**

#### THETA\_MAX\_PROBABILITY

public static final double **THETA\_MAX\_PROBABILITY**

## THETA\_MIN\_FLOW

public static final double **THETA\_MIN\_FLOW**

## THETA\_MIN\_PROBABILITY

public static final double **THETA\_MIN\_PROBABILITY**

## Constructors

### SPSA3

public **SPSA3** ()

## Methods

### getLossFunctionEvaluations

public [List<Double>](#) **getLossFunctionEvaluations** ()  
Getter function for loss function evaluations for all iterations.  
**Returns** Loss function evaluations for all iterations.

### getThetaEstimate

public [RealODMatrix](#) **getThetaEstimate** ()  
Getter function for the optimisation result (OD matrix).  
**Returns** Estimated OD matrix.

### getThetaEstimateEnd

public [HashMap<Integer, Double>](#) **getThetaEstimateEnd** ()  
Getter function for the optimisation result (end nodes probabilities).  
**Returns** Estimated end nodes probabilities.

### getThetaEstimateStart

public [HashMap<Integer, Double>](#) **getThetaEstimateStart** ()  
Getter function for the optimisation result (start nodes probabilities).  
**Returns** Estimated start nodes probabilities.

## initialise

```
public void initialise (RoadNetworkAssignment rna, RouteSetGenerator rsg, Properties routeChoiceParams, RealODMatrix initialTheta, HashMap<Integer, Double> initialThetaStart, HashMap<Integer, Double> initialThetaEnd, double a1, double A1, double c1, double a2, double A2, double c2, double alpha, double gamma)
```

Initialise the SPSA algorithm with starting values.

### Parameters

- **rna** – Road network assignment.
- **rsg** – Route set generator.
- **routeChoiceParams** – Route choice parameters.
- **initialTheta** – Initial OD matrix.
- **initialThetaStart** – Initial start node probabilities.
- **initialThetaEnd** – Initial end node probabilities.
- **a1** – SPSA parameter for OD estimation.
- **A1** – SPSA parameter for OD estimation.
- **c1** – SPSA parameter for OD estimation.
- **a2** – SPSA parameter for nodes probability estimation.
- **A2** – SPSA parameter for nodes probability estimation.
- **c2** – SPSA parameter for nodes probability estimation.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

## lossFunction

```
public double lossFunction ()
```

Calculate the loss function of the latest theta estimate (OD matrix).

**Returns** RMSN for the difference between volumes and traffic counts.

## runSPSA

```
public void runSPSA (int maxIterations)
```

Run the algorithm.

### Parameters

- **maxIterations** – Maximum number of iterations.

## saveNodeProbabilities

```
public void saveNodeProbabilities (String outputFile)
```

Saves node probabilities to an output file.

### Parameters

- **outputFile** – Output file name (with path).

## SPSA4

public class **SPSA4**

Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version optimises Tempro level OD matrix. <http://www.jhuapl.edu/SPSA/>

**Author** Milan Lovric

## Fields

### THETA\_MAX

public static final double **THETA\_MAX**

### THETA\_MIN

public static final double **THETA\_MIN**

## Constructors

### SPSA4

public **SPSA4** (*Properties props*)

## Methods

### getLossFunctionEvaluations

public *List<Double>* **getLossFunctionEvaluations** ()

**Returns** Loss function evaluations for all iterations.

### getThetaEstimate

public *RealODMatrixTempo* **getThetaEstimate** ()

Getter function for the optimisation result (OD matrix).

**Returns** Estimated OD matrix.

## initialise

public void **initialise** (*RoadNetworkAssignment rna*, *Zoning zoning*, *RouteSetGenerator rsg*, *RealOD-MatrixTempo initialTheta*, double *a*, double *A*, double *c*, double *alpha*, double *gamma*)

Initialise the SPSA algorithm with starting values.



### Parameters

- **rna** – Road network assignment.
- **zoning** – Zoning system for tempo zones.
- **rsg** – Route set generator with routes to be used in assignment.
- **initialTheta** – Initial OD matrix.
- **a** – SPSA parameter.
- **A** – SPSA parameter.
- **c** – SPSA parameter.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

### lossFunction

public double **lossFunction** ()

Calculate the loss function of the latest theta estimate (OD matrix).

**Returns** Loss function.

### runSPSA

public void **runSPSA** (int *maxIterations*)

Run the algorithm.

#### Parameters

- **maxIterations** – Maximum number of iterations.

### SPSA5

public class **SPSA5**

Implements SPSA optimisation algorithm (Simultaneous Perturbation Stochastic Approximation). This version optimises Tempo level OD matrix. <http://www.jhuapl.edu/SPSA/>

**Author** Milan Lovric

### Fields

#### THETA\_MAX

public static final double **THETA\_MAX**

#### THETA\_MIN

public static final double **THETA\_MIN**

## Constructors

### SPSA5

public **SPSA5** (*Properties props*)

## Methods

### getLossFunctionEvaluations

public *List<Double>* **getLossFunctionEvaluations** ()

**Returns** Loss function evaluations for all iterations.

### getThetaEstimate

public *RealODMatrixTempro* **getThetaEstimate** ()

Getter function for the optimisation result (OD matrix).

**Returns** Estimated OD matrix.

### initialise

public void **initialise** (*RoadNetworkAssignment rna*, *Zoning zoning*, *RouteSetGenerator rsg*, *RealOD-MatrixTempro initialTheta*, double *a*, double *A*, double *c*, double *alpha*, double *gamma*)

Initialise the SPSA algorithm with starting values.

#### Parameters

- **rna** – Road network assignment.
- **zoning** – Zoning system for temprow zones.
- **rsg** – Route set generator with routes to be used in assignment.
- **initialTheta** – Initial OD matrix.
- **a** – SPSA parameter.
- **A** – SPSA parameter.
- **c** – SPSA parameter.
- **alpha** – SPSA parameter.
- **gamma** – SPSA parameter.

### lossFunction

public double **lossFunction** ()

Calculate the loss function of the latest theta estimate (OD matrix).

**Returns** Loss function.

## runSPSA

public void **runSPSA** (int *maxIterations*)

Run the algorithm.

### Parameters

- **maxIterations** – Maximum number of iterations.

## 1.2.8 nismod.transport.rail

### RailDemandModel

public class **RailDemandModel**

### Fields

#### baseYear

public static int **baseYear**

### Constructors

#### RailDemandModel

public **RailDemandModel** (String *baseYearRailStationUsageFile*, String *populationFile*, String *GVAFile*, String *elasticitiesFile*, String *railStationJourneyFaresFile*, String *railStationGeneralisedJourneyTimesFile*, String *carZonalJourneyCostsFile*, String *railTripRatesFile*, List<Intervention> *interventions*, Properties *props*)

Constructor for the rail demand model.

### Parameters

- **baseYearRailStationUsageFile** – Base year rail station usage file (demand).
- **populationFile** – Population file.
- **GVAFile** – GVA file.
- **elasticitiesFile** – Elasticities file.
- **railStationJourneyFaresFile** – Rail fares file.
- **railStationGeneralisedJourneyTimesFile** – GJT file.
- **carZonalJourneyCostsFile** – Zonal car journey costs file.
- **railTripRatesFile** – Rail trip rates file.
- **interventions** – List of interventions.
- **props** – Properties.

### Throws

- **IOException** –
- **FileNotFoundException** –

## Methods

### addNLCofDevelopedStation

public void **addNLCofDevelopedStation** (int *NLC*)  
Adds NLC of a newly built rail station.

#### Parameters

- **NLC** – Id of a newly built rail station.

### addYearOfDevelopment

public void **addYearOfDevelopment** (int *year*)  
Adds a year in which a new rail station is built.

#### Parameters

- **year** – Year in which a new rail station is built.

### getRailStationDemand

public *RailStationDemand* **getRailStationDemand** (int *year*)  
Getter method for the passenger rail station demand in a given year.

#### Parameters

- **year** – Year for which the demand is requested.

**Returns** Rail station demand with total passenger counts (entry + exit).

### predictAndSaveRailwayDemands

public void **predictAndSaveRailwayDemands** (int *toYear*, int *fromYear*)  
Predicts rail station demand (total passenger counts at each station) up to toYear (if flag is true, also intermediate years) and saves results.

#### Parameters

- **toYear** – The final year for which the demand is predicted.
- **fromYear** – The year from which the predictions are made.

### predictRailwayDemand

public void **predictRailwayDemand** (int *predictedYear*, int *fromYear*)  
Predicts passenger railway demand (passenger counts at each station). Rail station demand for fromYear needs to be contained in the memory.

#### Parameters

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

### **predictRailwayDemandUsingResultsOfFromYear**

public void **predictRailwayDemandUsingResultsOfFromYear** (int *predictedYear*, int *fromYear*)  
Predicts passenger railway demand (passenger counts at each station). Uses already existing results of the fromYear, from the output folder.

#### **Parameters**

- **predictedYear** – The year for which the demand is predicted.
- **fromYear** – The year from which demand the prediction is made.

### **predictRailwayDemands**

public void **predictRailwayDemands** (int *toYear*, int *baseYear*)  
Predicts rail station demand (total passenger counts at each station) for all years from baseYear to toYear.

#### **Parameters**

- **toYear** – The final year for which the demand is predicted.
- **baseYear** – The base year from which the predictions are made.

### **printNLCsOfNewStations**

public void **printNLCsOfNewStations** ()  
Prints NLCs of new rail stations.

### **printYearsOfNewStations**

public void **printYearsOfNewStations** ()  
Prints years in which development of new rail stations takes place.

### **saveAllResults**

public void **saveAllResults** (int *year*)  
Saves all results into the output folder.

#### **Parameters**

- **year** – Year of the data.

### **saveRailStationDemand**

public void **saveRailStationDemand** (int *year*, String *outputFile*)  
Saves rail station demand to an output file.

#### **Parameters**

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

## saveZonalRailStationDemand

public void **saveZonalRailStationDemand** (int *year*, *String* *outputFile*)  
Saves zonal rail station demand to an output file.

### Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

## setRailStationDemand

public void **setRailStationDemand** (int *year*, *RailStationDemand* *demand*)  
Setter method for the passenger rail station demand in a given year.

### Parameters

- **year** – Year for which the demand is set.
- **demand** – Rail station demand.

## RailDemandModel.ElasticityArea

public static enum **ElasticityArea**

### Enum Constants

#### LT

public static final *RailDemandModel.ElasticityArea* **LT**

#### OTHER

public static final *RailDemandModel.ElasticityArea* **OTHER**

#### PTE

public static final *RailDemandModel.ElasticityArea* **PTE**

#### SE

public static final *RailDemandModel.ElasticityArea* **SE**

## RailDemandModel.ElasticityTypes

public static enum **ElasticityTypes**

## Enum Constants

### COST\_CAR

public static final *RailDemandModel.ElasticityTypes* **COST\_CAR**

### COST\_RAIL

public static final *RailDemandModel.ElasticityTypes* **COST\_RAIL**

### GVA

public static final *RailDemandModel.ElasticityTypes* **GVA**

### POPULATION

public static final *RailDemandModel.ElasticityTypes* **POPULATION**

### TIME

public static final *RailDemandModel.ElasticityTypes* **TIME**

## RailStation

public class **RailStation**

This class stores information about a rail station.

**Author** Milan Lovric

## Constructors

### RailStation

public **RailStation** (int *nlc*, *RailModeType* *mode*, *String* *stationName*, *String* *naptanName*, int *easting*, int *northing*, int *yearUsage*, double *dayUsage*, int *runDays*, *String* *ladCode*, *String* *ladName*, *ElasticityArea* *area*)

Constructor for a station.

#### Parameters

- **nlc** – National Location Code.
- **mode** – Which mode is served by this station.
- **stationName** – Station name.
- **naptanName** – Longer name from NaPTAN.
- **easting** – Easting coordinate.
- **northing** – Northing coordinate.

- **yearUsage** – Yearly station usage (entries and exits combined).
- **dayUsage** – Daily station usage (yearly usage divided by the number of operational days in a year).
- **runDays** – The number of operational days in a year.
- **ladCode** – LAD code of the zone in which the station is located.
- **ladName** – LAD name of the zone in which the station is located.
- **area** – Elasticity area in which the station is located.

## RailStation

public **RailStation** (*RailStation* station)  
Constructor for a station.

### Parameters

- **station** – Rail station which data is going to be copied.

## Methods

### getArea

public *ElasticityArea* **getArea** ()  
Getter method for the elasticity area in which station is located.  
**Returns** LAD name.

### getDayUsage

public double **getDayUsage** ()  
Getter method for daily usage.  
**Returns** Daily usage.

### getEasting

public int **getEasting** ()  
Getter method for easting.  
**Returns** Easting.

### getLADCode

public *String* **getLADCode** ()  
Getter method for the LAD code in which station is located.  
**Returns** LAD code.



### getLADName

```
public String getLADName ()
```

Getter method for the LAD name in which station is located.

**Returns** LAD name.

### getMode

```
public RailModeType getMode ()
```

Getter method for the rail mode type.

**Returns** Rail mode type.

### getNLC

```
public int getNLC ()
```

Getter method for the NLC (National Location Code) of the station.

**Returns** NLC.

### getNaPTANName

```
public String getNaPTANName ()
```

Getter method for the station NaPTAN name.

**Returns** NaPTAN name.

### getName

```
public String getName ()
```

Getter method for the station name.

**Returns** Name.

### getNorthing

```
public int getNorthing ()
```

Getter method for Northing.

**Returns** Northing.

### getRunDays

```
public int getRunDays ()
```

Getter method for number of operational days.

**Returns** Number of operaional days.

### getYearlyUsage

```
public int getYearlyUsage ()
    Getter method for yearly usage.

    Returns Yearly usage.
```

### setDailyUsage

```
public void setDailyUsage (double usage)
    Setter method for daily usage.
```

#### Parameters

- **usage** – Daily usage.

### setYearlyUsage

```
public void setYearlyUsage (int usage)
    Setter method for yearly usage.
```

#### Parameters

- **usage** – Yearly usage.

### toString

```
public String toString ()
```

### RailStation.RailModeType

```
public static enum RailModeType
```

#### Enum Constants

##### DLR

```
public static final RailStation.RailModeType DLR
```

##### LRAIL

```
public static final RailStation.RailModeType LRAIL
```

##### NRAIL

```
public static final RailStation.RailModeType NRAIL
```

## TUBE

public static final *RailStation.RailModeType* **TUBE**

### RailStationDemand

public class **RailStationDemand**

This class stores passenger rail demand = station usage data (entries + exists).

**Author** Milan Lovric

### Constructors

#### RailStationDemand

public **RailStationDemand** (*String* *fileName*)

#### RailStationDemand

public **RailStationDemand** (*List<String>* *header*)

Constructor for empty rail station demand.

##### Parameters

- **header** –

### Methods

#### addStation

public void **addStation** (*RailStation* *station*)

Add a rail station data to the rail demand (overwrites existing one).

##### Parameters

- **station** –

#### calculateDailyZonalUsageAverage

public *HashMap<String, Double>* **calculateDailyZonalUsageAverage** ()

Calculates daily zonal usage (the average for all stations within LAD).

**Returns** Daily zonal usage per station.

#### calculateDailyZonalUsageTotal

public *HashMap<String, Double>* **calculateDailyZonalUsageTotal** ()

Calculates daily zonal usage (the sum for all stations within LAD).

**Returns** Daily zonal usage.

### calculateYearlyZonalUsageAverage

public `HashMap<String, Integer>` **calculateYearlyZonalUsageAverage** ()

Calculates yearly zonal usage (the average for all stations within LAD).

**Returns** Yearly zonal usage per station.

### calculateYearlyZonalUsageTotal

public `HashMap<String, Integer>` **calculateYearlyZonalUsageTotal** ()

Calculates yearly zonal usage (the sum for all stations within LAD).

**Returns** Yearly zonal usage.

### createListOfStationsWithinEachLAD

public `HashMap<String, List<RailStation>>` **createListOfStationsWithinEachLAD** ()

Creates a list of stations within each LAD.

**Returns** List of stations within each LAD.

### getHeader

public `List<String>` **getHeader** ()

Getter method for the header.

**Returns** header

### getRailDemandList

public `List<RailStation>` **getRailDemandList** ()

Getter method for the rail demand list.

**Returns** Rail demand list

### getRailDemandMap

public `Map<Integer, RailStation>` **getRailDemandMap** ()

Getter method for the rail demand map.

**Returns** Rail demand map.

### printRailDemand

public void **printRailDemand** (`String message`)

Print rail demand.

**Parameters**

- **message** – Message to print before the demand.

### printRailDemandNLCSorted

public void **printRailDemandNLCSorted** (*String message*)  
Print rail demand sorted on NLC.

#### Parameters

- **message** – Message to print before the demand.

### printRailDemandNameSorted

public void **printRailDemandNameSorted** (*String message*)  
Print rail demand sorted on station name.

#### Parameters

- **message** – Message to print before the demand.

### printRailDemandUsageSorted

public void **printRailDemandUsageSorted** (*String message*)  
Print rail demand sorted on station usage.

#### Parameters

- **message** – Message to print before the demand.

### removeStation

public boolean **removeStation** (int *NLC*)  
Remove station with a given NLC code.

#### Parameters

- **NLC** – Station code.

**Returns** true if station existed in demand and was successfully removed.

### saveRailStationDemand

public void **saveRailStationDemand** (int *year*, *String outputFile*)  
Saves rail station demand to an output file.

#### Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

### saveZonalRailStationDemand

public void **saveZonalRailStationDemand** (int *year*, *String outputFile*)  
Saves zonal rail station demand to an output file.

#### Parameters

- **year** – Year of the data.
- **outputFile** – Output file name (with path).

### **sortStationsOnNLC**

public void **sortStationsOnNLC** ()  
Sorts stations on NLC in an ascending order.

### **sortStationsOnName**

public void **sortStationsOnName** ()  
Sorts stations on station name in an ascending order.

### **sortStationsOnUsage**

public void **sortStationsOnUsage** ()  
Sorts stations on usage in a descending order.

## **1.2.9 nismod.transport.scripts**

### **ArcAnalysis**

public class **ArcAnalysis**  
This scripts parses the outputs for the CaMkOx case study (adjust paths as needed).  
**Author** Milan Lovric

### **Fields**

#### **TABLE\_FONT**

public static final **Font** **TABLE\_FONT**

### **arcZones**

static final **List<String>** **arcZones**

### **Methods**

#### **main**

public static void **main** (**String**[] *args*)

### **RunArcRail**

public class **RunArcRail**

## Methods

### main

```
public static void main (String[] args)
```

## 1.2.10 nismod.transport.showcase

### CapacityUtilisationLegend

```
public class CapacityUtilisationLegend extends JPanel
```

Capacity utilisation legend to include in the dashboards.

**Author** Milan Lovric

## Fields

### LEGEND\_FONT

```
public static final Font LEGEND_FONT
```

## Constructors

### CapacityUtilisationLegend

```
public CapacityUtilisationLegend ()
```

Create the panel.

### CongestionChargingDashboard

```
public class CongestionChargingDashboard extends JFrame
```

Dashboard for the road expansion policy intervention.

**Author** Milan Lovric

## Fields

### AFTER\_MAP\_X

```
public static final int AFTER_MAP_X
```

### AFTER\_MAP\_Y

```
public static final int AFTER_MAP_Y
```

## AFTER\_TABLE\_SHIFT

public static final int **AFTER\_TABLE\_SHIFT**

## BEFORE\_MAP\_X

public static final int **BEFORE\_MAP\_X**

## BEFORE\_MAP\_Y

public static final int **BEFORE\_MAP\_Y**

## BETWEEN\_MAP\_SPACE

public static final int **BETWEEN\_MAP\_SPACE**

## COMBOBOX\_BORDER

public static final [Border](#) **COMBOBOX\_BORDER**

## EMPTY\_BORDER

public static final [Border](#) **EMPTY\_BORDER**

## LEFT\_MARGIN

public static final int **LEFT\_MARGIN**

## MAP\_HEIGHT

public static final int **MAP\_HEIGHT**

## MAP\_WIDTH

public static final int **MAP\_WIDTH**

## MATRIX\_SCALING\_FACTOR

public static final double **MATRIX\_SCALING\_FACTOR**

## OPACITY\_FACTOR

public static final double **OPACITY\_FACTOR**



## RUN\_BUTTON\_BORDER

```
public static final Border RUN_BUTTON_BORDER
```

## SECOND\_MARGIN

```
public static final int SECOND_MARGIN
```

## TABLE\_BORDER

```
public static final Border TABLE_BORDER
```

## TABLE\_FONT

```
public static final Font TABLE_FONT
```

## TABLE\_LABEL\_WIDTH

```
public static final int TABLE_LABEL_WIDTH
```

## TABLE\_ROW\_HEIGHT

```
public static final int TABLE_ROW_HEIGHT
```

## TOTAL\_DEMAND\_BORDER

```
public static final Border TOTAL_DEMAND_BORDER
```

## Constructors

### CongestionChargingDashboard

```
public CongestionChargingDashboard()  
    Create the frame.
```

#### Throws

- [IOException](#) – if any.
- [AWTException](#) – if any.

## Methods

### main

```
public static void main(String[] args)  
    Launch the application.
```

### Parameters

- **args** – Arguments.

## LandingGUI

public class **LandingGUI**  
Main GUI for the Show-case Demo.  
**Author** Milan Lovric

### Fields

#### **BUTTON\_HEIGHT**

public static final int **BUTTON\_HEIGHT**

#### **BUTTON\_SPACE**

public static final int **BUTTON\_SPACE**

#### **BUTTON\_WIDTH**

public static final int **BUTTON\_WIDTH**

#### **BUTTON\_X**

public static final int **BUTTON\_X**

#### **BUTTON\_Y**

public static final int **BUTTON\_Y**

#### **CREDITS\_FONT\_SIZE**

public static final int **CREDITS\_FONT\_SIZE**

#### **DARK\_GRAY**

public static final [Color](#) **DARK\_GRAY**

#### **DASHBOARD**

public static final [Color](#) **DASHBOARD**

### ICON1\_HEIGHT

public static final int **ICON1\_HEIGHT**

### ICON1\_WIDTH

public static final int **ICON1\_WIDTH**

### ICON2\_HEIGHT

public static final int **ICON2\_HEIGHT**

### ICON2\_WIDTH

public static final int **ICON2\_WIDTH**

### ICON3\_HEIGHT

public static final int **ICON3\_HEIGHT**

### ICON3\_WIDTH

public static final int **ICON3\_WIDTH**

### LABEL1\_Y

public static final int **LABEL1\_Y**

### LABEL2\_Y

public static final int **LABEL2\_Y**

### LABEL3\_Y

public static final int **LABEL3\_Y**

### LABEL\_HEIGHT

public static final int **LABEL\_HEIGHT**

### LABEL\_WIDTH

public static final int **LABEL\_WIDTH**

## **LABEL\_X**

public static final int **LABEL\_X**

## **LIGHT\_GRAY**

public static final [Color](#) **LIGHT\_GRAY**

## **MAIN\_TITLE\_FONT\_SIZE**

public static final int **MAIN\_TITLE\_FONT\_SIZE**

## **MID\_GRAY**

public static final [Color](#) **MID\_GRAY**

## **PASTEL\_BLUE**

public static final [Color](#) **PASTEL\_BLUE**

## **PASTEL\_GREEN**

public static final [Color](#) **PASTEL\_GREEN**

## **PASTEL\_YELLOW**

public static final [Color](#) **PASTEL\_YELLOW**

## **SCREEN\_HEIGHT**

public static final int **SCREEN\_HEIGHT**

## **SCREEN\_WIDTH**

public static final int **SCREEN\_WIDTH**

## **SUBTITLE\_FONT\_SIZE**

public static final int **SUBTITLE\_FONT\_SIZE**

## **TOOLBAR**

public static final [Color](#) **TOOLBAR**

## counter

public static int **counter**

## Constructors

### LandingGUI

public **LandingGUI** ()  
Create the application.

## Methods

### main

public static void **main** (*String*[] *args*)  
Launch the application.

#### Parameters

- **args** – Arguments.

## NetworkVisualiserDemo

public class **NetworkVisualiserDemo**  
For visualising the road network.  
**Author** Milan Lovric

## Fields

### ROAD\_LINK\_WIDTH

public static final float **ROAD\_LINK\_WIDTH**

## Constructors

### NetworkVisualiserDemo

protected **NetworkVisualiserDemo** ()

## Methods

### getMap

public static *JFrame* **getMap** (*URL* *zonesUrl*, *URL* *networkUrl*, *URL* *nodesUrl*, *URL* *AADFurl*, *String* *mapTitle*, *String* *linkDataLabel*)  
Visualises the road network with link data.

**Parameters**

- **zonesUrl** – Url for the zones shapefile.
- **networkUrl** – Url for the road network shapefile.
- **nodesUrl** – Url for the nodes shapefile.
- **AADFurl** – Url for the traffic counts shapefile.
- **mapTitle** – Map title for the window.
- **linkDataLabel** – Label describing the link data used.

**Throws**

- **IOException** – if any.

**Returns** JFrame with the map.

**main**

public static void **main** (*String[] args*)

**Parameters**

- **args** – Arguments.

**Throws**

- **IOException** – if any.

**visualise**

public static void **visualise** (*RoadNetwork roadNetwork, String mapTitle*)

Visualises the road network as loaded from the shapefiles.

**Parameters**

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.

**Throws**

- **IOException** – if any.

**visualise**

public static JFrame **visualise** (*URL zonesUrl, URL networkUrl, URL nodesUrl, URL AADFurl, String mapTitle, String linkDataLabel*)

Visualises the road network with link data.

**Parameters**

- **zonesUrl** – Url for the zones shapefile.
- **networkUrl** – Url for the road network shapefile.
- **nodesUrl** – Url for the nodes shapefile.
- **AADFurl** – Url for the traffic counts shapefile.

- **mapTitle** – Map title for the window.
- **linkDataLabel** – Label describing the link data used.

**Throws**

- **IOException** – if any.

**Returns** JFrame with the map.

## visualise

```
public static JFrame visualise (RoadNetwork roadNetwork, String mapTitle, Map<Integer, Double> link-  
Data, String linkDataLabel, String shapefilePath)
```

Visualises the road network with dailyVolume.

**Parameters**

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.
- **linkData** – Data used to classify and colour road links.
- **linkDataLabel** – Label describing the link data used.
- **shapefilePath** – The path to the shapefile into which data will be stored.

**Throws**

- **IOException** – if any.

**Returns** JFrame with the map.

## visualise

```
public static JFrame visualise (RoadNetwork roadNetwork, String mapTitle, Map<Integer, Double> link-  
Data, String linkDataLabel, String shapefilePath, URL congestionCharge-  
ZoneUrl)
```

Visualises the road network with link data and congestion charging zone.

**Parameters**

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.
- **linkData** – Data used to classify and colour road links.
- **linkDataLabel** – Label describing the link data used.
- **shapefilePath** – The path to the shapefile into which data will be stored.
- **congestionChargeZoneUrl** – The path to the shapefile with the congestion charge zone boundary.

**Throws**

- **IOException** – if any.

**Returns** JFrame with the map.

## RoadDevelopmentDashboard

public class **RoadDevelopmentDashboard** extends [JFrame](#)  
Dashboard for the road development policy intervention.

**Author** Milan Lovric

### Fields

#### AFTER\_MAP\_X

public static final int **AFTER\_MAP\_X**

#### AFTER\_MAP\_Y

public static final int **AFTER\_MAP\_Y**

#### BEFORE\_MAP\_X

public static final int **BEFORE\_MAP\_X**

#### BEFORE\_MAP\_Y

public static final int **BEFORE\_MAP\_Y**

#### BETWEEN\_MAP\_SPACE

public static final int **BETWEEN\_MAP\_SPACE**

#### COMBOBOX\_BORDER

public static final [Border](#) **COMBOBOX\_BORDER**

#### EMPTY\_BORDER

public static final [Border](#) **EMPTY\_BORDER**

#### LEFT\_MARGIN

public static final int **LEFT\_MARGIN**

#### MAP\_HEIGHT

public static final int **MAP\_HEIGHT**



## MAP\_WIDTH

public static final int **MAP\_WIDTH**

## MATRIX\_SCALING\_FACTOR

public static final double **MATRIX\_SCALING\_FACTOR**

## OPACITY\_FACTOR

public static final double **OPACITY\_FACTOR**

## RUN\_BUTTON\_BORDER

public static final [Border](#) **RUN\_BUTTON\_BORDER**

## SECOND\_MARGIN

public static final int **SECOND\_MARGIN**

## TABLE\_BORDER

public static final [Border](#) **TABLE\_BORDER**

## TABLE\_FONT

public static final [Font](#) **TABLE\_FONT**

## TABLE\_LABEL\_WIDTH

public static final int **TABLE\_LABEL\_WIDTH**

## TABLE\_ROW\_HEIGHT

public static final int **TABLE\_ROW\_HEIGHT**

## TOTAL\_DEMAND\_BORDER

public static final [Border](#) **TOTAL\_DEMAND\_BORDER**

## Constructors

### RoadDevelopmentDashboard

public **RoadDevelopmentDashboard** ()  
Create the frame.

#### Throws

- **IOException** – if any.
- **AWTException** – if any.

## Methods

### main

public static void **main** (*String[] args*)  
Launch the application.

#### Parameters

- **args** – Arguments.

### RoadExpansionDashboard

public class **RoadExpansionDashboard** extends *JFrame*  
Dashboard for the road expansion policy intervention.

**Author** Milan Lovric

## Fields

### AFTER\_MAP\_X

public static final int **AFTER\_MAP\_X**

### AFTER\_MAP\_Y

public static final int **AFTER\_MAP\_Y**

### BEFORE\_MAP\_X

public static final int **BEFORE\_MAP\_X**

### BEFORE\_MAP\_Y

public static final int **BEFORE\_MAP\_Y**

## **BETWEEN\_MAP\_SPACE**

public static final int **BETWEEN\_MAP\_SPACE**

## **COMBOBOX\_BORDER**

public static final [Border](#) **COMBOBOX\_BORDER**

## **EMPTY\_BORDER**

public static final [Border](#) **EMPTY\_BORDER**

## **FLAG\_EXPAND\_MORE\_LINKS**

public static boolean **FLAG\_EXPAND\_MORE\_LINKS**

## **LEFT\_MARGIN**

public static final int **LEFT\_MARGIN**

## **MAP\_HEIGHT**

public static final int **MAP\_HEIGHT**

## **MAP\_WIDTH**

public static final int **MAP\_WIDTH**

## **MATRIX\_SCALING\_FACTOR**

public static final double **MATRIX\_SCALING\_FACTOR**

## **OPACITY\_FACTOR**

public static final double **OPACITY\_FACTOR**

## **RUN\_BUTTON\_BORDER**

public static final [Border](#) **RUN\_BUTTON\_BORDER**

## **SECOND\_MARGIN**

public static final int **SECOND\_MARGIN**

## TABLE\_BORDER

public static final [Border](#) **TABLE\_BORDER**

## TABLE\_FONT

public static final [Font](#) **TABLE\_FONT**

## TABLE\_LABEL\_WIDTH

public static final int **TABLE\_LABEL\_WIDTH**

## TABLE\_ROW\_HEIGHT

public static final int **TABLE\_ROW\_HEIGHT**

## TOTAL\_DEMAND\_BORDER

public static final [Border](#) **TOTAL\_DEMAND\_BORDER**

## Constructors

### RoadExpansionDashboard

public **RoadExpansionDashboard** ()  
Create the frame.

#### Throws

- [IOException](#) – if any.
- [AWTException](#) – if any.

## Methods

### main

public static void **main** ([String](#)[] *args*)  
Launch the application.

#### Parameters

- **args** – Arguments.

## TableChangeLegend

public class **TableChangeLegend** extends [JPanel](#)  
Table change legend (vertical) to include in the dashboards.

**Author** Milan Lovric

## Fields

### LEGEND\_FONT

public static final `Font` **LEGEND\_FONT**

## Constructors

### TableChangeLegend

public **TableChangeLegend** ()  
Create the panel.

### TableChangeLegendHorizontal

public class **TableChangeLegendHorizontal** extends `JPanel`  
Table change legend (horizontal) to include in the dashboards.  
**Author** Milan Lovric

## Fields

### LEGEND\_FONT

public static final `Font` **LEGEND\_FONT**

## Constructors

### TableChangeLegendHorizontal

public **TableChangeLegendHorizontal** ()  
Create the panel.

## 1.2.11 nismod.transport.utility

### ConfigReader

public class **ConfigReader**  
Configuration file reader.  
**Author** Milan Lovric

## Constructors

### ConfigReader

public **ConfigReader** ()

## Methods

### getProperties

public static [Properties](#) **getProperties** ([String](#) *configFile*)

Reads properties from the configuration file.

#### Parameters

- **configFile** – Path to the configuration file.

**Returns** Loaded properties.

### InputFileReader

public class **InputFileReader**

InputFileReader reads input files and provides them as various data structures required by other classes.

**Author** Milan Lovric

## Constructors

### InputFileReader

public **InputFileReader** ()

## Methods

### readAVFractionsFile

public static [HashMap](#)<[Integer](#), [Map](#)<[VehicleType](#), [Double](#)>> **readAVFractionsFile** ([String](#) *fileName*)

Reads autonomous vehicles fractions file.

#### Parameters

- **fileName** – File name.

**Returns** Map with predictions of autonomous vehicles fractions.

### readAirElasticitiesFile

public static [Map](#)<[AirDemandModel.ElasticityTypes](#), [Double](#)> **readAirElasticitiesFile** ([String](#) *fileName*)

Reads air elasticities file.

#### Parameters

- **fileName** – File name.

**Returns** Map with elasticity parameters.

### readAirportFareIndexFile

```
public static HashMap<Integer, HashMap<String, Double>> readAirportFareIndexFile (String fileName)
```

Reads airport fare index file.

#### Parameters

- **fileName** – File name.

**Returns** Map with airport fare indices.

### readDomesticAirportsFile

```
public static Map<String, Airport> readDomesticAirportsFile (String fileName)
```

Reads domestic airports file.

#### Parameters

- **fileName** – File name.

**Returns** Mapping between IATA code and airport information.

### readElasticitiesFile

```
public static Map<ElasticityTypes, Double> readElasticitiesFile (String fileName)
```

Reads elasticities file.

#### Parameters

- **fileName** – File name.

**Returns** Map with elasticity parameters.

### readEnergyConsumptionParamsFile

```
public static Map<VehicleType, Map<EngineType, Map<WebTAG, Double>>> readEnergyConsumptionParamsFile (String fileName)
```

Reads engine type fractions file.

#### Parameters

- **fileName** – File name.

**Returns** Map with engine type fractions.

### readEnergyUnitCostsFile

```
public static HashMap<Integer, Map<EnergyType, Double>> readEnergyUnitCostsFile (String fileName)
```

Reads energy unit costs file.

#### Parameters

- **fileName** – File name.

**Returns** Map with energy unit costs.

### readEngineTypeFractionsFile

```
public static HashMap<Integer, Map<VehicleType, Map<EngineType, Double>>> readEngineTypeFractionsFile (String  
file-  
Name)
```

Reads engine type fractions file.

#### Parameters

- **fileName** – File name.

**Returns** Map with engine type fractions.

### readFreightTripRatesFile

```
public static HashMap<Integer, Map<VehicleType, Double>> readFreightTripRatesFile (String file-  
Name)
```

Reads freight trip rates file.

#### Parameters

- **fileName** – File name.

**Returns** Map with yearly trip rates for freight vehicles.

### readGVAFile

```
public static HashMap<Integer, HashMap<String, Double>> readGVAFile (String fileName)  
Reads GVA file.
```

#### Parameters

- **fileName** – File name.

**Returns** Map with GVA data.

### readInternationalAirportsFile

```
public static Map<String, Airport> readInternationalAirportsFile (String fileName)  
Reads international airports file.
```

#### Parameters

- **fileName** – File name.

**Returns** Mapping between IATA code and airport information.

### readLinkTravelTimeFile

```
public static Map<TimeOfDay, Map<Integer, Double>> readLinkTravelTimeFile (int year, String file-  
Name)
```

Reads link travel time file.

#### Parameters

- **year** – Year of the assignment.
- **fileName** – File name.



**Returns** Link travel time per time of day.

### readPopulationFile

```
public static HashMap<Integer, HashMap<String, Integer>> readPopulationFile (String fileName)
```

Reads population file.

#### Parameters

- **fileName** – File name.

**Returns** Map with population data.

### readRailElasticitiesFile

```
public static Map<RailDemandModel.ElasticityTypes, Map<RailDemandModel.ElasticityArea, Double>> readRailElasticitiesFile (String fileName)
```

Reads rail elasticities file.

#### Parameters

- **fileName** – File name.

**Returns** Map with elasticity parameters.

### readRailStationCostsFile

```
public static HashMap<Integer, HashMap<Integer, Double>> readRailStationCostsFile (String fileName)
```

Reads rail station costs file.

#### Parameters

- **fileName** – File name.

**Returns** Map with rail journey costs.

### readRelativeFuelEfficiencyFile

```
public static HashMap<Integer, Map<VehicleType, Map<EngineType, Double>>> readRelativeFuelEfficiencyFile (String fileName)
```

Reads relative fuel efficiency file.

#### Parameters

- **fileName** – File name.

**Returns** Map with relative fuel efficiency.

### readTimeOfDayDistributionFile

```
public static Map<Integer, Map<TimeOfDay, Double>> readTimeOfDayDistributionFile (String  
file-  
Name)
```

Reads time of day distribution file for passenger car vehicles.

#### Parameters

- **fileName** – File name.

**Returns** Time of day distribution.

### readTimeOfDayDistributionFreightFile

```
public static Map<Integer, Map<VehicleType, Map<TimeOfDay, Double>>> readTimeOfDayDistributionFreightFile (Str  
file  
Na
```

Reads time of day distribution file for freight vehicles.

#### Parameters

- **fileName** – File name.

**Returns** Time of day distribution.

### readTripRatesFile

```
public static HashMap<Integer, Double> readTripRatesFile (String fileName)  
Reads trip rates file.
```

#### Parameters

- **fileName** – File name.

**Returns** Map with yearly trip rates.

### readUnitCO2EmissionFile

```
public static HashMap<Integer, Map<EnergyType, Double>> readUnitCO2EmissionFile (String file-  
Name)
```

Reads unit CO2 emissions file.

#### Parameters

- **fileName** – File name.

**Returns** Map with unit CO2 emissions.

### readVehicleTypeToPCUFile

```
public static Map<VehicleType, Double> readVehicleTypeToPCUFile (String fileName)  
Reads vehicle type to PCU conversion file.
```

#### Parameters

- **fileName** – File name.

**Returns** Map with PCU equivalents.

### readZonalCarCostsFile

```
public static HashMap<Integer, HashMap<String, Double>> readZonalCarCostsFile (String file-  
Name)
```

Reads zonal car journey costs file.

**Parameters**

- **fileName** – File name.

**Returns** Map with cost data.

### readZonalCarEnergyConsumptionsFile

```
public static HashMap<Integer, HashMap<EnergyType, HashMap<String, Double>>> readZonalCarEnergyConsumptionsFile
```

Reads zonal car energy consumptions file.

**Parameters**

- **fileName** – File name.

**Returns** Map with zonal energy consumptions data.

### readZonalVehicleCO2EmissionsFile

```
public static HashMap<Integer, HashMap<VehicleType, HashMap<String, Double>>> readZonalVehicleCO2EmissionsFile
```

Reads zonal vehicle CO2 emissions file.

**Parameters**

- **fileName** – File name.

**Returns** Map with CO2 emissions data.

## PropertiesReader

```
public class PropertiesReader
```

Properties file reader.

**Author** Milan Lovric

### Constructors

#### PropertiesReader

```
public PropertiesReader ()
```

## Methods

### getProperties

public static [Properties](#) **getProperties** (*String configFile*)

Reads properties from the configuration file.

#### Parameters

- **configFile** – Path to the configuration file.

**Returns** Loaded properties.

### RandomSingleton

public class **RandomSingleton**

Creates only one instance of the random number generator that can be used throughout the whole model. Simulation results can then be reproduced by using the same seed.

**Author** Milan Lovric

## Methods

### getInstance

public static [RandomSingleton](#) **getInstance** ()

Getter for the singleton instance of the random number generator.

**Returns** Random number generator.

### nextDouble

public double **nextDouble** ()

Generates a pseudorandom real number between 0 and 1.

**Returns** Pseudorandom real double.

### nextInt

public int **nextInt** (int *bound*)

Generates a pseudorandom whole number smaller than the bound.

#### Parameters

- **bound** – Upper bound.

**Returns** Pseudorandom whole number.

### setSeed

public void **setSeed** (long *seed*)

Setter method for the seed of the random number generator.

### Parameters

- **seed** – Seed of the random number generator.

## 1.2.12 nismod.transport.visualisation

### BarVisualiser

public class **BarVisualiser** extends `JFrame`  
For visualising bar charts using JFreeChart.

**Author** Milan Lovric

### Constructors

#### BarVisualiser

public **BarVisualiser** (DefaultCategoryDataset *dataset*, `String` *title*, `String` *paletteName*, boolean *invert-  
Colours*)

### Methods

#### main

public static void **main** (`String`[] *args*)

#### saveToPNG

public void **saveToPNG** (`String` *fileName*)

### LineVisualiser

public class **LineVisualiser** extends `JFrame`  
For visualising pie charts using JFreeChart.

**Author** Milan Lovric

### Constructors

#### LineVisualiser

public **LineVisualiser** (DefaultCategoryDataset *dataset*, `String` *title*)

### Methods

#### main

public static void **main** (`String`[] *args*)

## saveToPNG

```
public void saveToPNG (String fileName)
```

## NetworkVisualiser

```
public class NetworkVisualiser  
    For visualising the road network.  
  
    Author Milan Lovric
```

## Fields

### ROAD\_LINK\_WIDTH

```
public static final float ROAD_LINK_WIDTH
```

## Constructors

### NetworkVisualiser

```
protected NetworkVisualiser ()
```

## Methods

### visualise

```
public static void visualise (RoadNetwork roadNetwork, String mapTitle)  
    Visualises the road network as loaded from the shapefiles.
```

#### Parameters

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.

#### Throws

- **IOException** – if any.

### visualise

```
public static void visualise (RoadNetwork roadNetwork, String mapTitle, Map<Integer, Double> linkData,  
                             String linkDataLabel, String shapefilePath)  
    Visualises the road network with dailyVolume.
```

#### Parameters

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.
- **linkData** – Data used to classify and colour road links.

- **linkDataLabel** – Label describing the link data used.
- **shapefilePath** – The path to the shapefile into which data will be stored.

**Throws**

- **IOException** – if any.

**visualise**

public static void **visualise** (*RoadNetwork* roadNetwork, *String* mapTitle, *Map*<*Integer*, *Double*> linkData, *String* linkDataLabel, *String* shapefilePath, *URL* congestionChargeZoneUrl)  
Visualises the road network with dailyVolume.

**Parameters**

- **roadNetwork** – Road network.
- **mapTitle** – Map title for the window.
- **linkData** – Data used to classify and colour road links.
- **linkDataLabel** – Label describing the link data used.
- **shapefilePath** – The path to the shapefile into which data will be stored.
- **congestionChargeZoneUrl** – The path to the shapefile with the congestion charge zone boundary.

**Throws**

- **IOException** – if any.

**PieChartVisualiser**

public class **PieChartVisualiser** extends *JFrame*  
For visualising pie charts using JFreeChart.

**Author** Milan Lovric

**Constructors****PieChartVisualiser**

public **PieChartVisualiser** (*DefaultPieDataset* dataset, *String* title, *String* paletteName, boolean threeD)

**Methods****main**

public static void **main** (*String*[] args)

**saveToPNG**

public void **saveToPNG** (*String* fileName)

### 1.2.13 nismod.transport.zone

#### Zoning

public class **Zoning**

For mapping Tempro zones to the nodes of the road network.

**Author** Milan Lovric

#### Fields

##### MAX\_NEAREST\_NODES

public static int **MAX\_NEAREST\_NODES**

##### TOP\_LAD\_NODES

public static int **TOP\_LAD\_NODES**

#### Constructors

#### Zoning

public **Zoning** (*URL zonesUrl*, *URL nodesUrl*, *RoadNetwork rn*, *Properties params*)

Constructor for the zoning system.

##### Parameters

- **zonesUrl** – Url for the zones shapefile.
- **nodesUrl** – Url for the nodes shapefile.
- **rn** – Road network.
- **params** – Properties file with parameters.

##### Throws

- **IOException** – if any.

#### Methods

##### getAccessEgressFactor

public double **getAccessEgressFactor** ()

Getter for access/egress scaling factor.

**Returns** Access/egress scaling factor.



### getLADToListOfContainedZones

```
public HashMap<String, List<String>> getLADToListOfContainedZones ()
```

Getter for LAD to list of contained Tempro zones mapping.

**Returns** LAD to list of contained zones.

### getLADToName

```
public HashMap<String, String> getLADToName ()
```

Getter for LAD code to LAD name mapping.

**Returns** LAD code to LAD name mapping.

### getLadCodeToIDMap

```
public HashMap<String, Integer> getLadCodeToIDMap ()
```

Getter for LAD zone ONS code to ID.

**Returns** LAD zone ONS code to LAD zone ID map.

### getLadIDToCodeMap

```
public String[] getLadIDToCodeMap ()
```

Getter for LAD zone ID to ONS code.

**Returns** LAD zone ID to LAD zone ONS code.

### getNodeToZoneMap

```
public HashMap<Integer, String> getNodeToZoneMap ()
```

Getter for node to zone mapping (for each node gives the zone in which it is located).

**Returns** Node to zone map.

### getTemproCodeToIDMap

```
public HashMap<String, Integer> getTemproCodeToIDMap ()
```

Getter for Tempro zone ONS code to ID.

**Returns** Tempro zone code to Tempro zone ID map.

### getTemproIDToCodeMap

```
public String[] getTemproIDToCodeMap ()
```

Getter for LAD zone ID to ONS code.

**Returns** Tempro zone ID to Tempro zone ONS code.

### getZoneIDToLadID

```
public int[] getZoneIDToLadID ()
```

Getter for Tempro zone ID to LAD zone ID mapping.

**Returns** Tempro zone ID to LAD zone ID array.

### getZoneIDToNearestNodeDistanceMap

```
public double[] getZoneIDToNearestNodeDistanceMap ()
```

Getter for zone ID to nearest node distance mapping (in meters).

**Returns** Zone to distance map.

### getZoneIDToNearestNodeIDFromLADTopNodesMap

```
public int[] getZoneIDToNearestNodeIDFromLADTopNodesMap ()
```

Getter for zone ID to nearest node ID among top LAD nodes mapping.

**Returns** Zone to node map.

### getZoneIDToNearestNodeIDMap

```
public int[] getZoneIDToNearestNodeIDMap ()
```

Getter for Tempro zone ID to nearest node ID mapping.

**Returns** Zone to node map.

### getZoneToCentroid

```
public HashMap<String, Point> getZoneToCentroid ()
```

Getter for Tempro zone to its centroid mapping.

**Returns** Tempro zone to centroid mapping.

### getZoneToLADMap

```
public HashMap<String, String> getZoneToLADMap ()
```

Getter for Tempro zone to LAD zone mapping.

**Returns** Tempro zone to LAD zone map.

### getZoneToListOfContainedNodes

```
public HashMap<String, List<Integer>> getZoneToListOfContainedNodes ()
```

Getter for Tempro zone to list of contained nodes mapping.

**Returns** Zone to list of contained nodes.

### getZoneToMinMaxDimension

```
public double[][] getZoneToMinMaxDimension ()
```

Getter for Tempro zone ID to min [0] and max [1] dimension of the zone bounding box (envelope) [in km].

**Returns** Zone min and max dimension (width or height).

### getZoneToNearestNodeDistanceMap

```
public HashMap<String, Double> getZoneToNearestNodeDistanceMap ()
```

Getter for zone centroid to nearest node distance mapping (in meters).

**Returns** Zone to distance map.

### getZoneToNearestNodeIDFromLADTopNodesMap

```
public HashMap<String, Integer> getZoneToNearestNodeIDFromLADTopNodesMap ()
```

Getter for zone centroid to nearest node ID among top LAD nodes mapping.

**Returns** Zone to node map.

### getZoneToNearestNodeIDMap

```
public HashMap<String, Integer> getZoneToNearestNodeIDMap ()
```

Getter for Tempro zone centroid to nearest node ID mapping.

**Returns** Zone to node map.

### getZoneToNodeDistanceMatrix

```
public double[][] getZoneToNodeDistanceMatrix ()
```

Getter for Tempro zone to all nodes distance matrix [in metres].

**Returns** Zone to node distance matrix.

### getZoneToSortedListOfNodeAndDistancePairs

```
public HashMap<String, List<Pair<Integer, Double>>> getZoneToSortedListOfNodeAndDistancePairs ()
```

Getter for Tempro zone to sorted node distances mapping (distances to ALL nodes in the network).

**Returns** Zone to sorted list of nodes and distances.

### getZoneToZoneDistanceMatrix

```
public double[][] getZoneToZoneDistanceMatrix ()
```

Getter for Tempro zone (centroid) to Tempro zone (centroid) distance matrix [in metres].

**Returns** Zone to node distance matrix.

## 1.3 Authors

### 1.3.1 Development

- Milan Lovric <M.Lovric@soton.ac.uk>

### 1.3.2 Integration

- Will Usher <william.usher@ouce.ox.ac.uk>
- Tom Russell <tom.russell@ouce.ox.ac.uk>
- Roald Schoenmakers <roald.schoenmakers@ouce.ox.ac.uk>
- Thibault Lestang <thibault.lestang@cs.ox.ac.uk>

### 1.3.3 Management

- Simon Blainey <S.P.Blainey@soton.ac.uk>
- John Preston <J.M.Preston@soton.ac.uk>

## 1.4 License

MIT License

Copyright (c) 2018 National Infrastructure Systems Model

Permission **is** hereby granted, free of charge, to **any** person obtaining a copy of this software **and** associated documentation files (the "**Software**"), to deal **in** the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, **and/or** sell copies of the Software, **and** to permit persons to whom the Software **is** furnished to do so, subject to the following conditions:

The above copyright notice **and** this permission notice shall be included **in all** copies **or** substantial portions of the Software.

THE SOFTWARE IS PROVIDED "**AS IS**", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## CHAPTER 2

---

### Indices and tables

---

- `genindex`
- `modindex`
- `search`



## A

A (*Java field*), 160addCongestionCharges(int, PricingPolicy) (*Java method*), 36addEdge(DirectedEdge) (*Java method*), 155addEdgeWithoutValidityCheck(DirectedEdge) (*Java method*), 155addEdgeWithoutValidityCheck(int) (*Java method*), 155addNLCoFDevelopedStation(int) (*Java method*), 200addRoadLink(Edge) (*Java method*), 111addRoute(Route) (*Java method*), 161, 166addRouteWithoutAnyChecks(Route) (*Java method*), 161addRouteWithoutValidityAndEndNodesCheck(Route) (*Java method*), 161addRouteWithoutValidityCheck(Route) (*Java method*), 161, 166addStation(RailStation) (*Java method*), 207addYearOfDevelopment(int) (*Java method*), 200AF (*Java field*), 18AFTER\_MAP\_X (*Java field*), 211, 220, 222AFTER\_MAP\_Y (*Java field*), 211, 220, 222AFTER\_TABLE\_SHIFT (*Java field*), 212AirDemandModel (*Java class*), 11AirDemandModel(String, String, String, String, String, String, String, String, String, List, Properties) (*Java constructor*), 12Airport (*Java class*), 14Airport(Airport) (*Java constructor*), 15Airport(String, String, String, double, double, String, String, long, long) (*Java constructor*), 15AirportGroup (*Java enum*), 17AirportGroupCAA (*Java enum*), 17alpha (*Java field*), 121AN (*Java field*), 18App (*Java class*), 11ArcAnalysis (*Java class*), 210arcZones (*Java field*), 210AROAD (*Java field*), 120ARTIC (*Java field*), 153ARTIC\_AV (*Java field*), 153AS (*Java field*), 18AssignableODMatrix (*Java interface*), 34assignAndCalculateRMSN() (*Java method*), 81, 82, 84assignBaseYear() (*Java method*), 37assignFlowsAndUpdateLinkTravelTimes(AssignableODMatrix, FreightMatrix, RouteSetGenerator, Properties, double) (*Java method*), 124assignFlowsAndUpdateLinkTravelTimes(AssignableODMatrix, FreightMatrix, RouteSetGenerator, Zoning, Properties, double) (*Java method*), 124assignFlowsAndUpdateLinkTravelTimesIterated(AssignableODMatrix, FreightMatrix, RouteSetGenerator, Properties, double, int) (*Java method*), 124assignFlowsAndUpdateLinkTravelTimesIterated(AssignableODMatrix, FreightMatrix, RouteSetGenerator, Zoning, Properties, double, int) (*Java method*), 125assignFreightFlowsHourlyRouting(FreightMatrix, Map, Properties) (*Java method*), 125assignFreightFlowsRouteChoice(FreightMatrix, RouteSetGenerator, Properties) (*Java method*), 126assignFreightFlowsRouting(FreightMatrix, RouteSetGenerator, Properties) (*Java method*), 126assignmentFraction (*Java field*), 121assignmentIterations (*Java field*), 35

assignPassengerFlowsHourlyRouting(AssignableODMatrix, Map, Properties) (*Java method*), 127  
 assignPassengerFlowsRouteChoice(AssignableODMatrix, RouteSetGenerator, Properties) (*Java method*), 127  
 assignPassengerFlowsRouteChoiceTemporality(AssignableODMatrix, Zoning, RouteSetGenerator, Properties) (*Java method*), 127  
 assignPassengerFlowsRouteChoiceTemporalityDistanceBased(AssignableODMatrix, Zoning, RouteSetGenerator, Properties) (*Java method*), 127  
 assignPassengerFlowsRouting(AssignableODMatrix, TimeOfDay, double[], Map, Map, RouteSetGenerator, Properties) (*Java method*), 128  
 assignPassengerFlowsTemporality(AssignableODMatrix, Zoning, RouteSetGenerator, Properties) (*Java method*), 128  
 ATLANTIC\_OCEAN\_ISLANDS (*Java field*), 19  
 AUSTRALASIA (*Java field*), 19  
 averageAccessEgressSpeedCar (*Java field*), 121  
 averageAccessEgressSpeedFreight (*Java field*), 121  
 averageIntersectionDelay (*Java field*), 121  
 averageSpeedFerry (*Java field*), 109

## B

B (*Java field*), 160  
 BarVisualiser (*Java class*), 233  
 BarVisualiser(DefaultCategoryDataset, String, String, boolean) (*Java constructor*), 233  
 baseYear (*Java field*), 11, 35, 121, 199  
 baseYearFreight (*Java field*), 35  
 BEFORE\_MAP\_X (*Java field*), 212, 220, 222  
 BEFORE\_MAP\_Y (*Java field*), 212, 220, 222  
 betaARoad (*Java field*), 121  
 betaMRoad (*Java field*), 122  
 BETWEEN\_MAP\_SPACE (*Java field*), 212, 220, 223  
 BEV (*Java field*), 149  
 BIN\_LIMITS\_KM (*Java field*), 42  
 BIN\_LIMITS\_MILES (*Java field*), 42  
 buildEdges() (*Java method*), 154  
 BUTTON\_HEIGHT (*Java field*), 214  
 BUTTON\_SPACE (*Java field*), 214  
 BUTTON\_WIDTH (*Java field*), 214  
 BUTTON\_X (*Java field*), 214  
 BUTTON\_Y (*Java field*), 214

## C

C (*Java field*), 160  
 calculateAbsDifferenceCarCounts() (*Java method*), 128  
 calculateAllPathSizes() (*Java method*), 166  
 calculateAssignedFreightMatrix() (*Java method*), 129  
 calculateCarEnergyConsumptions() (*Java method*), 129  
 calculateCO2Emissions() (*Java method*), 129  
 calculateConsumption(VehicleType, EngineType, double, double, double, double) (*Java method*), 156  
 calculateCost(VehicleType, EngineType, Map, List) (*Java method*), 156  
 calculateCostSkimMatrix() (*Java method*), 129  
 calculateCostSkimMatrixFreight() (*Java method*), 129  
 calculateDailyZonalUsageAverage() (*Java method*), 207  
 calculateDailyZonalUsageTotal() (*Java method*), 207  
 calculateDifferenceCarCounts() (*Java method*), 129  
 calculateDirectionAveragedAbsoluteDifferenceCarCounts() (*Java method*), 130  
 calculateDirectionAveragedPeakLinkCapacityUtilisation() (*Java method*), 130  
 calculateDistanceSkimMatrix() (*Java method*), 130  
 calculateDistanceSkimMatrixFreight() (*Java method*), 130  
 calculateDistanceSkimMatrixTemporality() (*Java method*), 130  
 calculateEnergyConsumptions() (*Java method*), 130  
 calculateEnergyConsumptionsPerVehicleType() (*Java method*), 131  
 calculateFreightEnergyConsumptions() (*Java method*), 131  
 calculateFreightLADTripEnds() (*Java method*), 131  
 calculateFreightLADTripStarts() (*Java method*), 131  
 calculateGEHStatisticForCarCounts(double) (*Java method*), 131  
 calculateGEHStatisticForFreightCounts(double) (*Java method*), 131  
 calculateGEHStatisticPerTimeOfDay(TimeOfDay) (*Java method*), 132  
 calculateLADTripEnds() (*Java method*), 132  
 calculateLADTripStarts() (*Java method*), 132  
 calculateLength() (*Java method*), 156  
 calculateLinkVolumeInPCU(List) (*Java method*), 132  
 calculateLinkVolumeInPCUPerTimeOfDay(List) (*Java method*), 132



- (*Java method*), 132
- calculateLinkVolumePerVehicleType(List) (*Java method*), 133
- calculateMADforExpandedSimulatedVolumes (*Java method*), 133
- calculateODCarEnergyConsumptions() (*Java method*), 133
- calculatePathsizes() (*Java method*), 162
- calculatePeakLinkCapacityUtilisation() (*Java method*), 133
- calculatePeakLinkDensities() (*Java method*), 133
- calculatePeakLinkPointCapacities() (*Java method*), 133
- calculateProbabilities() (*Java method*), 162
- calculateRMSNforExpandedSimulatedVolumes (*Java method*), 134
- calculateRMSNforFreightCounts() (*Java method*), 134
- calculateRMSNforSimulatedVolumes() (*Java method*), 134
- calculateTimeSkimMatrix() (*Java method*), 134
- calculateTimeSkimMatrixFreight() (*Java method*), 134
- calculateTravelTime(double[], double) (*Java method*), 157
- calculateTripEnds() (*Java method*), 50, 55, 62, 68, 73
- calculateTripStarts() (*Java method*), 50, 56, 62, 68, 74
- calculateUtilities(VehicleType, EngineType, TimeOfDay, double[], Map, Map, Map, List, Map) (*Java method*), 162
- calculateUtility(VehicleType, EngineType, TimeOfDay, double[], Map, Map, Map, List, Map) (*Java method*), 157
- calculateYearlyZonalUsageAverage() (*Java method*), 208
- calculateYearlyZonalUsageTotal() (*Java method*), 208
- calculateZonalCarEnergyConsumptions(double) (*Java method*), 134
- calculateZonalTemporalTripStartsForElectricVehicle() (*Java method*), 135
- calculateZonalTemporalTripStartsForHydrogenVehicle() (*Java method*), 135
- calculateZonalTemporalVehicleElectricityConsumption(double) (*Java method*), 135
- calculateZonalTemporalVehicleHydrogenConsumption(double) (*Java method*), 136
- calculateZonalVehicleCO2Emissions(double) (*Java method*), 136
- calculateZonalVehicleKilometresPerVehicleType() (*Java method*), 136
- calculateZonalVehicleKilometresPerVehicleTypeFromTrips(double, boolean) (*Java method*), 137
- calculateZonalVehicleKilometresPerVehicleTypeFromTrips() (*Java method*), 137
- CANADA (*Java field*), 19
- CapacityUtilisationLegend (*Java class*), 211
- CapacityUtilisationLegend() (*Java constructor*), 211
- CAR (*Java field*), 153
- CAR\_AV (*Java field*), 153
- CARRIBEAN\_AREA (*Java field*), 19
- ceilMatrixValues() (*Java method*), 74
- CENTRAL\_AFRICA (*Java field*), 19
- CENTRAL\_AMERICA (*Java field*), 19
- CHARTER (*Java field*), 26
- choose() (*Java method*), 162
- clearRoutes() (*Java method*), 166
- clone() (*Java method*), 56, 62, 68, 74
- CNG (*Java field*), 148
- COMBOBOX\_BORDER (*Java field*), 212, 220, 223
- ConfigReader (*Java class*), 225
- ConfigReader() (*Java constructor*), 225
- CongestionCharging (*Java class*), 27
- CongestionCharging (*Java field*), 29
- CongestionCharging(Properties) (*Java constructor*), 27
- CongestionCharging(String) (*Java constructor*), 27
- CongestionChargingDashboard (*Java class*), 211
- CongestionChargingDashboard() (*Java constructor*), 213
- contains(Edge) (*Java method*), 157
- contains(int) (*Java method*), 157
- ContinentCode (*Java enum*), 18
- correctUtilityWithPathSize(int) (*Java method*), 162
- COST (*Java field*), 41, 165
- COST\_CAR (*Java field*), 203
- COST\_DOMESTIC (*Java field*), 14
- COST\_INTERNATIONAL (*Java field*), 14
- COST\_RAIL (*Java field*), 203
- createCustomFeatureType(String) (*Java method*), 137
- createCustomFeatureType(VehicleType) (*Java method*), 137
- createLadMatrixFromTEMPProMatrix(ODMatrixArrayTemporal) (*Java method*), 50, 56
- createLadMatrixFromTEMPProMatrix(ODMatrixMultiKey, Zoning) (*Java method*), 62
- createLadMatrixFromTEMPProMatrix(RealODMatrix, Zoning) (*Java method*), 68

[createLadMatrixFromTEMProMatrix\(RealODMatrix, DemandModel, RoadNetwork, String, Zoning\) \(Java method\), 74](#)  
[createListOfStationsWithinEachLAD\(\) \(Java method\), 208](#)  
[createNetworkFeatureCollection\(Map, String, String\) \(Java method\), 111](#)  
[createNewRoadLink\(Node, Node, int, char, double, int\) \(Java method\), 111](#)  
[createSparseUnitMatrix\(Set, HashMap, double\) \(Java method\), 62](#)  
[createTEMProFromLadMatrix\(ODMatrixArray, ODMatrixArrayTempo, Zoning\) \(Java method\), 50, 56](#)  
[createTEMProFromLadMatrix\(ODMatrixMultiKey, ODMatrixArrayTempo, Zoning\) \(Java method\), 56](#)  
[createTEMProFromLadMatrix\(ODMatrixMultiKey, ODMatrixMultiKey, Zoning\) \(Java method\), 63](#)  
[createTEMProFromLadMatrix\(ODMatrixMultiKey, RealODMatrixTempo, Zoning\) \(Java method\), 74](#)  
[createTEMProFromLadMatrix\(RealODMatrix, RealODMatrix, Zoning\) \(Java method\), 68](#)  
[createTEMProFromLadMatrix\(RealODMatrix, RealODMatrixTempo, Zoning\) \(Java method\), 74](#)  
[createUnitBYFMMatrix\(\) \(Java method\), 46](#)  
[createUnitMatrix\(\) \(Java method\), 43, 81, 82, 84](#)  
[createUnitMatrix\(List\) \(Java method\), 63, 69](#)  
[createUnitMatrix\(List, List\) \(Java method\), 46, 63, 69](#)  
[createUnitMatrix\(List, List, Zoning\) \(Java method\), 50, 57, 75](#)  
[createUnitMatrix\(List, Zoning\) \(Java method\), 51, 57, 75](#)  
[createUnitMatrix\(Set\) \(Java method\), 63, 69](#)  
[createUnitMatrix\(Set, Zoning\) \(Java method\), 51, 57, 75](#)  
[createUnitMatrix\(Zoning\) \(Java method\), 57, 75](#)  
[CREDITS\\_FONT\\_SIZE \(Java field\), 214](#)

## D

[D \(Java field\), 160](#)  
[DARK\\_GRAY \(Java field\), 214](#)  
[DASHBOARD \(Java field\), 214](#)  
[data \(Java field\), 26](#)  
[DELAY \(Java field\), 165](#)  
[deleteInterzonalFlows\(int\) \(Java method\), 46](#)  
[deleteInterzonalFlows\(String\) \(Java method\), 43, 51, 57, 63, 69, 76](#)  
[DemandModel \(Java class\), 35](#)

[DemandModel \(RoadNetwork, String, String, String, String, String, String, String, String, String, List, RouteSetGenerator, Zoning, Properties\) \(Java constructor\), 36](#)  
[destination \(Java field\), 175](#)  
[DIESEL \(Java field\), 148](#)  
[Disruption \(Java class\), 106](#)  
[Disruption\(Properties\) \(Java constructor\), 106](#)  
[Disruption\(String\) \(Java constructor\), 106](#)  
[DLR \(Java field\), 206](#)  
[DO \(Java field\), 17](#)  
[DomesticAirport \(Java class\), 21](#)  
[DomesticAirport\(DomesticAirport\) \(Java constructor\), 22](#)  
[DomesticAirport\(String, String, String, String, String, int, int, double, double, String, String, long, long\) \(Java constructor\), 21](#)  
[domesticAirports \(Java field\), 11](#)  
[DomesticInternodalPassengerDemand \(Java class\), 23](#)  
[DomesticInternodalPassengerDemand\(\) \(Java constructor\), 23](#)  
[DomesticInternodalPassengerDemand\(String\) \(Java constructor\), 23](#)

## E

[EAST\\_AFRICA \(Java field\), 19](#)  
[EASTERN\\_EUROPE\\_EU \(Java field\), 19](#)  
[EASTERN\\_EUROPE\\_OTHER \(Java field\), 19](#)  
[EdgeType \(Java enum\), 120](#)  
[EIGHTAM \(Java field\), 150](#)  
[EIGHTPM \(Java field\), 150](#)  
[ElasticityArea \(Java enum\), 202](#)  
[ElasticityTypes \(Java enum\), 14, 41, 202](#)  
[ELECTRICITY \(Java field\), 148](#)  
[ELEVENAM \(Java field\), 150](#)  
[ELEVENPM \(Java field\), 150](#)  
[EMPTY\\_BORDER \(Java field\), 212, 220, 223](#)  
[EnergyType \(Java enum\), 148](#)  
[engine \(Java field\), 175](#)  
[EngineType \(Java enum\), 149](#)  
[equals\(Object\) \(Java method\), 158](#)  
[EstimatedODMatrix \(Java class\), 42](#)  
[EstimatedODMatrix\(HashMap, HashMap, SkimMatrix, double\[\], double\[\]\) \(Java constructor\), 42](#)  
[EstimatedODMatrix\(String, SkimMatrix, double\[\], double\[\]\) \(Java constructor\), 43](#)  
[EU \(Java field\), 17, 18](#)

`exportToShapefile(String)` (*Java method*), 112

## F

`FAR_EAST` (*Java field*), 20

`FCEV_H2` (*Java field*), 149

`FERRY` (*Java field*), 121

`FIVEAM` (*Java field*), 151

`FIVEPM` (*Java field*), 151

`FLAG_EXPAND_MORE_LINKS` (*Java field*), 223

`flagAStarIfEmptyRouteSet` (*Java field*), 122

`flagIncludeAccessEgress` (*Java field*), 122

`flagIntrazonalAssignmentReplacement` (*Java field*), 122

`flagUseRouteChoiceModel` (*Java field*), 122

`floorMatrixValues()` (*Java method*), 76

`ForeignRegionCAA` (*Java enum*), 19

`FOURAM` (*Java field*), 151

`FOURPM` (*Java field*), 151

`freeFlowSpeedARoad` (*Java field*), 109

`freeFlowSpeedMRoad` (*Java field*), 109

`FreightMatrix` (*Java class*), 45

`FreightMatrix()` (*Java constructor*), 45

`FreightMatrix(String)` (*Java constructor*), 45

`freightScalingFactor` (*Java field*), 35

## G

`generateRouteSetBetweenFreightZones(int, int)` (*Java method*), 166

`generateRouteSetBetweenFreightZones(int, int, int)` (*Java method*), 167

`generateRouteSetForFreightMatrix(FreightMatrix, int)` (*Java method*), 167

`generateRouteSetForFreightMatrix(FreightMatrix, int, int)` (*Java method*), 167

`generateRouteSetForFreightMatrix(FreightMatrix, int, int, int)` (*Java method*), 168

`generateRouteSetForODMatrix(ODMatrixMultiKey)` (*Java method*), 168

`generateRouteSetForODMatrix(ODMatrixMultiKey, int)` (*Java method*), 168

`generateRouteSetForODMatrix(ODMatrixMultiKey, int, int)` (*Java method*), 169

`generateRouteSetForODMatrix(ODMatrixMultiKey, int, int, int)` (*Java method*), 168

`generateRouteSetForODMatrixTempo(ODMatrixMultiKey, Zoning)` (*Java method*), 169

`generateRouteSetForODMatrixTempo(RealODMatrixTempo, Zoning, int, int)` (*Java method*), 169

`generateRouteSetForODMatrixTempoDistanceBased(RealODMatrixTempo, Zoning, int, int)` (*Java method*), 169

`generateRouteSetNodeToNode(int, int)` (*Java method*), 170

`generateRouteSetWithLinkElimination(int, int)` (*Java method*), 170

`generateRouteSetWithRandomLinkEliminationRestriction(int)` (*Java method*), 170

`generateRouteSetWithRandomLinkEliminationRestriction(int, int, int)` (*Java method*), 170

`generateRouteSetZoneToZone(String, String)` (*Java method*), 171

`generateRouteSetZoneToZone(String, String, int)` (*Java method*), 171

`generateRouteSetZoneToZoneTempo(String, String, Zoning)` (*Java method*), 171

`generateRouteSetZoneToZoneTempoDistanceBased(String, String, Zoning)` (*Java method*), 171

`generateSingleNodeRoutes()` (*Java method*), 171

`getAADFCarTrafficCounts()` (*Java method*), 112, 137

`getAADFFreightTrafficCounts()` (*Java method*), 112, 137

`getAADFShapefile()` (*Java method*), 112

`getAbsoluteDifference(FreightMatrix)` (*Java method*), 46

`getAbsoluteDifference(ODMatrixArray)` (*Java method*), 51

`getAbsoluteDifference(ODMatrixArrayTempo)` (*Java method*), 58

`getAbsoluteDifference(ODMatrixMultiKey)` (*Java method*), 64

`getAbsoluteDifference(RealODMatrix)` (*Java method*), 69

`getAbsoluteDifference(RealODMatrixTempo)` (*Java method*), 76

`getAbsoluteDifference(SkimMatrix)` (*Java method*), 85, 88, 92, 102

`getAbsoluteDifference(SkimMatrixFreight)` (*Java method*), 95, 98, 100

`getAccessEgressConsumption(double[], double[], double, Map, Map)` (*Java method*), 177, 183, 188

`getAccessEgressFactor()` (*Java method*), 236

`getArea()` (*Java method*), 204

`getAreaCodeProbabilities()` (*Java method*), 138

`getAreaCodeToNearestNodeID()` (*Java method*), 112

`getAreaCodeToPopulation()` (*Java method*), 112

`getAstarFunctions(Node)` (*Java method*), 113

`getAstarFunctionsTime(Node, double[])` (*Java method*), 113

`getAtGoCode()` (*Java method*), 22

`getAttractions()` (*Java method*), 43

`getAverageAccessEgressDistance(int)` (*Java method*), 113

`getAverageAccessEgressDistanceFreight(int)` (*Java method*), 113

- getAverageCost () (*Java method*), 88, 92, 96, 98, 100, 102
- getAverageCost (FreightMatrix) (*Java method*), 96, 98, 100
- getAverageCost (ODMatrixMultiKey) (*Java method*), 88, 92, 103
- getAverageSpeedFerry () (*Java method*), 113
- getAverageZonalCosts (List) (*Java method*), 103
- getAverageZonalCosts (List, ODMatrixMultiKey) (*Java method*), 103
- getBinIndexMatrix () (*Java method*), 43
- getCAAName () (*Java method*), 15
- getChoiceSet () (*Java method*), 163
- getCO2emission (double, Map, Map, Map) (*Java method*), 183
- getCO2emission (double[], double[], double, Map, Map, Map, boolean) (*Java method*), 177, 183, 188
- getCongestionCharges (int) (*Java method*), 37
- getConsumption (double, Map, Map) (*Java method*), 183
- getConsumption (double[], double[], double, Map, Map, boolean) (*Java method*), 178, 184, 188
- getContinent () (*Java method*), 15
- getCopyOfLinkTravelTimes () (*Java method*), 138
- getCopyOfLinkTravelTimesAsMap () (*Java method*), 138
- getCost () (*Java method*), 158
- getCost (double, Map, Map, Map) (*Java method*), 184
- getCost (double[], double[], double, Map, Map, Map, List, boolean) (*Java method*), 178, 184, 188
- getCost (int, int) (*Java method*), 85, 89, 93, 103
- getCost (int, int, int) (*Java method*), 96, 98, 100
- getCost (String, String) (*Java method*), 85, 89, 92, 103
- getCostSkimMatrix (int) (*Java method*), 37
- getCostSkimMatrixFreight (int) (*Java method*), 37
- getCountry () (*Java method*), 16
- getDayUsage () (*Java method*), 204
- getDemand (String, String) (*Java method*), 26
- getDestination () (*Java method*), 179
- getDestinationLAD () (*Java method*), 184, 188
- getDestinationLAD (Map) (*Java method*), 179, 185, 189
- getDestinationLadID () (*Java method*), 179, 185, 189
- getDestinationNode () (*Java method*), 158, 163, 179
- getDestinationTempoZone () (*Java method*), 185, 189
- getDevelopedEdgeID () (*Java method*), 32
- getDevelopedEdgeID2 () (*Java method*), 32
- getDijkstraTimeWeighter (double[]) (*Java method*), 114
- getDijkstraWeighter () (*Java method*), 114
- getDomesticAirPassengerDemand (int) (*Java method*), 12
- getEasting () (*Java method*), 22, 204
- getEdgeIDtoEdge () (*Java method*), 114
- getEdgeIDtoOtherDirectionEdgeID () (*Java method*), 114
- getEdgeLength (int) (*Java method*), 114
- getEdges () (*Java method*), 158
- getEdgesType () (*Java method*), 115
- getEdgeToZone () (*Java method*), 114
- getEndNodeBlacklist () (*Java method*), 115
- getEndNodeProbabilities () (*Java method*), 138
- getEndYear () (*Java method*), 28, 107
- getEnergyConsumptionParameters () (*Java method*), 138
- getEnergyUnitCosts () (*Java method*), 138
- getEngine () (*Java method*), 179
- getEngineTypeFractions () (*Java method*), 138
- getEngineTypeFractions (int) (*Java method*), 37
- getExpandedEdgeID (RoadNetwork) (*Java method*), 33
- getFastestPath (DirectedNode, DirectedNode, double[]) (*Java method*), 115
- getFastestPathDijkstra (DirectedNode, DirectedNode, double[]) (*Java method*), 115
- getFlagUseRouteChoiceModel () (*Java method*), 139
- getFlow (int, int) (*Java method*), 52
- getFlow (int, int, int) (*Java method*), 46
- getFlow (String, String) (*Java method*), 52, 58, 64, 70, 76
- getFormattedString () (*Java method*), 158
- getFormattedStringEdgeIDsOnly () (*Java method*), 158
- getFreeFlowSpeedARoad () (*Java method*), 115
- getFreeFlowSpeedMRoad () (*Java method*), 115
- getFreeFlowTravelTime () (*Java method*), 116
- getFreightDemand (int) (*Java method*), 37
- getFreightZoneToLAD () (*Java method*), 116
- getFreightZoneToNearestNode () (*Java method*), 116
- getGravitatingPopulation (int) (*Java*

*method*), 116  
 getGravitatingWorkplacePopulation(int) (Java method), 116  
 getHeader() (Java method), 208  
 getIataCode() (Java method), 16  
 getIndexOfRoute(Route) (Java method), 163  
 getInstance() (Java method), 232  
 getInternationalAirPassengerDemand(int) (Java method), 13  
 getIntFlow(int, int) (Java method), 52  
 getIntFlow(String, String) (Java method), 34, 52, 58, 64, 70, 76  
 getIsEdgeUrban() (Java method), 116  
 getKeySet() (Java method), 47, 64, 70, 101, 104  
 getLADCode() (Java method), 22, 204  
 getLadCodeToIDMap() (Java method), 237  
 getLadIDToCodeMap() (Java method), 237  
 getLADName() (Java method), 22, 205  
 getLADToListOfContainedZones() (Java method), 237  
 getLADToName() (Java method), 237  
 getLatitude() (Java method), 16  
 getLength() (Java method), 158, 185, 189  
 getLength(double[]) (Java method), 179  
 getLinkCharges(VehicleType, TimeOfDay) (Java method), 31  
 getLinkFreeFlowTravelTimes() (Java method), 139  
 getLinkTravelTimes() (Java method), 139  
 getLinkVolumeInPCU() (Java method), 139  
 getLinkVolumeInPCUPerTimeOfDay() (Java method), 139  
 getLinkVolumePerVehicleType() (Java method), 139  
 getListOfDisruptedEdgesIDs() (Java method), 108  
 getListOfRemovedRoutes() (Java method), 108  
 getListsOfLADsForNewRouteGeneration() (Java method), 38  
 getLongitude() (Java method), 16  
 getLossFunctionEvaluations() (Java method), 191, 192, 194, 196, 198  
 getMap(URL, URL, URL, URL, String, String) (Java method), 217  
 getMatrixSubset(List, List) (Java method), 64  
 getMaximumEdgeID() (Java method), 117  
 getMaximumNodeID() (Java method), 117  
 getMode() (Java method), 205  
 getMultiplier() (Java method), 180  
 getName() (Java method), 18, 21, 205  
 getNaPTANName() (Java method), 23, 205  
 getNetwork() (Java method), 117  
 getNetworkShapefile() (Java method), 117  
 getNewNetworkShapefile() (Java method), 117  
 getNLC() (Java method), 30, 205  
 getNodeIDtoNode() (Java method), 117  
 getNodesShapefile() (Java method), 118  
 getNodeToAverageAccessEgressDistance() (Java method), 117  
 getNodeToAverageAccessEgressDistanceFreight() (Java method), 117  
 getNodeToGravitatingPopulation() (Java method), 118  
 getNodeToZone() (Java method), 118  
 getNodeToZoneMap() (Java method), 237  
 getNorthing() (Java method), 23, 205  
 getNumberOfIntersections() (Java method), 159  
 getNumberOfLanes() (Java method), 118  
 getNumberOfLanesARoad(String) (Java method), 118  
 getNumberOfLanesMRoad(String) (Java method), 118  
 getNumberOfRoutes() (Java method), 172  
 getNumberOfRouteSets() (Java method), 172  
 getObservedTripLengthDistribution() (Java method), 44  
 getObservedTripLengthDistribution(double[], boolean, boolean) (Java method), 139  
 getObservedTripLengthFrequencies(double[], boolean, boolean) (Java method), 140  
 getOrigin() (Java method), 180  
 getOriginLAD() (Java method), 185, 189  
 getOriginLAD(Map) (Java method), 180, 185, 189  
 getOriginLadID() (Java method), 180, 186, 190  
 getOriginNode() (Java method), 159, 163, 180  
 getOriginTempZone() (Java method), 186, 190  
 getOurAirportsName() (Java method), 16  
 getPassengerDemand(int) (Java method), 38  
 getPathsizes() (Java method), 163  
 getPolicy() (Java method), 31  
 getPolicyEdges() (Java method), 31  
 getPolicyName() (Java method), 31  
 getProbabilities() (Java method), 163  
 getProbabilitiesAsList() (Java method), 164  
 getProductions() (Java method), 44  
 getProperties(String) (Java method), 226, 232  
 getProperty(String) (Java method), 28, 107  
 getRailDemandList() (Java method), 208  
 getRailDemandMap() (Java method), 208  
 getRailStationDemand(int) (Java method), 200  
 getRMSNvalues() (Java method), 81, 82, 84  
 getRoadNetwork() (Java method), 38, 140, 159, 172  
 getRoadNetworkAssignment(int) (Java method), 38  
 getRoute() (Java method), 180  
 getRouteSet(int, int) (Java method), 172



`getRunDays()` (*Java method*), 205  
`getRunwayCapacity()` (*Java method*), 16  
`getScaledMatrix(double)` (*Java method*), 47  
`getScalingFactors()` (*Java method*), 81, 82, 84  
`getSize()` (*Java method*), 164  
`getSortedDestinations()` (*Java method*), 34, 47, 52, 58, 65, 70, 77, 84, 86, 89, 93, 104  
`getSortedOrigins()` (*Java method*), 34, 47, 53, 58, 65, 70, 77, 84, 86, 89, 93, 104  
`getStartNodeBlacklist()` (*Java method*), 118  
`getStartNodeProbabilities()` (*Java method*), 140  
`getStartYear()` (*Java method*), 28, 107  
`getState()` (*Java method*), 28, 107  
`getStatistics()` (*Java method*), 172  
`getSumOfCosts()` (*Java method*), 89, 93, 104  
`getSumOfCosts(ODMatrixMultiKey)` (*Java method*), 90, 93, 104  
`getSumOfFlows()` (*Java method*), 65, 70, 77  
`getTemproCodeToIDMap()` (*Java method*), 237  
`getTemproIDToCodeMap()` (*Java method*), 237  
`getTerminalCapacity()` (*Java method*), 16  
`getThetaEstimate()` (*Java method*), 191, 194, 196, 198  
`getThetaEstimateEnd()` (*Java method*), 194  
`getThetaEstimateStart()` (*Java method*), 194  
`getTime()` (*Java method*), 159  
`getTimeOfDay()` (*Java method*), 180  
`getTimeSkimMatrix(int)` (*Java method*), 38  
`getTimeSkimMatrixFreight(int)` (*Java method*), 39  
`getTotalFlow()` (*Java method*), 53, 65  
`getTotalIntFlow()` (*Java method*), 34, 47, 53, 59, 65, 71, 77  
`getTravelTime(double)` (*Java method*), 186  
`getTravelTime(double[], double, double[], double, boolean)` (*Java method*), 181, 186, 190  
`getTripLengthDistribution()` (*Java method*), 44  
`getTripList()` (*Java method*), 140  
`getUnsortedDestinations()` (*Java method*), 34, 47, 53, 59, 65, 71, 77, 86, 90, 94, 104  
`getUnsortedOrigins()` (*Java method*), 35, 47, 53, 59, 66, 71, 77, 86, 90, 94, 105  
`getUtilities()` (*Java method*), 164  
`getUtility()` (*Java method*), 159  
`getValue()` (*Java method*), 154  
`getVehicle()` (*Java method*), 181  
`getVehicleTypes()` (*Java method*), 48  
`getVolumeToFlowFactor()` (*Java method*), 140  
`getWorkplaceCodeToPopulation()` (*Java method*), 118  
`getWorkplaceZoneProbabilities()` (*Java method*), 141  
`getWorkplaceZoneToNearestNode()` (*Java method*), 119  
`getYearlyUsage()` (*Java method*), 206  
`getZoneIDToLadID()` (*Java method*), 238  
`getZoneIDToNearestNodeDistanceMap()` (*Java method*), 238  
`getZoneIDToNearestNodeIDFromLADTopNodesMap()` (*Java method*), 238  
`getZoneIDToNearestNodeIDMap()` (*Java method*), 238  
`getZonesShapefile()` (*Java method*), 119  
`getZoneToAreaCodes()` (*Java method*), 119  
`getZoneToCentroid()` (*Java method*), 238  
`getZoneToLADMap()` (*Java method*), 238  
`getZoneToListOfContainedNodes()` (*Java method*), 238  
`getZoneToMinMaxDimension()` (*Java method*), 239  
`getZoneToNearestNodeDistanceMap()` (*Java method*), 239  
`getZoneToNearestNodeIDFromLADTopNodesMap()` (*Java method*), 239  
`getZoneToNearestNodeIDMap()` (*Java method*), 239  
`getZoneToNodeDistanceMatrix()` (*Java method*), 239  
`getZoneToNodes()` (*Java method*), 119  
`getZoneToSortedListOfNodeAndDistancePairs()` (*Java method*), 239  
`getZoneToWorkplaceCodes()` (*Java method*), 119  
`getZoneToZoneDistanceMatrix()` (*Java method*), 239  
`getZoning()` (*Java method*), 186, 190  
GVA (*Java field*), 14, 41, 203

## H

`hashCode()` (*Java method*), 159  
HEV\_DIESEL (*Java field*), 149  
HEV\_PETROL (*Java field*), 149  
hour (*Java field*), 175  
HYDROGEN (*Java field*), 149

## I

ICE\_CNG (*Java field*), 149  
ICE\_DIESEL (*Java field*), 149  
ICE\_H2 (*Java field*), 150  
ICE\_LPG (*Java field*), 150  
ICE\_PETROL (*Java field*), 150  
ICON1\_HEIGHT (*Java field*), 215  
ICON1\_WIDTH (*Java field*), 215  
ICON2\_HEIGHT (*Java field*), 215  
ICON2\_WIDTH (*Java field*), 215

- ICON3\_HEIGHT (*Java field*), 215  
 ICON3\_WIDTH (*Java field*), 215  
 INDIAN\_OCEAN\_ISLANDS (*Java field*), 20  
 INDIAN\_SUB\_CONTINENT (*Java field*), 20  
 INITIAL\_ROUTE\_CAPACITY (*Java field*), 165  
 INITIAL\_ROUTE\_SET\_CAPACITY (*Java field*), 165  
 initialise (RoadNetworkAssignment, Properties, ODMatrixMultiKey, HashMap, HashMap, double, double, double, double, double) (*Java method*), 192  
 initialise (RoadNetworkAssignment, Properties, RealODMatrix, double, double, double, double, double) (*Java method*), 191  
 initialise (RoadNetworkAssignment, RouteSetGenerator, Properties, RealODMatrix, HashMap, HashMap, double, double, double, double, double, double, double, double) (*Java method*), 195  
 initialise (RoadNetworkAssignment, Zoning, RouteSetGenerator, RealODMatrixTempo, double, double, double, double, double) (*Java method*), 196, 198  
 initialiseTripList (int) (*Java method*), 141  
 InputFileReader (*Java class*), 226  
 InputFileReader () (*Java constructor*), 226  
 install (Object) (*Java method*), 27, 29, 30, 32, 33, 107, 108  
 installed (*Java field*), 28, 106  
 INT (*Java field*), 17  
 InternationalAirport (*Java class*), 24  
 InternationalAirport (InternationalAirport) (*Java constructor*), 24  
 InternationalAirport (String, String, String, double, double, String, String, long, long) (*Java constructor*), 24  
 internationalAirports (*Java field*), 11  
 InternationalInternodalPassengerDemand (*Java class*), 25  
 InternationalInternodalPassengerDemand () (*Java constructor*), 25  
 InternationalInternodalPassengerDemand (String) (*Java constructor*), 25  
 InternodalPassengerDemand (*Java class*), 25  
 InternodalPassengerDemand () (*Java constructor*), 26  
 INTERSEC (*Java field*), 165  
 Intervention (*Java class*), 27  
 Intervention (Properties) (*Java constructor*), 28  
 Intervention (String) (*Java constructor*), 28  
 InterventionType (*Java enum*), 29  
 interzonalTopNodes (*Java field*), 122  
 isBlacklistedAsEndNode (int) (*Java method*), 119  
 isBlacklistedAsStartNode (int) (*Java method*), 119  
 isEmpty () (*Java method*), 159  
 isTripGoingThroughCongestionChargingZone (String, List) (*Java method*), 181  
 isValid () (*Java method*), 154, 160  
 iterate () (*Java method*), 44  
 iterate (int) (*Java method*), 81, 82, 84
- ## L
- LABEL1\_Y (*Java field*), 215  
 LABEL2\_Y (*Java field*), 215  
 LABEL3\_Y (*Java field*), 215  
 LABEL\_HEIGHT (*Java field*), 215  
 LABEL\_WIDTH (*Java field*), 215  
 LABEL\_X (*Java field*), 216  
 LandingGUI (*Java class*), 214  
 LandingGUI () (*Java constructor*), 217  
 LEFT\_MARGIN (*Java field*), 212, 220, 223  
 LEGEND\_FONT (*Java field*), 211, 225  
 LENGTH (*Java field*), 165  
 LH (*Java field*), 17  
 LIGHT\_GRAY (*Java field*), 216  
 LineVisualiser (*Java class*), 233  
 LineVisualiser (DefaultCategoryDataset, String) (*Java constructor*), 233  
 linkTravelTimeAveragingWeight (*Java field*), 35  
 loadLinkTravelTimes (int, String) (*Java method*), 141  
 lossFunction () (*Java method*), 191, 193, 195, 197, 198  
 LPG (*Java field*), 149  
 LRAIL (*Java field*), 206  
 LT (*Java field*), 202
- ## M
- main (String[]) (*Java method*), 11, 210, 211, 213, 217, 218, 222, 224, 233, 235  
 MAIN\_TITLE\_FONT\_SIZE (*Java field*), 216  
 makeEdgesAdmissible () (*Java method*), 120  
 MAP\_HEIGHT (*Java field*), 212, 220, 223  
 MAP\_WIDTH (*Java field*), 212, 221, 223  
 MATRIX\_SCALING\_FACTOR (*Java field*), 212, 221, 223  
 MAX\_FREIGHT\_ZONE\_ID (*Java field*), 45, 95  
 MAX\_NEAREST\_NODES (*Java field*), 236  
 MAX\_VEHICLE\_ID (*Java field*), 45, 95  
 maximumCapacityARoad (*Java field*), 122

maximumCapacityMRoad (*Java field*), 122  
maximumEdgeID (*Java field*), 109  
maximumNodeID (*Java field*), 109  
MID\_GRAY (*Java field*), 216  
MIDDLE\_EAST (*Java field*), 20  
MIDNIGHT (*Java field*), 151  
MOTORWAY (*Java field*), 121  
multiplier (*Java field*), 175

## N

NA (*Java field*), 18  
NEAR\_EAST (*Java field*), 20  
NetworkVisualiser (*Java class*), 234  
NetworkVisualiser() (*Java constructor*), 234  
NetworkVisualiserDemo (*Java class*), 217  
NetworkVisualiserDemo() (*Java constructor*), 217  
NewRailStation (*Java class*), 29  
NewRailStation (*Java field*), 29  
NewRailStation(Properties) (*Java constructor*), 30  
NewRailStation(String) (*Java constructor*), 30  
nextDouble() (*Java method*), 232  
nextInt(int) (*Java method*), 232  
NINEAM (*Java field*), 151  
NINEPM (*Java field*), 151  
nismod.transport (*package*), 11  
nismod.transport.air (*package*), 11  
nismod.transport.decision (*package*), 27  
nismod.transport.demand (*package*), 34  
nismod.transport.disruption (*package*), 106  
nismod.transport.network.road (*package*), 108  
nismod.transport.optimisation (*package*), 190  
nismod.transport.rail (*package*), 199  
nismod.transport.scripts (*package*), 210  
nismod.transport.showcase (*package*), 211  
nismod.transport.utility (*package*), 225  
nismod.transport.visualisation (*package*), 233  
nismod.transport.zone (*package*), 236  
nodesProbabilityWeighting (*Java field*), 122  
nodesProbabilityWeightingFreight (*Java field*), 122  
NOON (*Java field*), 151  
NORTH\_AFRICA (*Java field*), 20  
NRAIL (*Java field*), 206  
numberOfLanesARoadCollapsedDualCarriageway (*Java field*), 109  
numberOfLanesARoadDualCarriageway (*Java field*), 109  
numberOfLanesARoadRoundabout (*Java field*), 109

numberOfLanesARoadSingleCarriageway (*Java field*), 109  
numberOfLanesARoadSlipRoad (*Java field*), 109  
numberOfLanesMRoadCollapsedDualCarriageway (*Java field*), 110  
numberOfLanesMRoadDualCarriageway (*Java field*), 110  
numberOfLanesMRoadSlipRoad (*Java field*), 110

## O

OC (*Java field*), 18  
ODMatrixArray (*Java class*), 49  
ODMatrixArray(RealODMatrix, Zoning) (*Java constructor*), 49  
ODMatrixArray(String, Zoning) (*Java constructor*), 49  
ODMatrixArray(Zoning) (*Java constructor*), 49  
ODMatrixArrayTempro (*Java class*), 55  
ODMatrixArrayTempro(String, Zoning) (*Java constructor*), 55  
ODMatrixArrayTempro(Zoning) (*Java constructor*), 55  
ODMatrixMultiKey (*Java class*), 61  
ODMatrixMultiKey() (*Java constructor*), 61  
ODMatrixMultiKey(RealODMatrix) (*Java constructor*), 61  
ODMatrixMultiKey(String) (*Java constructor*), 61  
OIL\_RIGS (*Java field*), 20  
ONEAM (*Java field*), 151  
ONEPM (*Java field*), 151  
OPACITY\_FACTOR (*Java field*), 212, 221, 223  
origin (*Java field*), 175  
OTHER (*Java field*), 202  
OTLD (*Java field*), 42

## P

PACIFIC\_OCEAN\_ISLANDS (*Java field*), 20  
Passengers (*Java enum*), 26  
PASTEL\_BLUE (*Java field*), 216  
PASTEL\_GREEN (*Java field*), 216  
PASTEL\_YELLOW (*Java field*), 216  
peakHourPercentage (*Java field*), 122  
PETROL (*Java field*), 149  
PHEV\_DIESEL (*Java field*), 150  
PHEV\_PETROL (*Java field*), 150  
PieChartVisualiser (*Java class*), 235  
PieChartVisualiser(DefaultPieDataset, String, String, boolean) (*Java constructor*), 235  
POPULATION (*Java field*), 14, 42, 203  
predictAndSaveAirDemands(int, int) (*Java method*), 13



[predictAndSaveRailwayDemands\(int, int\)](#) [142](#)  
[\(Java method\)](#), [200](#)  
[predictDomesticAirDemandUsingResultsOfFromYear\(int\)](#) [\(Java method\)](#), [13](#)  
[predictHighwayDemand\(int, int\)](#) [\(Java method\)](#), [39](#)  
[predictHighwayDemands\(int, int\)](#) [\(Java method\)](#), [39](#)  
[predictHighwayDemandUsingResultsOfFromYear\(int, int\)](#) [\(Java method\)](#), [39](#)  
[predictInternationalAirDemandUsingResultsOfFromYear\(int\)](#) [\(Java method\)](#), [13](#)  
[predictionIterations](#) [\(Java field\)](#), [35](#)  
[predictRailwayDemand\(int, int\)](#) [\(Java method\)](#), [200](#)  
[predictRailwayDemands\(int, int\)](#) [\(Java method\)](#), [201](#)  
[predictRailwayDemandUsingResultsOfFromYear\(int, int\)](#) [\(Java method\)](#), [201](#)  
[PricingPolicy](#) [\(Java class\)](#), [30](#)  
[PricingPolicy\(String, String, int, List\)](#) [\(Java constructor\)](#), [30](#)  
[printChoiceSet\(\)](#) [\(Java method\)](#), [164](#)  
[printChoiceSets\(\)](#) [\(Java method\)](#), [172](#)  
[printDemand\(\)](#) [\(Java method\)](#), [26](#)  
[printGEHstatistic\(\)](#) [\(Java method\)](#), [141](#)  
[printGEHstatistic\(double\)](#) [\(Java method\)](#), [141](#)  
[printGEHstatisticFreight\(\)](#) [\(Java method\)](#), [141](#)  
[printGEHstatisticFreight\(double\)](#) [\(Java method\)](#), [141](#)  
[printHourlyGEHstatistic\(\)](#) [\(Java method\)](#), [142](#)  
[printMatrix\(\)](#) [\(Java method\)](#), [48](#), [53](#), [59](#), [66](#), [71](#), [77](#), [86](#), [90](#), [94](#), [96](#), [98](#), [101](#), [105](#)  
[printMatrixFormatted\(\)](#) [\(Java method\)](#), [48](#), [53](#), [59](#), [66](#), [86](#), [90](#), [94](#), [96](#), [98](#), [101](#), [105](#)  
[printMatrixFormatted\(int\)](#) [\(Java method\)](#), [44](#), [71](#), [78](#)  
[printMatrixFormatted\(String\)](#) [\(Java method\)](#), [48](#), [54](#), [59](#), [66](#), [86](#), [90](#), [94](#), [96](#), [99](#), [101](#), [105](#)  
[printMatrixFormatted\(String, int\)](#) [\(Java method\)](#), [44](#), [71](#), [78](#)  
[printNLCsOfNewStations\(\)](#) [\(Java method\)](#), [201](#)  
[printPathsizes\(\)](#) [\(Java method\)](#), [164](#)  
[printProbabilities\(\)](#) [\(Java method\)](#), [164](#)  
[printRailDemand\(String\)](#) [\(Java method\)](#), [208](#)  
[printRailDemandNameSorted\(String\)](#) [\(Java method\)](#), [209](#)  
[printRailDemandNLCSorted\(String\)](#) [\(Java method\)](#), [209](#)  
[printRailDemandUsageSorted\(String\)](#) [\(Java method\)](#), [209](#)  
[printRMSNstatistic\(\)](#) [\(Java method\)](#), [142](#)  
[printRMSNstatisticFreight\(\)](#) [\(Java method\)](#), [142](#)  
[printStatistics\(\)](#) [\(Java method\)](#), [164](#), [172](#)  
[printYearsOfNewStations\(\)](#) [\(Java method\)](#), [201](#)  
[PropertiesReader](#) [\(Java class\)](#), [231](#)  
[PropertiesReader\(\)](#) [\(Java constructor\)](#), [231](#)  
[props](#) [\(Java field\)](#), [28](#), [106](#)  
[PTE](#) [\(Java field\)](#), [202](#)

## R

[RailDemandModel](#) [\(Java class\)](#), [199](#)  
[RailDemandModel\(String, String, String, String, String, String, List, Properties\)](#) [\(Java constructor\)](#), [199](#)  
[RailModeType](#) [\(Java enum\)](#), [206](#)  
[RailStation](#) [\(Java class\)](#), [203](#)  
[RailStation\(int, RailModeType, String, String, int, int, int, double, int, String, String, ElasticityArea\)](#) [\(Java constructor\)](#), [203](#)  
[RailStation\(RailStation\)](#) [\(Java constructor\)](#), [204](#)  
[RailStationDemand](#) [\(Java class\)](#), [207](#)  
[RailStationDemand\(List\)](#) [\(Java constructor\)](#), [207](#)  
[RailStationDemand\(String\)](#) [\(Java constructor\)](#), [207](#)  
[RandomSingleton](#) [\(Java class\)](#), [232](#)  
[readAirElasticitiesFile\(String\)](#) [\(Java method\)](#), [226](#)  
[readAirportFareIndexFile\(String\)](#) [\(Java method\)](#), [227](#)  
[readAVFractionsFile\(String\)](#) [\(Java method\)](#), [226](#)  
[readDomesticAirportsFile\(String\)](#) [\(Java method\)](#), [227](#)  
[readElasticitiesFile\(String\)](#) [\(Java method\)](#), [227](#)  
[readEnergyConsumptionParamsFile\(String\)](#) [\(Java method\)](#), [227](#)  
[readEnergyUnitCostsFile\(String\)](#) [\(Java method\)](#), [227](#)  
[readEngineTypeFractionsFile\(String\)](#) [\(Java method\)](#), [228](#)  
[readFreightTripRatesFile\(String\)](#) [\(Java method\)](#), [228](#)  
[readGVAFile\(String\)](#) [\(Java method\)](#), [228](#)  
[readInternationalAirportsFile\(String\)](#) [\(Java method\)](#), [228](#)  
[readLinkTravelTimeFile\(int, String\)](#) [\(Java method\)](#), [228](#)  
[readPopulationFile\(String\)](#) [\(Java method\)](#), [229](#)

[readRailElasticitiesFile\(String\) \(Java method\), 229](#)  
[readRailStationCostsFile\(String\) \(Java method\), 229](#)  
[readRelativeFuelEfficiencyFile\(String\) \(Java method\), 229](#)  
[readRoutes\(String\) \(Java method\), 173](#)  
[readRoutesBinary\(String\) \(Java method\), 173](#)  
[readRoutesBinaryGZIPpedWithoutValidityCheck\(String\) \(Java method\), 173](#)  
[readRoutesBinaryShortWithoutValidityCheck\(String\) \(Java method\), 173](#)  
[readRoutesBinaryWithoutValidityCheck\(String\) \(Java method\), 173](#)  
[readRoutesWithoutValidityCheck\(String\) \(Java method\), 173](#)  
[readTimeOfDayDistributionFile\(String\) \(Java method\), 230](#)  
[readTimeOfDayDistributionFreightFile\(String\) \(Java method\), 230](#)  
[readTripRatesFile\(String\) \(Java method\), 230](#)  
[readUnitCO2EmissionFile\(String\) \(Java method\), 230](#)  
[readVehicleTypeToPCUFile\(String\) \(Java method\), 230](#)  
[readZonalCarCostsFile\(String\) \(Java method\), 231](#)  
[readZonalCarEnergyConsumptionsFile\(String\) \(Java method\), 231](#)  
[readZonalVehicleCO2EmissionsFile\(String\) \(Java method\), 231](#)  
[RealODMatrix \(Java class\), 67](#)  
[RealODMatrix\(\) \(Java constructor\), 67](#)  
[RealODMatrix\(String\) \(Java constructor\), 67](#)  
[RealODMatrixTempro \(Java class\), 73](#)  
[RealODMatrixTempro\(String, Zoning\) \(Java constructor\), 73](#)  
[RealODMatrixTempro\(Zoning\) \(Java constructor\), 73](#)  
[RebalancedFreightMatrix \(Java class\), 80](#)  
[RebalancedFreightMatrix\(RoadNetworkAssignment, RouteSetGenerator, Properties\) \(Java constructor\), 80](#)  
[RebalancedFreightMatrix\(String, RoadNetworkAssignment, RouteSetGenerator, Properties\) \(Java constructor\), 80](#)  
[RebalancedODMatrix \(Java class\), 81](#)  
[RebalancedODMatrix\(List, List, RoadNetworkAssignment, RouteSetGenerator, Properties\) \(Java constructor\), 82](#)  
[RebalancedTemproODMatrix \(Java class\), 83](#)  
[RebalancedTemproODMatrix\(List, List, RoadNetworkAssignment, RouteSetGenerator, Zoning, Properties\) \(Java constructor\), 83](#)  
[RebalancedTemproODMatrix\(String, RoadNetworkAssignment, RouteSetGenerator, Zoning, Properties\) \(Java constructor\), 83](#)  
[removeCongestionCharges\(int, String\) \(Java method\), 39](#)  
[removeCongestionCharges\(int, String\) \(Java method\), 40](#)  
[removeRoadLink\(Edge\) \(Java method\), 120](#)  
[removeRoutesWithEdge\(int\) \(Java method\), 174](#)  
[removeRoutesWithEdge\(int, List\) \(Java method\), 174](#)  
[removeStation\(int\) \(Java method\), 209](#)  
[replaceNetworkEdgeIDs\(URL\) \(Java method\), 120](#)  
[resetLinkVolumes\(\) \(Java method\), 142](#)  
[resetTripList\(\) \(Java method\), 142](#)  
[RIGID \(Java field\), 153](#)  
[RIGID\\_AV \(Java field\), 153](#)  
[ROAD\\_LINK\\_WIDTH \(Java field\), 217, 234](#)  
[RoadDevelopment \(Java class\), 32](#)  
[RoadDevelopment \(Java field\), 29](#)  
[RoadDevelopment\(Properties\) \(Java constructor\), 32](#)  
[RoadDevelopment\(String\) \(Java constructor\), 32](#)  
[RoadDevelopmentDashboard \(Java class\), 220](#)  
[RoadDevelopmentDashboard\(\) \(Java constructor\), 222](#)  
[RoadDisruption \(Java class\), 107](#)  
[RoadDisruption\(Properties\) \(Java constructor\), 108](#)  
[RoadDisruption\(String\) \(Java constructor\), 108](#)  
[RoadExpansion \(Java class\), 33](#)  
[RoadExpansion \(Java field\), 29](#)  
[RoadExpansion\(Properties\) \(Java constructor\), 33](#)  
[RoadExpansion\(String\) \(Java constructor\), 33](#)  
[RoadExpansionDashboard \(Java class\), 222](#)  
[RoadExpansionDashboard\(\) \(Java constructor\), 224](#)  
[RoadNetwork \(Java class\), 108](#)  
[RoadNetwork\(URL, URL, URL, URL, String, String, String, String, String, String, Properties\) \(Java constructor\), 110](#)  
[RoadNetworkAssignment \(Java class\), 121](#)  
[RoadNetworkAssignment\(RoadNetwork, Zoning, Map, Map, Map, Map, Map, Map, Map, Map, Map, HashMap, HashMap, List, Properties\) \(Java constructor\), 123](#)

RoadPath (*Java class*), 154  
 RoadPath() (*Java constructor*), 154  
 RoadPath(Collection) (*Java constructor*), 154  
 roundMatrixValues() (*Java method*), 71, 78  
 Route (*Java class*), 154  
 route (*Java field*), 175  
 Route(RoadNetwork) (*Java constructor*), 155  
 Route(RoadPath, RoadNetwork) (*Java constructor*), 155  
 RouteChoiceParams (*Java enum*), 164  
 RouteSet (*Java class*), 161  
 RouteSet(RoadNetwork) (*Java constructor*), 161  
 RouteSetGenerator (*Java class*), 165  
 RouteSetGenerator(RoadNetwork, Properties) (*Java constructor*), 165  
 RUN\_BUTTON\_BORDER (*Java field*), 213, 221, 223  
 RunArcRail (*Java class*), 210  
 runSPSA(int) (*Java method*), 192, 193, 195, 197, 199

## S

SA (*Java field*), 18  
 saveAirPassengerDemand(int, String) (*Java method*), 23, 25, 26  
 saveAllResults(int) (*Java method*), 13, 40, 201  
 saveAllResults(int, int) (*Java method*), 40  
 saveAssignmentResults(int, String) (*Java method*), 40, 142  
 saveDomesticAirDemand(int, String) (*Java method*), 14  
 saveEnergyConsumptions(int, String) (*Java method*), 40  
 saveEnergyConsumptionsPerVehicleType(int, String) (*Java method*), 142  
 saveHourlyCarVolumes(int, String) (*Java method*), 143  
 saveInternationalAirDemand(int, String) (*Java method*), 14  
 saveLinkTravelTimes(int, String) (*Java method*), 143  
 saveMatrixFormatted(String) (*Java method*), 48, 54, 59, 66, 72, 78, 87, 90, 94, 97, 99, 101, 105  
 saveMatrixFormatted2(String) (*Java method*), 60, 66, 72, 78  
 saveMatrixFormatted3(String) (*Java method*), 60, 78  
 saveMatrixFormattedList(String) (*Java method*), 54, 87, 91, 94, 105  
 saveNodeProbabilities(String) (*Java method*), 195  
 saveOriginDestinationCarElectricityConsumption(String) (*Java method*), 143  
 savePeakLinkPointCapacities(int, String) (*Java method*), 143  
 saveRailStationDemand(int, String) (*Java method*), 201, 209  
 saveRoutes(String, boolean) (*Java method*), 174  
 saveRoutesBinary(String, boolean) (*Java method*), 174  
 saveRoutesBinaryGZIPped(String, boolean) (*Java method*), 174  
 saveRoutesBinaryShort(String, boolean) (*Java method*), 175  
 saveToPNG(String) (*Java method*), 233–235  
 saveTotalCO2Emissions(int, String) (*Java method*), 143  
 saveTotalEnergyConsumptions(int, String) (*Java method*), 144  
 saveZonalCarEnergyConsumptions(int, double, String) (*Java method*), 144  
 saveZonalRailStationDemand(int, String) (*Java method*), 202, 209  
 saveZonalTemporalTripStartsForEVs(int, VehicleType, String) (*Java method*), 144  
 saveZonalTemporalTripStartsForH2(int, VehicleType, String) (*Java method*), 144  
 saveZonalTemporalVehicleElectricity(int, VehicleType, double, String) (*Java method*), 144  
 saveZonalTemporalVehicleHydrogen(int, VehicleType, double, String) (*Java method*), 145  
 saveZonalVehicleCO2Emissions(int, double, String) (*Java method*), 145  
 saveZonalVehicleKilometres(int, String) (*Java method*), 145  
 saveZonalVehicleKilometresWithAccessEgress(int, String) (*Java method*), 145  
 scaleMatrix(SkimMatrixFreight) (*Java method*), 48  
 scaleMatrixValue(double) (*Java method*), 54, 60, 66, 72, 79  
 scaleMatrixValue(ODMatrixArrayTempo) (*Java method*), 60  
 scaleMatrixValue(RealODMatrix) (*Java method*), 72  
 scaleMatrixValue(RealODMatrixTempo) (*Java method*), 79  
 scaleToAttractions() (*Java method*), 44  
 scaleToObservedTripLengthDistribution() (*Java method*), 44  
 scaleToProductions() (*Java method*), 45  
 scaleToTrafficCounts() (*Java method*), 81, 83, 85  
 SCHEDULED (*Java field*), 26

SCREEN\_HEIGHT (*Java field*), 216  
 SCREEN\_WIDTH (*Java field*), 216  
 SE (*Java field*), 202  
 SECOND\_MARGIN (*Java field*), 213, 221, 223  
 setCongestionCharges(int, List) (*Java method*), 41  
 setCost(int, int, double) (*Java method*), 87, 91, 95, 106  
 setCost(int, int, int, double) (*Java method*), 97, 99, 101  
 setCost(String, String, double) (*Java method*), 87, 91, 95, 105  
 setDailyUsage(double) (*Java method*), 206  
 setDemand(String, String, long, long, long) (*Java method*), 26  
 setElectricityUnitCost(double) (*Java method*), 146  
 setEndNodeProbabilities(HashMap) (*Java method*), 146  
 setEnergyConsumptionParameters(VehicleType, EngineType, Map) (*Java method*), 146  
 setEnergyUnitCost(EnergyType, double) (*Java method*), 146  
 setEngineTypeFractions(int, Map) (*Java method*), 41  
 setEngineTypeFractions(int, VehicleType, Map) (*Java method*), 41  
 setEngineTypeFractions(VehicleType, Map) (*Java method*), 146  
 setFlow(int, int, double) (*Java method*), 79  
 setFlow(int, int, int) (*Java method*), 54, 60  
 setFlow(int, int, int, int) (*Java method*), 48  
 setFlow(String, String, double) (*Java method*), 72, 79  
 setFlow(String, String, int) (*Java method*), 54, 60, 67  
 setRailStationDemand(int, RailStationDemand) (*Java method*), 202  
 setSeed(long) (*Java method*), 232  
 setStartNodeProbabilities(HashMap) (*Java method*), 147  
 setUtility(double) (*Java method*), 160  
 setYearlyUsage(int) (*Java method*), 206  
 SEVENAM (*Java field*), 151  
 SEVENPM (*Java field*), 152  
 SH (*Java field*), 17  
 SIXAM (*Java field*), 152  
 SIXPM (*Java field*), 152  
 SkimMatrix (*Java interface*), 85  
 SkimMatrixArray (*Java class*), 87  
 SkimMatrixArray(String, Zoning) (*Java constructor*), 88  
 SkimMatrixArray(Zoning) (*Java constructor*), 88  
 SkimMatrixArrayTempo (*Java class*), 91  
 SkimMatrixArrayTempo(String, Zoning) (*Java constructor*), 92  
 SkimMatrixArrayTempo(Zoning) (*Java constructor*), 91  
 SkimMatrixFreight (*Java interface*), 95  
 SkimMatrixFreightArray (*Java class*), 97  
 SkimMatrixFreightArray() (*Java constructor*), 97  
 SkimMatrixFreightArray(String) (*Java constructor*), 97  
 SkimMatrixFreightMultiKey (*Java class*), 99  
 SkimMatrixFreightMultiKey() (*Java constructor*), 99  
 SkimMatrixFreightMultiKey(String) (*Java constructor*), 99  
 SkimMatrixMultiKey (*Java class*), 102  
 SkimMatrixMultiKey(String, Zoning) (*Java constructor*), 102  
 SkimMatrixMultiKey(Zoning) (*Java constructor*), 102  
 sortGravityNodes() (*Java method*), 120  
 sortGravityNodesFreight() (*Java method*), 120  
 sortStationsOnName() (*Java method*), 210  
 sortStationsOnNLC() (*Java method*), 210  
 sortStationsOnUsage() (*Java method*), 210  
 SOUTH\_AMERICA (*Java field*), 20  
 SOUTHERN\_AFRICA (*Java field*), 20  
 SPSA (*Java class*), 190  
 SPSA() (*Java constructor*), 191  
 SPSA2 (*Java class*), 192  
 SPSA2() (*Java constructor*), 192  
 SPSA3 (*Java class*), 193  
 SPSA3() (*Java constructor*), 194  
 SPSA4 (*Java class*), 196  
 SPSA4(Properties) (*Java constructor*), 196  
 SPSA5 (*Java class*), 197  
 SPSA5(Properties) (*Java constructor*), 198  
 SUBTITLE\_FONT\_SIZE (*Java field*), 216  
 sumMatrixSubset(List, List) (*Java method*), 61, 67, 72, 79

## T

TABLE\_BORDER (*Java field*), 213, 221, 224  
 TABLE\_FONT (*Java field*), 210, 213, 221, 224  
 TABLE\_LABEL\_WIDTH (*Java field*), 213, 221, 224  
 TABLE\_ROW\_HEIGHT (*Java field*), 213, 221, 224  
 TableChangeLegend (*Java class*), 224  
 TableChangeLegend() (*Java constructor*), 225  
 TableChangeLegendHorizontal (*Java class*), 225

TableChangeLegendHorizontal() (*Java constructor*), 225  
 TENAM (*Java field*), 152  
 TENPM (*Java field*), 152  
 THETA\_MAX (*Java field*), 190, 192, 196, 197  
 THETA\_MAX\_FLOW (*Java field*), 193  
 THETA\_MAX\_PROBABILITY (*Java field*), 193  
 THETA\_MIN (*Java field*), 190, 192, 196, 197  
 THETA\_MIN\_FLOW (*Java field*), 194  
 THETA\_MIN\_PROBABILITY (*Java field*), 194  
 THREEAM (*Java field*), 152  
 THREEPM (*Java field*), 152  
 TIME (*Java field*), 42, 165, 203  
 TimeOfDay (*Java enum*), 150  
 TOOLBAR (*Java field*), 216  
 TOP\_LAD\_NODES (*Java field*), 236  
 topTempProNodes (*Java field*), 123  
 toString() (*Java method*), 17, 23, 25, 29, 107, 120, 160, 181, 186, 206  
 TOTAL (*Java field*), 27  
 TOTAL\_DEMAND\_BORDER (*Java field*), 213, 221, 224  
 trimToSize() (*Java method*), 160  
 Trip (*Java class*), 175  
 Trip(VehicleType, EngineType, Route, TimeOfDay, Integer, Integer) (*Java constructor*), 176  
 Trip(VehicleType, EngineType, Route, TimeOfDay, Integer, Integer, int) (*Java constructor*), 176  
 TripMinor (*Java class*), 181  
 TripMinor(VehicleType, EngineType, TimeOfDay, Integer, Integer, double, Zoning) (*Java constructor*), 182  
 TripMinor(VehicleType, EngineType, TimeOfDay, Integer, Integer, double, Zoning, int) (*Java constructor*), 182  
 TripTempPro (*Java class*), 186  
 TripTempPro(VehicleType, EngineType, Route, TimeOfDay, Integer, Integer, Zoning) (*Java constructor*), 187  
 TripTempPro(VehicleType, EngineType, Route, TimeOfDay, Integer, Integer, Zoning, int) (*Java constructor*), 187  
 TUBE (*Java field*), 207  
 TWOAM (*Java field*), 152  
 TWOPM (*Java field*), 152  
 UNITED\_STATES\_OF\_AMERICA (*Java field*), 20  
 updateCostSkimMatrix(SkimMatrix) (*Java method*), 147  
 updateCostSkimMatrixFreight(SkimMatrixFreight) (*Java method*), 147  
 updateLinkTravelTimes() (*Java method*), 147  
 updateLinkTravelTimes(double) (*Java method*), 147  
 updateLinkVolumeInPCU() (*Java method*), 147  
 updateLinkVolumeInPCUPerTimeOfDay() (*Java method*), 148  
 updateLinkVolumePerVehicleType() (*Java method*), 148  
 updateTimeSkimMatrix(SkimMatrix) (*Java method*), 148  
 updateTimeSkimMatrixFreight(SkimMatrixFreight) (*Java method*), 148  
 updateTripLengthDistribution() (*Java method*), 45  
**V**  
 value (*Java field*), 153  
 VAN (*Java field*), 153  
 VAN\_AV (*Java field*), 153  
 vehicle (*Java field*), 175  
 VehicleType (*Java enum*), 152  
 visualise(RoadNetwork, String) (*Java method*), 218, 234  
 visualise(RoadNetwork, String, Map, String, String) (*Java method*), 219, 234  
 visualise(RoadNetwork, String, Map, String, String, URL) (*Java method*), 219, 235  
 visualise(URL, URL, URL, URL, String, String) (*Java method*), 218  
 volumeToFlowFactor (*Java field*), 123  
**W**  
 WebTAG (*Java enum*), 160  
 WEST\_AFRICA (*Java field*), 21  
 WESTERN\_EUROPE\_EU (*Java field*), 21  
 WESTERN\_EUROPE\_OTHER (*Java field*), 21  
**Z**  
 Zoning (*Java class*), 236  
 zoning (*Java field*), 182, 187  
 Zoning(URL, URL, RoadNetwork, Properties) (*Java constructor*), 236

## U

UK (*Java field*), 17  
 uninstall(Object) (*Java method*), 27, 29, 30, 32, 33, 107, 108