

Mezzo file formats

Version 0.0.9

Wilco Burghout, 2009-06-25

Contents

1. Introduction	3
2. Master file (*.mezzo, *.mime)	3
3. Input files	7
3.1 Network file	7
3.2 Turnings file	11
3.3 Signal Control	12
3.3 Histtimes	13
3.4 Routes	14
3.5 Demand	14
3.6 Incident File	15
3.7 Vehicle types	16
3.8 Virtual Links	16
3.9 Server Rates	17
3.10 Assignment links	17
3.11 Parameters file	18
4. Output files	22
4.1 Linktimes	22
4.2 Output	22
4.3 Summary	23
4.4 Speeds	23
4.5 Inflows/Outflows	24
4.6 Queue lengths	24
4.7 Densities	24
4.8 Assignment matrix	25
4.9 Virtual queues	26
References	26

1. Introduction

In this document the (current) structure of the Mezzo input and output files is described. This structure is likely to change over time as new needs come up, and possibly due to structural changes in Mezzo. The explanations are illustrated by (excerpts from) example files, which will be printed in `courier`.

2. Master file (*.mezzo, *.mime)

The master file contains all relevant scenario information, including all the names and paths of input and output files, as well as simulation parameters such as simulation duration etc. There is no difference (yet) between *.mezzo and *.mime files, but in the future, those elements that are specific to MiMe (hybrid micro-meso) applications will be removed from the *.mezzo file format.

```
#input_files
network= net5.dat
turnings= turnings.dat
signals= signal.dat
histtimes= histtimes.dat
routes= routes.dat
demand= input.dat
incident= noincident.dat
vehicletypes= vehiclemix.dat
virtuallinks= virtuallinks.dat
serverrates= serverrates.dat
#output_files
linktimes= output/linktimes.dat
output= output/output.dat
summary= output/summary.dat
speeds= output/speeds.dat
inflows= output/inflows.dat
```

```
outflows= output/outflows.dat
queuelengths= output/queuelengths.dat
densities= output/densities.dat
#scenario
runtime= 8100
calc_paths= 0
traveltime_alpha= 0.5
parameters= parameters.dat
no_background
```

Figure 1. Master file (*.mezzo, *.mime)

The first part of the master file defines the locations of the input files. Note that the keywords and spacing need to be exactly as depicted in figure 1. I.e. the first line of the file should always be:

```
#input_files
```

And the keyword for the network file location is:

```
network=
```

Due to the way the input files are parsed, a small change (such as `network =` (note the space)) would cause problems.

The meaning of the files indicated in the `#input_files` section is:

- Network: network definition, i.e. nodes, links, node servers, speed-density functions
- Signals: definition of the signal control plans, stages, timings etc.
- Turnings: definition of the turning movements. Can be generated automatically if missing or incomplete.
- Histtimes: Historical link travel times. These times are used for the pre-trip route choice.
- Routes: Description of the set of known routes for all OD pairs (chains of links).
- Demand: OD demand. Starts with a base matrix in which all active OD pairs need to be listed (with non-zero demand), followed by slices which become

active at a certain time period. These slices *update* the demand for the OD pairs listed in the slice. That is, not all OD pairs need to be repeated in the slice, only those that get an updated rate.

- Incident: file that describes all incidents, when they occur, on what link, what the penalty will be, what the broadcasted delay will be, new s/d functions that are used to describe the speed/density on the affected link(s) during the incident. In case of no incidents, use dummy file “noincident.dat”.
- Vehicletypes: In this file the vehicle mix is described. Per vehicletype the type name, percentage and length in meters.
- Virtuallinks: This file is MiMe specific. If Mezzo is used in combination with a microscopic model, this file indicates the virtual links that indicate the paths through the microscopic model. In other cases an empty file is used (with `virtuallinks: 0` on the first line).
- Serrates: In this file changes to server rates are modelled, to model variation over time in node capacities. Typical examples are incidents, modelling time-varying capacity at exit links (e.g. when a congestion outside the network boundary is known to propagate into the network).

#output_files

- Linktimes: This file contains the output average link travel times. The time periods are the same as the input file (histtimes)
- Output: This file contains detailed Origin Destination output. Per OD pair it outputs for each arrived vehicle: `Origin_id`, `Destination_id`, `Vehicle_id`, `Start_time`(in seconds), `Arrival_time` (sec), `Travel_time` (sec), `Travelled distance` (in meters) and whether the vehicle switched from its original route (0=no, 1=yes)
- Summary: This file contains a summary of the OD output. Per OD pair it outputs: `Origin_id`, `Destination_id`, `Total_vehicles_arrived`, `Total_Travel_Time` (sec), `Total_Distance_Travelled` (m)
- Speeds: Contains average link (traversal) speeds for each time period defined in the MOE (Measures Of Effectiveness) collection parameter (currently defined in the Parameters.h file, will be moved to this file, or a separate parameter file later on)
- Inflows: Contains average link inflows for each MOE time period.
- Outflows: Contains average link outflows for each MOE period.

- Queuelengths: Contains the average queue length on each link for each MOE period.
- Densities: Contains average densities on each link for each MOE period.
- In addition to these output files, two more outputfiles are generated in the directory from which Mezzo is started:
 - Allmoes.dat: contains per link (links in columns) the speed, inflow, outflow, density and queue length for each time period (periods in rows). The name and location of this file will be defined in the Master file in the future.
 - Timestamps.dat: specific for MiMe applications. Contains the time headways with which vehicles are sent into the microscopic area. This is used to check time headway distributions generated by the Mezzo boundaries with real data and those distributions generated by the microscopic model. In case of standalone Mezzo runs this files is empty.

#scenario

- Runtime: the total runtime (duration) of the scenario in seconds.
- Calc_paths: Sets whether Mezzo should look for new shortest paths given the input time-dependent travel times (histtimes). 0=no and 1=yes. If yes (1), any newly found paths will be added to the Routes input file. If no (0), the Routes input file will be used as is.
- Traveltime_alpha: is the parameter used for smoothing the output travel times generated. If the output travel times are used for analysis, set to 1. The output travel times are calculated as follows:

$$tt_i^{n+1}(t) = \alpha TT_i^n(t) + (1 - \alpha) tt_i^{n+1}(t) \quad (1)$$

Where,

- $tt_i^{n+1}(t)$ = historical travel time on link i for iteration $n+1$, when entering at time t
- $tt_i^n(t)$ = historical travel time on link i for iteration n , when entering at time t
- $TT_i^n(t)$ = Simulated (output) travel time on link i for iteration n , when entering at time t

α = moving average parameter, $\in [0,1]$

- Parameters: in this file the set of parameters for Mezzo are stored.
- (Only in a MiMe application with VISSIM) Vissim file (E.g. “vissimfile=e:\mezzo\vissimtest\test2.inp”) indicating the location of the VISSIM input file that contains the part of the network in VISSIM.
- (Optional) Background: if a background image (*.png) is desired, indicate here with:

background= image.png

Otherwise indicate that there is no background:

no_background

3. Input files

3.1 Network file

The network file contains definitions of four key components: Node-Servers, Nodes, Speed-density functions and Links. The first part of the file contains the node-servers (or better: turning-servers, as they can be turning movement specific).

```
servers: 44
{ 0 1 1.44 0.02 0 }
{ 1 1 2.00 0.02 0 }
{ 2 1 2.25 0.02 3 }
...
```

Figure 2. Servers part of Network file

The grammar of the server part is as follows:

S \leftarrow "servers:" NUMBER SERVER*

NUMBER \leftarrow integer

SERVER \leftarrow "{" SID STYPE MEAN SD DELAY "}"

SID \leftarrow integer

STYPE \leftarrow integer

MEAN ←double
 SD ←double
 DELAY ←double

The meaning in words of the above structure is:

- The first line contains the `servers:` keyword, followed by the number of server definitions that will follow.
- Each server definition starts with `{` and closes with `}`:
`{ ServerID ServerType Mean StdDev Delay }`
- Server ID is the ID with which the Server will be identified in the Node definitions and Turning definitions (in the Turnings file).
- ServerType indicates the type of server. Valid types are (for now):
 0. Dummy server. Transfers vehicles directly, without delay
 1. Normal (Mean, StdDev) server
- New server types can be added in the future.
- Mean and StdDev are the mean and the standard deviation of the server (in seconds).
- Delay is an optional extra delay to add to the vehicles passing this server. This is used (for instance) to model turning movements that have a non-negligible length (such as on/off ramps).

```
nodes: 128
{ 0 1 0 32 }
{ 1 2 25 9 0 }
{ 2 3 236 344 }
...
```

Figure 3. Node definitions in Network file

The Node definitions are similar to the Server definitions. The `nodes:` keyword is followed by the number of Node definitions. Each node definition starts with `{` and closes with `}`:

`{ NodeID NodeType Xcoordinate Ycoordinate }`
 Or
`{ NodeID NodeType Xcoordinate Ycoordinate ServerID }`

The NodeType can be one of:

1. Origin
2. Destination
3. Junction
4. BoundaryIn (MiMe only)
5. BoundaryOut (MiMe only)

In the case of a Destination or BoundaryOut, the ServerID is included, in all other cases it is left out. The serverID refers to a server listed in the Servers section. This server determines the rate at which vehicles arriving at Destinations or BoundaryOut nodes are processed.

The Xcoordinate and Ycoordinate are used for plotting in the GUI only, they do not affect link lengths between nodes, as these are coded explicitly.

NodeID, NodeType, Xcoordinate, Ycoordinate, ServerID are integers.

sdfuncs:	4								
{	0	2	28	7	130	6	1.7	15	}
{	1	2	23	7	130	10	1.5	6	}
{	2	2	19	7	130	8	1.5	10	}
{	3	2	16	7	130	8	1.5	10	}

Figure 4. Speed-Density function section of Network file.

The speed-density functions are defined in a similar fashion as the servers and nodes:

{ SDID SDType VMax VMin KMax } in case SDType = 0
 { SDID SDType VMax VMin KMax Alpha Beta } in case
 SDType = 1
 { SDID SDType VMax VMin KMax KMin Alpha Beta } in case
 SDType = 2

Where:

- SDType indicates the type of SD function:
 0. Greenshields
 1. Generalised according to (May 1990)
 2. Generalised according to (Burghout 2004) (recommended)

VMax and Vmin are the maximum and minimum speeds (m/s), Kmax and Kmin the minimum and maximum densities and Alpha and Beta exponents of the generalised SD functions. For more details on the SD functions see Burghout 2004. SDID, SDType, VMax, VMin, Kmax, Kmin are integers. Alpha and Beta are doubles.

```

links:      155
{ 0 0 2 400 3 0 Hornsgatan }
{ 1 3 1 400 3 0 Highstreet }
{ 2 2 4 200 4 0 Slussen }
...

```

Figure 5. Link section in Network file.

The link definitions are as follows:

```

{ LinkID      StartNodeID EndNodeID Length      Nr_Lanes      SDID
Linkname      }

```

Where StartNodeID and EndNodeID refer to the IDs of start (upstream) and end (downstream) nodes defined in the nodes section. Length is the link length in meters, Nr_Lanes is the number of lanes and SDID is the ID of the Speed-Density function associated with the link. Linkname is the street name and has to consist of one word. So streetnames consisting of more than one word should be joined by dashes or underscores (i.e. Olof-Palmes-gata).

LinkID, StartNodeID, EndNodeID, Length, Nr_Lanes and SDID are integers.

```

linkpoints:      155
{ 0 3 { 9897 1343 10017 1157 10080 1066 } }
{ 1 3 { 10050 1039 9858 1340 9801 1415 } }
{ 2 3 { 10220 1758 10448 1651 10495 1621 } }
...

```

Figure 6. Linkpoints in Network file.

The linkpoint definitions are as follows:

```

{ LinkID      NrPoints { X1      Y1      X2      Y2 ... Xn      Yn } }

```

Link points describe the shape of the links in the network, and are optional.

3.2 Turnings file

Update 2008-11-14: Give-way definitions after the turning definitions

This file contains all the specific turning movement definitions. Per node (junction) there is a turning movement defined for each pair of incoming and outgoing links. These turning movements can be generated automatically if Mezzo is supplied with an empty Turnings file. In this case all turning movements will have the first Server specified in the Servers section in the Network file.

```
turnings: 232
{ 231 2 0 0 2 80 }
{ 230 3 0 3 1 80 }
{ 229 4 3 2 4 80 }
...
giveways: 26
{ 20 1 2 }
{ 21 3 2 }
...
```

Figure 7. Turnings File.

The turnings are specified as follows:

```
{ TurningID NodeID ServerID IncomingLinkID OutgoingLinkID LookBack }
```

Where NodeID is the ID of the node (junction), the ServerID refers to the server used to transfer vehicles from the Incoming link to the Outgoing link. The LookBack value determines how far back in the incoming link queue the turning process may look to find vehicles that are taking this turning. This approximates the effect of a turning becoming inaccessible if vehicles heading for other turnings are queuing beyond a certain point.

Update 2008-11-14: Give way definitions

The giveways are specified as follows:

```
{ NodeID MinorTurningID MajorTurningID }
```

Defines give-way relations where minor turning gives way to major turning. The saturation flows are already in the turnings server specification. In the future more parameters may be added, for instance a maximum wait, or guaranteed minimum rate.

3.3 Signal Control

Signal control can be defined in a signal.dat file which should be present in the same directory as the master (*.mezzo) file. Mezzo has the ability to simulate fixed signal plans, with the ability to have any number of signal plans (in sequence) per controller. Each plan steps through a number of stages (“phases” in American English), and each stage contains a number (1 or more) turning movements that are controlled by it. For each stage the only pieces of information are the start of green in the cycle and its duration, in addition to the ids of the turning movements it controls. For signal plans the start and stop time define when it is active, the offset defines the offset the first stage has with respect to the activation of the signalplan. The structure is as follows:

```
controls: Nr_Controls
{ ControlID      Nr_Plans
  { PlanID_1 Start Stop  Offset      Cycletime  Nr_Stages
    { StageID_1      Start Duration  Nr_Turnings
      {Turning_1 Turning_2 ...      Turning_n }
    }
    ...
    {StageID_n ...
      { ...
    }
  }
  ...
  { PlanID_n ...
    { ...
      { ...
    }
  }
}
```

```

    }
  }
}

```

So the top level definitions are the traffic controls, with the signal plans being sub-defined per traffic control, and the stages sub-defined per signal plan. In figure 13 an example is shown of a signal.dat file with only one control defined, which has only one signal plan. The signal plan has two stages, that control two turnings each. Note that the turnings are defined in a separate file.

```

controls: 1
{
  0 1
  {
    0 0 36000 0 600 2
    {
      0 0 300 2 { 0 1 } }
      {
        1 300 300 2 { 2 3 } }
    }
  }
}

```

Figure 8. Example of signal.dat signal control file.

3.3 Histtimes

This file contains the time-dependent historical (or ‘experienced’) link travel times based on which the drivers make their route choices.

```

links:      155
periods:    4
periodlength: 600
{
  0 25.2949 23.079 29.408 43.4632 }
{
  1 15.953 15.733 16.7801 17.2534 }
{
  2 7.67487 7.59963 7.63349 8.0142 }
...

```

Figure 9. Historical travel times.

links: indicates the number of links for which the link travel times are defined.

periods: indicates the number of time periods for the link travel times.
 periodlength: indicates the duration of each period in seconds.

Per link the travel times are defined as follows:

```
{ LinkID    Traveltime1  Traveltime2  ...    TraveltimeN }
```

3.4 Routes

In the routes file the known routes from each origin to destination are listed. Multiple paths per OD pair are possible, and the file is augmented by Mezzo is the calc_routes is enabled and new shortest paths are found.

```
routes: 222
{ 1 124 123 22 { 150 151 153 8 15 21 22 27 31 33 36 40 37 34 32 30
26 25 11 154 152 149 } }
{ 2 124 115 6 { 150 148 146 143 140 137 } }
...
```

Figure 10. Route file.

Each route definition is constructed as follows:

```
{ RouteID OriginID DestinationID NumberOfLinks { LinkID1 LinkID2 ...
LinkIDn } }
```

3.5 Demand

The demand file contains the time-dependent Origin Destination demand matrices. The first section contains the *base matrix* which should contain *all active OD pairs*. Moreover *each OD pair should have a positive demand (>0)*. **UPDATE: Rates can now be real numbers ('double').**

```
od_pairs: 122
scale: 1.0
{ 0 9 119.0 }
{ 0 24 42.0 }
```

```
...
```

Figure 11. Demand file

`scale`: indicates the factor by which all odpairs in the current time-slice should be scaled.

The OD element definitions are as follows:

```
{ OriginID  DestinationID  Rate }
```

The Rate is expressed in hourly flow rates.

The second section of the demand file defines the subsequent time slices.

```
slices:      8
od_pairs:   111
scale:      1.0
loadtime:   900
{    0    9    181.0 }
{    0   24    42.0 }
{    0   34   489.0 }
...
```

Figure 12. Time slice definitions in the Demand file.

`slices`: indicates the number of subsequent matrix time slices.

For each matrix time slice the number of `od_pairs` is defined, the `scale` as well as the `loadtime`. The `loadtime` indicates the time (in seconds, from the start of the simulation) when the matrix time slice should be loaded.

The OD elements defined in the matrix time slice are defined as in the base matrix. However, only those pairs need to be listed, for which the demand is different from the previously defined demand. Of course one can simply repeat those elements with unchanged rates, but this is not necessary.

3.6 Incident File

Will be explained later, and is probably going to change.

3.7 Vehicle types

This file contains the vehicle mix.

```
vtypes:      5
{      1      NewCars      0.410 6.0  }
{      2      OldCars      0.400 6.0  }
...
```

Figure 13. Vehicle types file

The each vehicle type definition is of the following form:

```
{      VehTypeID  VehTypeName      Percentage  Length      }
```

The VehTypeID is an integer, VehTypeName is a string and the Percentage and Length are doubles. The Percentage is the proportion of this vehicle type in the total vehicle mix. The Length is the space occupied by the vehicle in a queue. In other words, it's the vehicle's length in meters + the headway to the vehicle in front in case of queuing. This parameter is used to know the amount of space occupied by the vehicles in a queue.

3.8 Virtual Links

This file is used only in case of a MiMe application, to define the virtual links that represent the paths inside the microscopic area. The definition of a virtual link is identical to that of a normal link:

```
{ LinkID      StartNodeID  EndNodeID  Length      Nr_Lanes      SDID }
```

The SDID parameter is ignored, the Nr_Lanes should be set to the nr of lanes in the first link of the corresponding microscopic path. For more information, see Burghout 2004.

!!!!!!!!!!!! UPDATE !!!!!!!!!!!!!!!!!!!!!!!!!!!!!

In INTERMEZZO (VISSIM+MEZZO) the virtual links have the following form

```
{ LinkID      StartNodeID  EndNodeID  Length      Nr_Lanes      SDID }
```

```
V_EnterParkingLot V_Last_link V_Nr_Nodes {V_Node_1 ... V_Node_n }
}
```

3.9 Server Rates

The Server rates file is used to vary the throughput rates of servers over time during the simulation run. This feature can be used to temporarily limit exit capacities of certain destinations for certain time periods, to mimic observed capacity reductions (for instance when congestion is known to occur downstream of such destinations, i.e. *outside* the boundaries of the network). Other applications include capacity reduction due to incidents or roadworks, temporary buslanes, etc.

The format of the server rates is as follows:

Rates: followed by the number of server_rates

Each server rate is in fact an event, with an associated time of execution. The format of server rates is:

```
{ ServerID Time Mean StdDev }
```

At time Time the server with ServerID will have a changed capacity with mean Mean and standarddeviation StdDev. The new capacity will persist, until a new change is executed. ServerID is and integer, Time, Mean, StdDev are doubles.

3.10 Assignment links

If Mezzo is compiled with the ‘Assignment matrix’ option, it will gather assignment information for a number of links. The link ids are specified in a file called “assign_links.dat”, that should be in the same directory as the master (*.mezzo) file. The structure of the assign_links.dat file is as follows (Fig. 12).

```
{ LinkID1 LinkID2 ... LinkIDn }
```

```
no_obs_links: 8
{ 2 3 4 5 6 7 8 9 }
```

Figure 14. Assign_links.dat file format

The data in the assignment matrix contains flow counts at the links specified, grouped by OD pair, departure time period at the origin and time period of arrival at the link. This information is used for (re-)estimation of the OD matrix by external modules.

3.11 Parameters file

The Parameters file contains the values of a set of parameters for many different processes, varying from GUI parameters to parameters for output data collection. While any of the parameters may be adjusted, the default values can be used for most applications.

```
#drawing_parameters
  draw_link_ids= 0
  link_thickness= 1
  node_thickness= 2
  node_radius= 10
  queue_thickness= 6
  selected_thickness= 3
  show_background_image= 1
  linkcolor= black
  nodecolor= yellow
  queuecolor= red
  backgroundcolor= darkBlue
  selectedcolor= green
#moe_parameters
  moe_speed_update= 900.0
  moe_inflow_update= 900.0
  moe_outflow_update= 900.0
  moe_queue_update= 900.0
  moe_density_update= 900.0
  linktime_alpha= 0.2
#assignment_matrix_parameters
  use_ass_matrix= 1
  ass_link_period= 1200.0
```

```

    ass_od_period= 1200.0
#turning_parameters
    default_lookback_size= 20
    turn_penalty_cost= 99999.0
#server_parameters
    od_servers_deterministic= 1
    odserver_sigma= 0.2
#vehicle_parameters
    standard_veh_length= 7
#route_parameters
    update_interval_routes= 300.0
#mime_parameters
    mime_comm_step= 0.5
    mime_min_queue_length= 20
    mime_queue_dis_speed= 6

```

Figure 15. Example of Parameters file

In figure 15 an example of the parameters file is shown, which is divided into a number of sections. The meaning of the parameters is the following:

- **#drawing_parameters:** The drawing parameters section
 - **draw_link_ids:** *0 or 1*. Determines whether or not the link IDs are shown in the drawing of the network
 - **link_thickness:** *1..n*. Determines the thickness with which the links are drawn.
 - **node_thickness:** *1..n*. Determines the thickness with which the nodes are drawn.
 - **node_radius:** *1..n*. Determines the radius with which the nodes are drawn
 - **queue_thickness:** *1..n*. Determines the thickness with which the size of the queues on links are drawn.
 - **selected_thickness:** Determines the thickness with which selected objects (nodes or links) are drawn.
 - **show_background_image:** *0 or 1*. Determines whether the background image (if available) is shown or not.
 - **linkcolor:** *string*. Determines the color with which the links are drawn

- **nodecolor:** *string*. Determines the color with which the nodes are drawn.
- **queuecolor:** *string*. Determines the color with which the queue-bars on links are drawn.
- **backgroundcolor:** *string*. Determines the color with which the background is filled (has no effect in case a background image is displayed).
- **selectedcolor:** *string*. Determines the color with which the selected objects are drawn.
- **#moe_parameters:** the section for MOE (Measures Of Effectiveness) parameters.
 - **moe_speed_update:** *double*. Determines the interval size for the collection of link speed data.
 - **moe_inflow_update=** *double*. Determines the interval size for the collection of link inflow data.
 - **moe_outflow_update=** *double*. Determines the interval size for the collection of link outflow data.
 - **moe_queue_update=** *double*. Determines the interval size for the collection of link queue length data.
 - **moe_density_update=** *double*. Determines the interval size for the collection of link density data.
 - **linktime_alpha=** *double*. Determines the parameter for smoothing the output link traveltimes with the input (historical) link travel times. NOTE: this parameter is also present in the Master file (see p.3-4). The value given in the master file overrides the one in the parameter set. I am not sure yet where is the right location of this parameter.
- **#assignment_matrix_parameters**
 - **use_ass_matrix:** *0 or 1*. Determines whether data for the assignment matrix is collected or not.
 - **ass_link_period:** *double*. Determines the interval size for arrival times at links for the collection of assignment data.
 - **ass_od_period=** *double*. Determines the interval size for departure times from the origin for the collection of assignment data
- **#turning_parameters:** Parameters specific to the turning movements

- **default_lookback_size:** *int*. In case a new turnings file needs to be generated, this value is used for the default look-back limit of the generated turnings.
- **turn_penalty_cost:** *double*. This value is used for calculating alternatives to blocked turnings. The penalty is added to 'blocked turnings' in the shortest path tree and alternatives are calculated.
- **#server_parameters**
 - **od_servers_deterministic:** *0 or 1*. Determines whether the traffic generation at origins is deterministic or stochastic. If stochastic, the headways of consecutively generated vehicles follow a shifted negative exponential distribution with mean equal to $1/\text{od_flow_rate}$.
 - **odserver_sigma:** *double*. For future use. In case other processes are used (combined neg-exp with truncated normal for example) the OD servers need a standard deviation in addition to the mean.
- **#vehicle_parameters**
 - **standard_veh_length:** *int*. Determines the length in meters for average vehicles. Used to initialise links with a certain average capacity. Depending on vehicle types and their lengths, the actual capacity of a link in number of vehicles may be lower.
- **#route_parameters**
 - **update_interval_routes:** *double*. The interval with which the route costs are re-calculated based on the historical link travel times. (and updated link travel times in case of real-time travel time information).
- **#mime_parameters**
 - **mime_comm_step:** *double*. The interval with which Mezzo communicates with the Micro model (VISSIM, MITSIM or another model). In case of VISSIM it determines the VISSIM simulation time step.
 - **mime_min_queue_length:** *int*. The minimal queue length before the queue dissipation speed is applied.
 - **mime_queue_dis_speed:** *int*. The queue dissipation speed of vehicles discharging from a queue that transcends a Mezzo-VISSIM boundary.

4. Output files

The output files are formatted in much the same manner as the input files. However, with exception of the Linktimes file, the output files do not have curly brackets { and }, to enable easy processing of the data in programs such as Excel or Matlab.

4.1 Linktimes

The Linktimes file contains the time-dependent link travel times produced by the simulation. As explained in section 1. they can be averaged with the input travel times (to allow for easy iteration) dependent on the value of `traveltime_alpha` in the master file. To eliminate averaging set `traveltime_alpha` to 1. The format of Linktimes is the same as for Histtimes:

`links`: indicates the number of links for which the link travel times are defined.
`periods`: indicates the number of time periods for the link travel times.
`periodlength`: indicates the duration of each period in seconds.

Per link the travel times are defined as follows:

```
{ LinkID   Traveltime1  Traveltime2  ...   TraveltimeN }
```

Update: from now in addition to the `linktimes.dat` file, a clean version of the link travel times (without averaging with the input link travel times) is stored in a file called `linktimes.dat.clean`

4.2 Output

This file contains detailed Origin Destination output. Per OD pair it outputs for each arrived vehicle: `Origin_id`, `Destination_id`, `Vehicle_id`, `Start_time`(in seconds), `Arrival_time` (sec), `Travel_time` (sec), `Travelled distance` (in meters) and whether the vehicle switched from its original route (0=no, 1=yes). The format is:

```
OriginID DestID VehicleID Start_time Arrival_time Travel_time Travelled_distance  
Switched
```

```

124 115 1 0.1 91.0482 90.9482 1730 0
124 115 301 44.0306 134.979 90.9482 1730 0
124 115 907 172.017 264.063 92.0462 1730 0
...

```

Figure 14. Output file with OD data per arrived vehicle.

OriginID, DestID, VehicleID and Switched are integers, Start_time, Arrival_time, Travelled_distance are doubles.

4.3 Summary

This file contains a summary of the OD output. Per OD pair it outputs:

OriginID DestID Total_vehicles_arrived Total_Travel_Time Total_Dist_Travelled

```

124 115 101 12807.7 174730
124 98 305 48322.9 817400
124 75 428 247433 2.43461e+006
...

```

Figure 16. Summary file with aggregated OD data.

OriginID, DestinationID, Total_vehicles_arrived are integers, Total_Travel_Time and Total_Dist_Travelled are doubles. Total_Travel_Time is in seconds, Total_Dist_Travelled is in meters.

4.4 Speeds

Contains average link (traversal) speeds for each time period defined in the MOE (Measures Of Effectiveness) collection parameters (currently defined in the Parameters.h file, will be moved to master file, or a separate parameter file later on).

The format of the data is:

```

LinkID      Speed1      Speed2      ...      Speedn

```

0	88.9641	89.8519	89.16	91.7894	90.801	
1	0	100.8	100.8	100.667	98.1426	95.9354
2	94.7019	97.3699	97.7938	98.3335	98.0281	
...						

Figure 17. Speeds output file.

The current time period for MOE collection of speeds is set to 60 seconds.

4.5 Inflows/Outflows

Contains average link inflows/outflows (in Vehicles / hour / lane) for each MOE time period (set in Parameters.h for the moment). The current MOE time period for inflows is set to 60 seconds. The format of the data is:

LinkID Flow1 Flow2 Flow3 ... Flown

0	1480	1340	1260	1280	1300	1760	1460	1320	1440	1720	1220
1	0	140	180	540	1100	1020	1120	1220	900	980	1480
2	825	1020	975	975	945	1125	1185	1110	990	1185	1125
...											

Figure 17. Inflows/Outflows output files.

4.6 Queuelengths

Contains the average queuelengths on links for each MOE time period. The format is similar to that of Inflows/Outflows or Speeds:

LinkID Queuelength1 Queuelength2 ... Queuelengthn

4.7 Densities

Contains the average densities on links for each MOE time period. The format is similar to that of Inflows/Outflows:

LinkID Density1 Density2 ... Densityn

4.8 Assignment matrix

The assignment matrix is stored in a file called “assign.dat” in the same directory as the master file and “assign_links.dat” input file. This file is mainly used to determine the which proportions of flows observed at certain links (defined in assign_links.dat) come from the different OD pairs, and different time periods. This information can be used by external modules to provide (re-)estimation of the Mezzo input OD matrices.

```
no_obs_links: 2
no_link_pers: 30

link_period: 0
link_id: 0
no_entries: 2
{ 0 2 0 399 }
{ 0 4 0 199 }
link_id: 1
no_entries: 2
{ 0 2 0 295 }
{ 3 2 0 198 }
...
```

Figure 18. Example of assign.dat assignment matrix

In Figure 18 an example of such an assignment matrix is shown.

- No_obs_links indicates the number of observed links (as defined in the assign_links.dat file).
- No_link_pers indicates the number of link time periods in the matrix.
- After that for each Link_period and each Link_id a number of entries are defined:

{ OriginID DestinationID DeparturePeriod NumberOfVehicles }

Each entry records for the link and time period defined the Number of vehicles that passed, to which OD pair they belong, and in which DeparturePeriod they departed from the origin.

4.9 Virtual queues

Another file that aids external OD estimation modules is the “v_queues.dat” which is also written in the same directory as the master file (and the assign.dat file). It contains the lengths of queues at the origins, which may arise if the capacity of the simulated network does not allow all vehicles to enter. For each departure time period these virtual queue lengths are recorded. This information can be used in OD estimation procedures to signal when demand for certain OD pairs has exceeded the capacity of the network.

```
{ DeparturePeriod  OriginID      NumberOfVehicles }
```

Each entry in the v_queues.dat file specifies the number of vehicles in the virtual queue at an origin, at the end of DeparturePeriod, which are the same periods as used in the assignment matrix. Only entries with a NumberOfVehicles > 0 are recorded in the v_queues.dat file.

References

Burghout, W., 2004. Hybrid microscopic-mesoscopic traffic simulation, PhD. Thesis, Royal Institute of Technology, Stockholm.

May, A.D., 1990. Traffic Flow Fundamentals. Englewood Cliffs, NJ