Mezzo file formats

Version 0.1.9

MEZZO Version 0.61.2

Wilco Burghout, 2012-08-01

Contents

1. Introduction 3

1.0 What's new in this version 3

1.1 How to install and run Mezzo 3

2. Master file (*.mezzo, *.mime) 4

3. Input files 9

3.1 Network file 9

3.2 Turnings file 13

3.3 Signal Control 14

3.3 Histtimes 16

3.4 Routes 17

3.5 Demand 17

3.6 Incident File 19

3.7 Vehicle types 19

3.8 Virtual Links 19

3.9 Server Rates 20

3.10 Assignment links 21

3.11 Parameters file 21

4. Output files 29

4.1 Linktimes 29

4.2 Output 30

4.3 Summary 31

4.4 Speeds 31

4.5 Inflows/Outflows 32

4.6 Queuelengths 32

4.7 Densities 32

4.8 Assignment matrix 32

4.9 Virtual queues 33

References 34

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.`

1.0 What's new in this version

1. A number of new parameters have been added to the parameters file, see the Parameters file section (3.11)
2. Inflow in links is now regulated by a max inflow capacity, represented by the min_inflow_headway parameter in the parameters secion
3. Signal control can now be either stage-based or signal group based. For ease of use the signal group definitions are still called "stages" but are now independent of each other, whereas the stage-based type will treat stages sequentially and mutually exclusive.
4. Speed/density functions now accept doubles for the Vmax, Vmin, Kmax and Kmin parameters, instead of integers.

1.1 How to install and run Mezzo

There are two versions of Mezzo, one with and one without the hybrid option. Moreover each version (hybrid or regular) exists as a GUI or standalone version. The installer mezzo_setup.exe (or mezzo_hybrid_setup.exe) installs all relevant files to the path chosen by the user, the default being c:\Program Files\Mezzo (or C:\Program Files\MezzoHybrid). The necessary Program icons and shortcuts for the GUI are installed as well (Mezzo / MezzoHybrid program group) and the user will usually run the GUI versions from there.

Executables:

- Mezzo_gui.exe : GUI version of Mezzo
- Mezzo_Editor.exe : Editor for Mezzo Networks
- Mezzo_s.exe : Standalone version for Mezzo.

The Mezzo_s standalone version is compiled without any of the GUI structures, and is therefore much faster to run, especially in batch files, or other scripts. It will automatically save the results after finishing. Mezzo_s has the following command line structure:

```
mezzo_s.exe  <masterfile>  <random_seed>
```

also the GUI version can be called with a masterfile and random seed if wanted

```
mezzo_gui.exe          <masterfile>  <random_seed>
```

In both cases the random seed is optional. In case the mezzo_gui is called with a masterfile, it will behave as mezzo_s, opening the network, running the iterations as specified in the #iteration_control section of the Parameters file, and saving the results, overwriting the input link travel times with the output times.

The master file should end in *.mezzo (or *.mime). The random_seed is optional.

If in the #iteration_control section of the Parameters file the max_iter > 1, the model will iterate until the convergence criterion is reached, or the max_iter is reached.

If the random_seed is specified, this will be used for all random processes in the simulation. If random_seed is not specified, the random processes will be started with randomized seeds.

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 may 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
starttime= 0
stoptime= 8100
calc_paths= 0
parameters= parameters.dat
background= Stockholm.png
```

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. Obligatory
- Signals: definition of the signal control plans, stages, timings etc. Not obligatory.
- Turnings: definition of the turning movements. Not obligatory, can be generated automatically (using default values) if missing. Give-way relations between turnings are defined here as well.
- Histtimes: Historical link travel times. These times are used for the pre-trip route choice. If no times are available, free-flow times are used. Output link times have the same format and can be copied to this file. Not obligatory.
- Routes: Description of the set of known routes for all OD pairs (chains of links). Not obligatory.
- 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. Obligatory.
- Incident: file that describes all incidents, when they occur, on what link, what the link penalty (delay) 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". Not obligatory.
- Vehicletypes: In this file the vehicle mix is described. Per vehicletype the type name, percentage and length in meters. Obligatory.
- Virtuallinks: This file is MiMe (hybrid) 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). Not obligatory.
- Serverrates: 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). Not obligatory.

#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_generated, 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 (defined in the Parameters.h file) (km/h)
- Inflows: Contains average link inflows for each MOE time period. (veh/h)
- Outflows: Contains average link outflows for each MOE period. (veh/h)
- Queuelengths: Contains the average queue length on each link for each MOE period.
- Densities: Contains average densities on each link for each MOE period. (veh/km/lane)
- In addition to these output files, several more output files are generated in the same directory as the master (*.mezzo/*.mime) file:
    - convergence.dat: contains the Relative Gap convergence measures for link trvale times and route flows, for each iteration :

```
Iteration  RGAP_Linktimes  RGAP_Routeflows
1    0.774263   1
2    0.281116   0.0103783
3    0.109971   0.00379886
4    0.0441117  0.00140988
5    0.0177938  0.000587452
```

```
6     0.00785029 0.000274144
```

- routeflows.dat: contains the routef lows (in counts: number of vehicles) for each route and each OD matrix time period (each OD slice) format: Route_ID   Routeflow_1 .... Routeflow_n

```
796   0     2     2     2     2     1     4     3
797   2     4     0     3     1     0     0     0
798   3     1     3     1     2     2     4     2
799   2     1     0     2     1     2     2     1
```

- v_queues.dat: Virtual queues at origins. In case vehicles cannot be loaded into the network at origins, they are put into a virtual queue, from where they will be loaded when space becomes available. This file logs virtual queue length
- assign.dat: contains the assignment matrix in case the use_ass_matrix = 1 in the parameters file and the assign_links.dat contains > 0 links. See Section 3.10 Assignment links
- debug_log.txt: contains debug info regarding possible runtime problems such as reading of input files, creation of objects etc. This file is created in the directory from which Mezzo is started. (default location in case of Mezzo GUI: C:\Program Files\Mezzo)

`#scenario`

- Starttime (for future use): starting time of the scenario. For future use, should be 0 for now.
- Stoptime: 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.
- 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:

```
Background=
```

3. Input files

3.1 Network file

The network file contains definitions of four key components: 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 ←"servers:" NUMBER SERVER*
NUMBER ←integer
SERVER ← "{" SID STYPE MEAN SD DELAY "}"
SID ← integer
STYPE ←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, unlimited capacity.
    1. Normal (Mean, StdDev) server (truncated at one side by min_headway=0.1 (constant))
    2. Deterministic (Mean) server.
    For types 0 and 2, unused attributes such as StdDev are ignored, but should still be specified (to keep standard syntax when switching server types)
- 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/travel times (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, 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 } when SDType = 0

{ SDID SDType VMax VMin KMax KMin } when SDType = 1

{ SDID SDType VMax VMin KMax KMin Alpha Beta } when SDType = 2

Where:

- SDType indicates the type of SD function:
    0. Dummy Server, returns VMax for any density
    1. Piece-wise linear (same as 2. with Alpha = Beta = 1)
    2. Generalised non-linear according to (Burghout 2004) Flexible, but computationally slower than 1.

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

SDID, and SDType are integers; VMax, VMin, Kmax, Kmin, 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. Note that the number of lanes is now a double, so may be non-integer! 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

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 the filename is empty in the Master file. In this case all turning movements will have the first Server specified in the Servers section in the Network file, and default values for lookback and delay.

```
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.
Note that multiple turnings may be defined between the same in- and outgoing links. For instance to provide multiple lane-turns where one turn comes from a shared lane and the other from a dedicated lane.

The give-way relations 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 signal plan. The structure is as follows:

```
controls: Nr_Controls
{ ControlID      ControlType      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.
The Control type defines if the signal is stage-based or signal-group based:

- Control type 1 = Stage based
- Control type 2 = Signal group based

If the control type is stage-based, the stages are sequential and mutually exclusive, so stage 1 has to end before stage 2 can start etc. (with a possible all-red in between)
If the control type is signal group based, the stages represent signal groups, which have their own timing, independent from each other. I.e. multiple stages(signal groups) can have green at the same time
In figure 13 an example is shown of a signal.dat file with only one control defined,which is stage-based and 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       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.

In case no traffic signals are specified, provide the following stub:

```
controls: 0
```

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 9a. 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   }

In case no historical link travel times are available, an empty stub can be provided, where the number of links are set to 0:

```
links:      0
periods:    4
periodlength:    600
```

Figure 9b. Empty historical travel times.

In this case the link travel times are calculated from the link free-flow speeds.
The number of periods and periodlength will still be used when creating the output link travel times, which can serve as input (historical travel times) for subsequent runs.

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 if the calc_routes = 1 (in the master file) 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 10a. Route file.

Each route definition is constructed as follows:
{ RouteID OriginID DestinationID NumberOfLinks { LinkID1 LinkID2 … LinkIDn } }

When no routes are available an empty stub should be provided:

```
routes: 0
```

Figure 10b. Route file.

Also the calc_routes = 1 should be defined in the master file, to ensure new routes are calculated and added to the routes file.

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*. OD pairs may have 0 demand (meaning they are inactive until they get demand in later slices). Likewise when in subsequent slices demand goes to 0, the OD pair becomes inactive. For instance in the last slice one can have all OD pairs set to 0 to let the network empty itself.

```
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 od-pairs in the following 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 Hybrid application, to define the virtual links that represent the paths inside the microscopic area.

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

{ LinkID StartNodeID EndNodeID Length Nr_Lanes SDID V_EnterParkingLot V_ExitParkingLot V_Last_link V_Nr_Nodes {V_Node_1 … V_Node_n } }

LinkID, StartNodeID, EndNodeID, Length, Nr_Lanes, SDID are exactly as in normal links, where StartNodeID is the BoundaryOut node and EndNodeID the BoundaryIn node.
V_EnterParkingLot and V_ExitParkingLot are the entry and exit parking lots in VISSIM, corresponding to the BoundaryOut and BoundaryIn nodes in Mezzo. V_Last_link is the exiting link in VISSIM (leading to the V_ExitParkingLot). The V_Nodes are the VISSIM Nodes through which the path must lead. To enable easier conversion from VISSIM paths, double nodes are ignored.

NOTE this format is subject to changes as the format of communication between VISSIM and MEZZO changes. For instance, in the near future the V_Nr_Nodes and the list of V_Nodes may be dropped, when we can use VISSIMs own dynamic paths.

3.9 Server Rates

The Server rates file is used to vary the throughput rates (capacities) 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 standard deviation 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 in the Parameters files the parameter *use_ass_matrix= 1*, Mezzo will gather assignment map 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. The output assignment matrix (4.8 Assignment matrix) is written to 'assign.dat' also in the same directory as the master file.

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
  #simulation_view
   draw_link_ids= 0
   link_thickness= 1
   node_thickness= 1
   node_radius= 6
   queue_thickness= 8
   selected_thickness= 10
   text_size= 12
   show_background= 1
   background_x= 0
   background_y= 0
   background_scale= 5
   linkcolor= grey
   nodecolor= grey
   queuecolor= red
   backgroundcolor= white
   selectedcolor= green
   gui_update_step= 0.2
   zerotime= 27000
  #output_view
   thickness_width= 20
   cutoff= 5
   show_link_names= 0
   show_link_ids= 0
   show_data_values= 0
#moe_parameters
   moe_speed_update= 1800.0
   moe_inflow_update= 1800.0
   moe_outflow_update= 1800.0
   moe_queue_update= 1800.0
   moe_density_update= 1800.0
   linktime_alpha= 0.6
#assignment_matrix_parameters
   use_ass_matrix= 0
   ass_link_period= 1800.0
   ass_od_period= 1800.0
#turning_parameters
   default_lookback_size= 20
   turn_penalty_cost= 99999.0
   use_giveway= 1
   max_wait= 1800.0
   min_headway_inflow= 1.44
#server_parameters
   od_servers_deterministic= 1
   odserver_sigma= 0.2
   implicit_nr_servers= 0
#vehicle_parameters
   standard_veh_length= 7
#route_parameters
   update_interval_routes= 1800.0
   mnl_theta= -0.00417
   kirchoff_alpha= -1.0
   delete_bad_routes= 0
   max_rel_route_cost= 2.0
   small_od_rate= 1.0
   use_linktime_disturbances= 1
   linktime_disturbance= 0.1
   routesearch_random_draws= 3
   scale_demand= 0
   scale_demand_factor= 0.5
   renum_routes= 1
   overwrite_histtimes= 0
#mime_parameters
   mime_comm_step= 0.4
   mime_min_queue_length= 20
   mime_queue_dis_speed= 6
   vissim_step= 0.1
   sim_speed_factor= 1.9
#iteration_control
   max_iter= 10
   rel_gap_threshold= 0.01
   max_route_iter= 3
```
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
- **#simulation_view:** for the simulation view

- **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:** *1..n* Determines the thickness with which selected objects (nodes or links) are drawn.
- **text_size:** >=*6* Determines the text size when link or node id's and names are drawn on the map
- **show_background:** *0 or 1*. Determines whether the background image (if available) is shown or not.
- **background_x:** *double* Determines the x-offset of the background image to the network
- **background_y:** *double* Determines the y-offset of the background image to the network
- **background_scale:** *double* Determines the scaling of the background image.
- **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.
- **gui_update_step:** *string*. Determines the gui refresh rate.
- **zerotime:** *0..n* Determines the clock time of the start of the simulation. Only for GUI display purposes, the internal simulation time is always from 0.

- **#output_view**: The GUI parameters for the output view mode, which is used for the display and analysis of the results
    - **thickness_width:** *1..n* The maximum link width of the selected output MOE.
    - **cutoff:** *1..100* Percentage below which link MOE values (as well as names and ids) are not displayed, with regard to the selected thickness MOE.
    - **show_link_names=** *0 or 1*. If 1 the link names are displayed.
    - **show_link_ids=** *0 or 1*. If 1 the link ids are displayed.
    - **show_data_values=** *0 or 1*. If 1 the link MOE data values are displayed (if above the cutoff) in the format "(thickness MOE / colour MOE)"
- **#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. The output travel times are calculated as follows:

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

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]

- **#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.
  - **use_giveway=** *0 or 1*. If 1 the give-way logic is used for those turning movements that have give-way relations defined in the turnings file.
  - **max_wait=** *double*. Defines the maximum wait for a vehicle on a minor turn (giving way to a major turn), in seconds. After this period the vehicle get through regardless of give-way priorities. (Defines minimum capacity of minor turn (=3600/max_wait)
  - **min_headway_inflow=** *double*. Defines the minimum headway (or max capacity = 3600/min_headway_inflow, per lane) for inflow on links, multiplied by the number of lanes on the link. This guarantees a max capacity flowing into a link, as the sum of all inflows from turnings into a link is capped by this value.
- **#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/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.
  - **implicit_nr_servers:** *0 or 1*. For backward compatibility. If 1, the server capacity per turning is multiplied by the number of effective turning lanes (i.e. minimum of ingoing and outgoing lanes). Standard value is now 0 and servers should explicitly reflect the turning capacity, for ALL effective turning lanes, OR multiple turnings should be defined for each effective turning lane.
- **#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).
  - **mnl_theta:** *double*. The scaling parameter for the Multi Nomial Logit route choice, in case this is used (currently kirchoff is active). < 0.
  - **kirchoff_alpha:** *double*. The exponent for Kirchhoff route choice (currently active). < 0. -1 equals proportional utilities, larger negative numbers imply stronger than proportional preference for better alternatives.
  - **delete_bad_routes:** *1 or 0*. If 1 the route set is sorted for each OD pair and bad routes (according to max relative route cost) are deleted.
  - **max_rel_route_cost:** *double*. If delete_bad_routes = 1, for each OD pair all routes that are worse than this factor times the best route are deleted.
  - **small_od_rate:** *double*. For future use. To be used to limit the amount of routes available for OD pairs with small rates.
  - **use_linktime_disturbances:** *1 or 0*. If 1 route searches are repeated with a slight noise (linktime_disturbance) added to the linktimes, to find multiple very similar routes in one RouteSearch iteration.

- **linktime_disturbance:** *double, 0.0 – 1.0* (% of link travel times) If use_linktime_disturbances is 1, this defines the factor of disturbance. Disturbances are drawn from a U(-0.5,0.5) distribution, multiplied by this factor and the link travel time.
- **routesearch_random_draws:** *int 0-…* Defines the number of re-draws for the link time disturbances in the route search. For each redraw a new search is performed for all time periods.
- **scale_demand:** *1 or 0*. For future use. If 1 the demand is scaled down in early iterations
- **scale_demand_factor:** *double*. If scale_demand is 1, this is the factor with which the demand is scaled down for early iterations.
- **renum_routes=** *1 or 0*. If 1 the routes are renumbered after each new route search.
- **overwrite_histtimes:** *1 or 0*. If 1the histtimes are overwritten with the new linktimes after execution of the SDUE loops, when the program quits.

- **#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.
  - **vissim_step:** *double*. sets the VISSIM simulation time step.
  - **sim_speed_factor:** *double*. If > 0 sets a fixed simulation speed relative to real time. FOR FUTURE USE.
- **#iteration_control**
  - **max_iter:** *int*. Determines max number of SDUE iterations, if 1 only 1 iteration is run
  - **rel_gap_threshold:** *double*. Determines the threshold for the Relative Gap for output and input link travel times and / or route flows (depending on what is selected in Batch run window):

    Relgap Linktimes:

$$rel_{gap} = \frac{\sum_t \sum_{l \in L} \left| \hat{\tau}_l^t - \tau_l^t \right|}{\sum_t \sum_{l \in L} \hat{\tau}_l^t}$$

Where,
$\hat{\tau}_l^t$ are the input link travel times
τ_l^t are the output link travel times
For all time periods t and links l.

Relgap Routeflows:

$$rel_{gap} = \frac{\sum_t \sum_{i \in I} \sum_{k \in K_i^t} \left| f_k^t - q_i^t P_k^t(\tau_k^t) \right|}{\sum_t \sum_{i \in I} \sum_{k \in K_i^t} f_k^t}$$

Where,
f_k^t are the route flows (based on input travel times)
τ_k^t are the output travel times
q_i^t is the OD demand
$P_k^t(\tau_k^t)$ are the route proportions based on output travel times.
For all time periods t, OD pairs i and routes k.

When the relative gap < rel_gap_threshold, or when max_iter is reached, the iterations stop.

- **max_route_iter:** *int.* Determines max number of Route Search iterations, if 1 only 1 iteration is run.

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 (and usually are) averaged with the input travel times (to allow for easy iteration) dependent on the value of traveltime_alpha in the master file. In addition, a file is written with the clean output link times (no averaging) in a file with the string '.clean' appended to the linktimes filename. 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 }

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), **Route_ID** and whether the vehicle switched from its original route (0=no, 1=yes). The first line contains the field names. The format is:

OriginID DestID VehicleID Start_time Arrival_time Travel_time Travelled_distance **RouteID** Switched

```
origin_id  dest_id    veh_id     start_time end_time
     travel_time      mileage    route_id   switched_route
124 115 1 0.1 91.0482 90.9482 1730 3 0
124 115 301 44.0306 134.979 90.9482 1730 6 0
124 115 907 172.017 264.063 92.0462 1730 4 0
…
```

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

OriginID, DestID, VehicleID and Switched are integers, Start_time, Arrival_time, Travel_time and 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_generated 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, defined in the parameters file. The format of the data is:

LinkID Speed1 Speed2 … Speedn

```
0     78.2024    87.7841    91.0737    94.1908
1     100.337    100.187    99.8486    99.502
2     99.4499    100.206    100.271    99.7674
...
```

Figure 17. Speeds output file.

4.5 Inflows/Outflows

Contains average link inflows/outflows (*in vehicles/hour*) for each MOE time period (set in parameters file) 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. In the parameters file the variable use_ass_matrix = 1 should be set. Also the ass_link_period and ass_od_period define the link and OD time intervals.

```
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