CoEXist

Deliverable 2.7

AV-ready macroscopic modelling tool

Version: 5.0

Date: 2020-02-02

Author: Jörg Sonnleitner, Markus Friedrich

The sole responsibility for the content of this document lies with the authors. It does not necessarily reflect the opinion of the European Union. Neither the EASME nor the European Commission are responsible for any use that may be made of the information contained therein.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201-2

Document Control Page

Title	AV-ready macroscopic modelling tool
Creator	Jörg Sonnleitner
Editor	Jörg Sonnleitner, Markus Friedrich
Brief Description	Documentation about the AV-ready macroscopic modelling tool
Publisher	
Contributors	Jörg Sonnleitner, Markus Friedrich, Maximilian Hartl, Johann Hartleb, Emely Richter, Alexander Migl (University of Stuttgart)
Type (Deliverable/Milestone)	Deliverable
Format	Document related to script and dll files
Creation date	2018-10-08
Version number	5.0
Version date	2020-02-05
Last modified by	Jörg Sonnleitner
Rights	
Audience	☐ Internal ☒ Public ☐ Restricted, access granted to: EU Commission
Action requested	☐ To be revised by Partners involved in the preparation of the Deliverable ☐ For approval of the WP Manager ☐ For approval of the Internal Reviewer (if required) ☐ For approval of the Project Co-ordinator
Deadline for approval	2018-10-31

Version	Date	Modified by	Comments
5.0	2020-02-05	Jörg Sonnleitner	Revision to include code for Visum 2020

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Table of contents

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

1 Introduction

1.1 Purpose of this document

This document describes the tools developed by USTUTT within the CoEXist project. The purpose of these tools is to enable macroscopic travel demand models implemented with the software Visum to incorporate automated vehicles (AV) and new mobility services like ridesharing and to compute their impacts. The methodology applied through the tools may be used to develop features for other macroscopic modelling software as well. The modeller must take care on the proper usage and required adjustments of the presented tools.

A detailed instruction on how to apply the tools, the approaches together with assumptions made and further recommendations for the model user is included in the 'Guide for the simulation of AV with macroscopic modelling tool' (D2.8).

1.2 Scope

The tools provide extensions to Visum by adding functionality to the software in form of Visum compatible scripts, Visum procedure files or Visum Add-Ins. The tools provided are not integrated directly in Visum. The model user has to plug them into Visum to make them work and to perform a certain task. They assist the model developer or model user by extending the capabilities of Visum.

Every model is different and has specific characteristics. The development and testing of the tools was mainly done using the Stuttgart Region travel demand model and partly for smaller, pilot models. Still, the tools have been designed in such a way that they can be applied to all travel demand models implemented in Visum, if certain inputs are available.

Apart from the automation of some tasks, the modeller still needs to adapt settings or adjust parts in Visum that cannot be accessed in another way. Therefore, it is required that the model user is familiar with working with Visum.

1.3 Overview

Traditional travel demand models apply the four-stage algorithm, where trip generation, destination choice, mode choice and route choice are covered to replicate people's behavior and their movement. Departure time choice may also be considered as a step. Integrating automated vehicles or new mobility services into these models requires to establish and include new steps in the procedure. This could be for example the bundling of trips as well as the scheduling of vehicles for a ridesharing system with self-driving vehicles. Besides adding new model stages, impacts of AV on supply and demand have to be taken into account on all stages of a travel demand model.

Figure 1 shows the extended sequence of a travel demand model, which includes assumed impacts of AV. Topics covered in this document are labelled with the corresponding chapters. Each main chapter consists of a short introduction where the relevance of the specific topic is discussed along with a

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

description of the related tool. The tools are included as code in the Appendix and are available for download.

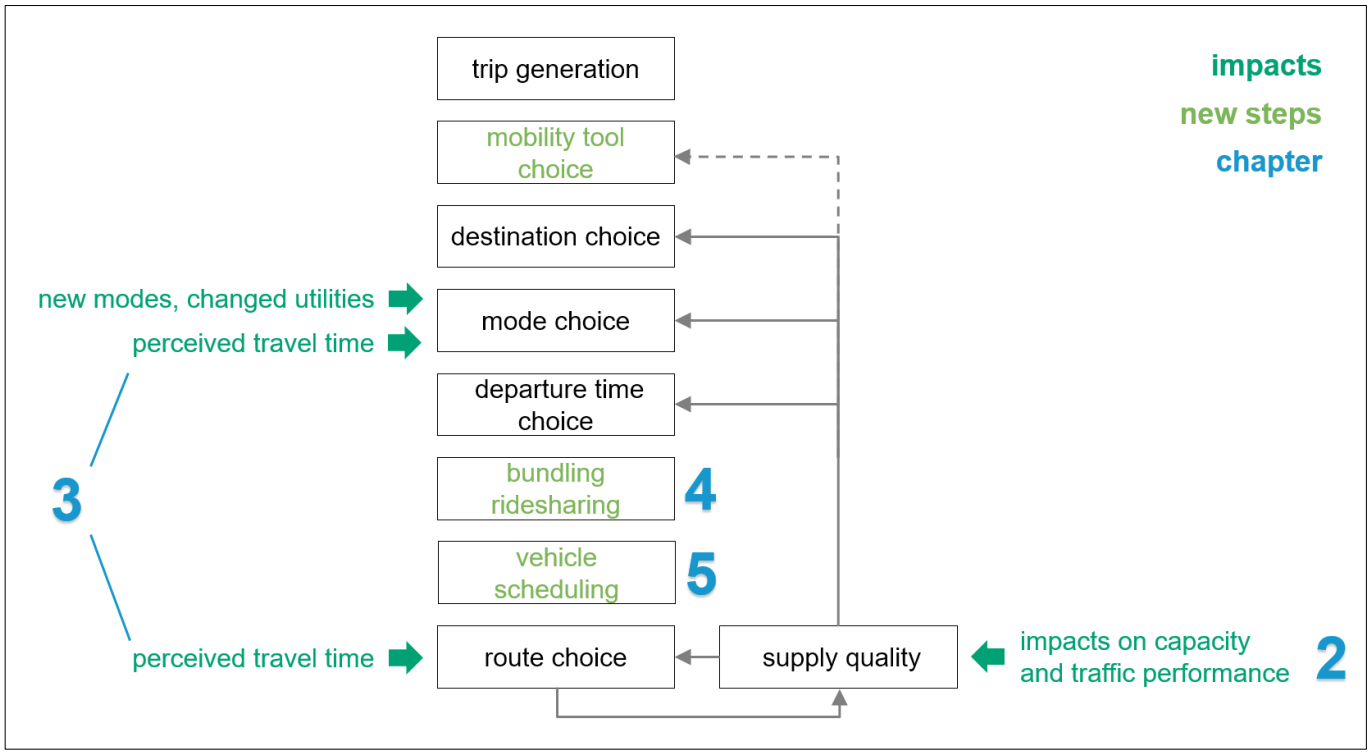


Figure 1: Modelling AV with macroscopic travel demand models

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

2 Volume-Delay Functions

2.1 Relevance

Volume-delay functions determine the congested travel times of links and turns in macroscopic travel demand models. The congested travel times depend on the free flow travel time and the level of saturation, i.e. ratio of traffic volume to capacity. The higher this ratio, the larger the delay time. The relationship between saturation and travel time is influenced by the type and the parameters of the volume-delay function. Usually a model applies a set of volume-delay functions and assigns specific volume-delay functions to every road or intersection type.

Incorporating AV into a travel demand model requires volume-delay functions considering the share of AV and the characteristics of AV.

2.2 Tool characteristics

The tool for extending volume-delay functions to incorporate AV consists of four parts, which are described in Table 1. The content of the corresponding files is provided in Chapter 6.1 as code.

Table 1: Overview on tools related to volume-delay functions with AV

Short description	Input	Output	Implementation
create user-defined attributes for handling volume-delay functions	Visum travel demand model	specific user-defined attributes on network, link and link type level required for modelling AV	Visual Basic Script (VBS)
create formula matrices for splitting the original demand of car driver into a demand for CV and AV according to the AV share	Visum travel demand model with an appropriate set up regarding attributes and names for transport systems	formula demand matrices on zone level for CV and AV	Visual Basic Script (VBS)
procedure sequence for setting the attributes used by user-defined volume-delay functions	an existing procedure sequence in a Visum travel demand model	added group of procedures containing a set of operations to enable the model to handle the user-defined volume-delay functions	Extensible Markup Language (XML)
user-defined volume-delay function with constant or variable PCU factors for automated vehicles	Visum travel demand model with an appropriate set up regarding attributes and names for transport systems	specific user-defined volume-delay functions including a visual representation for the calculation rule (formula)	Dynamic Link Library (DLL) and Windows Bitmap (BMP)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

3 Perception of automated travel time

3.1 Relevance

The way people perceive time during a trip depends on the type of activity they are engaged in: walking to a vehicle, waiting time at a stop, driving time as a driver or as a passenger. To capture this perception in a choice model, every time component is weighted with a specific factor to describe the perceived travel time of a trip.

Automated vehicles that are able to drive automatically on certain road types or on AV-ready network sections offer the possibility for the driver to spend some time of the trip on non-driving tasks. This share of the trip time is henceforth referred to as ‘automated travel time’. Depending on the duration of the automated travel time, people may experience and value the in-vehicle time differently. A one hour automated drive with time for non-driving tasks will be perceived as a shorter time period. Consequently, the attractiveness of private car transport will increase. This should be taken into consideration in a travel demand model within destination, mode and route choice.

3.2 Tool characteristics

The toolbox for integrating the impacts of perceived automated travel time consists of three parts, which are described in Table 2. The content of the corresponding files is provided in Chapter 6.2 as code.

Table 2: Overview on tools related to perceived travel time

Short description	Input	Output	Implementation
create user-defined attributes for handling a different perception of automated travel time	Visum travel demand model	specific user-defined attributes on network, link type, link and matrix level	Visual Basic Script (VBS)
procedure sequence for setting the values for including perceived automated travel time	an existing procedure sequence in a Visum travel demand model	added group of procedure parameters to set the needed attribute values for handling automated travel time with a different perception	Extensible Markup Language (XML)
create formula matrices for extended skim calculations and formula demand matrices for CV and AV	Visum travel demand model with an appropriate set up regarding attributes and names for transport systems	specific formula skim and demand matrices on zone level for calculations related to perceived travel time	Visual Basic Script (VBS)

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

4 Ridematching

4.1 Relevance

State of the art travel demand models for urban areas typically distinguish four or five main modes: walking, cycling, public transport and car. The mode car can be further split into car-driver and car-passenger. As the importance of ridesharing may increase in the coming years, ridesharing should be addressed as an additional sub or main mode in travel demand modelling. This requires an algorithm for matching the trips of suppliers (today typically drivers of conventional vehicles, in the future mobility-as-a-service providers) and demanders (travellers). Therefore a matching algorithm is necessary, which can be integrated in existing travel demand models.

4.2 Tool characteristics

The tool for matching ridesharing trips comprises several script files that are not discussed in detail. Table 3 describes input and output. The content of the corresponding files is provided in Chapter 6.3 as code.

A demo version of an example is available for download on the following website hosted by the University of Stuttgart: https://www.isv.uni-stuttgart.de/en/vuv/tools/index.html

Table 3: Summary of characteristics of the ridematching tool

Short description	Input	Output	Implementation
Matching trips for ridesharing	travel demand discretised into time intervals, e.g. 96x15min	one matrix with matched ridesharing trips for each time interval	Visual Basic Scripts (VBS)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

5 Vehicle scheduling

5.1 Relevance

As soon as automated vehicles are able to operate fully automated, there is no need for a driver anymore. Mobility-as-a-service (MaaS) providers will benefit from this, because labour costs for transporting passengers and relocating vehicles are reduced significantly. This will influence the operation of vehicles and the vehicle traffic in the road network: fewer cars can transport more people, but empty vehicle trips will increase traffic. Estimating the number of required cars and the amount of empty trips requires a model extension which replicates the operation of MaaS. The model extension computes schedules for fleets of MaaS vehicles by minimizing the fleet size and determining the required empty runs between drop-off and pick-up locations. As result the model extension provides the required fleet size and the number of empty vehicle trips for a predicted or given demand.

5.2 Tool characteristics

Table 4 describes the basic characteristics of the tool for vehicle scheduling. The development of the tool is not completed and ready to be applied for any travel demand model, therefore the related code is not provided in the appendix.

Table 4: Characteristics of the vehicle scheduling application

Short description	Input	Output	Implementation
concatenates vehicle trips according to initial user settings	demand matrix for each time interval containing vehicle trips (carsharing or ridesharing) matrix with temporal distance between zones as time intervals	number of required vehicles vehicle trip matrix for empty vehicle trips, one matrix for each time interval	Executable (EXE)

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

6 Appendix

Disclaimer: This document has been updated to provide the tools for Visum 2020 users as well. Therefore, the code for scripts, procedure parameters and volume-delay functions for Visum 2020 can be found in the respective subchapters of 6.1 and 6.2.

6.1 Tool: Volume-Delay Functions

6.1.1 Script file: User-defined attributes

The submitted file “CoEXist_Create_User-Defined_Attributes_-_Extension_for_handling_AV_in_volume-delay_functions.vbs” contains the following code:

Visum 18 or lower

```
'*****************************************************
' This script creates user-defined attributes for different network elements in Visum
' CoEXist - WP2 Macroscopic Modelling Tool
' USTUTT - University of Stuttgart
' October 2018
'*****************************************************

' One call for creating each user-defined attribute (UDA)
' AddUDA is defined below
Call Add_UDA("Links",       "CX_AV-READY",      "CX_AV-READY",      1,  0,  0,  1,      0,  "", false)
Call Add_UDA("Linktypes",   "CX_AV-READY",      "CX_AV-READY",      1,  0,  0,  1,      0,  "", false)
Call Add_UDA("Linktypes",   "CX_F_PCU_AV_A",    "CX_F_PCU_AV_A",    2,  2,  0,  3,      1,  "", false)
Call Add_UDA("Linktypes",   "CX_F_PCU_AV_B",    "CX_F_PCU_AV_B",    2,  2,  0,  3,      1,  "", false)
Call Add_UDA("Net",         "CX_AV-SHARE",      "CX_AV-SHARE",      1,  0,  0,  100,    0,  "", false)
Call Add_UDA("Matrices",    "CX_ID",            "CX_ID",            5,  0,  0,  0,      "", "", false)

' One call for adding a comment to each user-defined attribute
' SetUDAComment is defined below
Call Set_UDA_Comment("Links",       "CX_AV-READY",      "0: link is not AV-ready, 1: link is AV-ready")
Call Set_UDA_Comment("Linktypes",   "CX_AV-READY",      "0: link type is not AV-ready, 1: link type is AV-ready")
Call Set_UDA_Comment("Linktypes",   "CX_F_PCU_AV_A",    "PCU factor A for AV, which can be used stand-alone")
Call Set_UDA_Comment("Linktypes",   "CX_F_PCU_AV_B",    "PCU factor B for AV, which can be used additionally to factor A for a varying
resulting PCU factor depending on the AV share")
Call Set_UDA_Comment("Net",         "CX_AV-SHARE",      "Fixed AV share for splitting the demand as percentage")
Call Set_UDA_Comment("Matrices",    "CX_ID",            "CoEXist-unique identifier for working with formula matrices")

'*****************************************************
' Commonly used ValueTypes:
' Member                    Value   Summary
' ValueType_Int             1       Integer value (int)
' ValueType_Real            2       Real value (real)
' ValueType_String          5       String value (char*)
' ValueType_Duration        6       Duration (seconds or minutes depending on time format option)
' ValueType_TimePoint       7       Time stamp in seconds
' ValueType_Bool            9       Boolean value (true / false)
' ValueType_LongDuration    165     Precise duration (seconds or minutes depending on time format option)
'*****************************************************


'*****************************************************
' Definitions of subs and functions below
'*****************************************************

' Creates a user-defined attribute as specified above
Sub Add_UDA(ObjId, UDA_Code, UDA_Name, ValueType, Decplaces, MinVal, MaxVal, DefVal, Formula, canBeEmpty)
    If UDA_Name= "" then UDA_Name=UDA_Code
    On Error Resume Next
    Set VisObjects = GetVisObj(ObjId)
    VisObjects.AddUserDefinedAttribute UDA_Code, UDA_Code, UDA_Name, ValueType, Decplaces, , MinVal, MaxVal, DefVal, , , Formula, canBeEmpty
End Sub

' Sets a comment for a user-defined attribute as specified above
Sub Set_UDA_Comment(ObjId, UDA_Code, UDA_Comment)
    Set VisObjects = GetVisObj(ObjId)
    For Each Obj In VisObjects.Attributes.GetAll
    If Obj.Code = UDA_Code Then
        Obj.Comment = UDA_Comment
        Exit For
    End If
    Next
End Sub

' Gets a pointer to a Visum object class
```

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

```
Function GetVisObj(ObjID)
ObjID = LCase(ObjID)
    If ObjId = "net" Then
        Set VisObjects=Visum.Net
    ElseIf ObjId = "links" Then
        Set VisObjects=Visum.Net.Links
    ElseIf ObjId = "linktypes" Then
        Set VisObjects=Visum.Net.LinkTypes
    ElseIf ObjId = "matrices" Then
        Set VisObjects = Visum.Net.Matrices
    End If
Set GetVisObj=VisObjects
End Function
```

Visum 2020

```
'*****************************************************
' This script creates user-defined attributes for different network elements in Visum
' CoEXist - WP2 Macroscopic Modelling Tool
' USTUTT - University of Stuttgart
' February 2020
'*****************************************************

' One call for creating each user-defined attribute (UDA)
' AddUDA is defined below
Call Add_UDA("Links",    "CX_AV-READY",      "CX_AV-READY",      1,  0,  0,  1,      0,  "", false)
Call Add_UDA("Links",    "CX_F_PCU_AV_1",    "CX_F_PCU_AV_1",    2,  2,  0,  3,      1,  "", false)
Call Add_UDA("Links",    "CX_F_PCU_AV_0",    "CX_F_PCU_AV_0",    2,  2,  0,  3,      1,  "", false)
Call Add_UDA("Net",      "CX_AV-SHARE",      "CX_AV-SHARE",      1,  0,  0,  100,    0,  "", false)
Call Add_UDA("Matrices","CX_ID",             "CX_ID",            5,  0,  0,  0,      "", "", false)

' One call for adding a comment to each user-defined attribute
' SetUDAComment is defined below
Call Set_UDA_Comment("Links",    "CX_AV-READY",      "0: link is not AV-ready, 1: link is AV-ready")
Call Set_UDA_Comment("Links",    "CX_F_PCU_AV_1",    "PCU factor 1 for AV, which can be used stand-alone")
Call Set_UDA_Comment("Links",    "CX_F_PCU_AV_0",    "PCU factor 0 for AV, which can be used additionally to factor 1 for a varying resulting
PCU factor depending on the AV share")
Call Set_UDA_Comment("Net",      "CX_AV-SHARE",      "Fixed AV share for splitting the demand as percentage")
Call Set_UDA_Comment("Matrices","CX_ID",             "CoEXist-unique identifier for working with formula matrices")

'*****************************************************
' Commonly used ValueTypes:
' Member                    Value   Summary
' ValueType_Int             1       Integer value (int)
' ValueType_Real            2       Real value (real)
' ValueType_String          5       String value (char*)
' ValueType_Duration        6       Duration (seconds or minutes depending on time format option)
' ValueType_TimePoint       7       Time stamp in seconds
' ValueType_Bool            9       Boolean value (true / false)
' ValueType_LongDuration    165     Precise duration (seconds or minutes depending on time format option)
'*****************************************************

'*****************************************************
' Definitions of subs and functions below
'*****************************************************

' Creates a user-defined attribute as specified above
Sub Add_UDA(ObjId, UDA_Code, UDA_Name, ValueType, Decplaces, MinVal, MaxVal, DefVal, Formula, canBeEmpty)
    If UDA_Name= "" then UDA_Name=UDA_Code
    On Error Resume Next
    Set VisObjects = GetVisObj(ObjId)
    VisObjects.AddUserDefinedAttribute UDA_Code, UDA_Code, UDA_Name, ValueType, Decplaces, , MinVal, MaxVal, DefVal, , , Formula, ,
canBeEmpty
End Sub

' Sets a comment for a user-defined attribute as specified above
Sub Set_UDA_Comment(ObjId, UDA_Code, UDA_Comment)
    Set VisObjects = GetVisObj(ObjId)
    For Each Obj In VisObjects.Attributes.GetAll
    If Obj.Code = UDA_Code Then
        Obj.Comment = UDA_Comment
        Exit For
    End If
    Next
End Sub

' Gets a pointer to a Visum object class
Function GetVisObj(ObjID)
ObjID = LCase(ObjID)
    If ObjId = "net" Then
        Set VisObjects=Visum.Net
    ElseIf ObjId = "links" Then
        Set VisObjects=Visum.Net.Links
    ElseIf ObjId = "linktypes" Then
        Set VisObjects=Visum.Net.LinkTypes
    ElseIf ObjId = "matrices" Then
        Set VisObjects = Visum.Net.Matrices
    End If
Set GetVisObj=VisObjects
End Function
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

6.1.2 Script file: Formula matrices

The submitted file “CoEXist_Create_Formula_Matrices_-_Extension_for_handling_AV_in_volume-delay_functions.vbs” contains the following code:

Visum 18 or lower

```
'*****************************************************
' This script creates formula matrices in Visum
' CoEXist - WP2 Macroscopic Modelling Tool
' USTUTT - University of Stuttgart
' October 2018
'*****************************************************

' One call for creating each formula matrix
Call AddFormulaMat(-1, "CX_CV_DEMAND",              "", 2,  3,  "Matrix([CX_ID] = "+Chr(34)+"CX_CAR_DEMAND"+Chr(34)+")*(1-[CX_AV-
SHARE]/100)")

Call AddFormulaMat(-1, "CX_AV_DEMAND",              "", 2,  3,  "Matrix([CX_ID] = "+Chr(34)+"CX_CAR_DEMAND"+Chr(34)+")*[CX_AV-SHARE]/100")

'*****************************************************
' Commonly used values for MatrixType and ObjectTypeRef
' Member                    Value   Summary
' MATRIXTYPE_ANY            2       Any matrix type
' MATRIXTYPE_DEMAND         3       Demand matrix
' MATRIXTYPE_SKIM           4       Skim matrix
' OBJECTTYPEREF_ZONE        2       Zones
' OBJECTTYPEREF_MAINZONE    3       Main zones
' OBJECTTYPEREF_STOPAREA    4       Stop areas
'*****************************************************

' Creates a formula matrix on zone level
Function AddFormulaMat(MatNo, Code, Name, ObjectTypeRef, Matrixtype, Formula)
    If Name="" Then Name=Code
    On Error Resume Next
    Set x = Visum.Net.AddMatrixWithFormula (MatNo, Formula, ObjectTypeRef, Matrixtype)
        x.attvalue("Code")  = Code
        x.attvalue("Name")  = Name
        x.attvalue("CX_ID") = Code
End Function
```

Visum 2020

```
'*****************************************************
' This script creates formula matrices in Visum
' CoEXist - WP2 Macroscopic Modelling Tool
' USTUTT - University of Stuttgart
' February 2020
'*****************************************************

' One call for creating each formula matrix
Call AddFormulaMat(-1, "CX_CV_DEMAND",  "", 2,  3,  "Matrix([CX_ID] = "+Chr(34)+"CX_CAR_DEMAND"+Chr(34)+")*(1-[CX_AV-SHARE]/100)")

Call AddFormulaMat(-1, "CX_AV_DEMAND",  "", 2,  3,  "Matrix([CX_ID] = "+Chr(34)+"CX_CAR_DEMAND"+Chr(34)+")*[CX_AV-SHARE]/100")

'*****************************************************
' Commonly used values for MatrixType and ObjectTypeRef
' Member                    Value   Summary
' MATRIXTYPE_ANY            2       Any matrix type
' MATRIXTYPE_DEMAND         3       Demand matrix
' MATRIXTYPE_SKIM           4       Skim matrix
' OBJECTTYPEREF_ZONE        2       Zones
' OBJECTTYPEREF_MAINZONE    3       Main zones
' OBJECTTYPEREF_STOPAREA    4       Stop areas
'*****************************************************

' Creates a formula matrix on zone level
Function AddFormulaMat(MatNo, Code, Name, ObjectTypeRef, Matrixtype, Formula)
    If Name="" Then Name=Code
    On Error Resume Next
    Set x = Visum.Net.AddMatrixWithFormula (MatNo, Formula, ObjectTypeRef, Matrixtype)
        x.attvalue("Code")  = Code
        x.attvalue("Name")  = Name
        x.attvalue("CX_ID") = Code
End Function
```

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

6.1.3 Procedure sequence

The submitted file “CoEXist_Procedure_Parameters_-_Extension_for_handling_AV_in_volume-delay_functions.xml” extends an existing procedure sequence using the following code:

Visum 18 or lower

```
<?xml  version = "1.0"  encoding = "UTF-8"?>
<PROCEDURES  VERSION = "1705">
  <OPERATIONS>
    <OPERATION
      NO = "1"
      CODE = ""
      OPERATIONTYPE = "Group"
      ACTIVE = "1"
      COMMENT = "Set AV-related attributes regarding capacity/performance"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <GROUPPARA        ISEXPANDED = "1"      />
    </OPERATION>
    <OPERATION
      NO = "2"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "User input: AV share [percentage]"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "NETWORK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "CX_AV-SHARE"
        ONLYACTIVE = "0"
        FORMULA = "50"
      />
    </OPERATION>
    <OPERATION
      NO = "3"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "Transfer AV-readiness [0/1] from link type level to uval1"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "LINK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "ADDVAL1"
        ONLYACTIVE = "0"
        FORMULA = "[LINKTYPE\CX_AV-READY]"
      />
    </OPERATION>
    <OPERATION
      NO = "4"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "0"
      COMMENT = "Transfer AV-readiness [0/1] from link level to uval1"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
```

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

```
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "LINK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "ADDVAL1"
        ONLYACTIVE = "0"
        FORMULA = "[CX_AV-READY]"
      />
    </OPERATION>
    <OPERATION
      NO = "5"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "Transfer PCU-factor A for AV from link type level to uval2"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "LINK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "ADDVAL2"
        ONLYACTIVE = "0"
        FORMULA = "100*[LINKTYPE\CX_F_PCU_AV_A]"
      />
    </OPERATION>
    <OPERATION
      NO = "6"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "0"
      COMMENT = "Transfer PCU-factor B for AV from link type level to uval3"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "LINK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "ADDVAL3"
        ONLYACTIVE = "0"
        FORMULA = "100*[LINKTYPE\CX_F_PCU_AV_B]"
      />
    </OPERATION>
  </OPERATIONS>
</PROCEDURES>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Visum 2020

```
<?xml  version = "1.0"  encoding = "UTF-8"?>
<PROCEDURES  VERSION = "1902">
  <OPERATIONS>
    <OPERATION
      NO = "1"
      CODE = ""
      OPERATIONTYPE = "Group"
      ACTIVE = "1"
      COMMENT = "Set AV-related attributes regarding capacity/performance"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODES = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = ""
      OPERATIONVARIABLECOUNT = "">
      <GROUPPARA        ISEXPANDED = "1"      />
    </OPERATION>
    <OPERATION
      NO = "2"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "User input: AV share [percentage: 50 = 50%]"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODES = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = ""
      OPERATIONVARIABLECOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "NETWORK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "CX_AV-SHARE"
        ONLYACTIVE = "0"
        FORMULA = "50"
      />
    </OPERATION>
  </OPERATIONS>
</PROCEDURES>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

6.1.4 Dynamic Link Libraries: User-defined volume-delay functions

The DLL files are the result of compiled CPP files. Those contain the code provided on the following pages.

Visum 18 or lower

VisumVDF_CX_AV_PCU_CONST_BPR_x64.dll

```
// #include "..."
// => file needs to be in the project's directory
#include "UserDefinedVDF.h"

// #include <...>
// => file is searched in the project's environment folder
#include <tchar.h>
#include <math.h>

// VDF_Name appears as an entry in the dropdown list of volume-delay function types
wchar_t VDFName[] = _T("CX_AV_PCU_CONST_BPR");

// VDFID is the internal name in the version file
char  VDFID[] = "CX_AV_PCU_CONST_BPR";

// indexes of the respective TSys
int ind_P;
int ind_CX_AV;
int ind_LkwS_BV;
int ind_LkwS_DV;
int ind_LkwS_RV;
int ind_Lkw_BV;
int ind_Lkw_DV;
int ind_Lkw_RV;

int INTERFACE_VERSION = 1;

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

char Init()
{
	// make sure that no indexes can be incorrectly assigned
	ind_P		= -1;
	ind_CX_AV  = -1;
	ind_LkwS_BV = -1;
	ind_LkwS_DV = -1;
	ind_LkwS_RV = -1;
	ind_Lkw_BV = -1;
	ind_Lkw_DV = -1;
	ind_Lkw_RV = -1;

	return TRUE;
}

void Destroy()
{
}

char IsThreadSafe()
{
	// TRUE = function may be called multiple times in parallel
	// good for multithreaded assignment procedures
	return TRUE;
}

char DependsOnTSys()
{
	// TRUE = vehicle volumes by TSys may be used in Calc-Function
	return TRUE;
}

const wchar_t* GetName(const char *langid)
{
	// setting VDF_Name as the name for this volume-delay function type
	return VDFName;
}

const char* GetID()
{
	return VDFID;
}
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
int GetInterfaceVersion()
{
	return INTERFACE_VERSION;
}

void SetTsysInfo(int numtsys, const wchar_t * tsysids[])
{
	// assign TSys Code to the related index within the TSys array
	// will be done once at the beginning of an assignment by Visum
	int i;
	// go through all the positions in the array
	for (i = 0; i < numtsys; i++)
	{
		// wcscmp executes string comparison
		// if return value == 0, strings are identical
		if (wcscmp(tsysids[i], _T("P")) == 0) {
			ind_P = i;
			continue; // exit for-loop and begin with next iteration
		}
		if (wcscmp(tsysids[i], _T("CX_AV")) == 0) {
			ind_CX_AV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_BV")) == 0) {
			ind_LkwS_BV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_DV")) == 0) {
			ind_LkwS_DV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_RV")) == 0) {
			ind_LkwS_RV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_BV")) == 0) {
			ind_Lkw_BV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_DV")) == 0) {
			ind_Lkw_DV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_RV")) == 0)
			ind_Lkw_RV = i;
	}
}

// specify calculation rule below
// transferred parameters must remain the same even if they are not used
double Calc(int tsysind, char tsysisopen,
	int typ, int numlanes, double length, double cap, double v0, double t0, double gradient,
	double pcuvol, double basevol, double vehvolsys[],
	int uval1, int uval2, int uval3, int uvaltsys,
	double para_a, double para_b, double para_c, double para_d, double para_f, double para_a2, double para_b2, double para_d2, double para_f2,
double satcrit)
{
	// declaration
	double truck_vol;
	double pcu_factor;
	double userdef_pcuvol;
	// sat = saturation is not assigend by default
	double sat;

	// captures negative or zero values of capacity and returns huge TTC
	if (cap <= 0 || para_c <= 0)
		return 1E10;

	// add up all truck volumes
	truck_vol = vehvolsys[ind_LkwS_BV] + vehvolsys[ind_LkwS_DV] + vehvolsys[ind_LkwS_RV] + vehvolsys[ind_Lkw_BV] + vehvolsys[ind_Lkw_DV] +
vehvolsys[ind_Lkw_RV];

	// if there are no vehicles on the link, TTC corresponds to T0
	// this is important for the skim matrix calculation of walking
	if (truck_vol + vehvolsys[ind_CX_AV] + vehvolsys[ind_P] == 0)
		return t0;

	// traffic volume in passenger car units
	// considers CX_AV-READY property through AddVal1=uval1 and PCU factor through AddVal2=uval2
	userdef_pcuvol = truck_vol*2.0 + vehvolsys[ind_P]*1.0 + vehvolsys[ind_CX_AV]*(1 - uval1) + vehvolsys[ind_CX_AV]*uval1*(uval2 / 100.0);

	// calculate saturation, para_c is factor for full-day capacity and is set in Visum
	sat = (userdef_pcuvol) / (cap * para_c);

	// return TTC (standard function as BPR)
	return t0 * (1 + para_a * (pow(sat, para_b)));
}
```

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No. 723201

Figure 2 shows the preview picture “VisumVDF_CX_AV_PCU_CONST_BPR_x64.bmp”, which is displayed in Visum, if the user selects the related volume-delay function.

$$t_{cur.}(sat) = t_0 \cdot (1 + a \cdot sat^b)$$

$$sat = \frac{q_{HDV} \cdot f_{HDV}^{PCU} + q_{LDV} \cdot f_{LDV}^{PCU} + q_{AV} \cdot f_{AV}^{PCU}}{q_{\max}}$$

if CX_AV-READY=1, $f_{AV}^{PCU} = AddVal2$, else $f_{AV}^{PCU} = f_{LDV}^{PCU}$

Figure 2: Preview picture with calculation rule

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

VisumVDF_CX_AV_PCU_CONST_LOHSE_x64.dll

```
// #include "..."
// => file needs to be in the project's directory
#include "UserDefinedVDF.h"

// #include <...>
// => file is searched in the project's environment folder
#include <tchar.h>
#include <math.h>

// VDF_Name appears as an entry in the dropdown list of volume-delay function types
wchar_t VDFName[] = _T("CX_AV_PCU_CONST_LOHSE");

// VDFID is the internal name in the version file
char  VDFID[] = "CX_AV_PCU_CONST_LOHSE";

// indexes of the respective TSys
int ind_P;
int ind_CX_AV;
int ind_LkwS_BV;
int ind_LkwS_DV;
int ind_LkwS_RV;
int ind_Lkw_BV;
int ind_Lkw_DV;
int ind_Lkw_RV;

int INTERFACE_VERSION = 1;

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

char Init()
{
	// make sure that no indexes can be incorrectly assigned
	ind_P = -1;
	ind_CX_AV = -1;
	ind_LkwS_BV = -1;
	ind_LkwS_DV = -1;
	ind_LkwS_RV = -1;
	ind_Lkw_BV = -1;
	ind_Lkw_DV = -1;
	ind_Lkw_RV = -1;

	return TRUE;
}

void Destroy()
{
}

char IsThreadSafe()
{
	// TRUE = function may be called multiple times in parallel
	// good for multithreaded assignment procedures
	return TRUE;
}

char DependsOnTSys()
{
	// TRUE = vehicle volumes by TSys may be used in Calc-Function
	return TRUE;
}

const wchar_t* GetName(const char *langid)
{
	// setting VDF_Name as the name for this volume-delay function type
	return VDFName;
}

const char* GetID()
{
	return VDFID;
}

int GetInterfaceVersion()
{
	return INTERFACE_VERSION;
}

void SetTsysInfo(int numtsys, const wchar_t * tsysids[])
{
	// assign TSys Code to the related index within the TSys array
	// will be done once at the beginning of an assignment by Visum
	int i;
	// go through all the positions in the array
	for (i = 0; i < numtsys; i++)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
	{
		// wcscmp executes string comparison
		// if return value == 0, strings are identical
		if (wcscmp(tsysids[i], _T("P")) == 0) {
			ind_P = i;
			continue; // exit for-loop and begin with next iteration
		}
		if (wcscmp(tsysids[i], _T("CX_AV")) == 0) {
			ind_CX_AV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_BV")) == 0) {
			ind_LkwS_BV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_DV")) == 0) {
			ind_LkwS_DV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_RV")) == 0) {
			ind_LkwS_RV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_BV")) == 0) {
			ind_Lkw_BV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_DV")) == 0) {
			ind_Lkw_DV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_RV")) == 0)
			ind_Lkw_RV = i;
	}
}

// specify calculation rule below
// transferred parameters must remain the same even if they are not used
double Calc(int tsysind, char tsysisopen,
	int typ, int numlanes, double length, double cap, double v0, double t0, double gradient,
	double pcuvol, double basevol, double vehvolsys[],
	int uval1, int uval2, int uval3, int uvaltsys,
	double para_a, double para_b, double para_c, double para_d, double para_f, double para_a2, double para_b2, double para_d2, double para_f2,
double satcrit)
{
	// declaration
	double truck_vol;
	double pcu_factor;
	double userdef_pcuvol;
	// sat = saturation is not assigend by default
	double sat;

	// captures negative or zero values of capacity and returns huge TTC
	if (cap <= 0 || para_c <= 0)
		return 1E10;

	// add up all truck volumes
	truck_vol = vehvolsys[ind_LkwS_BV] + vehvolsys[ind_LkwS_DV] + vehvolsys[ind_LkwS_RV] + vehvolsys[ind_Lkw_BV] + vehvolsys[ind_Lkw_DV] +
vehvolsys[ind_Lkw_RV];

	// if there are no vehicles on the link, TTC corresponds to T0
	// this is important for the skim matrix calculation of walking
	if (truck_vol + vehvolsys[ind_CX_AV] + vehvolsys[ind_P] == 0)
		return t0;

	// traffic volume in passenger car units
	// considers CX_AV-READY property through AddVal1=uval1 and PCU factor through AddVal2=uval2
	userdef_pcuvol = truck_vol * 2.0 + vehvolsys[ind_P] * 1.0 + vehvolsys[ind_CX_AV] * (1 - uval1) + vehvolsys[ind_CX_AV] * uval1*(uval2 / 100.0);

	// calculate saturation, para_c is factor for full-day capacity and is set in Visum
	sat = (userdef_pcuvol) / (cap * para_c);

	// return TTC (standard function as LOHSE)
	if (sat <= satcrit)
		return t0 * (1 + para_a * (pow(sat, para_b)));
	else // sat > satcrit
		return t0 * (1 + para_a * (pow(satcrit, para_b))) + para_a * para_b * t0 * (sat - satcrit) * (pow(satcrit, para_b - 1));
}
```

Figure 3 shows the preview picture “VisumVDF_CX_AV_PCU_CONST_LOHSE_x64.bmp”, which is displayed in Visum, if the user selects the related volume-delay function.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

$$t_{cur}(sat) = \begin{cases} t_0(1 + a\,sat^b) & sat \le sat_{crit} \\ t_0(1 + a\,sat_{crit}^{\ b}) + ab\,t_0(sat_{crit})^{b-1}(sat - sat_{crit}) & sat > sat_{crit} \end{cases}$$

$$sat = \frac{q_{HDV} \cdot f_{HDV}^{PCU} + q_{LDV} \cdot f_{LDV}^{PCU} + q_{AV} \cdot f_{AV}^{PCU}}{q_{\max}}$$

if CX_AV-READY=1, $f_{AV}^{PCU} = AddVal2$, else $f_{AV}^{PCU} = f_{LDV}^{PCU}$

Figure 3: Preview picture with calculation rule

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

VisumVDF_CX_AV_PCU_VAR_BPR_x64.dll

```
// #include "..."
// => file needs to be in the project's directory
#include "UserDefinedVDF.h"

// #include <...>
// => file is searched in the project's environment folder
#include <tchar.h>
#include <math.h>

// VDF_Name appears as an entry in the dropdown list of volume-delay function types
wchar_t VDFName[] = _T("CX_AV_PCU_VAR_BPR");

// VDFID is the internal name in the version file
char  VDFID[] = "CX_AV_PCU_VAR_BPR";

// indexes of the respective TSys
int ind_P;
int ind_CX_AV;
int ind_LkwS_BV;
int ind_LkwS_DV;
int ind_LkwS_RV;
int ind_Lkw_BV;
int ind_Lkw_DV;
int ind_Lkw_RV;

int INTERFACE_VERSION = 1;

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

char Init()
{
	// make sure that no indexes can be incorrectly assigned
	ind_P = -1;
	ind_CX_AV = -1;
	ind_LkwS_BV = -1;
	ind_LkwS_DV = -1;
	ind_LkwS_RV = -1;
	ind_Lkw_BV = -1;
	ind_Lkw_DV = -1;
	ind_Lkw_RV = -1;

	return TRUE;
}

void Destroy()
{
}

char IsThreadSafe()
{
	// TRUE = function may be called multiple times in parallel
	// good for multithreaded assignment procedures
	return TRUE;
}

char DependsOnTSys()
{
	// TRUE = vehicle volumes by TSys may be used in Calc-Function
	return TRUE;
}

const wchar_t* GetName(const char *langid)
{
	// setting VDF_Name as the name for this volume-delay function type
	return VDFName;
}

const char* GetID()
{
	return VDFID;
}

int GetInterfaceVersion()
{
	return INTERFACE_VERSION;
}

void SetTsysInfo(int numtsys, const wchar_t * tsysids[])
{
	// assign TSys Code to the related index within the TSys array
	// will be done once at the beginning of an assignment by Visum
	int i;
	// go through all the positions in the array
	for (i = 0; i < numtsys; i++)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
	{
		// wcscmp executes string comparison
		// if return value == 0, strings are identical
		if (wcscmp(tsysids[i], _T("P")) == 0) {
			ind_P = i;
			continue; // exit for-loop and begin with next iteration
		}
		if (wcscmp(tsysids[i], _T("CX_AV")) == 0) {
			ind_CX_AV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_BV")) == 0) {
			ind_LkwS_BV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_DV")) == 0) {
			ind_LkwS_DV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_RV")) == 0) {
			ind_LkwS_RV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_BV")) == 0) {
			ind_Lkw_BV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_DV")) == 0) {
			ind_Lkw_DV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_RV")) == 0)
			ind_Lkw_RV = i;
	}
}

// specify calculation rule below
// transferred parameters must remain the same even if they are not used
double Calc(int tsysind, char tsysisopen,
	int typ, int numlanes, double length, double cap, double v0, double t0, double gradient,
	double pcuvol, double basevol, double vehvolsys[],
	int uval1, int uval2, int uval3, int uvaltsys,
	double para_a, double para_b, double para_c, double para_d, double para_f, double para_a2, double para_b2, double para_d2, double para_f2,
double satcrit)
{
	// declaration
	double truck_vol;
	double pcu_factor;
	double userdef_pcuvol;
	// sat = saturation is not assigend by default
	double sat;

	// captures negative or zero values of capacity and returns huge TTC
	if (cap <= 0 || para_c <= 0)
		return 1E10;

	// add up all truck volumes
	truck_vol = vehvolsys[ind_LkwS_BV] + vehvolsys[ind_LkwS_DV] + vehvolsys[ind_LkwS_RV] + vehvolsys[ind_Lkw_BV] + vehvolsys[ind_Lkw_DV] +
vehvolsys[ind_Lkw_RV];

	// if there are no vehicles on the link, TTC corresponds to T0
	// this is important for the skim matrix calculation of walking
	if (truck_vol + vehvolsys[ind_CX_AV] + vehvolsys[ind_P] == 0)
		return t0;

	// calculate pcu factor dependend on AV share, for each link individually
	// the AV share on a link does not correspond to the global AV share
	// please note, that uvals are integer values, so division by 100 is necessary
	pcu_factor = (uval3 - (vehvolsys[ind_CX_AV] / (truck_vol + vehvolsys[ind_CX_AV] + vehvolsys[ind_P]))*(uval3 - uval2)) / 100.0;

	// traffic volume in passenger car units
	// considers CX_AV-READY property through AddVal1=uval1 and PCU factor through AddVal2=uval2
	userdef_pcuvol = truck_vol * 2.0 + vehvolsys[ind_P] * 1.0 + vehvolsys[ind_CX_AV] * (1 - uval1) + vehvolsys[ind_CX_AV] * uval1*pcu_factor;

	// calculate saturation, para_c is factor for full-day capacity and is set in Visum
	sat = (userdef_pcuvol) / (cap * para_c);

	// return TTC (standard function as BPR)
	return t0 * (1 + para_a * (pow(sat, para_b)));
}
```

Figure 4 shows the preview picture “VisumVDF_CX_AV_PCU_VAR_BPR_x64.bmp”, which is displayed in Visum, if the user selects the related volume-delay function.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

$$t_{car}(sat) = t_0 \cdot (1 + a \cdot sat^b) \qquad sat = \frac{q_{HDV} \cdot f_{HDV}^{PCU} + q_{LDV} \cdot f_{LDV}^{PCU} + q_{AV} \cdot f_{AV}^{PCU}(p_{AV})}{q_{\max}}$$

$$f_{AV}^{PCU}(p_{AV}) = f_{AV}^{PCU,0\%} - p_{AV} \cdot \left(f_{AV}^{PCU,0\%} - f_{AV}^{PCU,100\%} \right)$$

if CX_AV-READY=1, $f_{AV}^{PCU,100\%} = AddVal2$ and $f_{AV}^{PCU,0\%} = AddVal3$, else $f_{AV}^{PCU} = f_{LDV}^{PCU}$

Figure 4: Preview picture with calculation rule

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

VisumVDF_CX_AV_PCU_VAR_LOHSE_x64.dll

```
// #include "..."
// => file needs to be in the project's directory
#include "UserDefinedVDF.h"

// #include <...>
// => file is searched in the project's environment folder
#include <tchar.h>
#include <math.h>

// VDF_Name appears as an entry in the dropdown list of volume-delay function types
wchar_t VDFName[] = _T("CX_AV_PCU_VAR_LOHSE");

// VDFID is the internal name in the version file
char  VDFID[] = "CX_AV_PCU_VAR_LOHSE";

// indexes of the respective TSys
int ind_P;
int ind_CX_AV;
int ind_LkwS_BV;
int ind_LkwS_DV;
int ind_LkwS_RV;
int ind_Lkw_BV;
int ind_Lkw_DV;
int ind_Lkw_RV;

int INTERFACE_VERSION = 1;

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

char Init()
{
	// make sure that no indexes can be incorrectly assigned
	ind_P = -1;
	ind_CX_AV = -1;
	ind_LkwS_BV = -1;
	ind_LkwS_DV = -1;
	ind_LkwS_RV = -1;
	ind_Lkw_BV = -1;
	ind_Lkw_DV = -1;
	ind_Lkw_RV = -1;

	return TRUE;
}

void Destroy()
{
}

char IsThreadSafe()
{
	// TRUE = function may be called multiple times in parallel
	// good for multithreaded assignment procedures
	return TRUE;
}

char DependsOnTSys()
{
	// TRUE = vehicle volumes by TSys may be used in Calc-Function
	return TRUE;
}

const wchar_t* GetName(const char *langid)
{
	// setting VDF_Name as the name for this volume-delay function type
	return VDFName;
}

const char* GetID()
{
	return VDFID;
}

int GetInterfaceVersion()
{
	return INTERFACE_VERSION;
}

void SetTsysInfo(int numtsys, const wchar_t * tsysids[])
{
	// assign TSys Code to the related index within the TSys array
	// will be done once at the beginning of an assignment by Visum
	int i;
	// go through all the positions in the array
	for (i = 0; i < numtsys; i++)
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
	{
		// wcscmp executes string comparison
		// if return value == 0, strings are identical
		if (wcscmp(tsysids[i], _T("P")) == 0) {
			ind_P = i;
			continue; // exit for-loop and begin with next iteration
		}
		if (wcscmp(tsysids[i], _T("CX_AV")) == 0) {
			ind_CX_AV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_BV")) == 0) {
			ind_LkwS_BV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_DV")) == 0) {
			ind_LkwS_DV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("LkwS_RV")) == 0) {
			ind_LkwS_RV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_BV")) == 0) {
			ind_Lkw_BV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_DV")) == 0) {
			ind_Lkw_DV = i;
			continue;
		}
		if (wcscmp(tsysids[i], _T("Lkw_RV")) == 0)
			ind_Lkw_RV = i;
	}
}

// specify calculation rule below
// transferred parameters must remain the same even if they are not used
double Calc(int tsysind, char tsysisopen,
	int typ, int numlanes, double length, double cap, double v0, double t0, double gradient,
	double pcuvol, double basevol, double vehvolsys[],
	int uval1, int uval2, int uval3, int uvaltsys,
	double para_a, double para_b, double para_c, double para_d, double para_f, double para_a2, double para_b2, double para_d2, double para_f2,
double satcrit)
{
	// declaration
	double truck_vol;
	double pcu_factor;
	double userdef_pcuvol;
	// sat = saturation is not assigend by default
	double sat;

	// captures negative or zero values of capacity and returns huge TTC
	if (cap <= 0 || para_c <= 0)
		return 1E10;

	// add up all truck volumes
	truck_vol = vehvolsys[ind_LkwS_BV] + vehvolsys[ind_LkwS_DV] + vehvolsys[ind_LkwS_RV] + vehvolsys[ind_Lkw_BV] + vehvolsys[ind_Lkw_DV] +
vehvolsys[ind_Lkw_RV];

	// if there are no vehicles on the link, TTC corresponds to T0
	// this is important for the skim matrix calculation of walking
	if (truck_vol + vehvolsys[ind_CX_AV] + vehvolsys[ind_P] == 0)
		return t0;

	// calculate pcu factor dependend on AV share, for each link individually
	// the AV share on a link does not correspond to the global AV share
	// please note, that uvals are integer values, so division by 100 is necessary
	pcu_factor = (uval3 - (vehvolsys[ind_CX_AV] / (truck_vol + vehvolsys[ind_CX_AV] + vehvolsys[ind_P]))*(uval3 - uval2))/100.0;

	// traffic volume in passenger car units
	// considers CX_AV-READY property through AddVal1=uval1 and PCU factor through AddVal2=uval2
	userdef_pcuvol = truck_vol * 2.0 + vehvolsys[ind_P] * 1.0 + vehvolsys[ind_CX_AV] * (1 - uval1) + vehvolsys[ind_CX_AV]*uval1*pcu_factor;

	// calculate saturation, para_c is factor for full-day capacity and is set in Visum
	sat = (userdef_pcuvol) / (cap * para_c);

	// return TTC (standard function as LOHSE)
	if (sat <= satcrit)
		return t0 * (1 + para_a * (pow(sat, para_b)));
	else // sat > satcrit
		return t0 * (1 + para_a * (pow(satcrit, para_b))) + para_a * para_b * t0 * (sat - satcrit) * (pow(satcrit, para_b - 1));
}
```

Figure 5 shows the preview picture “VisumVDF_CX_AV_PCU_VAR_LOHSE_x64.bmp”, which is displayed in Visum, if the user selects the related volume-delay function.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

$$t_{cur}(sat) = \begin{cases} t_0(1 + a\,sat^b) & sat \le sat_{crit} \\ t_0(1 + a\,sat_{crit}^{\ b}) + ab\,t_0(sat_{crit})^{b-1}(sat - sat_{crit}) & sat > sat_{crit} \end{cases}$$

$$sat = \frac{q_{HDV} \cdot f_{HDV}^{PCU} + q_{LDV} \cdot f_{LDV}^{PCU} + q_{AV} \cdot f_{AV}^{PCU}(p_{AV})}{q_{\max}}$$

$$f_{AV}^{PCU}(p_{AV}) = f_{AV}^{PCU,0\%} - p_{AV} \cdot \left(f_{AV}^{PCU,0\%} - f_{AV}^{PCU,100\%}\right)$$

if CX_AV-READY=1, $f_{AV}^{PCU,100\%} = AddVal2$ and $f_{AV}^{PCU,0\%} = AddVal3$, else $f_{AV}^{PCU} = f_{LDV}^{PCU}$

Figure 5: Preview picture with calculation rule

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Visum 2020

VisumVDF_CX_AV_PCU_CONST_BPR_2020_x64.dll

```
#include "UserDefinedVDF_2020.h"
#include "tchar.h"
#include <math.h>
#include <float.h>
#include "string.h"

// VDF_Name appears as an entry in the dropdown list of volume-delay function types
wchar_t VDFName[] = _T("CX_AV_PCU_CONST_BPR_2020");

// VDFID is the internal name in the version file
char VDFID[] = "CX_AV_PCU_CONST_BPR_2020";

int INTERFACE_VERSION = 1;

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

char Init()
{
    return TRUE;
}

enum AttributeIndices
{
    CX_AV_READY,
    PCU_AV,
    PCU_TSYS_P,
    PCU_TSYS_CX_AV,
    PCU_TSYS_LkwS_BV,
    PCU_TSYS_LkwS_DV,
    PCU_TSYS_LkwS_RV,
    PCU_TSYS_Lkw_BV,
    PCU_TSYS_Lkw_DV,
    PCU_TSYS_Lkw_RV,
    LastId
};

static const int MyMaxIDLength = 100; // Character limit for Strings
static wchar_t staticAttributeIDs[LastId][MyMaxIDLength] =
{
    // Strings of attributeIDs from links below must correspond to the order of AttributeIndices
    L"CX_AV-READY",
    L"CX_F_PCU_AV",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"P\")\\PCU",    // please note: \\ = \ and \" = "
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"CX_AV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_BV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_DV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_RV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_BV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_DV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_RV\")\\PCU",
};

int GetNumStaticAttributes()
{
    return LastId;
}

const wchar_t * GetStaticAttributeID(int attributesIndexZeroBased)
{
    return staticAttributeIDs[attributesIndexZeroBased];
}

void Destroy()
{
}

char IsThreadSafe()
{
    return TRUE;
}

char DependsOnTSys()
{
    return 2;
    // 2: VDF calculates the same value for all TSys and receives the the volumes per TSys
}

const wchar_t* GetName(const char *langid)
{
    return VDFName;
}

const char* GetID()
{
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
    return VDFID;
}

int GetInterfaceVersion()
{
    return INTERFACE_VERSION;
}

static int sTSysIndex_P = -1;
static int sTSysIndex_CX_AV = -1;
static int sTSysIndex_LkwS_BV = -1;
static int sTSysIndex_LkwS_DV = -1;
static int sTSysIndex_LkwS_RV = -1;
static int sTSysIndex_Lkw_BV = -1;
static int sTSysIndex_Lkw_DV = -1;
static int sTSysIndex_Lkw_RV = -1;

void SetTsysInfo(int numtsys, const wchar_t * tsysids[])
{
    // -1 means "TSys does not exist"
    sTSysIndex_P = -1;
    sTSysIndex_CX_AV = -1;
    sTSysIndex_LkwS_BV = -1;
    sTSysIndex_LkwS_DV = -1;
    sTSysIndex_LkwS_RV = -1;
    sTSysIndex_Lkw_BV = -1;
    sTSysIndex_Lkw_DV = -1;
    sTSysIndex_Lkw_RV = -1;

    // assign TSys Code to the related index within the TSys array
    // will be done once at the beginning of an assignment by Visum

    for (int tsysInd = 0; tsysInd < numtsys; ++tsysInd) {
        // wcscmp executes string comparison
        // if return value == 0, strings are identical
        if (wcscmp(tsysids[tsysInd], L"P") == 0) {
            sTSysIndex_P = tsysInd;
            continue; // exit for-loop and begin with next iteration
        }
        else if (wcscmp(tsysids[tsysInd], L"CX_AV") == 0) {
            sTSysIndex_CX_AV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_BV") == 0) {
            sTSysIndex_LkwS_BV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_DV") == 0) {
            sTSysIndex_LkwS_DV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_RV") == 0) {
            sTSysIndex_LkwS_RV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_BV") == 0) {
            sTSysIndex_Lkw_BV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_DV") == 0) {
            sTSysIndex_Lkw_DV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_RV") == 0) {
            sTSysIndex_Lkw_RV = tsysInd;
            continue;
        }
    }
}

// specify calculation rule with improved user-defined VDF
double CalculateWithStaticAttributes(int tsysind, char tsysisopen, double cap, double t0,
    double pcuvol, double basevol, double vehvolsys[],
    double staticAttributeValues[],
    double para_a, double para_b, double para_c, double para_d, double para_f,
    double para_a2, double para_b2, double para_d2, double para_f2, double satcrit)
{

    double const av_ready = staticAttributeValues[CX_AV_READY];
    // get default PCU values
    double const pcu_P = staticAttributeValues[PCU_TSYS_P];
    double const pcu_LkwS_BV = staticAttributeValues[PCU_TSYS_LkwS_BV];
    double const pcu_LkwS_DV = staticAttributeValues[PCU_TSYS_LkwS_DV];
    double const pcu_LkwS_RV = staticAttributeValues[PCU_TSYS_LkwS_RV];
    double const pcu_Lkw_BV = staticAttributeValues[PCU_TSYS_Lkw_BV];
    double const pcu_Lkw_DV = staticAttributeValues[PCU_TSYS_Lkw_DV];
    double const pcu_Lkw_RV = staticAttributeValues[PCU_TSYS_Lkw_RV];

    // get possible PCU values for AV
    double pcu_CX_AV_used;
    double const pcu_CX_AV_default = staticAttributeValues[PCU_TSYS_CX_AV];
    double const pcu_CX_AV = staticAttributeValues[PCU_AV]; // PCU factor for each AV

    if (cap <= 0 || para_c <= 0) {
        return DBL_MAX;
    }
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
    // add up vehicle numbers separately for CV and AV
    double vehvol_all_CV = vehvolsys[sTSysIndex_P] + vehvolsys[sTSysIndex_LkwS_BV] + vehvolsys[sTSysIndex_LkwS_DV] +
vehvolsys[sTSysIndex_LkwS_RV] + vehvolsys[sTSysIndex_Lkw_BV] + vehvolsys[sTSysIndex_Lkw_DV] + vehvolsys[sTSysIndex_Lkw_RV];
    double vehvol_all_AV = vehvolsys[sTSysIndex_CX_AV];

    // if there are no vehicles on the link, TTC corresponds to T0
    if (vehvol_all_CV + vehvol_all_AV == 0)
        return t0;

    // if link is not AV-ready, use TSys-default PCU factor for AV
    if (av_ready == 0)
        pcu_CX_AV_used = pcu_CX_AV_default;
    else // if link is AV-ready, used PCU factor depends on AV share
        pcu_CX_AV_used = pcu_CX_AV;

    // mutliply respective PCU factor with vehicle volume
    double pcuvol_CX_AV = pcu_CX_AV_used * vehvolsys[sTSysIndex_CX_AV];
    double pcuvol_P = pcu_P * vehvolsys[sTSysIndex_P];
    double pcuvol_LkwS_BV = pcu_LkwS_BV * vehvolsys[sTSysIndex_LkwS_BV];
    double pcuvol_LkwS_DV = pcu_LkwS_DV * vehvolsys[sTSysIndex_LkwS_DV];
    double pcuvol_LkwS_RV = pcu_LkwS_RV * vehvolsys[sTSysIndex_LkwS_RV];
    double pcuvol_Lkw_BV = pcu_Lkw_BV * vehvolsys[sTSysIndex_Lkw_BV];
    double pcuvol_Lkw_DV = pcu_Lkw_DV * vehvolsys[sTSysIndex_Lkw_DV];
    double pcuvol_Lkw_RV = pcu_Lkw_RV * vehvolsys[sTSysIndex_Lkw_RV];

    double const totalvol_pcu = pcuvol_P + pcuvol_CX_AV + pcuvol_LkwS_BV + pcuvol_LkwS_DV + pcuvol_LkwS_RV + pcuvol_Lkw_BV + pcuvol_Lkw_DV +
pcuvol_Lkw_RV;

    double const sat = totalvol_pcu / (cap * para_c);

    // return TTC (standard function as BPR)
    return t0 * (1 + para_a * (pow(sat, para_b)));
}
```

Figure 6 shows the preview picture “VisumVDF_CX_AV_PCU_CONST_BPR_2020_x64.bmp”, which is displayed in Visum, if the user selects the related volume-delay function.

$$t_{cur}(sat) = t_0 \cdot (1 + a \cdot sat^b)$$

$$sat = \frac{q_{HDV} \cdot f_{HDV}^{PCU} + q_{LDV} \cdot f_{LDV}^{PCU} + q_{AV} \cdot f_{AV}^{PCU}}{q_{\max}}$$

if CX_AV-READY = 1, $f_{AV}^{PCU} = CX\_F\_PCU\_AV\_1$

Figure 6: Preview picture with calculation rule

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

VisumVDF_CX_AV_PCU_CONST_LOHSE_2020_x64.dll

```
#include "UserDefinedVDF_2020.h"
#include "tchar.h"
#include <math.h>
#include <float.h>
#include "string.h"

// VDF_Name appears as an entry in the dropdown list of volume-delay function types
wchar_t VDFName[] = _T("CX_AV_PCU_CONST_LOHSE_2020");

// VDFID is the internal name in the version file
char VDFID[] = "CX_AV_PCU_CONST_LOHSE_2020";

int INTERFACE_VERSION = 1;

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

char Init()
{
    return TRUE;
}

enum AttributeIndices
{
    CX_AV_READY,
    PCU_AV,
    PCU_TSYS_P,
    PCU_TSYS_CX_AV,
    PCU_TSYS_LkwS_BV,
    PCU_TSYS_LkwS_DV,
    PCU_TSYS_LkwS_RV,
    PCU_TSYS_Lkw_BV,
    PCU_TSYS_Lkw_DV,
    PCU_TSYS_Lkw_RV,
    LastId
};

static const int MyMaxIDLength = 100; // Character limit for Strings
static wchar_t staticAttributeIDs[LastId][MyMaxIDLength] =
{
    // Strings of attributeIDs from links below must correspond to the order of AttributeIndices
    L"CX_AV-READY",
    L"CX_F_PCU_AV",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"P\")\\PCU",    // please note: \\ = \ and \" = "
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"CX_AV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_BV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_DV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_RV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_BV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_DV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_RV\")\\PCU",
};

int GetNumStaticAttributes()
{
    return LastId;
}

const wchar_t * GetStaticAttributeID(int attributesIndexZeroBased)
{
    return staticAttributeIDs[attributesIndexZeroBased];
}

void Destroy()
{
}

char IsThreadSafe()
{
    return TRUE;
}

char DependsOnTSys()
{
    return 2;
    // 2: VDF calculates the same value for all TSys and receives the the volumes per TSys
}

const wchar_t* GetName(const char *langid)
{
    return VDFName;
}

const char* GetID()
{
    return VDFID;
}

int GetInterfaceVersion()
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
{
    return INTERFACE_VERSION;
}

static int sTSysIndex_P = -1;
static int sTSysIndex_CX_AV = -1;
static int sTSysIndex_LkwS_BV = -1;
static int sTSysIndex_LkwS_DV = -1;
static int sTSysIndex_LkwS_RV = -1;
static int sTSysIndex_Lkw_BV = -1;
static int sTSysIndex_Lkw_DV = -1;
static int sTSysIndex_Lkw_RV = -1;

void SetTsysInfo(int numtsys, const wchar_t * tsysids[])
{
    // -1 means "TSys does not exist"
    sTSysIndex_P = -1;
    sTSysIndex_CX_AV = -1;
    sTSysIndex_LkwS_BV = -1;
    sTSysIndex_LkwS_DV = -1;
    sTSysIndex_LkwS_RV = -1;
    sTSysIndex_Lkw_BV = -1;
    sTSysIndex_Lkw_DV = -1;
    sTSysIndex_Lkw_RV = -1;

    // assign TSys Code to the related index within the TSys array
    // will be done once at the beginning of an assignment by Visum

    for (int tsysInd = 0; tsysInd < numtsys; ++tsysInd) {
        // wcscmp executes string comparison
        // if return value == 0, strings are identical
        if (wcscmp(tsysids[tsysInd], L"P") == 0) {
            sTSysIndex_P = tsysInd;
            continue; // exit for-loop and begin with next iteration
        }
        else if (wcscmp(tsysids[tsysInd], L"CX_AV") == 0) {
            sTSysIndex_CX_AV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_BV") == 0) {
            sTSysIndex_LkwS_BV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_DV") == 0) {
            sTSysIndex_LkwS_DV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_RV") == 0) {
            sTSysIndex_LkwS_RV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_BV") == 0) {
            sTSysIndex_Lkw_BV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_DV") == 0) {
            sTSysIndex_Lkw_DV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_RV") == 0) {
            sTSysIndex_Lkw_RV = tsysInd;
            continue;
        }
    }
}

// specify calculation rule with improved user-defined VDF
double CalculateWithStaticAttributes(int tsysind, char tsysisopen, double cap, double t0,
    double pcuvol, double basevol, double vehvolsys[],
    double staticAttributeValues[],
    double para_a, double para_b, double para_c, double para_d, double para_f,
    double para_a2, double para_b2, double para_d2, double para_f2, double satcrit)
{

    double const av_ready = staticAttributeValues[CX_AV_READY];
    // get default PCU values
    double const pcu_P = staticAttributeValues[PCU_TSYS_P];
    double const pcu_LkwS_BV = staticAttributeValues[PCU_TSYS_LkwS_BV];
    double const pcu_LkwS_DV = staticAttributeValues[PCU_TSYS_LkwS_DV];
    double const pcu_LkwS_RV = staticAttributeValues[PCU_TSYS_LkwS_RV];
    double const pcu_Lkw_BV = staticAttributeValues[PCU_TSYS_Lkw_BV];
    double const pcu_Lkw_DV = staticAttributeValues[PCU_TSYS_Lkw_DV];
    double const pcu_Lkw_RV = staticAttributeValues[PCU_TSYS_Lkw_RV];

    // get possible PCU values for AV
    double pcu_CX_AV_used;
    double const pcu_CX_AV_default = staticAttributeValues[PCU_TSYS_CX_AV];
    double const pcu_CX_AV = staticAttributeValues[PCU_AV]; // PCU factor for each AV

    if (cap <= 0 || para_c <= 0) {
        return DBL_MAX;
    }

    // add up vehicle numbers separately for CV and AV
    double vehvol_all_CV = vehvolsys[sTSysIndex_P] + vehvolsys[sTSysIndex_LkwS_BV] + vehvolsys[sTSysIndex_LkwS_DV] +
vehvolsys[sTSysIndex_LkwS_RV] + vehvolsys[sTSysIndex_Lkw_BV] + vehvolsys[sTSysIndex_Lkw_DV] + vehvolsys[sTSysIndex_Lkw_RV];
    double vehvol_all_AV = vehvolsys[sTSysIndex_CX_AV];
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
    // if there are no vehicles on the link, TTC corresponds to T0
    if (vehvol_all_CV + vehvol_all_AV == 0)
        return t0;

    // if link is not AV-ready, use TSys-default PCU factor for AV
    if (av_ready == 0)
        pcu_CX_AV_used = pcu_CX_AV_default;
    else // if link is AV-ready, used PCU factor depends on AV share
        pcu_CX_AV_used = pcu_CX_AV;

    // mutliply respective PCU factor with vehicle volume
    double pcuvol_CX_AV = pcu_CX_AV_used * vehvolsys[sTSysIndex_CX_AV];
    double pcuvol_P = pcu_P * vehvolsys[sTSysIndex_P];
    double pcuvol_LkwS_BV = pcu_LkwS_BV * vehvolsys[sTSysIndex_LkwS_BV];
    double pcuvol_LkwS_DV = pcu_LkwS_DV * vehvolsys[sTSysIndex_LkwS_DV];
    double pcuvol_LkwS_RV = pcu_LkwS_RV * vehvolsys[sTSysIndex_LkwS_RV];
    double pcuvol_Lkw_BV = pcu_Lkw_BV * vehvolsys[sTSysIndex_Lkw_BV];
    double pcuvol_Lkw_DV = pcu_Lkw_DV * vehvolsys[sTSysIndex_Lkw_DV];
    double pcuvol_Lkw_RV = pcu_Lkw_RV * vehvolsys[sTSysIndex_Lkw_RV];

    double const totalvol_pcu = pcuvol_P + pcuvol_CX_AV + pcuvol_LkwS_BV + pcuvol_LkwS_DV + pcuvol_LkwS_RV + pcuvol_Lkw_BV + pcuvol_Lkw_DV +
pcuvol_Lkw_RV;

    double const sat = totalvol_pcu / (cap * para_c);

    // return TTC (standard function as LOHSE)
    if (sat <= satcrit)
        return t0 * (1 + para_a * (pow(sat, para_b)));
    else // sat > satcrit
        return t0 * (1 + para_a * (pow(satcrit, para_b))) + para_a * para_b * t0 * (sat - satcrit) * (pow(satcrit, para_b - 1));
}
```

Figure 7 shows the preview picture “VisumVDF_CX_AV_PCU_CONST_LOHSE_2020_x64.bmp”, which is displayed in Visum, if the user selects the related volume-delay function.

$$t_{cur}(sat) = \begin{cases} t_0(1 + a\,sat^b) & sat \le sat_{crit} \\ t_0(1 + a\,sat_{crit}{}^b) + a\,b\,t_0(sat_{crit})^{b-1}(sat - sat_{crit}) & sat > sat_{crit} \end{cases}$$

$$sat = \frac{q_{HDV} \cdot f_{HDV}^{PCU} + q_{LDV} \cdot f_{LDV}^{PCU} + q_{AV} \cdot f_{AV}^{PCU}}{q_{\max}}$$

if CX_AV-READY = 1, f_{AV}^{PCU} = CX_F_PCU_AV_1

Figure 7: Preview picture with calculation rule

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

VisumVDF_CX_AV_PCU_VAR_BPR_2020_x64.dll

```
#include "UserDefinedVDF_2020.h"
#include "tchar.h"
#include <math.h>
#include <float.h>
#include "string.h"

// VDF_Name appears as an entry in the dropdown list of volume-delay function types
wchar_t VDFName[] = _T("CX_AV_PCU_VAR_BPR_2020");

// VDFID is the internal name in the version file
char VDFID[] = "CX_AV_PCU_VAR_BPR_2020";

int INTERFACE_VERSION = 1;

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

char Init()
{
    return TRUE;
}

enum AttributeIndices
{
    CX_AV_READY,
    PCU_AV_0,
    PCU_AV_1,
    PCU_TSYS_P,
    PCU_TSYS_CX_AV,
    PCU_TSYS_LkwS_BV,
    PCU_TSYS_LkwS_DV,
    PCU_TSYS_LkwS_RV,
    PCU_TSYS_Lkw_BV,
    PCU_TSYS_Lkw_DV,
    PCU_TSYS_Lkw_RV,
    LastId
};

static const int MyMaxIDLength = 100; // Character limit for Strings
static wchar_t staticAttributeIDs[LastId][MyMaxIDLength] =
{
    // Strings of attributeIDs from links below must correspond to the order of AttributeIndices
    L"CX_AV-READY",
    L"CX_F_PCU_AV_0",
    L"CX_F_PCU_AV_1",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"P\")\\PCU",    // please note: \\ = \ and \" = "
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"CX_AV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_BV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_DV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_RV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_BV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_DV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_RV\")\\PCU",
};

int GetNumStaticAttributes()
{
    return LastId;
}

const wchar_t * GetStaticAttributeID(int attributesIndexZeroBased)
{
    return staticAttributeIDs[attributesIndexZeroBased];
}

void Destroy()
{
}

char IsThreadSafe()
{
    return TRUE;
}

char DependsOnTSys()
{
    return 2;
    // 2: VDF calculates the same value for all TSys and receives the the volumes per TSys
}

const wchar_t* GetName(const char *langid)
{
    return VDFName;
}

const char* GetID()
{
    return VDFID;
}
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
int GetInterfaceVersion()
{
    return INTERFACE_VERSION;
}

static int sTSysIndex_P = -1;
static int sTSysIndex_CX_AV = -1;
static int sTSysIndex_LkwS_BV = -1;
static int sTSysIndex_LkwS_DV = -1;
static int sTSysIndex_LkwS_RV = -1;
static int sTSysIndex_Lkw_BV = -1;
static int sTSysIndex_Lkw_DV = -1;
static int sTSysIndex_Lkw_RV = -1;

void SetTsysInfo(int numtsys, const wchar_t * tsysids[])
{
    // -1 means "TSys does not exist"
    sTSysIndex_P = -1;
    sTSysIndex_CX_AV = -1;
    sTSysIndex_LkwS_BV = -1;
    sTSysIndex_LkwS_DV = -1;
    sTSysIndex_LkwS_RV = -1;
    sTSysIndex_Lkw_BV = -1;
    sTSysIndex_Lkw_DV = -1;
    sTSysIndex_Lkw_RV = -1;

    // assign TSys Code to the related index within the TSys array
    // will be done once at the beginning of an assignment by Visum

    for (int tsysInd = 0; tsysInd < numtsys; ++tsysInd) {
        // wcscmp executes string comparison
        // if return value == 0, strings are identical
        if (wcscmp(tsysids[tsysInd], L"P") == 0) {
            sTSysIndex_P = tsysInd;
            continue; // exit for-loop and begin with next iteration
        }
        else if (wcscmp(tsysids[tsysInd], L"CX_AV") == 0) {
            sTSysIndex_CX_AV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_BV") == 0) {
            sTSysIndex_LkwS_BV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_DV") == 0) {
            sTSysIndex_LkwS_DV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_RV") == 0) {
            sTSysIndex_LkwS_RV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_BV") == 0) {
            sTSysIndex_Lkw_BV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_DV") == 0) {
            sTSysIndex_Lkw_DV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_RV") == 0) {
            sTSysIndex_Lkw_RV = tsysInd;
            continue;
        }
    }
}

// specify calculation rule with improved user-defined VDF
double CalculateWithStaticAttributes(int tsysind, char tsysisopen, double cap, double t0,
    double pcuvol, double basevol, double vehvolsys[],
    double staticAttributeValues[],
    double para_a, double para_b, double para_c, double para_d, double para_f,
    double para_a2, double para_b2, double para_d2, double para_f2, double satcrit)
{

    double const av_ready = staticAttributeValues[CX_AV_READY];
    // get default PCU values
    double const pcu_P = staticAttributeValues[PCU_TSYS_P];
    double const pcu_LkwS_BV = staticAttributeValues[PCU_TSYS_LkwS_BV];
    double const pcu_LkwS_DV = staticAttributeValues[PCU_TSYS_LkwS_DV];
    double const pcu_LkwS_RV = staticAttributeValues[PCU_TSYS_LkwS_RV];
    double const pcu_Lkw_BV = staticAttributeValues[PCU_TSYS_Lkw_BV];
    double const pcu_Lkw_DV = staticAttributeValues[PCU_TSYS_Lkw_DV];
    double const pcu_Lkw_RV = staticAttributeValues[PCU_TSYS_Lkw_RV];

    // get possible PCU values for AV
    double pcu_CX_AV_used;
    double const pcu_CX_AV_default = staticAttributeValues[PCU_TSYS_CX_AV];
    double const pcu_CX_AV_0 = staticAttributeValues[PCU_AV_0]; // PCU factor for "first AV"
    double const pcu_CX_AV_1 = staticAttributeValues[PCU_AV_1]; // PCU factor for 100% AV

    if (cap <= 0 || para_c <= 0) {
        return DBL_MAX;
    }

    // add up vehicle numbers separately for CV and AV
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
    double vehvol_all_CV = vehvolsys[sTSysIndex_P] + vehvolsys[sTSysIndex_LkwS_BV] + vehvolsys[sTSysIndex_LkwS_DV] +
vehvolsys[sTSysIndex_LkwS_RV] + vehvolsys[sTSysIndex_Lkw_BV] + vehvolsys[sTSysIndex_Lkw_DV] + vehvolsys[sTSysIndex_Lkw_RV];
    double vehvol_all_AV = vehvolsys[sTSysIndex_CX_AV];
    // to calculate the current AV share on the link
    double av_share = vehvol_all_AV / (vehvol_all_CV + vehvol_all_AV);

    // if there are no vehicles on the link, TTC corresponds to T0
    if (vehvol_all_CV + vehvol_all_AV == 0)
        return t0;

    // if link is not AV-ready, use TSys-default PCU factor for AV
    if (av_ready == 0)
        pcu_CX_AV_used = pcu_CX_AV_default;
    else // if link is AV-ready, used PCU factor depends on AV share
        pcu_CX_AV_used = pcu_CX_AV_0 - av_share * (pcu_CX_AV_0 - pcu_CX_AV_1);

    // mutliply respective PCU factor with vehicle volume
    double pcuvol_CX_AV = pcu_CX_AV_used * vehvolsys[sTSysIndex_CX_AV];
    double pcuvol_P = pcu_P * vehvolsys[sTSysIndex_P];
    double pcuvol_LkwS_BV = pcu_LkwS_BV * vehvolsys[sTSysIndex_LkwS_BV];
    double pcuvol_LkwS_DV = pcu_LkwS_DV * vehvolsys[sTSysIndex_LkwS_DV];
    double pcuvol_LkwS_RV = pcu_LkwS_RV * vehvolsys[sTSysIndex_LkwS_RV];
    double pcuvol_Lkw_BV = pcu_Lkw_BV * vehvolsys[sTSysIndex_Lkw_BV];
    double pcuvol_Lkw_DV = pcu_Lkw_DV * vehvolsys[sTSysIndex_Lkw_DV];
    double pcuvol_Lkw_RV = pcu_Lkw_RV * vehvolsys[sTSysIndex_Lkw_RV];

    double const totalvol_pcu = pcuvol_P + pcuvol_CX_AV + pcuvol_LkwS_BV + pcuvol_LkwS_DV + pcuvol_LkwS_RV + pcuvol_Lkw_BV + pcuvol_Lkw_DV +
pcuvol_Lkw_RV;

    double const sat = totalvol_pcu / (cap * para_c);

    // return TTC (standard function as BPR)
    return t0 * (1 + para_a * (pow(sat, para_b)));
}
```

Figure 8 shows the preview picture “VisumVDF_CX_AV_PCU_VAR_BPR_2020_x64.bmp”, which is displayed in Visum, if the user selects the related volume-delay function.

$$t_{car}(sat) = t_0 \cdot (1 + a \cdot sat^b) \qquad sat = \frac{q_{HDV} \cdot f_{HDV}^{PCU} + q_{LDV} \cdot f_{LDV}^{PCU} + q_{AV} \cdot f_{AV}^{PCU}(p_{AV})}{q_{\max}}$$

$$f_{AV}^{PCU}(p_{AV}) = f_{AV}^{PCU,0\%} - p_{AV} \cdot \left(f_{AV}^{PCU,0\%} - f_{AV}^{PCU,100\%}\right)$$

if CX_AV-READY = 1, $f_{AV}^{PCU,100\%}$ = CX_F_PCU_AV_1 and $f_{AV}^{PCU,0\%}$ = CX_F_PCU_AV_0

Figure 8: Preview picture with calculation rule

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

VisumVDF_CX_AV_PCU_VAR_LOHSE_2020_x64.dll

```
#include "UserDefinedVDF_2020.h"
#include "tchar.h"
#include <math.h>
#include <float.h>
#include "string.h"

// VDF_Name appears as an entry in the dropdown list of volume-delay function types
wchar_t VDFName[] = _T("CX_AV_PCU_VAR_LOHSE_2020");

// VDFID is the internal name in the version file
char VDFID[] = "CX_AV_PCU_VAR_LOHSE_2020";

int INTERFACE_VERSION = 1;

#ifndef TRUE
#define TRUE 1
#endif

#ifndef FALSE
#define FALSE 0
#endif

char Init()
{
    return TRUE;
}

enum AttributeIndices
{
    CX_AV_READY,
    PCU_AV_0,
    PCU_AV_1,
    PCU_TSYS_P,
    PCU_TSYS_CX_AV,
    PCU_TSYS_LkwS_BV,
    PCU_TSYS_LkwS_DV,
    PCU_TSYS_LkwS_RV,
    PCU_TSYS_Lkw_BV,
    PCU_TSYS_Lkw_DV,
    PCU_TSYS_Lkw_RV,
    LastId
};

static const int MyMaxIDLength = 100; // Character limit for Strings
static wchar_t staticAttributeIDs[LastId][MyMaxIDLength] =
{
    // Strings of attributeIDs from links below must correspond to the order of AttributeIndices
    L"CX_AV-READY",
    L"CX_F_PCU_AV_0",
    L"CX_F_PCU_AV_1",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"P\")\\PCU",    // please note: \\ = \ and \" = "
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"CX_AV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_BV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_DV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"LkwS_RV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_BV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_DV\")\\PCU",
    L"NETWORK\\EXACTLYONE:TSYSS([CODE]=\"Lkw_RV\")\\PCU",
};

int GetNumStaticAttributes()
{
    return LastId;
}

const wchar_t * GetStaticAttributeID(int attributesIndexZeroBased)
{
    return staticAttributeIDs[attributesIndexZeroBased];
}

void Destroy()
{
}

char IsThreadSafe()
{
    return TRUE;
}

char DependsOnTSys()
{
    return 2;
    // 2: VDF calculates the same value for all TSys and receives the the volumes per TSys
}

const wchar_t* GetName(const char *langid)
{
    return VDFName;
}

const char* GetID()
{
    return VDFID;
}
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
int GetInterfaceVersion()
{
    return INTERFACE_VERSION;
}

static int sTSysIndex_P = -1;
static int sTSysIndex_CX_AV = -1;
static int sTSysIndex_LkwS_BV = -1;
static int sTSysIndex_LkwS_DV = -1;
static int sTSysIndex_LkwS_RV = -1;
static int sTSysIndex_Lkw_BV = -1;
static int sTSysIndex_Lkw_DV = -1;
static int sTSysIndex_Lkw_RV = -1;

void SetTsysInfo(int numtsys, const wchar_t * tsysids[])
{
    // -1 means "TSys does not exist"
    sTSysIndex_P = -1;
    sTSysIndex_CX_AV = -1;
    sTSysIndex_LkwS_BV = -1;
    sTSysIndex_LkwS_DV = -1;
    sTSysIndex_LkwS_RV = -1;
    sTSysIndex_Lkw_BV = -1;
    sTSysIndex_Lkw_DV = -1;
    sTSysIndex_Lkw_RV = -1;

    // assign TSys Code to the related index within the TSys array
    // will be done once at the beginning of an assignment by Visum

    for (int tsysInd = 0; tsysInd < numtsys; ++tsysInd) {
        // wcscmp executes string comparison
        // if return value == 0, strings are identical
        if (wcscmp(tsysids[tsysInd], L"P") == 0) {
            sTSysIndex_P = tsysInd;
            continue; // exit for-loop and begin with next iteration
        }
        else if (wcscmp(tsysids[tsysInd], L"CX_AV") == 0) {
            sTSysIndex_CX_AV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_BV") == 0) {
            sTSysIndex_LkwS_BV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_DV") == 0) {
            sTSysIndex_LkwS_DV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"LkwS_RV") == 0) {
            sTSysIndex_LkwS_RV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_BV") == 0) {
            sTSysIndex_Lkw_BV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_DV") == 0) {
            sTSysIndex_Lkw_DV = tsysInd;
            continue;
        }
        else if (wcscmp(tsysids[tsysInd], L"Lkw_RV") == 0) {
            sTSysIndex_Lkw_RV = tsysInd;
            continue;
        }
    }
}

// specify calculation rule with improved user-defined VDF
double CalculateWithStaticAttributes(int tsysind, char tsysisopen, double cap, double t0,
    double pcuvol, double basevol, double vehvolsys[],
    double staticAttributeValues[],
    double para_a, double para_b, double para_c, double para_d, double para_f,
    double para_a2, double para_b2, double para_d2, double para_f2, double satcrit)
{

    double const av_ready = staticAttributeValues[CX_AV_READY];
    // get default PCU values
    double const pcu_P = staticAttributeValues[PCU_TSYS_P];
    double const pcu_LkwS_BV = staticAttributeValues[PCU_TSYS_LkwS_BV];
    double const pcu_LkwS_DV = staticAttributeValues[PCU_TSYS_LkwS_DV];
    double const pcu_LkwS_RV = staticAttributeValues[PCU_TSYS_LkwS_RV];
    double const pcu_Lkw_BV = staticAttributeValues[PCU_TSYS_Lkw_BV];
    double const pcu_Lkw_DV = staticAttributeValues[PCU_TSYS_Lkw_DV];
    double const pcu_Lkw_RV = staticAttributeValues[PCU_TSYS_Lkw_RV];

    // get possible PCU values for AV
    double pcu_CX_AV_used;
    double const pcu_CX_AV_default = staticAttributeValues[PCU_TSYS_CX_AV];
    double const pcu_CX_AV_0 = staticAttributeValues[PCU_AV_0]; // PCU factor for "first AV"
    double const pcu_CX_AV_1 = staticAttributeValues[PCU_AV_1]; // PCU factor for 100% AV

    if (cap <= 0 || para_c <= 0) {
        return DBL_MAX;
    }

    // add up vehicle numbers separately for CV and AV
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
    double vehvol_all_CV = vehvolsys[sTSysIndex_P] + vehvolsys[sTSysIndex_LkwS_BV] + vehvolsys[sTSysIndex_LkwS_DV] +
vehvolsys[sTSysIndex_LkwS_RV] + vehvolsys[sTSysIndex_Lkw_BV] + vehvolsys[sTSysIndex_Lkw_DV] + vehvolsys[sTSysIndex_Lkw_RV];
    double vehvol_all_AV = vehvolsys[sTSysIndex_CX_AV];
    // to calculate the current AV share on the link
    double av_share = vehvol_all_AV / (vehvol_all_CV + vehvol_all_AV);

    // if there are no vehicles on the link, TTC corresponds to T0
    if (vehvol_all_CV + vehvol_all_AV == 0)
        return t0;

    // if link is not AV-ready, use TSys-default PCU factor for AV
    if (av_ready == 0)
        pcu_CX_AV_used = pcu_CX_AV_default;
    else // if link is AV-ready, used PCU factor depends on AV share
        pcu_CX_AV_used = pcu_CX_AV_0 - av_share * (pcu_CX_AV_0 - pcu_CX_AV_1);

    // mutliply respective PCU factor with vehicle volume
    double pcuvol_CX_AV = pcu_CX_AV_used * vehvolsys[sTSysIndex_CX_AV];
    double pcuvol_P = pcu_P * vehvolsys[sTSysIndex_P];
    double pcuvol_LkwS_BV = pcu_LkwS_BV * vehvolsys[sTSysIndex_LkwS_BV];
    double pcuvol_LkwS_DV = pcu_LkwS_DV * vehvolsys[sTSysIndex_LkwS_DV];
    double pcuvol_LkwS_RV = pcu_LkwS_RV * vehvolsys[sTSysIndex_LkwS_RV];
    double pcuvol_Lkw_BV = pcu_Lkw_BV * vehvolsys[sTSysIndex_Lkw_BV];
    double pcuvol_Lkw_DV = pcu_Lkw_DV * vehvolsys[sTSysIndex_Lkw_DV];
    double pcuvol_Lkw_RV = pcu_Lkw_RV * vehvolsys[sTSysIndex_Lkw_RV];

    double const totalvol_pcu = pcuvol_P + pcuvol_CX_AV + pcuvol_LkwS_BV + pcuvol_LkwS_DV + pcuvol_LkwS_RV + pcuvol_Lkw_BV + pcuvol_Lkw_DV +
pcuvol_Lkw_RV;

    double const sat = totalvol_pcu / (cap * para_c);

    // return TTC (standard function as LOHSE)
    if (sat <= satcrit)
        return t0 * (1 + para_a * (pow(sat, para_b)));
    else // sat > satcrit
        return t0 * (1 + para_a * (pow(satcrit, para_b))) + para_a * para_b * t0 * (sat - satcrit) * (pow(satcrit, para_b - 1));
}
```

Figure 9 shows the preview picture “VisumVDF_CX_AV_PCU_VAR_LOHSE_2020_x64.bmp”, which is displayed in Visum, if the user selects the related volume-delay function.

$$t_{cur}(sat) = \begin{cases} t_0(1 + a\,sat^b) & sat \le sat_{crit} \\ t_0(1 + a\,sat_{crit}{}^b) + a\,b\,t_0(sat_{crit})^{b-1}(sat - sat_{crit}) & sat > sat_{crit} \end{cases}$$

$$sat = \frac{q_{HDV} \cdot f_{HDV}^{PCU} + q_{LDV} \cdot f_{LDV}^{PCU} + q_{AV} \cdot f_{AV}^{PCU}(p_{AV})}{q_{\max}}$$

$$f_{AV}^{PCU}(p_{AV}) = f_{AV}^{PCU,0\%} - p_{AV} \cdot \left(f_{AV}^{PCU,0\%} - f_{AV}^{PCU,100\%}\right)$$

if CX_AV-READY = 1, $f_{AV}^{PCU,100\%}$ = CX_F_PCU_AV_1 and $f_{AV}^{PCU,0\%}$ = CX_F_PCU_AV_0

Figure 9: Preview picture with calculation rule

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

6.2 Tool: Perception of automated travel time

6.2.1 Script file: User-defined attributes

The submitted file “CoEXist_Create_User-Defined_Attributes_-_Extension_for_perceived_automated_travel_time_impacts.vbs” contains the following code:

Visum 18 or lower

```
'*****************************************************
' This script creates user-defined attributes for different network elements in Visum
' CoEXist - WP2 Macroscopic Modelling Tool
' USTUTT - University of Stuttgart
' October 2018
'*****************************************************

' One call for creating each user-defined attribute (UDA)
' AddUDA is defined below
Call Add_UDA("Links",       "CX_AV-READY",                    "CX_AV-READY",                    1,      0,  0,  1,      0,  "", false)
Call Add_UDA("Linktypes",   "CX_AV-READY",                    "CX_AV-READY",                    1,      0,  0,  1,      0,  "", false)
Call Add_UDA("Links", "CX_TTC_AV-READY", "CX_TTC_AV-READY", 165, 2,  0, 0, 0,  "IF([LINKTYPE\CX_AV-READY]=1,[TCUR_PRTSYS(P)],0)",   false)
Call Add_UDA("Net",         "CX_AV-SHARE",                    "CX_AV-SHARE",                    1,      0,  0,  100,    0,  "", false)
Call Add_UDA("Net",         "CX_THRESHOLD_IVT-PERCEPTION_A",  "CX_THRESHOLD_IVT-PERCEPTION_A",  1,      0,  0,  0,      0,  "", false)
Call Add_UDA("Net",         "CX_IVT-PERCEPTION_FACTOR",       "CX_IVT-PERCEPTION_FACTOR",       2,      2,  0,  0,      1,  "", false)
Call Add_UDA("Matrices",    "CX_ID",                          "CX_ID",                          5,      0,  0,  0,      "", "", false)

' One call for adding a comment to each user-defined attribute
' SetUDAComment is defined below
Call Set_UDA_Comment("Links",       "CX_AV-READY",                    "0: link is not AV-ready, 1: link is AV-ready")
Call Set_UDA_Comment("Links",       "CX_TTC_AV-READY",                "Current travel time on AV-ready links [min]")
Call Set_UDA_Comment("Linktypes",   "CX_AV-READY",                    "0: link type is not AV-ready, 1: link type is AV-ready")
Call Set_UDA_Comment("Net",         "CX_AV-SHARE",                    "Fixed AV share as a percentage for splitting the demand")
Call Set_UDA_Comment("Net",         "CX_THRESHOLD_IVT-PERCEPTION_A",  "Threshold A for the perception of in-vehicle time in automated
driving mode [min]. For a travel time in automated mode longer than A, the factor for perceived automated travel time will have an effect.")
Call Set_UDA_Comment("Net",         "CX_IVT-PERCEPTION_FACTOR",       "Factor for the perception of in-vehicle time in automated driving
mode")
Call Set_UDA_Comment("Matrices",    "CX_ID",                          "CoEXist-unique identifier for working with formula matrices")

'*****************************************************
' Commonly used ValueTypes:
' Member                    Value   Summary
' ValueType_Int             1       Integer value (int)
' ValueType_Real            2       Real value (real)
' ValueType_String          5       String value (char*)
' ValueType_Duration        6       Duration (seconds or minutes depending on time format option)
' ValueType_TimePoint       7       Time stamp in seconds
' ValueType_Bool            9       Boolean value (true / false)
' ValueType_LongDuration    165     Precise duration (seconds or minutes depending on time format option)
'*****************************************************

'*****************************************************
' Definitions of subs and functions below
'*****************************************************

' Creates a user-defined attribute as specified above
Sub Add_UDA(ObjId, UDA_Code, UDA_Name, ValueType, Decplaces, MinVal, MaxVal, DefVal, Formula, canBeEmpty)
    If UDA_Name= "" then UDA_Name=UDA_Code
    On Error Resume Next
    Set VisObjects = GetVisObj(ObjId)
    VisObjects.AddUserDefinedAttribute UDA_Code, UDA_Code, UDA_Name, ValueType, Decplaces, , MinVal, MaxVal, DefVal, , , Formula, canBeEmpty
End Sub

' Sets a comment for a user-defined attribute as specified above
Sub Set_UDA_Comment(ObjId, UDA_Code, UDA_Comment)
    Set VisObjects = GetVisObj(ObjId)
    For Each Obj In VisObjects.Attributes.GetAll
    If Obj.Code = UDA_Code Then
        Obj.Comment = UDA_Comment
        Exit For
    End If
    Next
End Sub

' Gets a pointer to a Visum object class
Function GetVisObj(ObjID)
ObjID = LCase(ObjID)
    If ObjId = "net" Then
        Set VisObjects=Visum.Net
    ElseIf ObjId = "links" Then
        Set VisObjects=Visum.Net.Links
    ElseIf ObjId = "linktypes" Then
        Set VisObjects=Visum.Net.LinkTypes
    ElseIf ObjId = "matrices" Then
        Set VisObjects = Visum.Net.Matrices
    End If
Set GetVisObj=VisObjects
End Function
```

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

Visum 2020

```
'*****************************************************
' This script creates user-defined attributes for different network elements in Visum
' CoEXist - WP2 Macroscopic Modelling Tool
' USTUTT - University of Stuttgart
' February 2020
'*****************************************************

' One call for creating each user-defined attribute (UDA)
' AddUDA is defined below
Call Add_UDA("Links",       "CX_AV-READY",                  "CX_AV-READY",                  1,    0,  0,  1,    0,  "",
false)
Call Add_UDA("Links",       "CX_TTC_AV-READY",              "CX_TTC_AV-READY",              165,  2,  0,  0,    0,  "IF([CX_AV-
READY]=1,[TCUR_PRTSYS(P)],0)",   false)
Call Add_UDA("Net",         "CX_AV-SHARE",                  "CX_AV-SHARE",                  1,    0,  0,  100,  0,  "",
false)
Call Add_UDA("Net",         "CX_THRESHOLD_IVT-PERCEPTION",  "CX_THRESHOLD_IVT-PERCEPTION",  1,    0,  0,  0,    0,  "",
false)
Call Add_UDA("Net",         "CX_IVT-PERCEPTION_FACTOR",     "CX_IVT-PERCEPTION_FACTOR",     2,    2,  0,  0,    1,  "",
false)
Call Add_UDA("Matrices",    "CX_ID",                        "CX_ID",                        5,    0,  0,  0,    "", "",
false)

' One call for adding a comment to each user-defined attribute
' SetUDAComment is defined below
Call Set_UDA_Comment("Links",       "CX_AV-READY",                  "0: link is not AV-ready, 1: link is AV-ready")
Call Set_UDA_Comment("Links",       "CX_TTC_AV-READY",              "Current travel time on AV-ready links [min]")
Call Set_UDA_Comment("Net",         "CX_AV-SHARE",                  "Fixed AV share as a percentage for splitting the demand")
Call Set_UDA_Comment("Net",         "CX_THRESHOLD_IVT-PERCEPTION",  "Threshold for the perception of in-vehicle time in automated driving
mode [min]. For a travel time in automated mode longer than A, the factor for perceived automated travel time will have an effect.")
Call Set_UDA_Comment("Net",         "CX_IVT-PERCEPTION_FACTOR",     "Factor for the perception of in-vehicle time in automated driving
mode")
Call Set_UDA_Comment("Matrices",    "CX_ID",                        "CoEXist-unique identifier for working with formula matrices")

'*****************************************************
' Commonly used ValueTypes:
' Member                  Value   Summary
' ValueType_Int           1       Integer value (int)
' ValueType_Real          2       Real value (real)
' ValueType_String        5       String value (char*)
' ValueType_Duration      6       Duration (seconds or minutes depending on time format option)
' ValueType_TimePoint     7       Time stamp in seconds
' ValueType_Bool          9       Boolean value (true / false)
' ValueType_LongDuration  165     Precise duration (seconds or minutes depending on time format option)
'*****************************************************

'*****************************************************
' Definitions of subs and functions below
'*****************************************************

' Creates a user-defined attribute as specified above
Sub Add_UDA(ObjId, UDA_Code, UDA_Name, ValueType, Decplaces, MinVal, MaxVal, DefVal, Formula, canBeEmpty)
    If UDA_Name= "" then UDA_Name=UDA_Code
    On Error Resume Next
    Set VisObjects = GetVisObj(ObjId)
    VisObjects.AddUserDefinedAttribute UDA_Code, UDA_Code, UDA_Name, ValueType, Decplaces, , MinVal, MaxVal, DefVal, , , Formula, ,
canBeEmpty
End Sub

' Sets a comment for a user-defined attribute as specified above
Sub Set_UDA_Comment(ObjId, UDA_Code, UDA_Comment)
    Set VisObjects = GetVisObj(ObjId)
    For Each Obj In VisObjects.Attributes.GetAll
    If Obj.Code = UDA_Code Then
        Obj.Comment = UDA_Comment
        Exit For
    End If
    Next
End Sub

' Gets a pointer to a Visum object class
Function GetVisObj(ObjID)
ObjID = LCase(ObjID)
    If ObjId = "net" Then
        Set VisObjects=Visum.Net
    ElseIf ObjId = "links" Then
        Set VisObjects=Visum.Net.Links
    ElseIf ObjId = "linktypes" Then
        Set VisObjects=Visum.Net.LinkTypes
        ElseIf ObjId = "matrices" Then
        Set VisObjects = Visum.Net.Matrices
    End If
Set GetVisObj=VisObjects
End Function
```

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No. 723201

6.2.2 Procedure sequence

The submitted file “CoEXist_Procedure_Parameters_-_Extension_for_perceived_automated_travel_time_impacts.xml” contains the following code:

Visum 18 or lower

```
<?xml  version = "1.0"  encoding = "UTF-8"?>
<PROCEDURES  VERSION = "1705">
  <OPERATIONS>
    <OPERATION
      NO = "1"
      CODE = ""
      OPERATIONTYPE = "Group"
      ACTIVE = "1"
      COMMENT = "Set AV-related attributes regarding perceived automated travel time"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <GROUPPARA        ISEXPANDED = "1"      />
    </OPERATION>
    <OPERATION
      NO = "2"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "User input: AV share [percentage]"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "NETWORK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "CX_AV-SHARE"
        ONLYACTIVE = "0"
        FORMULA = "50"
      />
    </OPERATION>
    <OPERATION
      NO = "3"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "User input: threshold A for automated in-vehicle time perception [min]"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "NETWORK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "CX_THRESHOLD_IVT-PERCEPTION_A"
        ONLYACTIVE = "0"
        FORMULA = "10"
      />
    </OPERATION>
    <OPERATION
      NO = "4"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "User input: factor A for automated in-vehicle time perception"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
```

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

```
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "NETWORK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "CX_IVT-PERCEPTION_FACTOR"
        ONLYACTIVE = "0"
        FORMULA = "0.8"
      />
    </OPERATION>
  </OPERATIONS>
</PROCEDURES>
```

Visum 2020

```
<?xml  version = "1.0"  encoding = "UTF-8"?>
<PROCEDURES  VERSION = "1902">
  <OPERATIONS>
    <OPERATION
      NO = "1"
      CODE = ""
      OPERATIONTYPE = "Group"
      ACTIVE = "1"
      COMMENT = "Set AV-related attributes regarding perceived automated travel time"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODES = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = ""
      OPERATIONVARIABLECOUNT = "">
      <GROUPPARA        ISEXPANDED = "1"      />
    </OPERATION>
    <OPERATION
      NO = "2"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "User input: AV share [percentage: 50 = 50%]"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODES = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = ""
      OPERATIONVARIABLECOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "NETWORK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "CX_AV-SHARE"
        ONLYACTIVE = "0"
        FORMULA = "50"
      />
    </OPERATION>
    <OPERATION
      NO = "3"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "User input: threshold A for automated in-vehicle time perception [min]"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODES = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = ""
      OPERATIONVARIABLECOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "NETWORK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "CX_THRESHOLD_IVT-PERCEPTION"
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

```
        ONLYACTIVE = "0"
        FORMULA = "10"
      />
    </OPERATION>
    <OPERATION
      NO = "4"
      CODE = ""
      OPERATIONTYPE = "EditAttribute"
      ACTIVE = "1"
      COMMENT = "User input: factor A for automated in-vehicle time perception"
      EXECUTED = "0"
      STARTTIME = ""
      ENDTIME = ""
      DURATION = ""
      MESSAGES = ""
      RESULTMESSAGE = ""
      SUCCESS = "0"
      COMPUTENODE = ""
      WARNINGCOUNT = ""
      ERRORCOUNT = ""
      INFORMATIONCOUNT = ""
      OPERATIONVARIABLECOUNT = "">
      <ATTRIBUTEFORMULAPARA
        NETOBJECTTYPE = "NETWORK"
        INCLUDESUBCATEGORIES = "0"
        RESULTATTRNAME = "CX_IVT-PERCEPTION_FACTOR"
        ONLYACTIVE = "0"
        FORMULA = "0.8"
      />
    </OPERATION>
  </OPERATIONS>
</PROCEDURES>
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

6.2.3 Script file: Formula matrices

The submitted file “CoEXist_Create_Formula_Matrices_-_Extension_for_perceived_automated_travel_time_impacts.vbs” contains the following code:

Visum 18 or lower

```
'*****************************************************
' This script creates formula matrices in Visum
' CoEXist - WP2 Macroscopic Modelling Tool
' USTUTT - University of Stuttgart
' October 2018
'*****************************************************

' One call for creating each formula matrix
Call AddFormulaMat(-1, "CX_CV_DEMAND",              "", 2,  3,  "Matrix([CX_ID] = "+Chr(34)+"CX_CAR_DEMAND"+Chr(34)+")*(1-[CX_AV-
SHARE]/100)")

Call AddFormulaMat(-1, "CX_AV_DEMAND",              "", 2,  3,  "Matrix([CX_ID] = "+Chr(34)+"CX_CAR_DEMAND"+Chr(34)+")*[CX_AV-SHARE]/100")

Call AddFormulaMat(-1, "CX_TTC_NOT_AV-READY",       "", 2,  4,  "Matrix([CX_ID] = "+Chr(34)+"CX_TTC_CAR"+Chr(34)+")-Matrix([CX_ID] =
"+Chr(34)+"CX_TTC_AV-READY"+Chr(34)+")")

Call AddFormulaMat(-1, "CX_TTC_AV-READY_PERCEIVED", "", 2,  4,  "IF(Matrix([CX_ID]="+Chr(34)+"CX_TTC_AV-READY"+Chr(34)+")<=[CX_THRESHOLD_IVT-
PERCEPTION_A],Matrix([CX_ID]="+ _
                                                                    Chr(34)+"CX_TTC_AV-READY"+Chr(34)+"),[CX_THRESHOLD_IVT-
PERCEPTION_A]+[CX_IVT-PERCEPTION_FACTOR]*(Matrix([CX_ID]="+ _
                                                                    Chr(34)+"CX_TTC_AV-READY"+Chr(34)+")-[CX_THRESHOLD_IVT-PERCEPTION_A]))")

Call AddFormulaMat(-1, "CX_TTC_CV_x_AV",            "", 2,  4,  "Matrix([CX_ID]="+Chr(34)+"CX_TTC_CAR"+Chr(34)+")*(1-[CX_AV-
SHARE]/100)+(Matrix([CX_ID]="+Chr(34)+"CX_TTC_NOT_AV-READY"+ _
                                                                    Chr(34)+")+Matrix([CX_ID]="+Chr(34)+"CX_TTC_AV-
READY_PERCEIVED"+Chr(34)+"))*([CX_AV-SHARE]/100)")

'*****************************************************
' Commonly used values for MatrixType and ObjectTypeRef
' Member                    Value   Summary
' MATRIXTYPE_ANY            2       Any matrix type
' MATRIXTYPE_DEMAND         3       Demand matrix
' MATRIXTYPE_SKIM           4       Skim matrix
' OBJECTTYPEREF_ZONE        2       Zones
' OBJECTTYPEREF_MAINZONE    3       Main zones
' OBJECTTYPEREF_STOPAREA    4       Stop areas
'*****************************************************

' Creates a formula matrix on zone level
Function AddFormulaMat(MatNo, Code, Name, ObjectTypeRef, Matrixtype, Formula)
    If Name="" Then Name=Code
    On Error Resume Next
    Set x = Visum.Net.AddMatrixWithFormula (MatNo, Formula, ObjectTypeRef, Matrixtype)
        x.attvalue("Code")  = Code
        x.attvalue("Name")  = Name
        x.attvalue("CX_ID") = Code
End Function
```

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

Visum 2020

```
'*****************************************************
' This script creates formula matrices in Visum
' CoEXist - WP2 Macroscopic Modelling Tool
' USTUTT - University of Stuttgart
' February 2020
'*****************************************************

' One call for creating each formula matrix
Call AddFormulaMat(-1, "CX_CV_DEMAND",              "", 2,  3,  "Matrix([CX_ID] = "+Chr(34)+"CX_CAR_DEMAND"+Chr(34)+")*(1-[CX_AV-
SHARE]/100)")

Call AddFormulaMat(-1, "CX_AV_DEMAND",              "", 2,  3,  "Matrix([CX_ID] = "+Chr(34)+"CX_CAR_DEMAND"+Chr(34)+")*[CX_AV-SHARE]/100")

Call AddFormulaMat(-1, "CX_TTC_NOT_AV-READY",       "", 2,  4,  "Matrix([CX_ID] = "+Chr(34)+"CX_TTC_CAR"+Chr(34)+")-Matrix([CX_ID] =
"+Chr(34)+"CX_TTC_AV-READY"+Chr(34)+")")

Call AddFormulaMat(-1, "CX_TTC_AV-READY_PERCEIVED", "", 2,  4,  "IF(Matrix([CX_ID]="+Chr(34)+"CX_TTC_AV-READY"+Chr(34)+")<=[CX_THRESHOLD_IVT-
PERCEPTION],Matrix([CX_ID]="+ _
                                                                  Chr(34)+"CX_TTC_AV-READY"+Chr(34)+"),[CX_THRESHOLD_IVT-
PERCEPTION]+[CX_IVT-PERCEPTION_FACTOR]*(Matrix([CX_ID]="+ _
                                                                  Chr(34)+"CX_TTC_AV-READY"+Chr(34)+")-[CX_THRESHOLD_IVT-PERCEPTION]))")

Call AddFormulaMat(-1, "CX_TTC_CV_x_AV",            "", 2,  4,  "Matrix([CX_ID]="+Chr(34)+"CX_TTC_CAR"+Chr(34)+")*(1-[CX_AV-
SHARE]/100)+(Matrix([CX_ID]="+Chr(34)+"CX_TTC_NOT_AV-READY"+ _
                                                                  Chr(34)+")+Matrix([CX_ID]="+Chr(34)+"CX_TTC_AV-
READY_PERCEIVED"+Chr(34)+"))*([CX_AV-SHARE]/100)")

'*****************************************************
' Commonly used values for MatrixType and ObjectTypeRef
' Member                    Value   Summary
' MATRIXTYPE_ANY            2       Any matrix type
' MATRIXTYPE_DEMAND         3       Demand matrix
' MATRIXTYPE_SKIM           4       Skim matrix
' OBJECTTYPEREF_ZONE        2       Zones
' OBJECTTYPEREF_MAINZONE    3       Main zones
' OBJECTTYPEREF_STOPAREA    4       Stop areas
'*****************************************************

' Creates a formula matrix on zone level
Function AddFormulaMat(MatNo, Code, Name, ObjectTypeRef, Matrixtype, Formula)
    If Name="" Then Name=Code
    On Error Resume Next
    Set x = Visum.Net.AddMatrixWithFormula (MatNo, Formula, ObjectTypeRef, Matrixtype)
        x.attvalue("Code")  = Code
        x.attvalue("Name")  = Name
        x.attvalue("CX_ID") = Code
End Function
```

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

6.3 Tool: Ridematching

6.3.1 Script file: rs_match_all_to_all.vbs

```
' **************************
' ps = pathset
' no = number
' vol = volumn
' cap = capacity
' net = network

ps_no = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("No", False)
ps_from_zone_no = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("FromZoneNo", False)
ps_to_zone_no = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("ToZoneNo", False)
ps_sequence_of_zones = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("rs_zone_sequence", False)
ps_vol = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("Vol", False)
ps_matched_path_id = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("rs_matched_path_id", False)
ps_replaced_path_id = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("rs_replaced_path_id", False)
ps_cap = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("rs_capacity", False)
net_veh_cap = Visum.Net.AttValue("Vehicle_Capacity")

For i = 0 To UBound(ps_no) ' loop:     i = Supplier
    For j = 0 To UBound(ps_no) ' loop:     j = Demander ' exit: RS_BefPers = 0
        ' check:    fromzone in sequence of zones and
        '           tozone in sequence of zones and
        '           fromzone before tozone in sequence of zones
        If InStr(1, ps_sequence_of_zones(i, 1), ps_from_zone_no(j, 1)) > 0 And _
           InStr(1, ps_sequence_of_zones(i, 1), ps_to_zone_no(j, 1)) > 0 And _
           InStr(1, ps_sequence_of_zones(i, 1), ps_from_zone_no(j, 1)) <= _
           InStr(1, ps_sequence_of_zones(i, 1), ps_to_zone_no(j, 1)) And _
           ps_vol(j, 1) <> 0 And _
           ps_cap(i, 1) > 0 And _
           i <> j Then
            If ps_cap(i, 1) >= ps_vol(j, 1) Then
                ps_cap(i, 1) = ps_cap(i, 1) - ps_vol(j, 1)
                ps_cap(j, 1) = 0
                ps_vol(i, 1) = ps_vol(i, 1) + ps_vol(j, 1)
                ps_matched_path_id(i, 1) = ps_matched_path_id(i, 1) & "," & ps_no(j, 1) & "(" & ps_vol(j, 1) & ")"
                ps_replaced_path_id(j, 1) = ps_replaced_path_id(j, 1) & "," & ps_no(i, 1) & "(" & ps_vol(j, 1) & ")"
                ps_vol(j, 1) = 0
            Else
                ps_vol(i, 1) = ps_vol(i, 1) + ps_cap(i, 1)
                ps_vol(j, 1) = ps_vol(j, 1) - ps_cap(i, 1)
                ps_matched_path_id(i, 1) = ps_matched_path_id(i, 1) & "," & ps_no(j, 1) & "(" & ps_cap(i, 1) & ")"
                ps_replaced_path_id(j, 1) = ps_replaced_path_id(j, 1) & "," & ps_no(i, 1) & "(" & ps_cap(i, 1) & ")"
                ps_cap(i, 1) = 0
                ps_cap(j, 1) = Round(ps_vol(j, 1) / net_veh_cap + 0.5) * net_veh_cap - ps_vol(j, 1)
            End If
        End If
    Next
Next

' write information back to Visum
Call Visum.Net.PathSets.ItemByKey(2).Paths.SetMultiAttValues("rs_satisfied_demand", ps_vol)
Call Visum.Net.PathSets.ItemByKey(2).Paths.SetMultiAttValues("rs_matched_path_id", ps_matched_path_id)
Call Visum.Net.PathSets.ItemByKey(2).Paths.SetMultiAttValues("rs_replaced_path_id", ps_replaced_path_id)
Call Visum.Net.PathSets.ItemByKey(2).Paths.SetMultiAttValues("rs_capacity", ps_cap)
```

6.3.2 Script file: rs_reduce_zone_sequence.vbs

```
' **************************
PathSetNo       = Visum.Net.AttValue("NoPathSet")

sequence_of_zones   = Visum.Net.PathSets.ItemByKey(PathSetNo).Paths.GetMultiAttValues("Concatenate:Nodes\node_intersect_with_zones", false)
from_zone_no        = Visum.Net.PathSets.ItemByKey(PathSetNo).Paths.GetMultiAttValues("FromZoneNo", false)
to_zone_no          = Visum.Net.PathSets.ItemByKey(PathSetNo).Paths.GetMultiAttValues("ToZoneNo", false)

For i = 0 To UBound(sequence_of_zones)
    sequence_of_zones(i, 1) = "," & from_zone_no(i, 1) & "," & sequence_of_zones(i, 1) & "," & to_zone_no(i, 1) & "," ' add: zone origin and
destination to ensure origin and destination are part of the sequence
    sequence_of_zones(i, 1) = zone_sequence(sequence_of_zones(i, 1)) ' call: function to reduce the zone sequence, keep necessary information
about the route
Next

call Visum.Net.PathSets.ItemByKey(PathSetNo).Paths.SetMultiAttValues("rs_zone_sequence", sequence_of_zones) ' return:   reduced zone sequence
to Visum

' uniques the sequence of (main)zones by keeping the order
Function zone_sequence(vSeq)
    vSeq = Replace(vSeq, ".", "")    ' replace string "." by ""
    vSeq = Split(vSeq, ",")          ' split string by ","
    For j = 0 To UBound(vSeq) - 1
        If vSeq(j) <> vSeq(j + 1) And vSeq(j) <> "" Then
            zone_sequence = zone_sequence & "," & vSeq(j)    ' if zone exist then consider otherwise continue with next entry
        End If
    Next
    If Len(zone_sequence) > 0 Then zone_sequence = Right(zone_sequence, Len(zone_sequence) - 1) ' remove first "," in string zone_sequence
```

This project has received funding from the European Union's Horizon 2020
research and innovation programme under grant agreement No. 723201

```
    zone_sequence = zone_sequence & ","
    vSeq = Split(zone_sequence, ",")
    zone_sequence = ""
    For j = 0 To UBound(vSeq) - 1
        If vSeq(j) <> vSeq(j + 1) Then
            zone_sequence = zone_sequence & "," & vSeq(j) ' if zone exist then consider otherwise continue with next entry
        End If
    Next
    If Len(zone_sequence) > 0 Then zone_sequence = Right(zone_sequence, Len(zone_sequence) - 1) ' remove first "," in string zone_sequence
End Function
```

6.3.3 Script file: rs_write_matrix_route_match.vbs

```
' **************************
zone_no = Visum.Net.Zones.GetMultiAttValues("NO")
ps_fromzoneno = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("FROMZONENO", True)
ps_tozoneno = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("TOZONENO", True)
ps_vol = Visum.Net.PathSets.ItemByKey(2).Paths.GetMultiAttValues("rs_satisfied_demand", True)

no_vehtrips_route_match = 13
veh_capacity = Visum.Net.AttValue("Vehicle_Capacity")
non_integer_vehicle_trips = Visum.Net.AttValue("non_integer_vehicle_trips")

Visum.Net.Matrices.ItemByKey(no_vehtrips_route_match).Init
matrix_values = Visum.Net.Matrices.ItemByKey(no_vehtrips_route_match).GetValues

For i = 0 To UBound(ps_fromzoneno)
    zone_from_index = bin_search(zone_no, ps_fromzoneno(i, 1))
    zone_to_index = bin_search(zone_no, ps_tozoneno(i, 1))
    If non_integer_vehicle_trips = False Then
        matrix_values(zone_from_index, zone_to_index) = ps_vol(i, 1) / veh_capacity
    Else
        matrix_values(zone_from_index, zone_to_index) = 1
    End If
Next

Visum.Net.Matrices.ItemByKey(no_vehtrips_route_match).SetValues matrix_values

' **************************
'binary search algorithm
Function bin_search(arr, srch)
   first = LBound(arr)
   last = UBound(arr)
   Do Until cancel = -1 And first <= last
      middle = (first + last) \ 2
      If arr(middle, 1) = srch Then
         cancel = middle
         Exit Do
      Else
         If srch > arr(middle, 1) Then
            first = middle + 1
         Else
            last = middle - 1
         End If
      End If
   Loop
   bin_search = middle
End Function
```

6.4 Report on Milestone 16 “Assumptions for macroscopic modelling”

Description and context

Work package 2 mainly deals with the development of the microscopic and macroscopic modelling tools to enable them to integrate the characteristics of automated vehicles (AV). These tools are required within the project to model, simulate and evaluate various scenarios and finally to help answer the questions from the partner cities about possible impacts of AV. Furthermore, the extended modelling tools will empower users and researchers to analyse and evaluate their own application cases with AV in the future.

Milestone 16 “Assumptions for the macroscopic modelling tool defined” builds upon examinations with the AV-ready microscopic simulation tool. According to the Grant Agreement, it is defined as follows:

“The result from the validated AV-ready microsimulation model (D2.4) will be a base for creating a first set of assumptions for the supply side of the macroscopic travel demand model by updating network capacities and capacity restraint functions (links and nodes). Means of verification: Report on assumption definitions.”

Hence, Milestone 16 serves as a crucial input for the development of the AV-ready macroscopic modelling tool, namely Task 2.4, which is described as follows:

The result from the validated AV-ready microsimulation model (D2.4) will be a base for creating assumptions for the supply-side of the macroscopic travel demand model by updating network capacities and capacity restraint functions (links and nodes). PTV will provide results from microscopic simulations for USTUTT in form of fundamental diagrams of the traffic flow created for selected networks.

USTUTT uses outputs from the microscopic simulations and the real-world experiments to estimate volume delay functions for links and nodes. In this way, the results are generalised for the application in macroscopic travel demand models. CoEXist will deliver recommendations for capacities, free flow speeds and volume delay functions for links and nodes. At the level of nodes, VISUM provides a method for an intersection capacity analysis (ICA), which computes delay times based on the Highway Capacity Manual (HCM). The parameters of this method (e.g. saturation flow, gap times) will also be adjusted for AVs.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

Methodology on macroscopic side

Definition of Capacity and Performance

The American Highway Capacity Manual (HCM, 2010) defines road capacity as the maximum sustainable hourly flow rate at which persons or vehicles reasonably can be expected to traverse a point or a uniform section of a lane or roadway during a given time period under prevailing roadway, environmental, traffic and control conditions. This definition treats capacity more or less as a constant value. Brilon et al.[1] indicate that this assumption is not appropriate as observations show, that the maximum traffic throughput varies even under constant external conditions. They introduce the concept of stochastic capacities to replicate the relationship between traffic flows and traffic breakdown in a better way. Lohmiller[2] shows that the throughput on a motorway depends on the traffic composition, i.e. the driver population influences the quality of the traffic flow. This leads to two interpretations for the relationship between demand, capacity and performance (=travel time):

- Variable (or stochastic) capacity: The performance depends on random capacity values.
- Variable demand composition: The performance depends on the ability of a given demand composition (driver / vehicle population) to use a given (constant) capacity
  → “better drivers use the capacity more efficiently”.

This specification uses the following terms:

- *Performance* is a measure to quantify the functioning of a road facility. Performance can be measured by the indicator delay time per vehicle.
- *Capacity* is a constant value describing the typical throughput of vehicles for a reference vehicle type. Conventional passenger cars define the reference vehicle type. Thus, a capacity value describes the throughput of a demand with 100% conventional passenger cars and 0% trucks.
- *Traffic composition* defines a certain combination of vehicle types (conventional and automated vehicles, passenger cars and trucks).

[1] Brilon, W., Geistefeldt, J., & Zurlinden, H. (2007). Implementing the concept of reliability for highway capacity analysis. *Transportation Research Record: Journal of the Transportation Research Board*, (2027), 1-8.
[2] Lohmiller, J. (2014). Qualität des Verkehrsablaufs auf Netzabschnitten von Autobahnen: Bewertung unter Berücksichtigung der Zuverlässigkeit und Analyse von Einflussfaktoren.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

Capacity and demand in passenger car units

Specification

Macroscopic route choice and assignment models for private transport apply volume-delay functions (VDF) to determine travel time in the road network. For links, the travel time is computed by multiplying the free flow travel time by a factor that is determined by a VDF as shown in equation (1). For nodes, a delay time is added to the free flow travel time as shown in equation (2). Equation (3) shows a simple example for a VDF suitable for links. The VDF-factor depends on the volume / capacity ratio, i.e. the saturation rate x_s of a network element. The relationship between volume and capacity is described in equation (4). It assumes a constant capacity and a volume, which considers the vehicle composition. The vehicle composition is described by passenger car units (PCU). Each vehicle type has a specific PCU-factor converting the vehicle volume into a volume equivalent to passenger cars. This concept of PCU is a common concept in macroscopic assignment models. It is mainly used to convert trucks into PCU. This specification suggests extending the PCU concept to AV. This extension can come in two forms making different assumptions:

Assumption 1: The performance of a network element changes proportionally to the share of AV

This assumption implies a linear relationship between vehicle-type and performance. The impact of one single AV depends on the type of road facility, but not on the penetration rate.

PCU-factor: The PCU-factor must be extended to distinguish road and intersection types (motorway / urban road, grade separated / at-grade intersections, signalised / unsignalised intersections). The PCU-factor is a constant value, which remains fixed during an assignment.

Assumption 2: The performance of a network element changes disproportionately to the share of AV

This assumption includes a nonlinear relationship between vehicle-type and performance. The impact of one single AV depends on the type of road facility and on the penetration rate. In case of a low penetration rate the impact of a single AV is smaller than in cases with a higher one.

PCU-factor: The PCU-factor must be extended to distinguish road and intersection types, but also the share of AV. This leads to a dynamic PCU-factor, which is adapted during an assignment. Equation (5) shows a possible function for a dynamic PCU-factor.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

$$t_{s=Link}\left(x_s\right)=t_s^{free}\cdot VDF\left(x_s\right) \tag{1}$$

$$t_{s=Node}\left(x_s\right)=t_s^{free}+VDF\left(x_s\right) \tag{2}$$

$$VDF\left(x_s\right)=1+\alpha\cdot x_s^{\beta} \tag{3}$$

$$x_s=\frac{\sum_{i\in VehType} q_{s,i}\cdot f}{q_s^{\max}} \qquad \text{where} \begin{cases} f = f_i^{PCU} & \text{current approach} \\ f = f_{s,i}^{PCU} & \text{linear impact of AV} \\ f = f_{s,i}^{PCU}(n) & \text{nonlinear impact AV} \end{cases} \tag{4}$$

$$f_{s,i=AV}^{PCU}(p_{AV}) = f_{s,i=AV}^{PCU,Max} - p_{AV}\cdot\left(f_{s,i=AV}^{PCU,Max} - f_{s,i=AV}^{PCU,Min}\right) \tag{5}$$

where

f_i^{PCU}	passenger car units of vehicle type i [PCU/veh]
$f_{s,i}^{PCU}$	passenger car units of vehicle type i on supply element type s [PCU/veh]
$f_{s,i}^{PCU}(p_{AV})$	passenger car unit function dependent on the share of AV p_{AV} [PCU/veh]
$f_{s,i=AV}^{PCU,Max}$	passenger car units of vehicle type AV on supply element type s for an AV-share of 0% (e.g. = 1.0)
$f_{s,i=AV}^{PCU,Min}$	passenger car units of vehicle type AV on supply element type s for an AV-share of 100% (e.g. = 0.7)
$q_{s,i}$	volume of vehicle type i on supply element s [veh/h]
$q_s^{\max}$	capacity of supply element s [PCU/h]
$t_s\left(x_s\right)$	travel time on supply element s at saturation rate x_s [sec]
t_s^{free}	travel time on supply element s at saturation rate $x_s=0$ [sec]
$VehType$	vehicle types: conventional car, AV, HGV
$VDF\left(x_s\right)$	volume-delay function
x_s	saturation rate (volume/capacity ratio) on supply element s [-]

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Application example

The following figure shows the impact of linear and nonlinear PCU-factors on travel time for the same penetration rates of AV. In the nonlinear case, AV have a smaller impact at all penetration rates below 100% compared to the linear case.

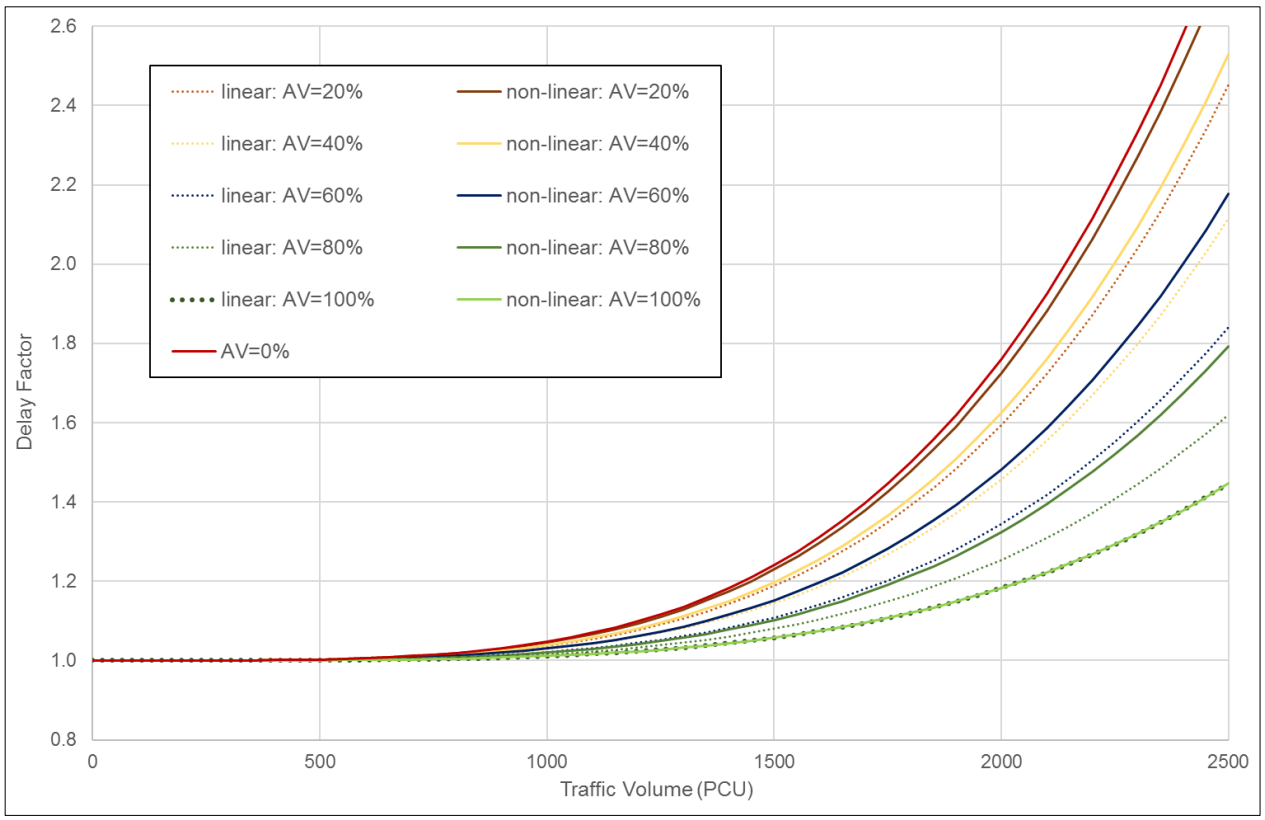


Figure 10: Impact of linear and non-linear PCU-factors on travel time for the same penetration rates of AV

Assumptions for this example:

Linear case: $f_{s,i=AV}^{PCU} = 0.7$

Nonlinear case:

$$f_{s,i=AV}^{PCU}(p_{AV}) = 1.0 - p_{AV} \cdot (1.0 - 0.7)$$

$$f_{s,i=AV}^{PCU}(p_{AV}) \in [0.7, 1.0]$$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Capacity and demand in vehicle units

Specification

Wagner (2016, 2017)[3] introduced an approach to incorporate effects of AV on performance by adapting the capacities of road facilities. It determines capacity depending on vehicle headways, vehicle lengths, share of AV and speed.

Equation (9) is applied to calculate the mean net time headway over all vehicles. It is simply based on the probability of specific vehicle types succeeding others, which in turn depends on the share of this vehicle type (e.g. 80% car, 20% HDV) and the penetration rate of AV of this type (e.g. 60% of cars are AV, 70% of HDV are AV). Then the mean gross space headway required by an average vehicle is determined from the net time headway, the speed and the mean vehicle length (see equation (10)) of the vehicle composition on the considered supply element. This leads to the vehicle density shown in equation (8). Multiplying the vehicle density with the speed determines the capacity as shown in equation (7), leading to the saturation in equation (6), which serves as an input for volume delay functions. Since the capacity depends on the share of AV, it must be updated during an assignment.

The (net) time headways between vehicle types have to be derived from the microscopic traffic flow simulations.

$$x_s = \frac{q_s}{q_s^{\max}} \tag{6}$$

$$q_s^{\max} = 3600\, v_s\, k_s(v_s) \tag{7}$$

$$k_s(v_s) = \frac{1}{v_s\, t_s^{mean}(p_{s,AV}) + l_s^{mean}} \tag{8}$$

$$\begin{aligned} t_s^{mean} = \sum_i \sum_j p_{s,i}\, p_{s,j} \big(& (1 - p_{s,AV,i})(1 - p_{s,AV,j})\, t_{ij,CC} \\ & + (1 - p_{s,AV,i})\, p_{s,AV,j}\, t_{ij,CA} \\ & + p_{s,AV,i}(1 - p_{s,AV,j})\, t_{ij,AC} \\ & + p_{s,AV,i}\, p_{s,AV,j}\, t_{ij,AA} \big) \qquad [i, j \in VehType] \end{aligned} \tag{9}$$

$$l_s^{mean} = \frac{\sum_i q_{s,i}\, l_i}{\sum_i q_{s,i}} \qquad [i \in VehType] \tag{10}$$

[3] Wagner, P., 2016. Traffic Control and Traffic Management in a Transportation System with Autonomous Vehicles. In Maurer, M., Gerdes, J., Lenz, B., Winner, H. (Eds.) *Autonomous Driving: Technical, Legal and Social Aspects, Springer, 2016, 301-316*
Wagner, P., 2017. Autonomer Verkehr und die Kapazität von Straßen. *Automatisiertes Fahren, FSV Schriftenreihe 017 | 2017, 23-26*

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

where

x_s	saturation of supply element s [-]
q_s	traffic volume on supply element s [veh/h]
$q_s^{\max}$	capacity of supply element s [veh/h]
v_s	speed limit on supply element s [m/sec]
$k_s(v_s)$	traffic density for speed v_s on supply element s [veh/m]
t_s^{mean}	average net time headway between vehicles on supply element s [sec]
l_s^{mean}	mean vehicle length on supply element s [m]
$p_{s,i}$	share of vehicle type i on supply element s [-]
$p_{s,AV,i}$	AV share of vehicle type i on supply element s [-]
$t_{ij,CC}$	net time headway between CV of type i and CV of type j [sec]
$t_{ij,CA}$	net time headway between CV of type i and AV of type j [sec]
$t_{ij,AC}$	net time headway between AV of type i and CV of type j [sec]
$t_{ij,AA}$	net time headway between AV of type i and AV of type j [sec]

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Application example

Figure 11 shows the correlation between the share of automated vehicles, the permitted speed and the resulting capacity for a road section with one lane. With the assumptions of decreased time headways between two AV in comparison to the usual gap between conventional vehicles, the capacity increases with higher shares of AV. It also grows with higher speed, although density decreases simultaneously. This can be explained with the average vehicle length, which reduces the density to such an extent, that the speed always predominates its reciprocal within the formula for vehicle density (see equations (7) and (8)). For a vehicle length of zero, capacity does not depend on speed anymore.

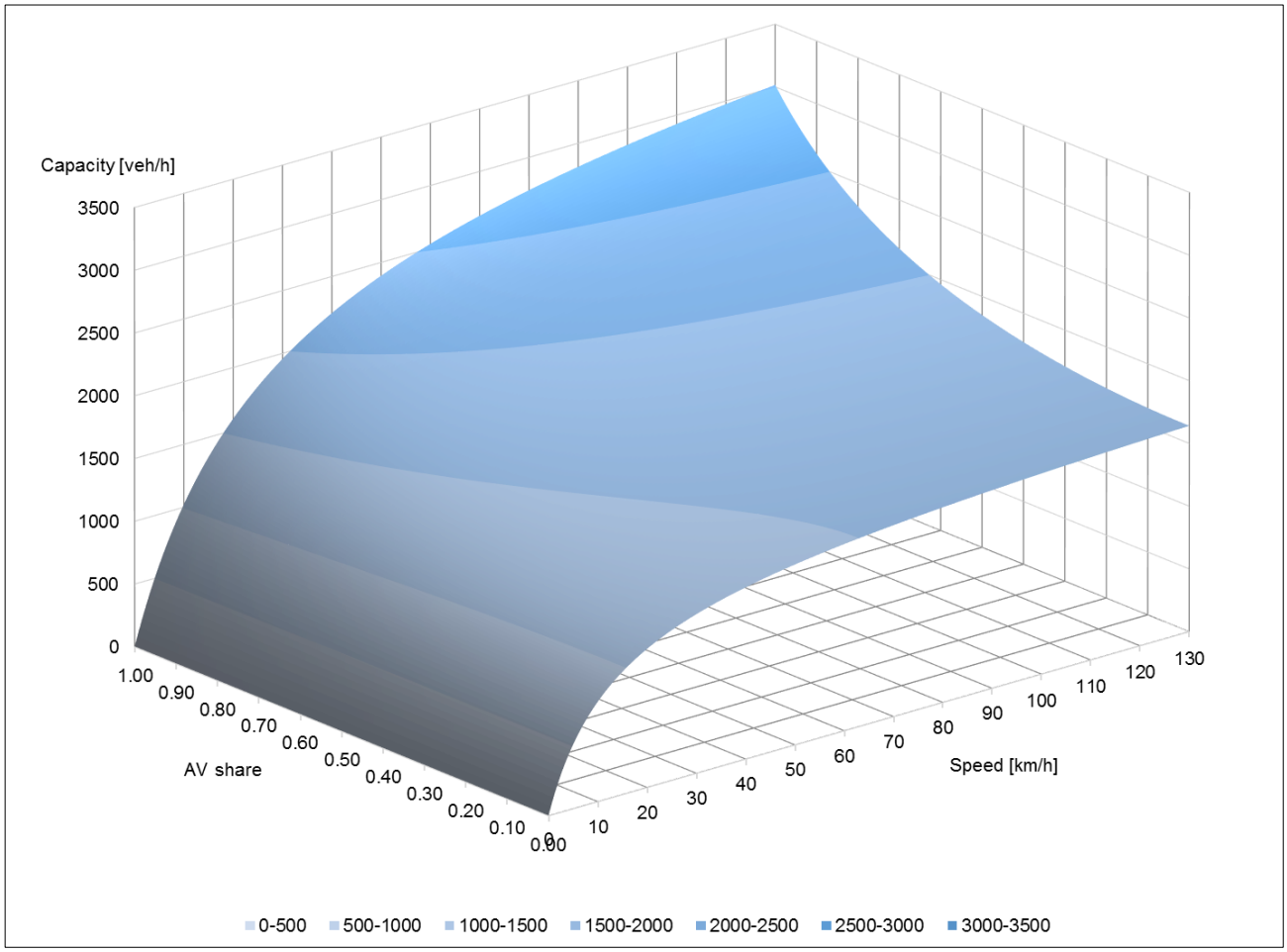


Figure 11: Capacity depending on AV share and speed for a one-lane road section

Assumptions for this example:

- 100% cars → one vehicle type
- Net time headways: $t_{AC} = 1s$; $t_{CC} = t_{CA} = t_{AC} = t_{other} = 2s$
- Equation (9) simplifies to: $t_s^{mean}(p_{s,AV}) = {p_{s,AV}}^2 t_{AA} + (1 - {p_{s,AV}}^2) t_{other}$
- Average vehicle length including standstill distance: $l_s^{mean} = 7m$

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Methodology on microscopic side

Assumptions on driving behaviour

For the simulations we used three different driving behaviour types:

- Cautious (AV1), the vehicle respects the road-code and always adopts a safe behaviour. The brick wall distance is always respected, unsignalized intersections and lane changes are possible but the vehicle will keep quite large gaps.
- Normal (AV2), very similar to a human driver with the additional capacity of measuring distances and speeds of the surrounding vehicles thanks to its sensor suite.
- All knowing (AV3), with perfect perception and prediction leading mainly to smaller gaps for all manoeuvres and situations. A kind of cooperative behaviour is expected.

The driving logics have been defined within the CoEXist project, for additional details see the driving logic paper. The “rail safe” driving logic mentioned in deliverable D2.3 was not tested, because of very special nature and very special use (mostly closed environments).

Table 5 Settings used for new features

	Used setting for new features			
driving logic	enforce absolute breaking distance (EABK)	use implicit stochastics	number of interaction vehicles (interaction objects)	increased desired acceleration
Cautious (AV1)	ON	OFF	1 (2)	100 %
Normal (AV2)	OFF	OFF	1 (2)	105 %
all knowing (AV3)	OFF	OFF	8 (10)	110 %
conventional	OFF	ON	99 (2)	100 %

For conventional vehicles we used the values defined by research project at KIT for standard German network components (like merging or diverging areas) which where calibrated to be in line with the German highway capacity manual (HBS). Driving parameter values for automated vehicles are based on CoEXist deliverable D2.3.

Following behaviour

These driving behaviour parameters have been used for the following behaviour. Chosen values are based on the data evaluation results from the test-track, results from the co-simulations and assumptions.

Table 6 Settings for following behaviour parameters for AV’s

			driving logic		
	model	Parameter	**cautious**	**normal**	**all knowing**
following	Wiedemann 99	CC0 – Standstill distance (m)	1.5	1.5	1
		CC1 – Spacing time (s)	1.5	0.9	0.7
		CC2 – Following variation (m)	0	0	0

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

		CC3 – Threshold for entering "following" (s)	-10	-8	-6
		CC4 – Negative „following" threshold (m/s)	-0.1	-0.1	-0.1
		CC5 – Positive „following" threshold (m/s)	0.1	0.1	0.1
		CC6 – Speed dependency of oscillation (10^{-4} rad/s)	0	0	0
		CC7 – Oscillation acceleration (m/s^2)	0.1	0.1	0.1
		CC8 – Standstill acceleration (m/s^2)	3	3.5	4
		CC9 – Acceleration at 80 km/h (m/s^2)	1.2	1.5	2

Lane changing behaviour

The appropriate values for lane change parameters for three driving logics have been set based on assumptions about the automated vehicles because of lack of empirical data (only COEXist data for following behaviour are available). These driving behaviour parameters have been used for lane changing behaviour:

Table 7 Used lane change behaviour parameters for AV's

	driving logic					
	cautious		**normal**		**all knowing**	
parameter for necessary lane change	own	trailing vehicle	own	trailing vehicle	own	trailing vehicle
maximum deceleration	-3.5	-2.5	-4	-3	-4	-4
- 1 m/s per distance	80	80	100	100	100	100
accepted deceleration	-1	-1	-1	-1	-1	-1.5

Table 8 Used settings for lane changing behaviour functionalities

	driving logic		
behavioral functionality	**cautious**	**normal**	**all knowing**
Advanced merging	on	on	on
Cooperative lane change	off	on	on
Safety distance reduction factor	1	0.6	0.75
min. headway (front/rear)	1	0.5	0.5
max. deceleration for cooperative braking	-2.5	-3	-6

Besides parameter mentioned above in some networks with multilane links the lane change is prohibited in certain direction (to left or right lane) in order to be in lane with real driving behaviour and the German highway capacity manual. For details, open the network and have a look into link dialog, tab. "Lanes".

Lateral behaviour

No changes have been realized in lateral behaviour (within the lane), basic Vissim's settings is used (position "middle of lane").

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Signal control behaviour

Table 9 Used settings for signal control behaviour

	driving logic		
attribute	**cautious**	**normal**	**all knowing**
behavior at amber signal	continuous check	one decision	one decision
behavior at red/amber signal	stop	stop	stop
reaction time distribution	-	-	-
reduced safety distance factor	1	1	1
reduced safety start upstream of stop line	100	100	100
reduced safety end upstream of stop line	100	100	100

Investigated network sections

All networks are equipped with sets of data collection points to measure the number of vehicles going through during a specific time interval. Data collection points are placed everywhere where an individual value or a flow change is possible. Results have been collected as raw data (*.mer files) and evaluated and aggregated using python scripts after the simulation.

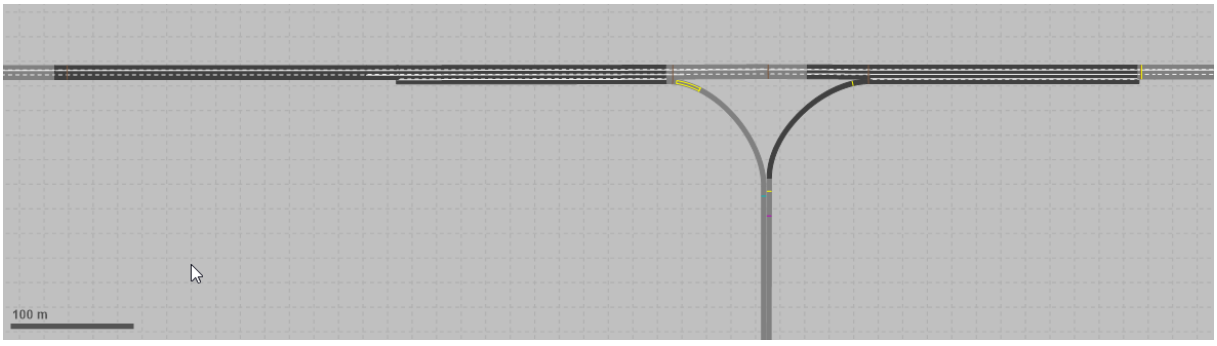
Desired speed of vehicles entering the network is set as a distribution, which has a larger span for conventional vehicles and very small span (+- 2 km/h) for automated vehicle. It is assumed, that the automated vehicles control the speed in a deterministic way and obey the speed limits.

“Lane change“ attributes of connectors are identical for all vehicles. That means, that all vehicle classes get the information about necessary lane change further downstream at the same location (cross section). From that point, vehicles start to try to make the lane change if needed because of the route (in order to reach the upstream connector).

These eight networks have been used for simulations:

1) Off ramp & on ramp with 3 lanes on the main freeway

   Desired speed of conventional vehicles entering the network on the main link is set to 140 km/m with a wider spread between 80 and 205 km/h. Automated vehicles use the desired speed 130 +-2 km/h.



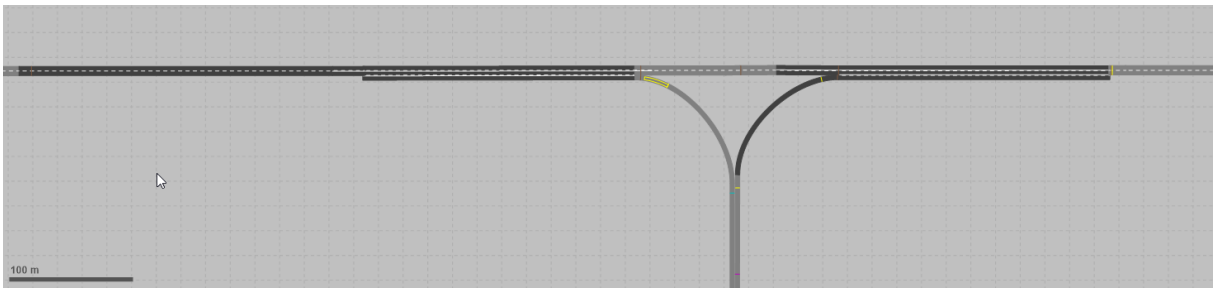
Relative share of demand between network elements:	
main link through	4

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

off ramp	1
on ramp	1

2) Off ramp & on ramp with 2 lanes on the main freeway

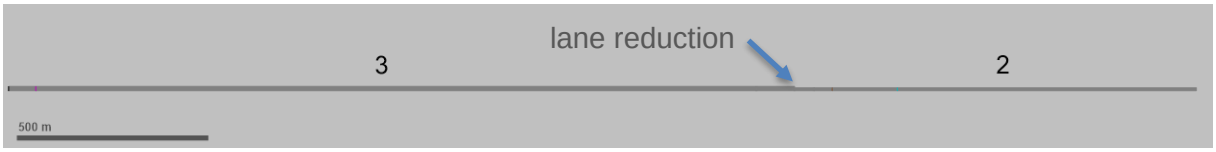
Desired speed of conventional vehicles entering the network on the main link is set to 140 km/m with a wider spread between 80 and 205 km/h. Automated vehicles use the desired speed 130 +-2 km/h.



Relative share of demand between network elements:	
main link through	4
off ramp	1
on ramp	1

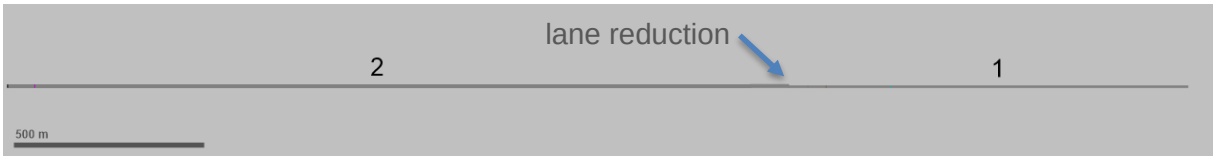
3) Lane number reduction from 3 to 2 lane

A three-lane link is reduced to 2-lane link, maximum flow is measured.



4) Lane number reduction from 2 to 1 lanes

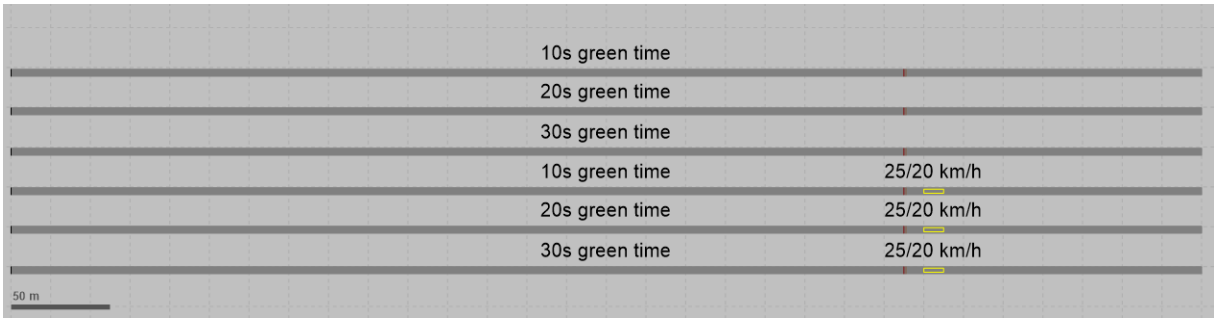
A two-lane link is reduced to 1-lane link, maximum flow is measured.



5) Saturation flow on one-lane link with signal

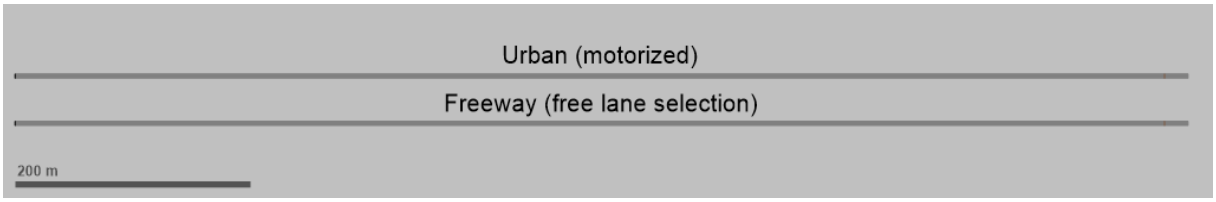
Results provide information for the calculation of the saturation flow. The green time used in the model was set to 10, 20 or 30 seconds. First three links reflects the situation for direction “through“, without a speed limitation. Second three links reflect the situation when the speed of vehicles is limited e.g. through curvature when turning left or right.

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201



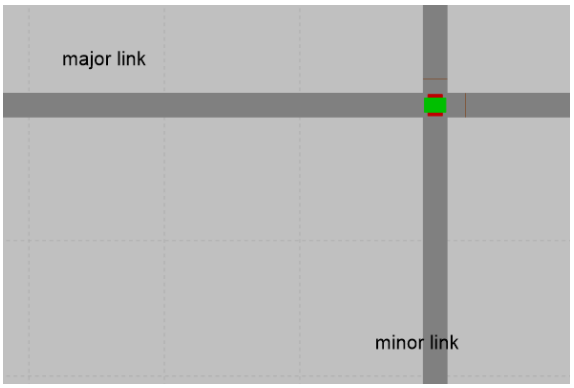
6) Simple one-lane link

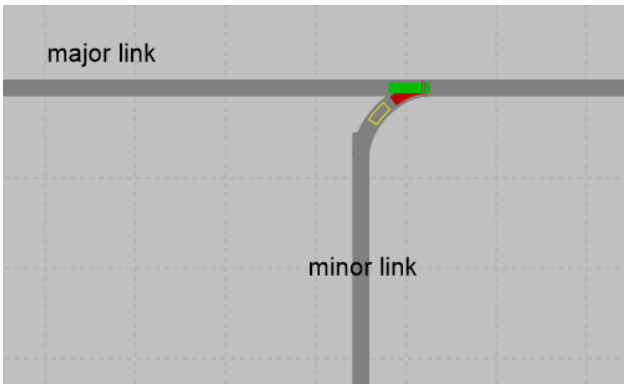
This network provides results for theoretical capacity on one lane link under ideal conditions without influence of intersections, parking manoeuvres or other sources of disturbance. The resulting maximum flow depends on speed and settings for following behaviour.



7) Simple crossing and simple merging conflict

Results from this network provide information about the capacity of a simple conflict area for relative comparison of different driving logics and penetration rates.

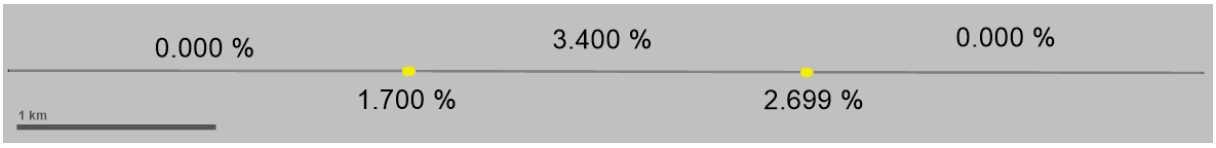




8) Influence of gradient (uphill slopes)

Results from this network are showing the impact of gradient. The value of gradient impacts the driving behaviour in Vissim via the maximum acceleration and maximum deceleration on a link:

- by -0.1 m/s² per gradient percent incline. The maximum accelerating power decreases when the deceleration power increases.
- by 0.1 m/s² per gradient percent downgrade. The accelerating power increases when the deceleration power decreases.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Methodology

For the described networks, several different scenarios were simulated. In the scenarios, we use the different AV driving logics as described in chapter 3.1 and additionally varied the penetration rate and the demand. The penetration varied in 10% steps from 0% to 100% and for varying the demand, we applied a factor between 0.1 and 3.6 in 0.5 steps. See the different values in the next table.

All possible combination between network, AV driving logic, penetration rate and demand factor were simulated 10 times with different random seeds, making a total of 8 * 3 * 11 * 7 * 10 = 18.480 simulation runs. Note: A combination of different AV driving logics, e.g. 20% cautious and 80 % all-knowing, has not been simulated. In each simulation we use one specific AV driving logic with a specific penetration rate, e.g. 20 % cautious, and the other 80 % are always conventional vehicles. The demand factor was applied to all vehicle inputs in the network to get results with different volumes, for example free-flow, saturated and over-saturated conditions. The base demand in the network was set in a way that a level of service C/D was achieved, which means an undersaturated condition.

Network	AV driving logic	Penetration rate	Demand factor
Off ramp & on ramp with 3 lanes on the main freeway	cautious	0 %	0.1
Off ramp & on ramp with 2 lanes on the main freeway	normal	10 %	0.6
Lane number reduction from 3 to 2 lane	all knowing	20 %	1.1
Lane number reduction from 2 to 1 lanes		30 %	1.6
Saturation flow on one-lane link with signal		40 %	2.1
Simple one-lane link		50 %	2.6
Simple crossing and simple merging conflict		60 %	3.1
Influence of gradient (uphill slopes)		70 %	
		80 %	
		90 %	
		100 %	

From the simulation we collected traffic counts and average speeds at cross sections. We collected the results over 30 minutes of simulation time after a 5-minute warmup period and aggregated the results in

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

5 minutes time intervals. With this data we describe the fundamental diagram which is expressed by the functional relation by van Aerde[4]. With this functional expression of the fundamental diagram, we can compare different scenarios, mainly the different penetration rate, to each other. The functional relations is expressed in the density – speed relation:

$$k(v) = \frac{1}{c_1 + \frac{c_2}{v_0 - v} + c_3 \cdot v}$$

with

k density [veh/km]

v speed [km/h]

v_0 free-flow speed [km/h]

c_1 headway parameter [km/veh]

c_2 parameter for difference to free-flow speed [km²/(h*veh)]

c_3 speed parameter[1/(h*veh)]

One van Aerde parameter set was determined for a network – AV driving logic – penetration rate combination. All different demand scenarios plus the results from the different random seeds where used for the parameter estimation of one van Aerde function. The density was calculated by k = q/v, where k is the density [veh/km], q is the volume [veh/h] and v is the speed [km/h].

To obtain a reasonable parameter estimation, we averaged speeds within different density intervals of 2 veh/km length. This result to only one datapoint for every 2 veh/km density intervals. Only with those data points we estimated the parameters of the van Aerde function. Additionally, a datapoint was added at speed v = 0 km/h and density k = number of lanes * 125 veh/km – this represents one vehicle every 8 m, because there were no scenarios where we had maximum density (at speed 0 km/h).

Results

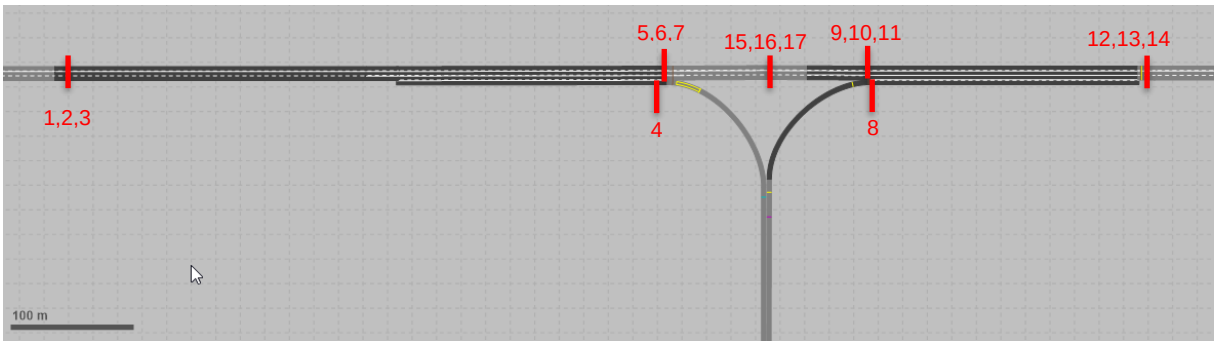
Important note: curve fitting and van Aerde diagrams are provided for almost all networks and data collection points for an overview, but only few of them are relevant for further consideration – these should be checked on curve fit quality and adjusted manually if needed.

1) Off ramp & on ramp with 3 lanes on the main freeway (A1-n3 SuP plus KAL_E1-n3_94789)

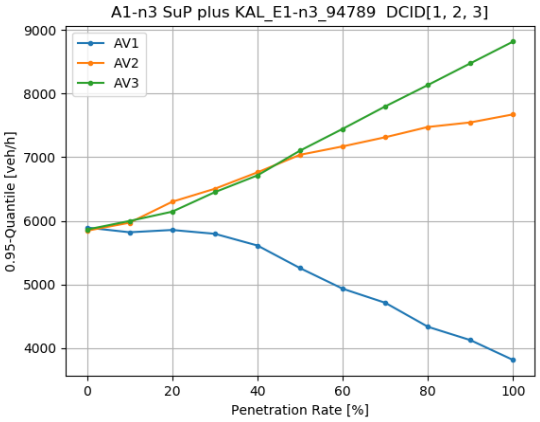
   Desired speed of conventional vehicles entering the network on the main link is set to 140 km/m with a wider spread between 80 and 205 km/h. Automated vehicles use the desired speed 130 +-2 km/h.

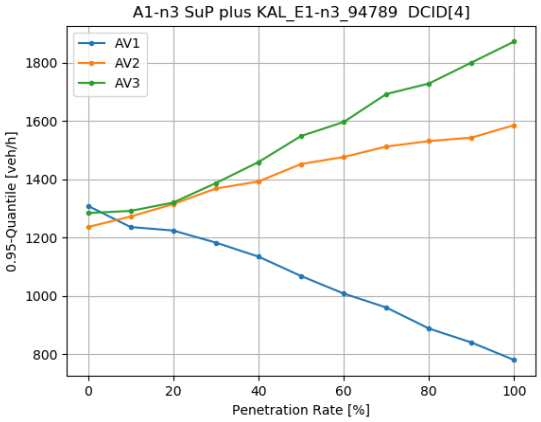
[4] VAN AERDE, M. (1995): A single regime speed-flow-density relationship for freeways and arterials, Proceedings of the 74th TRB annual meeting. Washington D. C..

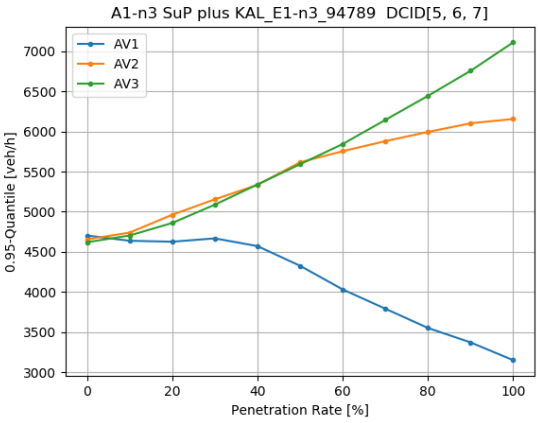
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

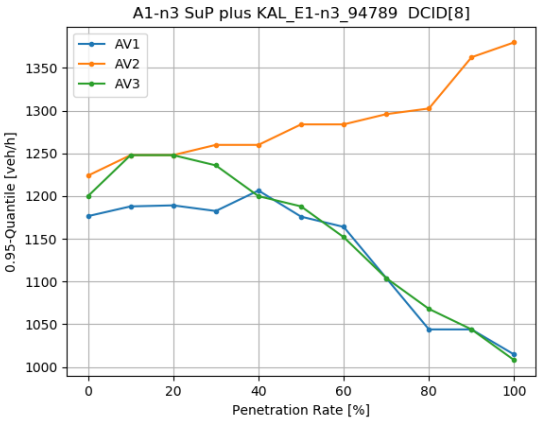


Pictures showing 95% quantile results for data collection groups:

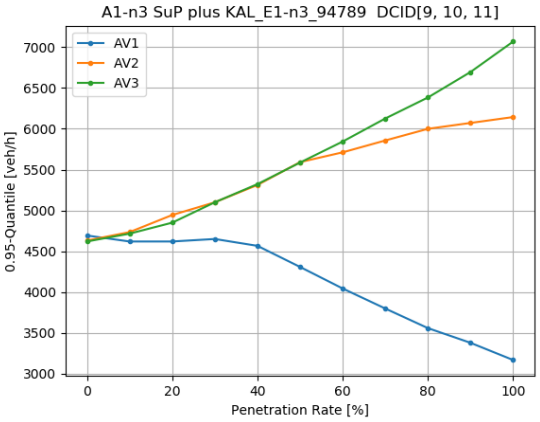


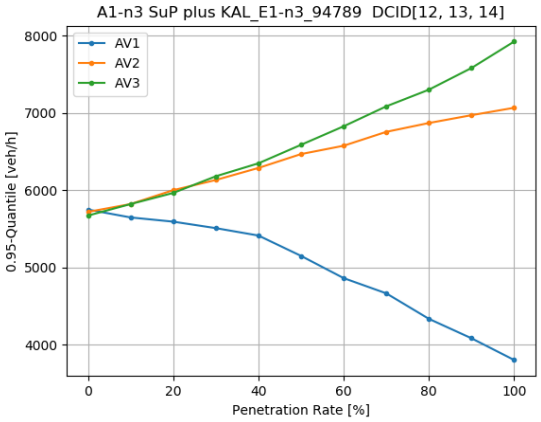


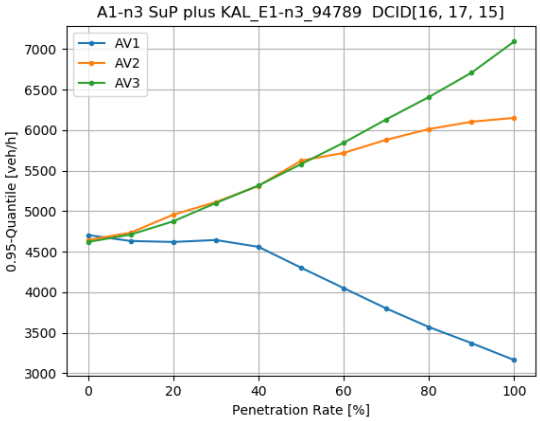




This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201







,

Table 10 Comments to 95%-Quantile results

DCID	Comments
general	The small difference between AV2 and AV3 in lower penetration rates is caused by almost the same safety distance for lane change. The lane change behaviour is important in this network because of diverging and merging area. Reduced safety factor for AV2 is set to 0,6 and for AV3 to 0,75 (0,6 x 0,9 vs. 0,75 x 0,7 = 0,54 vs 0,525 for the speed dependent part of the safety distance). Lower values for safety reduction factor for AV3 could lead to crashes in the simulation (vehicle overlaps) with existing following and lane changing model in Vissim.
[1,2,3]	Before the diverging area AV1 (cautious) shows decreasing throughput with growing penetration rate, especially from 30 %. AV2 and AV3 show the practically the same growing values up to penetration rate 50 %. Above the penetration rate 50 %, the AV3 values grow faster than AV2.
[4]	Deceleration lane at the end of the diverging area AV1 (cautious) shows decreasing throughput with growing penetration rate. AV2 and AV3 show the practically the same growing values up to penetration rate 30 %. Above the penetration rate 30 %, the AV3 values grow faster than AV2.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

[5,6,7]	Through lanes at the end of the diverging area AV1 (cautious) shows decreasing throughput with growing penetration rate, especially from 40 %. AV2 and AV3 show the practically the same growing values up to penetration rate 50 %. Above the penetration rate 50 %, the AV3 values grow faster than AV2.
[8]	Merging lane at the beginning of the merging area This data collection point is placed on the on-ramp in front of the merging area. AV2 seems to be the winner in sense of merging throughput and indicates growing values with growing penetration rate. AV1 and AV3 values decrease with penetration rate. The AV3 allows lower throughput values as AV2 for the merging flow, but higher values for the main flow and also for the sum of main flow and merging flow (see DCID [9,10,11] and [12,13,14] comments).
[9,10,11]	Main flow lanes at the beginning of the merging area AV1 (cautious) shows decreasing throughput with growing penetration rate, especially from 40 %. AV2 and AV3 show the practically the same growing values up to penetration rate 50 %. Above the penetration rate 50 %, the AV3 values grow faster than AV2.
[12,13,14]	Flow after the merging area AV1 (cautious) shows decreasing throughput with growing penetration rate. AV2 and AV3 show the practically the same growing values up to penetration rate 40 %. Above the penetration rate 50 %, the AV3 values grow faster than AV2.
[15,16,17]	Flow between the diverging and merging area AV1 (cautious) shows decreasing throughput with growing penetration rate, especially from 40 %. AV2 and AV3 show the practically the same growing values up to penetration rate 50 %. Above the penetration rate 50 %, the AV3 values grow faster than AV2.

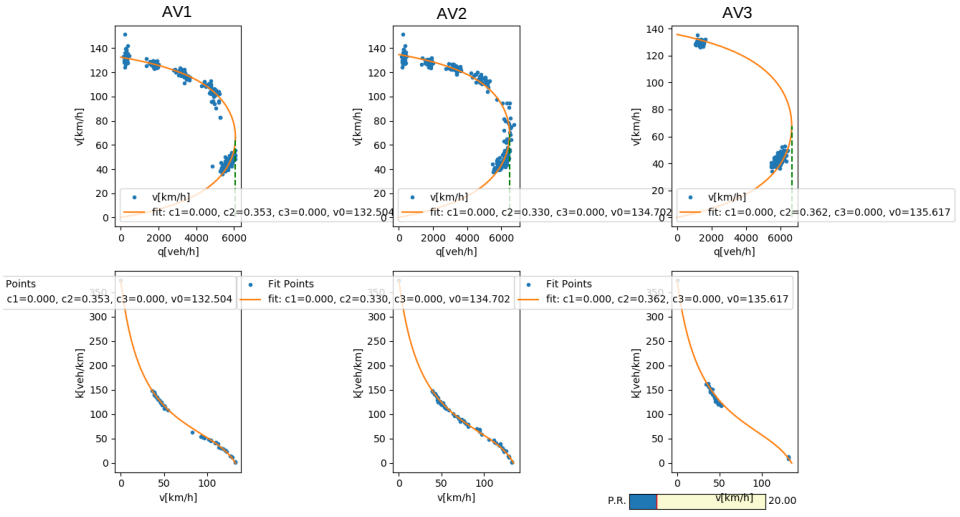


Figure 12 Curve-fitting example: penetration rate 20%, DCID [1,2,3]. See the attachment for other penetration rates and data collection points.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

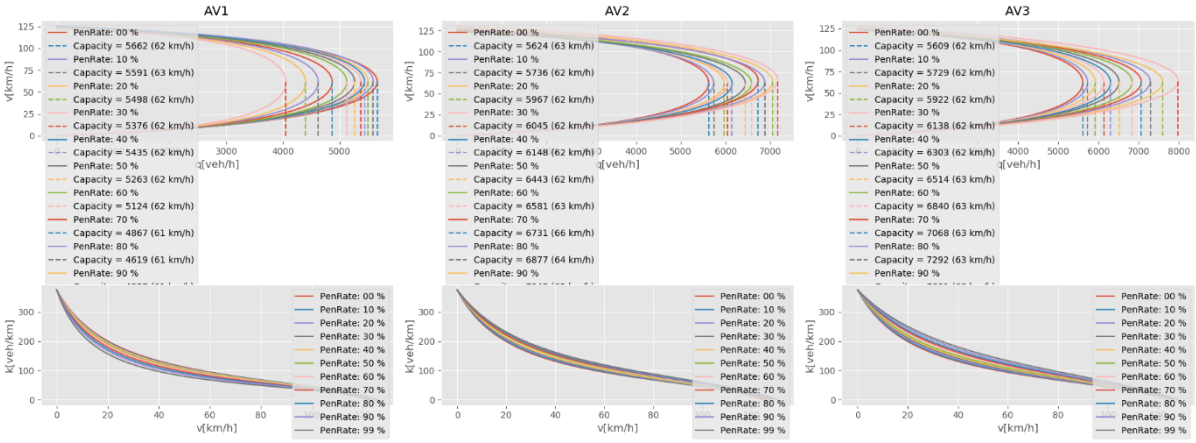
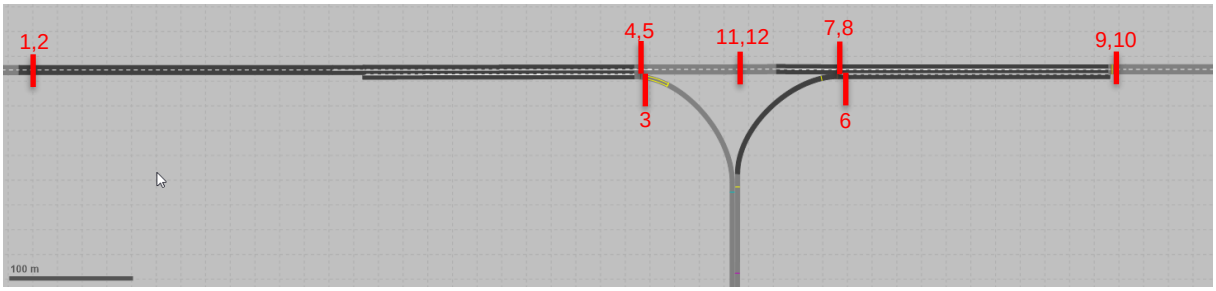


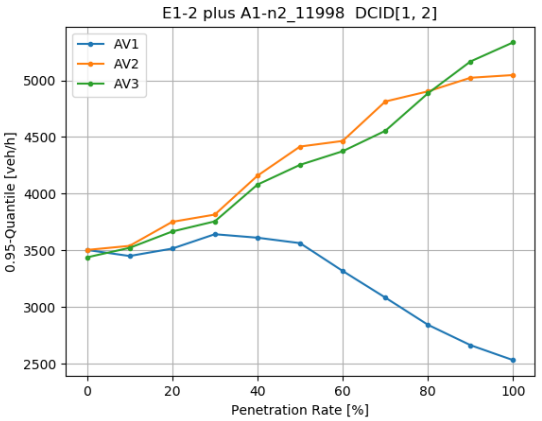
Figure 13 Van Aerde diagrams for DCID [12,13,14], all penetration rates. See the attachments for other DCIDs.

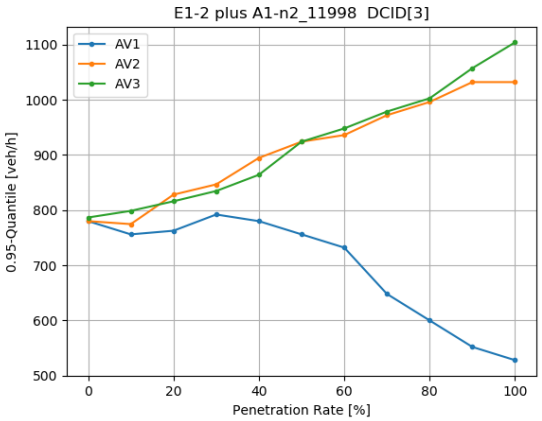
2) Off ramp & on ramp with 2 lanes on the main freeway (E1-2 plus A1-n2_11998.inpx)

Desired speed of conventional vehicles entering the network on the main link is set to 140 km/m with a wider spread between 80 and 205 km/h. Automated vehicles use the desired speed 130 +-2 km/h.

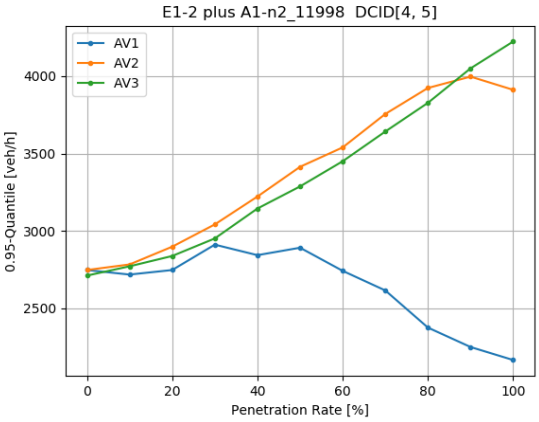


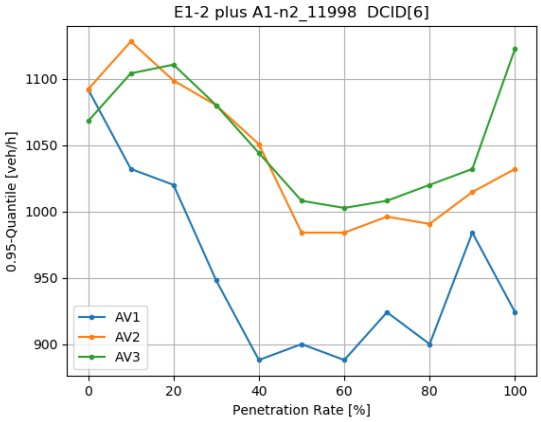
Pictures showing 95% quantile results for data collection groups:

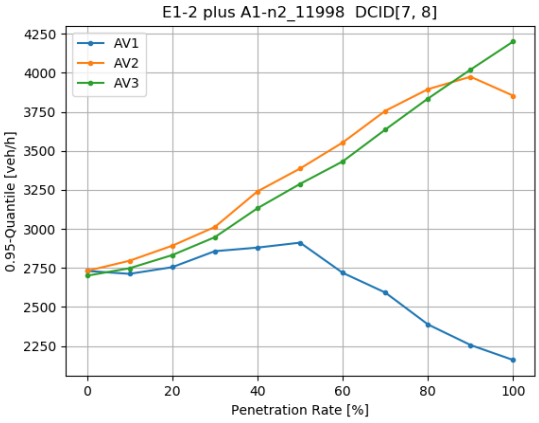


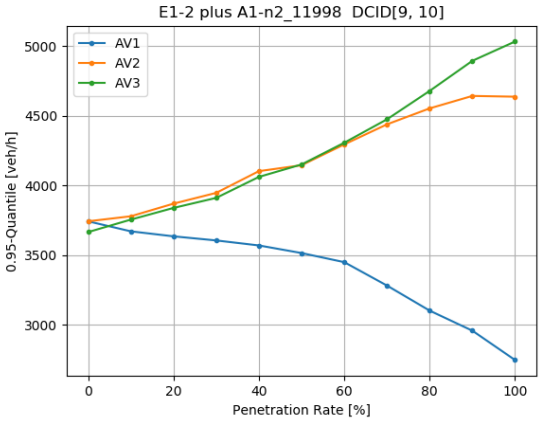


This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201









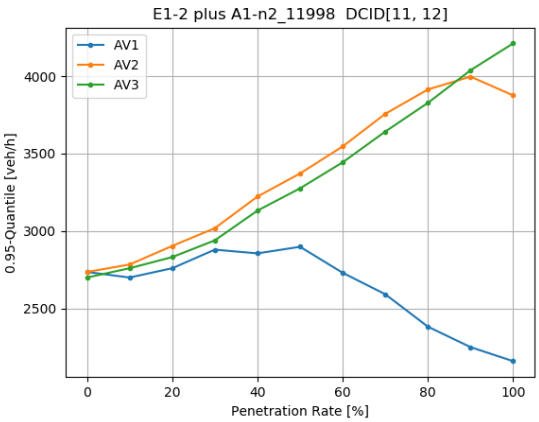


Table 11 Comments to 95%-Quantile results

DCID	Comments
General	The similarity between AV2 and is AV3 is correct because the network has merging and diverging areas = lane changes are necessary, there are only 2 through lanes and the safety distance for lane changing is almost the same for both driving behaviors. Better performance of AV3 would be possible only with some external control algorithm leading to perfect cooperation (such scenario is more realistic for full penetration rate).

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

[1,2]	Main flow (through lanes) before the diverging area. The throughput values for cautious vehicles with penetration rate 0-50 % oscillates around the same value and later declines with growing penetration rate. AV2 and AV3 show growth with growing penetration rate and very similar values. The higher values of AV3 from previous network with 3 through-lanes are not present here, because the disturbances in the diverging and merging area have a strong impact on both lanes.
[3]	Off-ramp lane at the end of the diverging area The throughput values for cautious vehicles with penetration rate 0-50 % oscillates around the same value and later declines with growing penetration rate. AV2 and AV3 show growth with growing penetration rate and very similar values.
[4,5]	Through lanes at the end of the diverging area The throughput values for cautious vehicles with penetration rate 0-50 % oscillates around the same value and later declines with growing penetration rate. AV2 and AV3 show growth with growing penetration rate and very similar values.
[6]	Merging lane at the beginning of the merging area The merging flow declines with growing penetration rates up to 50 %. Higher penetration rates show growing throughput. AV2 and AV3 values are very similar except of full penetration, where AV3 leads.
[7,8]	Through lanes at the beginning of the merging area The throughput values for cautious vehicles with penetration rate 0-50 % grows slightly and later (pen. rate > 50 %) declines significantly with growing penetration rate. AV2 and AV3 show growth with growing penetration rate and very similar values.
[9,10]	Through lanes after the merging area The throughput values for cautious vehicles decline with growing penetration rate. AV2 and AV3 show growth with growing penetration rate and very similar values, except the highest penetration rates, where the AV3 shows higher values.
[11,12]	Through lane between the diverging and merging area The throughput values for cautious vehicles with penetration rate 0-50 % grows slightly and later (pen. rate > 50 %) declines significantly with growing penetration rate. AV2 and AV3 show growth with growing penetration rate and very similar values except of full penetration, where AV3 shows higher value.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

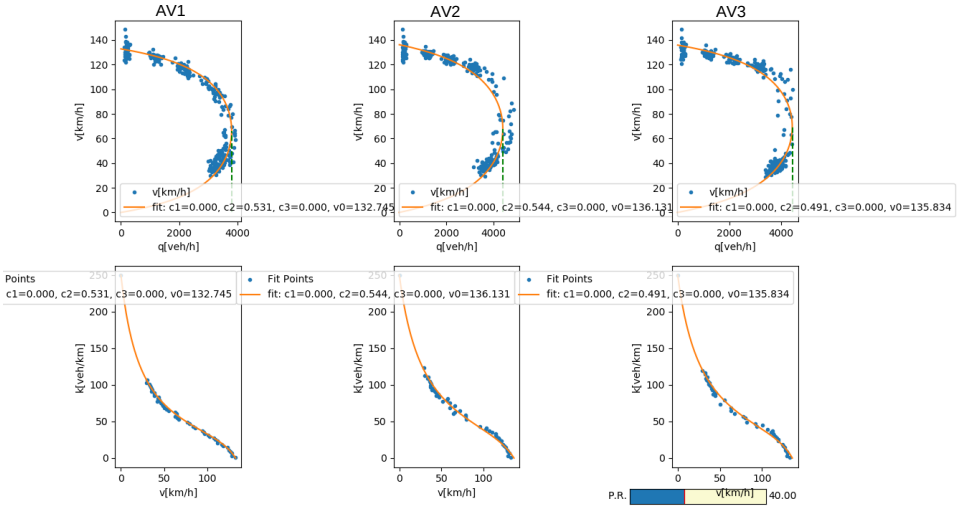


Figure 14 Curve-fitting example: penetration rate 40%, DCID [1,2]. See the attachment for other penetration rates and data collection points.

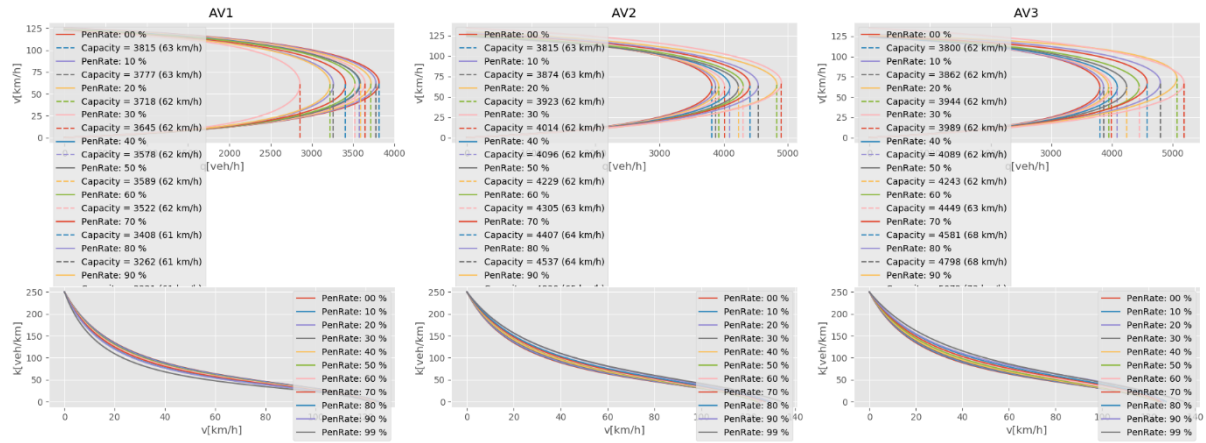
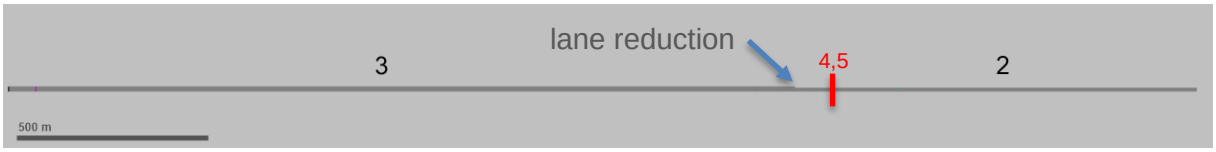


Figure 15 Van Aerde diagrams for DCID [9,10], all penetration rates. See the attachments for other DCIDs.

3) Lane number reduction from 3 to 2 lane (FSR-n2_4234.inpx)

A three-lane link is reduced to 2-lane link, maximum flow is measured. Desired speed of conventional vehicles entering the network on the main link is set to 140 km/m with a wider spread between 80 and 205 km/h. Automated vehicles use the desired speed 130 +-2 km/h.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Pictures showing 95% quantile results for data collection groups:

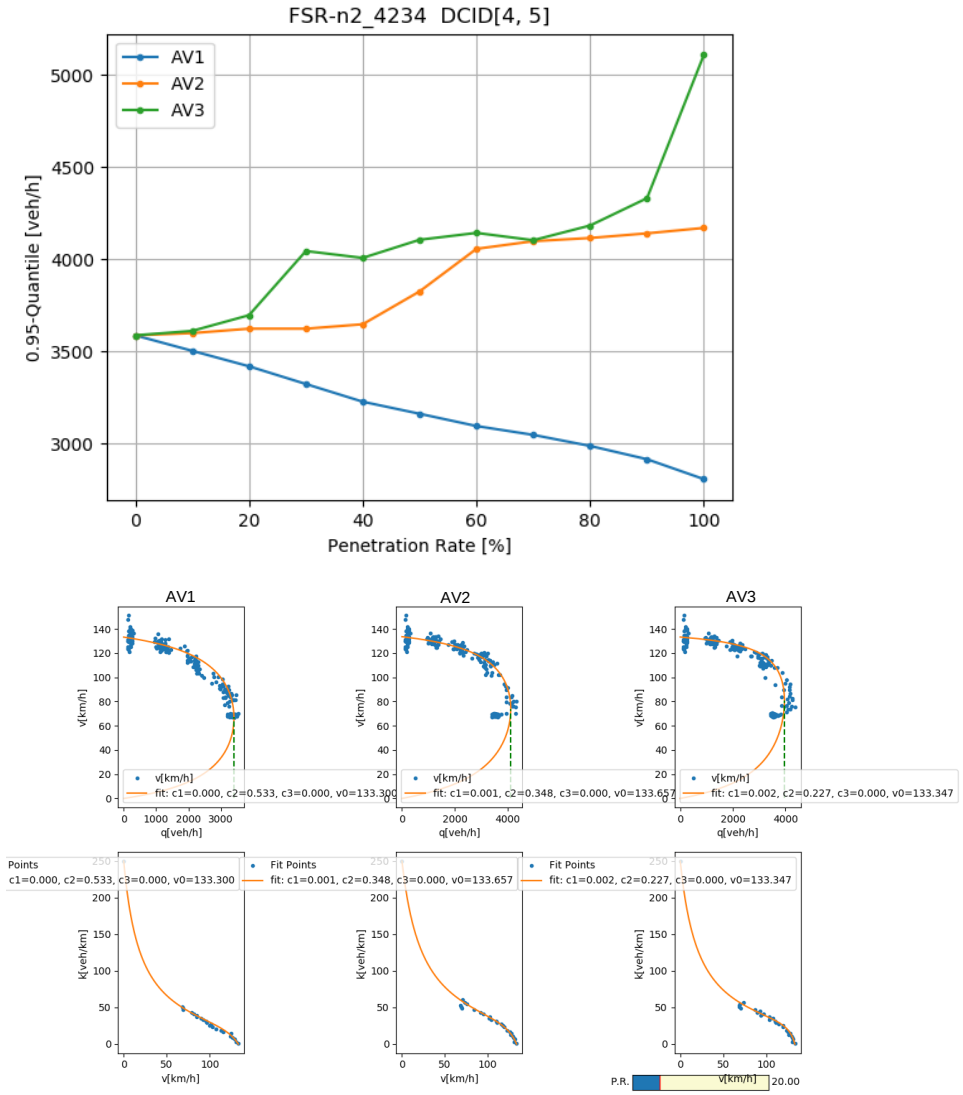


Figure 16 Curve-fitting example: penetration rate 20%, DCID [4,5]. See the attachment for other penetration rates.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

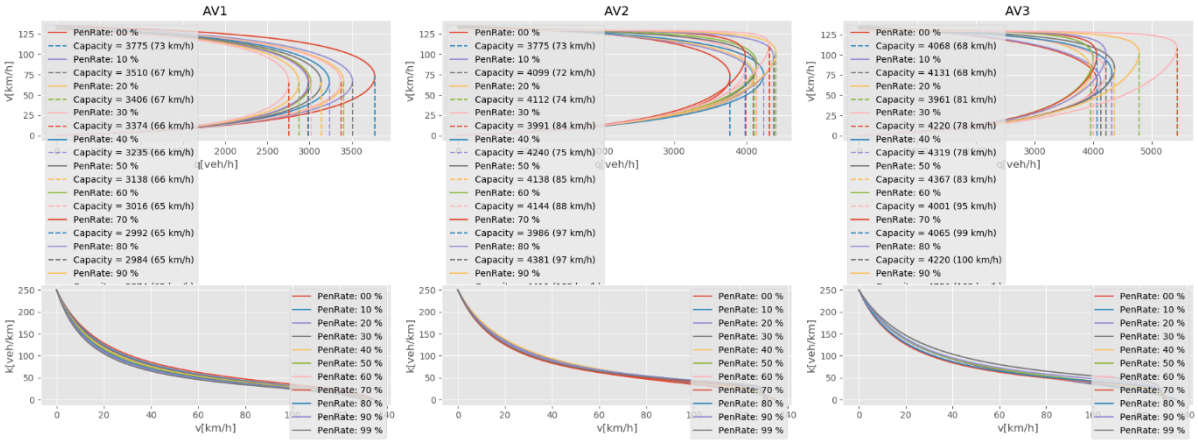


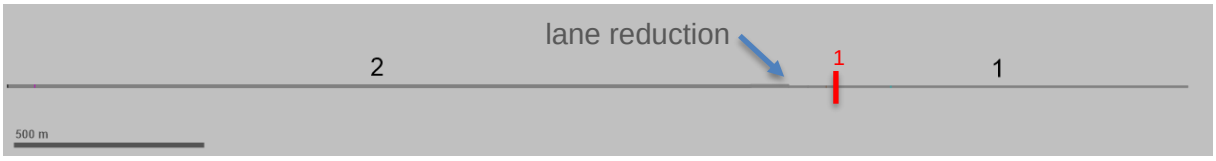
Figure 17 Van Aerde diagrams for DCID [4,5], all penetration rates.

Table 12 Comments to 95%-Quantile results

DCID	Comments
General	Reduction of the number of lanes from 3 to 2 shows better performance with AV2 and AV3 vehicles in comparison with conventional vehicles (zero penetration rate). The throughput value is growing with growing penetration rate. Cautious vehicles (AV1) perform worse – the throughput value decreases with increasing penetration rate (almost linearly). ⇨ There is a difference to the case where only one lane remains after the reduction (see the next network) and the throughput values decreases with lower penetration rates.
[4,5]	Cautious vehicle decreases the throughput value almost linearly in dependency on the penetration rate. AV2 and AV3 vehicles cause increase in the throughput with increasing penetration rate non-linearly. AV3 shows more or less higher values than AV2.

4) Lane number reduction from 2 to 1 lanes (FSR-n1-SuP.inpx)

A two-lane link is reduced to 1-lane link, maximum flow is measured. Desired speed of conventional vehicles entering the network on the main link is set to 140 km/m with a wider spread between 80 and 205 km/h. Automated vehicles use the desired speed 130 +-2 km/h.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Pictures showing 95% quantile results for data collection groups:

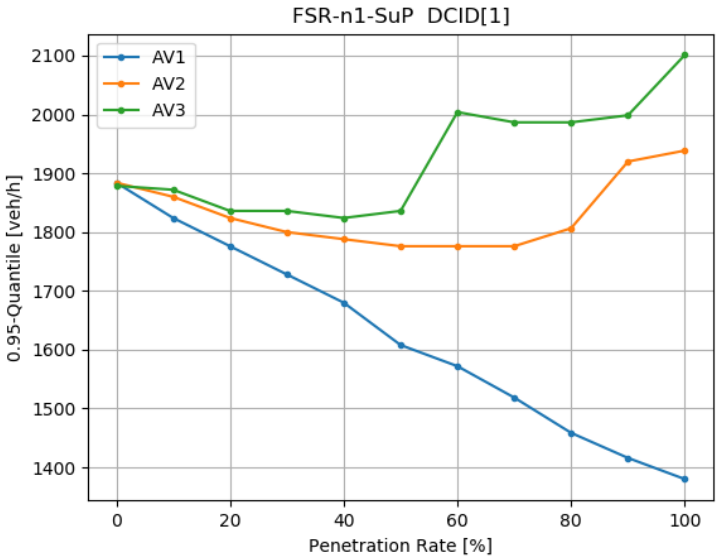


Table 13 Comments to 95%-Quantile results

DCID	Comments
General	The conventional vehicles are using CC1 = 1,05 in this network (KIT value), AV2 and AV3 operate with smaller headways (0,9 or 0,7). Besides that, AV2 is influenced by increased acceleration set to 105. That seems to cause the throughput drop. Changing increased acceleration to 100 % for AV2 leads to higher throughput, close to the result of 0 % penetrations rate. For AV3 changing the increased acceleration from 110 to 100 does not bring higher throughput. 0% penetration rate seems to be more fluent at lane changing area (less stops, changing during parallel driving). Penetration < 100%: lower headways (including lower standstill distance) in combination with increased acceleration bring higher saturation flow at signals but slightly lower throughput in lane reduction case 2 lanes -> 1 lane. Penetration nearly 100%: higher throughput also in lane reduction cases possible.
[1]	Cautious vehicles decrease the throughput almost linearly with growing penetration rate. AV2 and AV3 vehicles cause decrease in the throughput in lower penetration rates. With higher penetration rates (> 80% for AV2 or > 50% for AV3) the throughput goes up.

AV1 AV2 AV3

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

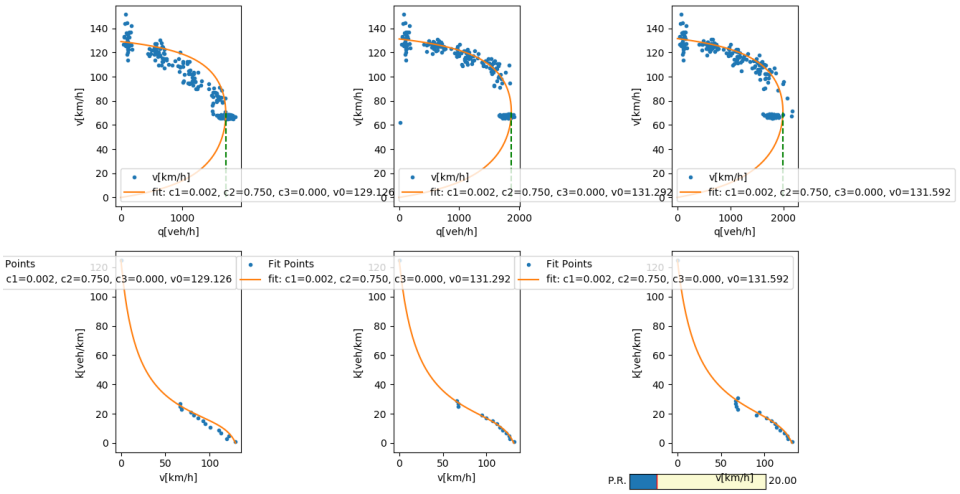


Figure 18 Curve-fitting example: penetration rate 40%, DCID [1]. See the attachment for other penetration rates.

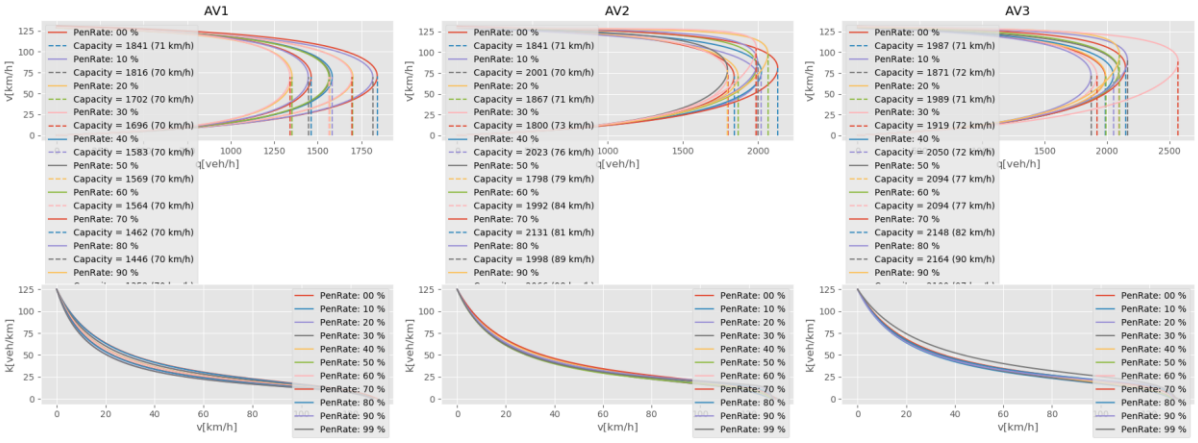
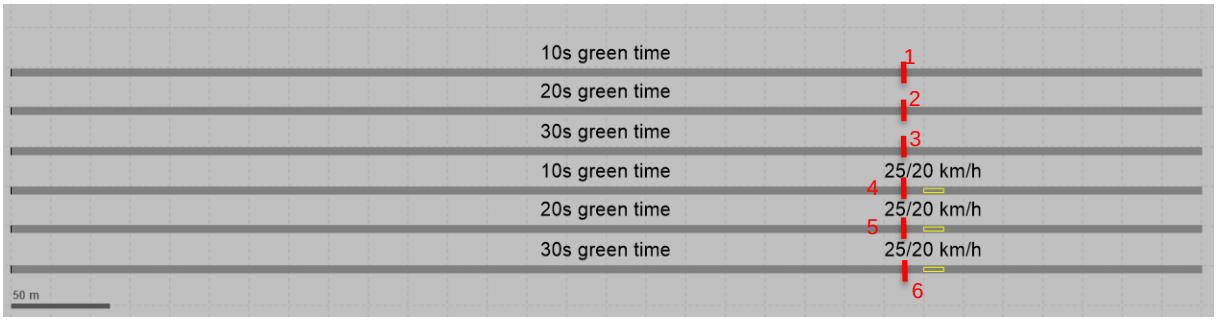


Figure 19 Van Aerde diagrams for DCID [1], all penetration rates.

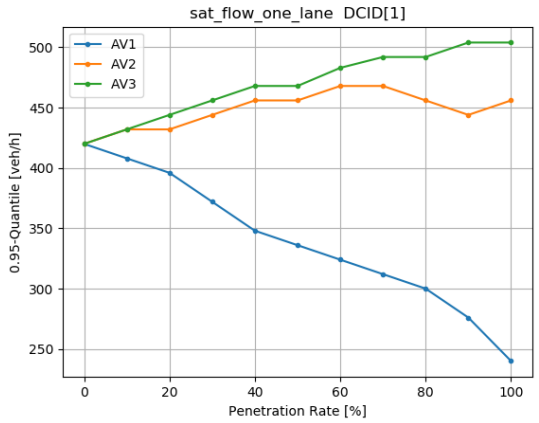
5) Saturation flow on one-lane link with signal (sat_flow_one_lane.inpx)

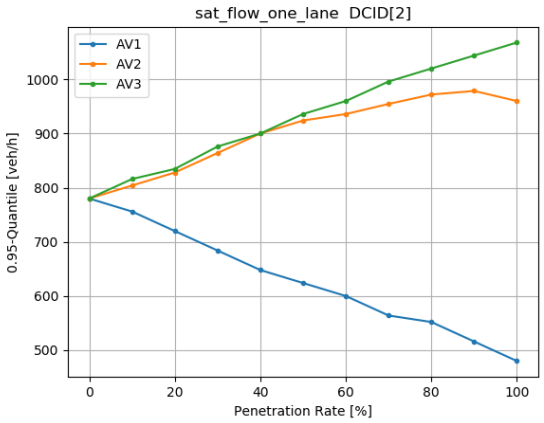
Results provide information for the calculation of the saturation flow. The green time used in the model was set to 10, 20 or 30 seconds. First three links reflects the situation for direction “through“, without a speed limitation. Second three links reflect the situation when the speed of vehicles is limited e.g. through curvature when turning left or right. Desired speed is set to 50 km/h with wider spread for conventional vehicles and 50+-2 km/h for autonomous vehicles.

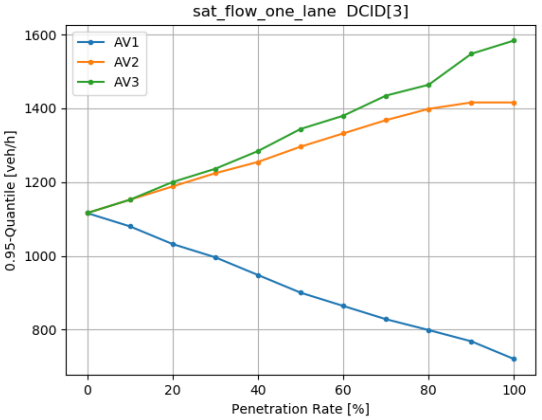


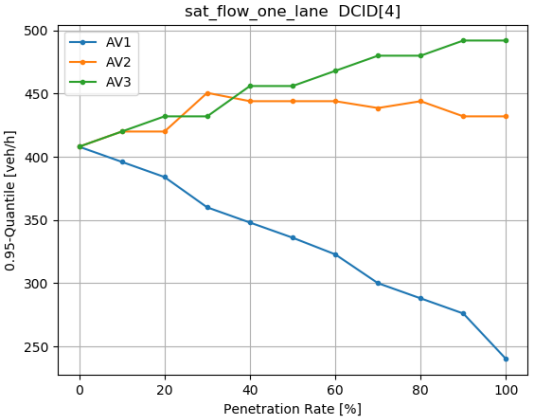
This project has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No. 723201

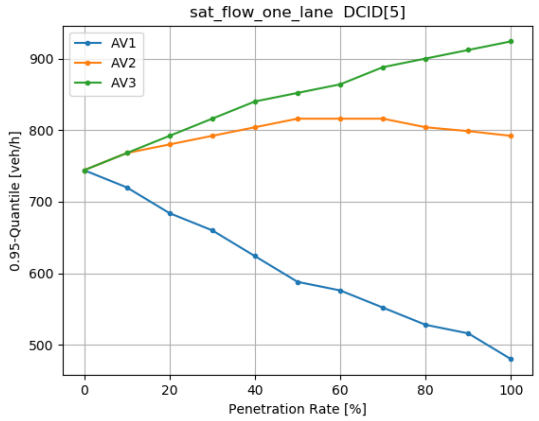
Pictures showing 95% quantile results for data collection groups:











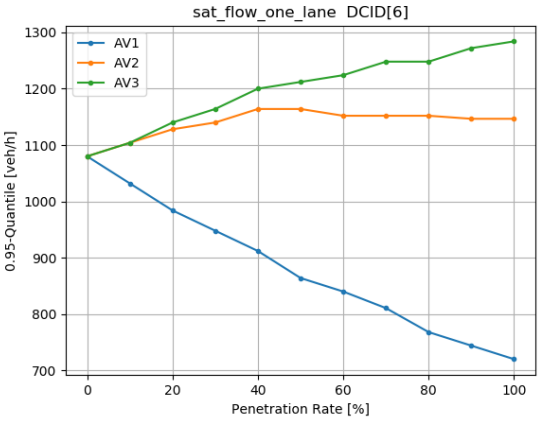


Table 14 Comments to 95%-Quantile results

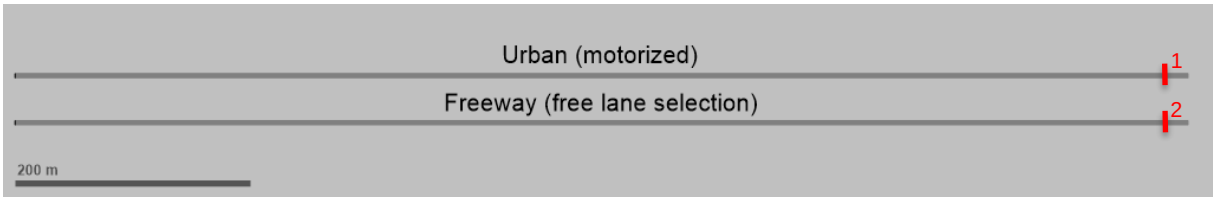
DCID	Comments
General	First three data collection points represent situation at signals with through traffic, second three data collection points represent the situation with turnings traffic where speed limitation due curvature is expected. Cautious vehicles show linear decrease in throughput with growing penetration rate. AV2 and AV3 vehicles show increase in throughput.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

[1]	AV3 shows a linear throughput growth with growing penetration rate. AV2 shows a small growth followed by stagnation. Caution vehicles (AV1) show linear decrease with growing penetration rates.
[2]	AV3 shows a linear throughput growth with growing penetration rate. AV2 shows a small growth followed by stagnation at high penetration rates. Caution vehicles (AV1) show linear decrease with growing penetration rates.
[3]	AV3 shows a linear throughput growth with growing penetration rate. AV2 shows a small growth followed by stagnation at high penetration rates. Caution vehicles (AV1) show linear decrease with growing penetration rates.
[4]	AV3 shows a linear throughput growth with growing penetration rate. AV2 shows a small growth followed by stagnation. Caution vehicles (AV1) show linear decrease with growing penetration rates.
[5]	AV3 shows a linear throughput growth with growing penetration rate. AV2 shows a small growth followed by stagnation. Caution vehicles (AV1) show linear decrease with growing penetration rates.
[6]	AV3 shows a linear throughput growth with growing penetration rate. AV2 shows a small growth followed by stagnation. Caution vehicles (AV1) show linear decrease with growing penetration rates.

6) Simple one-lane link (simple_link.inpx)

This network provides results for theoretical capacity on one lane link under ideal conditions without influence of intersections, parking manoeuvres or other sources of disturbance. The resulting maximum flow depends on speed and settings for following behaviour. Desired speed is set to 50 km/h with wider spread for conventional vehicles and 50+-2 km/h for autonomous vehicles.



In the micro model, we never see speeds slower then v0, because they are placed in that way into the network. The fundamental diagram is basically a straight line parallel to the flow.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Pictures showing 95% quantile results for data collection groups:

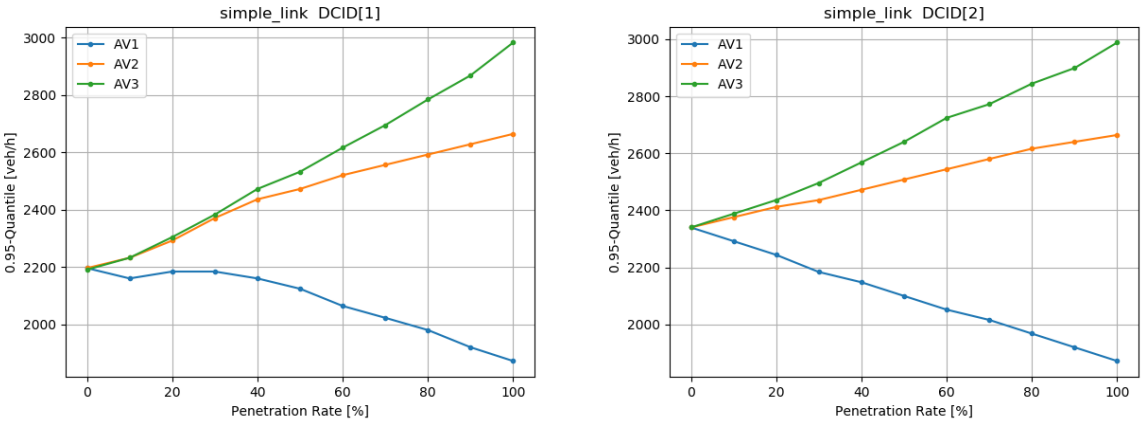


Table 15 Comments to 95%-Quantile results

DCID	Comments
General	Because the headway plays the most important role in following behaviour from the capacity perspective, the trends in results are clear.
[1]	Cautious vehicles cause a decrease in the capacity of the link, especially from penetration rate 40 % where the capacity decreases linearly. AV2 and AV3 vehicles lead into increase of the link capacity, which is more visible for AV3 at penetration rates > 30 %.
[2]	Cautious vehicles cause a linear decrease in the capacity of the link. AV2 and AV3 vehicles lead into increase of the link capacity, which is more visible for AV3 at penetration rates > 20 %.

In the micro model, we never see speeds slower then v0, because they are placed in that way into the network. The fundamental diagram is basically a straight line parallel to the flow.

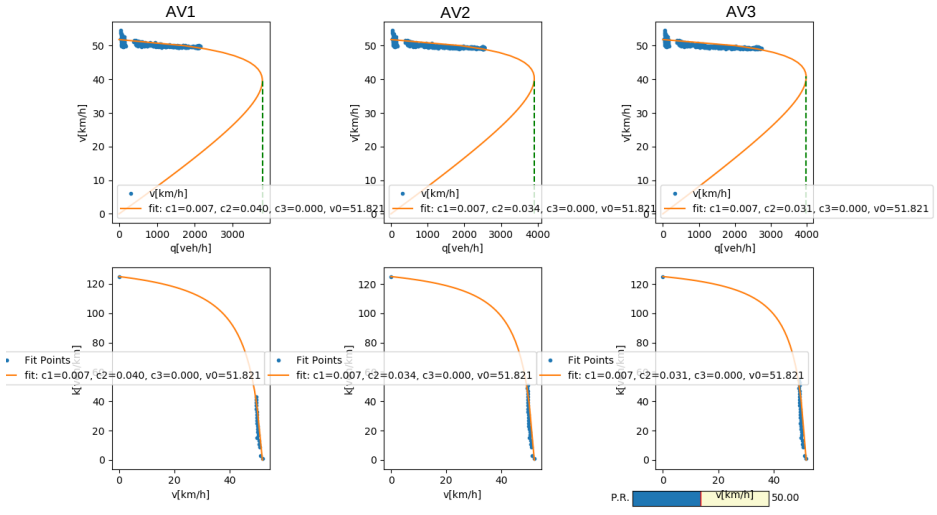


Figure 20 Curve-fitting example: penetration rate 50%, DCID [2]. See the attachment for other penetration rates. – NOT REASONABLE CURVEFITTING because of the use case

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

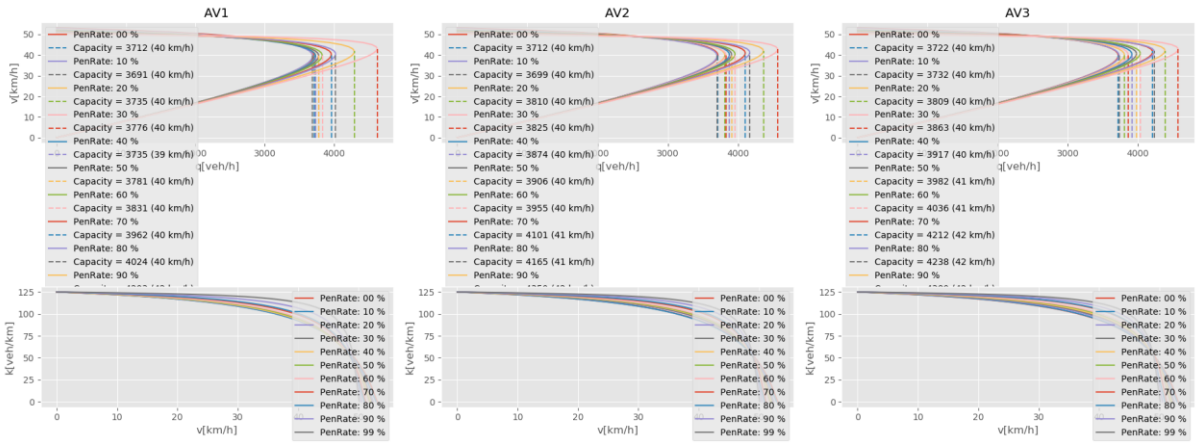
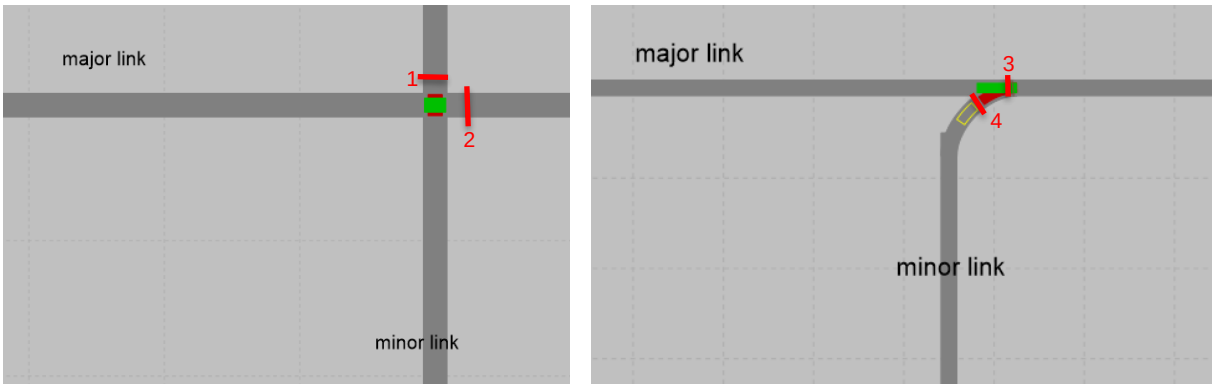


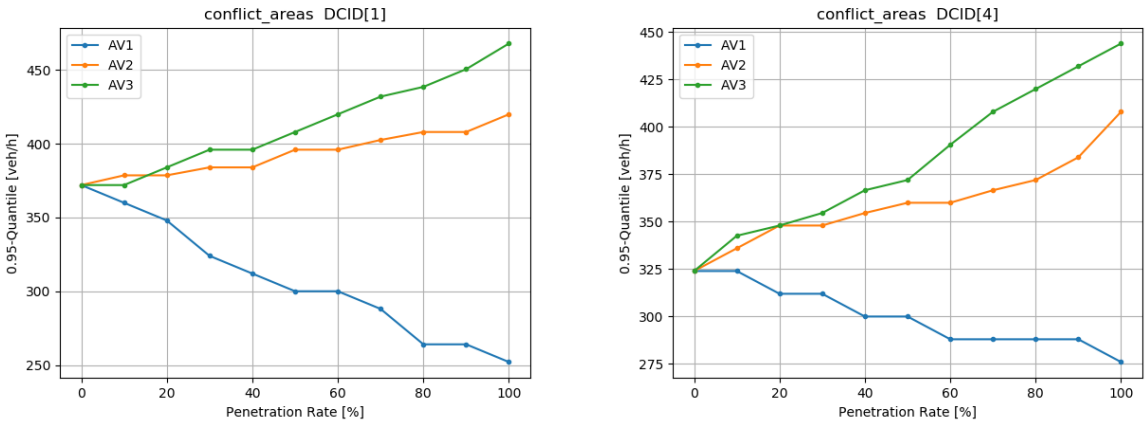
Figure 21 Van Aerde diagrams for DCID [2], all penetration rates – NOT REASONABLE CURVEFITTING because of the use case

7) Simple crossing and simple merging conflict

Results from this network provide information about the capacity of a simple conflict area for relative comparison of different driving logics and penetration rates. Desired speed is set to 50 km/h with wider spread for conventional vehicles and 50+-2 km/h for autonomous vehicles.



Pictures showing 95% quantile results for data collection groups:



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Table 16 Comments to 95%-Quantile results

DCID	Comments
General	The data collection point 1 and 4 are placed on links with the minor flow. There are visible trend in the diagrams – cautious vehicles lead to capacity decrease, AV2 and AV3 vehicles lead to capacity increase. The diagrams above show the case with high major flow. DCID [2] and [3] are not relevant.
[1]	Almost linear capacity decrease with caution vehicles and almost linear capacity increase with AV2 and AV3 vehicles.
[4]	Almost linear capacity decrease with caution vehicles and almost linear capacity increase with AV3. AV2 shows also capacity increase but nonlinearly.

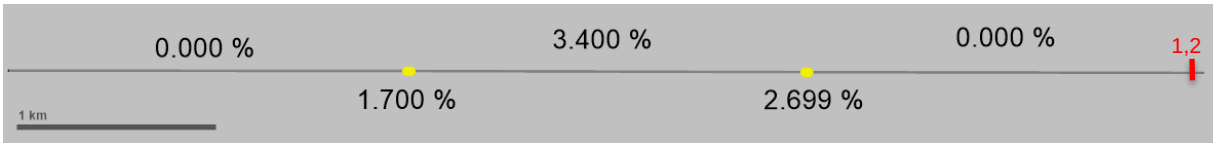
In this case, the van Aerde diagrams are not appropriate to show dependencies. The capacity depends on the traffic volume in the major flow.

8) Influence of gradient - uphill slopes (S-n2_SuP.inpx)

Results from this network are showing the impact of gradient. The value of gradient impacts the driving behaviour in Vissim via the maximum acceleration and maximum deceleration on a link:

- by -0.1 m/s² per gradient percent incline. The maximum accelerating power decreases when the deceleration power increases.
- by 0.1 m/s² per gradient percent downgrade. The accelerating power increases when the deceleration power decreases.

Desired speed is set to 130 km/h with wider spread for conventional vehicles and 130+-2 km/h for autonomous vehicles.



This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

Pictures showing 95% quantile results for data collection groups:

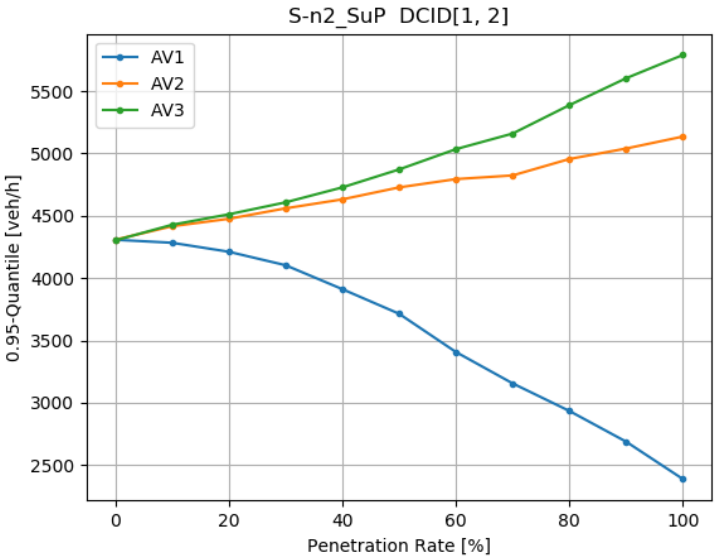


Table 17 Comments to 95%-Quantile results

DCID	Comments
General	Please note: these results are calculated with cars only (slope has higher impact on HGVs, than on cars).
[1,2]	AV2 and AV3 vehicles lead to capacity increase, AV3 significantly faster with growing penetration rate. Cautious vehicles in the traffic flow lead to capacity drop.

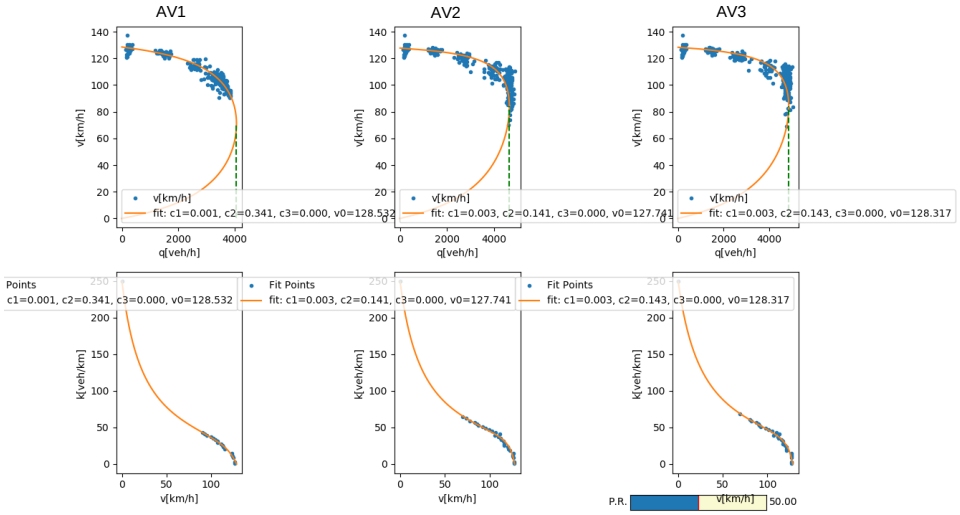


Figure 22 Curve-fitting example: penetration rate 50%, DCID [1,2]. See the attachment for other penetration rates.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201

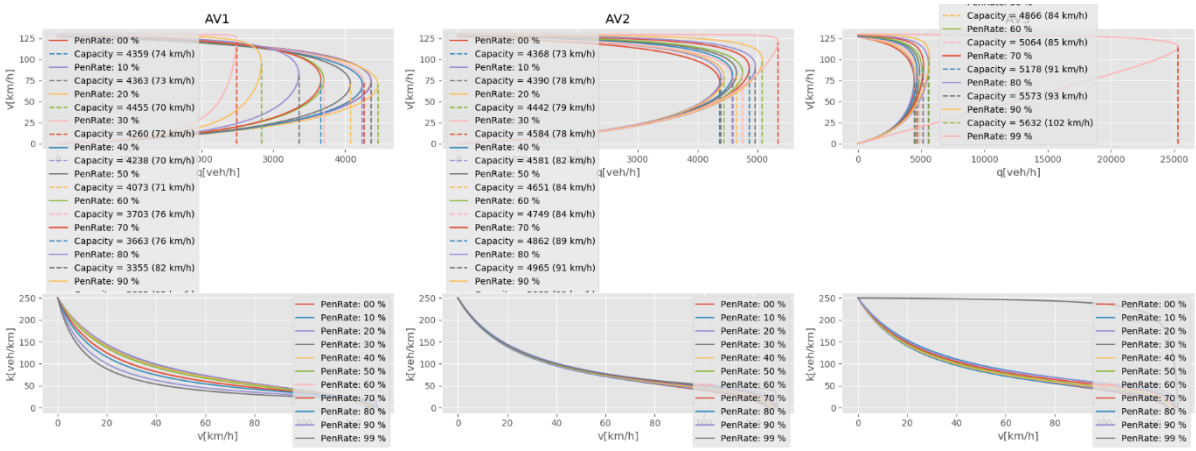


Figure 23 Van Aerde diagrams for DCID [1,2], all penetration rates.

Attachments

1) 95%-quantiles capacity diagrams (29 pictures)
   Example:

Network name	Data collection points
A1-n3 SuP plus KAL_E1-n3_94789	DCID[9, 10, 11].png

2) Van Aerde diagrams (348 pictures)
   Example:

Network name	Data collection points	Pene-tration rate
A1-n3 SuP plus KAL_E1-n3_94789_	DCID[1, 2, 3]	PR10.png

3) Van Aerde data (1 excel file)
4) Curve fitting tool including individual data for manual curve fitting if needed

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 723201