TraCI4Matlab: User's Manual

Facultad de Minas
GAUNAL
Grupo de Automática de la Universidad Nacional

POLITÉCNICO COLOMBIANO
JAIME ISAZA CADAVID

Versión 1

March 2014

Modifications by document version

Version history

Version	Authors	Date	Modification description
1	Andrés Felipe Acosta Gil	11/03/2014	Document creation

Contents

1. What is TraCI4Matlab?

TraCI4Matlab is an API (Application Programming Interface) developed in Matlab that allows the communication between any application developed in this language and the urban traffic simulator SUMO (Simulation of Urban Mobility). The functions that comprise TraCI4Matlab implement the TraCI (Traffic Control Interface) application level protocol, which is built on top of the TCP/IP stack, so the application developed in Matlab, which is the client, can access and modify the simulation environment provided by the server (SUMO). TraCI4Matlab allows controlling SUMO objects such as vehicles, traffic lights, junctions, etc, enabling applications like traffic lights predictive control and dynamic route assignment, among others.

2. TraCI4Matlab Installation

2.1 Prerequisites

- Operating System: Windows 7 x64 or higher
- Matlab R2011b x64 or higher
- SUMO 0.19.0 or higher. Installation instructions can be found at http://sumo-sim.org/userdoc/Installing.html

2.2 Installation

- **Step 1: Setting up the environment**

  Set up the SUMO_HOME environment variable with system scope, with a value corresponding to the SUMO installation root directory. For example, if the SUMO version 0.19.0 was installed in C:\, then the SUMO_HOME environment variable would have a value of C:\sumo-0.19.0. Environment variables can be configured as follows: First, click in the Windows start button, then right click in "My computer" and click in the properties option, as shown in figure 1. On the left side of the window that opens, click in the link "Advanced system configuration", as shown in figure 2. After that, in the window that opens, click in the button "Environment variables". In the "System variables" field, click on the button "New" and configure the variable according to the SUMO installation, as shown in figure 3. Finally, in the same field, look for the *"path"* variable and add the route to the *bin* directory of the SUMO installation. It can be accomplished by adding a semicolon to the current value and then the corresponding route, as shown in figure 4.

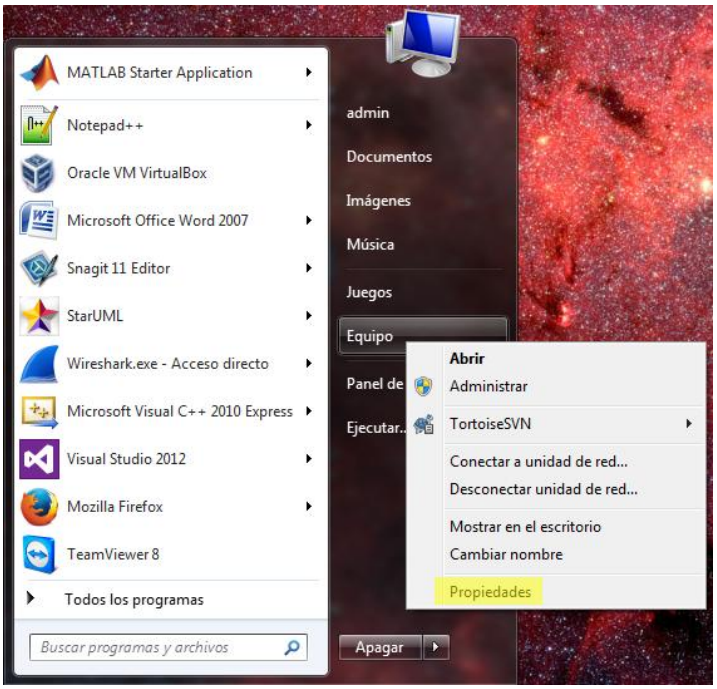

Figure 1 Environment variables configuration, access to the system properties

- **Step 2:** Download TraCI4Matlab and decompress it in a directory which is in the Matlab path. Normally, it's found at Documents\MATLAB. It can be verified in Matlab, in the File menu, option "set path".

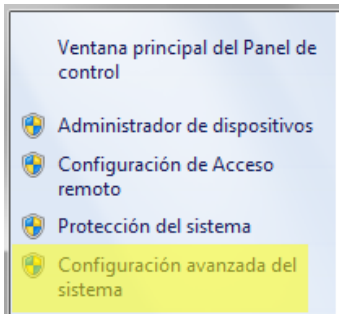

Figure 2 Environment variables configuration, advanced system configuration

- **Step 3: Add TraCI4Matlab to the Matlab's path**
  On Matlab, go to the File menu, option set *path*. Select the button *"Add with subfolders"* and select the TraCI4Matlab folder decompressed in the previous step, as shown in figure 5. Finally, click in the button *save* and *close*.

- **Step 4: Test TraCI4Matlab**
  Open the *script* traci_test2.m, found in the TraCI4Matlab 's examples folder and run it. The SUMO GUI should open, click in the play button inside it. A simulation is executed in which visual tracking to a blue vehicle is shown. When the simulation is finished, in the second 801, a Matlab plot is shown, as shown in figure 6.

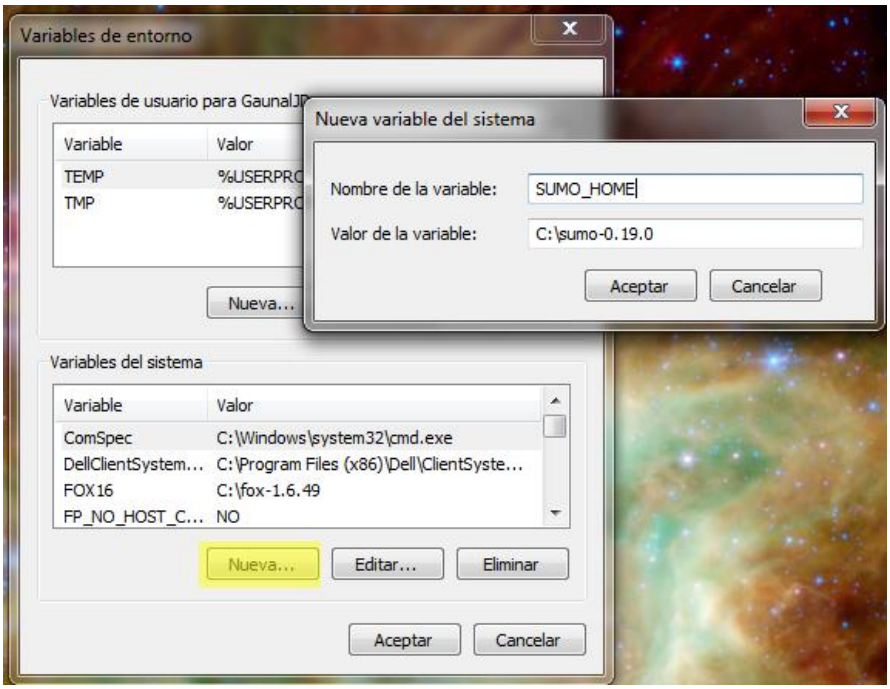

Figure 3: Environment variables configuration, creation

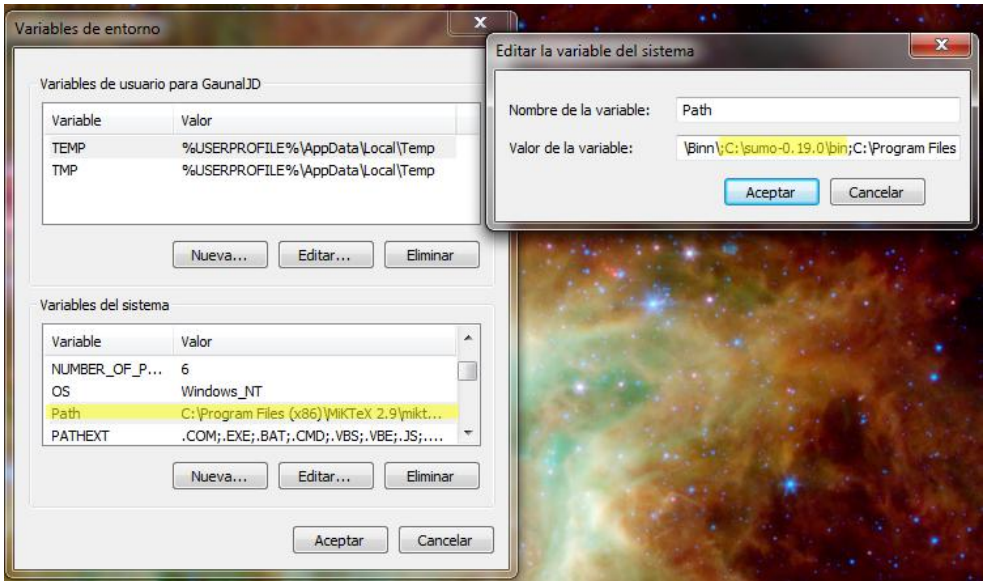

Figura 4: Environment variables configuration, editing the Windows path

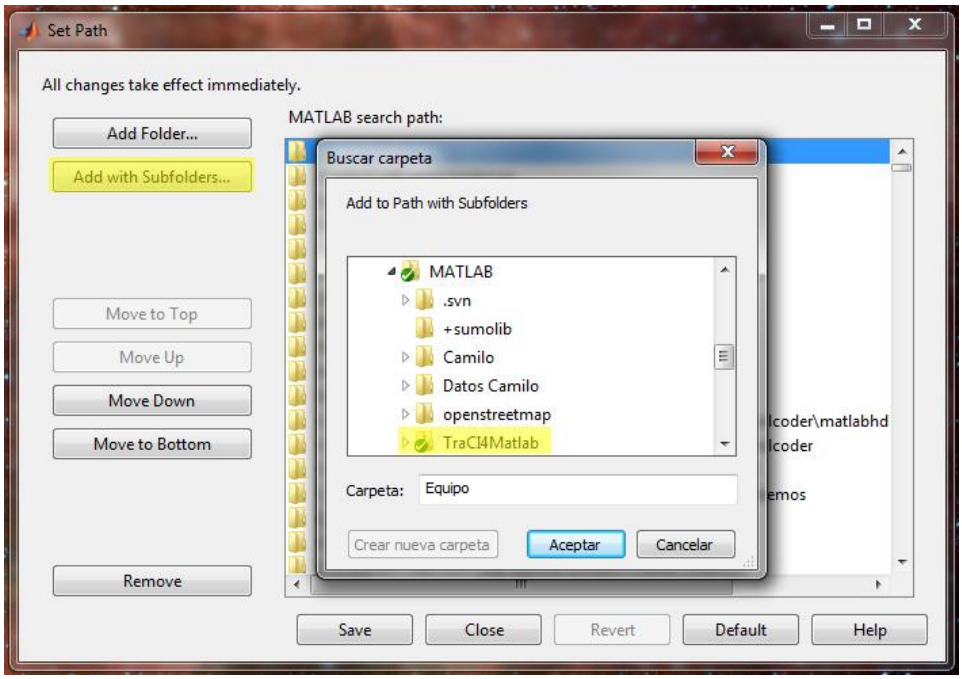

Figure 5: Adding TraCI4Matlab to the Matlab path

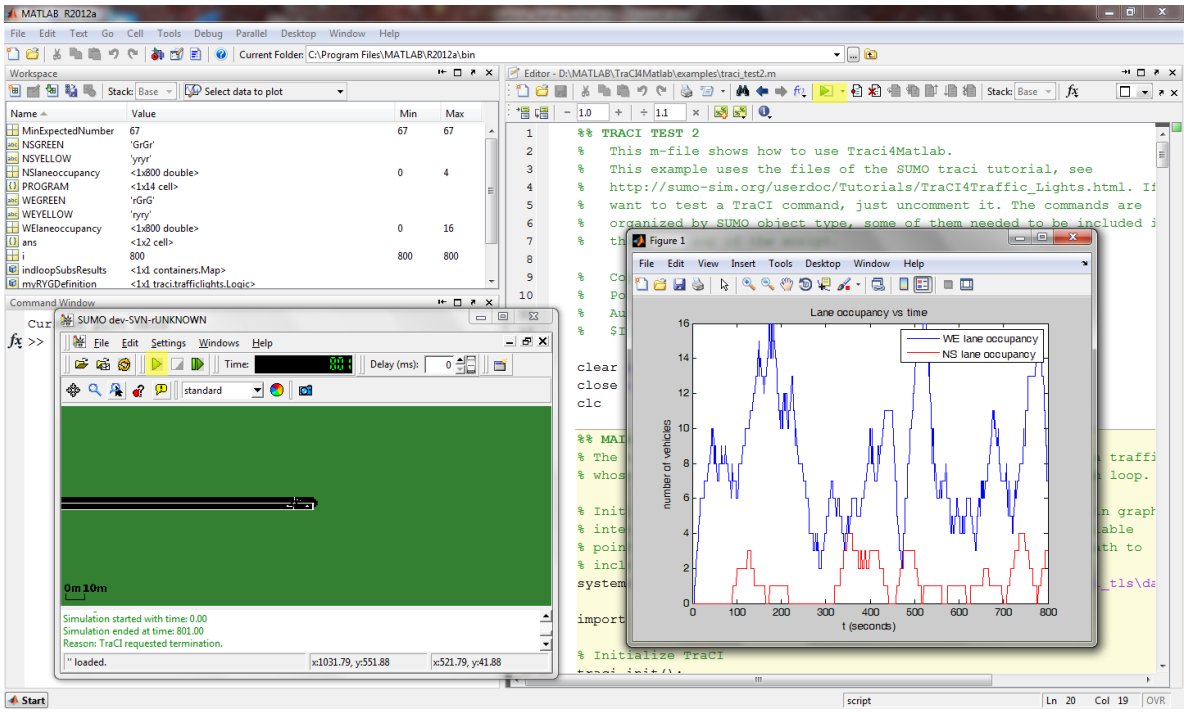

Figure 6: Testing TraCI4Matlab

3. Using TraCI4Matlab

3.1 Creating the simulation scenario in SUMO

To use TraCI4Matlab, the first step is to create a simulation scenario in SUMO. The creation of that scenario is out of the scope of this manual, but a official tutorial can be found at http://sumo-sim.org/userdoc/Tutorials/Quick_Start.html.

3.2 Configure SUMO in server mode

It's important to note that, to use TraCI4Matlab correctly , the configuration file of the SUMO scenario (the one with extension *.sumocfg*) must include the *traci_server* element configured to the 8813 port, which is the port used by default, as shown in figure 7. The *traci_server* element makes SUMO does not execute the simulation immediately, but to enter in a listening state on the 8813 port.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2
3  <configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4
5      <input>
6          <net-file value="cross.net.xml"/>
7          <route-files value="cross.rou.xml"/>
8          <additional-files value="cross.det.xml"/>
9      </input>
10
11     <time>
12         <begin value="0"/>
13     </time>
14
15     <report>
16         <verbose value="true"/>
17         <no-step-log value="true"/>
18     </report>
19
20     <traci_server>
21         <remote-port value="8813"/>
22     </traci_server>
23
24 </configuration>
```

Figure 7: Including the *traci_server* element in the SUMO configuration file

3.3 Creating the application in Matlab

- **Step 1: Execute SUMO from Matlab**

  Any application in Matlab that uses TraCI4Matlab must start by executing the commands: sumo if it is desired to execute the simulation without visualization or sumo-gui if it is desired to execute SUMO in GUI mode; specifying as a parameter the route where the configuration file of interest is found, which is explained in the previous step. This requirement is met through the Matlab's *system* command, as shown in figure 8.

```
1    %   Copyright 2013 Universidad Nacional de Colombia,
2    %   Politecnico Jaime Isaza Cadavid.
3    %   Authors: Andres Acosta, Jairo Espinosa, Jorge Espinosa.
4    %   $Id: traci_test2_clean.m 2 2013-12-21 21:39:57Z aacosta $
5
6 -  clear all
7 -  close all
8 -  clc
9
10 - import traci.constants
11
12 - system(['sumo-gui -c ' getenv('SUMO_HOME')...
13       '\docs\tutorial\traci_tls\data\cross.sumocfg&']);
```

Figure 8: Executing SUMO from Matlab

Optionally, the command *import traci.constants* can be added to facilitate referencing the TraCI constants, in case that TraCI subscriptions are made, which will be explained later. Thus, for instance, to reference the command LAST_STEP_VEHICLE_NUMBER of the TraCI constants, it's enough to write constants.LAST_STEP_VEHICLE_NUMBER instead of traci.constants.LAST_STEP_VEHICLE_NUMBER.

- **Step 2: Initialize the connection**

  After initializing the SUMO server in the previous step, the connection must be established with the function traci.init, as shown in figure 9. If the SUMO server was configured to use the 8813 port, the traci.init function doesn't need additional parameters. To use them, it's recommended to use the function help, by writing help traci.init in the Matlab's command window.

```
31   % Initialize TraCI
32 - traci.init();
33
34 - traci.inductionloop.subscribe('0');
35 - for i=1:length(steps)
36
37       % Perform a simulation step (one second)
38 -     traci.simulationStep();
39
40 -     programPointer = min(programPointer+1, length(PROGRAM));
41
```

Figure 9: Initializing the connection to the SUMO server

- **Step 3: Developing the application**

  Normally, TraCI4Matlab applications include a main loop, in which the simulation's time steps are executed through the command traci.simulationStep. In this loop, the attributes of the SUMO's simulation objects are accessed and modified. The SUMO objects are grouped in thirteen domains: gui, lane, poi, simulation, trafficlights, vehicletype, edge, inductionloop, junction, multientryexit, polygon, route, and vehicle. the general structure to access or modify a SUMO object is: traci.<*domain*>.<*get/set_wrapper()*>, where *domain* can take any of the domains listed previously and *get/set_wrapper()* are the functions to acces the values (*get*) or modify (*set*) the attributes of the object of interest.

For example, if the velocity's value of the vehicle with ID *veh_1* in the current time step is required, the command current_speed_veh1 = traci.vehicle.getSpeed('veh_1') can be executed. To obtain a list of all the commands of a specific domain, write help traci.<*domain*> in the Matlab's command window, where *domain* can take any of the values listed previously. Figure 10 shows an example in the case of the lane domain. further help regarding a function can be obtained by clicking it.

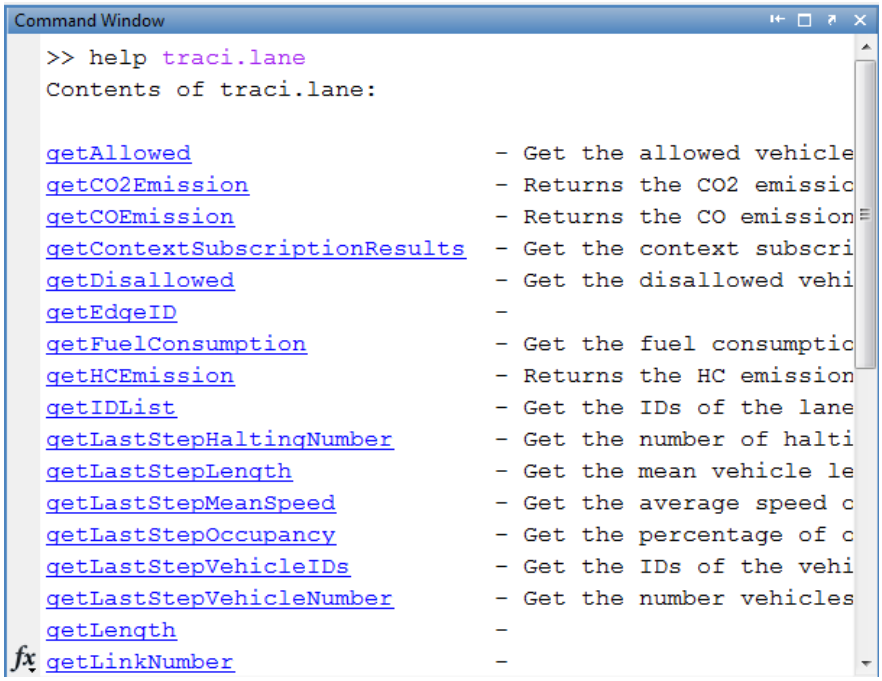

Figure 10: Obtaining a list of the functions related to a SUMO object

The simulation's main loop can be executed until a fixed time, or until all vehicles of the simulation have arrived to their destinations. In this case, the traci.simulation.getMinExpectedNumber function is used, as shown in figure 11.

```
35 -  while traci.simulation.getMinExpectedNumber > 0
36
37        % Perform a simulation step (one second)
38 -      traci.simulationStep();
39
40 -      programPointer = min(programPointer+1, length(PROGRAM));
41
42 -      indloopSubsResults = traci.inductionloop.getSubscriptionResults('0')
43 -      no = indloopSubsResults(constants.LAST_STEP_VEHICLE_NUMBER);
```

Figure 11: Condition to execute the simulation until all the vehicles have reached their destination

TraCI4Matlab includes functions to make TraCI subscriptions. TraCI subscriptions allow retrieving several SUMO by means of a single command. To use TraCI TraCI subscriptions, it's neccesary to know the TraCI constants containing the codes of different attributes related to a TraCI subscription. The TraCI constants can be found by typing edit traci.constants in the Matlab's command window, and locate the "*VARIABLE TYPES*" field.

For example, suppose that it's desired to make a TraCI subscription to acces the values of the attributes LAST_STEP_VEHICLE_NUMBER and LAST_STEP_MEAN_SPEED of the *induction loop* with ID '0'. In this case, the command shown in figure 12 shall be used. Note that the import traci.constants command must be issued at the beginning of the *script*, as explained in the step 1.

```
33
34 -    traci.inductionloop.subscribe('0',{constants.LAST_STEP_VEHICLE_NUMBER,..
35         constants.LAST_STEP_MEAN_SPEED});
```

Figure 12: TraCI subscriptions

Now, to access the values related to the TraCI subscription, the commands shown in the figure 13 shall be issued inside the main loop. Note that firstly, results are stored in a handle variable which later is indexed with the TraCI constants to which the subscription was made.

```
44 -    indloopSubsResults = traci.inductionloop.getSubscriptionResults('0');
45 -    no = indloopSubsResults(constants.LAST_STEP_VEHICLE_NUMBER);
46 -    lsms = indloopSubsResults(constants.LAST_STEP_MEAN_SPEED);
```

Figure 13: Getting the results of the TraCI subscription

- **Step 4: Closing the connection**

  Finally, the connection to the SUMO server is closed as shown in figure 14. Later, post-processing of the obtained data can be made thanks to the advantages of the Matlab tools.

```
65 -    end
66
67 -    traci.close()
68
69 -    plot(steps, WElaneoccupancy)
70 -    hold;
71 -    plot(steps, NSlaneoccupancy, 'r')
72 -    legend('WE lane occupancy', 'NS lane occupancy')
73 -    title('Lane occupancy vs time')
74 -    xlabel('t (seconds)')
75 -    ylabel('number of vehicles')
```

Figure 14: Closing the connection to the SUMO server