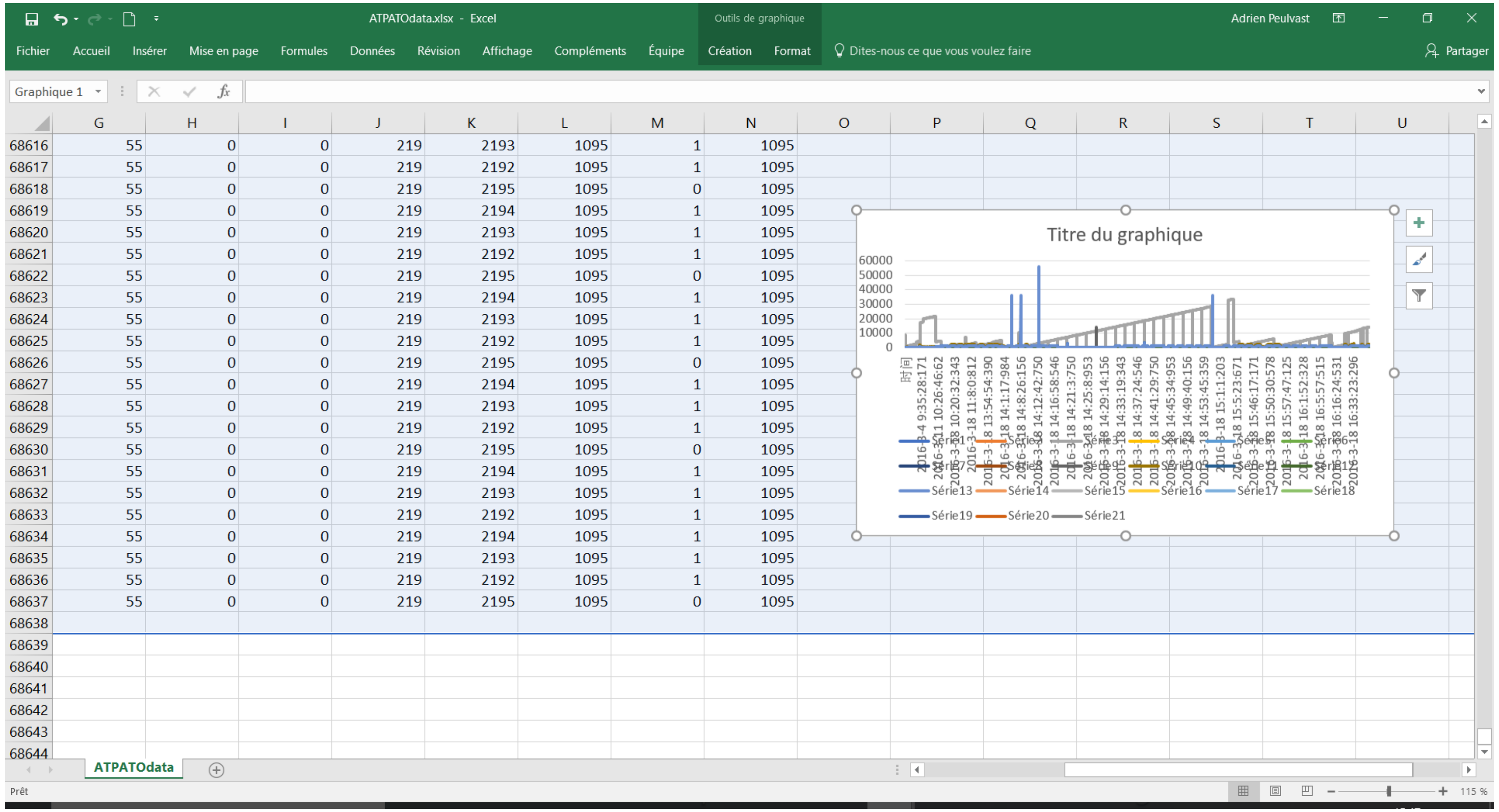
RBC (BLT) and canvas use for metro system testing and simulation

EuroTCL 2017 conference - Adrien Peulvast - SafeRail

1) RBC use for Odometry data display

- The story starts with a metro signalling system supplier, which have records of odometry data, but no correct solution to display them in a readable way
  - Files of ~4Mo - ~70 000 records, Excel format
  - No possibility to display graphs in a convenient way with Excel
    - Not easy zooming capability
    - Difficult to read the values
    - The way it is done : select parts of the data in the table to display a graph, which is also not convenient
  - No possibility to process data format errors, analyse data values

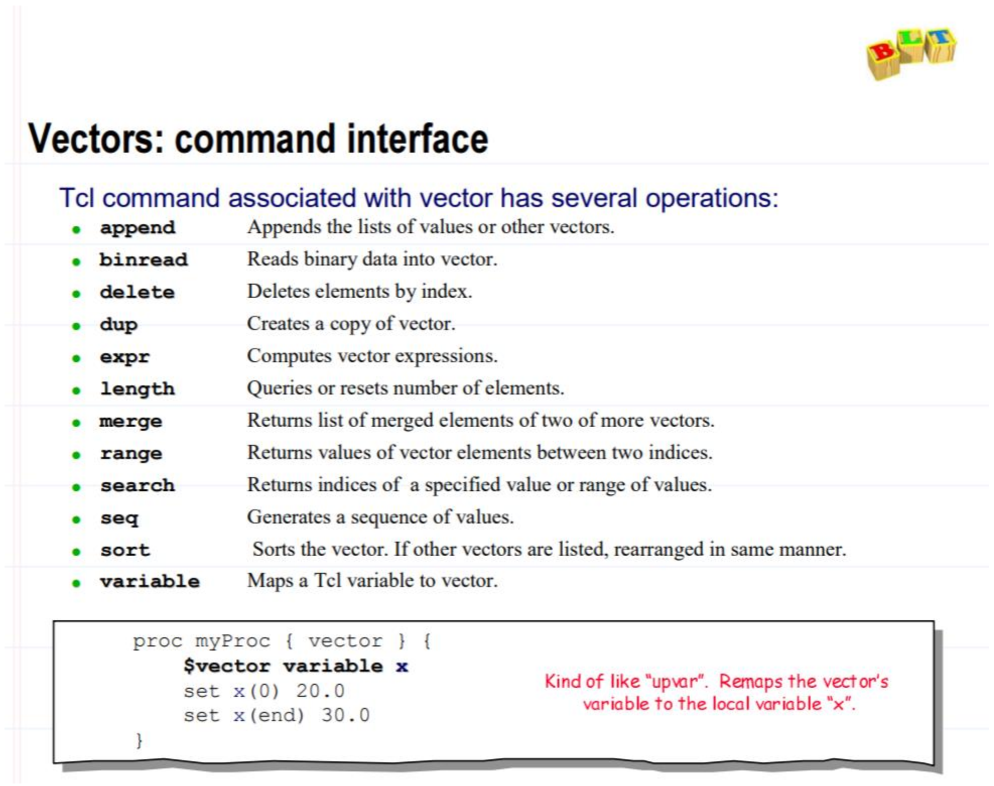
1) RBC use for Odometry data display



	G	H	I	J	K	L	M	N
68616	55	0	0	219	2193	1095	1	1095
68617	55	0	0	219	2192	1095	1	1095
68618	55	0	0	219	2195	1095	0	1095
68619	55	0	0	219	2194	1095	1	1095
68620	55	0	0	219	2193	1095	1	1095
68621	55	0	0	219	2192	1095	1	1095
68622	55	0	0	219	2195	1095	0	1095
68623	55	0	0	219	2194	1095	1	1095
68624	55	0	0	219	2193	1095	1	1095
68625	55	0	0	219	2192	1095	1	1095
68626	55	0	0	219	2195	1095	0	1095
68627	55	0	0	219	2194	1095	1	1095
68628	55	0	0	219	2193	1095	1	1095
68629	55	0	0	219	2192	1095	1	1095
68630	55	0	0	219	2195	1095	0	1095
68631	55	0	0	219	2194	1095	1	1095
68632	55	0	0	219	2193	1095	1	1095
68633	55	0	0	219	2192	1095	1	1095
68634	55	0	0	219	2194	1095	1	1095
68635	55	0	0	219	2193	1095	1	1095
68636	55	0	0	219	2192	1095	1	1095
68637	55	0	0	219	2195	1095	0	1095
68638								
68639								
68640								
68641								
68642								
68643								
68644								

1) RBC use for Odometry data display

- Use of RBC (Refactored Blt Components) to display and browse in the data
  - Why ? Initially, just to try…then used for analysis. Why RBC ? Teacup…
  - Quick (to code and to use), convenient display, zooming capability
  - Agile development : starts with a quick hack up to a complete tool
  - Operations on BLT's vectors allow to build data analysis modules (source from BLT slides) :



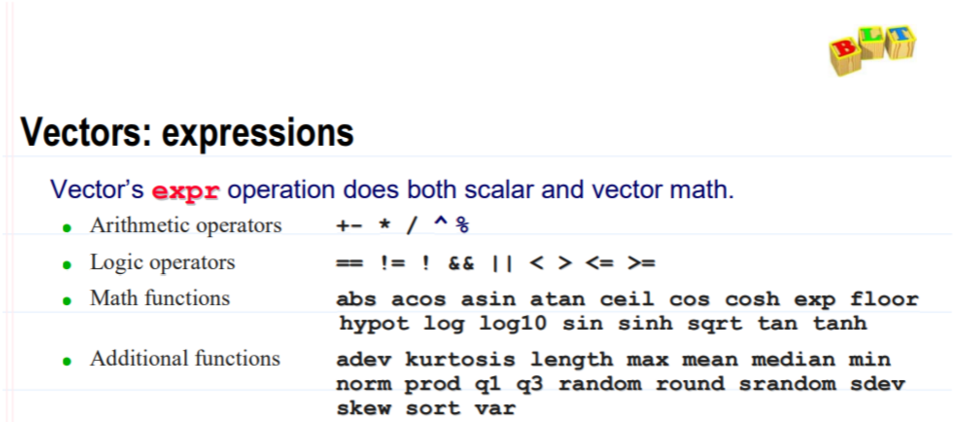
Vectors: command interface

Tcl command associated with vector has several operations:

• **append**	Appends the lists of values or other vectors.
• **binread**	Reads binary data into vector.
• **delete**	Deletes elements by index.
• **dup**	Creates a copy of vector.
• **expr**	Computes vector expressions.
• **length**	Queries or resets number of elements.
• **merge**	Returns list of merged elements of two of more vectors.
• **range**	Returns values of vector elements between two indices.
• **search**	Returns indices of a specified value or range of values.
• **seq**	Generates a sequence of values.
• **sort**	Sorts the vector. If other vectors are listed, rearranged in same manner.
• **variable**	Maps a Tcl variable to vector.

```
proc myProc { vector } {
    $vector variable x
    set x(0) 20.0
    set x(end) 30.0
}
```

Kind of like "upvar". Remaps the vector's variable to the local variable "x".

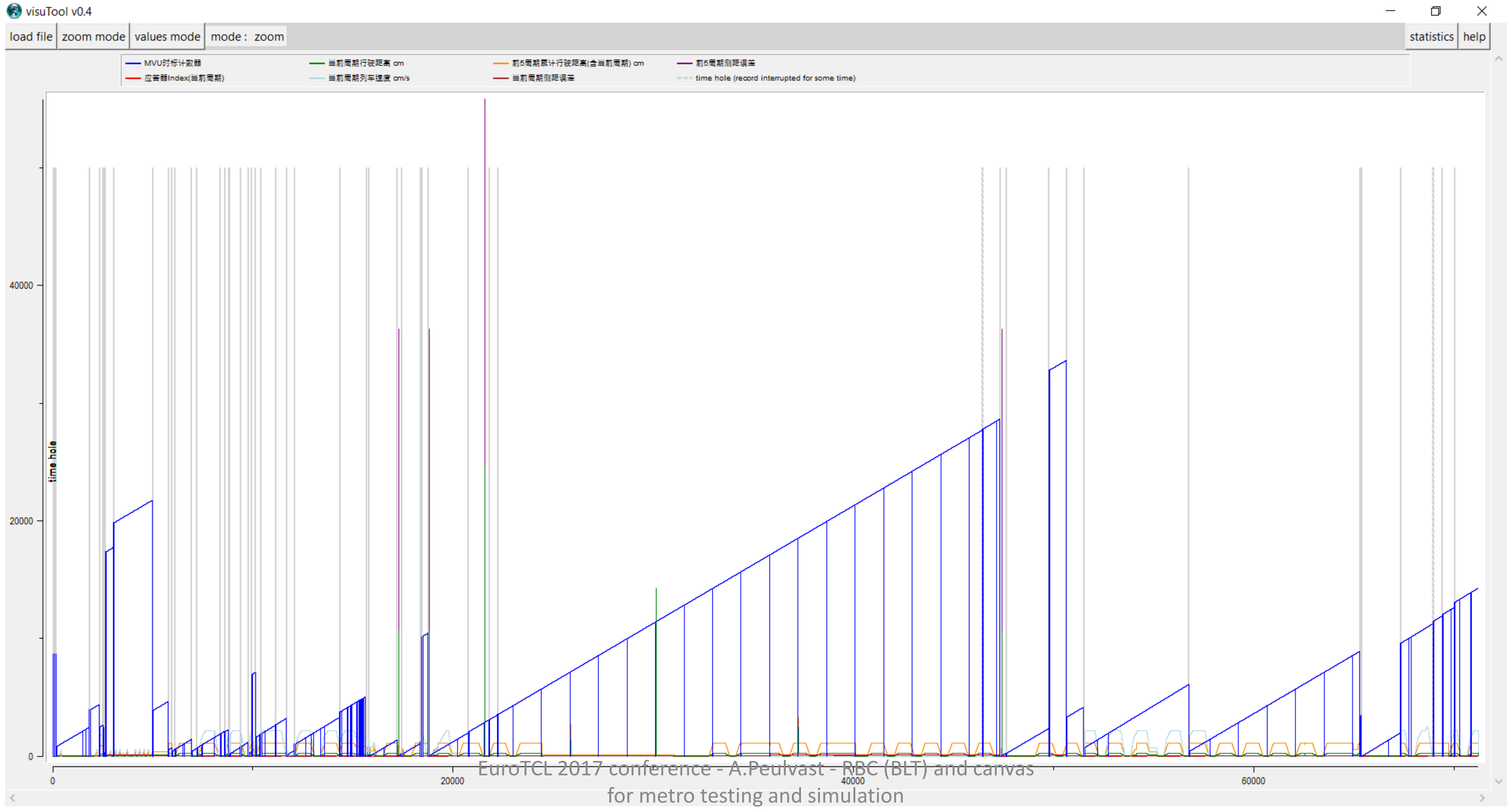


Vectors: expressions

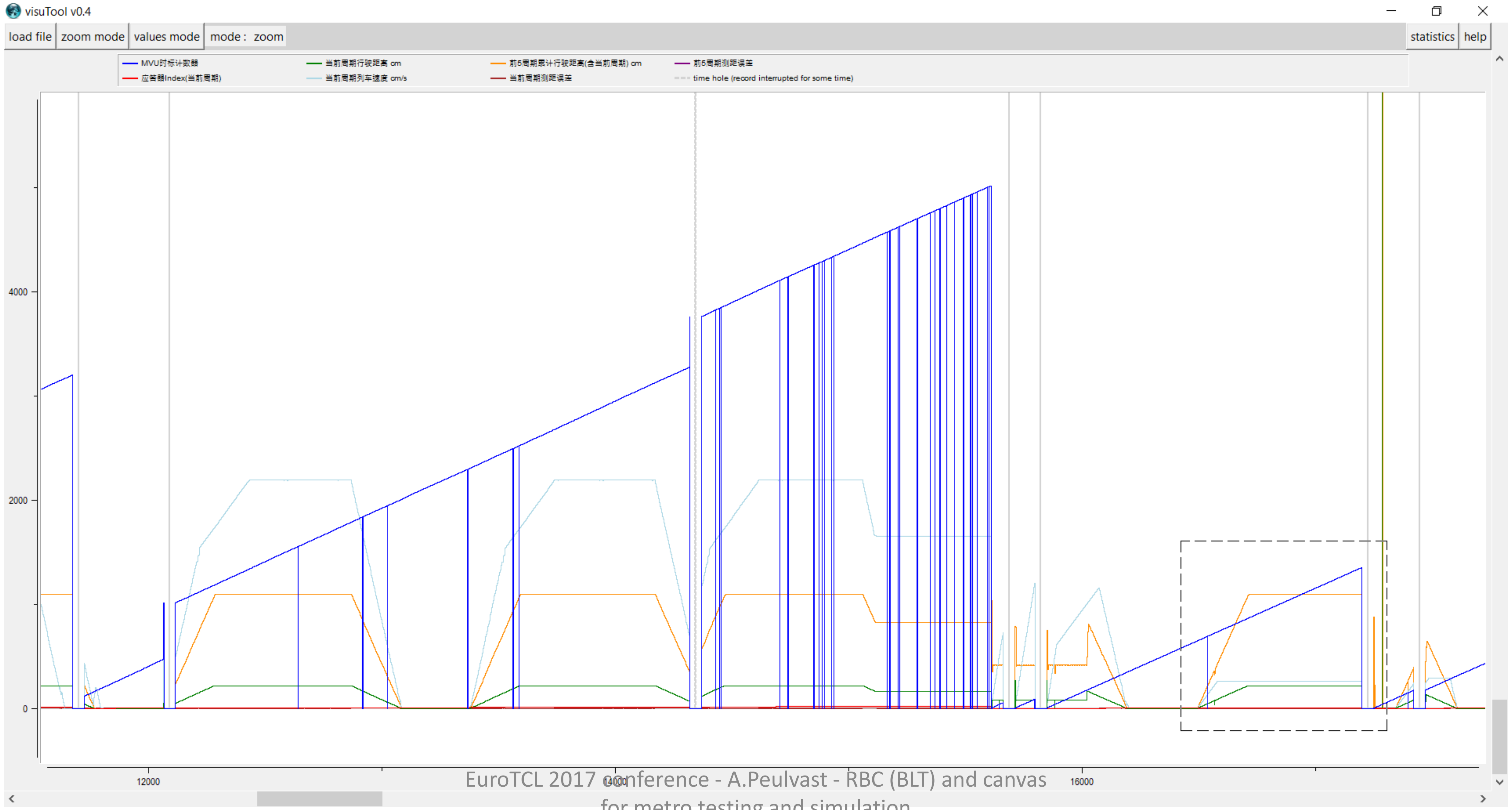
Vector's **expr** operation does both scalar and vector math.

• Arithmetic operators	`+- * / ^ %`
• Logic operators	`== != ! && || < > <= >=`
• Math functions	`abs acos asin atan ceil cos cosh exp floor hypot log log10 sin sinh sqrt tan tanh`
• Additional functions	`adev kurtosis length max mean median min norm prod q1 q3 random round srandom sdev skew sort var`

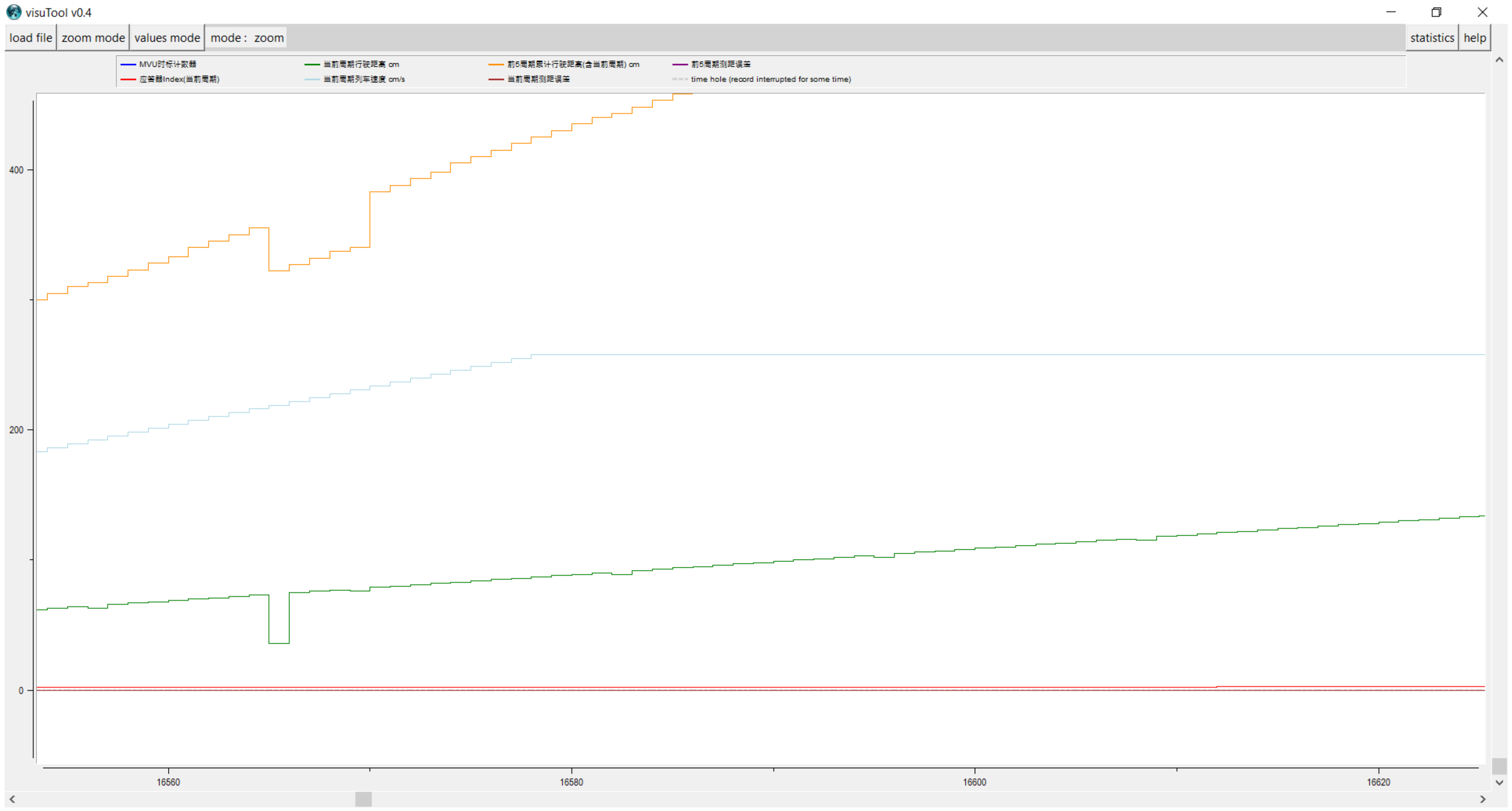
1) RBC use for Odometry data display



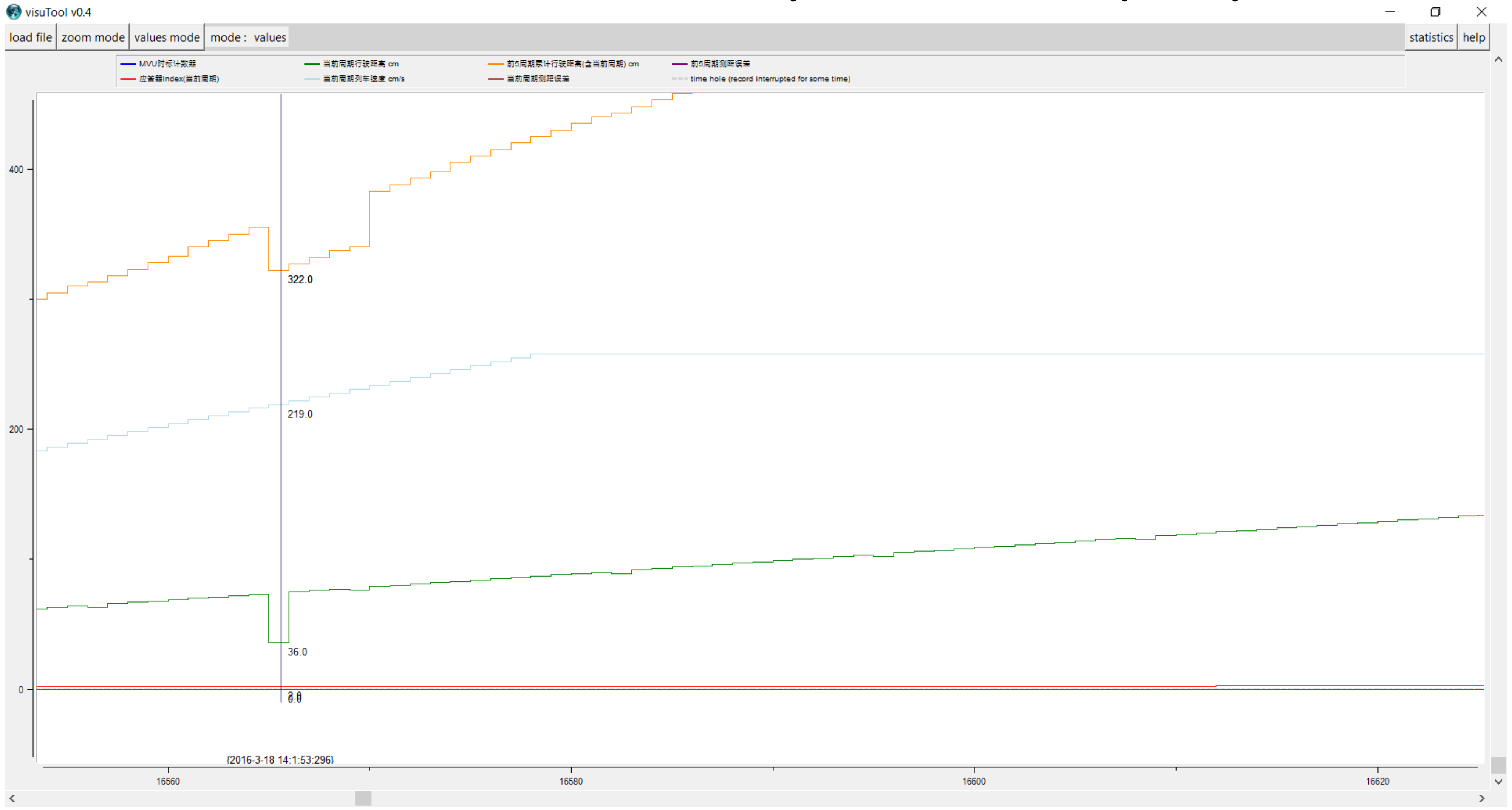
1) RBC use for Odometry data display



1) RBC use for Odometry data display



1) RBC use for Odometry data display



2) Use of canvas for metro simulation

- Did some tests using .Net or Qt to display a selection of data , but to obtain something usefull with Qt takes a lot of time to code, and whereas with .Net it is at a first glance quicker, then dealing with big amount of data is difficult : application reactivity is poor when trying to browse the data, scroll, filter, etc.

- Tcl/Tk : Tcl for the simulation core, in relation with Tk widgets and canvas for the display.

2) Use of canvas for metro simulation

- Simulation core :

•Topology configuration (line gradients)
•Models simulation
  -Each model is included in a global execution loop (based on an “every” loop, which is based on the “after” command)
  -Each model has a dedicated array for inputs and outputs
•GUI
  -Uses « -textvariable » option in widgets and « trace variable » mechanism in canvas
•Scenario running capabilities
  -Real time mode using “after xxx”
  -Cycles based mode using control on model cycles
  -Scenarios has set and check commands to set inputs and to check output values.

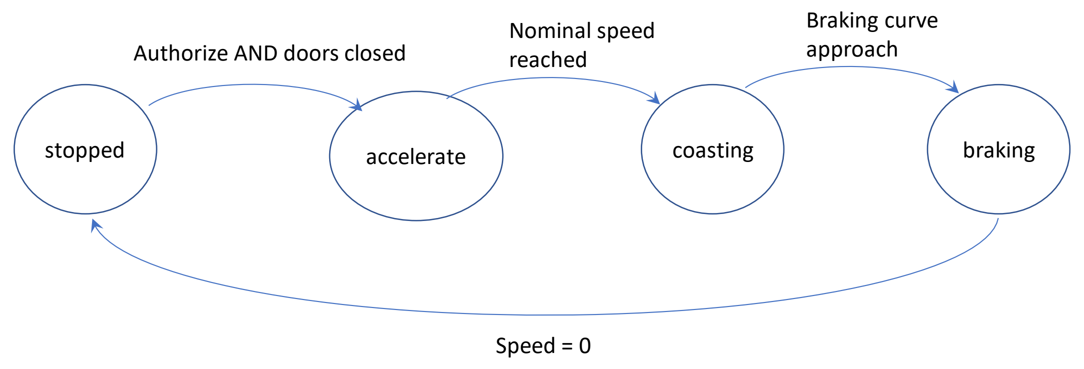
2) Use of canvas for metro simulation

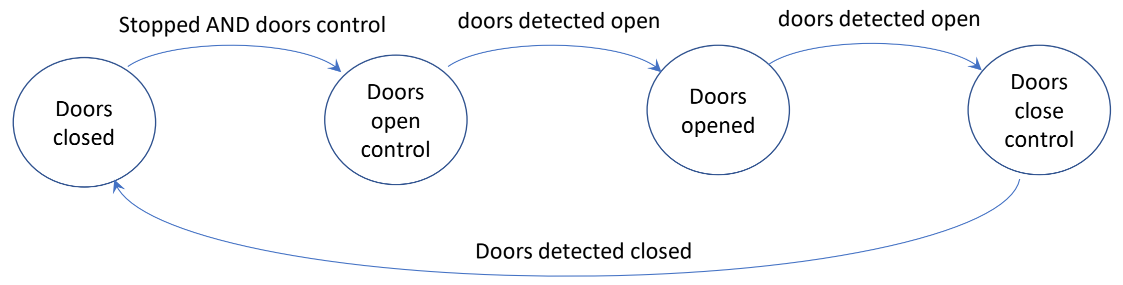
- Simulation core :
  - PLC control (through MODBUS protocol) to control 2 DC motors, that reproduce direction and train wheels rotation speed according to the simulator controls (through rolling stock model), the train wheel diameter, and some simulated failures.

  - Models :
    - Rolling stock model, which simulate the behaviour according to physics laws (gravity, track gradient, air resistance, acceleration, braking)
    - On-board ATP model, which monitors the train speed and check that the train never reach an overspeed limit
    - On-board ATO model, which drives the train automatically according to the train mission (start, brake, stop in stations, open the doors, etc.)
    - Trackside model, which gives track information about gradients, balises, platforms, etc.

2) Use of canvas for metro simulation

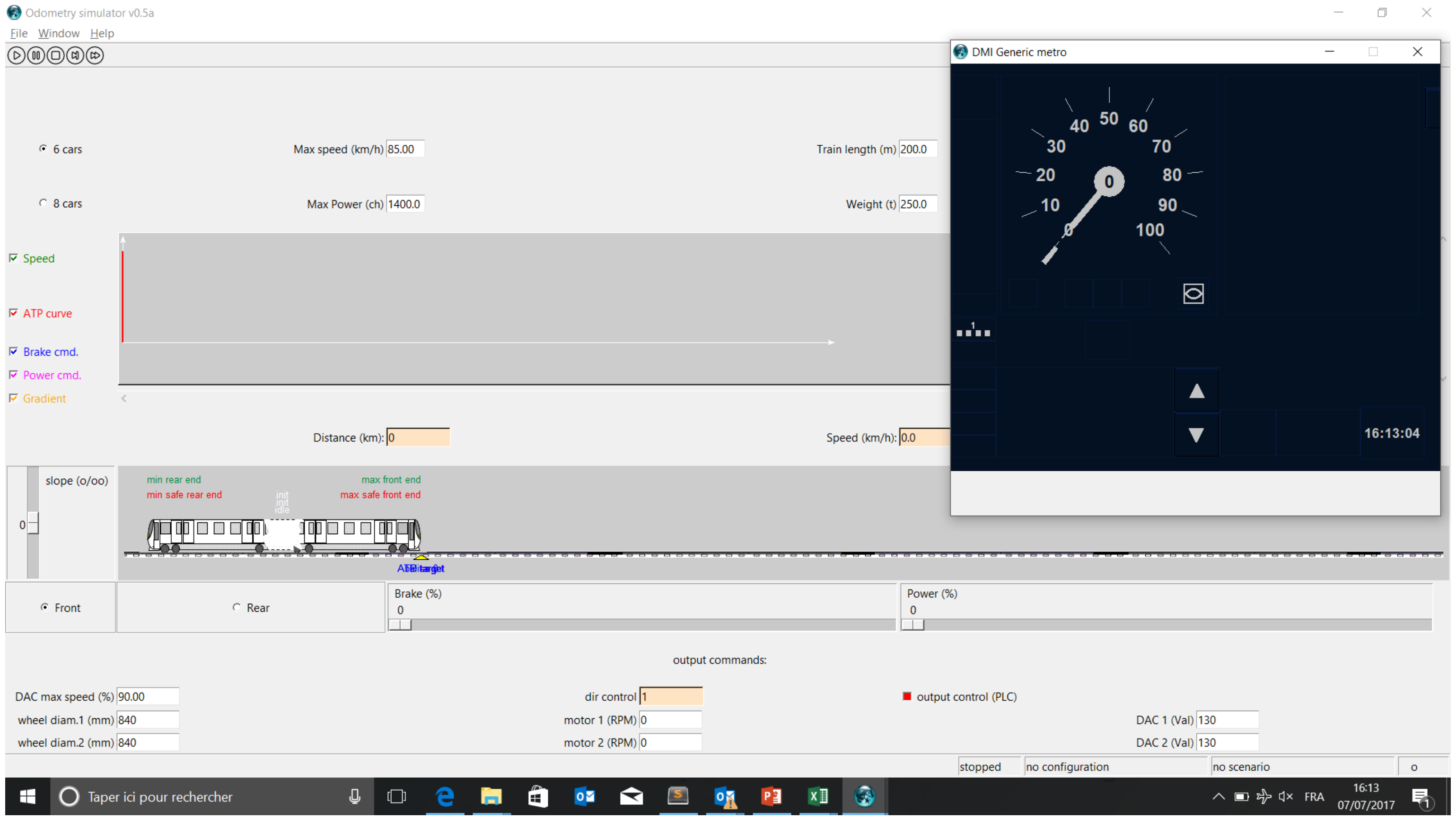
The current status is that simple models are implemented. But to go further, I need to implement state machines in order to put the model in different states to each specific operations can be performed. For example for the train control phases, and the train doors management :





2) Use of canvas for metro simulation

- Tk widgets used with arrays as « -textvariable » option
- Canvas for graph and train animation
- DMI : needle animation following simulation arrays
- Configuration of metro line topology and running of scenarios
- Simulation models (train, on-board automatic train protection, on-board automatic train operation) sharing arrays for variables update
- Lightweight, powerfull tool for training and design analysis



(• 6 cars) Max speed (km/h) 85.00 Train length (m) 200.0

(○ 8 cars) Max Power (ch) 1400.0 Weight (t) 250.0

Distance (km): 0 Speed (km/h): 0.0

slope (o/oo) 0

(• Front) (○ Rear) Brake (%) 0 Power (%) 0

output commands:

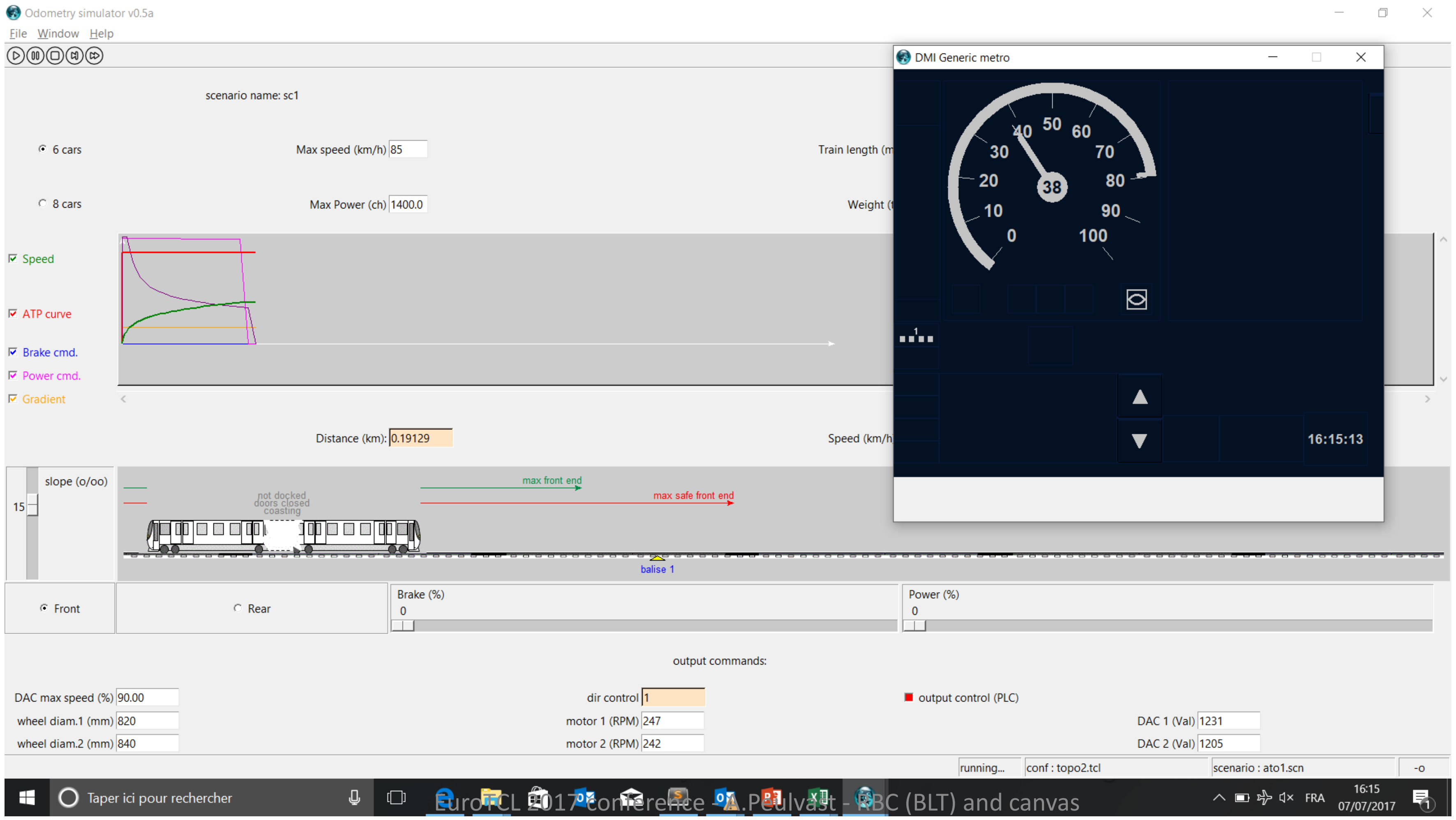
DAC max speed (%) 90.00 dir control 1 🟥 output control (PLC)

wheel diam.1 (mm) 840 motor 1 (RPM) 0 DAC 1 (Val) 130

wheel diam.2 (mm) 840 motor 2 (RPM) 0 DAC 2 (Val) 130

stopped | no configuration | no scenario | o

2) Use of canvas for metro simulation



scenario name: sc1

(● 6 cars) ○ 8 cars

Max speed (km/h) 85

Max Power (ch) 1400.0

Train length (m

Weight (

Distance (km): 0.19129

Speed (km/h

slope (o/oo) 15

(● Front) ○ Rear

Brake (%) 0

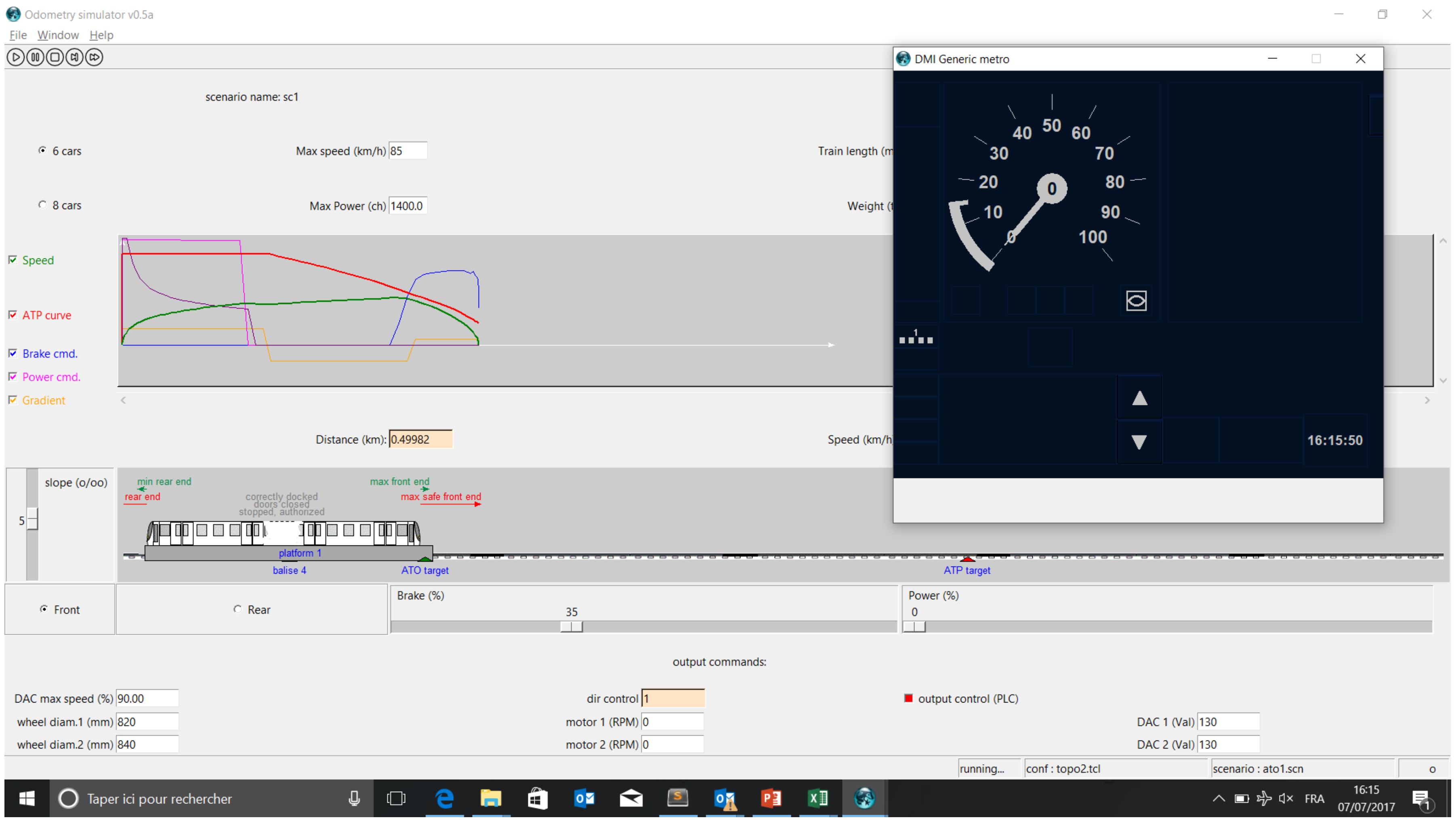
Power (%) 0

output commands:

DAC max speed (%)	90.00	dir control	1	🟥 output control (PLC)	
wheel diam.1 (mm)	820	motor 1 (RPM)	247	DAC 1 (Val)	1231
wheel diam.2 (mm)	840	motor 2 (RPM)	242	DAC 2 (Val)	1205

running... | conf : topo2.tcl | scenario : ato1.scn | -o

2) Use of canvas for metro simulation


Odometry simulator v0.5a

File Window Help

scenario name: sc1

(● 6 cars) (○ 8 cars)

Max speed (km/h) 85

Max Power (ch) 1400.0

Train length (m

Weight (

☑ Speed

☑ ATP curve

☑ Brake cmd.

☑ Power cmd.

☑ Gradient

Distance (km): 0.49982

Speed (km/h

slope (o/oo)

(● Front) (○ Rear)

Brake (%) 35

Power (%) 0

output commands:

DAC max speed (%) 90.00

wheel diam.1 (mm) 820

wheel diam.2 (mm) 840

dir control 1

motor 1 (RPM) 0

motor 2 (RPM) 0

output control (PLC)

DAC 1 (Val) 130

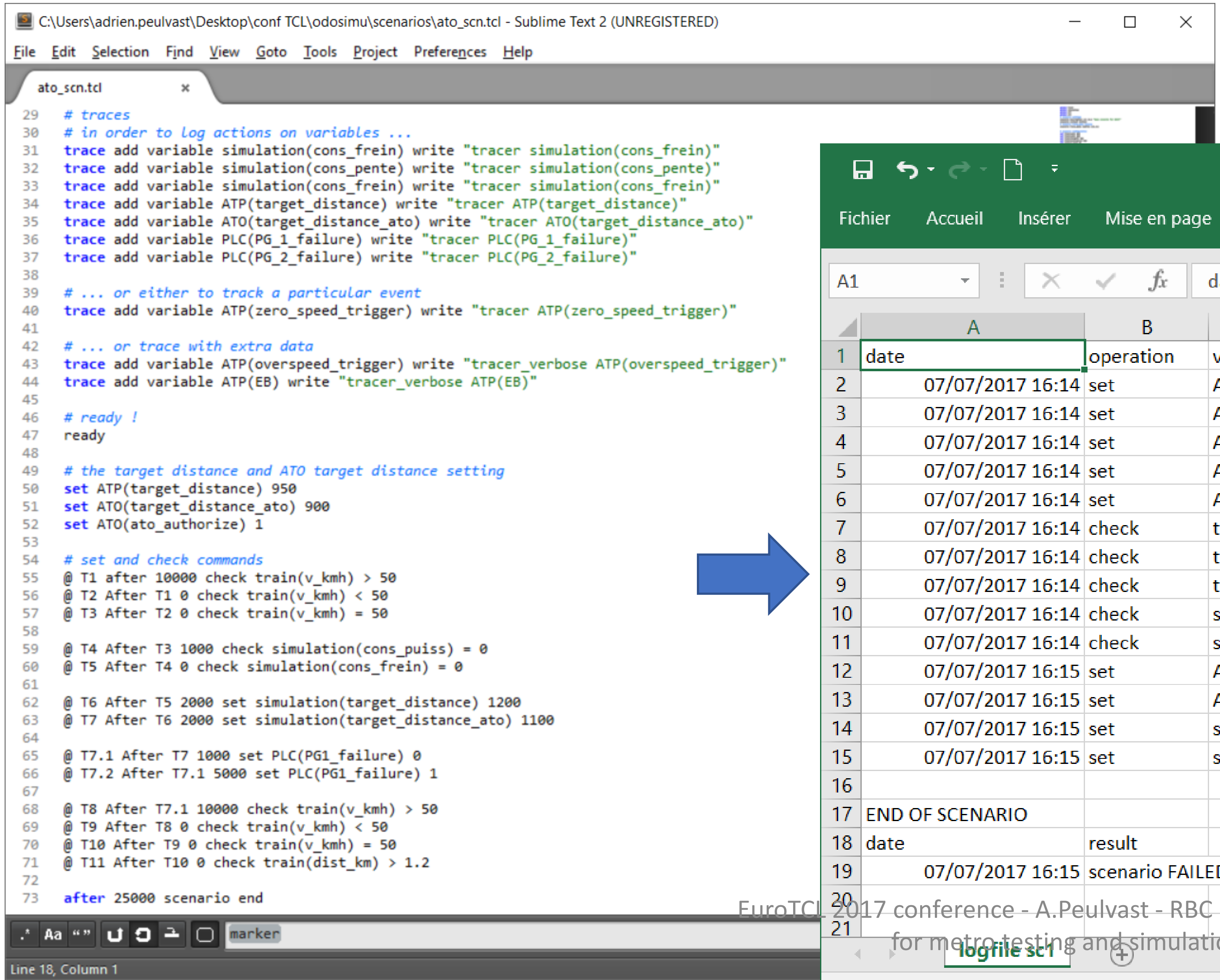
DAC 2 (Val) 130

running... conf : topo2.tcl scenario : ato1.scn

DMI Generic metro

16:15:50

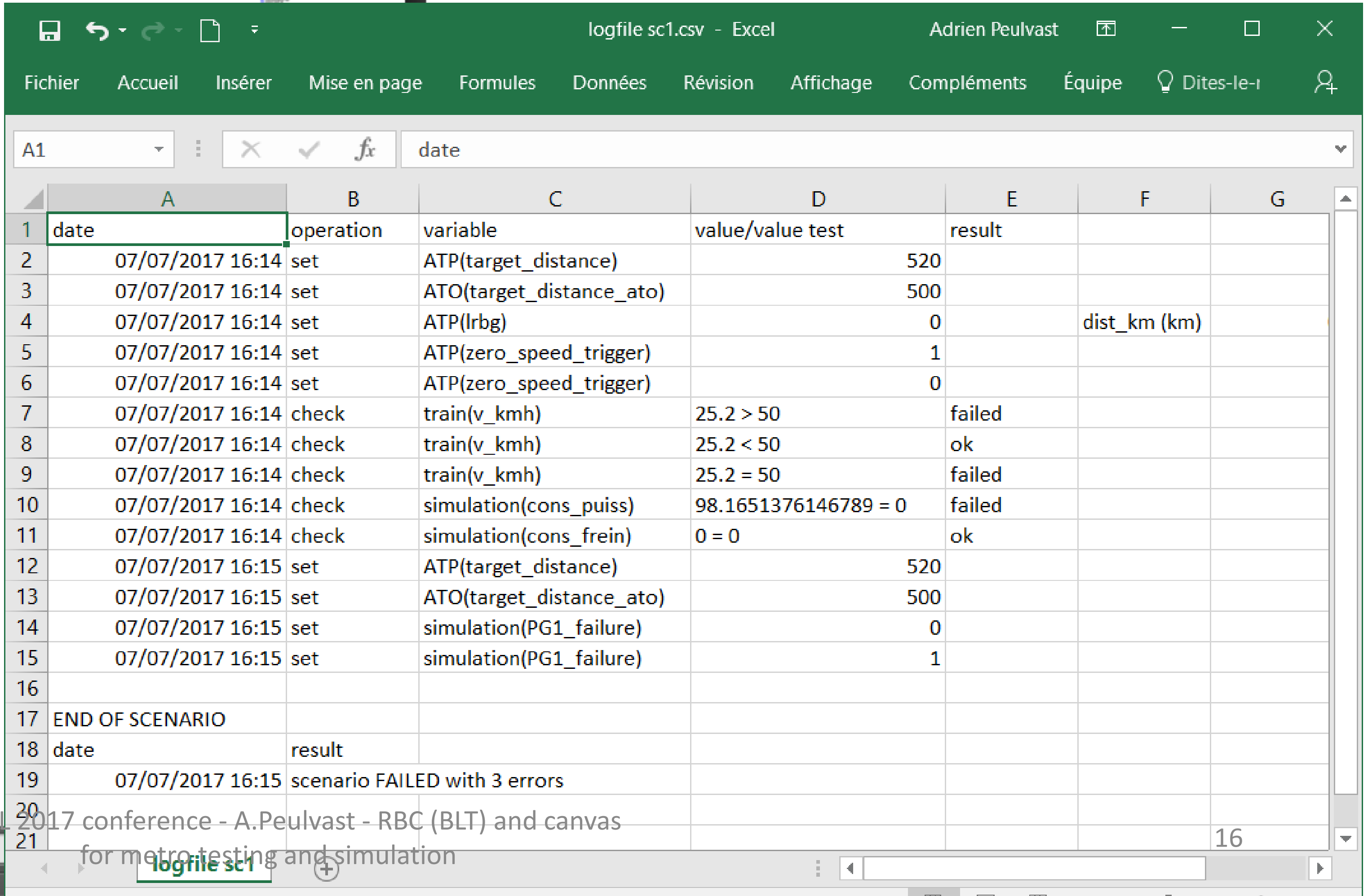
2) Use of canvas for metro simulation



C:\Users\adrien.peulvast\Desktop\conf TCL\odosimu\scenarios\ato_scn.tcl - Sublime Text 2 (UNREGISTERED)

ato_scn.tcl

```
# traces
# in order to log actions on variables ...
trace add variable simulation(cons_frein) write "tracer simulation(cons_frein)"
trace add variable simulation(cons_pente) write "tracer simulation(cons_pente)"
trace add variable simulation(cons_frein) write "tracer simulation(cons_frein)"
trace add variable ATP(target_distance) write "tracer ATP(target_distance)"
trace add variable ATO(target_distance_ato) write "tracer ATO(target_distance_ato)"
trace add variable PLC(PG_1_failure) write "tracer PLC(PG_1_failure)"
trace add variable PLC(PG_2_failure) write "tracer PLC(PG_2_failure)"

# ... or either to track a particular event
trace add variable ATP(zero_speed_trigger) write "tracer ATP(zero_speed_trigger)"

# ... or trace with extra data
trace add variable ATP(overspeed_trigger) write "tracer_verbose ATP(overspeed_trigger)"
trace add variable ATP(EB) write "tracer_verbose ATP(EB)"

# ready !
ready

# the target distance and ATO target distance setting
set ATP(target_distance) 950
set ATO(target_distance_ato) 900
set ATO(ato_authorize) 1

# set and check commands
@ T1 after 10000 check train(v_kmh) > 50
@ T2 After T1 0 check train(v_kmh) < 50
@ T3 After T2 0 check train(v_kmh) = 50

@ T4 After T3 1000 check simulation(cons_puiss) = 0
@ T5 After T4 0 check simulation(cons_frein) = 0

@ T6 After T5 2000 set simulation(target_distance) 1200
@ T7 After T6 2000 set simulation(target_distance_ato) 1100

@ T7.1 After T7 1000 set PLC(PG1_failure) 0
@ T7.2 After T7.1 5000 set PLC(PG1_failure) 1

@ T8 After T7.1 10000 check train(v_kmh) > 50
@ T9 After T8 0 check train(v_kmh) < 50
@ T10 After T9 0 check train(v_kmh) = 50
@ T11 After T10 0 check train(dist_km) > 1.2

after 25000 scenario end
```

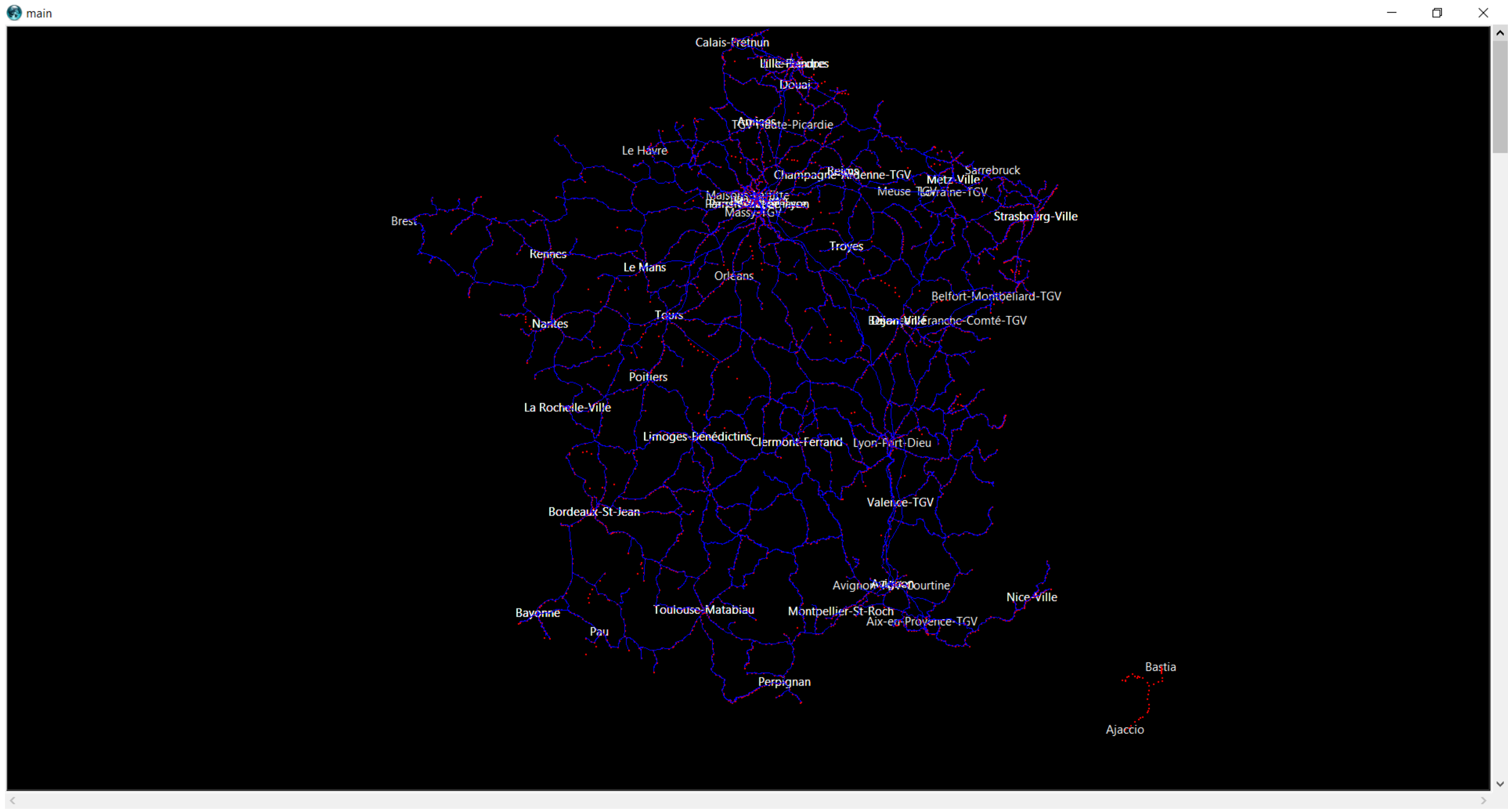


logfile sc1.csv - Excel

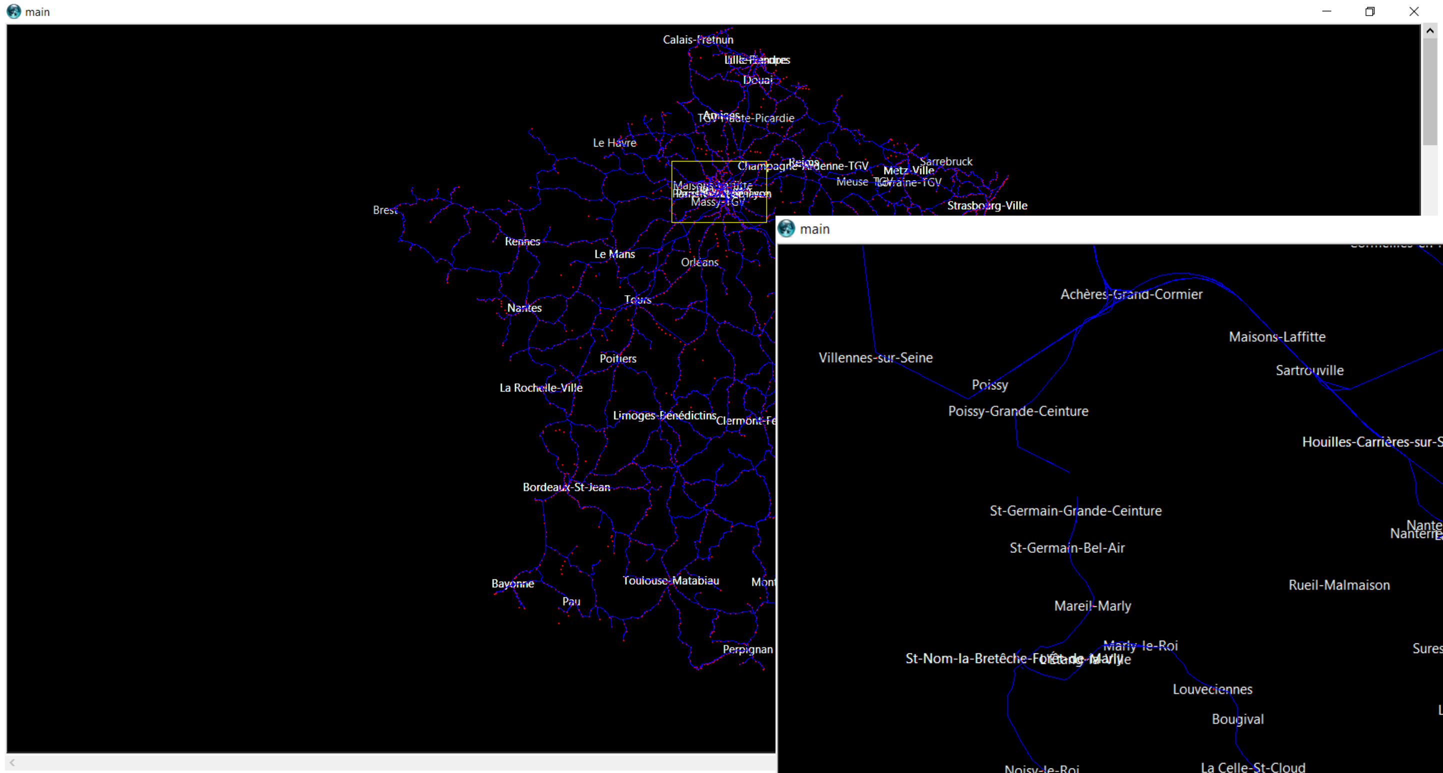
	A	B	C	D	E	F	G
1	date	operation	variable	value/value test	result		
2	07/07/2017 16:14	set	ATP(target_distance)	520			
3	07/07/2017 16:14	set	ATO(target_distance_ato)	500			
4	07/07/2017 16:14	set	ATP(lrbg)	0		dist_km (km)	
5	07/07/2017 16:14	set	ATP(zero_speed_trigger)	1			
6	07/07/2017 16:14	set	ATP(zero_speed_trigger)	0			
7	07/07/2017 16:14	check	train(v_kmh)	25.2 > 50	failed		
8	07/07/2017 16:14	check	train(v_kmh)	25.2 < 50	ok		
9	07/07/2017 16:14	check	train(v_kmh)	25.2 = 50	failed		
10	07/07/2017 16:14	check	simulation(cons_puiss)	98.1651376146789 = 0	failed		
11	07/07/2017 16:14	check	simulation(cons_frein)	0 = 0	ok		
12	07/07/2017 16:15	set	ATP(target_distance)	520			
13	07/07/2017 16:15	set	ATO(target_distance_ato)	500			
14	07/07/2017 16:15	set	simulation(PG1_failure)	0			
15	07/07/2017 16:15	set	simulation(PG1_failure)	1			
16							
17	END OF SCENARIO						
18	date	result					
19	07/07/2017 16:15	scenario FAILED with 3 errors					
20							
21							

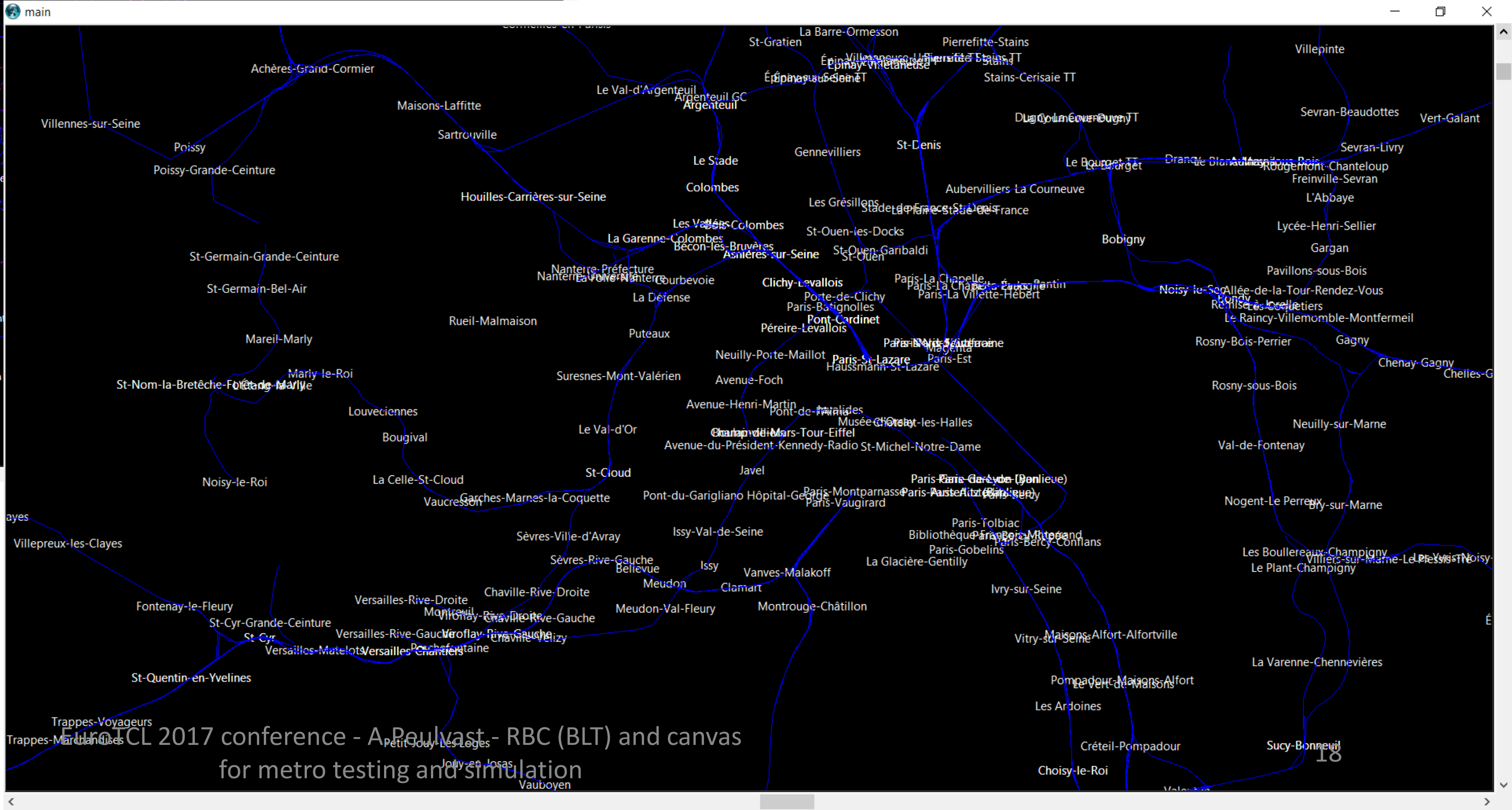
3) Open Data – SNCF data display

- Last year, SNCF released its network data in public domain
- Data provide all lines, stations, signalling objects (signals, level crossings, etc.)
- Open data : more and more operators (metro and railway) are publishing
- Thanks to the canvas, quite easy to display and control through tags



3) Open Data – SNCF data display





Conclusion

- For training, simulation and testing tools, using Tcl and Tk is efficient
  - Quick and incremental development, straight forward
    - Started small scripts, added features
    - Each tool is hundreds lines of code only
  - Ability to define and run test scenarios
    - Arrays as simulation data for loging and display using classic Tk widgets
  - Canvas capabilities !
  - BLT is a great lib for graphs !
  - …even for big data sets, which are more and more common as collecting data is easier and easier.

- The drawbacks
  - Language confidentiality
  - Documentation, support and maintenance (some libs are outdated)

- Next steps : I would like to spread the use of Tcl/Tk and share some good practices on the use of Tcl/Tk in simulation and testing