Written by: Andrea Ballanti - Student Intern, EPRI
Revised by: Roger Dugan –Sr. Technical Executive, Power System Studies, EPRI
March 2016 – OpenDSS 7.6.4.93

Property and Functions Visualization by COM: VBA, Python, and MATLAB

This Tech Note introduces the quite often hidden interfaces available in Python, MATLAB, and Excel-VBA to visualize the list of properties and functions of OpenDSS when connected by COM interface. This document is an integration of "OpenDSSComDoc" TechNote and requires the user to be familiar with the basic concepts on the COM interface introduced in the "User Manual" and the "Primer"[1].

The number of properties and functions available by COM interface is so large that is practically impossible to know all of them. Sometimes might be hard to remember the exact name of a property or function (e.g., DSSCircuit.Line.Length or DSSCircuit.Lines.Length?) while other times a user might need to know what properties are available for a specific class even before start coding in (e.g., what properties are available for a Line object?). This Tech Note addresses such problem by explaining how to use the interfaces available in VBA, Python, and MATLAB. Such interfaces might significantly speed up the use and familiarization process with the COM interface, especially for new users. It is worth noticing that, in some software, not all the properties and functions are listed (although most of them are). In doubt the OpenDSSComDoc Tech Note can be used.

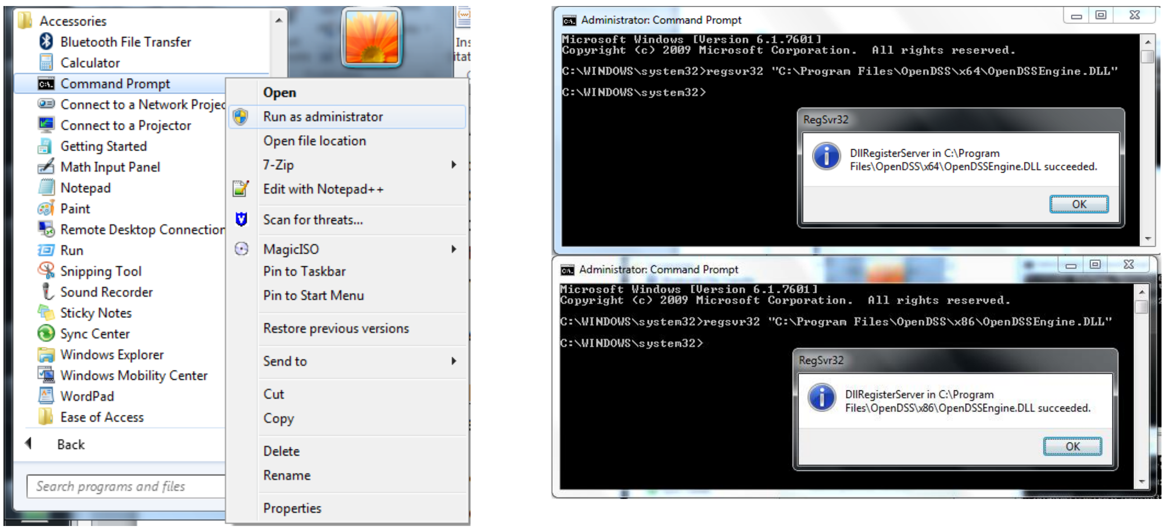
VBA Interface

EXCEL-VBA provides, among the three software here analyzed, the most complete and friendly interface to the OpenDSS object and functions. Indeed, it is not uncommon for users working in MATLAB or Python to use Excel-VBA only as an "OpenDSS dictionary" to have, on a side, all names of property and functions (indeed the name of property and functions are not dependent by the command that is driven OpenDSS). The list of OpenDSS properties and functions can be accessed by following the steps discussed below:

1) <u>Register the COM interface (Optional)</u>: Since version 7.6 OpenDSS is able to automatically register the COM interface for both 32 and 64 bit version[2], <u>hence, no user action is usually required</u>. However, if for any reason the COM interface has not been successfully registered, the following steps can be followed:
    a. Go to "Start" in Windows
    b. Go to the folder "Accessories"
    c. Right Click on "Command Prompt"
    d. Click on "Run as administrator" (Figure 1, left)
    e. Tape in it "regsvr32 OpenDSSPath\x64\OpenDSSEngine.DLL" (Figure 1, right)
    f. Press Enter (a confirmation window should appear, Figure 1, right)
    g. Tape in it "regsvr32 OpenDSSPath\x86\OpenDSSEngine.DLL" (Figure 1, right)
    h. Press Enter (a confirmation window should appear, Figure 1, right)

2) <u>Add OpenDSS library:</u> In this way EXCEL will load all the command to call OpenDSS. The following steps can be adopted:
    a. Open an EXCEL file
    b. Press Alt+F11 on the keyboard to open the VBA project window
    c. Go on "Tools-References" and tick the "OpenDSS Engine" as shown in Figure 2

[1] Both documents are available in either the computer Doc folder in which OpenDSS is installed and in the SourceForger repository at https://sourceforge.net/p/electricdss/code/HEAD/tree/trunk/Doc/
[2] Both versions are required as the COM server used is function of the calling program. Moreover, in a Windows PC it is not uncommon to have both 32 and 64 bit programs making the need to have 32 and 64 bit of OpenDSS installed as well



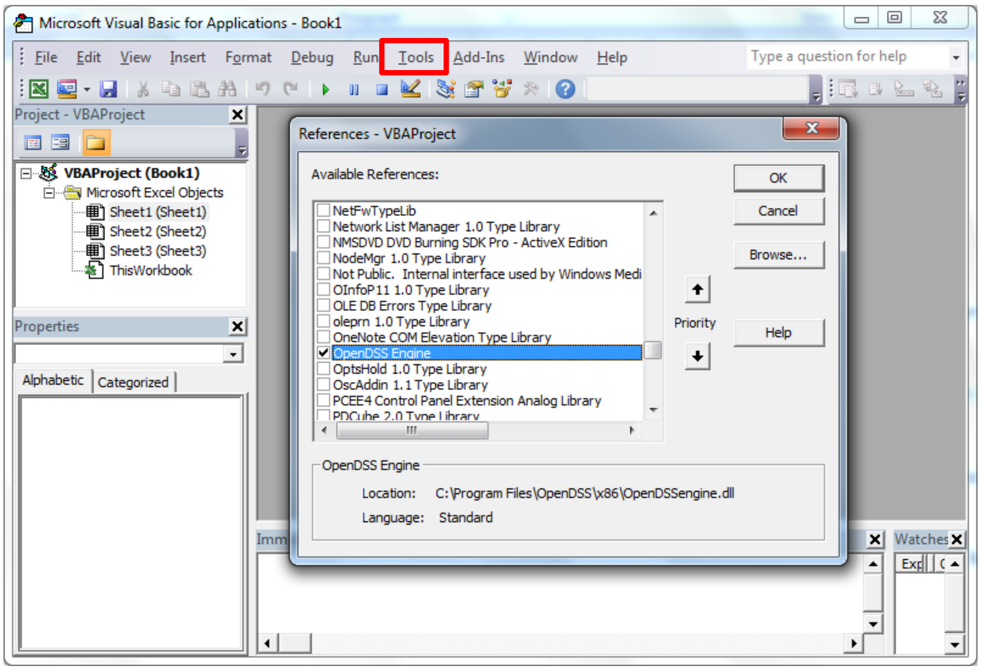
```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\WINDOWS\system32>regsvr32 "C:\Program Files\OpenDSS\x64\OpenDSSEngine.DLL"

C:\WINDOWS\system32>
```

DllRegisterServer in C:\Program Files\OpenDSS\x64\OpenDSSEngine.DLL succeeded.

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\WINDOWS\system32>regsvr32 "C:\Program Files\OpenDSS\x86\OpenDSSEngine.DLL"

C:\WINDOWS\system32>
```

DllRegisterServer in C:\Program Files\OpenDSS\x86\OpenDSSEngine.DLL succeeded.

Figure 1 Step 1: Register the COM interface (optional after OpenDSS version 7.6)



Available References:

- [ ] NetFwTypeLib
- [ ] Network List Manager 1.0 Type Library
- [ ] NMSDVD DVD Burning SDK Pro - ActiveX Edition
- [ ] NodeMgr 1.0 Type Library
- [ ] Not Public. Internal interface used by Windows Medi
- [ ] OInfoP11 1.0 Type Library
- [ ] OLE DB Errors Type Library
- [ ] oleprn 1.0 Type Library
- [ ] OneNote COM Elevation Type Library
- [✓] OpenDSS Engine
- [ ] OptsHold 1.0 Type Library
- [ ] OscAddin 1.1 Type Library
- [ ] PCEE4 Control Panel Extension Analog Library
- [ ] PDCube 2.0 Type Library

OpenDSS Engine
Location: C:\Program Files\OpenDSS\x86\OpenDSSengine.dll
Language: Standard

Figure 2 Step 2: Add OpenDSS library

3) Access to the VBA interface: All the classes, properties and functions can be accessed by:
    a. Going to "View" and "Object Browser"
    b. On the new opened window, at the place of <All Libraries> choose "OpenDSSengine" as shown in Figure 3-a.

   In the "Classes" and "Members" sub-windows the OpenDSS classes and corresponding properties and functions can be respectively found as shown in **Error! Reference source not found.**-b. Moreover, also a brief explanation on the selected property is shown in the lower window (**Error! Reference source not found.**-a).

Example: To understand how to use this interface, let's suppose that the length of the first line need to be saved as a variable called "LineLength". For this purpose, it is first necessary to find the "Lines" class in the class sub-window (left side in Figure 3-a). Then, by looking through all its property in the "Members of Lines" sub-window (right side in Figure 3-b) find the needed property ("First" and

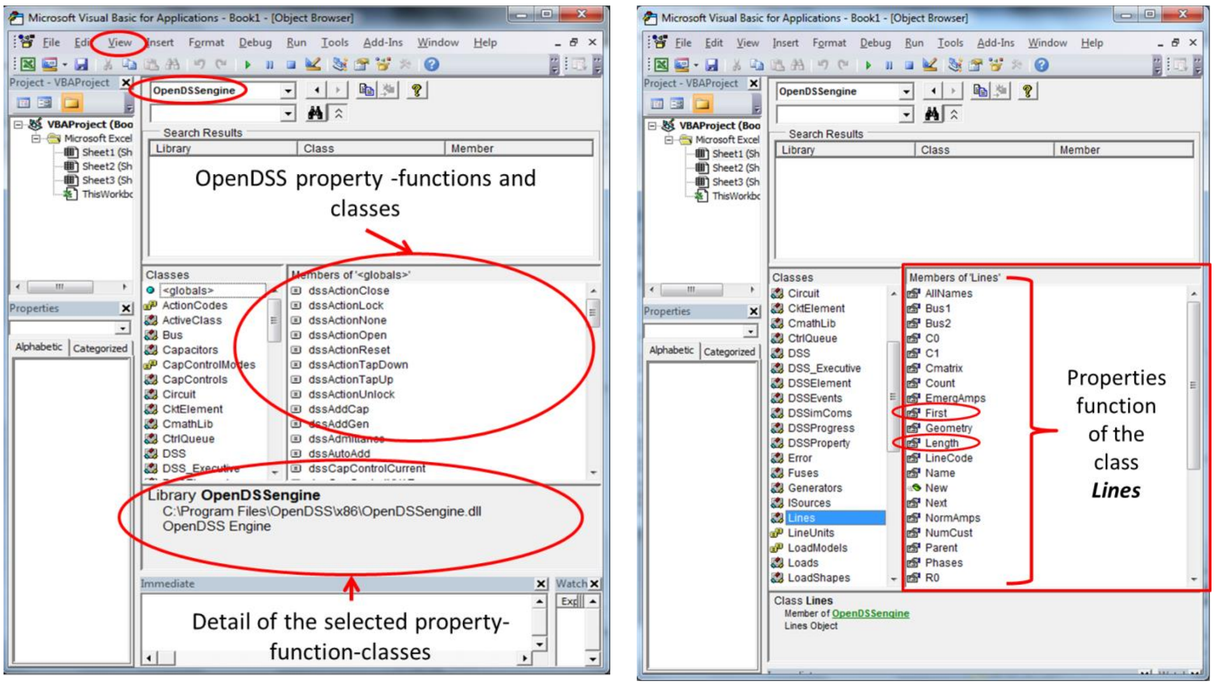
"Length" in this case). Consequently, after the OpenDSS COM interface has been instantiated by adopting the code for instance shown in **Error! Reference source not found.** (and discussed in more detail in the "Primer") the line interface can be created with the following command:

```
Dim DSSLines As OpenDSSengine.Lines
```

Then adopting the same spelling shown by VBA (Figure 3-b) the properties *first* and *length* are used:

```
        DSSLines.First
    LinesInterface.First
LineLength =LinesInterface.Length
```

The same identical process can be repeated for any other property of any OpenDSS class that can be explored. It is worth noticing that in VBA (differently from MATLAB and Python) it is not necessary to write and run the initialization code shown in Table 1 to only see the names of the property and functions.


OpenDSS property -functions and classes
Library OpenDSSengine
C:\Program Files\OpenDSS\x86\OpenDSSengine.dll
OpenDSS Engine
Detail of the selected property-function-classes
Members of 'Lines'
First
Length
Properties function of the class *Lines*
Class Lines
Member of OpenDSSengine
Lines Object

a) b)

Figure 3 a) Step 3: Access to the VBA interface b) List of properties for the Lines class ("First" and "Length" highlighted)

Table 1 Instantiated COM interface in VBA

```
' ****************************************************
' * Initialize OpenDSS
' ****************************************************
' Declare the OpenDSS related variables
Dim DSSObj As OpenDSSengine.DSS
Dim DSSText As OpenDSSengine.Text
Dim DSSCircuit As OpenDSSengine.Circuit
Dim DSSSolution As OpenDSSengine.Solution

' Instantiate the OpenDSS Object
DSSObj = New OpenDSSengine.DSS

' Start up the Solver
If Not DSSObj.Start(0) Then
    MsgBox("Unable to start the OpenDSS Engine")
    Return
End If
```

```
' *******************************************************
' * Define Circuit – Solve Power Flow
' *******************************************************
Etc…

' *******************************************************
' * Create the Line Interface
' *******************************************************
Dim Length as double
Dim DSSLines As OpenDSSengine.Lines
```

Python Interface

To visualize the Python interface a compiler is needed. More precisely, in this TechNote the free version of PyCharm is adopted. It is worth noticing, that differently from the previous discussed EXCEL VBA interface, in Python it is needed to run the instantiation code for the OpenDSS COM. In other words, it is first necessary to write a code as reported for simplicity in Table 1Table 2.

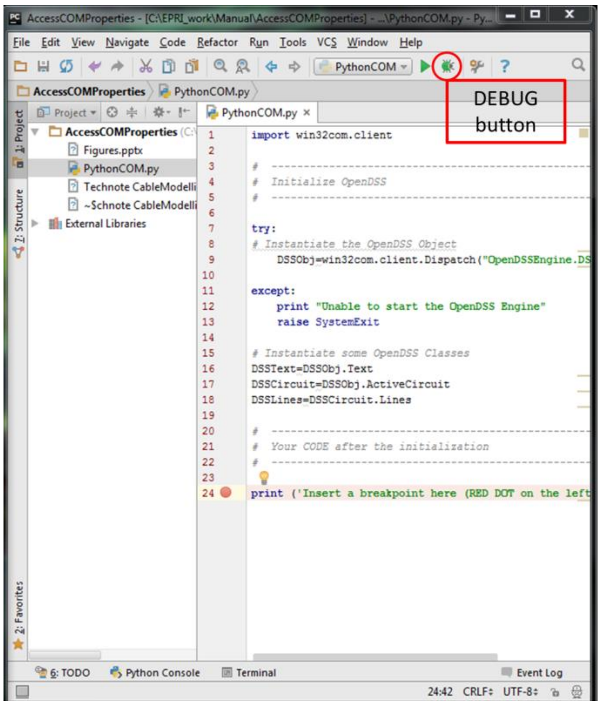
Table 2 Instantiated COM interface in Python

```
import win32com.client

#  ------------------------------------------------------------
#  Initialize OpenDSS
#  ------------------------------------------------------------
try:
#  Instantiate the OpenDSS Object
    DSSObj=win32com.client.Dispatch("OpenDSSEngine.DSS")
except:
    print "Unable to start the OpenDSS Engine"
    raise SystemExit

#  ------------------------------------------------------------
#  Create the Line Interface
#  ------------------------------------------------------------

# Instantiate some OpenDSS Classes by creating some interfaces
DSSText=DSSObj.Text
DSSCircuit=DSSObj.ActiveCircuit
DSSLines=DSSCircuit.Lines
```

It is then possible to access to the list of properties by following the steps described below:

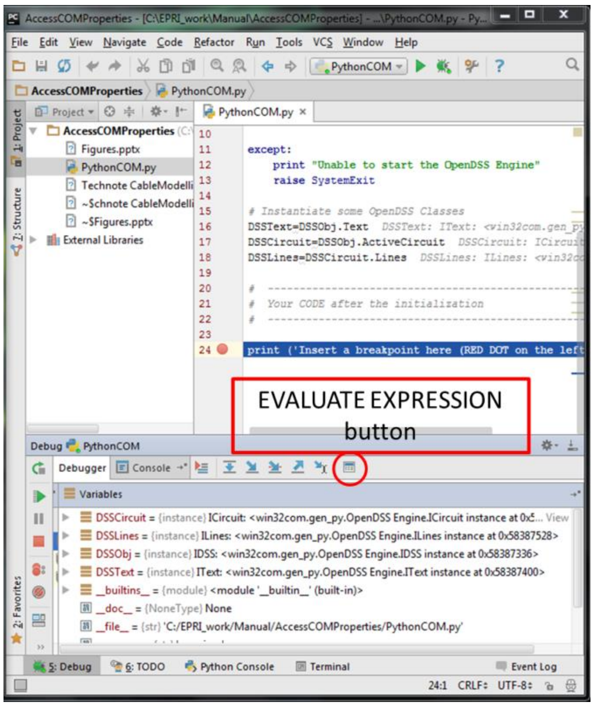
1. Insert a <u>debug breakpoint</u> at any point after the OpenDSS initialization (red dot in Figure 4-a)
2. <u>Run</u> the code in <u>Debug mode</u> (by pressing the button on the top of the screen, in red in Figure 4-a)
3. Once the code stopped open the “Evaluate Expression” window (by pressing the button, in red in Figure 4-b). Tape the name of one of the interface created in the initialization (such as DSSLines) and press “Evaluate” button (in Figure 4-c)
4. On the “Result” area select the “**_prop_map_get**” (shown in Figure 4-c)and all the properties corresponding to DSSLines (in this case) will be displayed (shown in Figure 4-d)

It is worth noticing that the properties are divided in two categories. Those that can be read (displayed by “**_prop_map_get**”) and those at which a value can be assigned (displayed by “**_prop_map_put”**).



```
import win32com.client

# -----------------------------------------------
#  Initialize OpenDSS
# -----------------------------------------------

try:
# Instantiate the OpenDSS Object
    DSSObj=win32com.client.Dispatch("OpenDSSEngine.DS

except:
    print "Unable to start the OpenDSS Engine"
    raise SystemExit

# Instantiate some OpenDSS Classes
DSSText=DSSObj.Text
DSSCircuit=DSSObj.ActiveCircuit
DSSLines=DSSCircuit.Lines

# -----------------------------------------------
#  Your CODE after the initialization
# -----------------------------------------------

print ('Insert a breakpoint here (RED DOT on the left
```

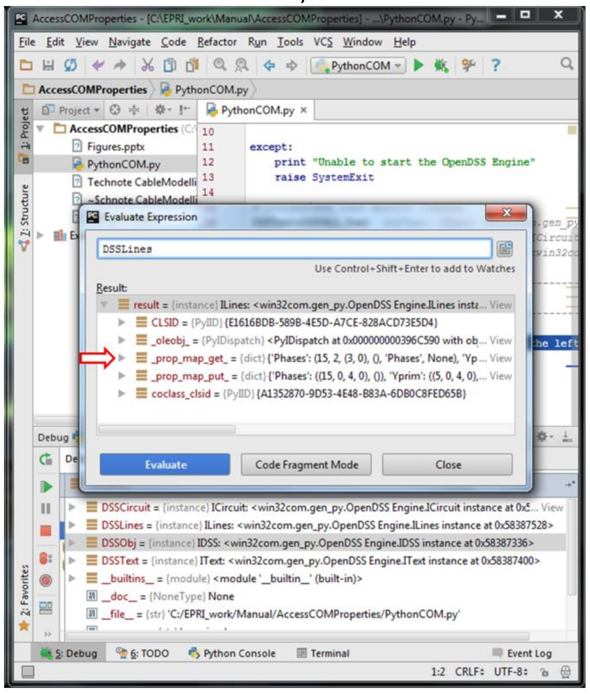
a)



```
except:
    print "Unable to start the OpenDSS Engine"
    raise SystemExit

# Instantiate some OpenDSS Classes
DSSText=DSSObj.Text  DSSText: IText: <win32com.gen_py
DSSCircuit=DSSObj.ActiveCircuit  DSSCircuit: ICircuit
DSSLines=DSSCircuit.Lines  DSSLines: ILines: <win32co

# -----------------------------------------------
#  Your CODE after the initialization
# -----------------------------------------------

print ('Insert a breakpoint here (RED DOT on the left
```

```
DSSCircuit = {instance} ICircuit: <win32com.gen_py.OpenDSS Engine.ICircuit instance at 0x5... View
DSSLines = {instance} ILines: <win32com.gen_py.OpenDSS Engine.ILines instance at 0x58387528>
DSSObj = {instance} IDSS: <win32com.gen_py.OpenDSS Engine.IDSS instance at 0x58387336>
DSSText = {instance} IText: <win32com.gen_py.OpenDSS Engine.IText instance at 0x58387400>
__builtins__ = {module} <module '__builtin__' (built-in)>
__doc__ = {NoneType} None
__file__ = {str} 'C:/EPRI_work/Manual/AccessCOMProperties/PythonCOM.py'
```

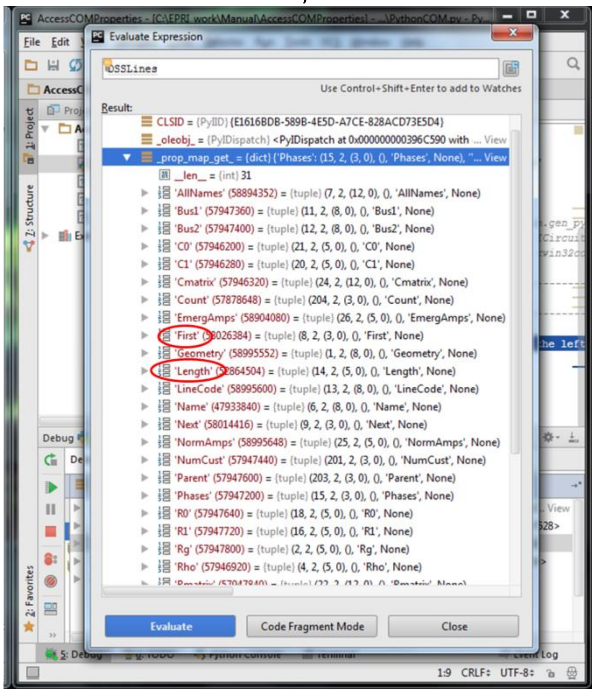
b)



```
DSSLines
```

```
result = {instance} ILines: <win32com.gen_py.OpenDSS Engine.ILines insta... View
    CLSID = {PyIID} {E1616BDB-589B-4E5D-A7CE-828ACD73E5D4}
    _oleobj_ = {PyIDispatch} <PyIDispatch at 0x000000000396C590 with ob... View
    _prop_map_get_ = {dict} {'Phases': (15, 2, (3, 0), (), 'Phases', None), 'Yp... View
    _prop_map_put_ = {dict} {'Phases': ((15, 0, 4, 0), ()), 'Yprim': ((5, 0, 4, 0),... View
    coclass_clsid = {PyIID} {A1352870-9D53-4E48-B83A-6DB0C8FED65B}
```

c)



```
DSSLines
```

```
CLSID = {PyIID} {E1616BDB-589B-4E5D-A7CE-828ACD73E5D4}
_oleobj_ = {PyIDispatch} <PyIDispatch at 0x000000000396C590 with ... View
_prop_map_get_ = {dict} {'Phases': (15, 2, (3, 0), (), 'Phases', None), "... View
    __len__ = {int} 31
    'AllNames' (58894352) = {tuple} (7, 2, (12, 0), (), 'AllNames', None)
    'Bus1' (57947360) = {tuple} (11, 2, (8, 0), (), 'Bus1', None)
    'Bus2' (57947400) = {tuple} (12, 2, (8, 0), (), 'Bus2', None)
    'C0' (57946200) = {tuple} (21, 2, (5, 0), (), 'C0', None)
    'C1' (57946280) = {tuple} (20, 2, (5, 0), (), 'C1', None)
    'Cmatrix' (57946320) = {tuple} (24, 2, (12, 0), (), 'Cmatrix', None)
    'Count' (57878648) = {tuple} (204, 2, (3, 0), (), 'Count', None)
    'EmergAmps' (58904080) = {tuple} (26, 2, (5, 0), (), 'EmergAmps', None)
    'First' (58026384) = {tuple} (8, 2, (3, 0), (), 'First', None)
    'Geometry' (58995552) = {tuple} (1, 2, (8, 0), (), 'Geometry', None)
    'Length' (52864504) = {tuple} (14, 2, (5, 0), (), 'Length', None)
    'LineCode' (58995600) = {tuple} (13, 2, (8, 0), (), 'LineCode', None)
    'Name' (47933840) = {tuple} (6, 2, (8, 0), (), 'Name', None)
    'Next' (58014416) = {tuple} (9, 2, (3, 0), (), 'Next', None)
    'NormAmps' (58995648) = {tuple} (25, 2, (5, 0), (), 'NormAmps', None)
    'NumCust' (57947440) = {tuple} (201, 2, (3, 0), (), 'NumCust', None)
    'Parent' (57947600) = {tuple} (203, 2, (3, 0), (), 'Parent', None)
    'Phases' (57947200) = {tuple} (15, 2, (3, 0), (), 'Phases', None)
    'R0' (57947640) = {tuple} (18, 2, (5, 0), (), 'R0', None)
    'R1' (57947720) = {tuple} (16, 2, (5, 0), (), 'R1', None)
    'Rg' (57947800) = {tuple} (2, 2, (5, 0), (), 'Rg', None)
    'Rho' (57946920) = {tuple} (4, 2, (5, 0), (), 'Rho', None)
```

d)

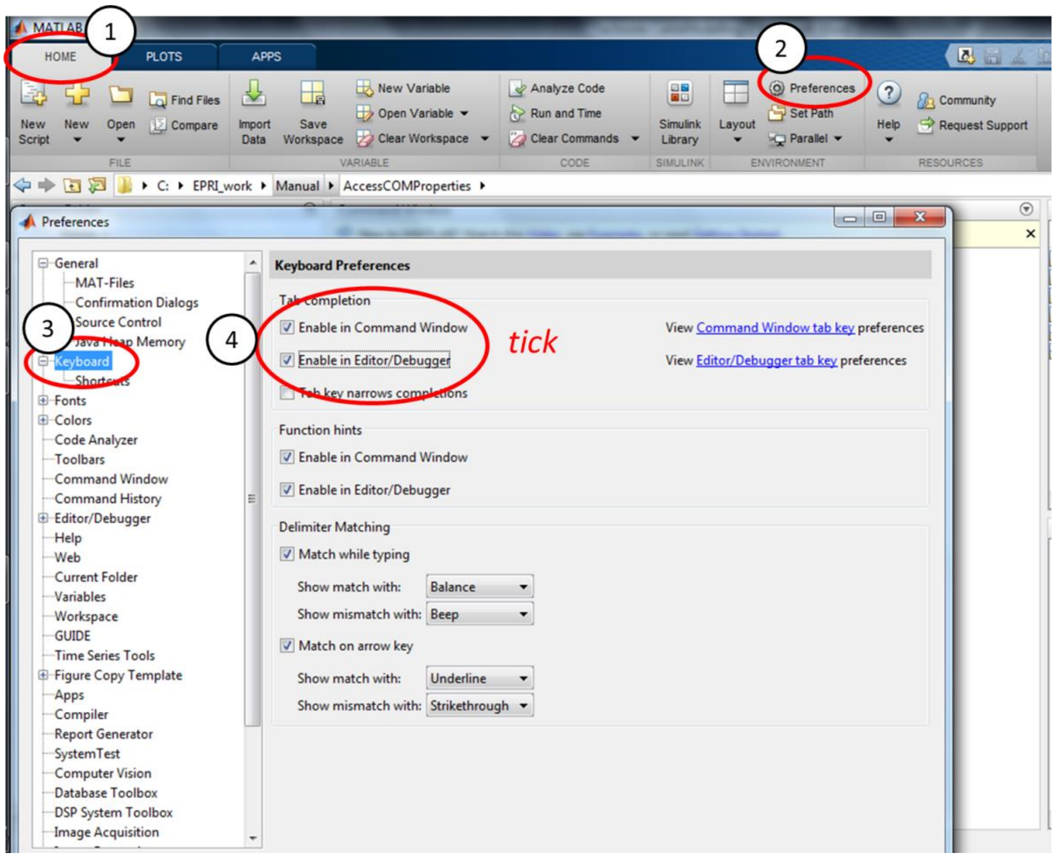
Figure 4 a) Step 1: Insert a debug breakpoint b) Step 2: Run in Debug Mode c) Step 3: Select "_prop_map_get" d) Step 4: List of all the properties of the class Lines with "First" and "Length" highlighted

Consequently, by the name of the properties visualized by the Python compiler in Figure 4-d it is possible to know the exact spelling (as well the whole list) of the commands necessary (for this example) to select the first line and extract the length as shown below:

```
DSSLines.First
LineLength=DSSLines.Length
```

MATLAB INTERFACE

Similarly to Python, MATLAB[3] provides an interface to see all the properties associated with a class object in OpenDSS. However, before writing any code it is necessary unlocking a feature of MATLAB (if not already activated) called "Tab Completion". For this purpose, within the tab "home" (1 in Figure 5), it is necessary to access the "Preferences" section (2 in Figure 5). Then, in the sub-section "Keyboard" (3 in Figure 5) the options "Enable in Command Window" and "Enable in Editor/Debugger" can be activated (tick them as shown in 4. in Figure 5).



Keyboard Preferences

Tab completion
- ☑ Enable in Command Window
- ☑ Enable in Editor/Debugger
- ☐ Tab key narrows completions

View Command Window tab key preferences
View Editor/Debugger tab key preferences

Function hints
- ☑ Enable in Command Window
- ☑ Enable in Editor/Debugger

Delimiter Matching
- ☑ Match while typing
  - Show match with: Balance
  - Show mismatch with: Beep
- ☑ Match on arrow key
  - Show match with: Underline
  - Show mismatch with: Strikethrough

Figure 5 Activation of the feature "Tap Completion" in MATLAB

Once the "Tab Completion" feature is activated it is then necessary to initialize the COM interface (as in Python) writing and running the code reported in Table 3.

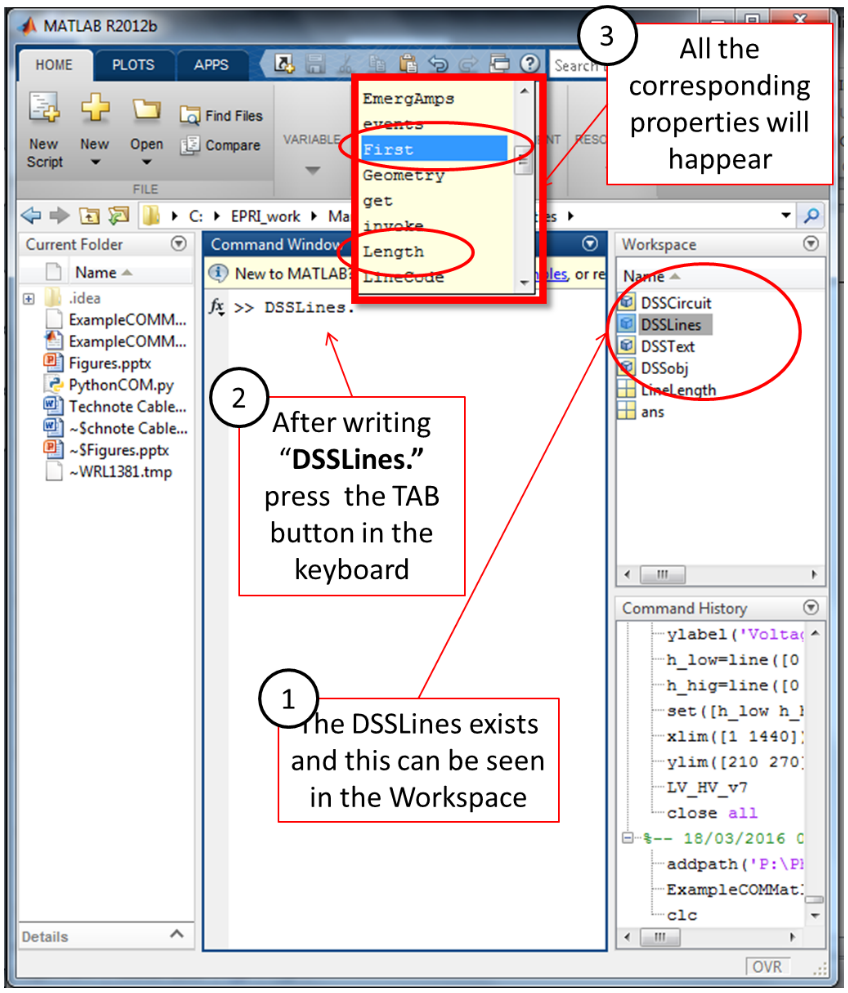
Once the interfaces have been created (i.e., the initialization code has been run and the interfaces so produced can be seen in the Workspace, 1 in Figure 6) in the "command window" it is necessary to insert the name of the class (whose properties list is needed to know) followed by a dot ("DSSLines." in this example as shown in 2 in Figure 6). The dot at the end tells MATLAB that, associated to this class, there might be properties. Then, leveraging the "Tab Completion" functionality previously activated, it is sufficient to press the "Tab" button on the keyboard and all the corresponding properties are visualized as shown in 3 in Figure 6. Consequently, it is possible to know what commands are available and (for this example) necessary to select the first line and extract the length of it as shown below:

```
DSSLines.First
LineLength=DSSLines.Length
```

[3] In this TechNote the version 2012b is adopted. However, similar commands should be present also in previous and next release of the software

Table 3 Instantiated COM interface in MATLAB

```
% ----------------------------------------------------------
% Initialize OpenDSS
% ----------------------------------------------------------
% Instantiate the OpenDSS Object
DSSobj = actxserver('OpenDSSEngine.DSS');
% Start up the Solver
if ~DSSobj.Start(0),
 disp('Unable to start the OpenDSS Engine')
return
end

% ----------------------------------------------------------
% Create the Line Interface
% ----------------------------------------------------------

% Instantiate some OpenDSS Classes by creating some interfaces
DSSText = DSSobj.Text;
DSSCircuit = DSSobj.ActiveCircuit;
DSSLines=DSSCircuit.Lines;
```



```
>> DSSLines.
```

```
ylabel('Volta
h_low=line([0
h_hig=line([0
set([h_low h_
xlim([1 1440]
ylim([210 270
LV_HV_v7
close all
%-- 18/03/2016
addpath('P:\P
ExampleCOMMat
clc
```

Figure 6 Visualization in MATLAB of the List of properties for the Lines class (with "First" and "Length" highlighted)