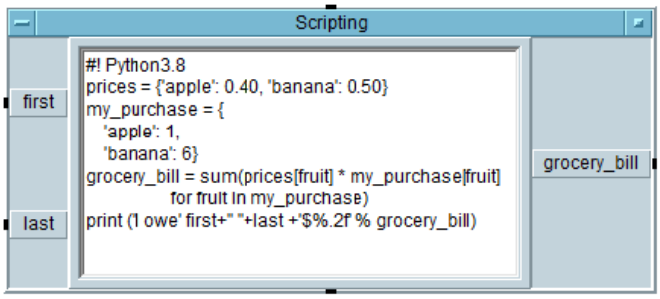
Script Interface

The purpose of this project is to provide a scripting object that allows external scripting languages to be incorporated in VEE programs. As with the Matlab Object, the Scripting object accepts arbitrary inputs and outputs and provides an editor for scripts. Unlike the Matlab object this can accept scripts in any supported language, as selected by a leading shebang.



- Why is this desired?

There are various ways in which this may be accomplished

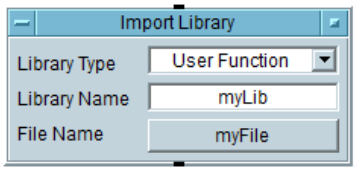
- Toolkit *<< maybe low effort but clumsy and perhaps fragile*
- VEE Handler *<< moderate effort but poor performance*
- External Handler *<< moderate effort but maybe most flexible*
- Matlab spoofing *<< high effort but most elegant*

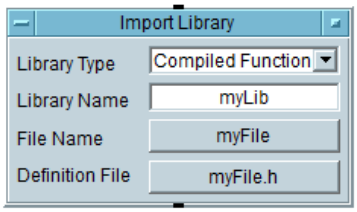
Why do we need a scripting interface?

It is perfectly fair and expected to ask "WHY?" do we need such a thing in the first place. The short answer is to extend the capabilities of VEE by providing a link to the modern world.

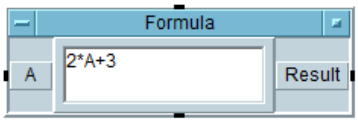
The math libraries in VEE date back to the late 1980's. While they were leading-edge at that time they have not been updated since are are showing their age. The existing libraries are 32-bit. Only a few accept complex arguments. Today there are a vast array of pure math, scientific, and engineering libraries available to be used- but no suitable link exists in VEE.

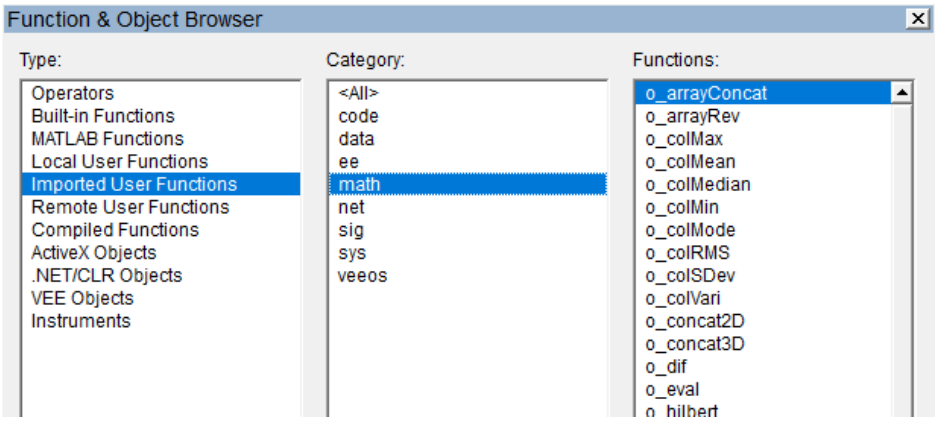
As it stands, VEE allows the import of libraries written in VEE or that have straight C interfaces via the Import Library object.





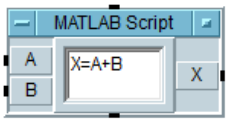
This capability allows the end user to transparently use the imported functions along side all the built-in functions. They show up in the Function Browser and can be used as desired in any expression, such as in the formula object





However this capability is limited to function written in VEE or whose import is available through a straight C interface. While straight C was common in the past, most modern libraries and tools do not provide such an interface. In some cases dotNet is available and can be used, but it tends to be clumsy and poorly documented.

To address this basic limitation and provide access to a broad range of functionality the MATLAB Object was introduced.

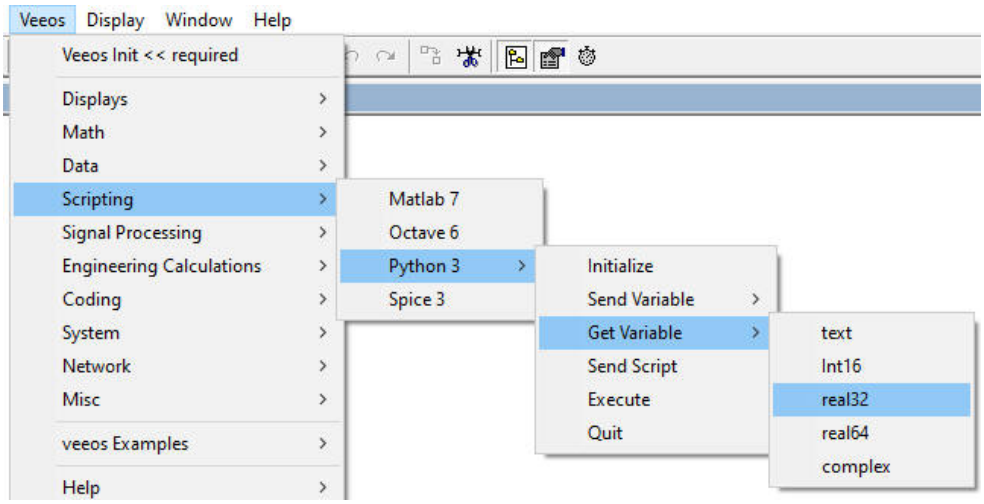


The Matlab Object provides a clean link to Matlab and hence to the broad array of Matlab libraries. In function it behaves just like a Formula object except that the expression (script) is interpreted by Matlab rather than the internal VEE engine.

Since the modern world has largely moved from compiled tools (C#, etc) to interpreted tools (Python, etc) it becomes highly desirable to develop a more general purpose link to scripting, but while maintaining the ease of use of the Matlab Script object.

Toolkit (All VEE)

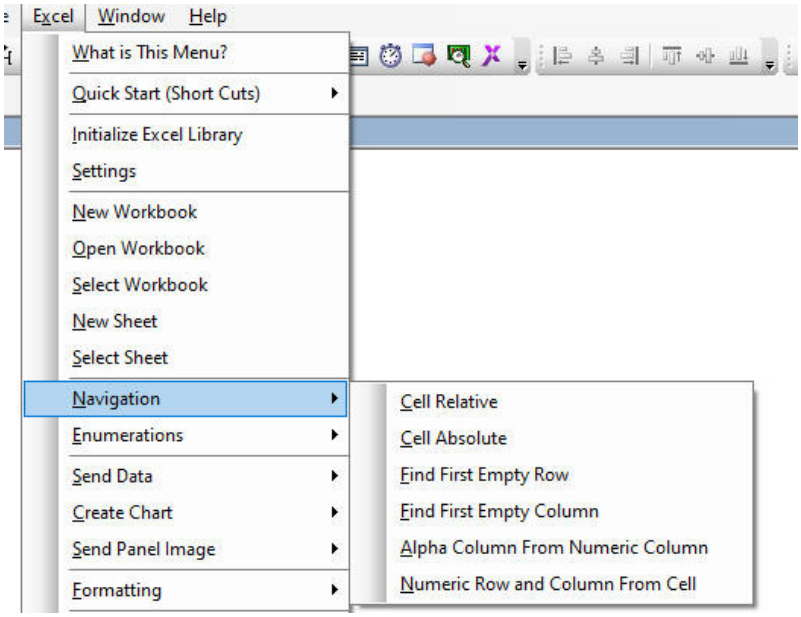
Implement any needed interface directly in VEE. For each scripting environment create a Toolkit, perhaps similar to the Excel or Database Toolkits.



The end user would string together the needed tools for each supported scripting language. It seems likely that in each case there would be similar needed functions such as

- Initialization - load needed libraries, set variables, etc.
- send variable
- send script
- execute
- get data
- quit

This would almost certainly succeed since all that is needed is to implement whatever API each scripting language has available. Like addons such as VEEOS, this Toolkit could be added to the menus. Since this method requires no changes to VEE itself it can be implemented by a third-party. An example is the Excel tookit.



However, like these addons are a programming aid and the end user faces the significant task of understanding the needed details which are likely different for each supported language. The usage model would be very different from Matlab Object wherein no VEE programming is needed- just specify the script and variables.

This is likely the lowest risk and easiest to implement from a VEE perspective, but also the clumsiest in that it leaves many programming details to the end user.

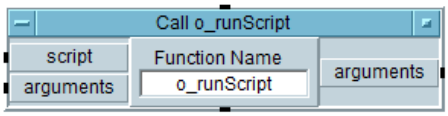
Disadvantages

- VEE is quite general purpose but does not support modern API's so it may be significant work interfacing to some engines
- End user has significant programming task

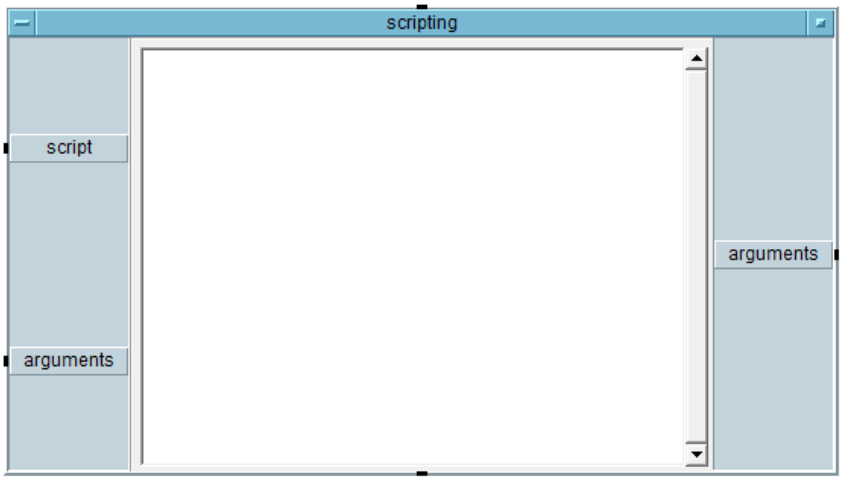
VEE Handler

Create an intermediary function or set of functions within VEE that communicates with scripting languages.

The underlying functionality is provided by a function that hides the scripting Toolkit created for the Toolkit method. The end user sees this function instead of having to deal with the various pieces of the Toolkit.



The end user could call this function directly or it could placed behind a user object that includes a simple script editor. The script editor could be as simple as a text constant or as useful as a "real" editor included as a dotnet object.



NOTE that since the function has exactly one input and one output for variables, if multiple variables are needed (probably most cases) they would need to be a single data structure. To do this it would make sense to use a record and have the variables be identified as field names with the accompanying values be of course the field values.

```
Record example- define two variables "A" and "B"
(Record
 (schema
  (numFields 2)
  (fieldName "A"
   (type Text)
  )
  (fieldName "B"
   (type Real64)
  )
 )
 (data
  (record
   ( "A" "Text field")
   ( "B" 1.25)
  )
)
)
```

Similarly, return variables would be part of the same input data structure, with dummy values as appropriate.

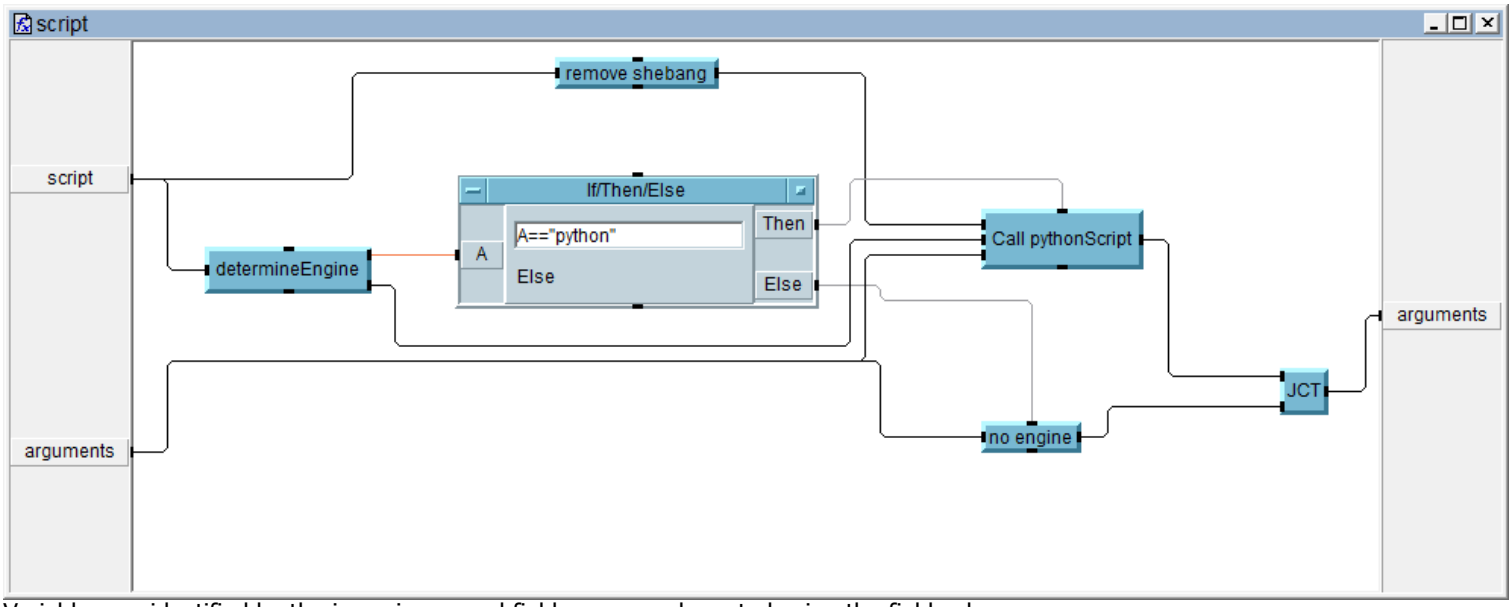
```
INPUT #################
(Record
 (schema
  (numFields 3)
  (fieldName "A"
   (type Text)
  )
  (fieldName "B"
   (type Real64)
  )
  (fieldName "X"
   (type Text)
  )
 )
 (data
  (record
   ( "A" "Text field")
   ( "B" 1.25)
   ( "X" "dummy value")
  )
)
)

OUTPUT ##################
(Record
 ...
   ( "X" "calculated value")
  )
)
)
```

This could be implemented with any version of VEE and with no internal VEE knowledge. Hence it is very low risk and mostly requires only knowledge of whatever language in which the handler is written. The handler function would examine the input record for variable names and values and then call the required Toolkit functions for the appropriate scripting language.

UPDATE: 20210220 -- initial version built using File IO and portable Python is successful. This is the least sophisticated handler and is slow but it works and should be fairly robust.

- Only Python is implemented but the structure is such that any language could be added
- The initial (shebang) line is read as intended and used to determine which language and executable to utilize. This way if python is installed then the end user can simply use *#!python* since the python executable will already be in the default path. If multiple pythons are available such as portable versions then the full path can be used, as in *#!//server/share/apps/python3.8/python* to select python independently for each script.



- Variables are identified by the incoming record field names and created using the field values.
- Variables are passed to Python by prepending lines to the given script.
- Variables are retrieved by appending *file.write(str(variable))* lines to the script, where each variable is written in ASCII format to a file whose name is that of the variable. Then in VEE these files are read in the same type as the original values. these value are used to update the arguments record and that record is outputted.

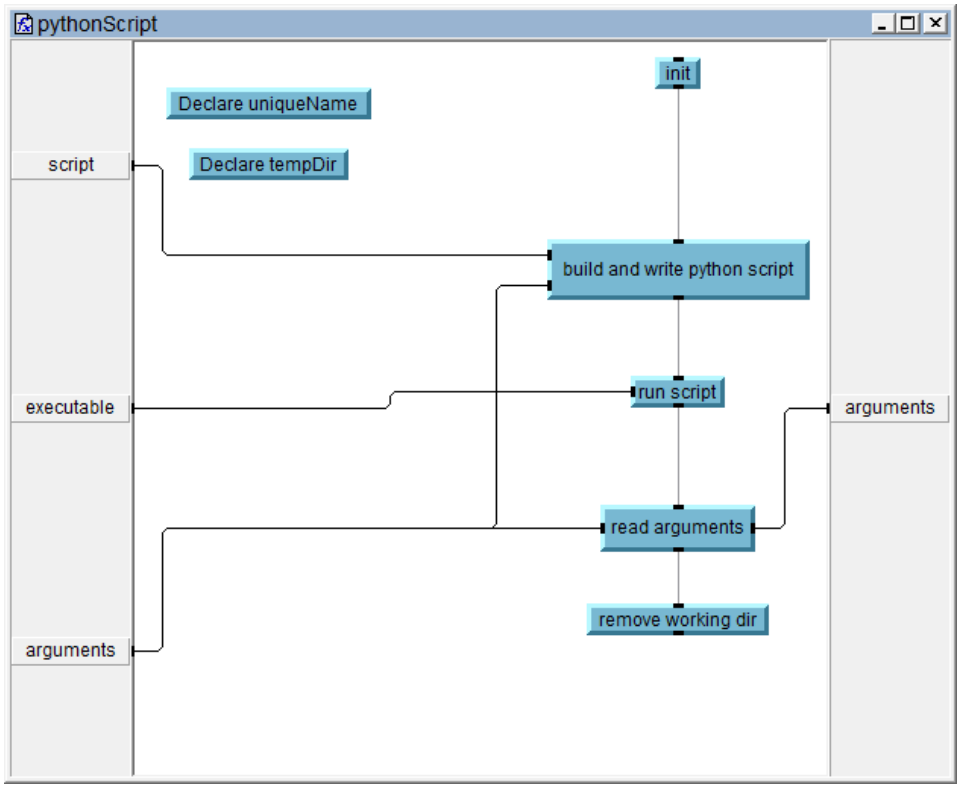
```
define variables
a = 45
B = 16

actual script
B= a + 2 * B

write variables to files
f = open("a", "w")
f.write(str(a))
f.close()
f = open("B", "w")
f.write(str(B))
f.close()

quit python
exit
```

- After the augmented script is written to a file, python is called in order to run that script.
- After retrieving all values python is closed and all the temporary files are deleted



Profiling shows that, not surprisingly, the time sinks are writing files and especially instantiating and running Python. BUT at the same time this method works and shows feasibility.

The next step in complexity is to maintain the same structure but instead of using file IO, look at embedded Python. The advantage here will be performance since one should be able to instantiate Python and leave it running, calling functions as needed. Odds are a C wrapper would be involved to create a DLL that can be referenced as a Compiled Library.

Advantages

- No need to read/write container format since using VEE directly
- Relatively quick and easy

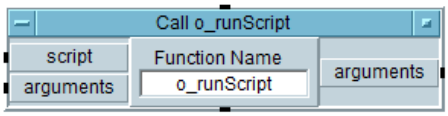
Disadvantages

- clumsy since user has access to exactly one input so must create a single piece of data. This since the needed function is fixed and therefore the programmer cannot change it. There would be no problem if only a predefined schema was needed, but that is not the case.
- Each new engine will have a unique interface that needs to be developed within VEE, mostly from scratch.

Create External Handler

Write an external handler that communicates with scripting languages and with VEE. VEE data and scripts are sent as VEE containers. The handler parses the containers, formats the script and data as needed, a executes the appropriate scripting language, then passes data back to VEE. By choosing an external language that is well supported and widely used, the task of incorporating new scripting engines could be made fairly simple.

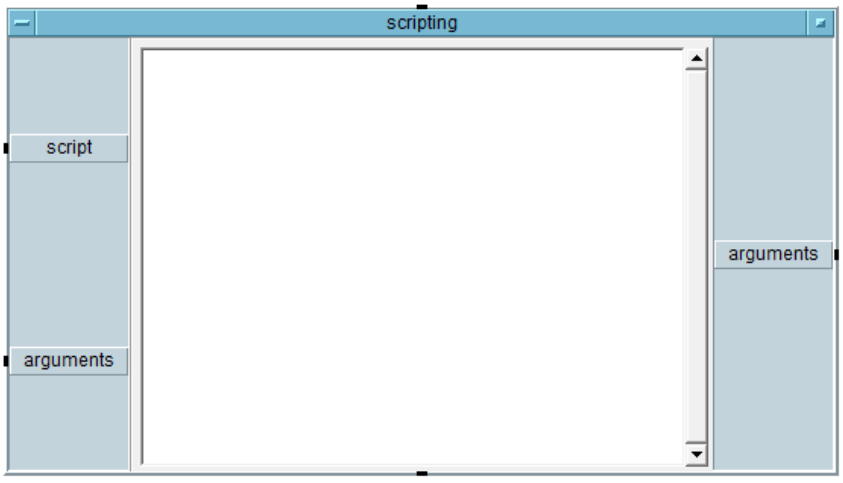
The underlying functionality is provided by a function.



This could be implemented in multiple ways, including

- A VEE function that communicates with the handler via sockets. the handler would be started and would remain running as needed as an external process.
- A compiled function that is part of the handler and is imported. The handler would be launched and run as needed and as an imported function would be part of the running VEE program.
- A compiled function that acts as intermediary to the handler. This could be a small C wrapper to a handler written in Python

The end user could call this function directly or it could placed behind a user object that includes a simple script editor. The script editor could be as simple as a text constant or as useful as a "real" editor included as a dotnet object.



The handler is the key here. It would communicate with both VEE and whatever scripting language is chosen. In the general case a standard shebang would be added as the first line of any script. This would identify which scripting language is used by the script. The handler would use this line to determine which of the supported scripting languages would be called. It would seem logical to create the handler in Python, since Python is very well supported and can interface to most anything.

The handler itself would need to read and write VEE container format, but that is stable and well defined structured ASCII so should be reasonably simple to implement. Standard parsing tools would likely be very close to working with minor modifications. Since any VEE data can

be placed in container format this would be very general purpose. Examples of VEE containers include:

```
Scalar complex ##########################
(Complex
 (data  (34, -56))
)

Real64 array ############################
(Real64
 (numDims 1)
 (size 3)
 (data
   [ 0.001251220703125 0.563568115234375 0.19329833984375 ]
 )
)
```

NOTE that since the function has exactly one input and one output, if multiple variables are needed (probably most cases) they would need to be a single data structure. To do this it would make sense to have the variables be identified as field names with the accompanying values be of course the field values.

```
Record example- define two variables "A" and "B"
(Record
 (schema
  (numFields 2)
  (fieldName "A"
   (type Text)
  )
  (fieldName "B"
   (type Real64)
  )
 )
 (data
  (record
   ( "A" "Text field")
   ( "B" 1.25)
  )
)
)
```

Similarly, return variables would be part of the same input data structure, with dummy values as appropriate.

```
INPUT #################
(Record
 (schema
  (numFields 3)
  (fieldName "A"
   (type Text)
  )
  (fieldName "B"
   (type Real64)
  )
  (fieldName "X"
   (type Text)
  )
 )
 (data
  (record
   ( "A" "Text field")
   ( "B" 1.25)
   ( "X" "dummy value")
  )
)
)

OUTPUT ##################
(Record
 ...
   ( "X" "calculated value")
  )
)
)
```

This could be implemented with any version of VEE and with no internal VEE knowledge. Hence it is very low risk and mostly requires only knowledge of whatever language in which the handler is written.

At the same time there are a few disadvantages

- clumsy since user has access to exactly one input so must create a single piece of data. This since the needed function is fixed and therefore the programmer cannot change it. There would be no problem if only a predefined schema was needed, but that is not the case.

NOTE that if one had sufficient knowledge of VEE data structures it should be possible to pass data by reference instead of container format. However the reverse-engineering needed is significant and may not be worth the trouble compared to using container format.

The primary advantage of this method would be in the great support and widespread usage of Python. Once the data is in python (variables and the script to run) there are likely tools already developed to link with Octave, Matlab, Spice and all sorts of other tools.

Conceptual Implementation:

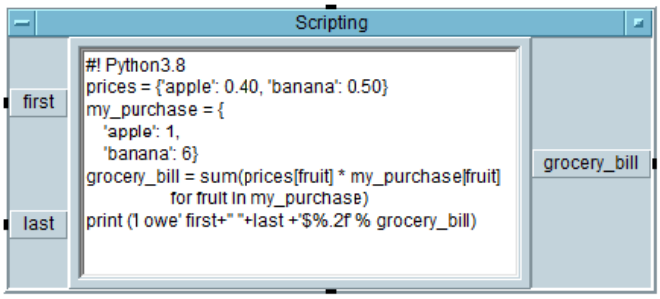
- Interface to VEE via
  - C DLL linked to VEE as a Compiled Function library
    or
  - Dotnet library written directly in Python
- In either case the interface to VEE could consist of a small number of functions that would be very similar to the existing Matlab interface

- Initialize - instantiates the handler process
- putContainer - transmits both the arguments and the script in the form of a VEE Record container. One field for each argument and one field ("script") with text array for the script itself. The handler creates variables as needed and also reads the script.
- execute - instructs the handler to proceed. This may not be necessary at all- it could proceed as soon as the above container is sent. The handler gets data back from the chosen engine and creates an output in VEE record container format
- getContainer - gets arguments back as VEE record

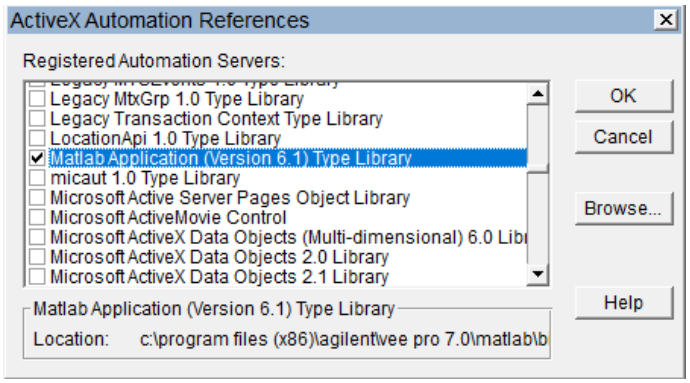
- There would be a minimal VEE code around this since all the real work would be done by the handler.

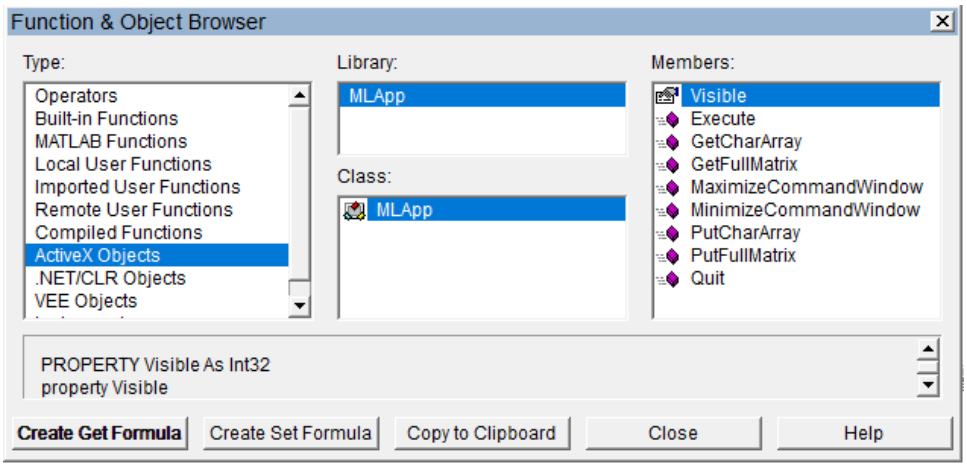
Matlab Spoofing and the Matlab Object

A handler provides a spoofed version of the Matlab interface and VEE preferences are set to use this interface. In VEE the Matlab Object becomes a general Script object but is otherwise exactly the same. For the end user this becomes extremely clean and simple since it works exactly like the native Matlab object. The user becomes free to add any inputs and outputs desired.



With VEE 7 the interface is a simple set of COM functions that are imported when the Matlab object is run. Note that when VEE is first installed this library is not registered, but it is registered when the Matlab object is first run.





There would still need to be a handler that determines which script engine to run and handles communications.

What is needed along with a handler is the spoofed interface

1. Determine the exact functionality of the Matlab interface functions (looks very simple and is probably documented by Mathworks)
2. Create library that spoofs Matlab
3. Determine appropriate install. Could replace the built-in Matlab, or maybe better be placed in the registry so that VEE can choose to use it in default preferences.

With VEE 9 this becomes much more involved.

It is quickly seen that there is NOT an obvious load of external libraries as with VEE 7. Digging further shows that even a full install of Matlab does not register any useful-looking libraries. However the link is established somehow and is almost certainly a function library of some sort.

Firing up the internal Matlab that comes with VEE and looking at the attached DLL's (procexp) shows quite a few

```
awt.dll Java(TM) Platform SE binary     Sun Microsystems, Inc.  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\sys\java\jre\win32\jre\bin\awt.dll
boost_date_time-vc90-mt-1_40.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\boost_date_time-vc90-mt-1_40.dll
boost_filesystem-vc90-mt-1_40.dll                       C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\boost_filesystem-vc90-mt-1_40.dll
boost_iostreams-vc90-mt-1_40.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\boost_iostreams-vc90-mt-1_40.dll
boost_regex-vc90-mt-1_40.dll                    C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\boost_regex-vc90-mt-1_40.dll
boost_signals-vc90-mt-1_40.dll                  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\boost_signals-vc90-mt-1_40.dll
boost_system-vc90-mt-1_40.dll                   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\boost_system-vc90-mt-1_40.dll
boost_thread-vc90-mt-1_40.dll                   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\boost_thread-vc90-mt-1_40.dll
comcli.dll      comcli  The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\comcli.dll
ctfarchiver.dll                 C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\ctfarchiver.dll
ctfrt.dll                       C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\ctfrt.dll
ctfrtcrypto.dll ctfrtcrypto     The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\ctfrtcrypto.dll
dservices.dll                   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\dservices.dll
fontmanager.dll Java(TM) Platform SE binary     Sun Microsystems, Inc.  C:\Program Files (x86)\Keysight\VEE Pro
9.3\matlab\sys\java\jre\win32\jre\bin\fontmanager.dll
graphics_util.dll                       C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\graphics_util.dll
hg.dll  hg      The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\hg.dll
hgbuiltins.dll                  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\hgbuiltins.dll
hgdatatypes.dll                 C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\hgdatatypes.dll
hgutils.dll                     C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\hgutils.dll
hpi.dll Java(TM) Platform SE binary     Sun Microsystems, Inc.  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\sys\java\jre\win32\jre\bin\hpi.dll
icudt42.dll     ICU Data DLL    IBM Corporation and others      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\icudt42.dll
icuin42.dll     IBM ICU I18N DLL        IBM Corporation and others      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\icuin42.dll
icuio42.dll     IBM ICU I/O DLL IBM Corporation and others       C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\icuio42.dll
icuuc42.dll     IBM ICU Common DLL      IBM Corporation and others      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\icuuc42.dll
iqm.dll                 C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\iqm.dll
ir_xfmr.dll                     C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\ir_xfmr.dll
java.dll        Java(TM) Platform SE binary     Sun Microsystems, Inc.  C:\Program Files (x86)\Keysight\VEE Pro
9.3\matlab\sys\java\jre\win32\jre\bin\java.dll
jmi.dll jmi     The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\jmi.dll
jproxy.dll                      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\jproxy.dll
jvm.dll Java HotSpot(TM) Client VM      Sun Microsystems, Inc.  C:\Program Files (x86)\Keysight\VEE Pro
9.3\matlab\sys\java\jre\win32\jre\bin\client\jvm.dll
libexpat.dll                    C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libexpat.dll
libhdf5.dll                     C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libhdf5.dll
libifcoremd.dll Intel(r) Fortran Compiler RTL (thread-safe)     Intel Corporation       C:\Program Files (x86)\Keysight\VEE Pro
9.3\matlab\bin\win32\libifcoremd.dll
libmat.dll      libmat  The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmat.dll
libmex.dll      libmex  The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmex.dll
libmmd.dll      Math Library for Intel(r) Compilers (thread-safe)       Intel Corporation       C:\Program Files (x86)\Keysight\VEE Pro
9.3\matlab\bin\win32\libmmd.dll
libmwamd.dll                    C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwamd.dll
libmwbinder.dll                 C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwbinder.dll
libmwblas.dll                   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwblas.dll
libmwbridge.dll                 C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwbridge.dll
libmwbuiltins.dll       libmwbuiltins   The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwbuiltins.dll
libmwcholmod.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwcholmod.dll
libmwcolamd.dll                 C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwcolamd.dll
libmwcsparse.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwcsparse.dll
libmwfl.dll                     C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwfl.dll
libmwgui.dll    gui     The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwgui.dll
libmwhardcopy.dll       hardcopy        The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwhardcopy.dll
libmwi18n.dll                   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwi18n.dll
libmwlapack.dll                 C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwlapack.dll
libmwma57.dll                   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwma57.dll
libmwmathcore.dll       mathcore        The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwmathcore.dll
libmwmathelem.dll       mathelem        The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwmathelem.dll
libmwmathlinalg.dll     mathlinalg      The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwmathlinalg.dll
libmwmathrng.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwmathrng.dll
libmwmathutil.dll                       C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwmathutil.dll
libmwmathxps.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwmathxps.dll
libmwMATLAB_res.dll                     C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwMATLAB_res.dll
libmwompwrapper.dll                     C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwompwrapper.dll
libmwresource_core.dll                  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwresource_core.dll
libmwrookfastbp.dll                     C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwrookfastbp.dll
libmwservices.dll       libmwservices   The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwservices.dll
libmwspmatrix.dll                       C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwspmatrix.dll
libmwspqr.dll                   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwspqr.dll
libmwumfpack.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmwumfpack.dll
libmx.dll       libmx   The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libmx.dll
libuij.dll      libuij  The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libuij.dll
libut.dll       libut   The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\libut.dll
m_dispatcher.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\m_dispatcher.dll
m_interpreter.dll       m_interpreter   The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\m_interpreter.dll
m_ir.dll        m_ir    The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\m_ir.dll
m_parser.dll    m_parser        The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\m_parser.dll
m_pcodegen.dll  m_pcodegen      The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\m_pcodegen.dll
m_pcodeio.dll   m_pcodeio       The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\m_pcodeio.dll
mclbase.dll                     C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mclbase.dll
mclmcr.dll                      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mclmcr.dll
mclmcrrt7_15.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\runtime\win32\mclmcrrt7_15.dll
mcos.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mcos.dll
mcr.dll mcr     The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mcr.dll
mlautoregister.dll      mlautoregister  The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mlautoregister.dll
mlint.dll                       C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mlint.dll
mlutil.dll                      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mlutil.dll
mpath.dll       mpath   The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mpath.dll
msvcr71.dll     Microsoft® C Runtime Library    Microsoft Corporation   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\msvcr71.dll
mtok.dll                        C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mtok.dll
mvm.dll                 C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mvm.dll
mwoles05.dll    MATLAB Automation Server        The MathWorks   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\mwoles05.dll
nativehg.dll                    C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\nativehg.dll
nativejava.dll                  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\nativejava.dll
nativejava_services.dll                 C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\nativejava_services.dll
nativejmi.dll                   C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\nativejmi.dll
nativeservices.dll                      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\nativeservices.dll
net.dll Java(TM) Platform SE binary     Sun Microsystems, Inc.  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\sys\java\jre\win32\jre\bin\net.dll
nio.dll Java(TM) Platform SE binary     Sun Microsystems, Inc.  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\sys\java\jre\win32\jre\bin\nio.dll
profiler.dll                    C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\profiler.dll
tbb.dll Threading Building Blocks library       Intel Corporation       C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\tbb.dll
tbbmalloc.dll   Scalable Allocator library      Intel Corporation       C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\tbbmalloc.dll
udd.dll udd     The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\udd.dll
udd_mi.dll      udd_mi  The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\udd_mi.dll
uinone.dll                      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\uinone.dll
uiw.dll uiw     The MathWorks Inc.      C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\uiw.dll
verify.dll      Java(TM) Platform SE binary     Sun Microsystems, Inc.  C:\Program Files (x86)\Keysight\VEE Pro
9.3\matlab\sys\java\jre\win32\jre\bin\verify.dll
xerces-c_2_7.dll        Shared Library for Xerces-C Version 2.7.0       Apache Software Foundation      C:\Program Files (x86)\Keysight\VEE Pro
9.3\matlab\bin\win32\xerces-c_2_7.dll
xmlcore.dll                     C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\xmlcore.dll
xsd_binder.dll                  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\xsd_binder.dll
zip.dll Java(TM) Platform SE binary     Sun Microsystems, Inc.  C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\sys\java\jre\win32\jre\bin\zip.dll
```

```
zlib1.dll       zlib data compression library           C:\Program Files (x86)\Keysight\VEE Pro 9.3\matlab\bin\win32\zlib1.dll
```

A full Matlab install (NOTE that one needs 32-bit version) includes a large number of DLL's

```
acml.dll
activation.dll
asynciocore.dll
asyncioimpl.dll
boost_date_time-vc90-mt-1_40.dll
boost_filesystem-vc90-mt-1_40.dll
boost_graph-vc90-mt-1_40.dll
boost_iostreams-vc90-mt-1_40.dll
boost_math_c99-vc90-mt-1_40.dll
boost_math_c99f-vc90-mt-1_40.dll
boost_math_c99l-vc90-mt-1_40.dll
boost_math_tr1-vc90-mt-1_40.dll
boost_math_tr1f-vc90-mt-1_40.dll
boost_math_tr1l-vc90-mt-1_40.dll
boost_prg_exec_monitor-vc90-mt-1_40.dll
boost_program_options-vc90-mt-1_40.dll
boost_regex-vc90-mt-1_40.dll
boost_serialization-vc90-mt-1_40.dll
boost_signals-vc90-mt-1_40.dll
boost_system-vc90-mt-1_40.dll
boost_thread-vc90-mt-1_40.dll
boost_unit_test_framework-vc90-mt-1_40.dll
boost_wave-vc90-mt-1_40.dll
boost_wserialization-vc90-mt-1_40.dll
comcli.dll
dastudio.dll
dastudio_hg.dll
dirmonitor.dll
dotnetcli.dll
dotnetcli_loader.dll
DXEnumerator.dll
EasyHook32.dll
emf.dll
glee.dll
glee2.dll
glee2_mi.dll
glee2_qt.dll
glee_util.dll
glren.dll
gluegen-rt.dll
gmp.dll
graphics_util.dll
gsdll32.dll
handle_graphics.dll
hd424m.dll
hg.dll
hgbuiltins.dll
hgdatatypes.dll
hgutils.dll
hm424m.dll
icudt42.dll
icuin42.dll
icuio42.dll
icule42.dll
icuuc42.dll
instutil.dll
iqm.dll
ir_interp.dll
ir_xfmr.dll
JavaAccessBridge.dll
JAWTAccessBridge.dll
jmi.dll
jogl.dll
jogl_awt.dll
Jp2Adapter.dll
kdu_a63R.dll
kdu_v63R.dll
libaudiodevice.dll
libcdf.dll
libeng.dll
libexpat.dll
libfftw3.dll
libfftw3f.dll
libfftw3i.dll
libGctp.dll
libhdf5.dll
libhdf5_hl.dll
libhdfeos.dll
libifcoremd.dll
libifportmd.dll
libiomp5md.dll
libmat.dll
libmex.dll
libmmd.dll
libmwamd.dll
libmwarpack.dll
libmwbinder.dll
libmwblas.dll
libmwblascompat32.dll
libmwbridge.dll
libmwbuiltins.dll
libmwchart.dll
libmwcholmod.dll
libmwcli.dll
libmwcolamd.dll
libmwcsparse.dll
libmwdastudio_res.dll
libmwfftw.dll
libmwfl.dll
libmwglee_res.dll
libmwglue2_res.dll
libmwgui.dll
libmwhardcopy.dll
libmwhghardcopy.dll
libmwi18n.dll
libmwlapack.dll
libmwma57.dll
libmwmathcgeo.dll
libmwmathcore.dll
libmwmathdsp.dll
libmwmathelem.dll
libmwmathlinalg.dll
libmwmathrng.dll
libmwmathspec.dll
libmwmathutil.dll
libmwmathxps.dll
libmwMATLAB_res.dll
libmwompwrapper.dll
libmwqhull.dll
libmwresource_core.dll
libmwrookfastbp.dll
libmwsavevars.dll
```

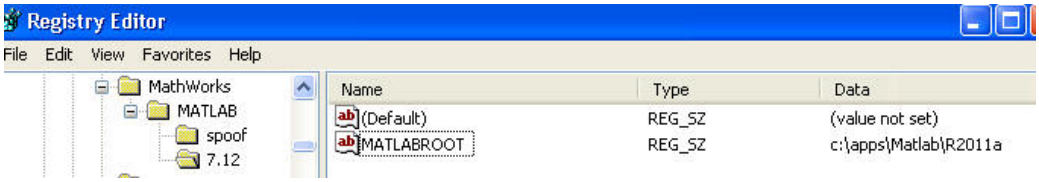
```
libmwservices.dll
libmwSimulinkTypes_res.dll
libmwSL_SERVICES_res.dll
libmwspmatrix.dll
libmwspqr.dll
libmwumfpack.dll
libmx.dll
libnetcdf.dll
libpng12.dll
libportaudio.dll
libtiff.dll
libuij.dll
libut.dll
m3i.dll
m3i_mi.dll
mcos.dll
mcos_factory.dll
mcr.dll
mkl.dll
mlautoregister.dll
mlib.dll
mlint.dll
mllapack.dll
mlutil.dll
MMCodecChooser.dll
mmreadercore.dll
mmreaderudd.dll
mpath.dll
mpfr.dll
msvcp71.dll
msvcr71.dll
mtok.dll
multimedia.dll
mvm.dll
mwdx.dll
mwoles05.dll
m_dispatcher.dll
m_interpreter.dll
m_ir.dll
m_parser.dll
m_pcodegen.dll
m_pcodeio.dll
nativecmdwin.dll
nativedirmonitor.dll
nativehg.dll
nativei18n.dll
nativejava.dll
nativejava_services.dll
nativejmi.dll
nativelex.dll
nativelmgr.dll
nativemat.dll
nativemlint.dll
nativeservices.dll
nativewindows.dll
osg65-osg.dll
osg65-osgDB.dll
osg65-osgFX.dll
osg65-osgGA.dll
osg65-osgManipulator.dll
osg65-osgParticle.dll
osg65-osgShadow.dll
osg65-osgSim.dll
osg65-osgTerrain.dll
osg65-osgText.dll
osg65-osgUtil.dll
osg65-osgViewer.dll
osgserver.dll
ot11-OpenThreads.dll
PreviewWindow.dll
profiler.dll
Qt3Support4.dll
QtCore4.dll
QtGui4.dll
QtNetwork4.dll
QtOpenGL4.dll
QtSql4.dll
QtSvg4.dll
QtWebKit4.dll
QtXml4.dll
QtXmlPatterns4.dll
refblas.dll
reflapackpt.dll
rxtxSerial.dll
servicesproxy.dll
sl_services.dll
sl_types.dll
sl_utility.dll
tamimframe.dll
tamimframemex.dll
tammex.dll
tamobjsys.dll
tamutil.dll
tbb.dll
tbbmalloc.dll
udd.dll
udd_mi.dll
uinone.dll
uiw.dll
VideoDeviceChooser.dll
VideoFormatInfo.dll
WindowsAccessBridge.dll
xerces-c_2_7.dll
xmlcore.dll
xmlxrcsc.dll
ziparchiver.dll
zlib1.dll
```

but looking at what VEE attaches when it runs a script shows a relatively small number of DLL's. these would seem a good starting point seeing what is necessary for the spoof.

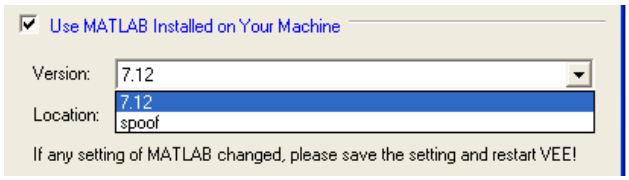
```
boost_date_time-vc90-mt-1_40.dll                        C:\apps\MATLAB\R2011a\bin\win32\boost_date_time-vc90-mt-1_40.dll
boost_filesystem-vc90-mt-1_40.dll                       C:\apps\MATLAB\R2011a\bin\win32\boost_filesystem-vc90-mt-1_40.dll
boost_signals-vc90-mt-1_40.dll                  C:\apps\MATLAB\R2011a\bin\win32\boost_signals-vc90-mt-1_40.dll
boost_system-vc90-mt-1_40.dll                   C:\apps\MATLAB\R2011a\bin\win32\boost_system-vc90-mt-1_40.dll
boost_thread-vc90-mt-1_40.dll                   C:\apps\MATLAB\R2011a\bin\win32\boost_thread-vc90-mt-1_40.dll
icudt42.dll     ICU Data DLL    IBM Corporation and others      C:\apps\MATLAB\R2011a\bin\win32\icudt42.dll
icuin42.dll     IBM ICU I18N DLL        IBM Corporation and others      C:\apps\MATLAB\R2011a\bin\win32\icuin42.dll
icuio42.dll     IBM ICU I/O DLL IBM Corporation and others      C:\apps\MATLAB\R2011a\bin\win32\icuio42.dll
icuuc42.dll     IBM ICU Common DLL      IBM Corporation and others      C:\apps\MATLAB\R2011a\bin\win32\icuuc42.dll
libeng.dll      libeng  The MathWorks Inc.      C:\apps\MATLAB\R2011a\bin\win32\libeng.dll
libexpat.dll                    C:\apps\MATLAB\R2011a\bin\win32\libexpat.dll
libmwfl.dll                     C:\apps\MATLAB\R2011a\bin\win32\libmwfl.dll
libmwi18n.dll                   C:\apps\MATLAB\R2011a\bin\win32\libmwi18n.dll
libmwMATLAB_res.dll                     C:\apps\MATLAB\R2011a\bin\win32\libmwMATLAB_res.dll
```

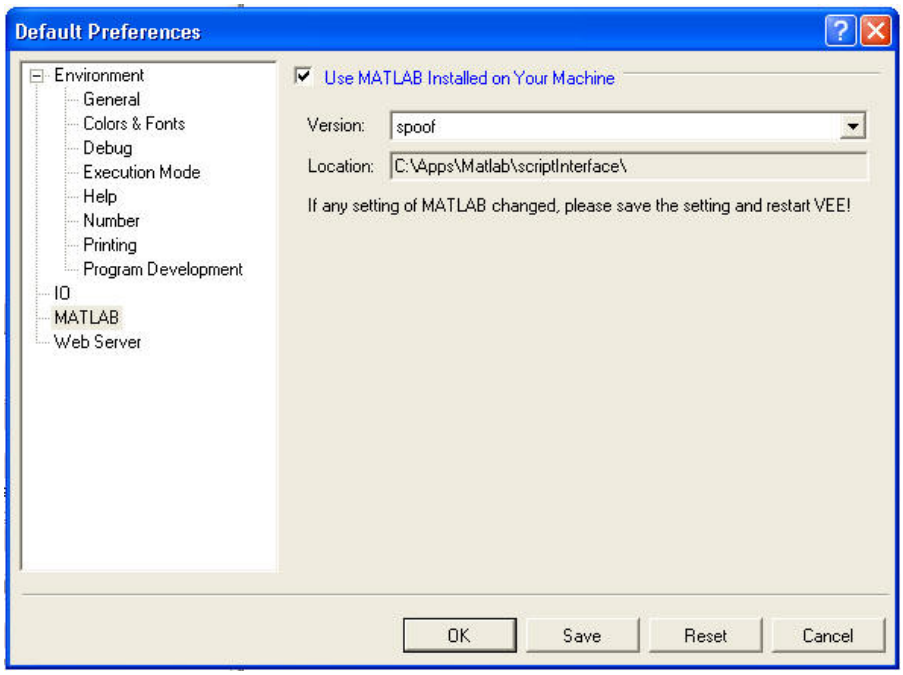
```
libmwresource_core.dll                  C:\apps\MATLAB\R2011a\bin\win32\libmwresource_core.dll
libmx.dll       libmx    The MathWorks Inc.       C:\apps\MATLAB\R2011a\bin\win32\libmx.dll
libut.dll       libut    The MathWorks Inc.       C:\apps\MATLAB\R2011a\bin\win32\libut.dll
zlib1.dll       zlib data compression library            C:\apps\MATLAB\R2011a\bin\win32\zlib1.dll234
```

A quick experiment shows that a real Matlab install is registered in
HKLM\SOFTWARE\WOW6432Node\MathWorks\MATLAB\7.12
and has a single key key
MATLABROOT pointing to install directory C:\apps\MATLAB\R2011a



VEE's Default Preferences looks at these registry entries to determine choices.





Hence the "spoof" install could be as simple as placing the bits and creating a single registry entry.

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Mathworks]

[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Mathworks\Matlab]

[HKEY_LOCAL_MACHINE\SOFTWARE\WOW6432Node\Mathworks\Matlab\spoof]
"MATLABROOT"="c:\\apps\\scriptInterface"
```

VEE would then recognize and link to it. Note that an empty directory will not work, but all that is needed for the Default Preferences choice is a file <location>\bin\win32\MATLAB.exe. Note that this can be a dummy file. What's left is to look at what resources "MATLAB.exe" needs to have in order for VEE to link to it. Note that the real "MATLAB.exe" is a small file so almost certainly essentially a header. The real work is probably a script engine incorporating files such as mcr.dll. This will need to be determined for a successful spoof.

What would be best would be to examine VEE source for the above link and for the actual interfacing to Matlab, though there may also be good information in Mathworks' External API documentation for R2011A (aka 7.12) as is used inside VEE.

With this approach the handler does not need to look at internal VEE data structures since the Matlab Object will format them to be Matlab compatible. The handler needs to understand Matlab data structures but fortunately those are well documented by Mathworks.

I would expect that the handler would be written in Python with perhaps a C wrapper for the interface to VEE. After dealing with Python scripting the first priority would be Matlab. But unlike the native Matlab Object, any version of Matlab could be linked as selected by the shebang. If architected well, hooks should be left so that virtually any script-driven tool could be incorporated. Python interfaces to many tools

already exist so such extensions would be relatively easy to add.

#! Matlab 7.13
#! Octave
#! Python 3.9
#! SciPy
#! SciLab
#! Spice 3F
etc.

This is the more difficult approach but by far the cleanest and most elegant since the user only sees native VEE.
