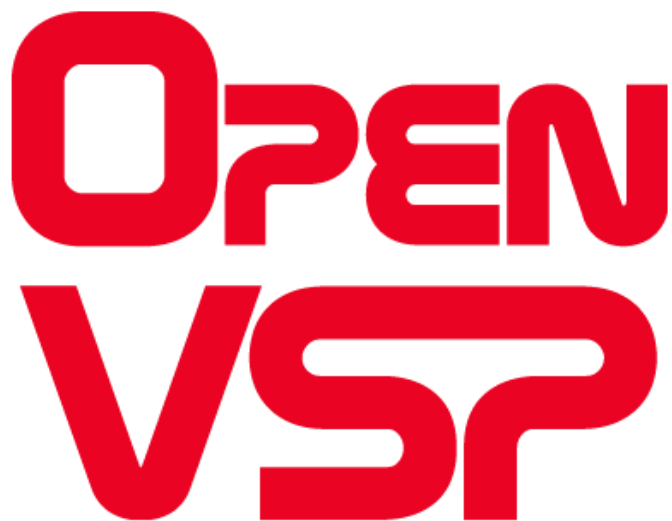


# VSP Workshop 2019

## OpenVSP – MATLAB API

The logo for OpenVSP, consisting of the words "OPEN" and "VSP" stacked vertically in a large, bold, red, sans-serif font.

**Presented by:**

Nick Brake

Empirical Systems Aerospace, Inc.

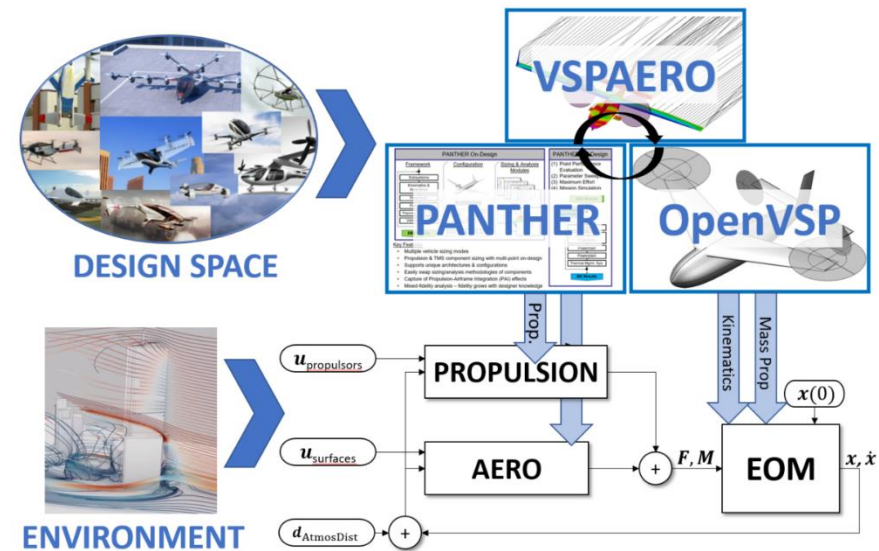
# Objectives, Approach

## Goal

- Provide API access to OpenVSP & VSPAERO through native Matlab

## Approach

- Utilize SWIG to 'wrap' C-API for Matlab (similar to Python API)
- Duplicate test suite to verify functionality
- Keep the build process simple as simple as possible
- OpenVSP 3.16.1 & MATLAB 2018



NASA SBIR PHASE I - UAM Disturbance Rejection in Conceptual Design

# SWIG & SWIG-Matlab


- SWIG – Simplified Wrapper and Interface Generator
  - Official distribution
  - Used to provide python interface to C-API
  - <http://www.swig.org/>
- SWIG-Matlab
  - Unofficial SWIG build with matlab bindings
  - <https://github.com/jaeandersson/swig.git>

# Build Process – Overview

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Create a 'matlab\_api' (use the 'python\_api' as a template)
4. Point the VSP CMAKE options to look for the swig.exe that you just created
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working

# Build Process – Overview

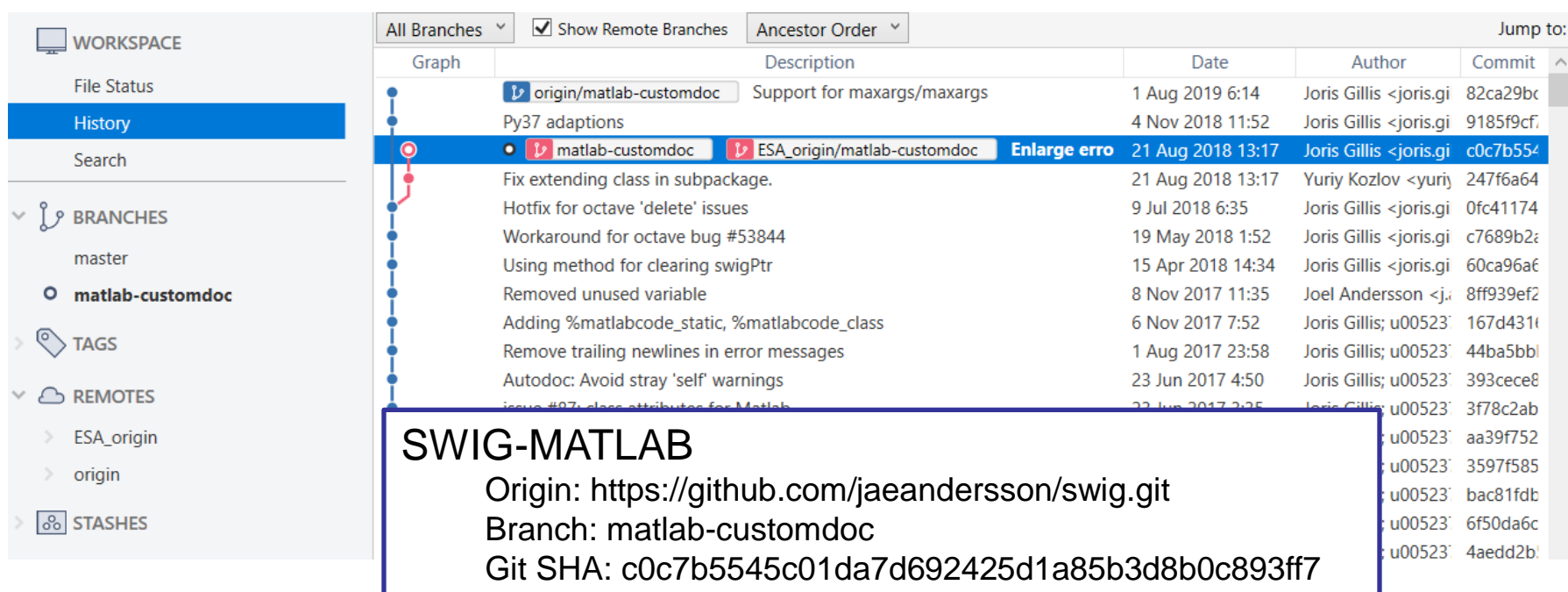
1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Create a 'matlab\_api' (use the 'python\_api' as a template)
4. Point the VSP CMAKE options to look for the swig.exe that you just created
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working



This was the hardest step. We are looking into getting these changes into the open source release

# Build Process – 1. Clone the swig-matlab

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Create a 'matlab\_api' (use the 'python\_api' as a template)
4. Point the VSP CMAKE options to look for the swig.exe that you just created
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working

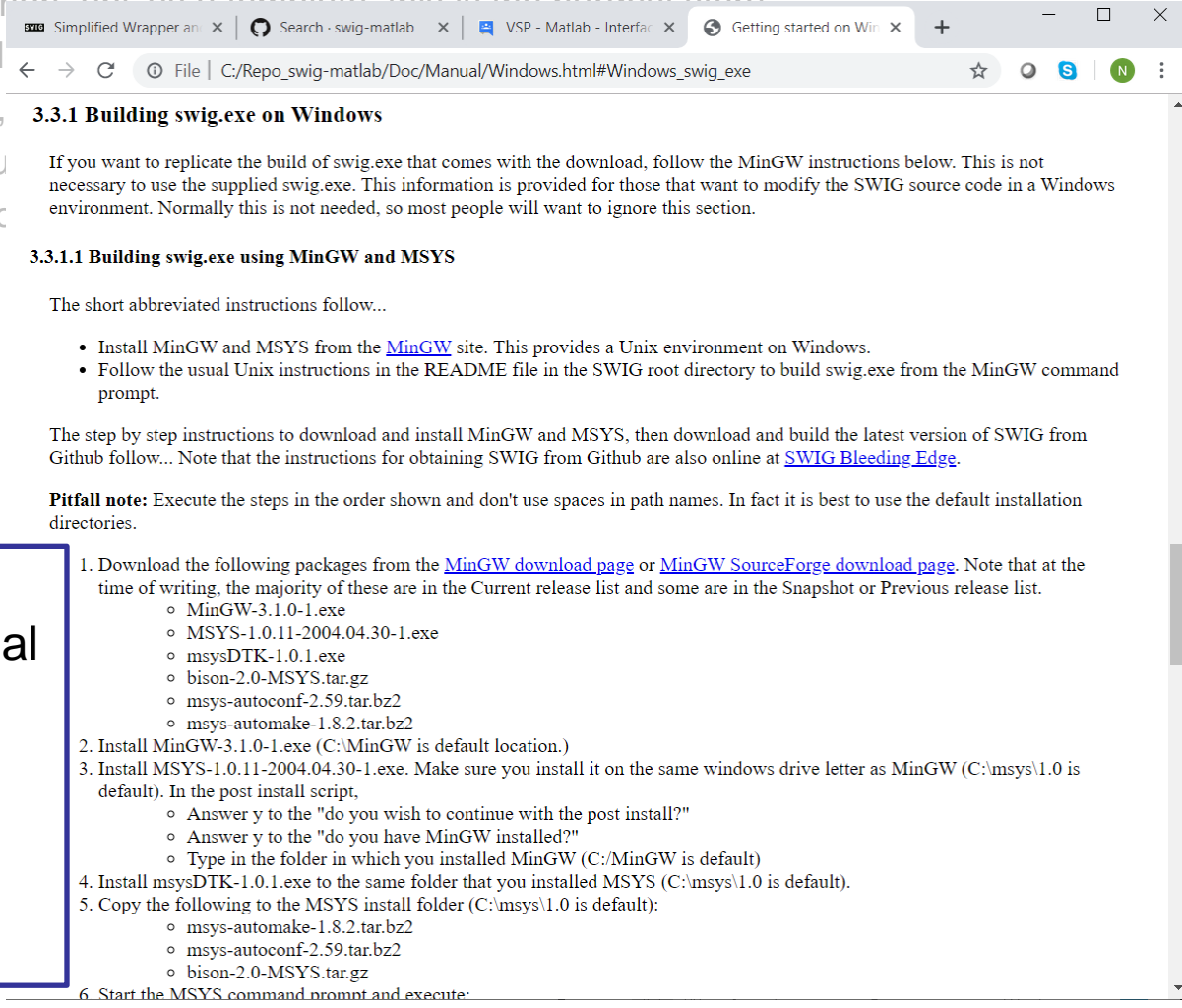


Graph	Description	Date	Author	Commit
origin/matlab-customdoc	Support for maxargs/maxargs	1 Aug 2019 6:14	Joris Gillis <joris.gi	82ca29bc
	Py37 adaption	4 Nov 2018 11:52	Joris Gillis <joris.gi	9185f9cf
matlab-customdoc	Support for maxargs/maxargs	21 Aug 2018 13:17	Joris Gillis <joris.gi	c0c7b554
	Fix extending class in subpackage.	21 Aug 2018 13:17	Yuriy Kozlov <yuriy	247f6a64
	Hotfix for octave 'delete' issues	9 Jul 2018 6:35	Joris Gillis <joris.gi	0fc41174
	Workaround for octave bug #53844	19 May 2018 1:52	Joris Gillis <joris.gi	c7689b2c
	Using method for clearing swigPtr	15 Apr 2018 14:34	Joris Gillis <joris.gi	60ca96af
	Removed unused variable	8 Nov 2017 11:35	Joel Andersson <j.	8ff939ef2
	Adding %matlabcode_static, %matlabcode_class	6 Nov 2017 7:52	Joris Gillis; u00523	167d431f
	Remove trailing newlines in error messages	1 Aug 2017 23:58	Joris Gillis; u00523	44ba5bbf
	Autodoc: Avoid stray 'self' warnings	23 Jun 2017 4:50	Joris Gillis; u00523	393cece8
	issue #97: class attributes for Matlab	22 Jun 2017 2:25	Joris Gillis; u00523	3f78c2ab
			Joris Gillis; u00523	aa39f752
			Joris Gillis; u00523	3597f585
			Joris Gillis; u00523	bac81fdb
			Joris Gillis; u00523	6f50da6c
			Joris Gillis; u00523	4aedd2b1

**SWIG-MATLAB**  
 Origin: <https://github.com/jaeandersson/swig.git>  
 Branch: matlab-customdoc  
 Git SHA: c0c7b5545c01da7d692425d1a85b3d8b0c893ff7

# Build Process – 2. Compile swig-matlab

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)
4. Point the VSP CMAKE options to l
5. Compile OpenVSP (Build solution,
6. Delete duplicate SwigClear(self) fu
7. Build some test scripts in matlab tc



**3.3.1 Building swig.exe on Windows**

If you want to replicate the build of swig.exe that comes with the download, follow the MinGW instructions below. This is not necessary to use the supplied swig.exe. This information is provided for those that want to modify the SWIG source code in a Windows environment. Normally this is not needed, so most people will want to ignore this section.

**3.3.1.1 Building swig.exe using MinGW and MSYS**

The short abbreviated instructions follow...

- Install MinGW and MSYS from the [MinGW](#) site. This provides a Unix environment on Windows.
- Follow the usual Unix instructions in the README file in the SWIG root directory to build swig.exe from the MinGW command prompt.

The step by step instructions to download and install MinGW and MSYS, then download and build the latest version of SWIG from Github follow... Note that the instructions for obtaining SWIG from Github are also online at [SWIG Bleeding Edge](#).

**Pitfall note:** Execute the steps in the order shown and don't use spaces in path names. In fact it is best to use the default installation directories.

1. Download the following packages from the [MinGW download page](#) or [MinGW SourceForge download page](#). Note that at the time of writing, the majority of these are in the Current release list and some are in the Snapshot or Previous release list.
  - MinGW-3.1.0-1.exe
  - MSYS-1.0.11-2004.04.30-1.exe
  - msysDTK-1.0.1.exe
  - bison-2.0-MSYS.tar.gz
  - msys-autoconf-2.59.tar.bz2
  - msys-automake-1.8.2.tar.bz2
2. Install MinGW-3.1.0-1.exe (C:\MinGW is default location.)
3. Install MSYS-1.0.11-2004.04.30-1.exe. Make sure you install it on the same windows drive letter as MinGW (C:\msys\1.0 is default). In the post install script,
  - Answer y to the "do you wish to continue with the post install?"
  - Answer y to the "do you have MinGW installed?"
  - Type in the folder in which you installed MinGW (C:\MinGW is default)
4. Install msysDTK-1.0.1.exe to the same folder that you installed MSYS (C:\msys\1.0 is default).
5. Copy the following to the MSYS install folder (C:\msys\1.0 is default):
  - msys-automake-1.8.2.tar.bz2
  - msys-autoconf-2.59.tar.bz2
  - bison-2.0-MSYS.tar.gz
6. Start the MSYS command prompt and execute:

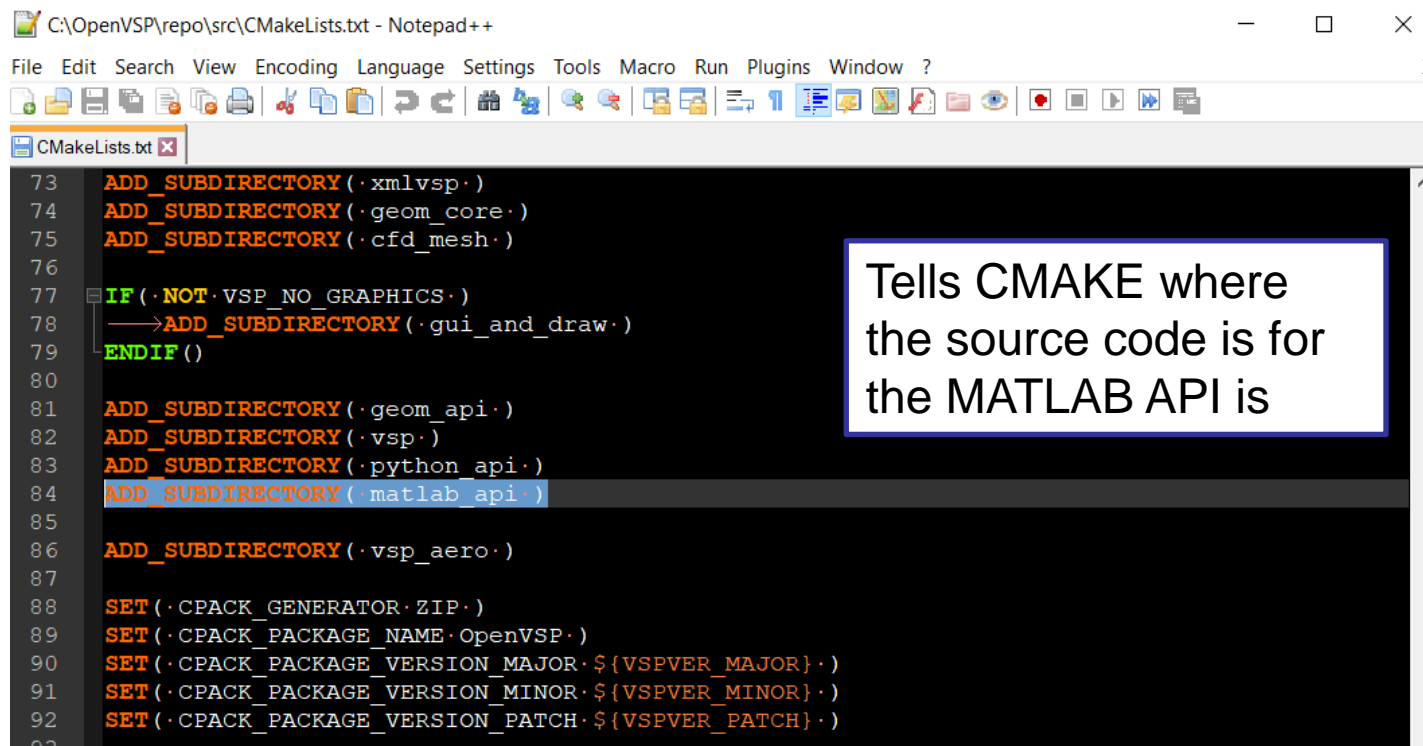
Documentation:  
.\Repo\_swig-matlab\Doc\Manual

Instructions exist to build using

- MinGW & MSYS
- Cygwin
- VisualStudio IDE (a bit manual)

# Build Process – 4. Create a 'matlab\_api'

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Point the VSP CMAKE options to look for the swig.exe that you just created
4. Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working



C:\OpenVSP\repo\src\CMakeLists.txt - Notepad++

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

```

73  ADD_SUBDIRECTORY (.xmlvsp)
74  ADD_SUBDIRECTORY (.geom_core)
75  ADD_SUBDIRECTORY (.cfd_mesh)
76
77  IF (.NOT VSP_NO_GRAPHICS)
78  →ADD_SUBDIRECTORY (.gui_and_draw)
79  ENDF
80
81  ADD_SUBDIRECTORY (.geom_api)
82  ADD_SUBDIRECTORY (.vsp)
83  ADD_SUBDIRECTORY (.python_api)
84  ADD_SUBDIRECTORY (.matlab_api)
85
86  ADD_SUBDIRECTORY (.vsp_aero)
87
88  SET (.CPACK_GENERATOR ZIP)
89  SET (.CPACK_PACKAGE_NAME OpenVSP)
90  SET (.CPACK_PACKAGE_VERSION_MAJOR ${VSPVER_MAJOR})
91  SET (.CPACK_PACKAGE_VERSION_MINOR ${VSPVER_MINOR})
92  SET (.CPACK_PACKAGE_VERSION_PATCH ${VSPVER_PATCH})
93

```

Tells CMAKE where the source code is for the MATLAB API is



# Build Process – 4. Create a 'matlab\_api'

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Point the VSP CMAKE options to look for the swig.exe that you just created
4. **Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)**
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working

File Explorer path: This PC > OS (C:) > OpenVSP > repo > src > matlab\_api

Name	Date modified	Type	Size
APITestMain.m	9/12/2019 1:35 PM	MATLAB Code	2 KB
APITestSuite_test.m	9/12/2019 1:35 PM	MATLAB Code	38 KB
APITestSuiteVSPAERO_test.m	9/12/2019 1:35 PM	MATLAB Code	84 KB
assert_delta.m	9/12/2019 1:35 PM	MATLAB Code	1 KB
CMakeLists.txt	9/12/2019 1:35 PM	TXT File	5 KB
Matlabdef.def	9/12/2019 1:35 PM	DEF File	1 KB
vsp.i	9/12/2019 1:35 PM	ideaMaker Print File	1 KB
vsp_common.i	9/12/2019 1:35 PM	ideaMaker Print File	2 KB
vsp_g.i	9/12/2019 1:35 PM	ideaMaker Print File	1 KB

Copy \*.i files from python API example

## Build Process – 4. Create a 'matlab\_api'

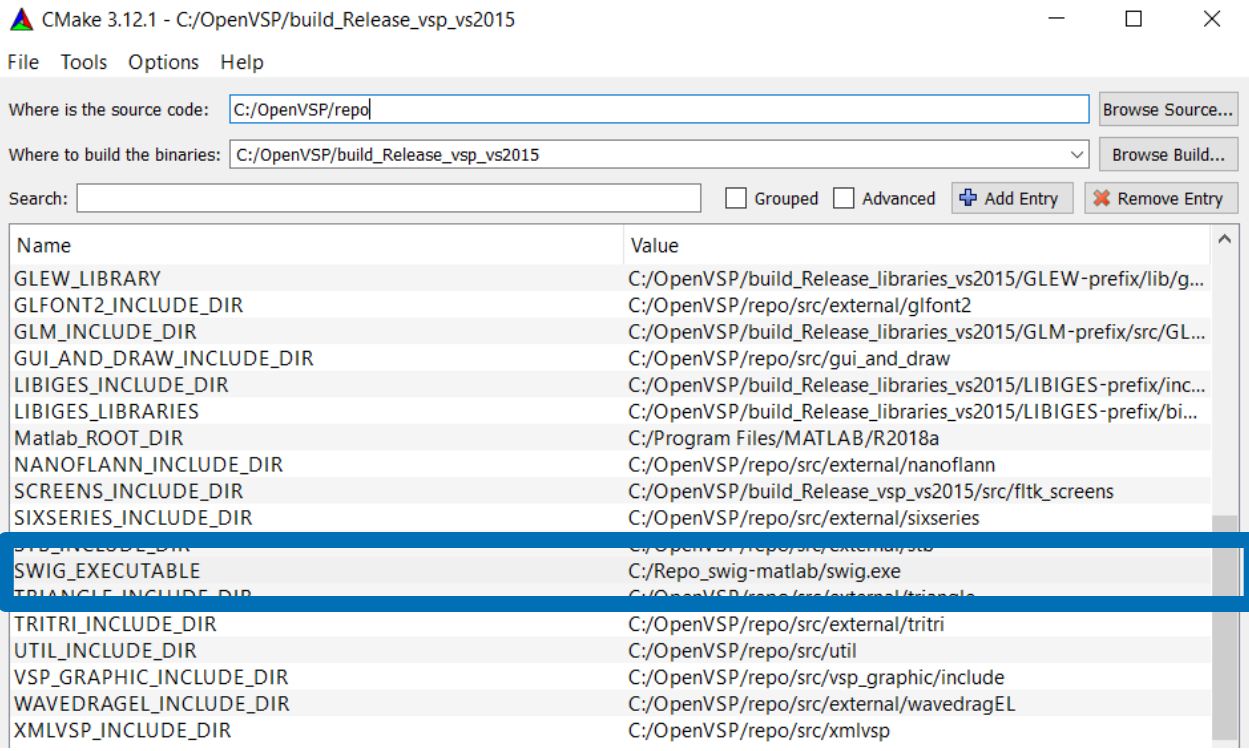
1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Point the VSP CMAKE options to look for the swig.exe that you just created
4. Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working

## Key changes to VSP source

- Explicit control of VSPAERO path
- Reference to Matrix.h

# Build Process – 4. Point the VSP CMAKE options to look for the swig.exe

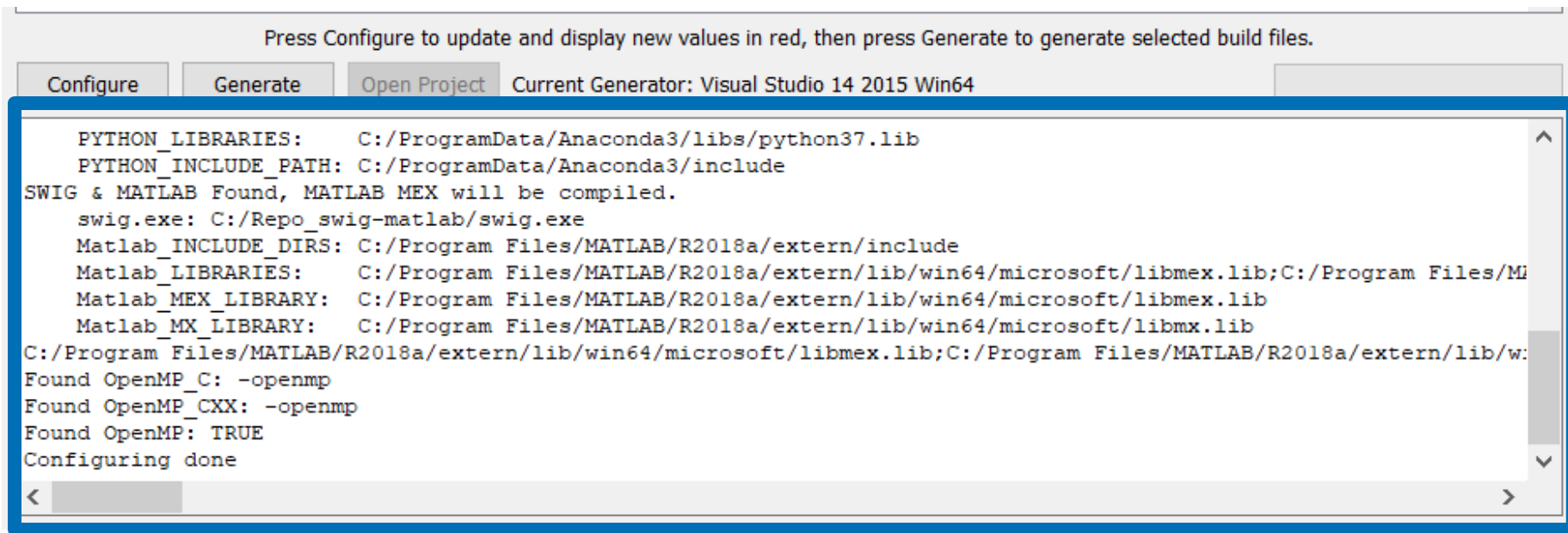
1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)
4. **Point the VSP CMAKE options to look for the swig.exe that you just created**
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working



# Build Process – 4. Point the VSP CMAKE options to look for the swig.exe

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)
4. Point the VSP CMAKE options to look for the swig.exe that you just created
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working

If you are successful....



Press Configure to update and display new values in red, then press Generate to generate selected build files.

Configure   Generate   Open Project   Current Generator: Visual Studio 14 2015 Win64

```

PYTHON_LIBRARIES: C:/ProgramData/Anaconda3/libs/python37.lib
PYTHON_INCLUDE_PATH: C:/ProgramData/Anaconda3/include
SWIG & MATLAB Found, MATLAB MEX will be compiled.
swig.exe: C:/Repo_swig-matlab/swig.exe
Matlab_INCLUDE_DIRS: C:/Program Files/MATLAB/R2018a/extern/include
Matlab_LIBRARIES: C:/Program Files/MATLAB/R2018a/extern/lib/win64/microsoft/libmex.lib;C:/Program Files/M
Matlab_MEX_LIBRARY: C:/Program Files/MATLAB/R2018a/extern/lib/win64/microsoft/libmex.lib
Matlab_MX_LIBRARY: C:/Program Files/MATLAB/R2018a/extern/lib/win64/microsoft/libmx.lib
C:/Program Files/MATLAB/R2018a/extern/lib/win64/microsoft/libmex.lib;C:/Program Files/MATLAB/R2018a/extern/lib/w
Found OpenMP_C: -openmp
Found OpenMP_CXX: -openmp
Found OpenMP: TRUE
Configuring done
  
```

# Build Process – 5. Compile OpenVSP

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Point the VSP CMAKE options to look for the swig.exe that you just created
4. Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)
5. **Compile OpenVSP (Build solution, Build INSTALL)**
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working

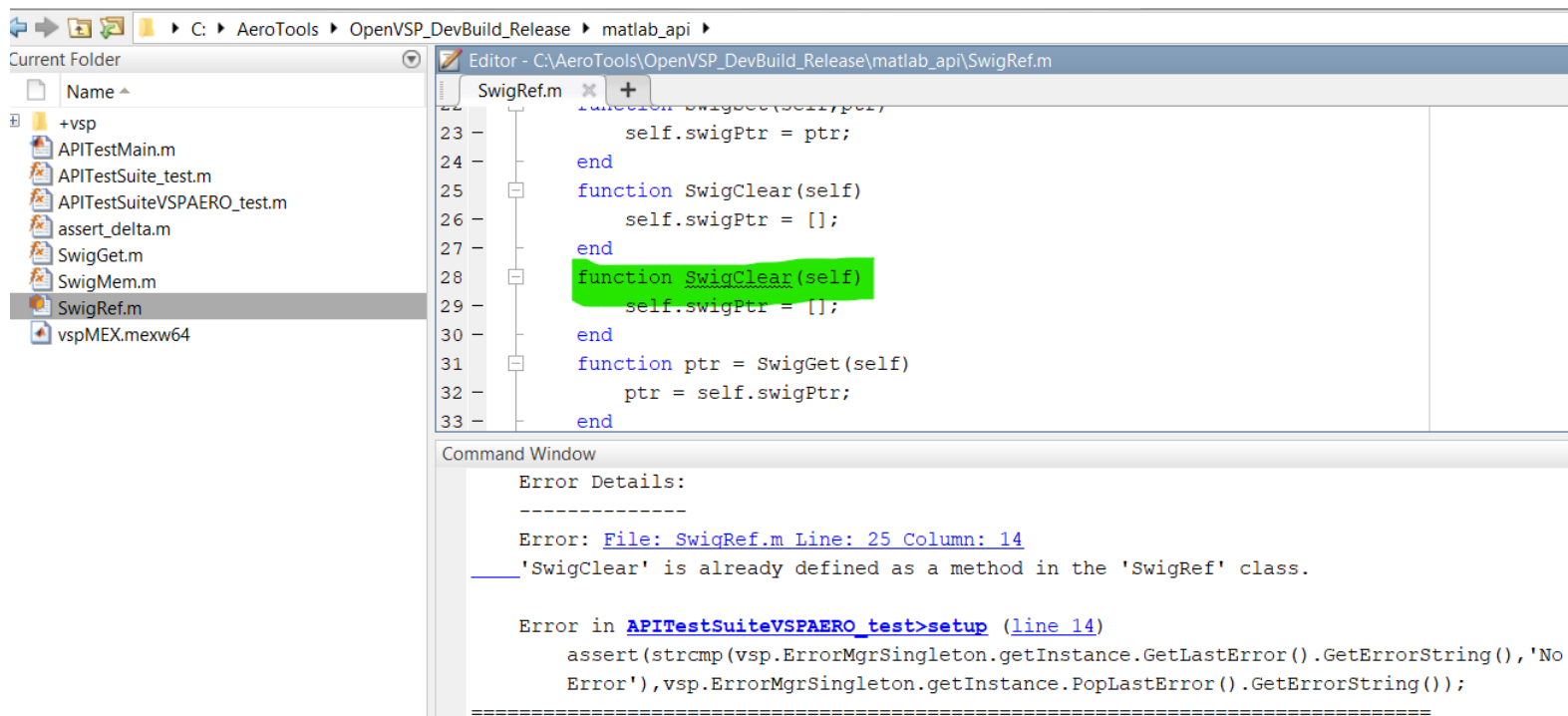
```

Administrator: Windows PowerShell
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/+vsp/Y_PROJ.m
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/+vsp/Z_DIR.m
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/+vsp/Z_PROJ.m
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/vspMEX.mexw64
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/SwigGet.m
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/SwigMem.m
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/SwigRef.m
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/APITestMain.m
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/APITestSuite_test.m
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/APITestSuiteVSPAERO_test.m
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/matlab_api/assert_delta.m
-- Up-to-date: C:/AeroTools/OpenVSP_DevBuild_Release/.msvcvp140.dll
-- Up-to-date: C:/AeroTools/OpenVSP_DevBuild_Release/.vcruntime140.dll
-- Up-to-date: C:/AeroTools/OpenVSP_DevBuild_Release/.concrtr140.dll
-- Up-to-date: C:/AeroTools/OpenVSP_DevBuild_Release/.vcomp140.dll
-- Up-to-date: C:/AeroTools/OpenVSP_DevBuild_Release/.
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/.vspero.exe
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/.vspviewer.exe
-- Installing: C:/AeroTools/OpenVSP_DevBuild_Release/.vsp slicer.exe
FinalizeBuildStatus:
  Deleting file "x64\Release\INSTALL\INSTALL.tlog\unsuccessfulbuild".
  Touching "x64\Release\INSTALL\INSTALL.tlog\INSTALL.lastbuildstate".
Done Building Project "C:\OpenVSP\build_Release_vsp_vs2015\INSTALL.vcxproj" (default targets).

Build succeeded.
    0 Warning(s)
    0 Error(s)
  
```

# Build Process – 6. Delete duplicate SwigClear()

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)
4. Point the VSP CMAKE options to look for the swig.exe that you just created
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working



The screenshot shows the MATLAB editor with the file `SwigRef.m` open. The file contains the following code:

```

22 - function SwigGet(self, ptr)
23 -     self.swigPtr = ptr;
24 - end
25 - function SwigClear(self)
26 -     self.swigPtr = [];
27 - end
28 - function SwigClear(self)
29 -     self.swigPtr = [];
30 - end
31 - function ptr = SwigGet(self)
32 -     ptr = self.swigPtr;
33 - end

```

The function `function SwigClear(self)` on line 28 is highlighted in green. The Command Window shows the following error message:

```

Error Details:
-----
Error: File: SwigRef.m Line: 25 Column: 14
_____ 'SwigClear' is already defined as a method in the 'SwigRef' class.

Error in APITestSuiteVSPAERO_test>setup (line 14)
    assert(strcmp(vsp.ErrorMgrSingleton.getInstance.GetLastError().GetErrorString(), 'No
    Error'), vsp.ErrorMgrSingleton.getInstance.PopLastError().GetErrorString());

```

# Build Process – 7. Build some test scripts

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)
4. Point the VSP CMAKE options to look for the swig.exe that you just created
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working

## APITestMain.m

```

1  %%APITestMain
2  clc
3  clear all
4  close all
5
6  %% Setup globals
7  global TEST_TOL;
8  TEST_TOL = 1e-3; %% used in functions for delta assessment
9
10 global m_vspfname_for_vspaerotests;
11 m_vspfname_for_vspaerotests = 'apitest_TestVSPAero.vsp3';
12
13 global m_vspfname_for_vspaerofunctionalitytests;
14 m_vspfname_for_vspaerofunctionalitytests = 'apitest_TestVSPAeroFunctionality.vsp3';
15
16 global VSPAERO_PATH
17 VSPAERO_PATH = [pwd '\..\'];
18
19 %% Execute tests
20 %% Simple run syntax
21 % results = run(APITestSuiteTest);
22 % rt = table(results)
23
24 % Complex runner with progress display
25 import matlab.unittest.TestRunner
26 import matlab.unittest.TestSuite
27 % import matlab.unittest.plugins.TestRunProgressPlugin
28 % import matlab.unittest.plugins.FailureDiagnosticsPlugin
29
30 % Create silent test runner and add plug-in to display progress
31 runner = TestRunner.withTextOutput;
32 % runner.addPlugin(TestRunProgressPlugin.withVerbosity(matlab.unittest.Verbosity.Detailed))
33 % runner.addPlugin(FailureDiagnosticsPlugin)
34
35 % Run general tests
36 suite = TestSuite.fromFile('APITestSuite_test.m');
37 result = run(runner, suite)
38
39 % Run VSPAERO unit tests
40 suiteVSPAERO = TestSuite.fromFile('APITestSuiteVSPAERO_test.m');
41 result = run(runner, suiteVSPAERO([?]))

```

## APITestSuiteTest.m

```

1  %%APITestSuite
2  function tests = APITestSuiteTest()
3  ... tests = fun
4  end
5
6  %% Fresh Fixture Functions
7  function setup(~) %% do not change function name
8  ... vsp.VSPCheckSetup();
9  ... vsp.VSPRenew();
10 end
11
12 function teardown(~) %% do not change function name
13 end
14
15 %% void APITestSuite::CheckSetup()
16 function CheckSetup_test(~)
17 ... fprintf('APITestSuite::CheckSetup()\n');
18 ...
19 ... vsp.VSPCheckSetup();
20 ... vsp.VSPRenew();
21
22 ... assert(strcmp(vsp.ErrorMgrSingleton.GetInstance.GetLastError().GetErrorString(), 'No Error'), vsp.ErrorMgr.PopErrorAndPrint(-stdout -)); ... %% // PopErrorAndPrint returns TRUE if there
23
24 end
25
26 %% void APITestSuite::CreateGeometry()
27 function CreateGeometry_test(~)
28
29 ... fprintf('APITestSuite::CreateGeometry()\n');
30 ...
31 ... types = vsp.GetGeomTypes();
32 ... assert(numel(types) > 0)
33
34 ... fprintf('\t[ndx]\t%20s\t[geom_id]\t%25s\t[num_geoms]\n', '[geom_type]', '[geom_type]');
35 ... extra_geoms = 0;
36
37 for i_geom_type = 1:numel(types)
38 ... %% // == Create geometry =====
39 ... fprintf('\t%d\t%20s', i_geom_type, types{i_geom_type});
40 ... if strcmp(types{i_geom_type}, 'CONFORMAL')
41 ... geom_id = vsp.AddGeom(.types{i_geom_type});

```

# Questions?



## Contact Information

Nick Brake

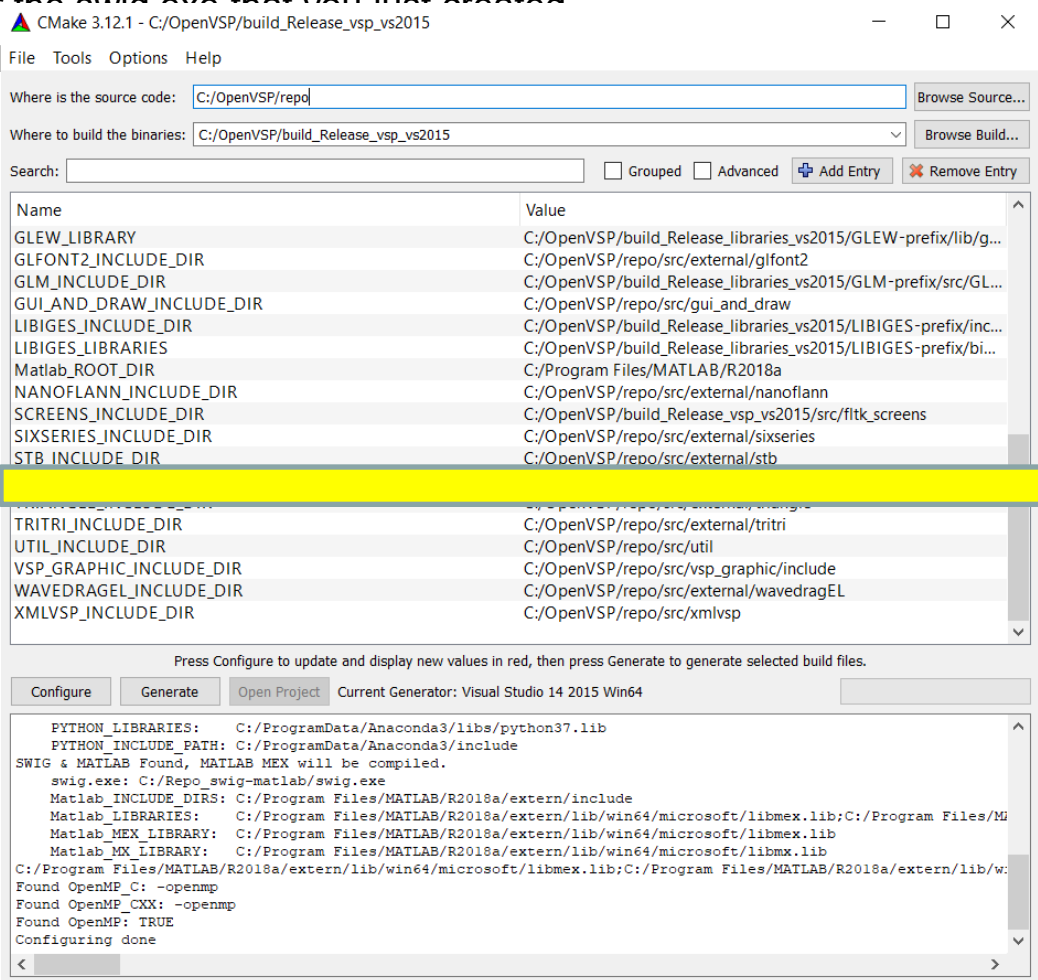
[nick.brake@esaero.com](mailto:nick.brake@esaero.com)

ESAero

[openvsp@esaero.com](mailto:openvsp@esaero.com)

# Build Process – 4. Point the VSP CMAKE options to look for the swig.exe

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)
4. Point the VSP CMAKE options to look for the swig.exe that you just created
5. Compile OpenVSP (Build solution, Build)
6. Delete duplicate SwigClear(self) function
7. Build some test scripts in matlab to verify



Documentation:  
.\Repo\_swig-matlab\Doc\Manual

Instructions exist to build using

- MinGW & MSYS
- Cygwin
- VisualStudio IDE (a bit manual)

# Build Process – Creation of Interface files

1. Clone the swig-matlab repo
2. Compile swig-matlab from source and generate a swig.exe
3. Point the VSP CMAKE options to look for the swig.exe that you just created
4. **Create a 'matlab\_api' (use the 'python\_api' as a template, this is the hardest step)**
5. Compile OpenVSP (Build solution, Build INSTALL)
6. Delete duplicate SwigClear(self) function in SwigRef.m
7. Build some test scripts in matlab to verify everything is working

Documentation: