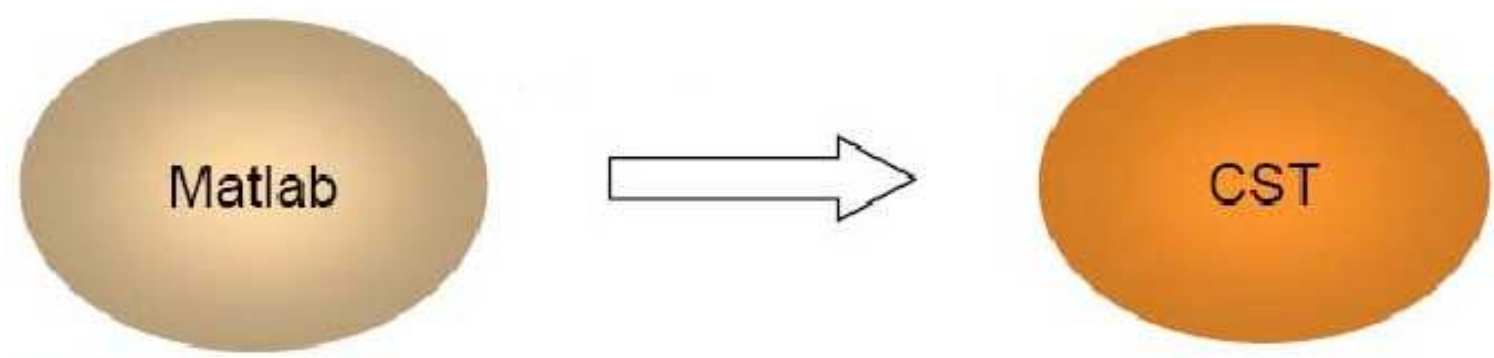
Application Note – Calling CST Studio from Matlab

Abstract

This application note gives background information on linking CST Studio to Matlab, by calling CST directly from Matlab.



For information on how to call Matlab from within CST Studio, please see the separate Application Note CST_USER_NOTE__MATLAB_COM_DDE.pdf .

Methods

There are two main ways to call CST Studio from within Matlab:

1) launch CST from Matlab command line and use a separate VBA file to instruct CST what operations to do
2) run a Matlab .m file containing CST Studio commands

1. Launch CST from Matlab command line

This is the simplest way to run CST from within Matlab. It is appropriate when the CST model name and operations to be performed are known in advance and do not need to change.

At the Matlab command line, enter:

```
>> ! "CST_DS_Path\CST DESIGN ENVIRONMENT.exe" -m cmdfile.bas
```

where *CST_DS_Path* is the path to the CST STUDIO SUITE installation directory, and *cmdfile*.bas is a Basic (VBA) file containing the commands to be executed within CST STUDIO SUITE.

1.1. Example 1: launch CST, open an existing model, solve, save

For example, when *CST_DS_Path* is

```
ProgramFiles\CSTSTUDIO SUITE 2006
```

and the command file name (including path) is

```
C:\Work\Start_CST.bas
```

the Matlab command line would look like follows:

```
>> ! "C:\ProgramFiles\CSTSTUDIO SUITE 2006\CST DESIGN ENVIRONMENT.exe" -m C:\Work\Start_CST.bas
```

The contents of the Start_CST.bas file could look as follows:

```
Sub Main
        OpenFile("C:\work\test1.cst")
        Solver.Start
        Save
End Sub
```

This simple VBA script opens the CST model called test1.cst, starts the solver, saves the model and then exits.

2. Launch and control CST from a Matlab .m file

In this case, all the commands that need to be executed in CST, instead of being contained in a VBA .bas file are contained in a Matlab .m file. They could as well be entered directly from the Matlab prompt.
This technique has the advantage that it offers complete control of CST MWS from within Matlab. Various operations, like solving several models, performing parameter studies, changing a model, saving or discarding MWS results, etc. are therefore accessible.

The user has access to all CST-specific VBA commands, by means of the Matlab command `invoke`.

2.1. Starting CST DESIGN ENVIRONMENT and opening an MWS window

The first step is typically **starting the CST DESIGN ENVIRONMENT**. To perform this operation, the following command syntax is used:

```
cst = actxserver('CSTStudio.application')
```

"`cst`" can be replaced by any other variable name.

To **open a new MWS window** in CST DS, the following syntax is used:

```
mws = invoke(cst, 'NewMWS')
```

where "`mws`" can be replaced by any other variable name.

To **open an existing MWS model**, you can (just as if you were working interactively within CST DS):
- open the model directly into CST DS:

```
mws = invoke(cst, 'OpenFile', 'filename.cst');
```

- or, first open an empty MWS window, then open the model in this window:

```
mws = invoke(cst, 'NewMWS')
invoke(mws, 'OpenFile', 'filename.cst');
```

2.2. Translating CST-specific VBA commands into Matlab

Any set of CST-specific VBA commands can be issued directly from Matlab. For example, the VBA code for the generation of a brick:

```
With Brick
        .Reset
        .Name "Brick1"
        .Component "component1"
        .Xrange "0", "1"
        .Yrange "0", "1"
        .Zrange "0", "1"
        .Material "Vacuum"
        .Create
End With
```

translates into the following Matlab code:

```
brick = invoke(mws, 'Brick');
      %create a brick in MWS
      invoke(brick, 'Reset');
      invoke(brick, 'Name', 'Brick1');
      invoke(brick, 'Component', 'component1');
      invoke(brick, 'Xrange', '0', '1');
      invoke(brick, 'Yrange', '0', '1');
      invoke(brick, 'Zrange', '0', '1');
      invoke(brick, 'Material', 'Vacuum');
      invoke(brick, 'Create');

      %... possibly create other bricks here
release(brick);
```

Note: if several bricks need to be created, the pair of commands `brick = invoke(mws, 'Brick')` and `release(brick)` only need to be issued once.

2.3. Useful commands – some examples

Find out the application name and version

```
      app = invoke(mws, 'GetApplicationName')
      ver = invoke(mws, 'GetApplicationVersion')
```

Set the units

```
      units = invoke(mws,'Units');
      invoke(units, 'Geometry','mm');
      invoke(units, 'Frequency','ghz');
      invoke(units, 'Time','s');
```

Set the value of a variable

```
      len1=40.00;
      invoke(mws, 'StoreParameter','len1',len1);
            %“mws” is the Matlab name of the MWS object, see &2.1
```

Solver commands examples

```
      solver = invoke(mws, 'Solver');
      invoke(solver, 'FrequencyRange', fr1, fr2);            % Set the
      frequency range
      invoke(solver, 'Start');
```

Working with 1D results

```
      result = invoke(mws, 'Result1D', 'a1(1)1(1)');   %Example for
                                                       %S11 linear
      numOfValue = invoke(result, 'GetN');  %Might be used to
                                            %initialize variables

      invoke(result, 'Save', 'C:\tmp\filename');
      A = importdata('C:\tmp\filename', ' ', 4)
      x = A.data(:, 1);      %x-data column
      y = A.data(:, 2);      %y-data column
```

2.4. Example 1 again: launch CST, open an existing model, solve, save

To reach the same effect as in the example shown in paragraph 1.1, the following set of commands need to be entered into the .m file:

```
cst = actxserver('CSTStudio.application')

mws = invoke(cst, 'NewMWS')
invoke(mws, 'OpenFile', 'C:\work\test1.cst');
solver = invoke(mws, 'Solver');
invoke(solver, 'Start');
invoke(mws, 'Save');

release(solver);
release(mws);
release(cst);
```

2.5. Example 2: Open an MWS file, perform a parameter study, store a result in a Matlab vector

The following code can be used to perform this operations. (The code is contained in the attached file studio3.m. It refers to the CST model tsplitter.cst.)

```
%matlab-example: tested 09-Jan 2008 with with MATLAB 7.5 (R2007b)
% Open an MWS file, perform a parameter study,
%   store a result in a Matlab vector

% To test, first copy the file "tsplitter.cst"
%   in the directory "C:\CST_Test"

% Start CST Studio
cst = actxserver('CSTStudio.application')
mws = invoke(cst, 'NewMWS')
app = invoke(mws, 'GetApplicationName')
ver = invoke(mws, 'GetApplicationVersion')

% Open the MWS model
invoke(mws, 'OpenFile', 'C:\CST_Test\tsplitter.cst');

for iii = 1:1:2       % Parameter sweep
     % Set value of parameter
     invoke(mws, 'StoreParameter','offset', iii);

     invoke(mws, 'Rebuild');

     % Start solver
     solver = invoke(mws, 'Solver');
     invoke(solver, 'Start');

     % Take over the result; here, as an example,
     %   the amplitude (linear) of S11
     result = invoke(mws, 'Result1D', 'a1(1)1(1)');
```

```
        numOfValue = invoke(result, 'GetN');  % Might be used to
                                              % initialize variables

        invoke(result, 'Save', 'C:\tmp\filename');
        A = importdata('C:\tmp\filename', ' ', 4)
        x(:, iii) = A.data(:, 1);             % x-data column
        y(:, iii) = A.data(:, 2);             % y-data column
end

invoke(mws, 'Save');
invoke(mws, 'Quit');

release(result);
release(solver);
release(mws);
release(cst);
```