MATLAB Production Server Interface for Tableau® software

Reference Architecture

Contents

Introduction

This reference architecture outlines the use of MATLAB and MATLAB Production Server for advanced analysis and analytics within the Tableau platform. There are two options for interfacing Tableau with MATLAB Production Server.

1) Web Data Connector: Retrieve data via MATLAB functions running on MATLAB Production Server for use in Tableau
2) External Service Connection: Send data from Tableau for analysis by functions running on MATLAB Production Server. This is similar to other external analytics integrations.

The following sections explain the steps involved for setting up and using both options. Please see 'Determining the correct option to use' in the Notes section at the end of this document for selecting the option most appropriate for your application.

Note: The configurations provided in this document are for reference only.

System Requirements

This reference architecture is comprised of the following components and was developed using the versions as listed. See the product documentation for any product specific requirements.

MathWorks Products

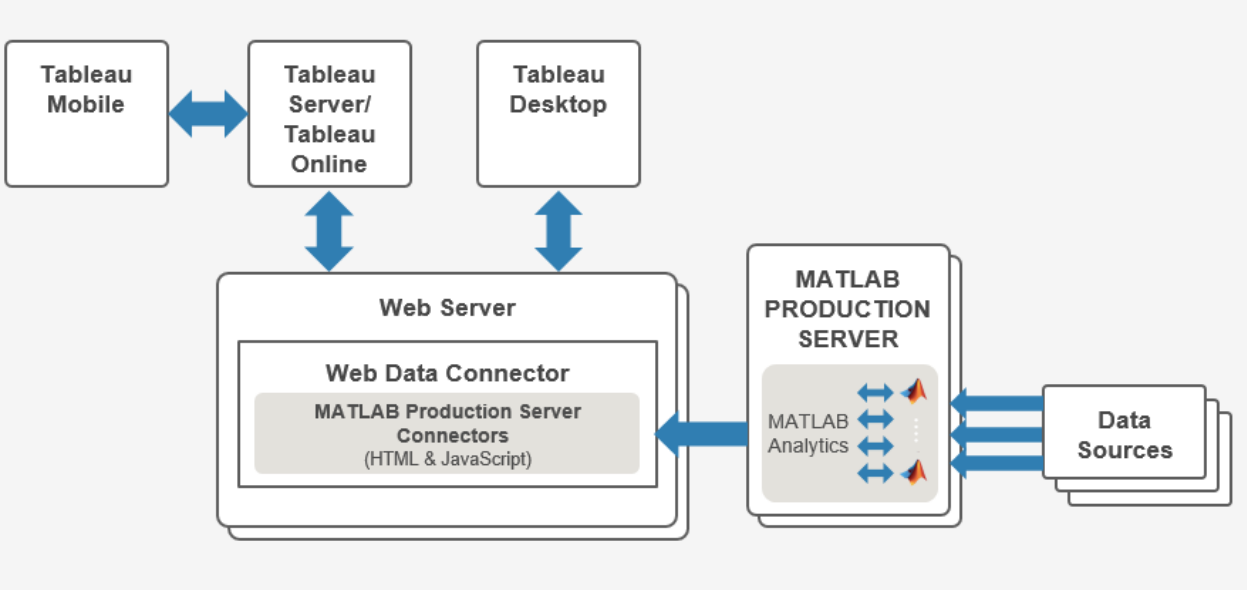
1. MATLAB (R2016b or later)
2. MATLAB Compiler SDK (R2016b or later)
3. MATLAB Production Server (R2016b or later)

Tableau Products

1. Tableau Desktop (10.3.1 or later)
2. Tableau Server (10.5.3 or later)

Option 1: Getting Started: Using Web Data Connector

Architecture Diagram



Installation and Configuration

The Web Data Connector (WDC) is a Tableau feature that helps Tableau users access any data available over HTTP from internal web services, REST APIs, JSON data etc. The WDC is an HTML page with JavaScript code to manage the communication with other web services/APIs. The HTML page can also display a UI to the Tableau user who can then select the data to be loaded.

Note that although this example uses HTTP, MATLAB Production Server supports calls using HTTPS as well. More information on security and enabling HTTPS is provided below:
https://www.mathworks.com/help/mps/security.html
Also note that for all examples in this document, the MATLAB application can be accessed via RESTful API either by

(a) Packaging and deploying the MATLAB application to MATLAB Production server or
(b) Using MATLAB Compiler SDK to start up a test server in local machine.

The first option to integrate MATLAB applications with Tableau utilizes a custom WDC component that connects to MATLAB Production Server as a data source. The HTML page is hosted on a web or application server so that it is accessible over HTTP from Tableau.

To set up the WDC for connecting to MATLAB Production Server, locate the MPS_WDC_Sunspot.html file in the package provided under \WDC\Examples\Web_Data_Connector\MPS_WDC_Sunspot.html

Although any web server can be used to host the HTML page, the example in this package has been tested using Internet Information Services(IIS) and 'http-server'

The Instructions to enable IIS on a Windows 10 machine are provided in Appendix A.

A second choice for hosting the web page is http-server, which is a simple, zero-configuration command-line HTTP server. Help on starting this server is below:

https://www.npmjs.com/package/http-server

Usage

The example used for this demo will allow Tableau users to call a MATLAB application and analyze cyclical data using a fast Fourier transform algorithm. Fourier transformations allows users to analyze variations in data, such as an event in nature over a period of time. The data retrieved here represents the number and size of sunspots for the last 300 years, using the Zurich sunspot relative number.

The function used in this MATLAB application to perform Fourier transformation is 'fft', which has a lower computational cost when compared to other direct implementations. By integrating the MATLAB analysis with Tableau, it is possible to provide Tableau users direct access to powerful analyzing capabilities in MATLAB.

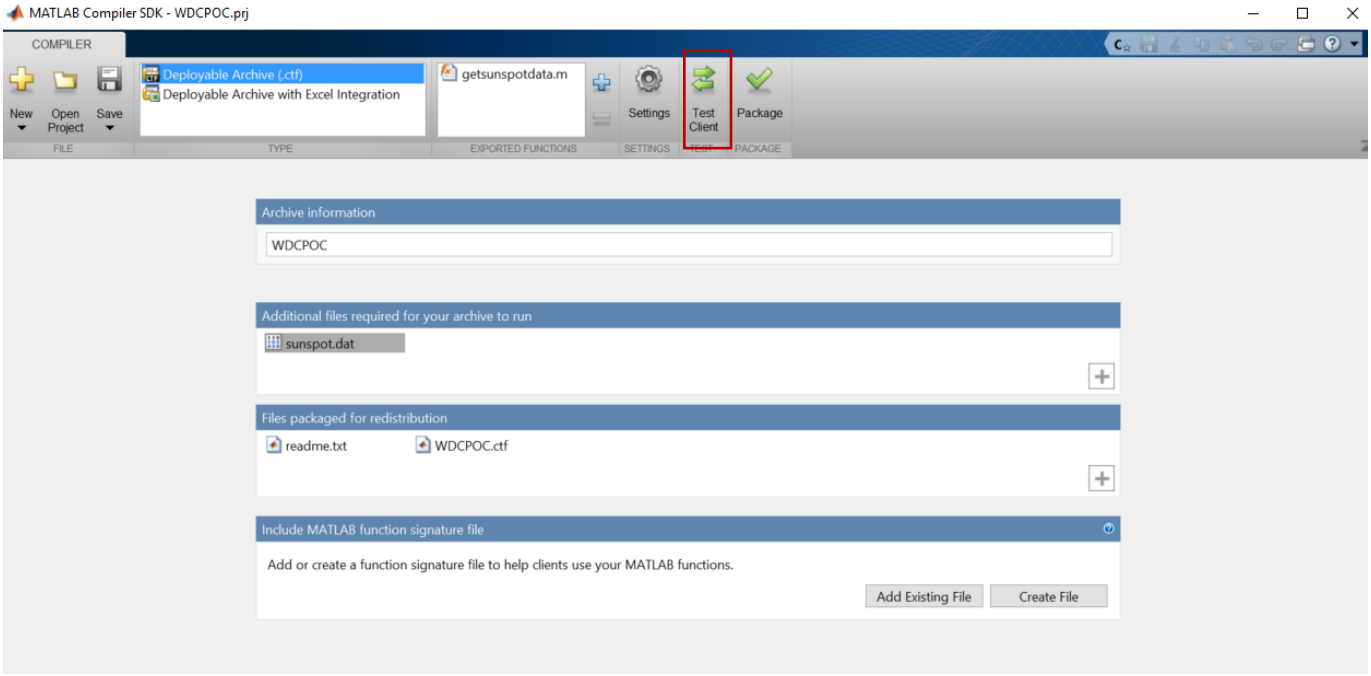
More information about this example is documented below:
https://www.mathworks.com/examples/matlab/mw/matlab_featured-ex37594814-analyzing-cyclical-data-with-fft

The data received from MATLAB is parsed by the Web Data Connector (HTML file) and sent back to Tableau, where the visualizations can be plotted to analyze trends and variations.

MATLAB Environment setup

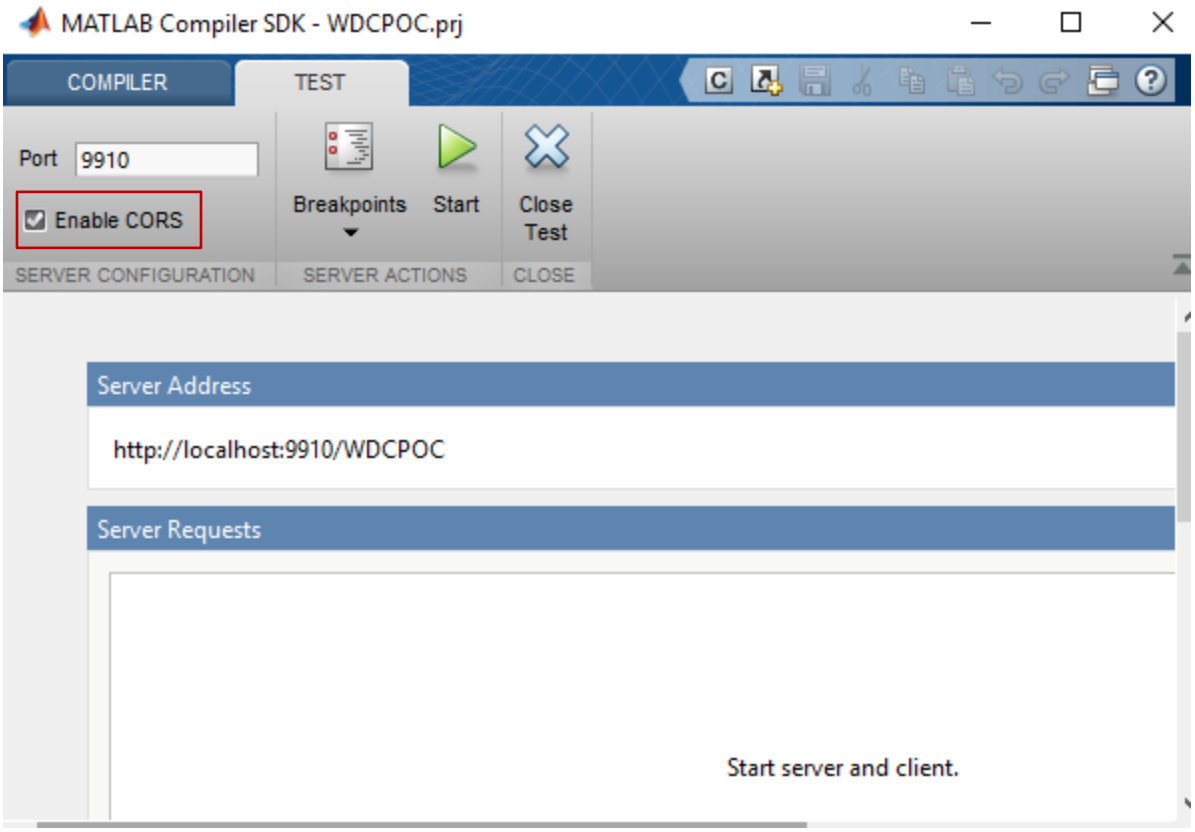
Under ~\WDC\Examples\MATLAB, locate the getsunspotdata.m file and the WDCPOC.prj file.

Open the WDCPOC.prj file in MATLAB (MATLAB Compiler SDK is required). This will bring up the UI as shown below:

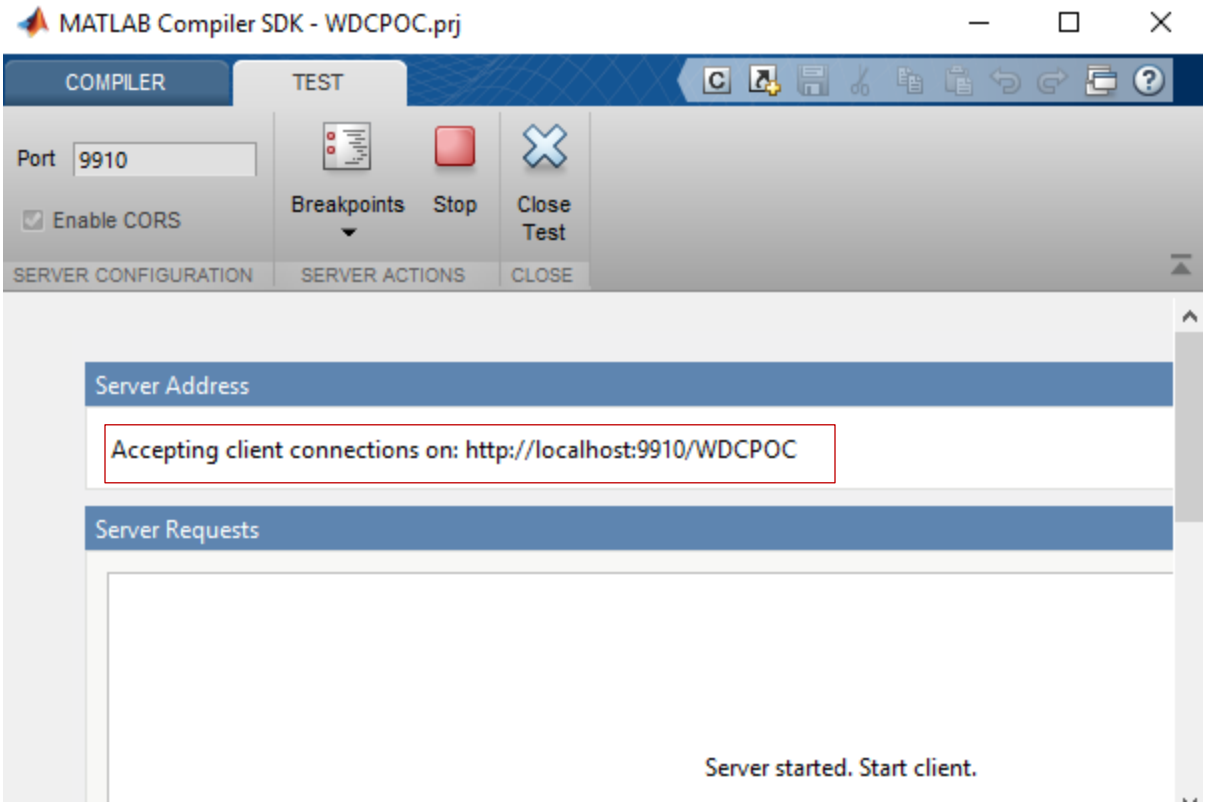


Click on the Test Client button highlighted above.

This will bring up the test environment for MATLAB Production Server. Check the ‘Enable CORS’ option and click the start button.



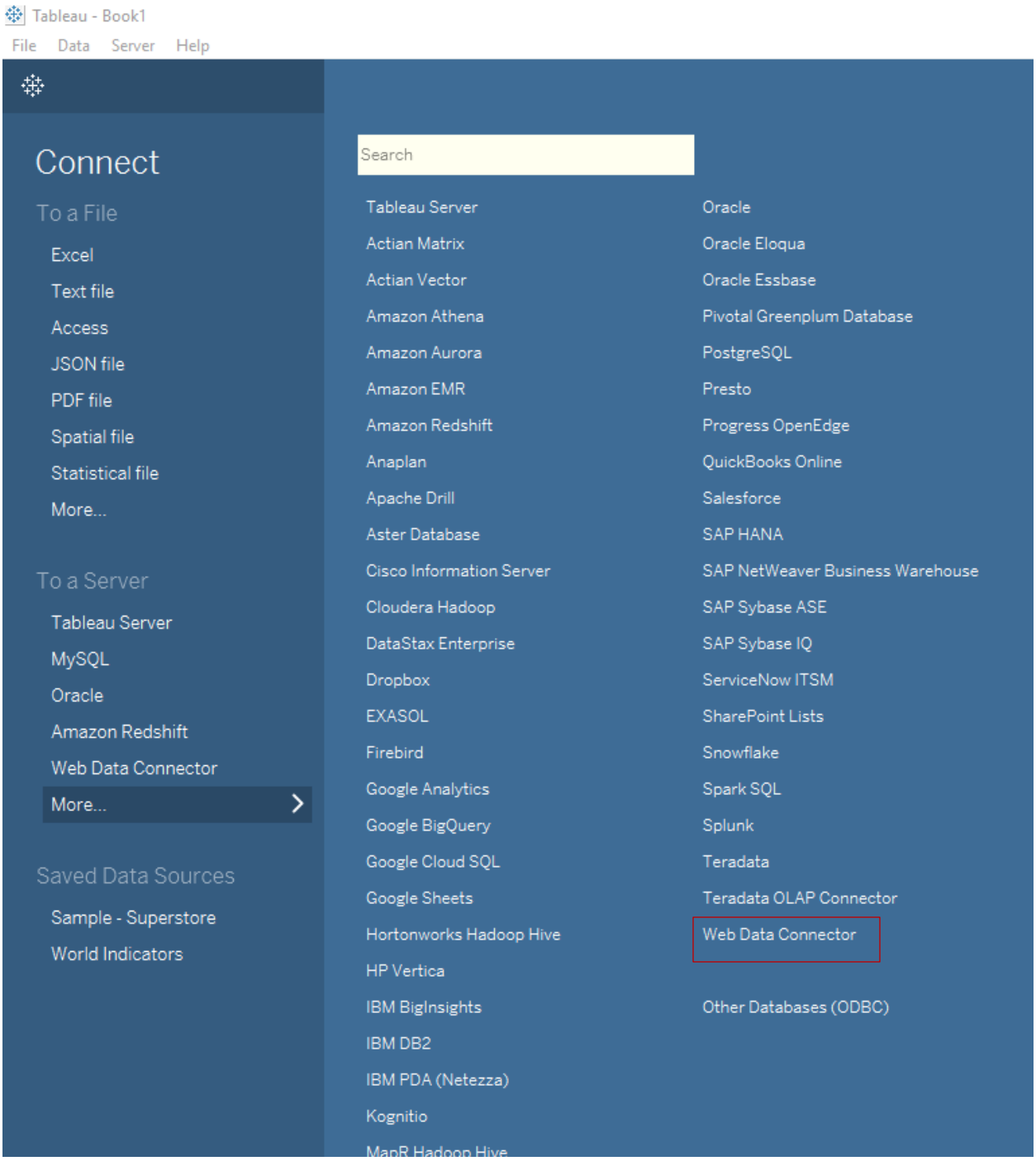
This should start up the test server that enables MATLAB developers to test MATLAB applications in a simulated deployed environment. The test server listens at localhost, and default port number 9910 as below:



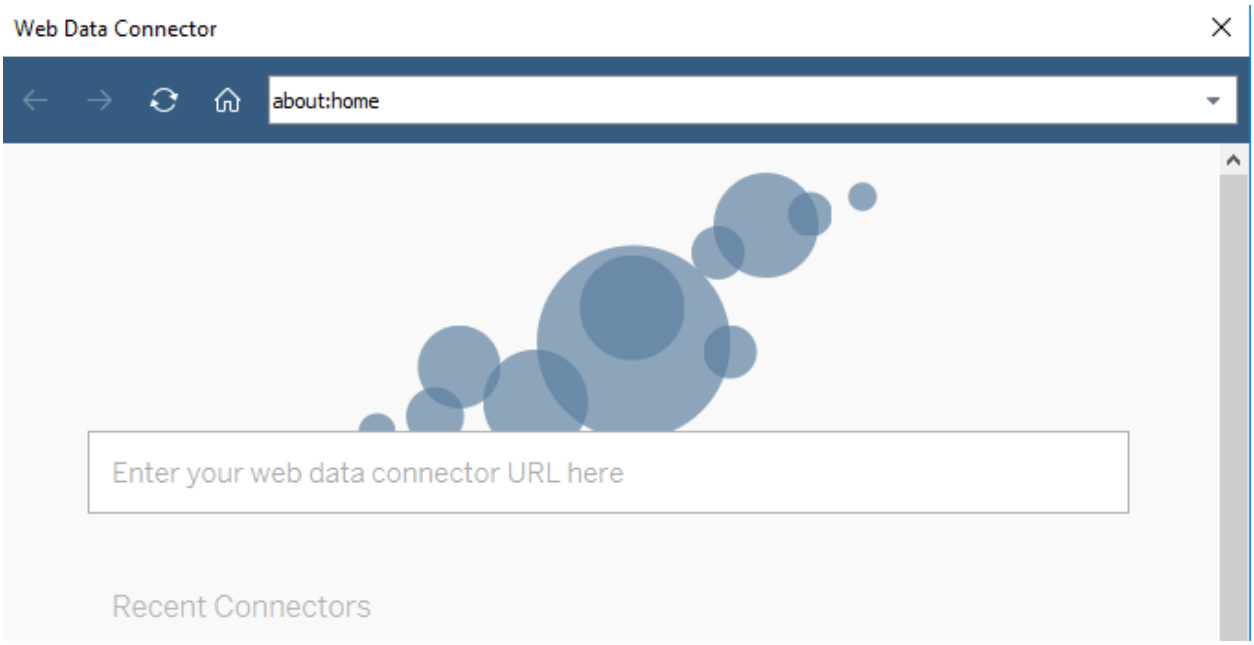
The figure above shows the test server in the MATLAB session listening at port 9910.

Tableau Environment Setup

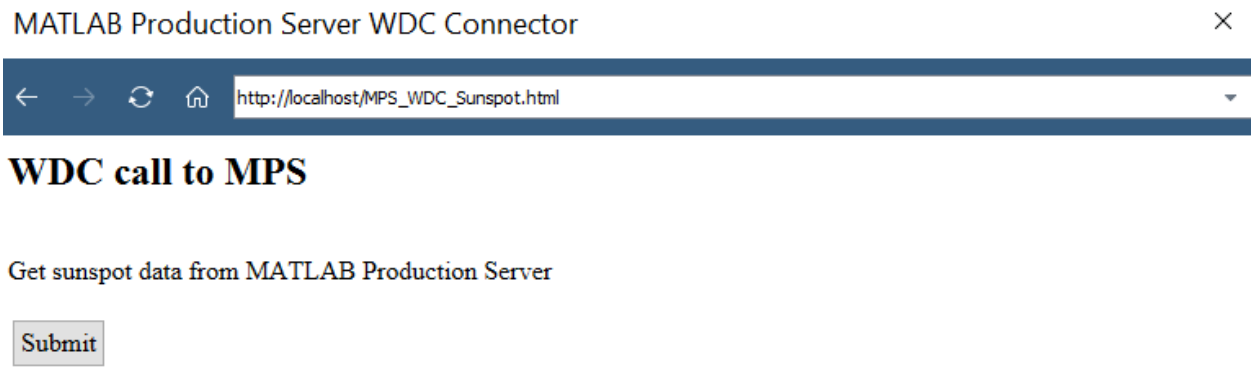
Open Tableau and under ‘Connect’ option on start page, click on Connect to Web Data Connector.



This will bring up the UI below:

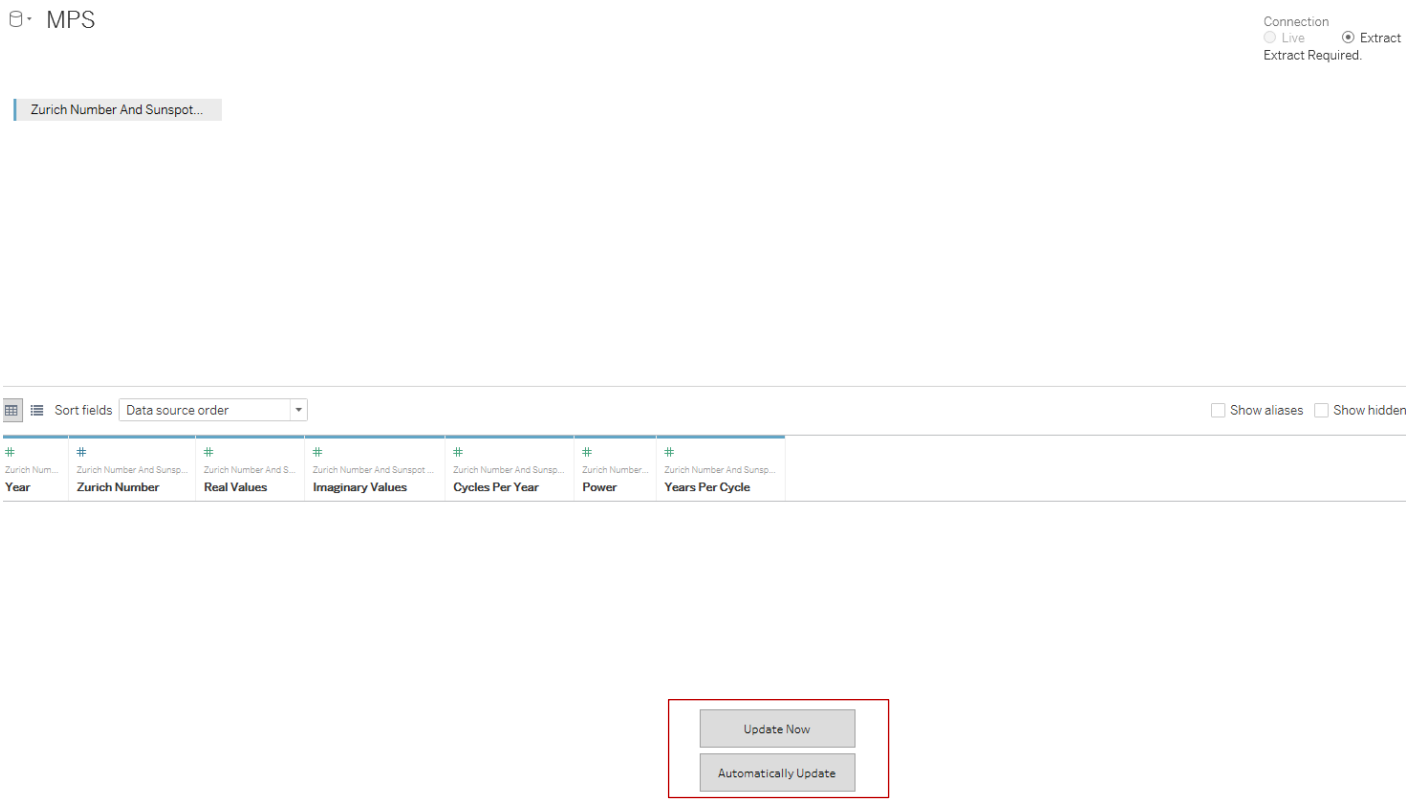


Type in http://localhost/MPS_WDC_Sunspot.html in the address bar. This should bring up the UI as below.



Click on 'Submit' button to register the connector with Tableau. The Web Data Connector provides Tableau with the schema of the table that will contain the data to be sent to Tableau from the external service (in this case from MATLAB Production Server).

Tableau will show the column names of the table to be retrieved as below:

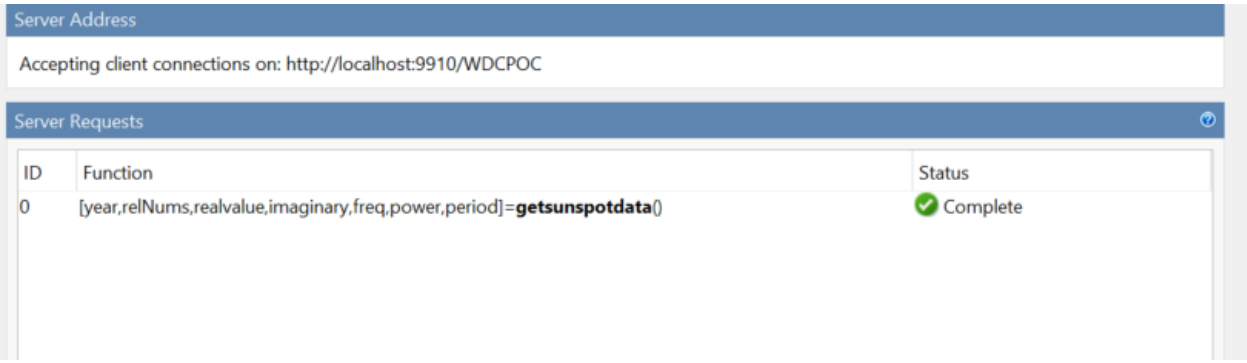


Click on the 'Update Now' button. This will initiate an HTTP call using WDC to the test server started in the previous step. The data from the sunspot data file in MATLAB is now available within Tableau as shown below:

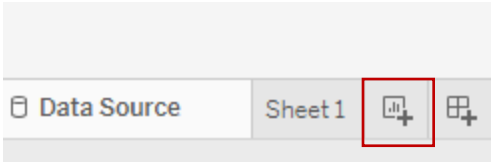
Sort fields Data source order

Year	Zurich Number	Real Values	Imaginary Values	Cycles Per Year	Power	Years Per Cycle
1700.000	5.00	790.05	405.67	0.003484	788,748.73	287.000
1701.000	11.00	119.60	879.31	0.006969	787,481.80	143.500
1702.000	16.00	-1,218.28	2,293.30	0.010453	6,743,424.55	95.667
1703.000	23.00	163.34	-141.84	0.013937	46,797.37	71.750
1704.000	36.00	-1,625.32	-63.19	0.017422	2,645,667.28	57.400
1705.000	58.00	-213.86	1,103.72	0.020906	1,263,932.29	47.833
1706.000	29.00	415.92	61.85	0.024390	176,812.70	41.000
1707.000	20.00	277.68	623.82	0.027875	466,255.08	35.875
1708.000	10.00	241.57	416.79	0.031359	232,070.73	31.889
1709.000	8.00	375.37	930.12	0.034843	1,006,034.66	28.700
1710.000	3.00	-336.91	20.44	0.038328	113,927.18	26.091
1711.000	0.00	17.98	620.58	0.041812	385,448.08	23.917
1712.000	0.00	-34.93	659.42	0.045296	436,055.51	22.077
1713.000	2.00	-563.51	-189.79	0.048780	353,567.25	20.500
1714.000	11.00	-333.92	348.18	0.052265	232,733.07	19.133
1715.000	27.00	-385.21	-100.76	0.055749	158,536.74	17.938

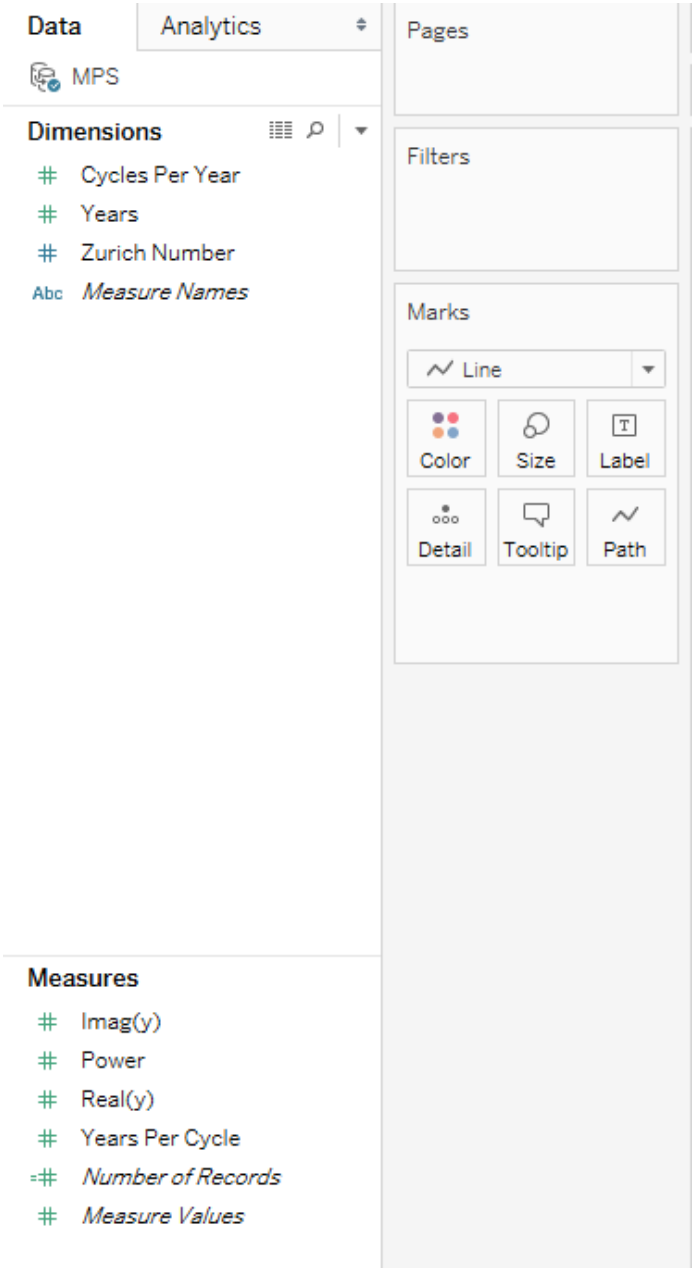
In the test server, the call from Tableau can be seen in the call logs:



Once data is available in Tableau, it is easy to plot the sunspot data and Fourier coefficients in the Tableau worksheet. To create a new sheet in Tableau, locate the icons on the bottom toolbar in Tableau as shown:

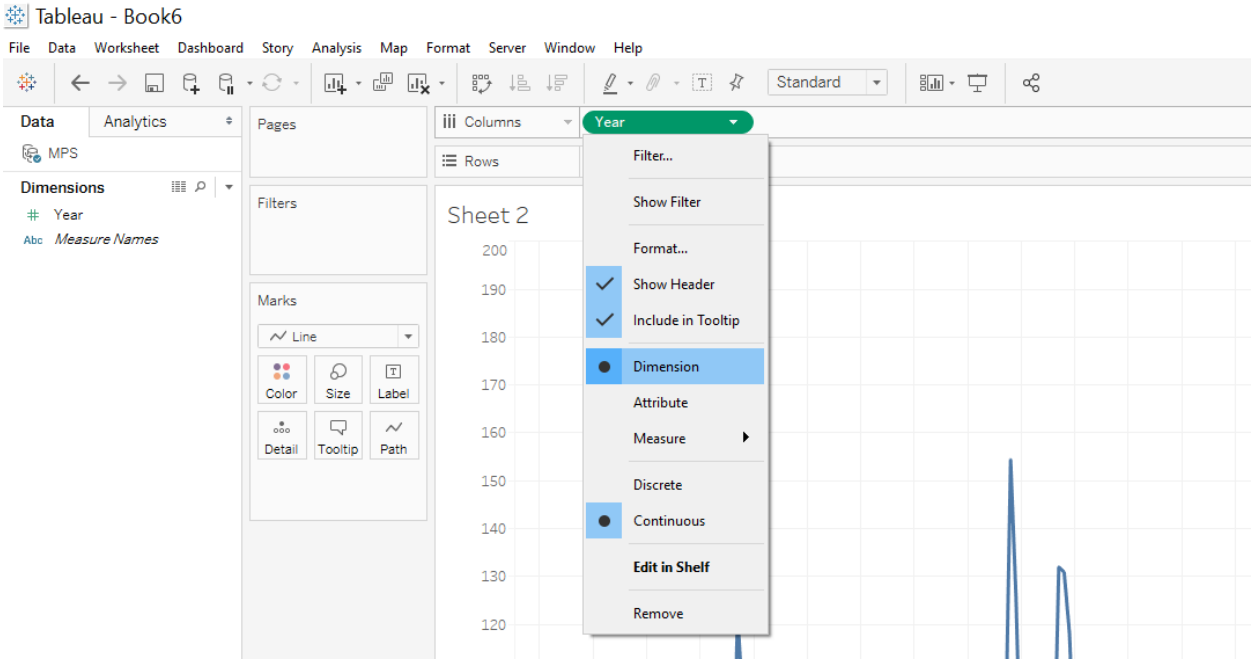


Click on a new worksheet icon highlighted above. This will open a new worksheet in Tableau with the data from MATLAB Production Server available as fields. Please note that the column names have been renamed to real(y) and imag(y). Ensure that 'Year' appears under the 'Dimensions' tab, and 'real(y), 'Zurich Number' and 'imag(y) appear under 'Measures' tab.

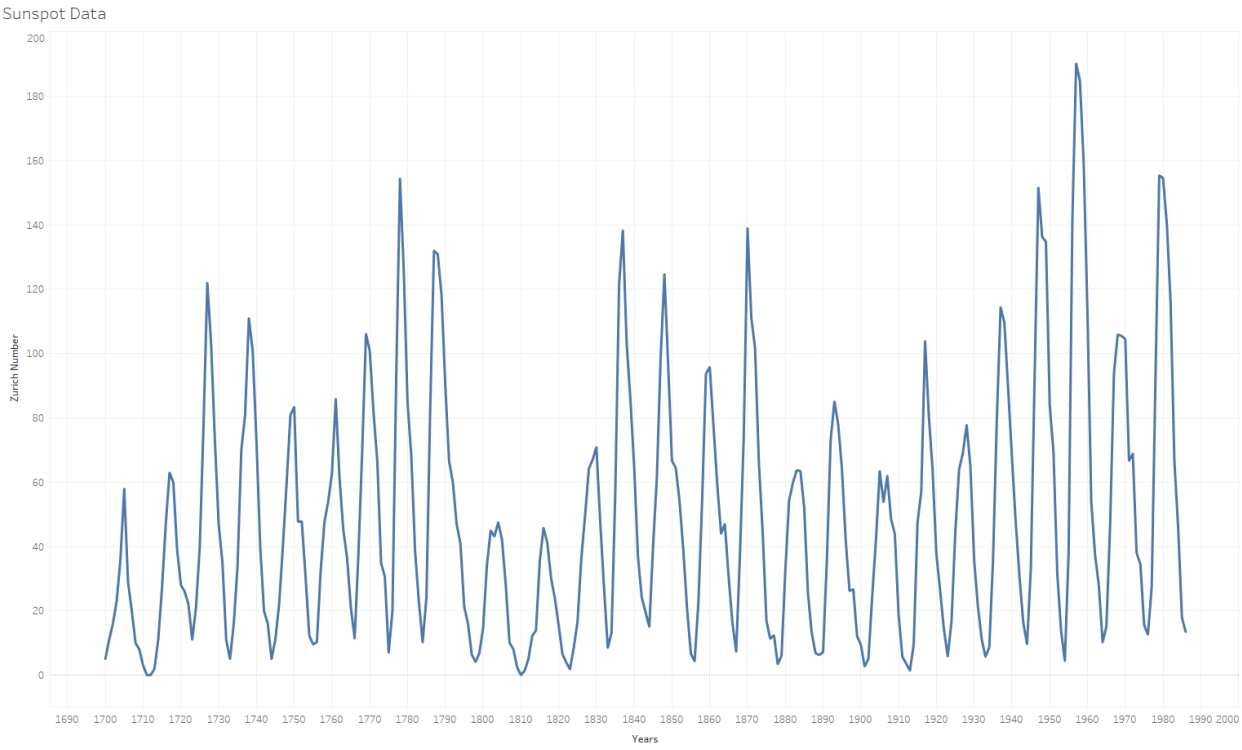


To plot the sunspot data, drag and drop the 'Year' field to Columns, and Zurich Number field to Rows. Right click Year under Columns and ensure that the dimension of the data is used, and not the attribute

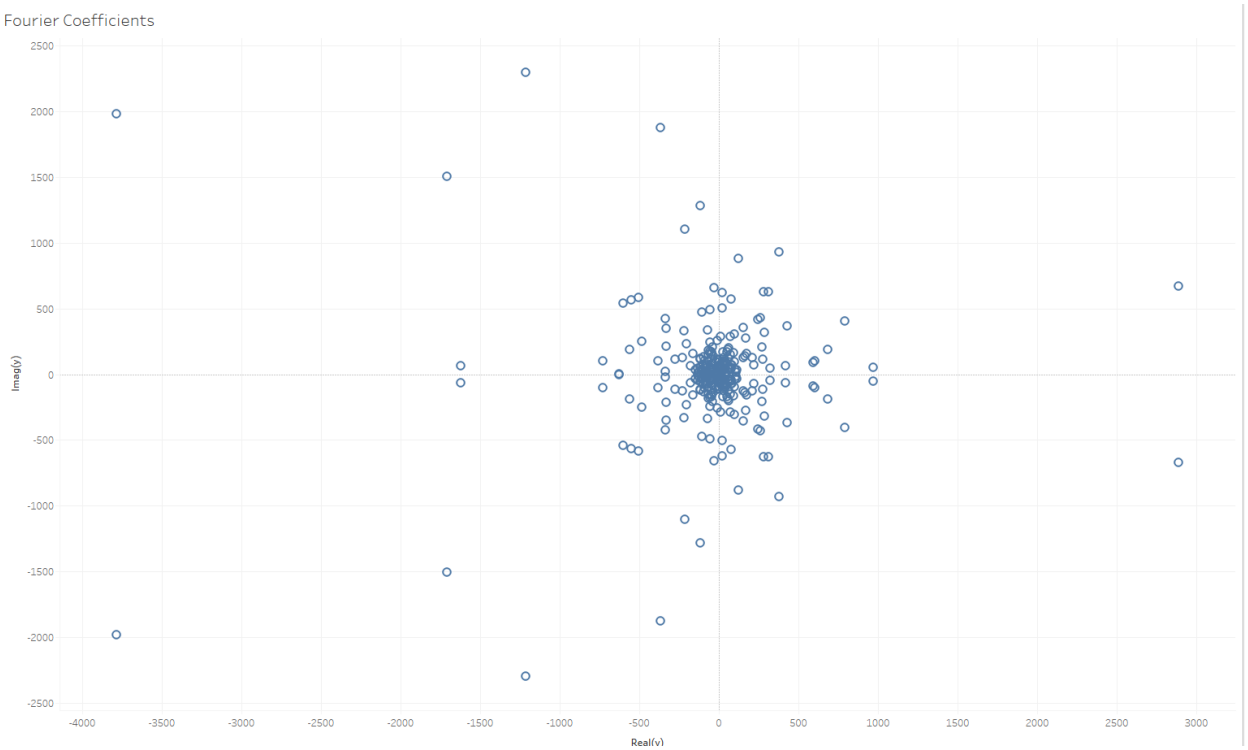
or measure as shown below. Similarly, right click on ZurichNumber in Rows and ensure ‘Dimension’ is selected.



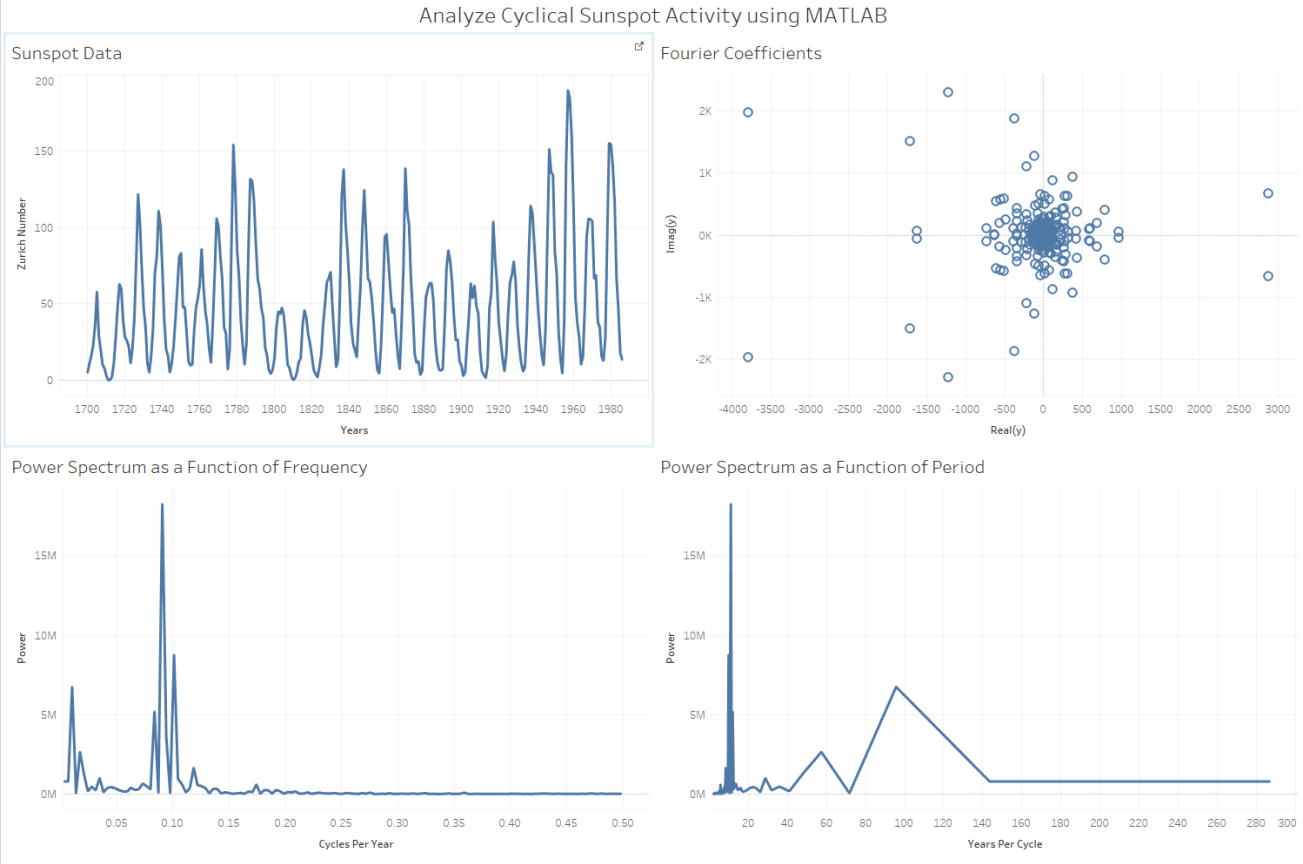
This will plot the sunspot data in Tableau as below:



To plot the Fourier coefficients, create a new sheet as described above, and drag and drop the Realvalues to Columns, and ImaginaryValues to Rows. As before, right click on the field names and ensure that that 'Dimension' is selected. This will plot the Fourier coefficients as below:



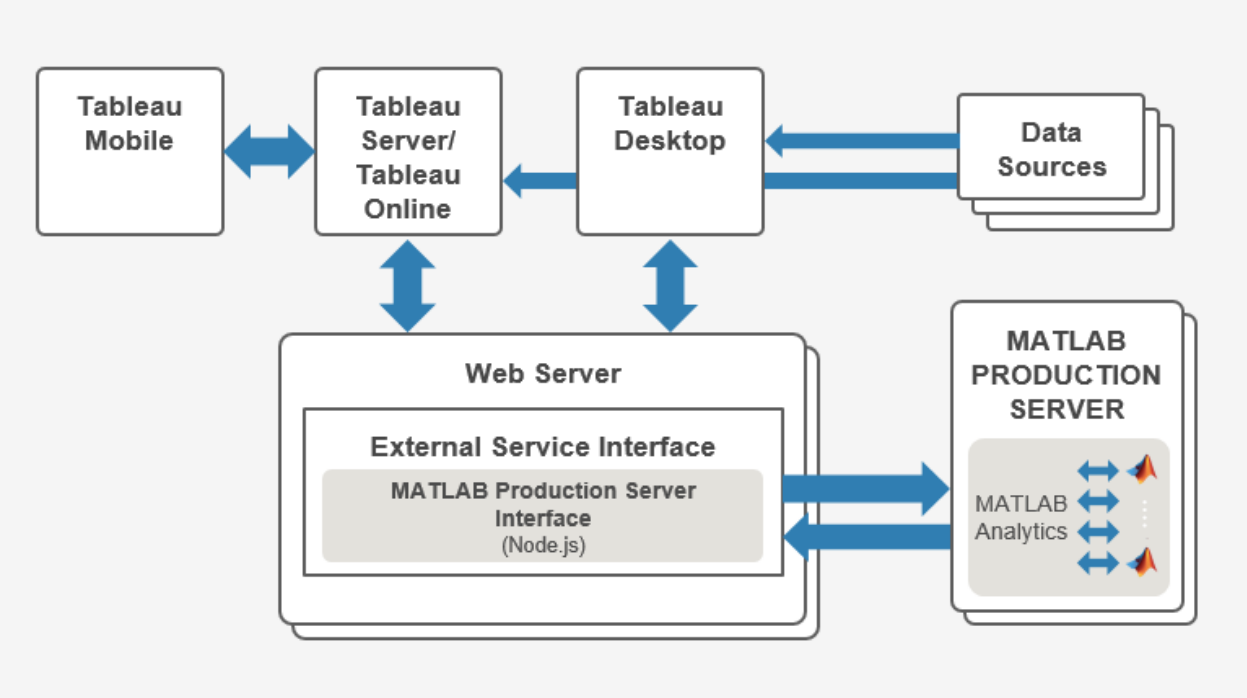
The example included in this package also plots the power spectrum as a function of frequency, and as a function of period. A dashboard can be created in Tableau displaying all the plots in a single screen.



This worksheet can also be published to Tableau Server, enabling access for multiple Tableau users using a browser. It is also possible to refresh data on a schedule on Tableau Server so that the latest data can be retrieved and plotted automatically.

Option 2: Getting Started: Using External Service Connection

Architecture Diagram



Installation and Configuration

The External Service Connection option in Tableau provides a set of functions that you can use to pass expressions to external services for integration with MATLAB. More information on this feature is available at https://onlinehelp.tableau.com/current/pro/desktop/en-us/r_connection_manage.html.

MATLAB Production Server Interface for Tableau software is an external service that Tableau users can connect to using External Service Connection. The interface is a Node.js server application that can run either on an end users' desktop machine in a test environment, or in a production environment depending upon the scaling and redundancy needs.

A license for the optimization toolbox is required to run the MATLAB example discussed in this document.

The steps involved in installation and configuration of the interface application are:

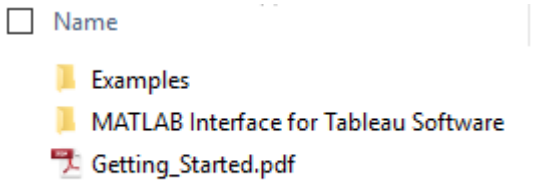
1. Install Node.js

The interface application requires Node.js to be installed in the machine before startup. Node.js can be installed from the below link:

https://nodejs.org/en/download/

2. Unpack the MATLAB interface for Tableau software

The MATLAB package “Setup.exe” contains the interface application, as well as the MATLAB code and Tableau workbook required for the example discussed in this document. Running the setup.exe installer will unpack the Getting_Started.pdf guide and 2 folders as below:



Please note that running the setup.exe will only unpack the application, installation requires an additional step as described below in (C).

The MATLAB interface for Tableau Software contains the Node.js application.

The Examples folder contains the MATLAB code and Tableau workbook required for the example discussed in this document.

3. Install the interface

Installation of Node.js will make available the package manager for JavaScript (npm). npm enables users to discover and download packages that other JavaScript users have created. Once Node.js has been installed, change directories to the \MATLAB Interface for Tableau Software\Interface folder and run the command

```
$ npm install
```

If ‘npm’ is not available from this location, use the complete path to the npm file. This default location is in the installation path for Node.js ‘C:\Program Files\nodejs’. This will install the dependencies required for the interface application.

4. Configure the environment

The interface application contains a configuration file in the location MATLAB Interface for Tableau Software\Interface\config\server.config that can be modified to reflect the environment in which the application will be running. A sample config file is as below:

```
{
  "name": "MATLAB Production Server Interface for Tableau software",
  "description": "A lightweight middleware for MATLAB Production Server",
  "state_path": "/tmp",
  "server_version": "Alpha",
  "creation_time": "",
  "port": 3001,
  "limit": "50mb",
  "mpsstarted": "MATLAB Production Server is available.",
  "mpsstopped": "MATLAB Production Server is not available",
  "authtoken":
"ap6Tp2kekrPrm5q1vUsHHVl6i8khWJrTispgOrczc20hIGRmhYPWArxOJXxrHsGf",
  "configFile": "main_config",
  "deployfolder": "C:\\MPS\\R2016B\\auto_deploy\\",
  "mps_server": "http://localhost",
  "mps_port": 9910
}
```


As can be seen in the image above, this is a JSON formatted string which can be edited to reflect the correct settings for the interface in your environment. The main values to validate are:

(i) port – This is the port number where the interface will listen and accept connections from Tableau. The default is 3001

(ii) mps_server – This is the URL for the MATLAB Production Server.

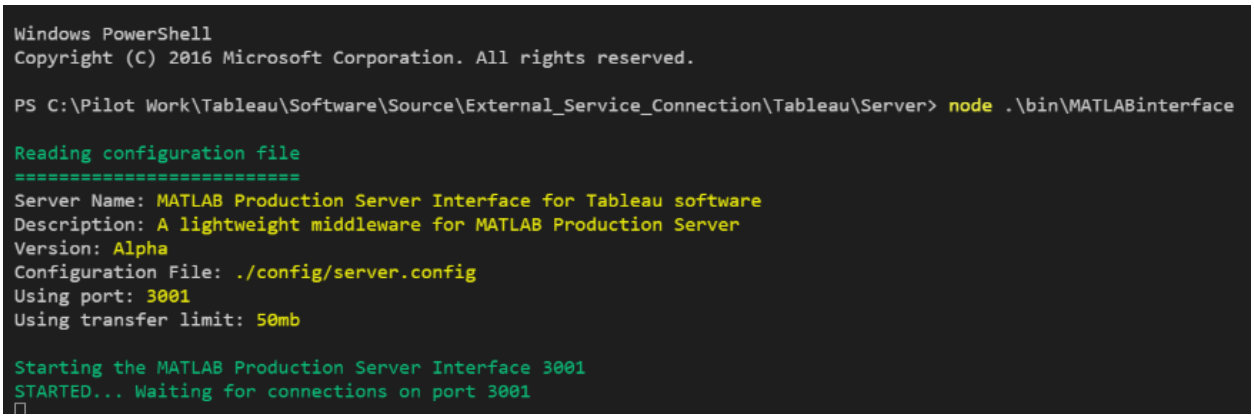
(iii) mpsPort – This is the port number where MATLAB Production Server is listening.

(iv) deployfolder – This is the folder where compiled MATLAB applications are deployed. This setting is required if the MATLAB developer wishes to publish compiled archives directly to MATLAB Production Server via the interface.

(v) authtoken - This is a unique key the MPS developer will need to provide to authorize the publishing of compiled MATLAB applications using the interface.

5. Start the server

Start the server using the command 'node bin/MATLABinterface'. This should start up the server listening at the port specified in server.config. If the command is not recognized, you may need to include the complete path to the node.exe file. Once the server starts, you should see the message as below:



Usage

The example used for this option solves the classic travelling salesman problem where the goal is to find the shortest closed path through a set of stops. MATLAB solves this problem using an iterative process by determining subtours, and rerunning the optimization until all subtours are eliminated. More information about this example can be found here:

https://www.mathworks.com/help/optim/ug/travelling-salesman-problem.html

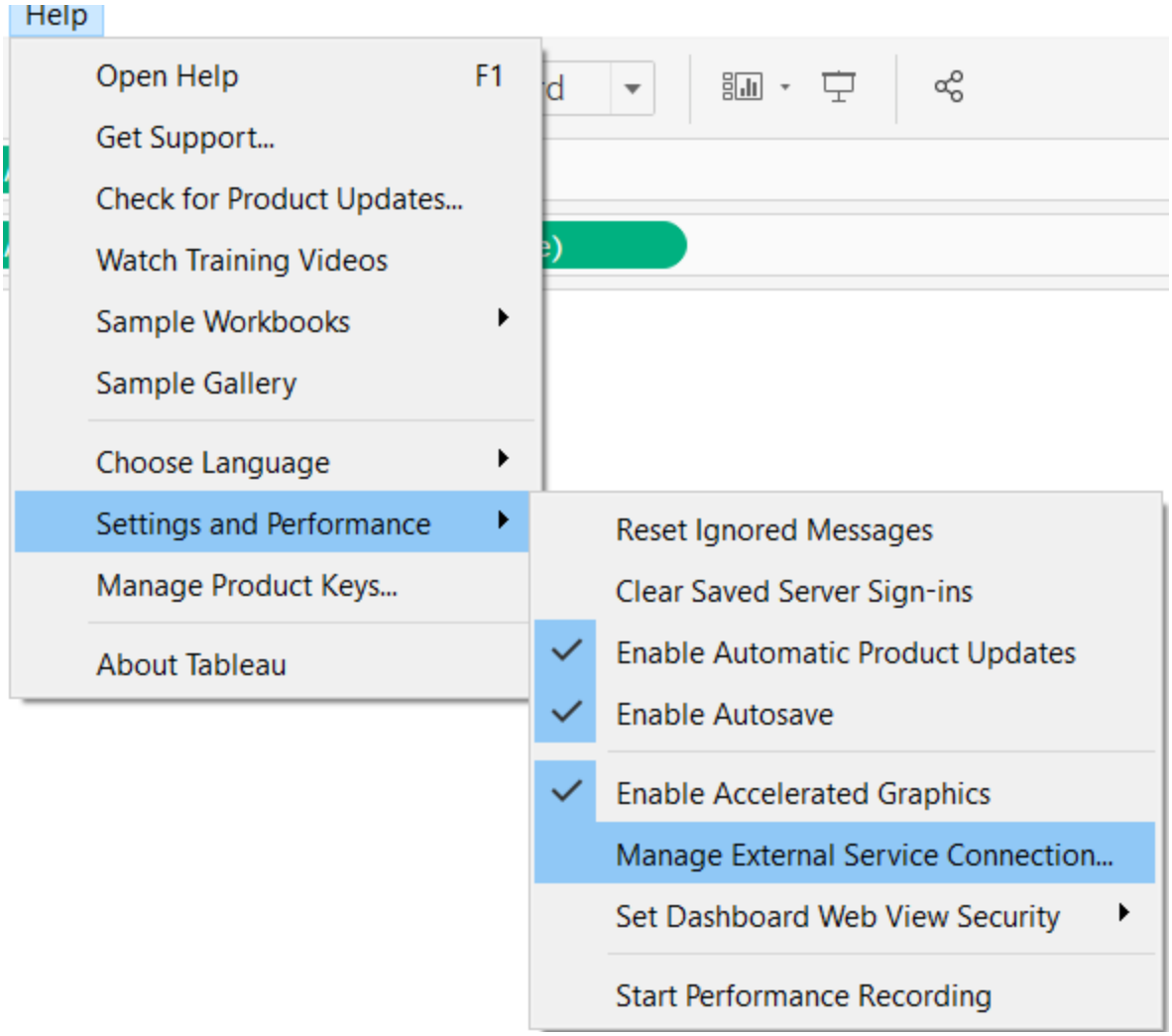
The Tableau worksheet that visualizes the shortest path is included in this package under "\ExternalServiceConnection\Examples\Travelling_Salesman_Problem".

The server interface set up above receives requests for MATLAB analytics from Tableau. The server formats the data received from Tableau as JSON, and makes a RESTful call via HTTP to MATLAB Production Server. The results from the MATLAB computations are sent back to the node.js server, which then returns it to Tableau.

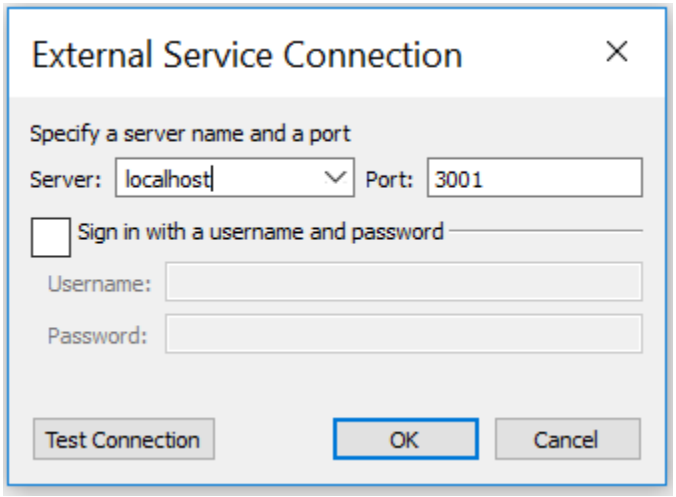
Tableau Environment Setup

To enable Tableau to make a service call to the interface, follow the below steps:

a) In the Help option, click on Settings and Performance, then 'Manage External Service Connection'



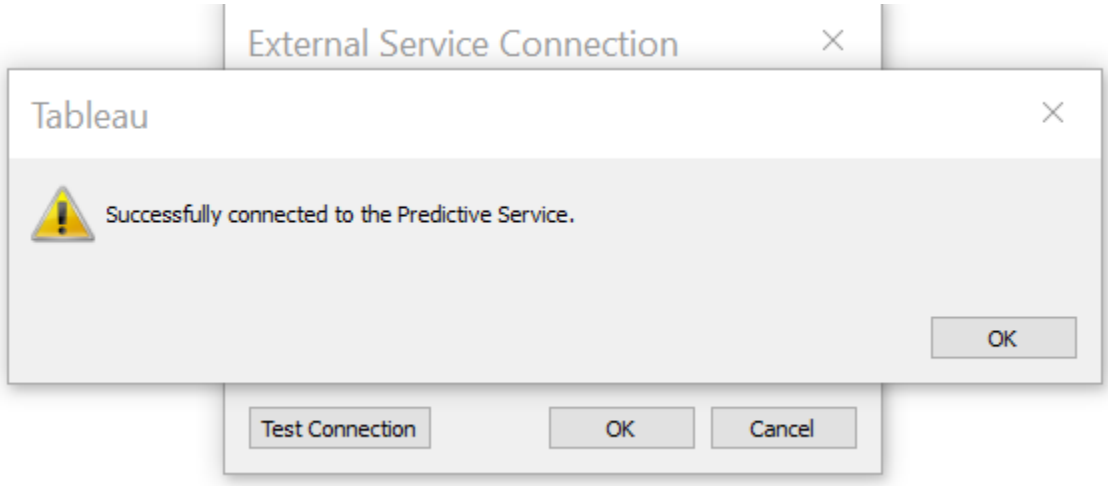
This will bring up the UI below:



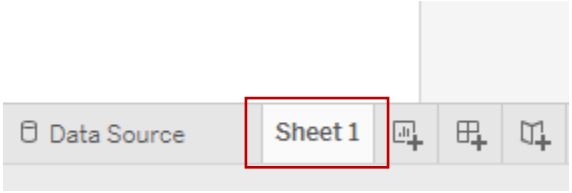
Enter the information for Server and port number for the interface in the dialog above. The server information is the hostname where the node.js application is running. The port number should match the port number configured in the server.config file in the config folder for the interface.

In the above figure, the interface is running locally, so the server is localhost. The port number configured in server.config file is 3001.

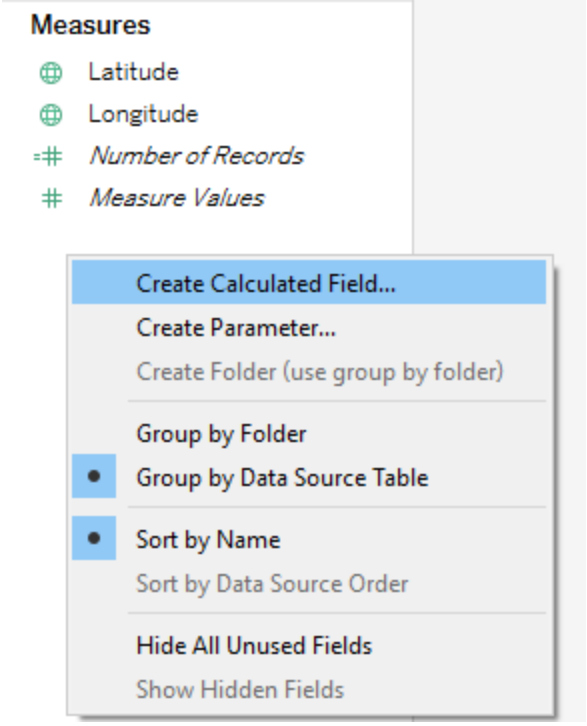
Click on Test Connection button to ensure successful connection to the interface.



b) Load the data file to work with in Tableau. This example uses the airports.csv file available under Travelling_Salesman folder. Click on "Sheet1" to view the Tableau sheet



c) Under the 'Measures' section in Tableau, right click and select 'Create Calculated Field'.



d) This will bring up the window to edit the calculated field. Enter the name of the calculated field in the textbox.

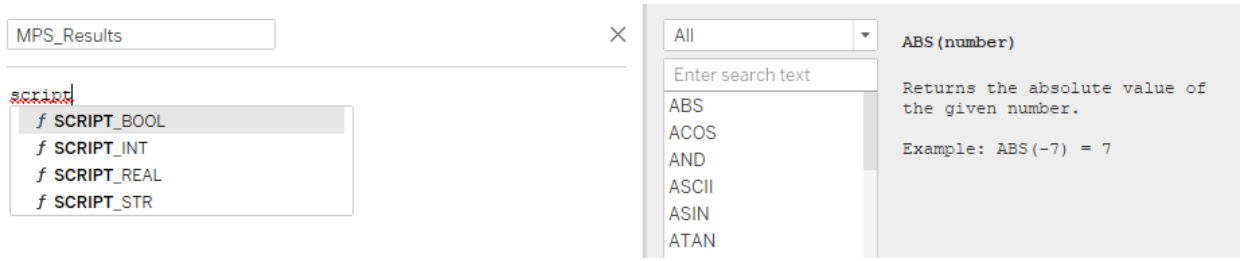
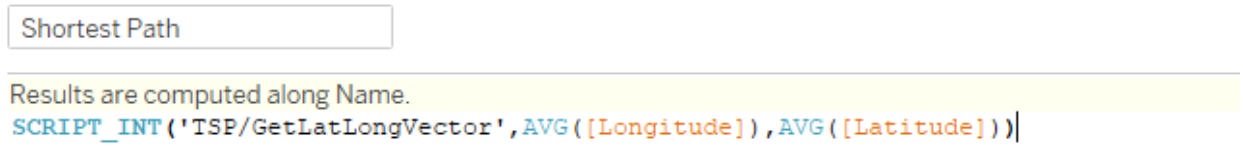


Tableau provides 4 SCRIPT_XXXX functions to make an external service call. Depending on the data type expected from MPS, use the appropriate function.

e) Specify the MATLAB package and function to call

The argument for the SCRIPT_XXXX function takes in the name of the MATLAB function, as well as the data to be passed to the function. Consider the example below:



Here, the name of the CTF archive is TSP and the name of the MATLAB function is GetLatLongVector. This is specified in the first argument as 'TSP/GetLatLongVector'

The data to be sent to MATLAB follows the first argument. In the above example, The Longitude and Latitude values are sent to MATLAB, and are specified in that order as arguments to the SCRIPT_XXXX function. Please see 'Limitations' in the Notes section at the end of this document for more information on data formats supported.

Click ok to accept the calculated field. This field can now be used in any of the plots or graphs in Tableau. Any time the field is included as part of a worksheet, Tableau sends a request to the MATLAB Production Server interface for Tableau and assigns the results of the call to the calculated field.

MATLAB Environment Setup

To ensure MATLAB Production Server handles the request from interface, follow the steps below:

a) Ensure that the MATLAB application is ready to accept input arguments from Tableau.
Input arguments from Tableau are structures with pre-determined names for each input argument. Ensure that the MATLAB function can access the data sent by Tableau by following either of the options described in Appendix B.

b) Deploy MATLAB package to MPS or use MATLAB Compiler SDK

Use the MATLAB Compiler SDK to package the MATLAB functions into deployable archives for deploying to MATLAB Production Server. For more information, click on the link below:

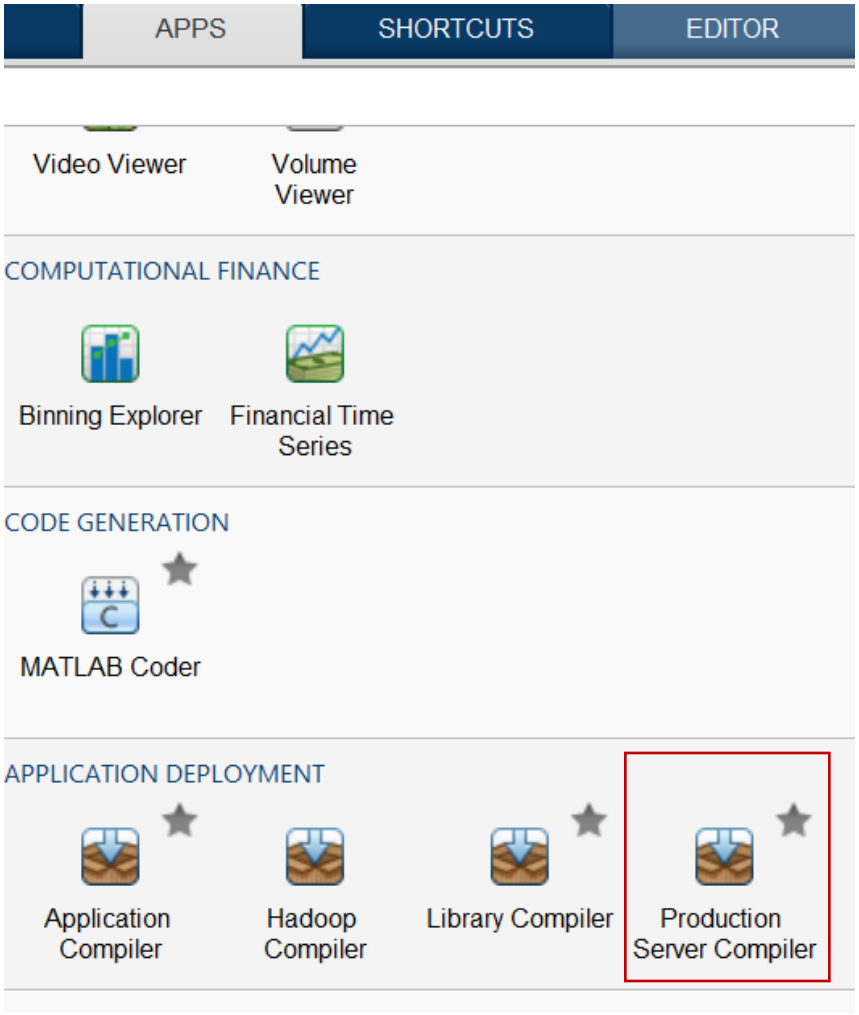
https://www.mathworks.com/help/mps/deployable-archive-creation.html

You can also use the testing environment available as part of the MATLAB to start up a test MPS server locally. More information is at:

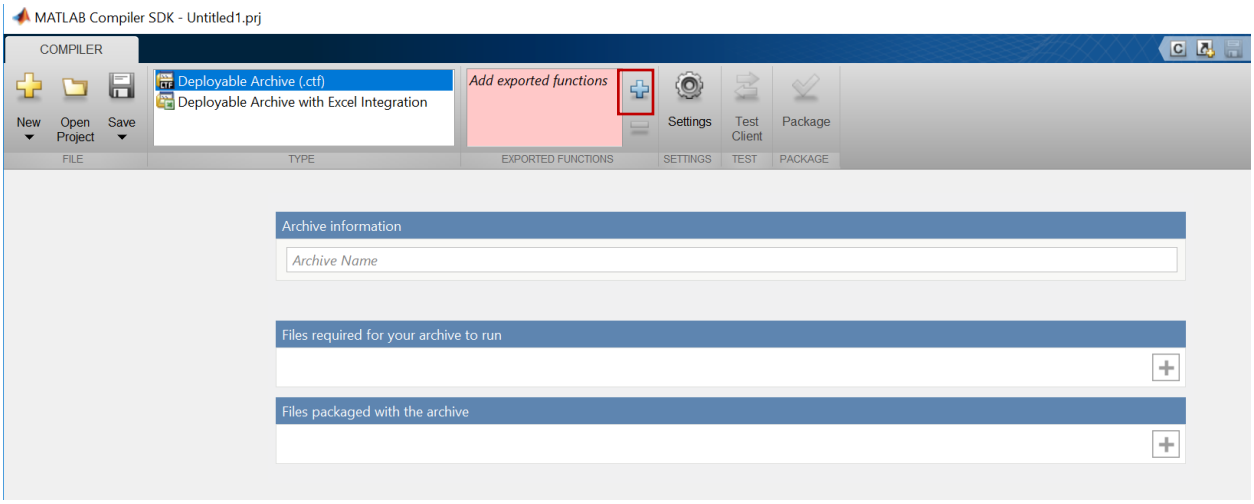
https://www.mathworks.com/help/compiler_sdk/mps_dev_test/test-in-process.html

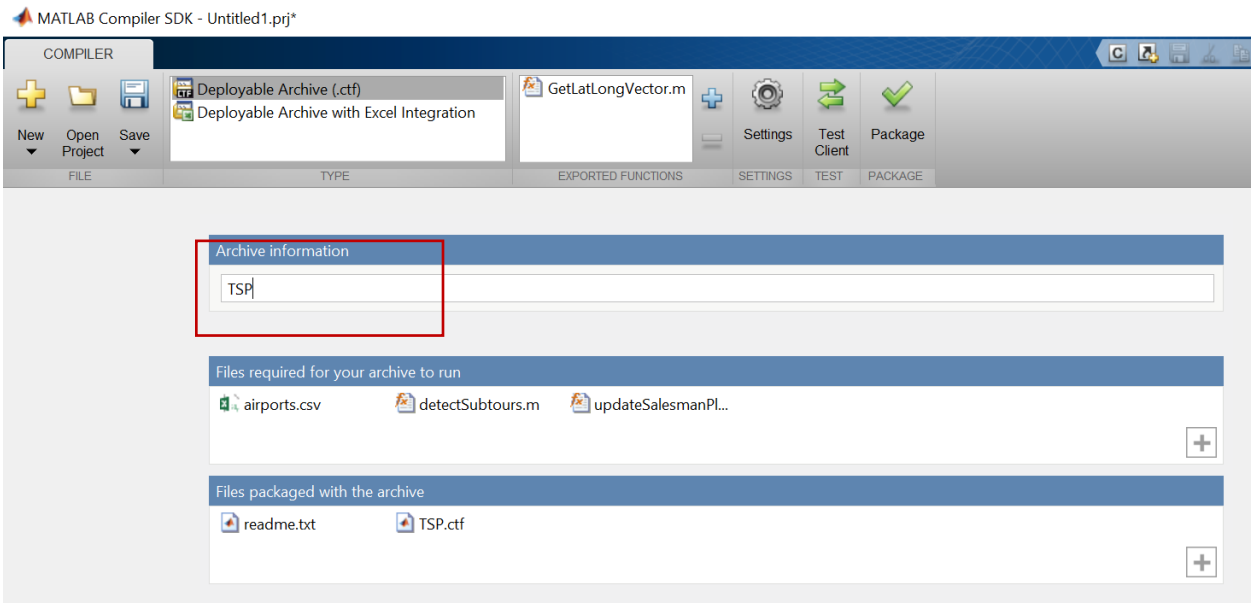
To quickly setup up a test MPS server using MATLAB Compiler SDK, follow the steps below:

(i) Click on Apps gallery and open Production Server Compiler App:



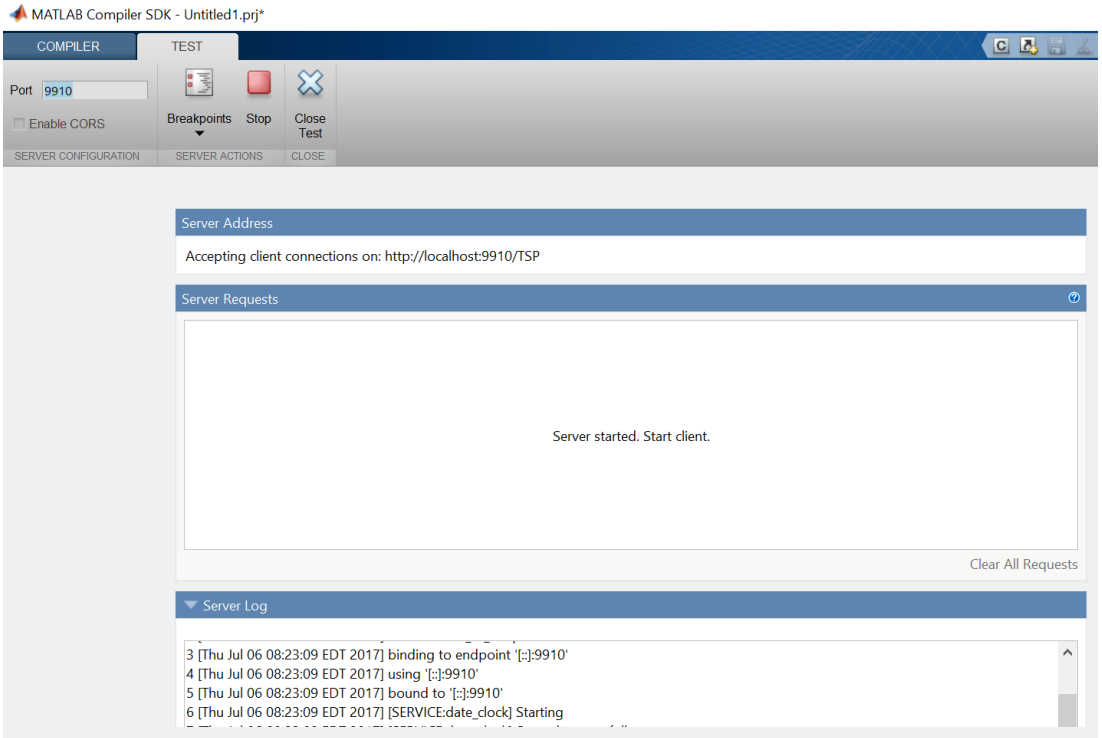
(ii) Click on the '+' button to bring up the file selection dialog and add the MATLAB code files to package:





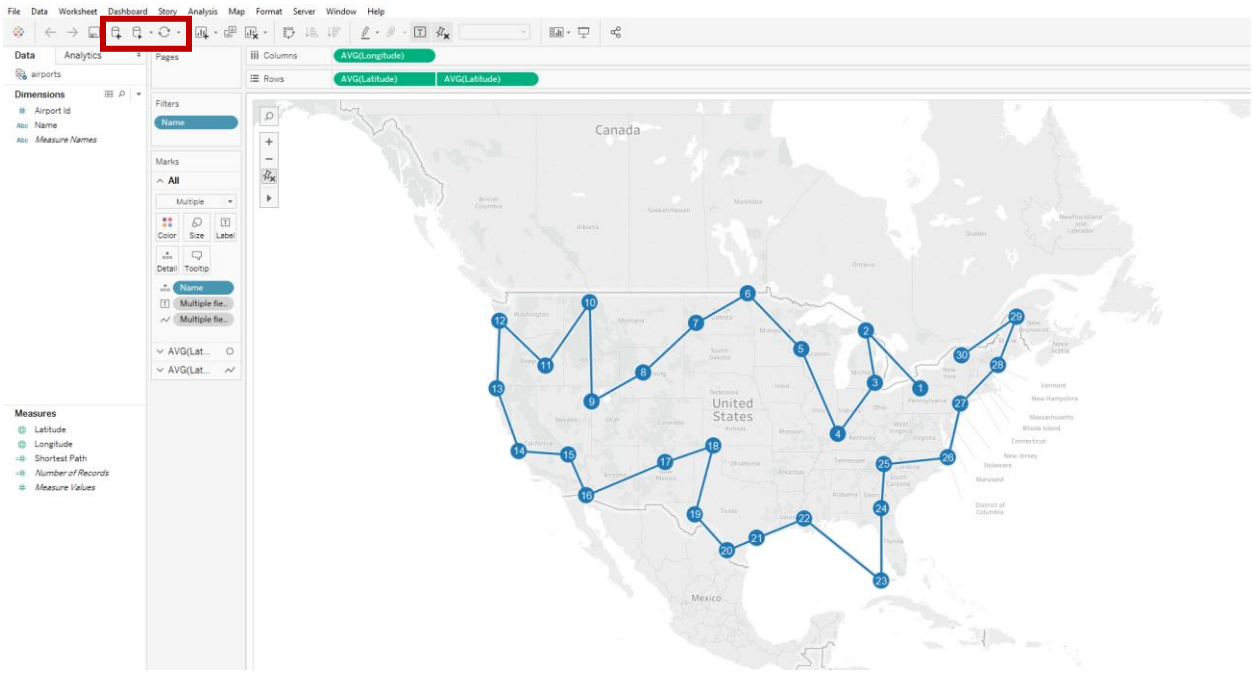
(iii) Once the MATLAB functions are selected, ensure that the Archive information matches your specifications as highlighted above. In this example, the name of the archive(CTF) is TSP, and the name of the function is the name of the MATLAB function added, i.e., GetLatLongVector.

(iv) Once the required files are added, click on the 'Test Client' button and then click start to start up a local instance of MATLAB Production Server listening at port 9910.

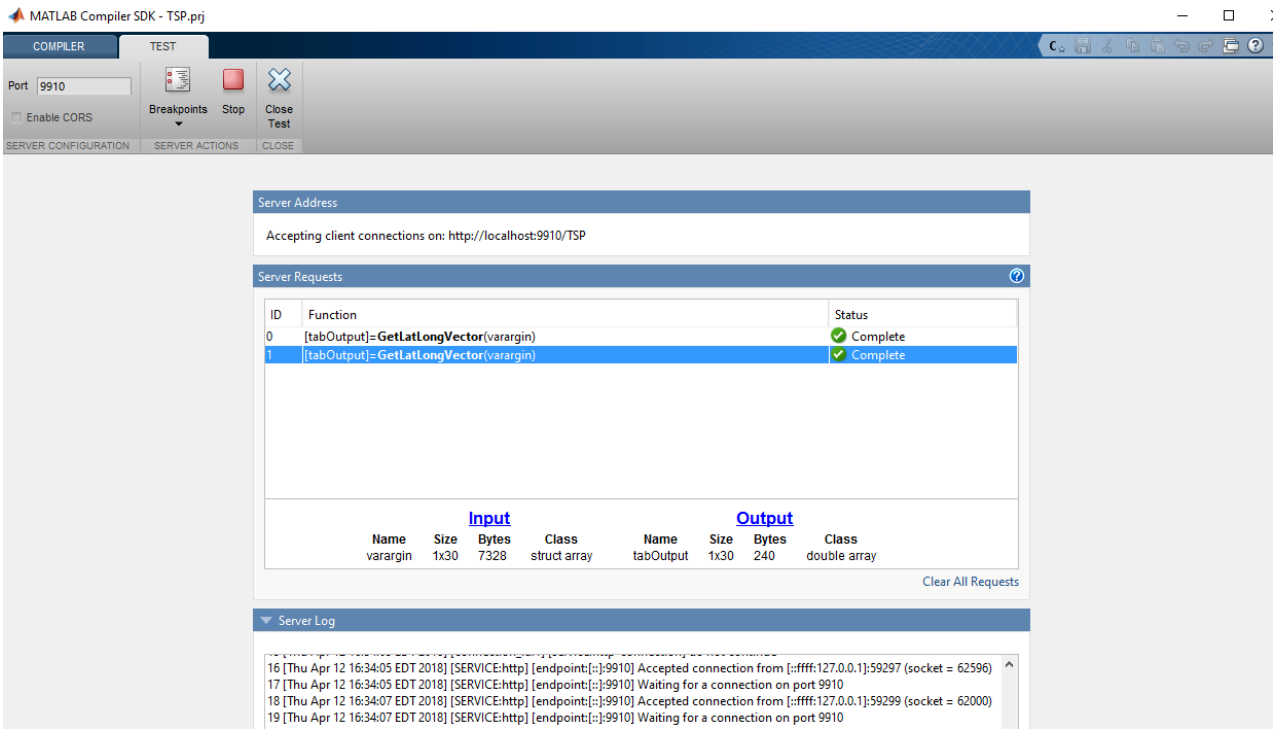


c) Once the packaged MATLAB function is hosted on MPS or on the testing environment, the calls from Tableau are routed to MATLAB by the interface via HTTP. Output from the MATLAB functions are provided back to the Tableau calculated field, where it can be visualized.

Once the MATLAB and Tableau environments are setup, test the interaction between the applications by clicking on the refresh button in Tableau. The refresh button is available in the toolbar in Tableau as highlighted below, and is disabled if Tableau is set for auto-update. Refreshing, or auto-updating within Tableau should send the data loaded in Tableau to MATLAB, and make the results from MATLAB available in the map.



The MATLAB Compiler SDK 'test Client' window should show the successful call as below:



Notes

Determining the correct option to use

Option 1: Web Data Connector

Use the Web Data Connector option when

(a) There is no data to be sent from Tableau to the MATLAB function

Tableau users may not always have access to the data to be loaded in Tableau. If the workflow requires a call to MATLAB to fetch the data, then the WDC option is the correct choice. This option does not require the Tableau user to send any data when making a function call to MATLAB.

(b) There is a need to collect information from several data sources

Since the JavaScript code in the WDC controls the mechanisms for data collection from different data sources, there is more control over the data provided to Tableau. The WDC can retrieve data not just from MATLAB, but from other APIs, services etc.

Limitations:

The WDC mechanism requires that the data to be sent back to Tableau be formatted and controlled using JavaScript code. Although this provides more control to the WDC developer, the disadvantage is that the JavaScript code needs to be edited and deployed to the web server if the data schema changes.

A second limitation is that debugging of the WDC code during development can be time-consuming, particularly if the data structure is complex. One possible option to test the WDC is using the simulator environment provided by Tableau: http://tableau.github.io/webdataconnector/docs/wdc_debugging

Option 2: External Service Connection

Use the External Service connection option when

(a) There is requirement for ad-hoc querying and visualizing in Tableau

During development phase, there may be a need to quickly query MATLAB and retrieve data sets. Implementing a WDC requires upfront investment in time and effort to develop the JavaScript code, so using the external services connection may be ideal in these situations.

(b) Tableau users expect an interface comparable to open source integration capabilities available within Tableau

The External service connection and SCRIPT_XXXX functions are used to access open source analytics in Tableau. Users accustomed to the syntax and structure of this call will find the environment and usage similar for integrating with MATLAB.

Limitations

One of the limitations of using the external service connection is that it requires that the number of rows in the result data set match the number of rows in the input data sent from Tableau. For example, If Tableau sends 30 rows to MATLAB, then it expects 30 rows in the result data set. If only a single value is returned, Tableau will automatically replicate it 30 times. If MATLAB only sends back 25 values, then the MATLAB function will need to be modified to create a blank array, fill 25 rows with data, and the rest as NULL. If there is a mismatch in data size, Tableau displays an error message. This is a feature of External Service Connection provided by Tableau, and applies to all calls to any external service.

A second limitation is that the four functions provided by Tableau (SCRIPT_STRING, SCRIPT_INT, SCRIPT_BOOL, SCRIPT_REAL) all expect input parameters to be specified in the function call. Therefore, it is necessary to have data loaded into Tableau, or type in 'dummy' data to use with the function call. It is not possible to call an external service function without sending input parameters.

Additional documentation on errors that can occur when communication with an external service is available here:

https://onlinehelp.tableau.com/current/pro/desktop/en-us/help.htm#r_connection_troubleshoot.html#any

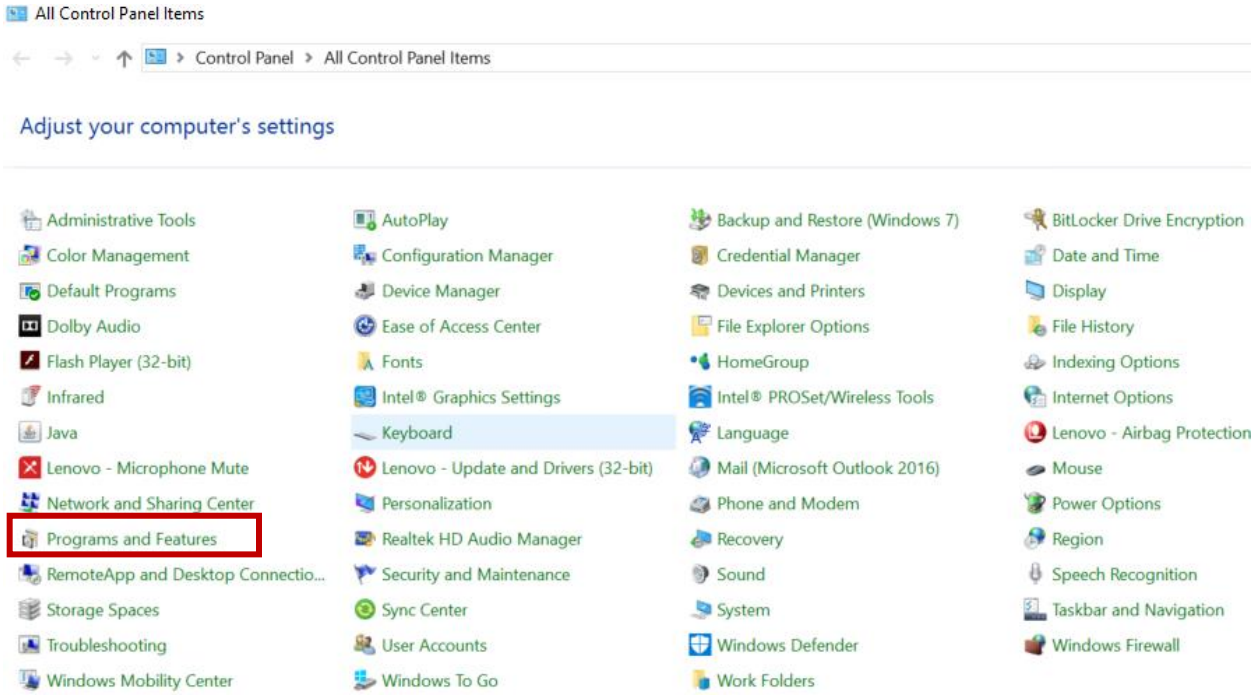
Contact Information

Please contact us at mwlab@mathworks.com for questions and feedback.

Appendix A: Set up Internet Information Services

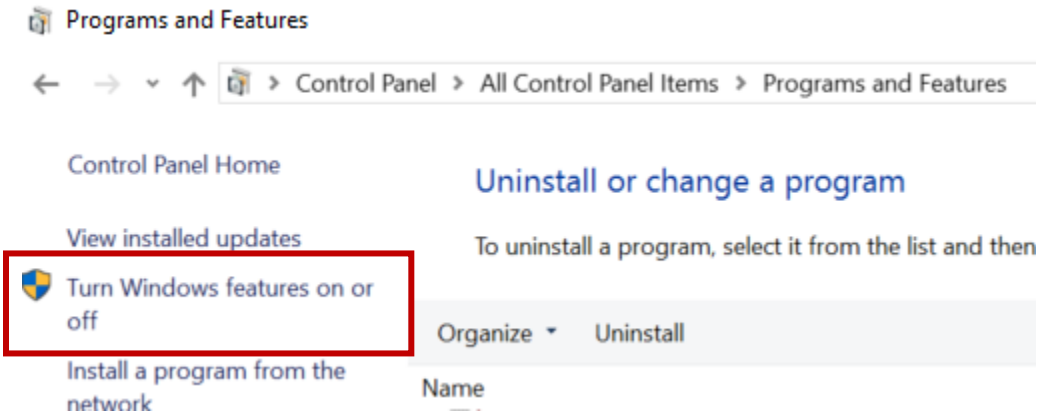
These instructions for setting up the IIS server are not intended for production use and are intended only for use in evaluating the interface.

A) In Windows search bar, type control panel and access the Control Panel App. This displays the UI with All Items:

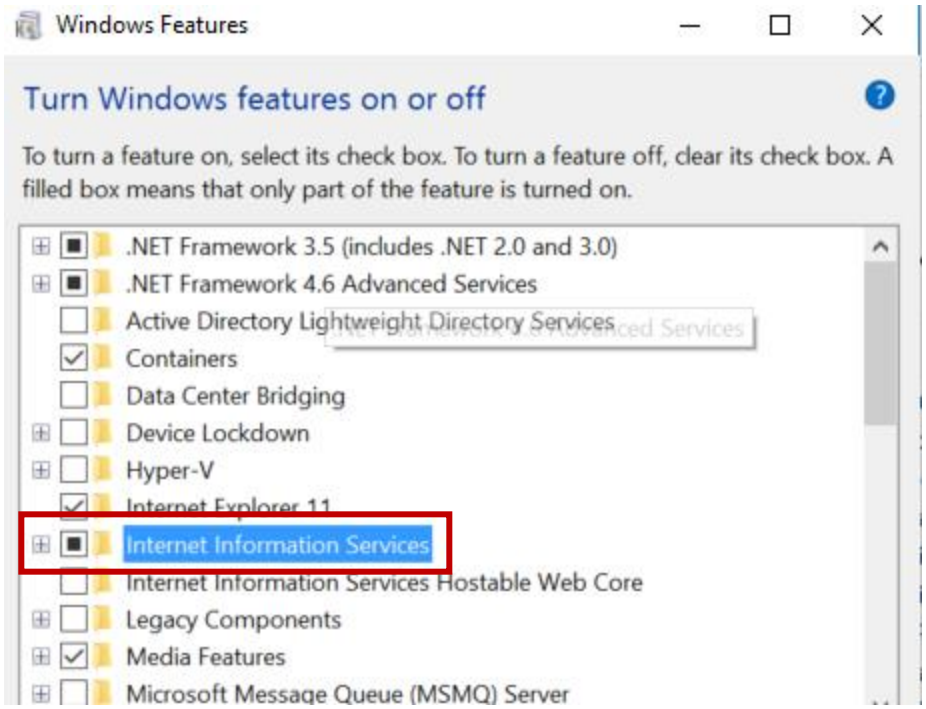


B) Click **Programs and Features** highlighted above

C) On the Uninstall or change a program page, click on 'Turn Windows features on or off' as shown below.



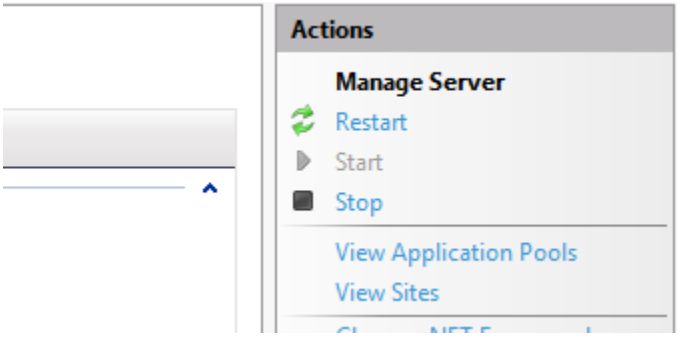
D) In the dialog box that pops up, select the **Internet Information Services (IIS)** check box, then click **OK**.

E) 

Once IIS is enabled, copy the MPS_WDC_Sunspot.html page to the 'C:\inetpub\wwwroot' folder.

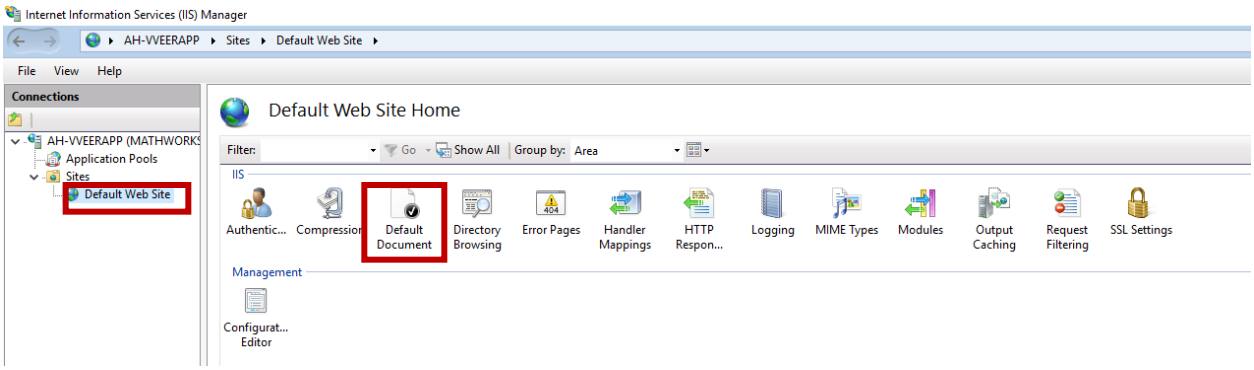
The default port number for the HTTP call to locally hosted HTML pages is 80. If the port needs to be changed, this can be configured in Internet Information Services Manager. To do this right-click IIS->Default Web site and choose option to edit bindings. This example uses the default port number.

To ensure that the web page is hosted on IIS, bring up the IIS manager by typing 'inetmgr' in the search bar in Windows. Click on the top-level name in the left panel and ensure that the server is started by clicking on 'Start' under Actions->Manage Server in the right pane:

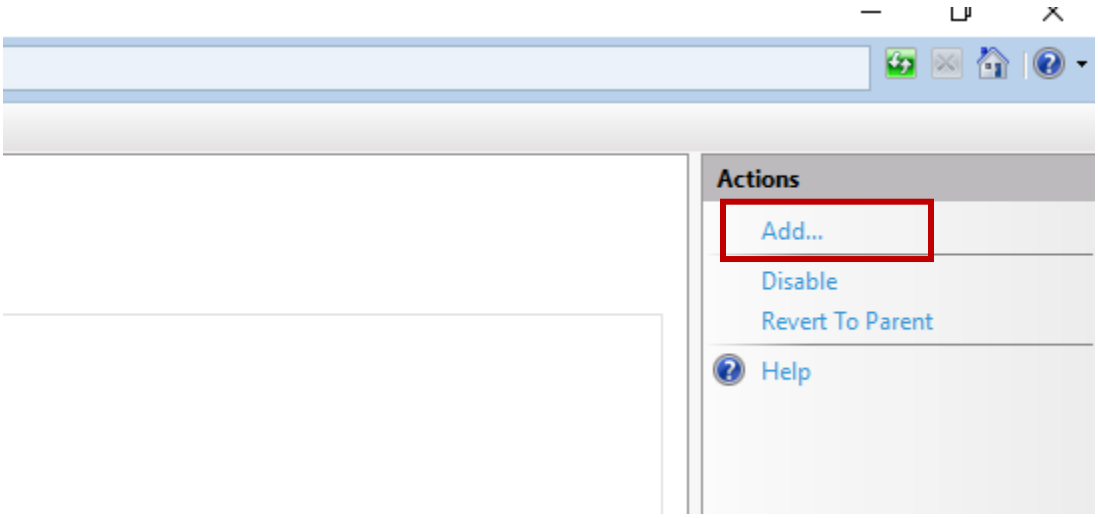


Note that if the server is already started, the start button will be disabled.

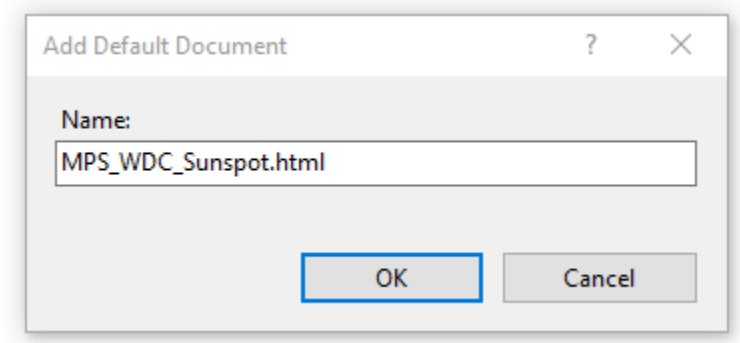
In the UI for the IIS manager, locate and expand 'Sites' on the left-hand pane as below:



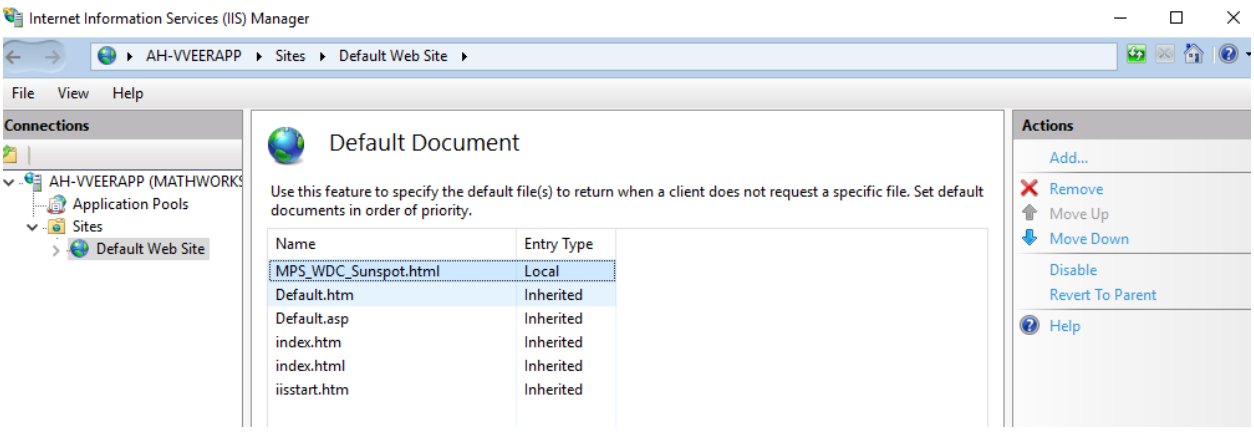
Double click on 'Default Web Site' highlighted above. In the right panel, click on 'Add'.



Type in MPS_WDC_Sunspot.html in the pop-up window and click **ok**:



The HTML page used for this demo will now appear in list of default documents. Ensure that this document appears first on the list by selecting the file name and clicking on 'Move Up' or 'Move Down' on the right pane:



The web page hosted on the web server is designed to function as a web data connector within a Tableau environment, and not designed to work independently in a browser as a web page. However, to test if the web page is hosted correctly, open a browser and type in http://localhost:80/MPS_WDC_Sunspot.html in the address bar. If the browser displays the below UI, then the web data connector is available for use within a Tableau environment.

Appendix B: Handling input arguments from Tableau in MATLAB

When using the external service connection feature in Tableau, the MATLAB developer needs to be aware of the data structure of the input arguments sent by Tableau. Tableau automatically assigns names that start with an underscore ('_') to the input arguments. As an example, if there are 3 input arguments to MATLAB, the first argument is named '_arg1', the second as '_arg2', and the third as '_arg3'.

This introduces an issue in MATLAB because the MATLAB structure will now have fieldnames as '_arg1', '_arg2' etc. MATLAB does not support accessing the data in the structure when the fieldname starts with an underscore ('_').

To access the data in the structure of the input argument, you can either

1. remove the underscore ('_') in the field names or
2. convert the structure to a cell array or table

Option 1: Remove the underscore ('_')

To replace the '_' in the fieldnames of the structure, create a new MATLAB function called **renameStructFields.m**. Copy and paste the below code to the file:

```
function newstruct = renameStructFields(str)
allNames = fieldnames(str);
newFieldNames = strrep(allNames, '_', '');
nstructs = numel(str);
nfields = numel(newFieldNames);
emptycell = cell(1, nfields);
for ii = 1:nfields
   emptycell{ii} = {str.(allNames{ii})};
end
cslist = [newFieldNames(:)'; emptycell];
newstruct = struct(cslist{:});
```

In your main MATLAB function, call the renameStructFields function and pass in the input arguments from Tableau. This will return a new structure with the '_' removed.
You can now access the data in the structure as you normally would.

Option 2: Convert the structure to a cell or table

If you want to pass in multiple input arguments to MATLAB, you can configure the MATLAB function to accept any number of input arguments from Tableau by using 'varargin'. Please see below link for more information:
https://www.mathworks.com/help/matlab/ref/varargin.html

An example of the change to be made is given below:

If the MATLAB function signature is

```
function output = MPSFunction(A,B,C)
```

then modify the function signature to be

```
Function output = MPSFunction(varargin)
```

You can then index into the parameters with:

```
inputParams = struct2cell(varargin{1});
```

Here inputParams is a cell. The parameters A, B and C can be accessed using standard MATLAB operations. For example,

```
A = inputParams{1}; B = inputParams{2} ; C = inputParams{3}
```