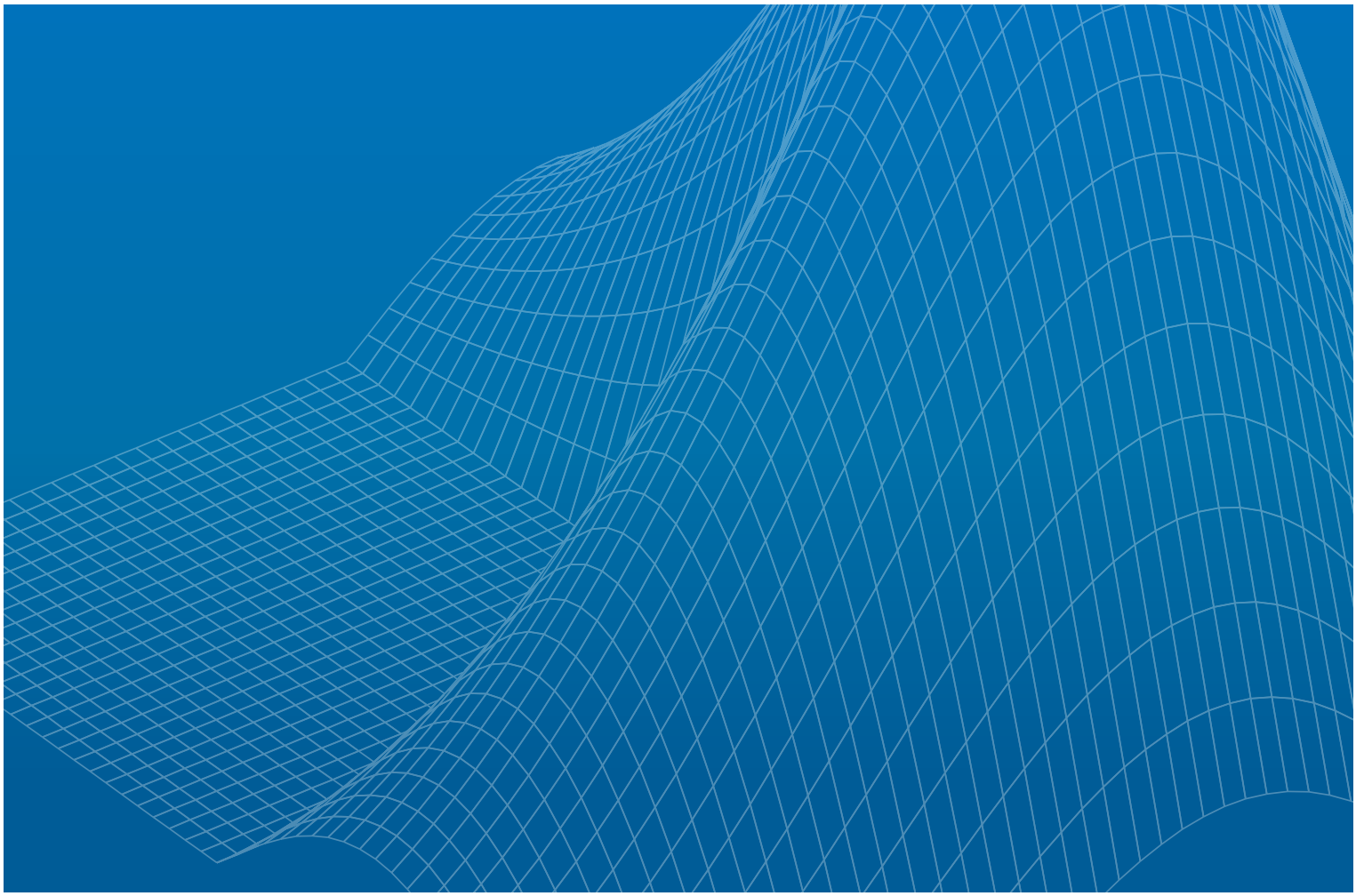


Building a Web Site with MATLAB Analytics

David Willingham and Chetan Rawal, MathWorks

WHITE PAPER

MathWorks®

Table of Contents

Introduction

A common question from engineers, quants, scientists, and data scientists is, "How can I develop analysis and analytics that will embed easily in a web site?" A typical follow up question then is, "How fast can I go from developing my algorithm to deploying it for multi-user access?"

To answer these questions, we recommend following a two-step approach:

Step 1. Selecting an analytics package

Select an analytics package that is user friendly, well documented, and tested so that you can focus on developing algorithms and so you have confidence in the results. Beyond desktop computations, the package must lend itself to integration with IT production systems and be able to provide:

- Distributed computing on a compute cluster with multiple cores
- Big data analysis from data on different databases, or a Hadoop Distributed File System (HDFS)
- Simultaneous processing of concurrent requests from multiple users and applications

MATLAB® fulfills all of these scalability demands. It lets you quickly develop and test code; speed up execution time with parallel and GPU computing options; utilize big data, including data residing on HDFS; and deploy to web environments via Java, .NET, and Python.

Step 2. Enlisting a web development expert

Many subject matter experts in math, engineering, and the sciences require not only an analytics package, but also the option to share these analytics with peers and clients, usually via a web site. While versed in mathematical analysis, they may feel taxed if trying to create a web site on their own, unless they already have a web development background.

In such scenarios, a web development expert can help integrate analytics into a web site. One option is to request for MathWorks consulting services, as they routinely help organizations deploy their MATLAB algorithms to web services.

The two steps above highlight the importance of two distinct roles required in deploying analytics to a web site: a **MATLAB developer** and a **web developer**. Could both roles be undertaken by the same person? Yes, but the process can become very time consuming. Keep in mind that it entails:

- Learning how to build a web site, which usually comes at the expense of developing analytics functions.
- Maintaining the web site once it is launched. This can become time consuming as users report bugs and request enhancements, especially as the web site gains popularity.
- Coordinating with different business units, especially in a larger organization. To successfully host your web site, a system architect or IT manager may impose additional architectural or security requirements.

Nonetheless, if you have basic web programming skills, this paper provides an overview of the topic and development roles. It outlines several examples that guide you through the process of creating simple web sites.

Designing a Web Site with MATLAB Analytics

Let us first describe the primary responsibilities of the two roles described in the previous section.

The MATLAB developer builds, tests, compiles, and deploys functions to *MATLAB Production Server*™, a centralized compute server that lets you run MATLAB programs as part of a web application.

The web developer creates the web application and establishes the connection between the application and the functions being run from MATLAB Production Server.

The overall workflow delineating the two roles is depicted in Figure 1.

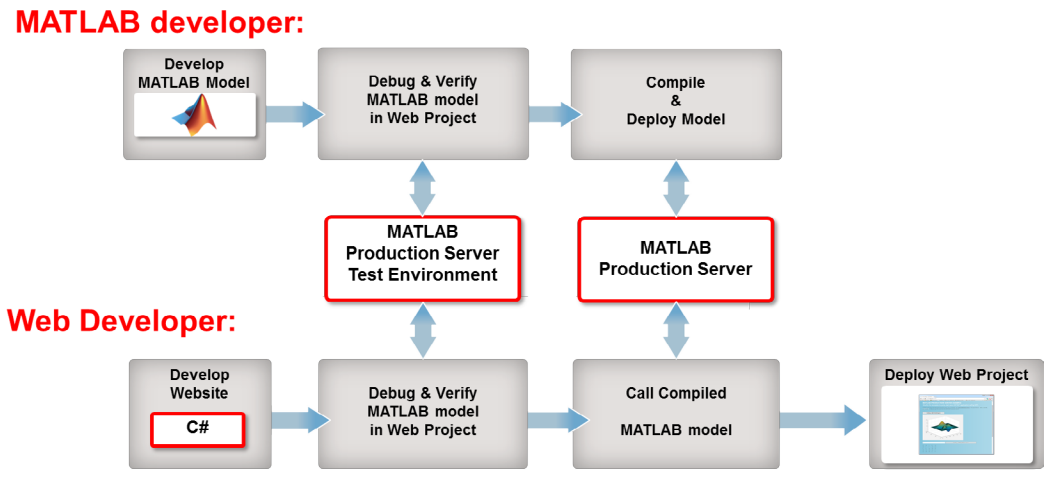


Figure 1. Workflow for deploying MATLAB functions to MATLAB Production Server for web sites and other business applications.

A Detailed Look at the MATLAB Developer's Role

Before starting with examples, here is an overview of the steps that the MATLAB developer needs to follow to deploy functions to the web for most applications:

1. Document inputs and outputs from their MATLAB functions to provide a reference guide for the web developer. The web developer will need this documentation to create an interface.
2. Test and debug the MATLAB function with the web developer in a live MATLAB session. This is done using the client Integration tool provided as part of MATLAB. Note: this requires the web developer to have created a web project (as per web developer role steps 1-4 below).
3. Once the test version has been verified, package the function into a deployable archive for use on MATLAB Production Server.

4. Copy the deployed function to MATLAB Production Server framework.

The products used in these steps include MATLAB and any toolboxes needed for the function, along with MATLAB Compiler™, MATLAB Compiler SDK™ (for deployment), and MATLAB Production Server™.

A Detailed Look at the Web Developer's Role

The web developer follows these steps to call deployed MATLAB analytics from the web site. For this paper, we will show how the web site can be developed using the C# language and Microsoft .NET Framework. You can also use Python and Java, following a similar workflow. The web developer must:

1. Create a new web project in Visual Studio.
2. Import the MATLAB Production Server library as a reference, and define it in the web application.
3. As per the documentation provided by the MATLAB developer, define the function inputs and outputs as an interface.
4. Test and debug the MATLAB function with a live MATLAB session. This requires the MATLAB developer to start a test session using the client Integration tool provided as part of MATLAB Compiler SDK. To call upon the MATLAB function:
    a. Reference the MATLAB Production Server library
    b. Initiate an HTTP object
    c. Call the MATLAB function via the Server IP, port number, and function name
5. If the test version is successful, integrate the output of the compiled function as necessary in the web site. Some examples of what this output could look like are:
    a. A matrix outputted as a table
    b. A figure outputted as a static image
    c. Data outputted and integrated with a third-party interactive charting library
6. Publish the web application to a web server. If using ASP.NET web forms, the web server is Internet Information Services (IIS) from Microsoft. See ***Microsoft's documentation*** for details on publishing ASP.NET web forms to IIS.

Examples

The following examples illustrate the workflow that a MATLAB developer can use to create a simple web site. The first two are introductory examples, followed by a more detailed application example on mining economics.

Example 1: Simple Web Site with Matrix and Figure Output

Example 2: Simple Web Site with Interactive Charts

Example 3: An Application Case Study in Mining Economics

Prerequisite: Installing MATLAB Production Server

All examples require MATLAB Production Server, which is typically installed on the same machine as the web server. Review detailed ***installation instructions.***

After installation, create and start a server Instance, called Server1 in the figure below, with the root folder c:\AE\MPS\Server1 (Figure 2).

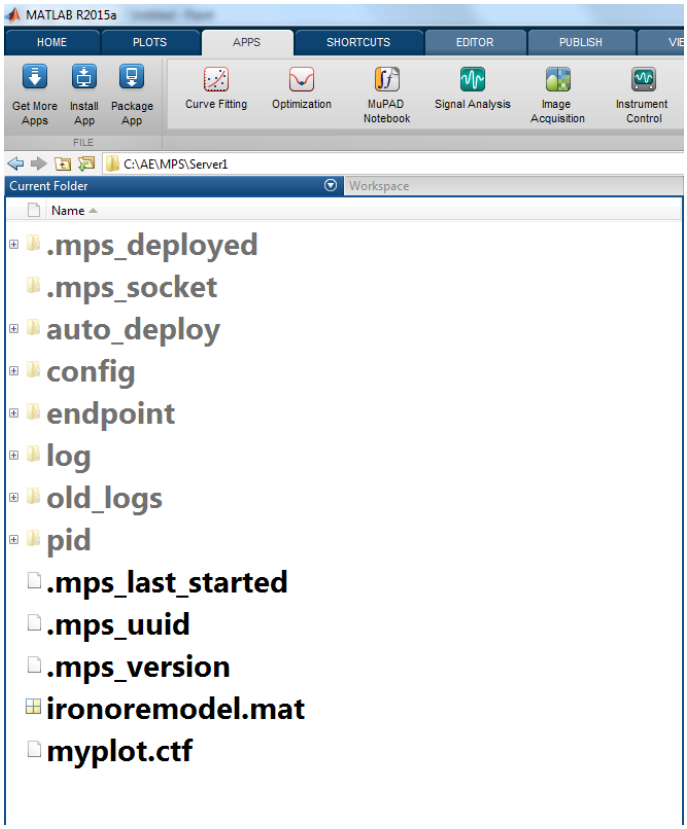


Figure 2. A directory listing of the MATLAB Production Server instance Server1.

Example 1: Simple Web Site with Matrix and Figure Output

This example shows how to output a simple matrix function, `mymagic.m`, and a static image of a simple plotting function, `myfigure.m`. The web site for this example is built in Microsoft Visual Studio 2012 using ASP.NET Web forms in a C# project.

Here is a snapshot of the final product:

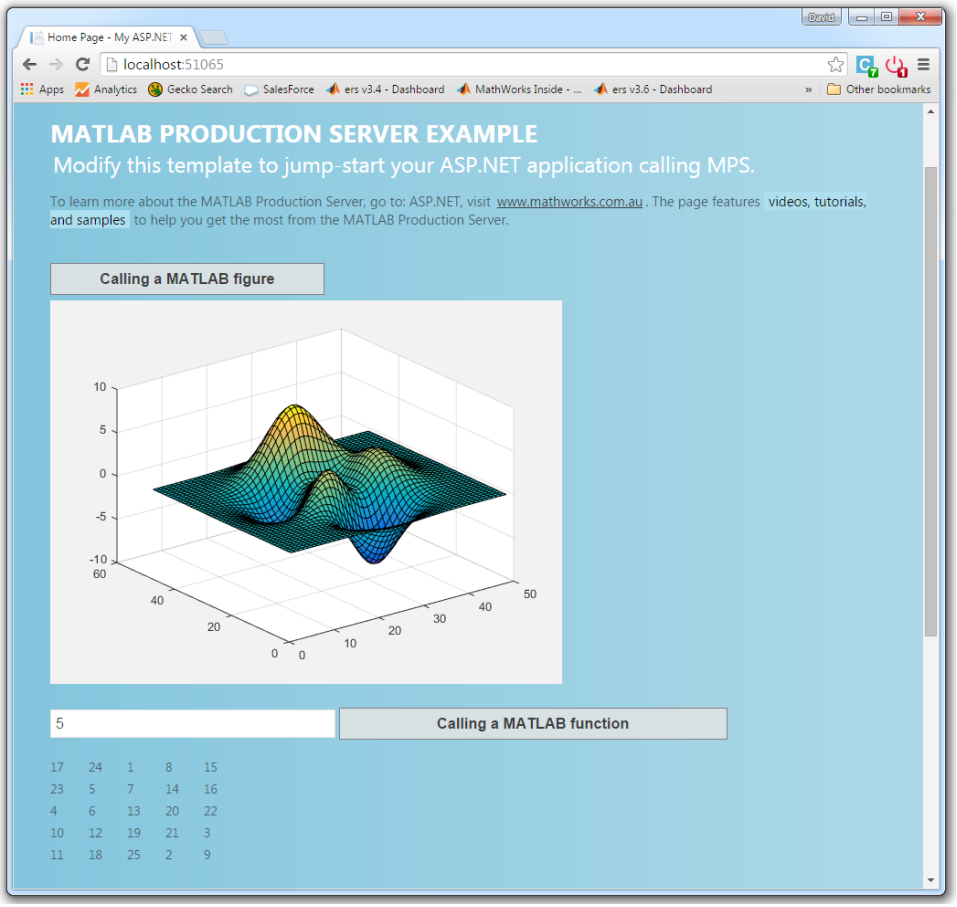


Figure 3. Snapshot of simple web site calling MATLAB Production Server.

The MATLAB Developer Workflow

1. Develop the function to deploy to a web site. Use the following two functions:

   `mymagic.m`:

```
function y = mymagic(x)
% input x - single integer
```

```
% output y - double matrix
y = magic(x);

myfigure.m:
function Surf _ Image = myfigure
%output figure bytestream

figure('visible','off')
surf(peaks)

Surf _ Image = figToImStream('outputType', 'uint8');
```

Note the extra line of code in `myfigure.m` using `figToImStream`. This code does not save the figure directly to an image, but converts it to a byte stream for the web developer.

2. Document the typical inputs and outputs of the function, which are used a reference guide for the web developer. This can be seen in the code above, `mymagic.m` has an integer input and a matrix output. In `myfigure.m` the output is a vector of type byte (or bytestream).
3. Test and then compile the function with MATLAB Compiler SDK to deploy to MATLAB Production Server. Here is the workflow:

    A.On the **Apps** tab, launch the **Production Server Compiler** app (Figure 4).

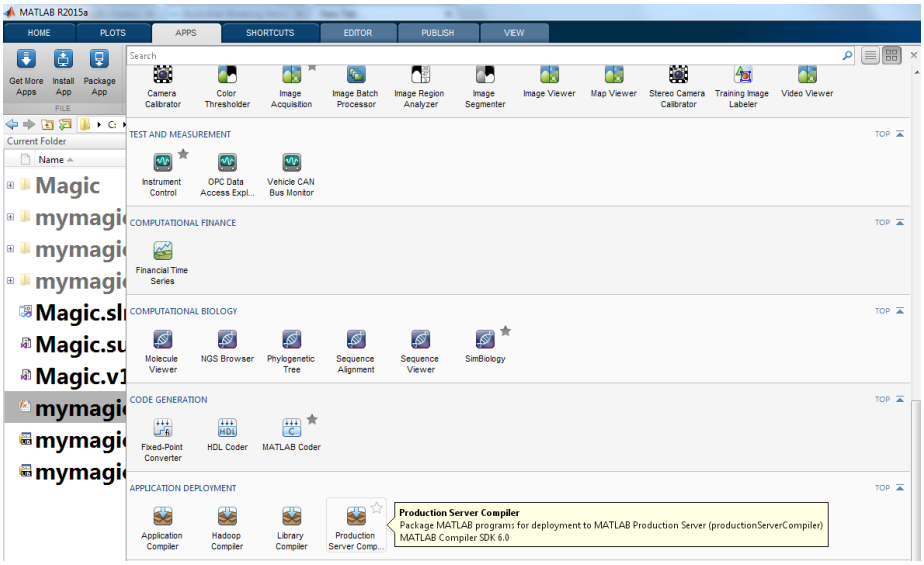


Figure 4. Launching the Production Server Compiler app.

B. Click **Client Integration** to start the test environment. Skip to the "Web Developer Workflow – Using .NET" section for steps on how to integrate the test environment with a web project (Figures 5 and 6).

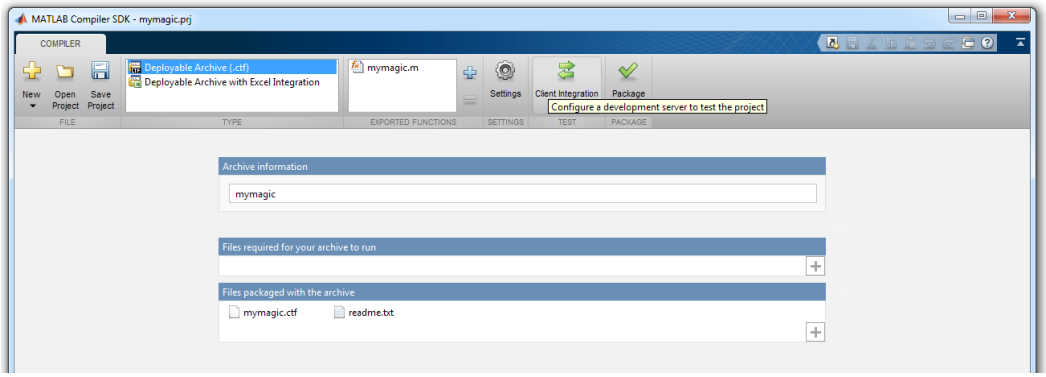


Figure 5. Launching the Client Integration tool to initiate a test (staging) environment of the MATLAB function.

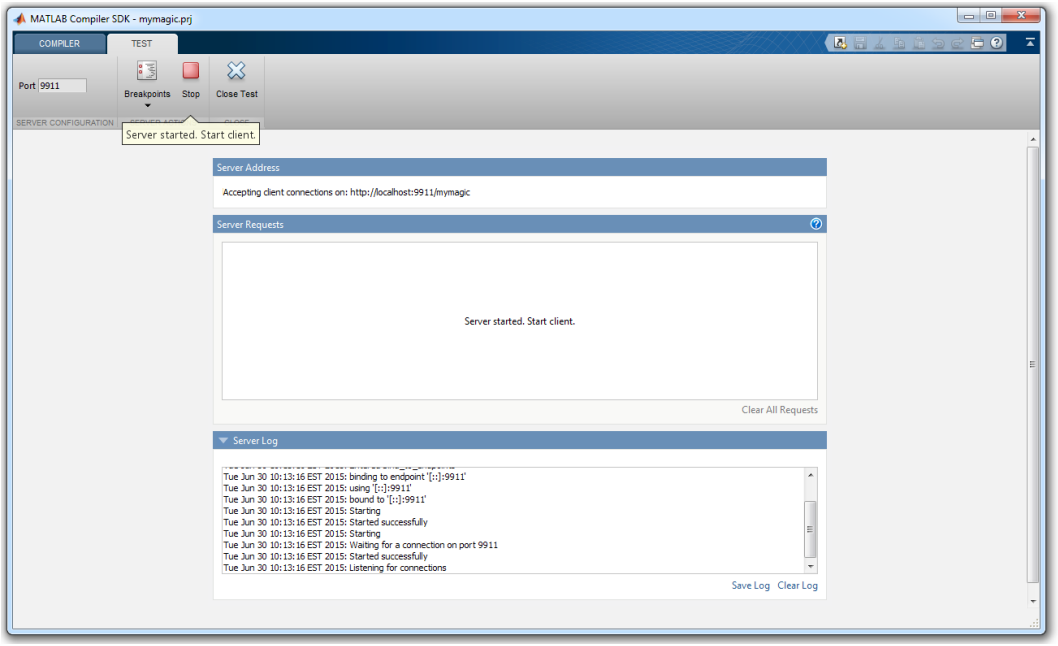


Figure 6. Starting a test (staging) environment of the MATLAB function.

C. Once the test version is verified, click **Package** to compile the function (Figure 7).

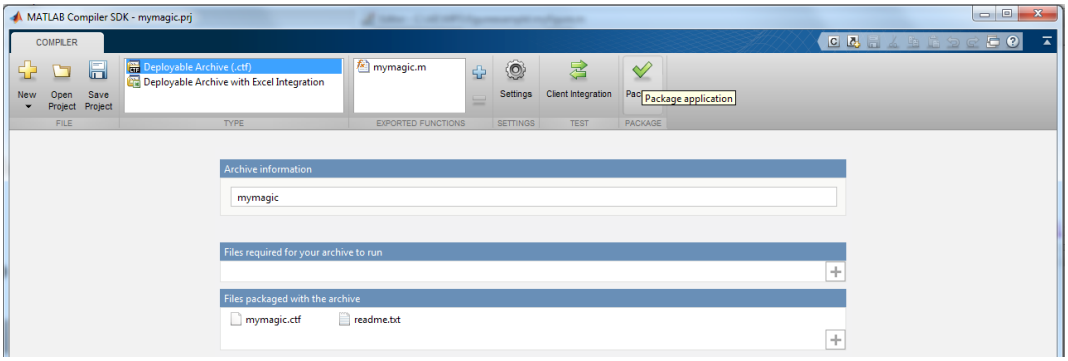


Figure 7. Packaging (compiling) the MATLAB function for the MATLAB Production Server.

4. Copy the deployed function to the MATLAB Production Server framework (Figure 8).
5. Copy the deployable archive `mymagic.ctf`.

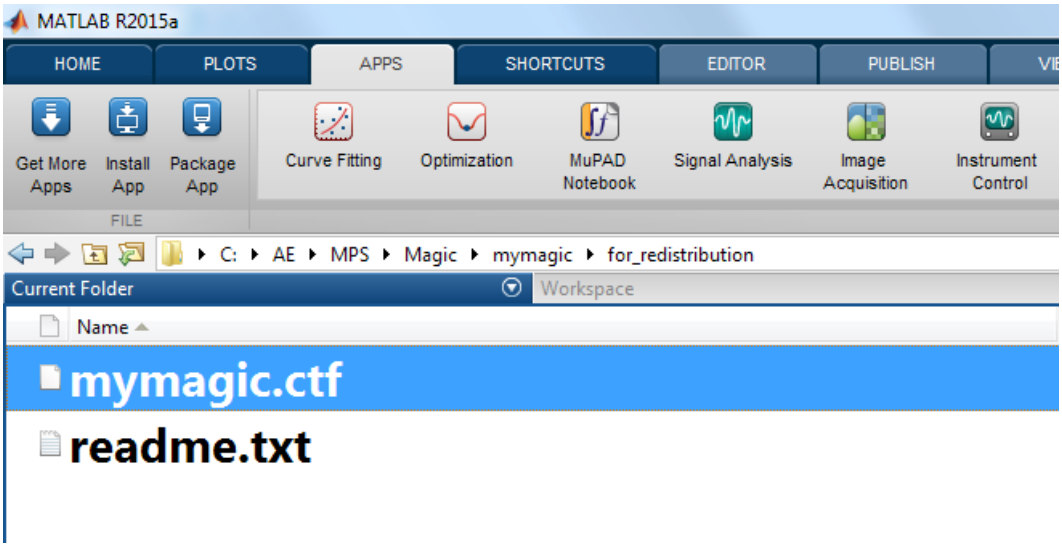


Figure 8. The compiled function ready to be copied is available under the for_redistribution folder of the project.

6. Paste `mymagic.ctf` into the MATLAB Production Server folder named `auto _ deploy` (Figure 9).

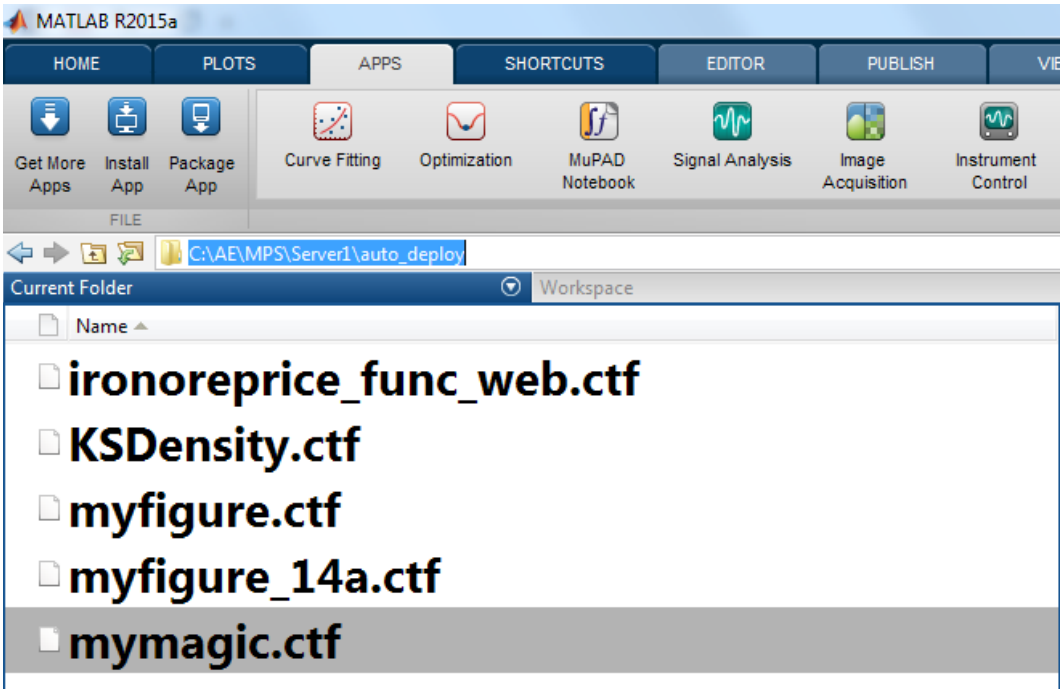


Figure 9. All deployed MATLAB functions are placed in the MATLAB Production Server "auto_deploy" folder.

Products used include: MATLAB, MATLAB Compiler, (for the deployment of the function) MATLAB Compiler SDK, and MATLAB Production Server.

The Web Developer Workflow: Using .NET

1. Create a new web project in Visual Studio (Figure 11).

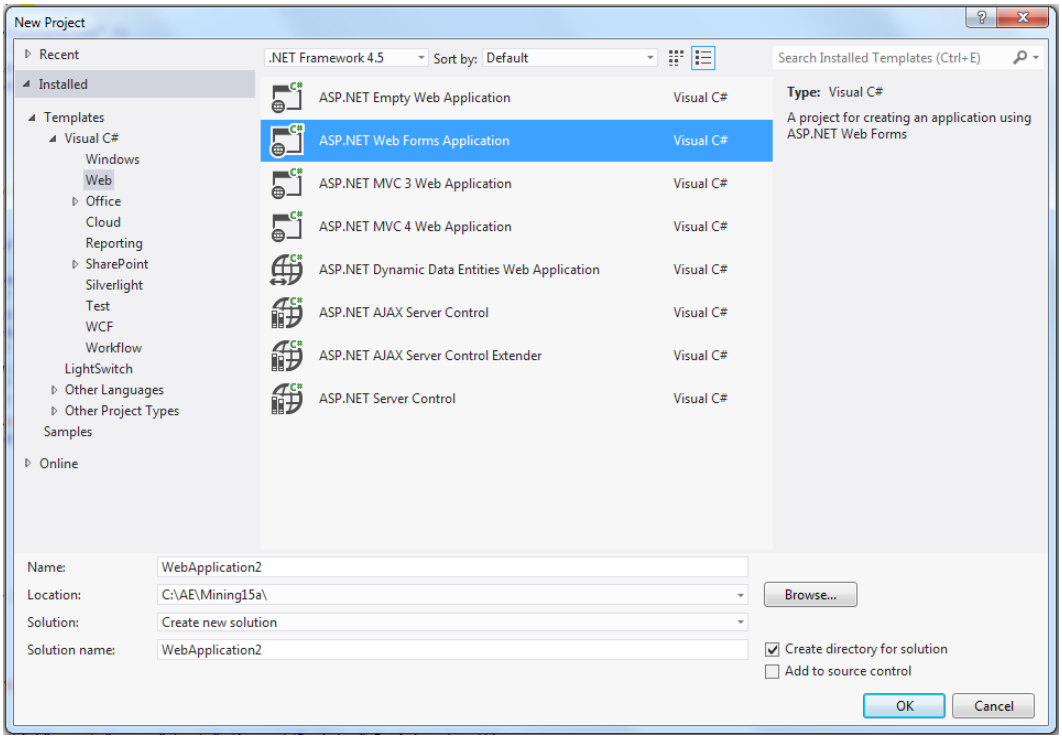


Figure 11. Creating a new web project in .NET.

2. Import the MATLAB Production Server library as a reference, and define it in the web applica tion (Figure 12). You can find the library in the MATLAB Production Server client library folder:

```
<matlabroot>\MATLAB Production Server\R2015a\client\dotnet
```

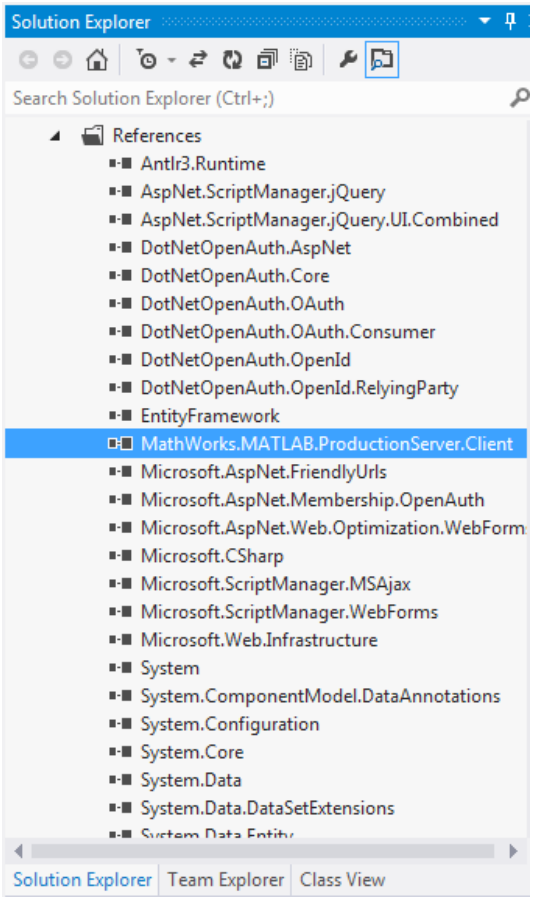


Figure 12. Referencing the MATLAB Production Server library.

3. Add a UI input to the web project's default.aspx design view, such as a pushbutton (Figure 13).

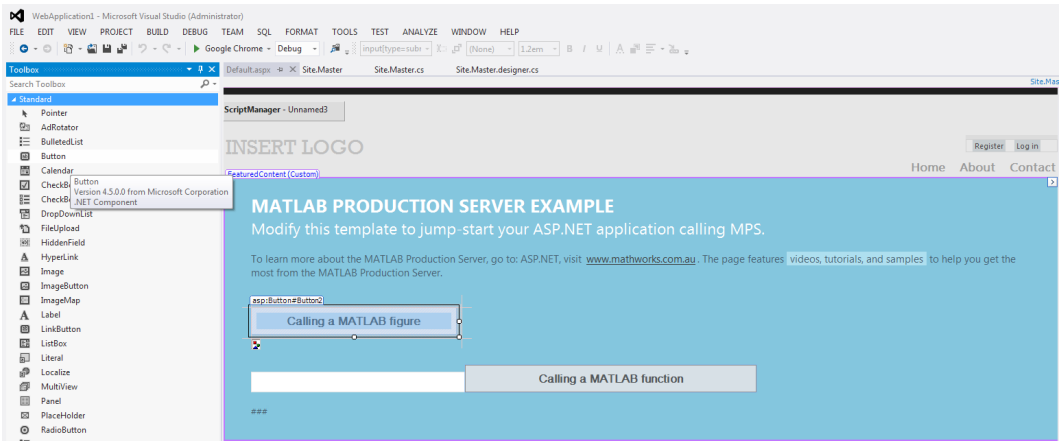


Figure 13. Adding UI objects to a web project.

4. Add a reference to the MATLAB Production Server in the C# code (Figure 14).

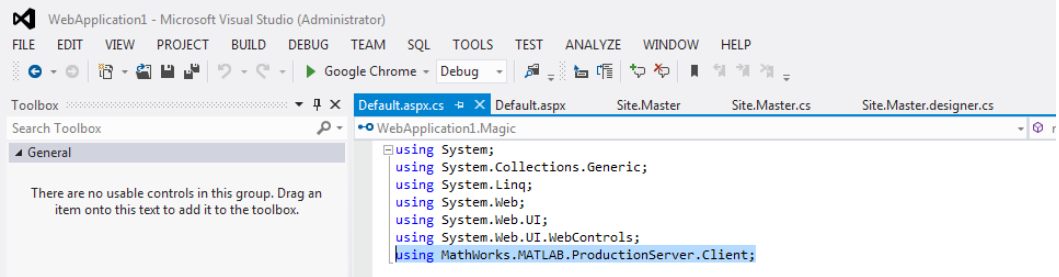


Figure 14. Referencing the MATLAB Production Server library in the C# code.

5. Define the function inputs and outputs as an interface (Figure 15).

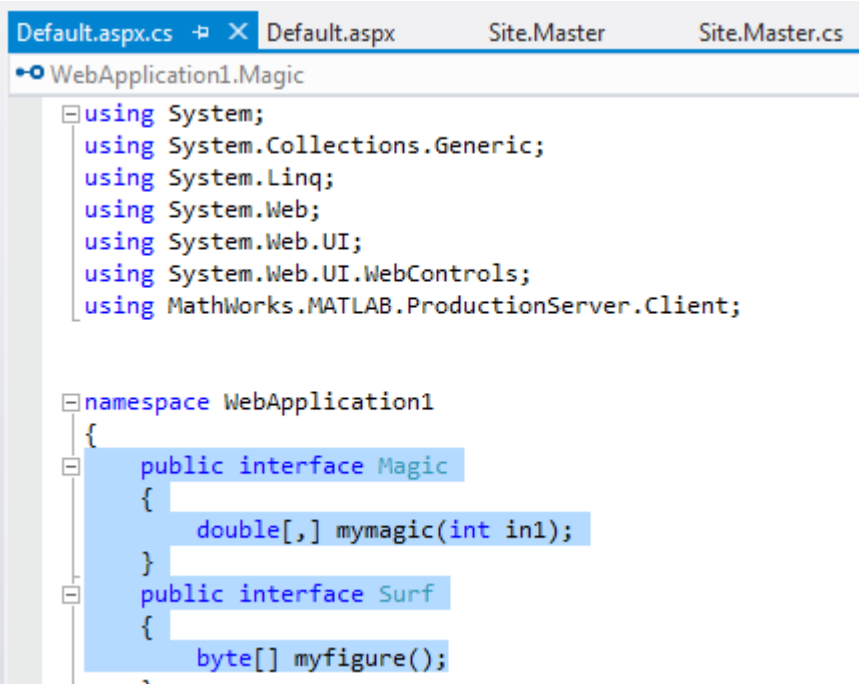


Figure 15. Defining the inputs and outputs to the compiled MATLAB function in C#.

6. Call the MATLAB function from the MATLAB Production Server client integration and then from the final compiled version (Figure 16). To do this:

   A. Initiate an HTTP object.

   B. Call the MATLAB function via the server IP, port number, and function name.

C. If the test version is verified, ask the MATLAB developer to complete the steps of compiling the function and copying it to the MATLAB Production Server frame work (as described in step 3 of the previous MATLAB Developer section). No changes should be needed to then call the compiled version. However, if you still have the client integration tool running, you may want to change the port number.

Default.aspx.cs | Default.aspx | Site.Master | Site.Master.cs | Site.Master.designer.cs

WebApplication1._Default | Button1_

```
namespace WebApplication1
{
    public interface Magic
    {
        double[,] mymagic(int in1);
    }
    public interface Surf
    {
        byte[] myfigure();
    }
    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            string a;
            a = TextBox1.Text;
            int b;
            b = Convert.ToInt16(a);
            MWClient client = new MWHttpClient();
            Magic me = client.CreateProxy<Magic>
                            (new Uri("http://localhost:9911/mymagic"));
            double[,] result1 = me.mymagic(b);
            int rank = result1.Rank;
            int[] dims = new int[rank];

            for (int i = 0; i < rank; i++)
            {
                dims[i] = result1.GetLength(i);
            }

            for (int j = 0; j < dims[0]; j++)
            {
                TableRow r = new TableRow();
                for (int k = 0; k < dims[1]; k++)
                {
                    double z = result1[j, k];
                    TableCell c = new TableCell();
                    c.Controls.Add(new LiteralControl(z.ToString()));
                    r.Cells.Add(c);
                }
                Table1.Rows.Add(r);
            }


        }
```

Figure 16. Calling the MATLAB function in C#.

7. Integrate the output of the function as necessary on the web site. Some examples include:

   A. A matrix output

```
protected void Button1 _ Click(object sender, EventArgs e)
     {
          string a;
          a = TextBox1.Text;
          int b;
          b = Convert.ToInt16(a);
          MWClient client = new MWHttpClient();
          Magic me = client.CreateProxy<Magic>
                              (new Uri("http://localhost:9911/
mymagic"));
          double[,] result1 = me.mymagic(b);
          int rank = result1.Rank;
          int[] dims = new int[rank];

          for (int i = 0; i < rank; i++)
          {
               dims[i] = result1.GetLength(i);
          }

          for (int j = 0; j < dims[0]; j++)
          {
               TableRow r = new TableRow();
               for (int k = 0; k < dims[1]; k++)
               {
                    double z = result1[j, k];
                    TableCell c = new TableCell();
                    c.Controls.Add(new LiteralControl(z.
ToString()));
                    r.Cells.Add(c);
               }
               Table1.Rows.Add(r);
          }
```

```
        }
```

B. A figure outputted as a static image

```
    protected void Button2 _ Click(object sender, EventArgs e)
        {
            MWClient client = new MWHttpClient();
            Surf me = client.CreateProxy<Surf>
                        (new Uri("http://localhost:9910myfigure"));
            byte[] imgArray = me.myfigure();
            string base64String = ...
Convert.ToBase64String(imgArray, 0, imgArray.Length);
            Image1.ImageUrl = "" +
base64String;
            Image1.Visible = true;
        }
```

C. Integrated with a third-party interactive charting library, such as HighCharts

Full C# code below:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using MathWorks.MATLAB.ProductionServer.Client;

namespace WebApplication1
{
    public interface Magic
    {
        double[,] mymagic(int in1);
    }
```

```
public interface Surf
{
     byte[] myfigure();
}
public partial class _Default : Page
{
     protected void Page_Load(object sender, EventArgs e)
     {

     }

     protected void Button1_Click(object sender, EventArgs e)
     {
          string a;
          a = TextBox1.Text;
          int b;
          b = Convert.ToInt16(a);
          MWClient client = new MWHttpClient();
          Magic me = client.CreateProxy<Magic>
                       (new Uri("http://localhost:9911/mymagic"));
          double[,] result1 = me.mymagic(b);
          int rank = result1.Rank;
          int[] dims = new int[rank];

          for (int i = 0; i < rank; i++)
          {
               dims[i] = result1.GetLength(i);
          }

          for (int j = 0; j < dims[0]; j++)
          {
               TableRow r = new TableRow();
               for (int k = 0; k < dims[1]; k++)
               {
                    double z = result1[j, k];
```

```
                    TableCell c = new TableCell();
                    c.Controls.Add(new LiteralControl(z.
ToString()));
                    r.Cells.Add(c);
                }
                Table1.Rows.Add(r);
            }

        }

        protected void TextBox1_TextChanged ...
(object sender, EventArgs e)
        {
            int c;
        }

        protected void Button2_Click(object sender, EventArgs e)
        {
            MWClient client = new MWHttpClient();
            Surf me = client.CreateProxy<Surf>
                        (new Uri("http://localhost:9910/myfigure"));
            byte[] imgArray = me.myfigure();
            string base64String = Convert.ToBase64String(imgArray,
0, imgArray.Length);
            Image1.ImageUrl = "" +
base64String;
            Image1.Visible = true;
        }

    }
}
```

Example 2: Simple Web Site with Interactive Charts

This example outputs a few simple MATLAB functions and integrates the output with the third-party interactive web charting package, ***Highcharts***. You could also use Vega or D3 charts.

The following is a screen shot of the output:

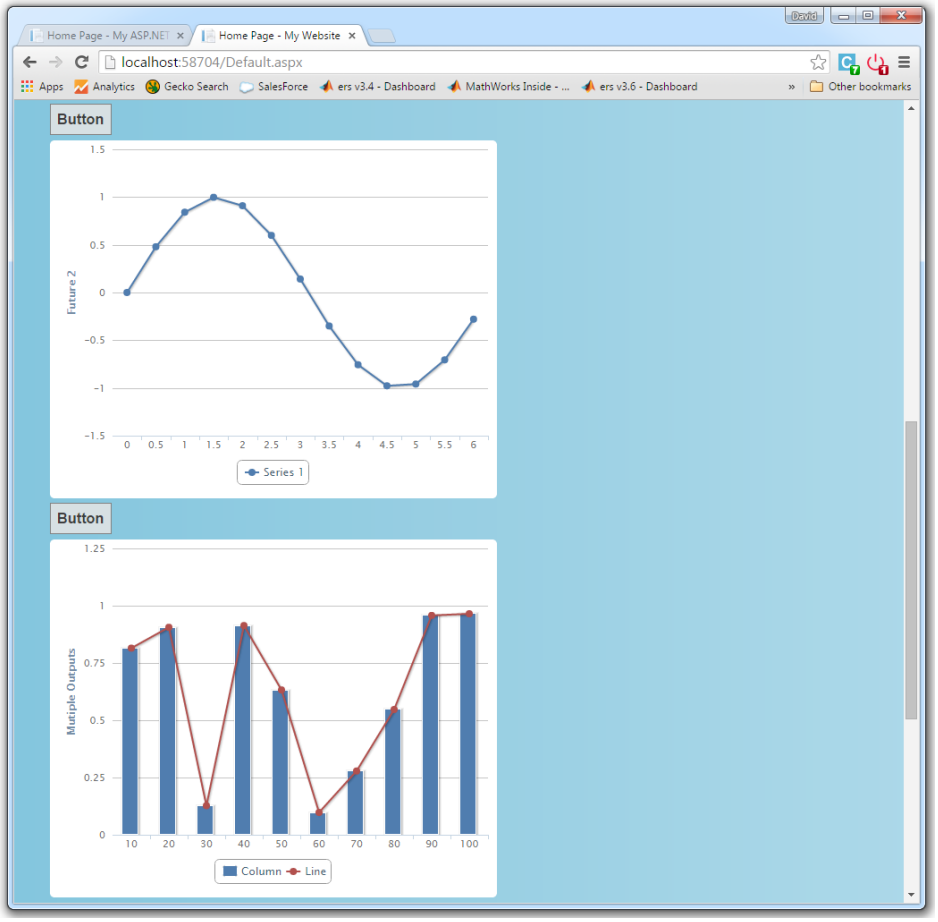


Figure 17. Snapshot integrating MATLAB analytics with interactive web charts.

The MATLAB Developer Workflow

1. Develop the function to deploy to a web site using the following two functions for plotting:

myplot.m:

```
function [x,y] = myrand
%x - output 1 double vector
%y - output 2 double vector
x = [10:10:100]';
y = rand(length(x),1);
```

myrand.m:

```
function y = myplot
%y(:,1) - output 1 double vector
%y(:,2)- output 2 double vector
y(:,1) = [0:0.5:2*pi]';
y(:,2) = sin(y(:,1));
```

2. Document the typical inputs and outputs for the function as a reference guide for the web developer. As seen in the previous step, both functions output two double vectors.
3. Test and compile the function with the MATLAB Compiler SDK to deploy to the MATLAB Production Server (Figures 6 and 7).
4. Copy the deployed function to the MATLAB Production Server framework (Figure 9).

Products used include: MATLAB, MATLAB Compiler, (for function deployment) MATLAB Compiler SDK, and MATLAB Production Server.

The Web Developer Workflow: Using .NET

1. Create a new web project in Visual Studio, as shown in Example 1.
2. Import the MATLAB Production Server library as a reference and define it in the web application (Figures 12 and 14).
3. Define the function inputs and outputs as an interface (Figure 15).
4. Call the MATLAB function from the MATLAB Production Server client integration tool and then the final compiled version (Figure 16):
    a. Initiate an HTTP object.
    b. Call the MATLAB function via the server IP, port number, and function name.

c. If the test version is verified, ask the MATLAB developer to finish compiling the function and copy it to the MATLAB Production Server framework. No changes should be needed to then call the compiled version. However, if you still have the client integration tool running, you may want to change the port number.

5. Integrate the output of the function as necessary in the web site with the third-party interactive charting library, Highcharts.

   a. The use of Highcharts requires you to download the library from the web, and then add it as a reference to the project. (Highcharts is a commercial charting package and requires you to purchase it if you deploy your web site to the web). Find the Highcharts library at:

   https://www.highcharts.com/

   b. To use the library:

      i. Highcharts is built in JavaScript. Reference the JS library for .NET in Site.Master:

```
<script src="http://ajax.googleapis.com/ajax/libs/ ...
jquery/1.3.2/jquery.min.js" type="text/javascript"></
... script>
```

      ii. Manually add the Highchart to the web design code Default.aspx. For example:

```
<highchart:LineChart runat="server" ID="hcline2" ...
Width="500" Height="400" />
```

   c. Convert the double output to the data type object, which is used by the Highcharts library.

   d. See the YouTube tutorial on integrating Highcharts in ASP.NET webforms:

   https://youtu.be/IhfKUUAndVo?list=PL8kKejlVENAUR7VRwf8fff5r9JEn78Zys

Full C# code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using MathWorks.MATLAB.ProductionServer.Client;
using Highchart.UI;
using Highchart.Core;
using Highchart.Core.Data.Chart;
```

```
using Highchart.Core.PlotOptions;

namespace WebApplication2
{
     public interface Magic
     {
          double[,] mymagic(int in1);
     }
     public interface Surf
     {
          byte[] myfigure();
     }
     public interface myplot
     {
          double[,] myplot();
     }
     public interface myrand
     {
          void myrand(out double[] o1, out double[] o2);
     }
     public partial class _Default : Page
     {
          protected void Page_Load(object sender, EventArgs e)
          {
          }
          protected void Button2_Click(object sender, EventArgs e)
          {
               string a;
               a = TextBox1.Text;
               int b;
               b = Convert.ToInt16(a);
               MWClient client = new MWHttpClient();
               Magic me = client.CreateProxy<Magic>
                              (new Uri("http://localhost:9910/mymagic"));
               double[,] result1 = me.mymagic(b);
```

```
            int rank = result1.Rank;
            int[] dims = new int[rank];

            for (int i = 0; i < rank; i++)
            {
                dims[i] = result1.GetLength(i);
            }

            for (int j = 0; j < dims[0]; j++)
            {
                TableRow r = new TableRow();
                for (int k = 0; k < dims[1]; k++)
                {
                    double z = result1[j, k];
                    TableCell c = new TableCell();
                  c.Controls.Add(new LiteralControl(z.ToString()));
                    r.Cells.Add(c);
                }
                Table1.Rows.Add(r);
            }

        }

        protected void TextBox1 _ TextChanged(object sender, ...
EventArgs e)
        {
            int c;
        }

        protected void Button1 _ Click(object sender, EventArgs e)
        {
            MWClient client = new MWHttpClient();
            Surf me = client.CreateProxy<Surf>
                        (new Uri("http://localhost:9910/myfigure"));
            byte[] imgArray = me.myfigure();
```

```
            string base64String = Convert.ToBase64String ...
(imgArray, 0, imgArray.Length);
            Image1.ImageUrl = "" + ...
base64String;
            Image1.Visible = true;
        }

        protected void Button4 _ Click(object sender, EventArgs e)
        {
            MWClient client = new MWHttpClient();
            myplot me = client.CreateProxy<myplot>
                        (new Uri("http://localhost:9910/myplot"));
            double[,] result2 = me.myplot();
            double[] col2 = new double[result2.GetLength(0)];
            int col = 1;

            for (int i = 0; i < result2.GetLength(0); i++)
            {
                col2[i] = result2[i, col];
            }

            double[] col1 = new double[result2.GetLength(0)];
            string[] col1str = new string[result2.GetLength(0)];

            int col _ 0 = 0;

            for (int i = 0; i < result2.GetLength(0); i++)
            {
                double temp = result2[i, col _ 0];
                col1[i] = result2[i, col _ 0];
                col1str[i] = temp.ToString();
            }

            object[] object _ array = new object[col2.Length];
            col2.CopyTo(object _ array, 0);
```

```
            object[] object_array2 = new object[col1str.Length];
            col1str.CopyTo(object_array2, 0);

            hcline2.YAxis.Add(new YAxisItem { title = new ...
Title("Future 2") });
            hcline2.XAxis.Add(new XAxisItem {categories =
col1str});
            //hcline2.XAxis.Add(new XAxisItem { categories =  {}
});

            //New data collection
            var series = new List<Serie>();
            series.Add(new Serie { data = object_array });

            //bind
            hcline2.DataSource = series;
            hcline2.DataBind();
        }

        protected void Button5_Click(object sender, EventArgs e)
        {
            MWClient client = new MWHttpClient();
            myrand me = client.CreateProxy<myrand>
                        (new Uri("http://localhost:9910/myrand"));
            double[] o1;
            double[] o2;
            me.myrand(out o1, out o2);

            object[] o1x = new object[o1.Length];
            o1.CopyTo(o1x, 0);
            object[] o2y = new object[o2.Length];
            o2.CopyTo(o2y, 0);

            string[] o1xstr = new string[o1.GetLength(0)];

            for (int i = 0; i < o1.GetLength(0); i++)
```

```
                {
                    o1xstr[i] = o1[i].ToString();
                }

                hcline3.YAxis.Add(new YAxisItem { title = new ...
Title("Mutiple Outputs") });
            hcline3.XAxis.Add(new XAxisItem { categories = o1xstr });
              //hcline2.XAxis.Add(new XAxisItem { categories =  {} });

                //New data collection
                var series = new List<Serie>();
               series.Add(new Serie { data = o2y , name= "Column" });
                series.Add(new Serie { data = o2y, name = "Line", ...
type = RenderType.line });
                //bind
                hcline3.DataSource = series;
                hcline3.DataBind();

            }

 protected void TextBox2 _ TextChanged(object sender, EventArgs e)
            {

            }
    }
}
```

Example 3: Mining Economics Case Study

This example shows how to use the concepts presented in Examples 1 and 2 to create a web site that forecasts the economics of an iron ore mine.

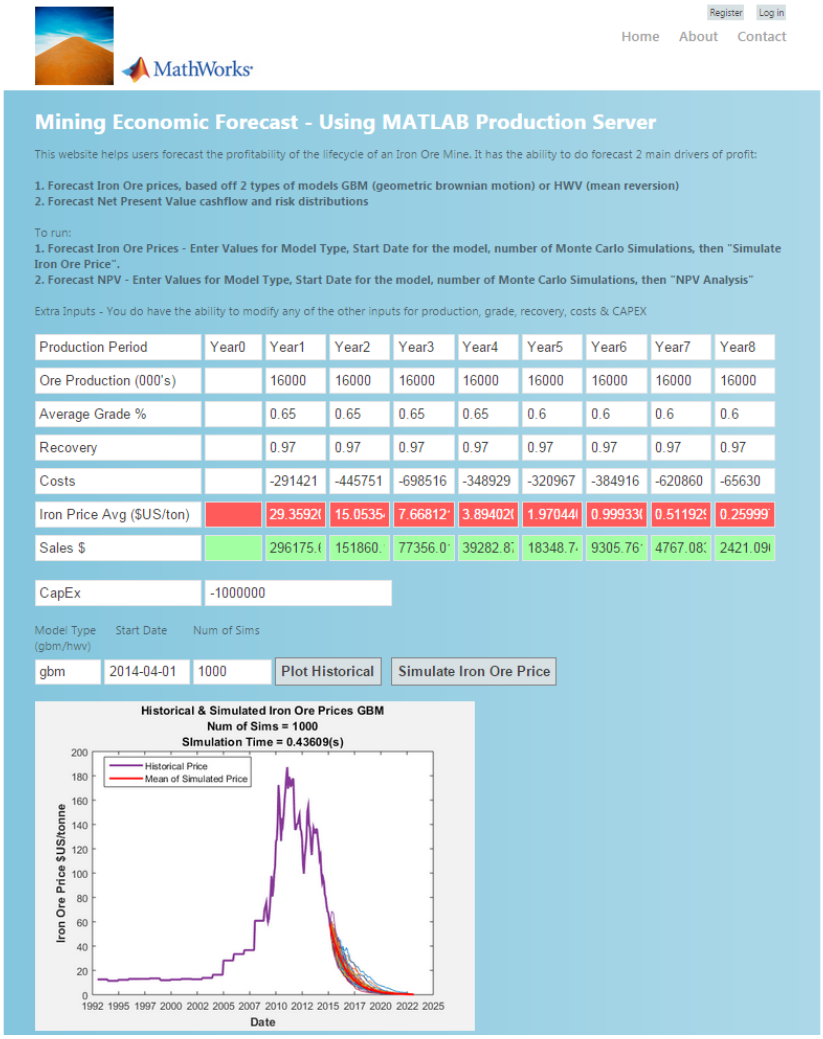
This example integrates powerful analytics into a web site, enabling an end user to run thousands of simulations and perform advanced financial analysis in a fraction of a second.

Application overview: To mine or not to mine? In recent times, this has become a very important question to ask. Would you be willing to make this decision without the proper research on the future economic potential of a mine? Or perhaps more importantly, what's the risk of making a decision without having the research backed up by detailed functions that forecast potential outcomes?

This web site enables an end user to:

- Simulate the future iron ore prices by Monte Carlo simulation based on different function types and function start dates
- Calculate the Net Present Value (NPV) cumulative average cash flow for the life of the mine
- Calculate the risk distribution of the range of the final Net Present Values from the simulation

The following is a screen shot of the output:



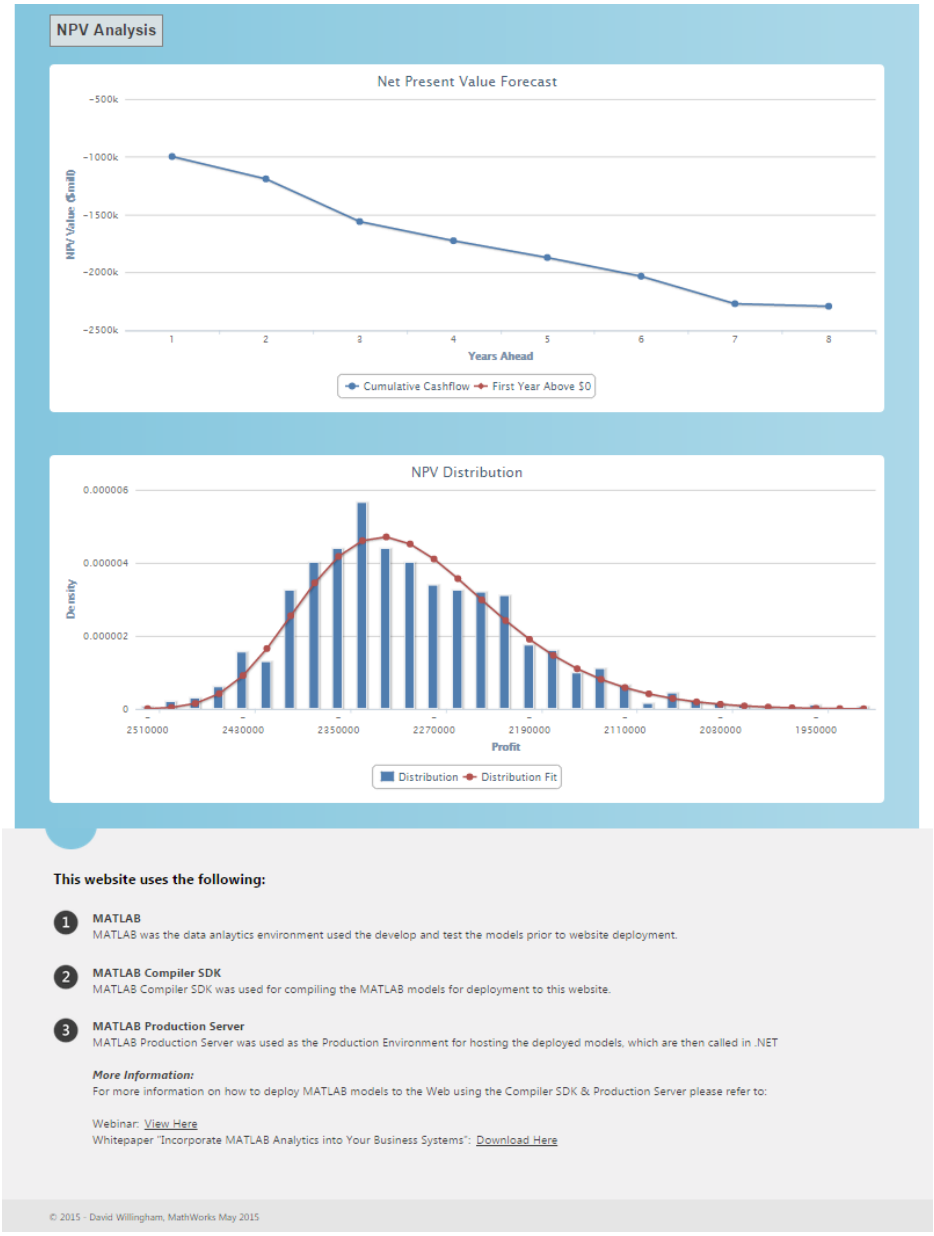


Figure 18. A mining application web site case study that leverages the full power of MATLAB with thousands of Monte Carlo simulations and financial risk analysis.

The MATLAB Developer Workflow

1. Develop the function to deploy to a web site. This example has two functions: `ironoreprice _ func _ web.m` and `NPV _ Analysis _ web.m`.

```
function [myfig,pricem] = ironoreprice _ func _ web(functionsel,star
tdate,NTrials,histonly)
% output myfig byte[]
% output pricem double[]
% input functionsel string
% input startdate string
% input NTrials double
% input hist only double
```

```
function [tempx,temp,tempx0,temp0,priceore,salesm,BinCenter,BinHeig
ht, YPlot] = NPV _ Analysis _ web(functionsel,startdate,NTrials,prod,
grade,rec,costsT,capex)
% output tempx double[]
% output temp double[]
% output tempx0 double[]
% output temp0 double[]
% output priceore double[]
% output salesm double[]
% output BinCenter double[]
% output BinHeight double[]
% output YPlot double[]
% input functionsel string
% input startdate string
% input NTrials double
% input prod double[]
% input grade double[]
% input rec double[]
% input costsT double[]
% input capex double
```

2. Document the typical inputs and outputs to the function, which are used a reference guide for the web developer. Refer to the preceding code.
3. Test and compile the function with MATLAB Compiler SDK to deploy to MATLAB Production Server.
4. Copy the deployed function to the MATLAB Production Server framework.

Products used include: MATLAB, Financial Toolbox™, Statistics and Machine Learning Toolbox™, Optimization Toolbox, MATLAB Compiler, (for function deployment) MATLAB Compiler SDK, and MATLAB Production Server.

The Web Developer Workflow: Using .NET

1. Create a new web project in Visual Studio.
2. Import the MATLAB Production Server library as a reference and define it in the Web Application.
3. Define the function inputs and outputs as an interface.
4. Call the MATLAB function from the MATLAB Production Server test environment and then the final compiled version:
    a. Initiate an HTTP object.
    b. Call the MATLAB function via the Server IP, port number, and function name.
    c. If the test version is verified, ask the MATLAB developer to complete the steps of compil ing the function and copying it to the MATLAB Production Server framework. No changes should be needed to then call the compiled version. However, if you still have the test envi ronment running, you may want to change the port number.
5. Integrate the output of the function as necessary in the web site. Examples include:
    a. A matrix outputted to text boxes
    b. A figure outputted as a static image
    c. Integrate with a third-party interactive charting library, such as HighCharts

Full C# code:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
```

```
using MathWorks.MATLAB.ProductionServer.Client;
using Highchart.UI;
using Highchart.Core;
using Highchart.Core.Data.Chart;
using Highchart.Core.PlotOptions;

namespace WebApplication1
{
    public interface myimage
    {
        void ironoreprice_func_web(out byte[] out1, out
double[] out2, string in1, string in2, double in3, double in4);
    }

    public interface NPV
    {
        void NPV_Analysis_web(out double[] out1, out double[]
out2, out double[] out3, out double[] out4, out double[] out5, out
double[] out6, out double[] out7, out double[] out8, out double[]
out9, string in1, string in2, double in3, double[] in4, double[]
in5, double[] in6, double[] in7,double in8);
    }
    public partial class _Default : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {

        }

        protected void Button1_Click(object sender, EventArgs e)
        {
            MWClient client = new MWHttpClient();
            myimage me = client.CreateProxy<myimage>
        (new Uri("http://localhost:9910/ironoreprice_func_web"));
            string in1;
            in1 = function.Text;
```

```
            string in2;
            in2 = datesel.Text;

            double in3;
            double in4;
            string a;
            a = ntrials.Text;
            in3 = Convert.ToDouble(a);
            in4 = 1;
            byte[] out1;
            double[] out2;

me.ironoreprice_func_web(out out1, out out2, in1, in2, ...
in3, in4);


string base64String = Convert.ToBase64String(out1, 0, out1.Length);
        Image1.ImageUrl = "" + base64String;
            Image1.Visible = true;
        }

 protected void TextBox5_TextChanged(object sender, EventArgs e)
        {

        }

 protected void TextBox6_TextChanged(object sender, EventArgs e)
        {

        }

 protected void TextBox4_TextChanged(object sender, EventArgs e)
        {
```

```
        }
        protected void Button2 _ Click(object sender, EventArgs e)
        {
            MWClient client = new MWHttpClient();
            myimage me = client.CreateProxy<myimage>
        (new Uri("http://localhost:9910/ironoreprice _ func _ web"));
            string in1;
            in1 = function.Text;

            string in2;
            in2 = datesel.Text;

            double in3;
            double in4;
            string a;
            a = ntrials.Text;
            in3 = Convert.ToDouble(a);
            in4 = 0;
            byte[] out1;
            double[] out2;

            me.ironoreprice _ func _ web(out out1, out out2, ...
in1,  in2, in3, in4);

            a = Convert.ToString(out2[0]);
            iron1.Text = a;
            a = Convert.ToString(out2[1]);
            iron2.Text = a;
            a = Convert.ToString(out2[2]);
            iron3.Text = a;
            a = Convert.ToString(out2[3]);
            iron4.Text = a;
            a = Convert.ToString(out2[4]);
            iron5.Text = a;
            a = Convert.ToString(out2[5]);
```

```
                iron6.Text = a;
                a = Convert.ToString(out2[6]);
                iron7.Text = a;
                a = Convert.ToString(out2[7]);
                iron8.Text = a;

string base64String = Convert.ToBase64String(out1, 0, out1.Length);
                Image1.ImageUrl = "" + ...
base64String;
                Image1.Visible = true;
          }

          protected void Button3 _ Click(object sender, EventArgs e)
          {
               MWClient client = new MWHttpClient();
               NPV me = client.CreateProxy<NPV>
                (new Uri("http://localhost:9910/NPV _ Analysis _ web"));

               string in1;
               in1 = function.Text;

               string in2;
               in2 = datesel.Text;

               double in3;

               string a;
               a = ntrials.Text;
               in3 = Convert.ToDouble(a);

               double[] out1;
               double[] out2;
               double[] out3;
               double[] out4;
               double[] out5;
```

```
double[] out6;
double[] out7;
double[] out8;
double[] out9;

double[] in4 = new double[8];
in4[0] = Convert.ToDouble(prod1.Text);
in4[1] = Convert.ToDouble(prod2.Text);
in4[2] = Convert.ToDouble(prod3.Text);
in4[3] = Convert.ToDouble(prod4.Text);
in4[4] = Convert.ToDouble(prod5.Text);
in4[5] = Convert.ToDouble(prod6.Text);
in4[6] = Convert.ToDouble(prod7.Text);
in4[7] = Convert.ToDouble(prod8.Text);

double[] in5 = new double[8];
in5[0] = Convert.ToDouble(avg1.Text);
in5[1] = Convert.ToDouble(avg2.Text);
in5[2] = Convert.ToDouble(avg3.Text);
in5[3] = Convert.ToDouble(avg4.Text);
in5[4] = Convert.ToDouble(avg5.Text);
in5[5] = Convert.ToDouble(avg6.Text);
in5[6] = Convert.ToDouble(avg7.Text);
in5[7] = Convert.ToDouble(avg8.Text);

double[] in6 = new double[8];
in6[0] = Convert.ToDouble(rec1.Text);
in6[1] = Convert.ToDouble(rec2.Text);
in6[2] = Convert.ToDouble(rec3.Text);
in6[3] = Convert.ToDouble(rec4.Text);
in6[4] = Convert.ToDouble(rec5.Text);
in6[5] = Convert.ToDouble(rec6.Text);
in6[6] = Convert.ToDouble(rec7.Text);
in6[7] = Convert.ToDouble(rec8.Text);
```

```
            double[] in7 = new double[8];
            in7[0] = Convert.ToDouble(cost1.Text);
            in7[1] = Convert.ToDouble(cost2.Text);
            in7[2] = Convert.ToDouble(cost3.Text);
            in7[3] = Convert.ToDouble(cost4.Text);
            in7[4] = Convert.ToDouble(cost5.Text);
            in7[5] = Convert.ToDouble(cost6.Text);
            in7[6] = Convert.ToDouble(cost7.Text);
            in7[7] = Convert.ToDouble(cost8.Text);

            double in8;
            in8 = Convert.ToDouble(capex.Text);

            me.NPV_Analysis_web(out out1, out out2, out out3,
out out4, out out5, out out6, out out7, out out8, out out9, in1,
in2, in3, in4, in5, in6, in7, in8);

            a = Convert.ToString(out5[0]);
            iron1.Text = a;
            a = Convert.ToString(out5[1]);
            iron2.Text = a;
            a = Convert.ToString(out5[2]);
            iron3.Text = a;
            a = Convert.ToString(out5[3]);
            iron4.Text = a;
            a = Convert.ToString(out5[4]);
            iron5.Text = a;
            a = Convert.ToString(out5[5]);
            iron6.Text = a;
            a = Convert.ToString(out5[6]);
            iron7.Text = a;
            a = Convert.ToString(out5[7]);
            iron8.Text = a;

            a = Convert.ToString(out6[0]);
```

```
sales1.Text = a;
a = Convert.ToString(out6[1]);
sales2.Text = a;
a = Convert.ToString(out6[2]);
sales3.Text = a;
a = Convert.ToString(out6[3]);
sales4.Text = a;
a = Convert.ToString(out6[4]);
sales5.Text = a;
a = Convert.ToString(out6[5]);
sales6.Text = a;
a = Convert.ToString(out6[6]);
sales7.Text = a;
a = Convert.ToString(out6[7]);
sales8.Text = a;

//string[] out1s;
//out1s = out1.ToString();

string[] out1s = new string[out1.GetLength(0)];
string[] out3s = new string[out3.GetLength(0)];

for (int i = 0; i < out1.GetLength(0); i++)
{
    out1s[i] = out1[i].ToString();
    out3s[i] = out3[i].ToString();
}

object[] out2x = new object[out2.Length];
out2.CopyTo(out2x, 0);
object[] out4x = new object[out4.Length];
out4.CopyTo(out4x, 0);

hcline.Series.Clear();
```

```
            hcline.Title = new Title ...
("Net Present Value Forecast");
            hcline.YAxis.Add(new YAxisItem ...
{ title = new Title("NPV Value ($mill)") });
            hcline.XAxis.Add(new XAxisItem ...
{ categories = out1s, title = new Title("Years Ahead") });

            //hcline.PlotOptions
            //hcline.PlotOptions = new PlotOptionsLine { marker =
new Marker("diamond")  };
            //hcline4.XAxis.Add(new XAxisItem { categories =  {}
});

            //New data collection
            var series = new List<Serie>();
            series.Add(new Serie { data = out2x, name =
"Cumulative Cashflow"});
            series.Add(new Serie { data = out4x, name = "First
Year Above $0"} );

            //series.Add(new Serie { data = o2x, name =
"Inflation (annualized quarterly)", type = RenderType.line });

            //bind
            hcline.DataSource = series;
            hcline.DataBind();

            string[] out7s = new string[out7.GetLength(0)];


            for (int i = 0; i < out7.GetLength(0); i++)
            {
                out7s[i] = out7[i].ToString();

            }

            object[] out8x = new object[out8.Length];
```

```
            out8.CopyTo(out8x, 0);
            object[] out9x = new object[out9.Length];
            out9.CopyTo(out9x, 0);

            hcline0.Series.Clear();
            hcline0.Title = new Title("NPV Distribution");
            hcline0.YAxis.Add(new YAxisItem ...
{ title = new Title("Density") });
            hcline0.XAxis.Add(new XAxisItem ...
{ categories = out7s, title = new Title("Profit"), ...
tickInterval = 4 });

            //hcline.PlotOptions
            //hcline.PlotOptions = new PlotOptionsLine { marker =
new Marker("diamond")  };
            //hcline4.XAxis.Add(new XAxisItem { categories =  {}
});

            //New data collection
            var series2 = new List<Serie>();
            series2.Add(new Serie { data = out8x, name =
"Distribution" });
            series2.Add(new Serie { data = out9x, name =
"Distribution Fit", type = RenderType.line });

            //series.Add(new Serie { data = o2x, name =
"Inflation (annualized quarterly)", type = RenderType.line });

            //bind
            hcline0.DataSource = series2;
            hcline0.DataBind();
        }

        protected void TextBox5 _ TextChanged1 ...
(object sender, EventArgs e)
        {
```

```
            }

            protected void capex_TextChanged ...
(object sender, EventArgs e)
            {

            }

        }
}
```

Summary

The examples in this article are intended to provide an overview of how to create and deploy MATLAB analytics to the web. This is in no way an extensive treatment of web deployment. In fact, there are entire books devoted just to the topic of web deployment.

It is important to recognize that the task of creating and deploying MATLAB analytics to a web site is best handled by at least two distinct roles:

- The MATLAB developer, who is an expert in science, data analysis and engineering
- A web developer adept in creating web sites and the server infrastructure to power them

MATLAB Production Server provides a ready-made deployment infrastructure to enable smooth interaction between the two roles. It is flexible enough to support many clients. If you need help with creating web sites from MATLAB applications, you can also use ***MathWorks Consulting Services,*** as the team routinely helps clients with such tasks. It also provides guidance towards making organizations self-sufficient in the web development process.