MATLAB®
Web Server

The Language of Technical Computing

Computation

Visualization

Programming

The MATH WORKS Inc.

MATLAB Web Server

Version 1

How to Contact The MathWorks:

508-647-7000 Phone

508-647-7001 Fax

The MathWorks, Inc.
24 Prime Park Way
Natick, MA 01760-1500 Mail

`http://www.mathworks.com`	Web
`ftp.mathworks.com`	Anonymous FTP server
`comp.soft-sys.matlab`	Newsgroup

`support@mathworks.com`	Technical support
`suggest@mathworks.com`	Product enhancement suggestions
`bugs@mathworks.com`	Bug reports
`doc@mathworks.com`	Documentation error reports
`subscribe@mathworks.com`	Subscribing user registration
`service@mathworks.com`	Order status, license renewals, passcodes
`info@mathworks.com`	Sales, pricing, and general information

MATLAB Web Server

Printing History: January 1999 First printing New for Version 1.0 (Release 11)

Contents

1 MATLAB on the Web

2 Inside the MATLAB Web Server

3 Reference

A Directory Structure

Administration and Troubleshooting

B

References

C

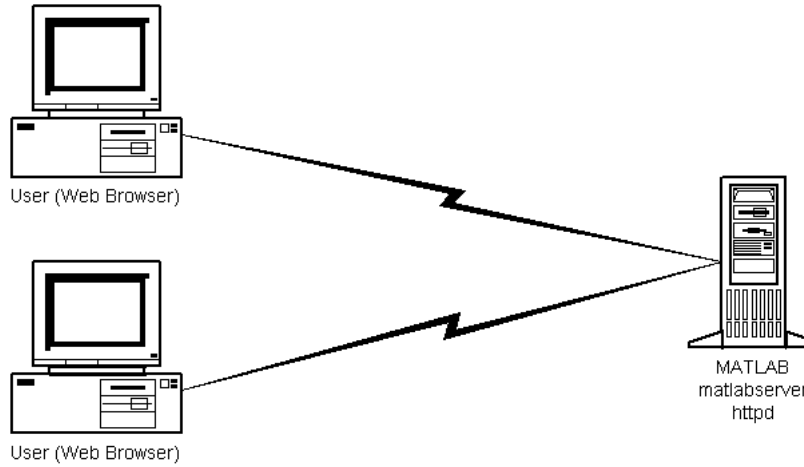
1

MATLAB on the Web

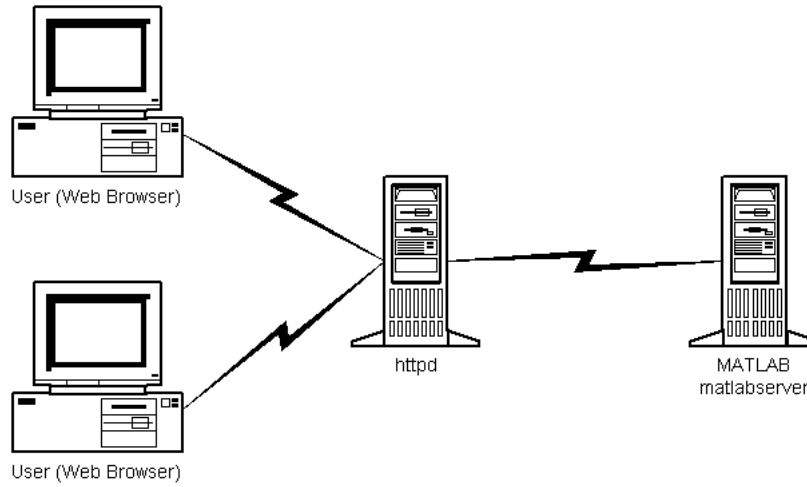
Introduction

MATLAB Web Server Environment

The MATLAB® Web Server enables you to create MATLAB applications that use the capabilities of the World Wide Web to send data to MATLAB for computation and to display the results in a Web browser. In the simplest configuration, a Web browser runs on your client workstation, while MATLAB, the MATLAB Web Server (`matlabserver`), and the Web server daemon (`httpd`) run on another machine:



In a more complex network, the Web server daemon can run on a machine apart from the others:



The MATLAB Web Server depends upon TCP/IP networking for transmission of data between the client system and MATLAB. The required networking software and hardware must be installed on your system prior to using the MATLAB Web Server.

Building MATLAB Web Server Applications

Using HTML

MATLAB Web Server applications are a combination of M-files, Hypertext Markup Language (HTML), and graphics. Knowledge of MATLAB programming and basic HTML are the only requirements.

The application development process requires a small number of simple steps:

1. Create the HTML documents for collection of the input data from users and display of output. You can code the input documents using a text editor to input HTML directly, or you can use one of the commercially available

HTML authoring systems, such as Front Page from Microsoft, PageMill from Adobe, or HoTMetaL from SoftQuad.

2. List the application name and associated configuration data in the configuration file `matweb.conf`. (See `matweb.conf` on page 2-6 for a description of this file.)

3. Write a MATLAB M-file that:

   a. Receives the data entered in the HTML input form.

   b. Analyzes the data and generates any requested graphics.

   c. Places the output data into a MATLAB structure.

   d. Calls `htmlrep` to place the output data into an HTML output document template. (See `htmlrep` on page 3-3 for a description of this process.) The maximum amount of HTML data you can receive from MATLAB is 256 KB.

Product Requirements

The MATLAB Web Server has the same supporting hardware and software requirements as MATLAB 5, except for memory required. MATLAB hardware and software requirements are documented in the *Installation Guide* for your computer.

Memory requirements while running the MATLAB Web Server vary with the number of MATLABs configured. Each MATLAB running under the MATLAB Web Server consumes 256 KB of memory.

The MATLAB Web Server requires that TCP/IP networking software must be installed on your computer.

Consult the document *Known Software And Documentation Problems* for any last minute changes to hardware or software requirements.

Web Requirements

Web Browsers

To submit input to and receive output from the MATLAB Web Server, you must install a Web browser suitable for your platform. Current versions of the MATLAB Web Server have been tested with Netscape Navigator Release 3.0 and Microsoft Internet Explorer 3.0.

The MathWorks does not redistribute these products. You can obtain them directly from the companies that developed them. You can find additional information at `www.netscape.com` or `www.microsoft.com`.

Web Server

You need to install Web server software (`httpd` or similar) on the system where MATLAB is running or on a machine that has network access to the machine where MATLAB is running. There are numerous sources for obtaining this software, including:

- Pre-installed Microsoft Peer Networking Services on your PC
- Netscape Enterprise Server, available by purchase from Netscape Communications, Inc.
- Free distribution over the Internet (Apache: `www.apache.org`)

The Web server software must be capable of running Common Gateway Interface (CGI) programs.

Installation

Availability

The MATLAB Web Server is available on UNIX (Solaris) workstations and IBM PC compatible computers running Microsoft Windows NT.

Installation Procedure

To install the MATLAB Web Server, follow the normal MATLAB installation procedure for your platform, as documented in the *MATLAB Installation Guide*. The MATLAB Web Server appears as one of the installation choices you can select as you proceed through the installation screens.

General Post-Installation Procedures

After installing MATLAB and the MATLAB Web Server, you must perform a number of steps that regulate communication between MATLAB and your Web browser. Check the `Readme` file located in `<matlab>toolbox/webserver/webserver` for any last minute information concerning installation of this release.

Note in particular:

1. To get the demonstration programs discussed in this chapter to work, you need to create a `matweb.conf` file in the `<matlab>/toolbox/webserver/wsdemos` directory. The `Readme` file shows the format for `matweb.conf`. Replace the notation `<MATLABROOT>` with the name of the root directory where you installed MATLAB. Use the `matlabroot` command to determine this directory. Also, replace `MATLABSERVER_HOST_NAME` with the TCP/IP hostname for your machine.

2. The installation procedure creates the file `matlabserver.conf` in the `<matlab>/webserver` directory. The file contains the notation

   ```
   -m 1
   ```

   This number represents the number of MATLABs that can run concurrently. After testing that everything is working properly, you can change this number to something more convenient. On UNIX use the

webconf script to set this number. On Windows NT edit matlabserver.conf directly with a text editor.

3 Follow the directions provided by your Web server (httpd) to create the needed aliases:

   a The home or default directory

   b /cgi-bin

   c /icons

Point each of these aliases to <matlab>/toolbox/webserver/wsdemos to get the demonstration programs to work.

If your application creates graphic (jpeg) files, you need to provide a location where MATLAB can write these for httpd access, e.g, /icons. The mldir entry associated with each application in the matweb.conf file indicates the location to MATLAB.

If you do not have permission to set up or change these aliases, it will be necessary to place copies of some files in locations where the httpd can find them.

- Copy matweb (matweb.exe on Windows NT), found in <matlab>/webserver/bin/sol2 on UNIX or <matlab>/webserver/bin on Windows NT, to the directory aliased by /cgi-bin or equivalent.
- Copy matweb.conf in <matlab>/toolbox/webserver/wsdemos to the directory aliased by /cgi-bin or equivalent. .
- Copy all demo HTML files in <matlab>/toolbox/webserver/wsdemos to the directory where the httpd keeps all HTML files (often referred to as the *home* or *default* alias).

Note that when aliases are different from those provided in the demo HTML files, you will have to make the corresponding changes in those HTML files.

UNIX Post-Installation Procedures

The MATLAB Web Server installation procedure places five scripts into the <matlab>/webserver directory:

- webconf: builds matlabserver configuration file (matlabserver.conf). See Chapter 2 for a discussion of matlabserver and the matlabserver.conf file. Use this script to specify the number of simultaneous MATLABs to run, the non-default TCP/IP port, and other variables.
- webstart: stops and restarts matlabserver via calls to webdown and webboot. These three scripts must all reside in the same directory.
- webdown: stops running matlabserver.
- webboot: starts matlabserver.
- webstat: displays matlabserver status information.

Enter the command

```
script_name –h
```

at the command prompt to see detailed information about a specific script.

After completing the MATLAB and MATLAB Web Server installation process, run the webconf script to generate the matlabserver.conf file. Then run webstart to start matlabserver. Run webdown at any time to stop matlabserver execution.

Graphics Generation

MATLAB uses X Windows on UNIX to render jpeg files efficiently. Using the function wsprintjpeg, the MATLAB Web Server can generate graphics without using X Windows, but at a cost to efficiency.

If X Windows is installed on your server, it must be running before matlabserver is started. In <matlab>/webserver/rc.web.sol2 there is a section of code which can attempt to start X Windows if you first remove the comment character from this code (see below). Alternatively, you or the system administrator can start X Windows prior to running webboot.

Automatic Startup at System Boot

To start matlabserver automatically at system boot, create the following links and file while logged in as root (superuser):

```
ln –s $MATLAB/webserver/webboot /etc/webboot_TMW5
ln –s $MATLAB/webserver/webdown /etc/webdown_TMW5
```

Note: Add the –c configuration file option to `webboot` and `webdown` if the `matlabserver.conf` file is not in `<matlab>/webserver` or in the directory where the script is located.

```
cp –p $MATLAB/webserver/rc.web.sol2 /etc/init.d/webserver
```

If you want to start X Windows here, edit the file `/etc/init.d/webserver` to remove the comments from the code that starts X Windows.

Now add the edited file to the system startup by creating the symbolic link:

```
cd /etc/rc3.d
ln –s ../init.d/webserver S17webserver
```

Reboot the system to stop and restart `matlabserver`.

Windows NT Post-Installation Procedures

Deinstallation

To remove `matlabserver` from the Windows NT Registry, open a **Command Prompt** (MS-DOS) window. Enter the command sequence

```
cd <matlab>/webserver/bin
matlabserver –remove
```

Getting Started

The process of creating a MATLAB Web Server application involves the creation of:

- An HTML input document for data submission to MATLAB
- An HTML output document for display of MATLAB's computations
- A MATLAB M-file to process input data and compute results
- A test file to validate code before distributing the application over the Web

To simplify this process as much as possible, we have provided four templates that you can find in the directory `<matlab>/toolbox/webserver/wsdemos`:

- `input_template.html`
- `output_template.html`
- `mfile_template.m`
- `tmfile_template.m`

Each template provides actual code that you need to incorporate into your application plus instructions on how to modify the template where necessary. If you follow the directions in these templates, you should be able to create MATLAB Web Server applications with reasonable effort.

Additionally provided in `<matlab>/toolbox/webserver/wsdemos` is `webmagic`, a magic squares demonstration program. A magic square produces the same sum along any row, column, or either of the two main matrix diagonals. There are four files associated with `webmagic`:

- `webmagic1.html`: the `webmagic` input document
- `webmagic2.html`: the `webmagic` output document
- `webmagic.m`: the `webmagic` MATLAB M-file
- `twebmagic.m`: the `webmagic` stand-alone test file

To learn how the four templates were modified to create the `webmagic` application, we will examine templates and note the specific changes applied. If you want to look at some other applications created with these templates, see the section "Additional Application Examples" on page 1-22.

Creating Input Documents

Input Template

The file input_template.html provides the code needed to create a MATLAB Web Server input document. An abbreviated version looks like:

```
<!-- STEP 1
Choose either the NT version or the Unix version of the form tag
(depending on which platform the matweb client program will be
run):
-->
<!-- NT version: -->
<form action="/cgi-bin/matweb.exe" method="POST">
<!-- Unix version: -->
<form action="/cgi-bin/matweb" method="POST">

<!-- STEP 2
Create a hidden field naming your M-file. Replace MY_M_FILE with
the name of main MATLAB function of your application.(An HTML
input field of type "hidden" is commonly used to pass variables
to a web server. It is not displayed by the browser.)
-->
<input type="hidden" name="mlmfile" value="my_m_file">

<!-- STEP 3
Add all your other HTML form tags here. Replace
MY_INPUT_VARIABLE_1 with the name of an input variable in your
application.
-->
<p>My input variable 1: <input type="text"
name="my_input_variable_1">
<!--
Create additional input variables here.
-->

<!-- STEP 4
Create a "submit" input tag for the user to click to send the
input to your program.
-->
<p><input type="submit" name="Submit" value="Submit"></p>
```

```
<!-- STEP 5
Add the name of your main application function to the file
matweb.conf. See the matweb.conf file in the wsdemos directory
and the documentation.)
-->
```

webmagic Input

Examine the significant part of the source for `webmagic1.html`:

```
<form action="/cgi-bin/matweb.exe" method="POST">
    <input type="hidden" name="mlmfile" value="webmagic">
    <p>Magic square size (minimum is 3, maximum is 20):
    <input type="text" size="2" maxlength="2" name="msize"></p>
    <p><input type="submit" name="Submit" value="Submit"></p>
</form>
```

The line

```
<form action="/cgi-bin/matweb.exe" method="POST">
```

calls `matweb`, the entry point to the MATLAB Web Server. `matweb.exe` is the Microsoft Windows NT name of the program used by the MATLAB Web Server to extract data from HTML forms. On UNIX this program is called just `matweb`. We refer to the program as `matweb` throughout this document except when the platform distinction is important. `matweb` is described more thoroughly in the next chapter.

The next line

```
<input type="hidden" name="mlmfile" value="webmagic">
```

provides the name of the MATLAB M-file (`mlmfile`) to run. In this application the M-file is named `webmagic`.

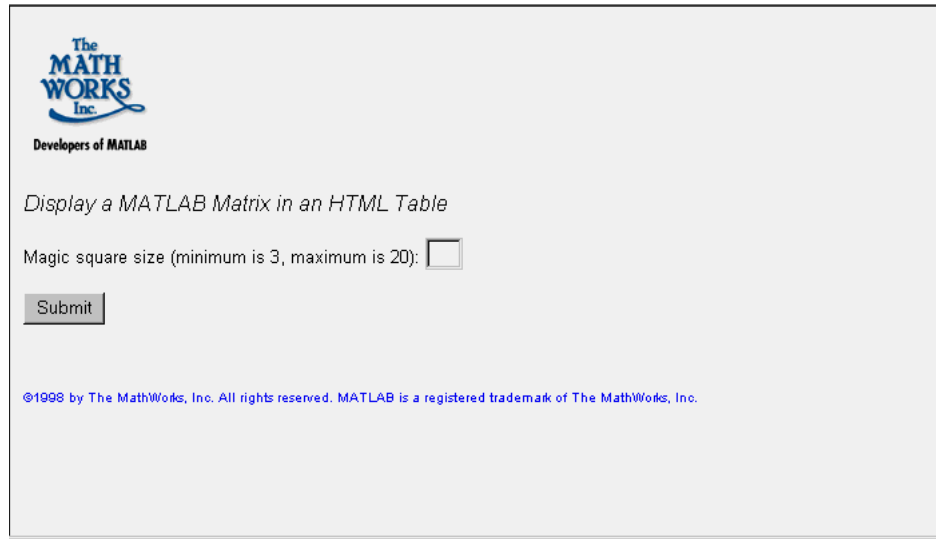
Lastly, the input

```
<input type="text" size="2" maxlength="2" name="msize"></p>
```

passes to `webmagic.m` a two-character field named `msize`, which contains the size of the magic square to compute.

To display the input document and run the magic squares demonstration locally on your computer, start your Web browser and set the URL to `http://<your_domain>/webmagic1.html`.

The magic squares input document, `webmagic1.html`, is displayed in your browser.



Enter the size of the magic square matrix you want to compute and press **Submit**.

Creating MATLAB Web Server M-Files

M-File Template

You use the M-file template to code your MATLAB application as you normally do. The template provides the additional code you need to accept input from your HTML input document and to return results to your HTML output document. An abbreviated version looks like:

```
function retstr = mfile_template(instruct, outfile)
% STEP 1
% Initialize the return string.
retstr = char('');

% STEP 2
% Set working directory.
% The variables INSTRUCT.MLDIR and INSTRUCT.MLID are provided
```

```
% automatically to all MATLAB Web Server applications that use
% the matweb program.
cd(instruct.mldir);

% STEP 3
% Get the HTML form input variables
my_input_variable_1 = instruct.my_input_variable_1;

% STEP 4
% Perform your MATLAB computations, graphics file creations,
% etc. here:

% STEP 5
% Put variables that you want to put into your HTML output
document in an output structure.  You create an HTML output
document from OUTPUT_TEMPLATE.HTML.
outstruct.my_output_variable_1 = More MATLAB computations
creating ...
   scalars, matrices, cell arrays, graphics files, etc.;

% STEP 6
% Call the function HTMLREP with the output structure you just
% created and the filename you created from OUTPUT_TEMPLATE.HTML.
% Replace <OUTPUT_TEMPLATE.HTML> with the name of the HTML output
% file you created using OUTPUT_TEMPLATE.HTML.
% This call fills the string RETSTR to return and optionally
% writes the output as a file if a valid filename is given as the
% second argument to the present function.
templatefile = which('<OUTPUT_TEMPLATE.HTML>');
if (nargin == 1)
   retstr = htmlrep(outstruct, templatefile);
elseif (nargin == 2)
   retstr = htmlrep(outstruct, templatefile, outfile);
end
```

webmagic M-File

The data entered on the `webmagic1.html` input document is automatically passed to MATLAB, which then runs the `webmagic` function. Notations in **boldface** refer to steps in the M-file template.

```
% Initialize the return string. (Step 1)
retstr = char('');
```

(Step 2. *Not needed. No generated graphics.***)**

```
% Get the msize (string) variable. Convert to a number. (Step 3)
% Check the range.
if(~length(instruct.msize))
   msize = 3; % Default empty field.
else
   msize = str2double(instruct.msize);
   if (msize > 20), msize = 20; end % Max size.
   if (msize < 3), msize = 3; end % Min square.
end
```

```
% Save size as a char string in structure OUTSTRUCT. (Step 4, 5)
outstruct.msize = msize;
```

```
% Create magic square in output structure OUTSTRUCT.
outstruct.msquare = magic(msize);
```

```
% Get column, row, and diagonal sum.  Put in OUTSTRUCT.
d = sum(outstruct.msquare,1);
outstruct.msum = d(1,1);
```

```
% Output the results and optionally write as a file if the
% filename was given as the second argument to WEBMAGIC. (Step 6)
templatefile = which('webmagic2.html');
if (nargin == 1)
   retstr = htmlrep(outstruct, templatefile);
elseif (nargin == 2)
   retstr = htmlrep(outstruct, templatefile, outfile);
end
```

Creating Output Documents

Output Template

The file `output_template.html` provides the code needed to create a MATLAB Web Server output document. An abbreviated version looks like:

```
<!--
Modify this file to create your own HTML output document
and save it as <MY_OUTPUT>.html, where <MY_OUTPUT> is replaced
by a name that has meaning within the context of your application.
-->

<!-- STEP 1
Display a MATLAB scalar or character string. Replace
<MY_OUTPUT_VARIABLE_1> in the following line with the name of the
MATLAB variable you want to display. Change the other text to
something meaningful within the context of your application.
-->
My output variable 1 has been computed to be:
$<my_output_variable_1>$

<!-- STEP 2
Put all your other HTML tags here.
-->
```

webmagic Output

The `webmagic` output document contains three variables:

`$msquare$` -- the completed magic square

`$msize$` -- the size of the magic square

`$msum$` -- the magic square sum along its rows, columns, or diagonals

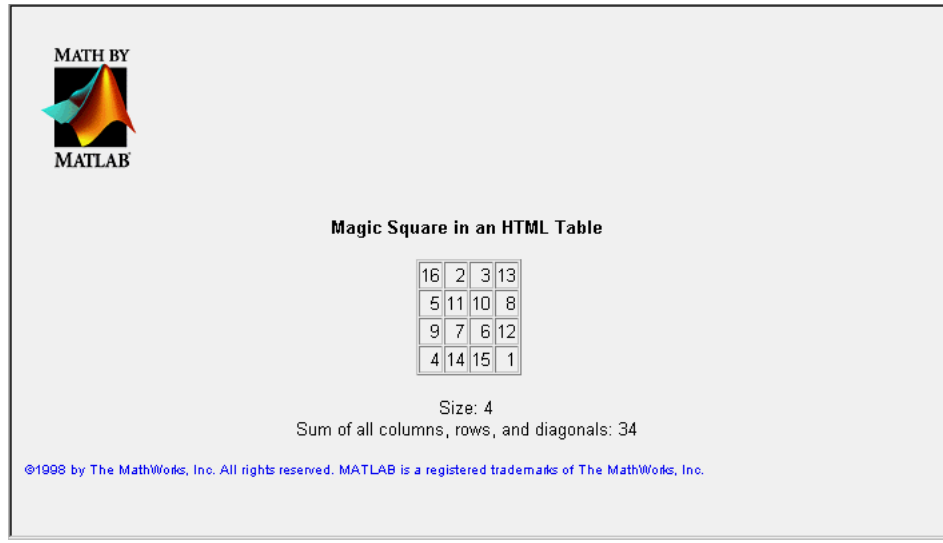
Using `htmlrep` the `webmagic` function replaces these variables with actual values, using the input obtained from `webmagic1.html`.

The source for the `webmagic2.html` template document looks like:

```
<!--
HTML output template used by webmagic.m in call to the
function HTMLREP. (HTMLREP replaces variable names delinated
by dollar signs, e.g. $msquare$, with their values. It also
```

```
generates HTML tables and select lists dynamically from matrics,
cell arrays, and vectors.)
-->

<html>
<head>
<title>Magic Square in an HTML Table</title>
</head>
<div align="center">
<strong>Magic Square in an HTML Table</strong></p>

<!--
Use the MATLAB "AUTOGENERATE" HTML attribute to generate a table
dynamically from the variable "msquare" which is a matrix (in
program webmagic.m).
-->

<table border="1" cellspacing="1" autogenerate="$msquare$">
    <tr>
        <td align="right">
        </td>
    </tr>
</table>

<!--
Replace $msize$ and $msum$ with the contents of MATLAB variables
named "msize" and "msum" respectively as computed in webmagic.m.
-->

<p>
Size: $msize$<br>
Sum of all columns, rows, and diagonals: $msum$
</div>
```

When displayed in a browser, `webmagic2.html` looks like:



Debugging Your Application

Debugging Procedure

You can use the MATLAB debugging facility plus your Web browser to debug your application before making it available to all users.

An effective method of debugging is to write your application M-file to accept two arguments, as in

```
retstr = webmagic(structure, testfile)
```

You can use the second argument to create an HTML file that displays test output.

As a first debugging step, create a driver program that sets up the input variables and calls the main function. You can use the MATLAB debugging facilities to check the logic of your test program. The file `twebmagic.m` located in `<matlab>/toolbox/webserver/wsdemos` is an example of such a driver program. (Note the use of the `wssetfield` function to create the test input

variables. This is optional. Elements of the structure `s` could be created directly, e.g., `s.msize = '5'`.)

```
function twebmagic()
%TWEBMAGIC Example standalone test of webmagic function.
%TWEBMAGIC Does setup and calls webmagic. Creates the output file,
%twebmagic.html.

% Set up input variables.
s = {};
s = wssetfield(s, 'mlmfile', 'webmagic');
s = wssetfield(s, 'msize', '5');
s = wssetfield(s, 'mldir', '.');

% Create an output test file.
str = webmagic(s, 'twebmagic.html');
```

Because the driver program calls `webmagic` with two arguments, `webmagic` writes its output to the file `twebmagic.html`. The call to `htmlrep` within the `webmagic` function handles this.

```
retstr = htmlrep(outstruct,'webmagic2.html', outfile)
```

`outfile` in this case is `twebmagic.html`, the second argument passed to `webmagic`.

The second step in debugging is to use your Web browser to examine your test file (`outfile`) and make appropriate changes until the output is displayed as you intend.

Debugging Template

To assist you in debugging, we have provided the template `tmfile_template.m`, shown below in abbreviated form:

```
function tmfile_template()

% STEP 1
% Set up input variables as they would come in from
% the HTML input form created from INPUT_TEMPLATE.HTML.
outstruct.my_input_variable_1 = some appropriate test value;

% STEP 2
% Call your application function that was created from
% <MFILE_TEMPLATE.M>. Replace <MFILE_TEMPLATE> with the
% name of your application M-file. Provide a test output
% file name for the optional argument by replacing
% <TEST_OUTPUT.HTML> with your test output HTML file name.
retstr = <MFILE_TEMPLATE>(outstruct, '<TEST_OUTPUT.HTML>');

% STEP 3
% Examine the file you supplied for <TEST_OUTPUT.HTML> in your
% web browser.
```

Additional Application Examples

The easiest way to learn how to create MATLAB applications that send and receive data over the Web is to analyze the sample programs included with your MATLAB Web Server distribution. To access these sample programs, open the file `<matlab>/toolbox/webserver/wsdemos/index.html` in your browser. By analyzing the code used to create these examples, you can expand upon them to create more complex MATLAB Web Server applications.

Data Display

The `players` demonstration function illustrates a basic use of the MATLAB Web Server to display data over the Web. The file `players.txt` contains a data base with information about The MathWorks' softball team:

PLAYER	POSITION	AVERAGE	AT BATS
Ron Berg	First Base	.337	30
John Theytaz	Second Base	.294	22
Charles Carmody	Third Base	.261	34
Debby Oldman	Shortstop	.287	29
Jack Pirrotta	Right Field	.256	27
Josh Tillson	Center Field	.301	31
Eugene Goldbrick	Left Field	.281	31
Donna Navillus	Pitcher	.222	32
Anna Alman	Catcher	.290	30

`players` uses a few MATLAB commands to read the tab-delimited file, convert the data to an HTML file, and display the output in a Web browser. No input form is needed for this application, only a URL. The URL can be entered directly in your browser or as a live link in another page. The URL for the `players` example is
`http://<your_domain>/cgi-bin/matweb.exe?mlmfile=players&`
on Windows NT. (On UNIX use `matweb` instead of `matweb.exe`.)

The output looks like:

MathWorks Softball Team Statistics

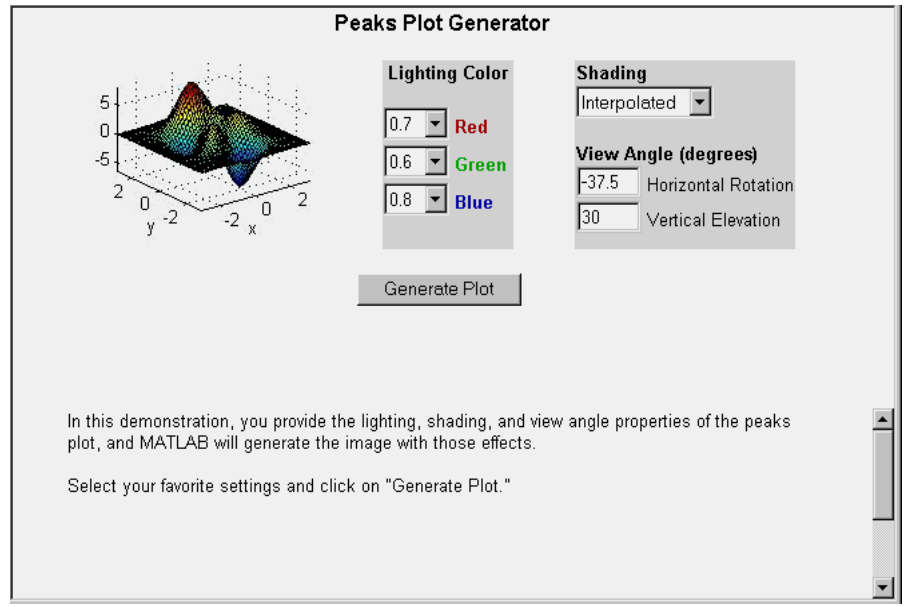
Ron Berg	First Base	.337	30
John Theytaz	Second Base	.294	22
Anna Alman	Catcher	.290	30
Debby Oldman	Shortstop	.287	29
Eugene Goldbrick	Left Field	.281	31
Charles Carmody	Third Base	.261	34
Jack Pirrotta	Right Field	.256	27
Donna Navillus	Pitcher	.242	32
Josh Tillson	Center Field	.221	31

If you would like to experiment on your own with a similar simple application, use `players` as a model to create an M-file that reads your own text file, e.g., `myfile.txt`, and places data into a MATLAB structure. To display the result in your Web browser, use the above URL, changing the value of the `mlmfile` argument to the name of your new M-file. Also, copy the entry for `players` in `matweb.conf` and change the name of the application within the brackets `[]` to the one you have chosen.

MATLAB Graphics

The `webpeaks` function, included as a demonstration program, creates a `peaks` plot and returns the output to your Web browser. In examining portions of the `webpeaks` code, you will see how to include MATLAB graphics as part of a MATLAB Web Server application.

To start the webpeaks demonstration, set the URL in your browser to `http://<your_domain>/webpeaks1.html`, the webpeaks input document.



This input document allows you to set the characteristics of the peaks plot you want to generate. This is a more complex input document than the one we used with webmagic, as it makes use of HTML frames. The code in the source file

```
<form action="/cgi-bin/matweb.exe" method="POST" target="outputwindow">
   <input type="hidden" name="mlmfile" value="webpeaks">
```

calls the webpeaks function and targets the output to a frame on the lower portion of the input document itself.

In addition to the code necessary to compute and display the peaks function, the file webpeaks.m contains additional code specific to the transmission of graphics data across the Web. In webpeaks.m the code

```
   mlid = getfield(h,'mlid')
```

extracts mlid from the structure h.

mlid is a unique identifier that matlabserver provides. Using the value of mlid to construct filenames ensures that filenames are unique. It can also be used

to maintain contexts among the different connections in an application.You can see this in the code

```
s.GraphFileName = sprintf('%speaks.jpeg',mlid)
```

which creates a name for a `jpeg` file. If `mlid` has the value `ml00277`, for example, the `jpeg` file will be named `ml00277peaks.jpeg`.

The function `htmlrep` replaces MATLAB variable names it finds in the HTML output template file `webpeaks2.html` with the values in the input structure `s`:

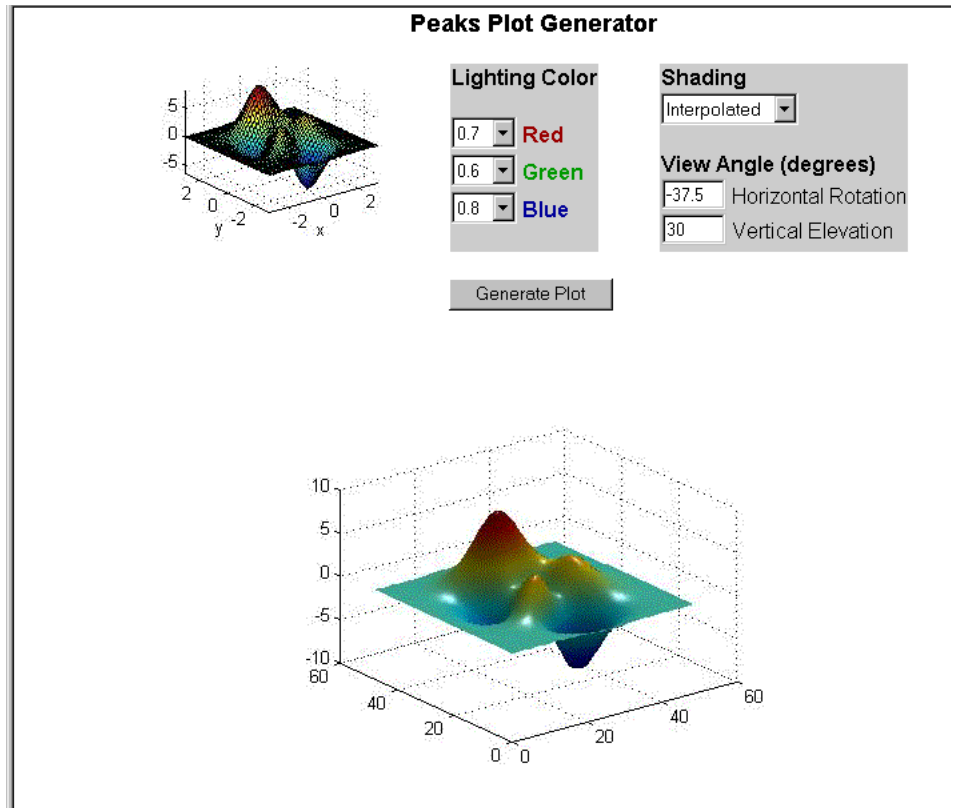
```
rs = htmlrep(s,'webpeaks2.html')
```

`$GraphFileName$`, the variable that represents the graphic output, is found in the line

```
<img border=0 src="$GraphFileName$">
```

in `webpeaks2.html`.

The final output document shows both the input and output frames:



Note: When the input form contains a "clickable" image created by an `<input>` tag of the form `<type = "image" name = "mymap">`, the variable names for the x and y coordinates are normally passed to the program as `mymap.x` and `mymap.y`. The MATLAB Web Server converts these to `mymap_x` and `mymap_y`. For example, the input
`<input type = image src = "mymap.jpeg" name = "mymap">` results in storing the x and y coordinates `mymap_x` and `mymap_y` in the structure passed to your program.

Stock Price Simulation

Now that you are familiar with the use and operation of the MATLAB Web Server, look at `webstock`, a program that uses MATLAB to simulate possible stock price scenarios based on user assumptions about estimated return and price volatility.

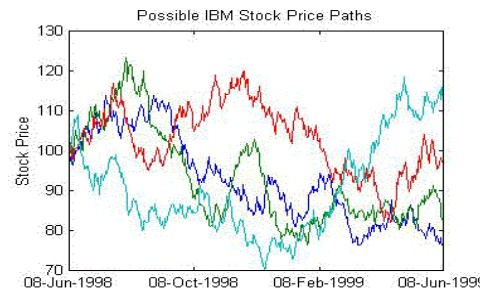
Simulation of Future Stock Prices

This is a Monte-Carlo simulation of the price of a stock over the next year. Input today's price, the expected rate of return, and the volatility of the stock. You can plot a number of possible price scenarios at once.

The prices are generated by sampling a lognormal stock price process. See, for example, N. Chriss "Black-Scholes and Beyond", Irwin, 1997. The lognormal stock price model is used in finance to value stock options.

Stock symbol: IBM
Current Stock Price: 100
Annualized Expected Return (percent): 10
Annualized Volatility (percent): 30
Number of Simulated Paths: 4

Submit



Set your browser to `http://<your_domain>/webstock1.html` to begin the simulation. When you are finished, examine the source of the input (`webstock1.html`) and output (`webstock2.html`) documents, the MATLAB M-file (`webstockrnd.m`), and the stand-alone test M-file `twebstockrnd.m`.

2

Inside the MATLAB Web Server

MATLAB Web Server Components

The MATLAB Web Server consists of a set of programs that enable MATLAB programmers to create MATLAB applications and access them on the Web:

- `matlabserver`: manages the communication between the Web application and MATLAB.

  `matlabserver` is a multithreaded TCP/IP server. It runs the MATLAB program (M-file) specified in a hidden field named `mlmfile` contained in the HTML document. `matlabserver` invokes `matweb.m` , which in turn runs the M-file.

  `matlabserver` can be configured to listen on any legal TCP/IP port by editing the `matlabserver.conf` file on Windows NT or running `webconf` on UNIX. The number of simultaneous MATLABs is specified here.
- `matweb`: a TCP/IP client of `matlabserver`. This program uses the Common Gateway Interface (CGI) to extract data from HTML documents and transfer it to `matlabserver`.
- `matweb.m`: calls the M-file that you want the Web application to run.
- `matweb.conf`: a configuration file that `matweb` needs for connecting to `matlabserver`. Applications must be listed in `matweb.conf`.

A diagram showing how MATLAB operates over the Web looks like:

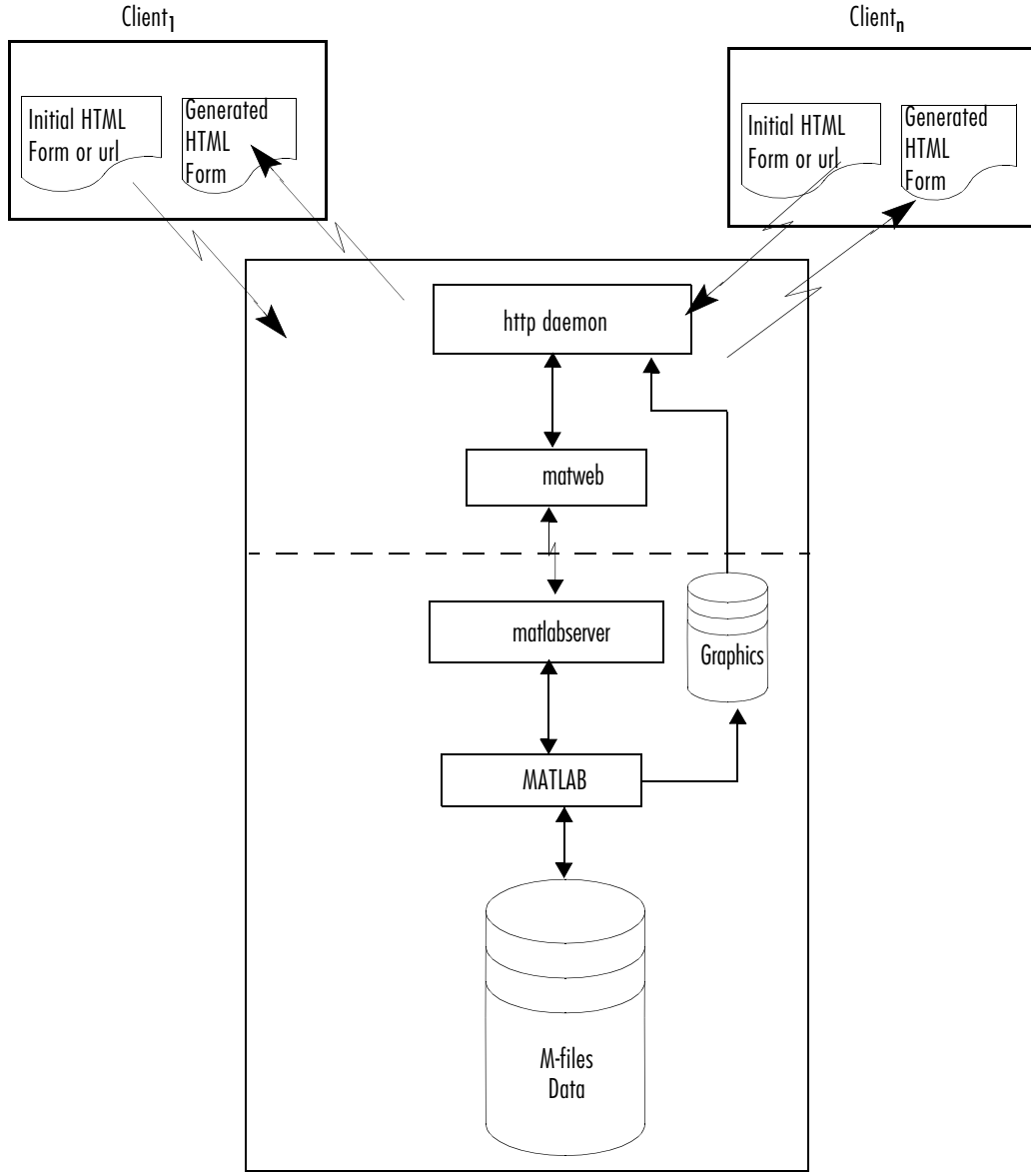


Figure 2-1: MATLAB on the Web

File Locations

Any M-files used in conjunction with a Web application, including `matweb.m`, must appear on the MATLAB path. The `matweb` and `matweb.conf` files must appear under a `/cgi-bin` alias. Any generated graphics must be located where the Web Server can find them and programs can write them.

Understanding matlabserver

matlabserver is designed to run continuously in the background as a Windows NT service or a UNIX background process. (Administrator privileges are normally required to install matlabserver.) For testing you can turn on terminal logging by entering the command matlabserver –t at a Windows NT command prompt (webstart –t on UNIX) when starting matlabserver. (See Appendix B.)

matlabserver.conf

When matlabserver starts up, it looks in the file matlabserver.conf for its initial setting data. On Windows NT the installation procedure creates this file in the <matlab>/webserver directory while installing the MATLAB Web Server. On UNIX systems you must first run the webconf script to establish values for some of the arguments in the file before starting the MATLAB Web Server.

Configuration settings must appear on the first line of the matlabserver.conf file. The basic options set by the data in matlabserver.conf are:

- Port number: p
- Threads (maximum number of simultaneous MATLABs): m

The default version of matlabserver.conf file is simply:

```
–m 1
```

meaning that you want to run one copy of MATLAB with the matlabserver port defaulted to 8888.

Edit the default version of matlabserver.conf on Windows NT or run the webconf script on UNIX if you want to change some of the options.

Table 2-1: matlabserver Basic Options

Option	Meaning
–p [*n*]	Port that matlabserver listens on. 8888 is the default.
–m [*n*]	MATLABs to run. Default is 1.

Note that a space is required between the flag and its value.

Type `matlabserver –h` on Windows NT (`webconf –h` on UNIX) for a list of additional editable `matlabserver.conf` options.

If `matlabserver` cannot locate a `matlabserver.conf file`, it uses the defaults.

Using matlabserver

In Chapter 1, when we viewed the source of the `webmagic1.html` file, we observed the line of HTML code

```
<FORM ACTION="/cgi–bin/matweb.exe" METHOD="POST">
```

and we noted that this line establishes communication with MATLAB. `matweb` is a program that resides on the HTTP server and communicates with `matlabserver`. `matweb` requires information found in `matweb.conf` to locate `matlabserver` (which could be running on a different machine).

matweb Program

`matweb` is a client of `matlabserver` that uses Common Gateway Interface (CGI) to get data from HTML forms. It transfers the information to `matlabserver`, which then runs applications written in M-files to produce responses.

When the MATLAB Web Server is installed, `matweb` is placed in `<matlab>/webserver/bin/arch` on UNIX and in `<matlab>/webserver/bin` on Windows NT. For HTTP server access you must also place a copy of `matweb` in the directory denoted by the `/cgi-bin` alias. The installation process places a copy in `<matlab>/toolbox/webserver/wsdemos` to run the sample applications.

matweb.conf

To connect with `matlabserver`, `matweb` requires information stored in the configuration file `matweb.conf`. Create this file inside the directory denoted by `/cgi-bin`, along with the `matweb` program. Use the copy in `<matlab>/toolbox/webserver/wsdemos` as a guide.

An instance of `matweb.conf` looks like

```
[webmagic]
mlserver=parrot
mldir=/matlab/toolbox/websever/wsdemos

[webpeaks]
mlserver=parrot
mldir=/matlab/toolbox/websever/wsdemos
```

Multiple application configurations must appear in the same file. Each variable appears on a separate line followed by an equal sign =, which is then followed by a value, e.g., `mlserver=parrot`. Applications are delineated by the main application entry point name (M-file) in square brackets `[]`. For example, `[webmagic]` is on one line followed by all its variables and corresponding values. All the fields that can be contained in `matweb.conf` are described in Table 2-2.

Table 2-2: matweb.conf Fields

Variable	Description	Sample Value
`[application]` (required)	Name of the MATLAB application to run	`webmagic`
`mldir` (optional)	Working directory for reading or writing files. If specified, this directory is automatically added to the MATLAB path.	`<matlab>/toolbox/webserver/wsdemos`
`mllog` (optional)	Produces an application-specific log file that records all exchanges between the application and MATLAB. Turn off logging when the program is running because logging has a negative impact on performance.	`<matlab>/toolbox/webserver/wsdemos/webmagic.log`
`mlserver` (required)	Name of host running `matlabserver`	`parrot`

Table 2-2: matweb.conf Fields (Continued)

Variable	Description	Sample Value
`mlport` (optional)	Port that `matlabserver` listens on. This value must correspond to the port number set in the `matlabserver.conf` file or on the command line (the `p` argument).	`8888` (default)
`mltimeout` (optional)	Seconds to wait for `matlabserver` before timing out	`180` (default)

After you create a new MATLAB Web Server application and enter its configuration data into `matweb.conf`, you will need to restart `matlabserver` before you can use the application.

matweb M-File

Looking again at the source from the `webmagic1.html` file in Chapter 1, observe that the line

```
<input type="hidden" name="mlmfile" value="webmagic">
```

sets argument `mlmfile` to the value `webmagic`. The `mlmfile` argument contains the name of the MATLAB M-file to run.

`matlabserver` uses the value of `mlmfile` obtained from the `matweb` M-file, `matweb.m`, (`webmagic` in this example) to run the MATLAB application. `webmagic` takes the input data from `webmagic1.html`, computes the magic square of the requested dimensions, and outputs the results using `webmagic2.html` as a template.

Returning Results via the Web

The MATLAB Web Server distribution kit contains the file `webmagic2.html`, which serves as an example of an HTML output document template. The `webmagic` function uses the `htmlrep` command to place the computed values into the `webmagic2.html` output template using the code:

```
str = htmlrep(s,'webmagic2.html');
```

In this example `s` is a MATLAB structure containing the results of the `webmagic` magic squares computation. `htmlrep` extracts data from `s` and

replaces variable fields in `webmagic2.html` with the results of MATLAB computation. The completed `webmagic2.html` form is transmitted to the user's browser.

3

Reference

This chapter provides detailed descriptions of the functions in the MATLAB Web Server.

Function Summary

Function	Purpose
`htmlrep`	Replace variable names with values in HTML document.
`matweb`	MATLAB Web Server main entry point.
`wscleanup`	Purge stale files from directory.
`wsprintjpeg`	Create JPEG file.
`wssetfield`	Add new field or append to existing field.

Purpose

Substitue values for variable names in HTML document

Syntax

```
outstring = htmlrep(instruct,infile)
outstring = htmlrep(instruct,infile,outfile)
outstring = htmlrep(instruct,infile,outfile,'noheader')
```

Description

`htmlrep(instruct,infile)` replaces all MATLAB variables in `infile`, an HTML document, with corresponding values of variables of the same name in `instruct`. Variables can be character strings, matrices, or cell arrays containing strings and scalars. String and scalar variables are replaced by straight substitution. Output is returned in `outstring`. Variable names in `infile` must be enclosed in dollar signs, e.g., `$varname$`.

`instruct` is a MATLAB structure containing variable names (field names) and corresponding values.

`infile` is an HTML template file with MATLAB variable names enclosed in dollar signs.

`outstring = htmlrep(instruct,infile,outfile)` additionally writes output to the HTML document `outfile` (for stand-alone testing).

`outstring = htmlrep(instruct,infile,outfile,'noheader')` additionally suppresses the output of the default HTTP header
`'Content-type: text/html\n\n'`
to `outfile` and `outstring`.

HTML tables and select lists can be generated dynamically from matrices or cell arrays containing strings and scalars.

1. Tables can be generated using the special MATLAB `AUTOGENERATE` HTML table attribute with the matrix or cell array name as the value. For example, the following code automatically generates all the HTML needed to display the the entire matrix, `msquare`, in an HTML table.

   ```
   <TABLE BORDER="1" CELLSPACING="1" AUTOGENERATE="$msquare$">
      <TR>
          <TD ALIGN="RIGHT">
          </TD>
      </TR>
   </TABLE>
   ```

At least one of each of the tags listed above is required.

`htmlrep` uses the HTML code from the `<TABLE>` tag to the `</TABLE>` tag as a template for generating the entire table. If different column attributes are required, additional pairs of cell tags (`<TD>` and `</TD>`) can be included up to the number of columns in the matrix or cell array. For example, adding these tags

```
<TD ALIGN="CENTER">
</TD>
```

after the `</TD>` tag above causes the second column to be center-justified.

If there are more columns in the matrix or cell array than `<TD>` `</TD>` pairs, the last pair is used for all subsequent columns.

2 `SELECT` lists are generated using the special MATLAB `AUTOGENERATE` HTML `SELECT` attribute with the vector, matrix or cell array name as the value. For example, the following code automatically generates all the HTML needed to display the the entire vector, `mylist`, in an HTML `SELECT` list. (`SELECT` lists must appear inside HTML `<FORM>` and `</FORM>` tags.)

```
<SELECT NAME="NAMELIST" SIZE=10 AUTOGENERATE=$mylist$ MULTIPLE>
<OPTION SIZE=6>
</SELECT>
```

`htmlrep` uses the HTML code from the `<SELECT>` tag to the `</SELECT>` tag as a template for generating the entire `SELECT` list. One of each of the tags shown above is required.

If `mylist` is a matrix or cell array, `htmlrep` uses only the the first column vector to construct the select list.

Purpose

MATLAB Web Server main entry point

Syntax

```
matweb(instruct)
```

Description

`matweb` is an M-file that in turn calls a MATLAB application M-file stored in the `mlmfile` field of MATLAB structure `instruct`. It also passes `instruct` to the application. The `matweb` function (M-file) is invoked by `matlabserver`. `instruct` contains the fields:

- All the data from the HTML input document
- `mlmfile`, which stores the name of the M-file to call
- `mldir`, the working directory specified in `matweb.conf`
- `mlid`, the unique identifier for creating filenames and maintaining contexts.

If a MATLAB warning or error is encountered, the text is captured and returned to the user's browser. You can disable error and warning notification if you want.

See Also

`eval` , `lasterr`, `lastwarn`, and `warning` in the online MATLAB Function Reference.

wscleanup

Purpose

Purge stale files from directory

Syntax

```
deletecount = wscleanup(filespec,timewindow,direc)
deletecount = wscleanup(filespec,timewindow)
```

Description

`deletecount = wscleanup(filespec,timewindow,direc)` deletes all files matching filespec in the directory direc that are older than the number of hours specified in timewindow. deletecount is the number of files actually deleted.

`deletecount = wscleanup(filespec,timewindow)` deletes all files matching `filespec` in the current default directory that are older than the number of hours specified in `timewindow`. `deletecount` is the number of files actually deleted.

Purpose

Create JPEG file

Syntax

```
status = wsprintjpeg(fig, jpegfilename)
```

Description

`status = wsprintjpeg(fig, jpegfilename)` creates a JPEG file called *`jpegfilename`*. `wsprintjpeg` attempts to create the JPEG file using the MATLAB print command with the `–djpeg` argument. If this fails, it creates a temporary PCX file and then calls `imread` and `imwrite` to create the JPEG output.

See Also

`imread`, `imwrite`, `print`

wssetfield

Purpose

Add new field or append to existing field

Syntax

```
s = wssetfield(s,name1,value1,...)
```

Description

`s = wssetfield(s,name1,value1,...)`sets the contents of the field `name1` to `value1` and returns the result in the changed structure `s`. A single value is stored as a character array. Items with multiple values have the values stored in a cell array of strings. Multiple calls serve to add values to an existing field.

Either use the MATLAB `getfield` function to retrieve the values or reference the structure fields directly.

See Also

`getfield` in the online MATLAB Function Reference.

A

Directory Structure

Directory Structure

MATLAB is distributed in compressed format on CD-ROM. The installation procedure moves the files to your hard disk, decompresses them, and installs them into your MATLAB root directory. After installation of the MATLAB Web Server, your MATLAB directory should include these additional files and subdirectories:

Table A-1: <matlab>/webserver

File	Purpose
`matlabsever.conf`	`matlabserver` options
`webbboot`	Start `matlabserver` script (UNIX)
`webconf`	`matlabserver` configuration file script (UNIX)
`webdown`	Stop `matlabserver` script (UNIX)
`webstart`	`matlabserver` restart script (UNIX)
`webstat`	Report `matlabserver` status (UNIX)

Table A-2: <matlab/webserver/bin/arch (Solaris)

File	Purpose
`matlabserver`	The MATLAB Web Server binary
`matweb`	MATLAB TCP/IP client of `matlabserver`

Table A-3: <matlab>/webserver/bin (Windows NT)

File	Purpose
`matlabserver.exe`	The MATLAB Web Server binary
`matweb.exe`	MATLAB TCP/IP client of `matlabserver`

You must also place a copy of `matweb.exe` (`matweb` on UNIX) and `matweb.conf` into the `/cgi-bin` aliased subdirectory of the HTTP server root directory.

Table A-4: <matlab>/toolbox/webserver/webserver

File	Purpose
`htmlrep.mexsol` (Solaris) `htmlrep.dll` (Windows NT)	Replace variable names with values in HTML document
`matweb.m`	Web Server main entry point
`wscleanup.m`	Purge stale files from directory
`wsprintjpeg`	Create JPEG file
`wssetfield.m`	Add new field or append to existing field

Table A-5: <matlab>/toolbox/webserver/wsdemos

File	Purpose
`dummy.html`	Temporary HTML document in bottom frame of `webpeaks1.html`
`index.html`	List of demos
`input_template.html`	HTML input template
`matweb.conf`	Sample `matweb.conf` file
`mfile_template.m`	M-file creation template
`output_template.html`	HTML output template
`peaksplot.html`	HTML document (input form) in top frame of `webpeaks1.html`
`players.html`	Softball players HTML output form
`players.m`	Softball statistics file
`players.txt`	Softball text data file

Table A-5: <matlab>/toolbox/webserver/wsdemos (Continued)

tplayers.m	Stand-alone test driver for players
tmfile_template.m	Test file template
thtmlrep.m	Test of htmlrep function
thtmlrep1.html	HTML input form
webmagic.m	Convert magic square into HTML table
webmagic1.html	Magic square input form
webmagic2.html	Magic square output template
twebmagic.m	Example stand-alone test of webmagic function
webpeaks.m	Web peaks plot
webpeaks1.html	HTML frame
webpeaks2.html	peaks plot output form
webstock.html	Stock price simulation input form
webstock1.html	Stock price simulation main frame
webstock2.html	Stock price simulation output template
webstockrnd.m	Stock future price path simulation
webstocktemp.html	HTML output place holder
twebstockrnd.m	Stand-alone test driver for webstockrnd
wstextread.m	Place a delimited file into a cell array of strings

B

Administration and Troubleshooting

Troubleshooting matlabserver

matlabserver provides an additional set of configuration options that are useful in troubleshooting matlabserver operations. These options may be set in the matlabserver.conf file or specified on the command line. Any option you specify on the command line overrides the value found in matlabserver.conf.

Use the matlabserver command on Windows NT and the webconf and webstart scripts on UNIX to set these options.

Table B-1: matlabserver Troubleshooting Configuration Options

Option	Purpose
–f [*log_filename*]	Event log file. Required when logging to a file. File content determined by –v setting. (File logging is inefficient. Use when debugging only.)
–l [*errorlog_filename*]	(UNIX only). Error log file. Default is matlabserver_error.log in current directory. On Windows NT use the Event Viewer to see a list of error events.
–t	Terminal logging. The logging level is set to 2 (transaction and buffer logging). (If you use this option with the matlabserver command on Windows NT, it must precede all other options on the command line.)
–v [*n*]	Verbosity. Controls file logging. Default is 0 (no logging). Additional values are 1 (transaction logging) and 2 (transaction and buffer logging).

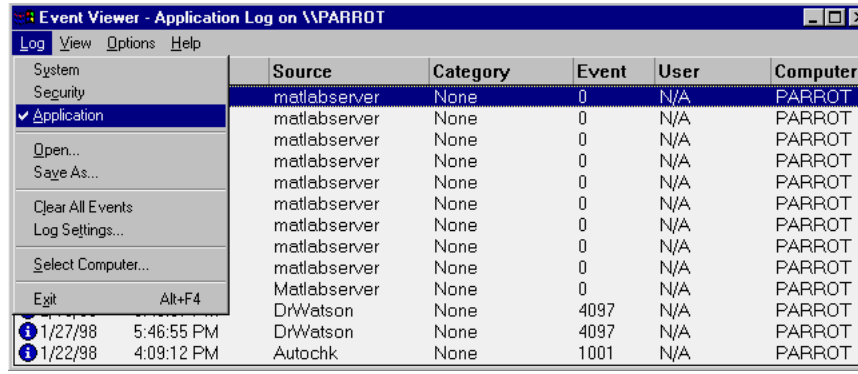
Network bind Error (Port in Use)

If matlabserver fails to start because of a bind error, as noted in matlabserver_error.log on UNIX or in the **Event Viewer** on Windows NT, the the port you are attempting to run matlabserver on is busy. To fix this

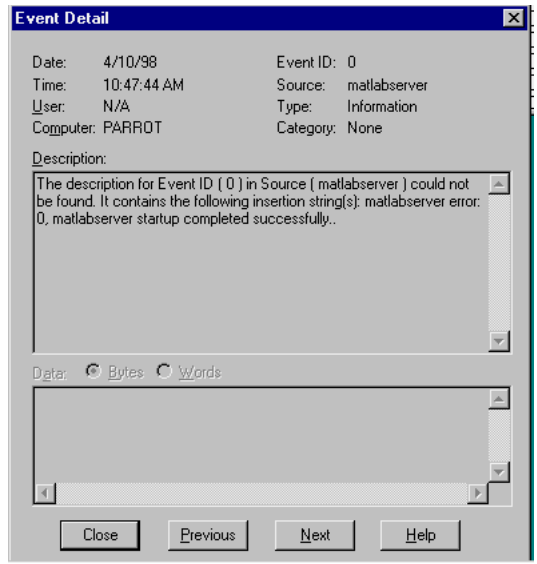
problem, you need to change the port number that matlabserver listens on. See the section “matlabserver.conf” in Chapter 2 for a discussion on how to change the port number. You may need to ask your system administrator to provide you with a valid unused port number.

Additional Troubleshooting for Windows NT

The Windows NT Event Viewer captures data that can be useful for debugging `matlabserver` operations even if you have not requested `matlabserver` logging through the command options. To access the Event Viewer, choose **Start ->Programs ->Administrative Tools ->Event Viewer**. When the Event Viewer appears, click on **Log** on the menu bar and choose **Application** from the pulldown menu.



Double-click on a `matlabserver` entry to receive additional detail that may be useful in determining the cause of a `matlabserver` problem.



Additional Troubleshooting for Solaris

Error Logging

For error logging information, look in `matlabserver_error.log` in `<matlab>/webserver` or in the file specified with the `–l` option in the `matlabserver.conf` file.

Creating New Applications

After you create a new MATLAB Web Server application and enter its configuration data into `matweb.conf`, you will need to restart `matlabserver` before you can use the application.

Locating matweb.conf

File Location

In some network configurations it is not possible to give programmers access to the `/cgi-bin` directory of the HTTP server. In such cases a `matweb.conf` file should be created with only one entry, containing the actual location of a configuration file that programmers can edit. This entry must appear inside angle brackets `< >`. An example of this type of `matweb.conf` file is:

```
</apps/projects/strategy/matweb.conf>
```

where `/apps/projects/strategy/matweb.conf` is a valid accessible file.

Connect() failure Error

There are two probable reasons why you may receive a `Error: connect() failure` message:

1. `matlabserver` is not running.

   On Unix run the `webstat` script. On Windows NT click the **Start** button, open the **Control Panel**, and choose **Services**. The status of `MATLAB Server` should be `Started`.

2. Port mismatch between `matlabserver.conf` and `matweb.conf`.

   The default TCP/IP port that `matlabserver` listens on is set at `8888`. You may change this setting in the `matlabserver.conf` file with the `–p` option. (See “matlabserver.conf” on page 2-5.) The port setting for each application configuration in the `matweb.conf` file must agree with the port setting in `matlabserver.conf`. (See “Table 2-2: matweb.conf Fields”.) If you have changed the port setting in `matlabserver.conf`, you must similarly change the port setting in `matweb.conf` using the `mlport` option. If `mlport` is not explicitly set, the default of `8888` is assumed.

M-File Programming Considerations

Make certain that each line of your M-file application is terminated with a `;` character. Otherwise, the HTML output will be corrupted.

C

References

References in Print and on the Web

Numerous books have been published about HTML programming and the World Wide Web. Two that we have found both useful and readable are:

Gundavaram, Shishir, *CGI Programing on the World Wide Web*, Sebastopol, CA, O'Reilly & Associates, Inc., 1996.

Musciano, Chuck and Bill Kennedy, *HTML The Definitive Guide*, Sebastopol, CA, O'Reilly & Associates, Inc., 1996.

On the Web, the World Wide Web Consortium (W3C) publishes comprehensive information about current and future directions of the Web and the HTML language. You will find their home page at `http://www.w3.org`. Look at `http://www.w3.org/MarkUp` for a comprehensive discussion of HTML.

Index

N

O

P

R

S

T

W