save

Save workspace variables on disk

Graphical Interface

As an alternative to the `save` function, select **Save Workspace As** from the **File** menu in the MATLAB desktop, or use the Workspace browser.

Syntax

```
save
save('filename')
save('filename', 'var1', 'var2', ...)
save('filename', '-struct', 's')
save('filename', '-struct', 's', 'f1', 'f2', ...)
save('-regexp', expr1, expr2, ...)
save('..., 'format')
save filename var1 var2 ...
```

Description

`save` by itself stores all workspace variables in a binary format in the current directory in a file named `matlab.mat`. Retrieve the data with `load`. MAT−files are double−precision, binary, MATLAB format files. They can be created on one machine and later read by MATLAB on another machine with a different floating−point format, retaining as much accuracy and range as the different formats allow. They can also be manipulated by other programs external to MATLAB.

`save('filename')` stores all workspace variables in the current directory in `filename.mat`. To save to another directory, use the full pathname for the `filename`. If `filename` is the special string `stdio`, the `save` command sends the data as standard output.

`save('filename', 'var1', 'var2', ...)` saves only the specified workspace variables in `filename.mat`. Use the `*` wildcard to save only those variables that match the specified pattern. For example, `save('A*')` saves all variables that start with `A`.

`save('filename', '-struct', 's')` saves all fields of the scalar structure `s` as individual variables within the file `filename`.

`save('filename', '-struct', 's', 'f1', 'f2', ...)` saves as individual variables only those structure fields specified (`s.f1`, `s.f2`, `...`).

`save('-regexp', expr1, expr2, ...)` saves those variables that match any of the regular expressions `expr1`, `expr2`, etc.

`save(..., 'format')` enables you to make use of other data formats available with the `save` function. See the following table.

Format	How Data Is Stored

`-append`	The specified existing MAT-file, appended to the end. See Remarks, below.
`-ascii`	8-digit ASCII format
`-ascii -double`	16-digit ASCII format
`-ascii -tabs`	Delimits with tabs
`-ascii -double -tabs`	16-digit ASCII format, tab delimited
`-mat`	Binary MAT-file form (default)
`-v4`	A format that MATLAB Version 4 can open
`-v6`	A format that MATLAB Version 6 and earlier can open

`save filename var1 var2 ...` is the command form of the syntax.

Remarks

By default, MATLAB compresses the data it saves to MAT-files. MATLAB also uses Unicode character encoding when saving character data. Specify the `-v6` option if you want to disable both of these features for a particular `save` operation. If you save data to a MAT-file that you intend to load using MATLAB Version 6 or earlier, then you must specify the `-v6` option when saving.

To override the compression and Unicode setting for all of your MATLAB sessions, use the **Preferences** dialog box. Open the **Preferences** dialog and select **General** and then **MAT-Files**. To disable data compression and Unicode encoding, click **Ensure backward compatibility (-v6)**. To turn these features back on, click **Use default features (Unicode and compression)**. See General Preferences for MATLAB in the Desktop Tools and Development Environment documentation for more information.

For information on any of the following topics related to saving to MAT-files, see Exporting Data to MAT-Files in the "MATLAB Programming" documentation:

- Appending variables to an existing MAT-file
- Compressing data in the MAT-file
- Saving in ASCII format
- Saving in MATLAB Version 4 format
- Saving with Unicode character encoding
- Data storage requirements
- Saving from external programs

For information on saving figures, see the documentation for `hgsave` and `saveas`. For information on exporting figures to other graphics formats, see the documentation for `print`.

Examples

Example 1

Save all variables from the workspace in binary MAT-file `test.mat`:

```
save test.mat
```

Example 2

Save variables `p` and `q` in binary MAT−file `test.mat`:

```
savefile = 'test.mat';
p = rand(1, 10);
q = ones(10);
save(savefile, 'p', 'q')
```

Example 3

Save the variables `vol` and `temp` in ASCII format to a file named `june10`:

```
save('d:\mymfiles\june10','vol','temp','-ASCII')
```

Example 4

Save the fields of structure `s1` as individual variables rather than as an entire structure.

```
s1.a = 12.7;  s1.b = {'abc', [4 5; 6 7]};  s1.c = 'Hello!';
save newstruct.mat -struct s1;
clear
```

Check what was saved to `newstruct.mat`:

```
whos -file newstruct.mat
  Name      Size                    Bytes  Class

  a         1x1                         8  double array
  b         1x2                       158  cell array
  c         1x6                        12  char array

Grand total is 16 elements using 178 bytes
```

Read only the `b` field into the MATLAB workspace.

```
str = load('newstruct.mat', 'b')
str =
    b: {'abc'  [2x2 double]}
```

Example 5

Using regular expressions, save in MAT−file `mydata.mat` those variables with names that begin with `Mon`, `Tue`, or `Wed`:

```
save('mydata', '-regexp', '^Mon|^Tue|^Wed');
```

Here is another way of doing the same thing. In this case, there are three separate expression arguments:

```
save('mydata', '-regexp', '^Mon', '^Tue', '^Wed');
```

Example 6

Save a 3000−by−3000 matrix uncompressed to file `c1.mat`, and compressed to file `c2.mat`. The compressed file uses about one quarter the disk space required to store the uncompressed data:

```
x = ones(3000);
y = uint32(rand(3000) * 100);

save c1 x y
save c2 x y −compress

d1 = dir('c1.mat');
d2 = dir('c2.mat');

d1.bytes
ans =
    45000240          % Size of the uncompressed data
d2.bytes
ans =
    11985634          % Size of the compressed data

d2.bytes/d1.bytes
ans =
    0.2663            % Ratio of compressed to uncompressed
```

See Also

load, clear, diary, fprintf, fwrite, genvarname, who, whos, workspace