Octave FAQ

Frequently asked questions about Octave
December 1, 1994

John W. Eaton

This is a list of frequently asked questions (FAQ) for Octave users.

Some information in this FAQ was developed for earlier versions of Octave and may now be obsolete.

I'm looking for new questions (*with* answers), better answers, or both. Please send suggestions to bug-octave@che.utexas.edu. If you have general questions about Octave, or need help for something that is not covered by the FAQ, please use the help-octave@che.utexas.edu mailing list.

This FAQ is intended to supplement, not replace, the Octave manual. Before posting a question to the hlpe-octave mailing list, you should first check to see if the topic is covered in the manual.

1 What is Octave?

Octave is a high-level interactive language, primarily intended for numerical computations that is mostly compatible with MATLAB.[1]

Octave can do arithmetic for real and complex scalars and matrices, solve sets of nonlinear algebraic equations, integrate functions over finite and infinite intervals, and integrate systems of ordinary differential and differential-algebraic equations.

Octave uses the GNU readline library to handle reading and editing input. By default, the line editing commands are similar to the cursor movement commands used by GNU Emacs, and a vi-style line editing interface is also available. At the end of each session, the command history is saved, so that commands entered during previous sessions are not lost.

The Octave distribution includes a 200+ page Texinfo manual. Access to the complete text of the manual is available via the help command at the Octave prompt.

Two and three dimensional plotting is fully supported using gnuplot.

The underlying numerical solvers are currently standard Fortran ones like Lapack, Linpack, Odepack, the Blas, etc., packaged in a library of C++ classes. If possible, the Fortran subroutines are compiled with the system's Fortran compiler, and called directly from the C++ functions. If that's not possible, you can still compile Octave if you have the free Fortran to C translator f2c.

Octave is also free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation.

2 What's new in version 1.1.0 of Octave

The long-awaited version 1.1.0 of Octave has now been released. Many bugs have been fixed and lots of new features added. Octave is now much more compatible with MATLAB.

---

[1] MATLAB is a registered trademark of The MathWorks, Inc.

Version 1.1.0 fixes many bugs, but as with any "x.y.0" release there will be a few glitches. You can expect a 1.1.1 shortly. You can help contribute to the quality of Octave by trying it out and submitting bug reports where you find them.

A list of user-visible changes in recent versions of Octave may be found in the file NEWS, distributed in both source and binary releases of Octave.

3 What features are unique to Octave?

3.1 Command and variable name completion

Typing a TAB character (ASCII code 9) on the command line causes Octave to attempt to complete variable, function, and file names. Octave uses the text before the cursor as the initial portion of the name to complete.

For example, if you type '`fu`' followed by TAB at the Octave prompt, Octave will complete the rest of the name '`function`' on the command line (unless you have other variables or functions defined that begin with the characters '`fu`'). If there is more than one possible completion, Octave will ring the terminal bell to let you know that your initial sequence of characters is not enough to specify a unique name. To complete the name, you may either edit the initial character sequence (usually adding more characters until completion is possible) or type another TAB to cause Octave to display the list of possible completions.

3.2 Command history

When running interactively, Octave saves the commands you type in an internal buffer so that you can recall and edit them. Emacs and vi editing modes are available with Emacs keybindings enabled by default.

When Octave exits, the current command history is saved to the file '`~/.octave_hist`', and each time Octave starts, it inserts the contents of the '`~/.octave_hist`' file in the history list so that it is easy to begin working where you left off.

3.3 Data structures

Octave includes a limited amount of support for organizing data in structures. The current implementation uses an associative array with indices limited to strings, but the syntax is more like C-style structures. Here are some examples of using data structures in Octave.

Elements of structures can be of any value type.

```
octave:1> x.a = 1; x.b = [1, 2; 3, 4]; x.c = "string";
octave:2> x.a
x.a = 1
octave:3> x.b
```

```
x.b =

  1  2
  3  4

octave:4> x.c
x.c = string
```

Structures may be copied.

```
octave:1> y = x
y =

<structure: a b c>
```

Structure elements may reference other structures.

```
octave:1> x.b.d = 3
x.b.d = 3
octave:2> x.b
x.b =

<structure: d>

octave:3> x.b.d
x.b.d = 3
```

Functions can return structures.

```
octave:1> function y = f (x)
> y.re = real (x);
> y.im = imag (x);
> endfunction

octave:2> f (rand + rand*I);
ans =

<structure: im re>

octave:3> ans.im, ans.re
ans.im = 0.93411
ans.re = 0.56234
```

Function return lists can include structure elements, and they may be indexed like any other variable.

```
octave:1> [x.u, x.s(2:3,2:3), x.v] = svd ([1, 2; 3, 4])
x.u =

  -0.40455  -0.91451
  -0.91451   0.40455

x.s =
```

```
  0.00000  0.00000  0.00000
  0.00000  5.46499  0.00000
  0.00000  0.00000  0.36597

x.v =

  -0.57605   0.81742
  -0.81742  -0.57605

octave:8> x
x =

<structure: s u v>
```

You can also use the function `is_struct` to determine whether a given value is a data structure. For example

```
is_struct (x)
```

returns 1 if the value of the variable *x* is a data structure.

This feature should be considered experimental, but you should expect it to work. Suggestions for ways to improve it are welcome.

3.4 Short-circuit boolean operators

Octave's '`&&`' and '`||`' logical operators are evaluated in a short-circuit fashion (like the corresponding operators in the C language) and work differently than the element by element operators '`&`' and '`|`'.

3.5 Increment and decrement operators

Octave includes the C-like increment and decrement operators '`++`' and '`--`' in both their prefix and postfix forms.

For example, to pre-increment the variable *x*, you would write `++`*x*. This would add one to *x* and then return the new value of *x* as the result of the expression. It is exactly the same as the expression *x* `=` *x* `+ 1`.

To post-increment a variable *x*, you would write *x*`++`. This adds one to the variable *x*, but returns the value that *x* had prior to incrementing it. For example, if *x* is equal to 2, the result of the expression *x*`++` is 2, and the new value of *x* is 3.

For matrix and vector arguments, the increment and decrement operators work on each element of the operand.

It is not currently possible to increment index expressions. For example, you might expect that the expression *v*`(4)++` would increment the fourth element of the vector *v*, but instead it results in a parse error. This problem may be fixed in a future release of Octave.

3.6 Unwind-protect

Octave supports a limited form of exception handling modelled after the unwind-protect form of Lisp. The general form of an `unwind_protect` block looks like this:

```
unwind_protect
  body
unwind_protect_cleanup
  cleanup
end_unwind_protect
```

Where *body* and *cleanup* are both optional and may contain any Octave expressions or commands. The statements in *cleanup* are guaranteed to be executed regardless of how control exits *body*.

The `unwind_protect` statement is often used to reliably restore the values of global variables that need to be temporarily changed.

3.7 Variable-length argument lists

Octave has a real mechanism for handling functions that take an unspecified number of arguments, so it is no longer necessary to place an upper bound on the number of optional arguments that a function can accept.

Here is an example of a function that uses the new syntax to print a header followed by an unspecified number of values:

```
function foo (heading, ...)
  disp (heading);
  va_start ();
  while (--nargin)
    disp (va_arg ());
  endwhile
endfunction
```

Calling `va_start()` positions an internal pointer to the first unnamed argument and allows you to cycle through the arguments more than once. It is not necessary to call `va_start()` if you do not plan to cycle through the arguments more than once.

The function `va_arg()` returns the value of the next available argument and moves the internal pointer to the next argument. It is an error to call `va_arg()` when there are no more arguments available.

It is also possible to use the keyword *all_va_args* to pass all unnamed arguments to another function.

3.8 Variable-length return lists

Octave also has a real mechanism for handling functions that return an unspecified number of values, so it is no longer necessary to place an upper bound on the number of outputs that a function can produce.

Here is an example of a function that uses the new syntax to produce 'N' values:

```
function [...] = foo (n)
  for i = 1:n
    vr_val (i);
  endfor
endfunction
```

3.9 Built-in ODE and DAE solvers

Octave includes LSODE and DASSL for solving systems of stiff differential and differential-algebraic equations. These functions are built in to the interpreter.

4 What documentation exists for Octave?

The Octave distribution includes a 220+ page manual that is also distributed under the terms of the GNU GPL.

The Octave manual is intended to be a complete reference for Octave, but it is not a finished document. If you have problems using it, or find that some topic is not adequately explained, indexed, or cross-referenced, please send a bug report to bug-octave@che.utexas.edu.

Because the Octave manual is written using Texinfo, the complete text of the Octave manual is also available on line using the GNU Info system via the GNU Emacs, info, or xinfo programs, or by using the '`help -i`' command to start the GNU info browser directly from the Octave prompt.

It is also possible to use WWW browsers such as Mosaic to read the Octave manual (or any other Info file) by using Roar Smith's info2www program to convert GNU Info files to HTML. The source for info2www is available via anonymous ftp from ftp.che.utexas.edu in the directory '`/pub/www`'.

5 Obtaining Source Code

5.1 How do I get a copy of Octave for Unix?

You can get Octave from a friend who has a copy, by anonymous FTP, or by ordering a tape or CD-ROM from the Free Software Foundation (FSF).

Octave was not developed by the FSF, but the FSF does distribute Octave, and the developers of Octave support the efforts of the FSF by encouraging users of Octave to order Octave on tape or CD directly from the FSF.

The FSF is a nonprofit organization that distributes software and manuals to raise funds for more GNU development. Buying a tape or CD from the FSF contributes directly to paying staff to develop GNU software. CD-ROMs cost $400 if an organization is buying, or $100 if an individual is buying. Tapes cost around $200 depending on media type.

The FSF only makes new CD releases a few times a year, so if you are interested specifically in Octave, I recommend asking for the latest release on tape.

For more information about ordering from the FSF, contact gnu@prep.ai.mit.edu, phone (617) 876-3296 or anonymous ftp file '`/pub/gnu/GNUinfo/ORDERS`' from prep.ai.mit.edu or one of the sites listed below.

If you are on the Internet, you can copy the latest distribution version of Octave from the file '`/pub/octave/octave-M.N.tar.gz`', on the host '`ftp.che.utexas.edu`'. This tar file has been compressed with GNU gzip, so be sure to use binary mode for the transfer. '`M`' and '`N`' stand for version numbers; look at a listing of the directory through ftp to see what version is available. After you unpack the distribution, be sure to look at the files '`README`' and '`INSTALL`'.

Binaries for several popular systems are also available. If you would like help out by making binaries available for other systems, please contact bug-octave@che.utexas.edu.

A list of user-visible changes since the last release is available in the file '`NEWS`'. The file '`ChangeLog`' in the source distribution contains a more detailed record of changes made since the last release.

5.2 How do I get a copy of Octave for (some other platform)?

Octave currently runs on Unix-like systems only. It should be possible to make Octave work on other systems. If you are interested in porting Octave to other systems, please contact bug-octave@che.utexas.edu.

5.3 What is the latest version of Octave

The latest version of Octave is 1.1.0, released January 1995.

6 Installation Issues and Problems

Octave requires approximately 50MB of disk storage to unpack and install (significantly less if you don't compile with debugging symbols).

Octave has been compiled and tested with g++ and libg++ on a SPARCstation 2 running SunOS 4.1.2, an IBM RS/6000 running AIX 3.2.5, DEC Alpha systems running OSF/1 1.3 and 3.0, a

DECstation 5000/240 running Ultrix 4.2a, and i486 systems running Linux. It should work on most other Unix systems that have a working port of g++ and libg++.

6.1 What else do I need?

In order to build Octave, you will need a current version of g++, libg++, and GNU make. If you don't have these tools, you can get them from many anonymous ftp archives, including ftp.che.utexas.edu, ftp.uu.net, prep.ai.mit.edu, and wuarchive.wustl.edu, or by writing to the FSF at 675 Mass Ave, Cambridge, MA 02139, USA.

6.2 Can I compile Octave with another C++ compiler?

Currently, Octave can only be compiled with the GNU C++ compiler. It would be nice to make it possible to compile Octave with other C++ compilers, but the maintainers do not have sufficient time to devote to this. If you are interested in working to make Octave portable to other compilers, please contact bug-octave@che.utexas.edu.

7 Common problems

This list is probably far too short. Feel free to suggest additional questions (preferably with answers!)

Octave takes a long time to find symbols.

Octave is probably spending this time recursively searching directories for function files. Check the value of your LOADPATH. For those elements that end in '`//`', do any name a very large directory tree? Does it contain directories that have a mixture of files and directories? In order for the recursive directory searching code to work efficiently, directories that are to be searched recursively should have either function files only, or subdirectories only, but not a mixture of both. Check to make sure that Octave's standard set of function files is installed this way.

8 Getting additional help

The mailing list

```
help-octave@che.utexas.edu
```

is available for questions related to using, installing, and porting Octave that are not adequately answered by the Octave manual or by this document.

If you would like to join the discussion and receive all messages sent to the list, please send a short note to

```
help-octave-request@che.utexas.edu
            ^^^^^^^
```

Please do not send requests to be added or removed from the the mailing list, or other administrative trivia to the list itself.

An archive of old postings to the help-octave mailing list is maintained on ftp.che.utexas.edu in the directory '`/pub/octave/MAILING-LISTS`'.

9 I think I have found a bug in Octave.

"I think I have found a bug in Octave, but I'm not sure. How do I know, and who should I tell?"

First, see the section on bugs and bug reports in the Octave manual. The Octave manual is included in the Octave distribution.

When you report a bug, make sure to describe the type of computer you are using, the version of the operating system it is running, and the version of Octave that you are using. Also provide enough code so that the Octave maintainers can duplicate your bug.

If you have Octave working at all, the easiest way to do this is to use the Octave function `bug_report`. When you execute this function, Octave will prompt you for a subject and then invoke the editor on a file that already contains all the configuration information. When you exit the editor, Octave will mail the bug report for you.

If for some reason you cannot use Octave's `bug_report` function, mail your bug report to "bug-octave@che.utexas.edu". Your message needs to include enough information to allow the maintainers of Octave to fix the bug. Please read the section on bugs and bug reports in the Octave manual for a list of things that should be included in every bug report.

10 Porting programs from MATLAB to Octave

"I wrote some code for MATLAB, and I want to get it running under Octave. Is there anything I should watch out for?"

The differences between Octave and MATLAB typically fall into one of three categories:

1. Irrelevant.
2. Known differences, perhaps configurable with a user preference variable.
3. Unknown differences.

The first category, irrelevant differences, do not affect computations and most likely do not affect the execution of function files. Some examples are:

When typing '`help function`', Octave displays the first set of comment lines *after* the function declaration, but MATLAB the first set of comment lines starting from the beginning of the file.

The differences of the second category are usually because the authors of Octave decided on a better (subjective) implementation that the way MATLAB does it, and so introduced "user preference variables" so that you can customize Octave's behavior to be either MATLAB-compatible or to use Octave's new features. To make Octave more MATLAB-compatible, put the following statements in your '~/.octaverc' file. This is a partial list of the user preference variables that should be changed to get MATLAB-compatible behavior. (It is partial because not all the differences are currently known, and when they become known, this document lags behind.)

```
do_fortran_indexing = 'true';
treat_neg_dim_as_zero = 'true';
empty_list_elements_ok = 'true';
implicit_str_to_num_ok = 'true';
whitespace_in_literal_matrix = 'traditional';
prefer_zero_one_indexing = 'true';
```

Some other known differences are:

String subscripting is not yet implemented in Octave. For example,

```
a = 'reknob';
a([6,1,5,3,2,4])
```

returns the string 'broken' in MATLAB, but generates an error in Octave. A future release of Octave will fix this along with providing a much more complete and powerful set of functions for manipulating strings.

The Octave plotting functions are mostly compatible with the ones from MATLAB 3.x, but not from MATLAB 4.x.

The C-style I/O functions are not completely compatible. It would be useful for someone to explore the differences so that they might be fixed, or at least noted in the manual.

The third category of differences is (hopefully) shrinking. If you find a difference between Octave behavior and MATLAB, then you should send a description of this difference (with code illustrating the difference, if possible) to bug-octave@che.utexas.edu.

An archive of old postings to the Octave mailing lists is maintained on ftp.che.utexas.edu in the directory '/pub/octave/MAILING-LISTS'.

Appendix A Concept Index

A

B

C

D

Table of Contents