

# Octave

## Introduction

GNU Octave is a high-level interpreted language, primarily intended for numerical computations. It provides capabilities for the numerical solution of linear and nonlinear problems, and for performing other numerical experiments. It also provides extensive graphics capabilities for data visualization and manipulation. Octave is normally used through its interactive command line interface, but it can also be used to write non-interactive programs. The Octave language is quite similar to Matlab so that most programs are easily portable. Read more on <http://www.gnu.org/software/octave/>\*\*\*

Two versions of octave are available on Anselm, via module

Version module	————— — — —————	Octave 3.8.2, compiled with GCC and Multithreaded MKL
		Octave/3.8.2-gimkl-2.11.5
		Octave 4.0.1, compiled with GCC and Multithreaded MKL
		Octave/4.0.1-gimkl-2.11.5
		Octave 4.0.0, compiled with >GCC and OpenBLAS
		Octave/4.0.0-foss-2015g

## Modules and execution

```
$ module load Octave
```

The octave on Anselm is linked to highly optimized MKL mathematical library. This provides threaded parallelization to many octave kernels, notably the linear algebra subroutines. Octave runs these heavy calculation kernels without any penalty. By default, octave would parallelize to 16 threads. You may control the threads by setting the OMP\_NUM\_THREADS environment variable.

To run octave interactively, log in with ssh -X parameter for X11 forwarding. Run octave:

```
$ octave
```

To run octave in batch mode, write an octave script, then write a bash jobscript and execute via the qsub command. By default, octave will use 16 threads when running MKL kernels.

```
#!/bin/bash
```

```
# change to local scratch directory
cd /lscratch/$PBS_JOBID || exit
```

```
# copy input file to scratch
cp $PBS_O_WORKDIR/octcode.m .
```

```

# load octave module
module load octave

# execute the calculation
octave -q --eval octcode > output.out

# copy output file to home
cp output.out $PBS_O_WORKDIR/.

#exit
exit

```

This script may be submitted directly to the PBS workload manager via the qsub command. The inputs are in octcode.m file, outputs in output.out file. See the single node jobscript example in the Job execution section.

The octave c compiler mkcofile calls the GNU gcc 4.8.1 for compiling native c code. This is very useful for running native c subroutines in octave environment.

```
$ mkcofile -v
```

Octave may use MPI for interprocess communication This functionality is currently not supported on Anselm cluster. In case you require the octave interface to MPI, please contact Anselm support.

## Xeon Phi Support

Octave may take advantage of the Xeon Phi accelerators. This will only work on the Intel Xeon Phi accelerated nodes.

### Automatic offload support

Octave can accelerate BLAS type operations (in particular the Matrix Matrix multiplications] on the Xeon Phi accelerator, via Automatic Offload using the MKL library

Example

```

$ export OFFLOAD_REPORT=2
$ export MKL_MIC_ENABLE=1
$ module load octave
$ octave -q
octave:1> A=rand(10000); B=rand(10000);
octave:2> tic; C=A*B; toc
[MKL] [MIC --] [AO Function]      DGEMM
[MKL] [MIC --] [AO DGEMM Workdivision]    0.32 0.68
[MKL] [MIC 00] [AO DGEMM CPU Time]    2.896003 seconds

```

```
[MKL] [MIC 00] [AO DGEMM MIC Time]      1.967384 seconds
[MKL] [MIC 00] [AO DGEMM CPU->MIC Data]   1347200000 bytes
[MKL] [MIC 00] [AO DGEMM MIC->CPU Data]   2188800000 bytes
Elapsed time is 2.93701 seconds.
```

In this example, the calculation was automatically divided among the CPU cores and the Xeon Phi MIC accelerator, reducing the total runtime from 6.3 secs down to 2.9 secs.

### Native support

A version of native Octave is compiled for Xeon Phi accelerators. Some limitations apply for this version:

- Only command line support. GUI, graph plotting etc. is not supported.
- Command history in interactive mode is not supported.

Octave is linked with parallel Intel MKL, so it best suited for batch processing of tasks that utilize BLAS, LAPACK and FFT operations. By default, number of threads is set to 120, you can control this with `> OMP_NUM_THREADS` environment variable.

Calculations that do not employ parallelism (either by using parallel MKL eg. via matrix operations, `fork()` function, parallel package or other mechanism) will actually run slower than on host CPU.

To use Octave on a node with Xeon Phi:

```
$ ssh mic0                                # login to the MIC card
$ source /apps/tools/octave/3.8.2-mic/bin/octave-env.sh # set up environment variables
$ octave -q /apps/tools/octave/3.8.2-mic/example/test0.m # run an example
```