Oscar Software Modules

List and instructions for software modules available on the Oscar cluster.

Currently Available Modules

If you require a software package that is not currently available on the Oscar cluster, please contact ICERM's IT staff and we will work with CCV to get the software installed.

This list is current as of January 7, 2025. To see the most up to date list of software modules, log into your Oscar account and run the command `module avail`.

```
---- /oscar/runtime/software/spack/0.20.1/share/spack/lmod/linux-rhel9-x86_64/Core ----
   abaqus-container/2021-akaeexs
   abaqus/2017-q4ghhm5
   abaqus/2021.1-i675dvw
   abaqus/2024-h5273a3                    (D)
   abaqus/2024-ir-a4m5ld5
   abaqus/2024-mbessa-ceayfuo
   abaqus/2024.1-7pcdqhp
   admixture/1.3.0-onwaqrp
   afni/23.3.07s-zm43m3u
   afni/24.2.01s-kstpoqt                  (D)
   anaconda/2023.09-0-7nso27y
   angsd/0.935-cbhuwc7
   ant/1.10.13-alpqj4j
   ants/2.4.3-75npyop
   aria2/1.36.0-lsb7zcs
   arm-forge/22.1.3-zq7lvdq
   armadillo/12.2.0-4clpczv
   atom/1.19.3-ty5sdsn
   autoconf/2.69-p4rpdx2
   avogadro2/1.99.0-5zl5qaw
   awscli/1.27.84-v22kngs
   bamtools/2.5.2-ki3mdef
   basilisk/2023.11.11s-x4isdvp
   bazel/6.1.1-vvtxktr
   bbmap/39.01-jnnkpwk
   bcftools/1.13-76jesdj
   bcftools/1.16-ewu6fpe                  (D)
   bcl2fastq2/2.20.0.422-z3wh636
```

beagle/5.4-e43mqsa
bedops/2.4.40-bjb2v2n
bedtools2/2.31.0-lsohc7s
bismark/0.23.0-eoksupu
blast-legacy/2.2.26-tcdku3a
blast-plus/2.2.30-cyxldrt
blat/37-ebfj5e6
blender/4.0.0s-v667vhv
boost/1.80.0-harukoy
bowtie/1.3.1-2kd7din
bowtie2/2.4.2-xdquyzq
bowtie2/2.5.3-qgscc2u (D)
brotli/1.0.9-h22dril
bwa/0.7.17-lu4b4dj
bxh-xcede-tools-container/1.11.14-4sphv7n
cadence/IC06.18-calibre2022.2-ascb7dw
cadence/IC06.18.090-6famfci
cadence/IC23.10.000-ppqll2n (D)
casa/6.6.0-20-py3.8.el7-dqvn5lw
cdhit/4.8.1-bqmf4jf
cellranger/arc-2.0.1-uamrhhu
cellranger/atac-2.0.0-m2tfcpk
cellranger/6.0.0-dbztt7r (D)
cfitsio/4.2.0-5grfqtu
cgal/5.4.1-64mikhl
chrome/119.0.6045.159s-avadhvk
cli11/2.3.2-pcucv7l
clustal-omega/1.2.4-mbj3dq5
cmake/3.6.1-il7bkvj
cmake/3.26.3-xi6h36u (D)
cnvnator/0.4.1-w3bkqjf
code-server/4.20.0-tcrmrcm
colordiff/1.0.21-ifskyqr
comsol/5.2-ufifhtv
comsol/5.6_yqi27-jnspqto
comsol/6.3_yqi27-7yf67lt (D)
conn/22a-nztrdv3
connectome-workbench/1.5.0-t66riqu
cppunit/1.14.0-h3hsjgu
crossrate-container/2016-27ofi4r

cuda/10.1.243-bxisbai
cuda/10.2.89-xnfjmrt
cuda/11.8.0-lpttyok
cuda/12.1.1-ebglvvq
cuda/12.2.0-4lgnkrh
cuda/12.3.0-r72aozf (D)
cudnn/7.5.1.10-10.1-hv4e2lt
cudnn/8.7.0.84-11.8-lg2dpd5
cudnn/8.9.6.50-12-56zgdoa (D)
cufflinks/2.2.1-ogzw3z5
cutensor/1.5.0.3-gqkzath
datamash/1.8-ib4aakp
dcm2niix/1.0.20220720-nwsidfo
diamond/2.0.15-h7xx24l
dicombrowser/20181217s-ikvqhyr
dlib/19.22-lxah7rq
dmtcp/3.0.0-xvfukfp
dorado/0.8.2-s42dhri
dos2unix/7.4.2-5a6dlgt
dotnet/8.0.100-5lr7bga
dropest/0.8.6-ewwx5ik
ds/9.8.5s-zpqg2jy
dsi-studio/chen-2023-sif-lytwlk2
dtitk/2.3.1s-bp7yqjh
eigen/3.4.0-uycckhi
eigensoft/7.2.1-6ctbhoz
emacs/28.2-rwds2pd
expat/2.5.0-zujcztp
fastme/2.1.5.1-kmg5til
fastp/0.23.4-xmfbk37
fastq-screen/0.15.3-7ymgrux
fastqc/0.11.9-mvd2uhw
fastqc/0.12.1-sk2rb3a (D)
fasttree/2.1.11-o5kvig7
fastx-toolkit/0.0.14-zhaxiyn
ferret/7.6.0-rzhafh6
ffmpeg/6.0-fy677gn
ffmpeg/7.0-xny2fb2 (D)
fiji/20231107-1617-espdc7g
filezilla/3.49.1-epfjuus

flashpca/2.0-zr2wflq
freebayes/1.3.6-v7rppcd
freeglut/3.2.2-76qqoqn
freesurfer/7.3.2-zop5n6m
fsl/6.0.7.7s-bul4mby
fv/5.5.2-g2ibb5x
fzf/0.45.0-pdwl7a4
gatk/4.3.0.0-234wqft
gaussian/09_v1-u6klkps
gaussian/09-D01_v2-tw73726
gaussian/09-D01-TEST_v3-vv6ar67
gaussian/16-C01-bb2r2gh (D)
gaussview/v05-mkdyw6j
gcc/6.5.0-lwshmxc
gcc/10.1.0-mojgbnp
gcc/13.1.0-nvrtbp3 (D)
gcm/2.4.1-lfqoarh
gdal/3.7.0-4p4onmf
geeqie/2.4-6vdnc4v
geos/3.11.2-a6hfu6a
ghostscript/10.0.0-3atesdh
gimp/2.10.32-tlknk2n
git-lfs/3.3.0-laphnvj
git/2.44.0-6f7n7ni
glew/2.2.0-plawm2j
glm/0.9.9.8-m3s6sze
glpk/5.0-zifs7bb
gmap-gsnap/2024-08-20-dur7jyc
gmp/6.2.1-qlaig4m
gnuplot/5.4.3-pdiiquy
go/1.17.1-f4mqosa
go/1.20.3-xknmcqd
go/1.23.3-d3wvs6z (D)
google-cloud-cli/456.0.0-3mtj4z6
gperf/3.1-56q4xf5
grace/5.1.25-duvo7rn
graphviz/8.0.1-75znavc
gsl/2.7.1-khmyfcy
guppy/6.0.1-wpaqayj
guppy/6.1.2-wwvvdfu (D)

gurobi/10.0.1-q7rc5dw
hdf5/1.12.2-s6aacp3
hdf5/1.14.1-2-rdd6y6v (D)
hisat2/2.2.1-gn4pb3l
homer/4.11.1-fpjs4l4
hpcx-mpi/4.1.5rc2-mts-ukpby4i
hpcx-mpi/4.1.5rc2s-yflad4v (D)
htop/3.2.2-kqsjlaj
htslib/1.12-ecidzx4
htslib/1.17-zxcat2k (D)
idba/1.1.3-nrxiqtw
idemp/201706-a45gc3d
idl/8.8.2-d5p4srq
igraph/0.7.1-wbiepb3
imagej/154-linux64-java8-jd6sflr
imagemagick/7.1.1-3-ex4k4u2
inkscape/1.3s-gshcpwc
intel-oneapi-compilers/2023.1.0-a7fw7qt
intel-oneapi-mkl/2023.1.0-xbcd2g3
intltool/0.51.0-vanhjsr
iq-tree/2.1.3-gu64b4j
iq-tree/2.3.6-fn5vscb (D)
iraf/2.17.1s-2r7ypc5
itk-snap-container/4.0.2-bychj73
jags/4.3.1-4mvaxc3
jellyfish/2.2.7-dywzm7z
jo/1.9-ki7xc2u
json-fortran/8.3.0-ehkzpjv
jsoncpp/1.9.5-vhsa2iy
julia/1.9.3s-i3zndt3
julia/1.11.1s-kueea5s (D)
kraken/1.1.1-e6r2aej
lemon/1.3.1-jh3h4xt
leptonica/1.81.0-pebiyok
leveldb/1.23-f76iwfr
lftp/4.9.2-vimt4vf
libarchive/3.6.2-mnc5shn
libbeef/Nov2020-xhrdwg5
libdeflate/1.10-5yi7m3g
libgd/2.2.4-2iyhgxa

libgd/2.3.3-ubu4k2f (D)
libgeotiff/1.6.0-voueb6b
libgit2/1.6.4-a432pgi
libiconv/1.17-jwjcds2
libjpeg-turbo/2.1.5-sewtk5u
libjpeg/9e-6djp5nd
libnsl/1.3.0-calriiy
libnsl/2.0.1-ed2i5hn (D)
libpng/1.2.57s-lve65hz
libpng/1.5.30-ru3zswz
libpng/1.6.39-ryxiwrd (D)
libreoffice/7.2.2.sif-drpjygp
libsdl/2.30.7-4rxs72s
libtiff/4.5.0-g6fga7e
libtree/3.1.1-yxst452
libvips/8.13.3-ex4pfpg
libwnck/3.24.1-4gvyjhg
libxc/4.3.4-uy5ogwb
libxc/5.2.3-ncc5ir4 (D)
libzip/1.3.2-qwqikw6
liggghts/3.8.0s-rqph5mk
linaro-forge/23.1.2-pk2lobu
llvm/16.0.2-mq6g5lb
lmod/8.7.24-w2akdkb
mafft/7.505-iuicuiv
magma-usyd/V2.28-8-p3zylpg
magma/2.7.1-cpueyjy
maple/22-cp2uld4
mark/2018.07.08-43snzfu
materialstudio/2024s-gbc3dg6
mathematica/13.2.0-n4i6yua
matlab/R2019a-rjyk3ws
matlab/R2023a-xd6f7ph (D)
matlab/R2024b-ipaztju
maven/3.8.4-w3zgh4v
mercurial/5.8-vly6btb
mesa/22.1.6-6dbg5gq
meson/1.6.0-eipcwzq
metis/5.1.0-7qoahod
miniconda3/23.11.0s-odstpk5

miniforge/23.11.0-0s-hwmjdtj
minimap2/2.14-33hmvx2
molden/6.7-isryqwj
molden/7.3-kytvh3m (D)
molpro-mpi/2023.2.0s-vyfv74n
molpro-mpi/2024.3.1-mpipr-hwakoux (D)
mpc/1.3.1-zbino7j
mpfr/4.2.0-n2tkxso
mriconvert/2.1.0-oaq24fz
mricrogl/2022.07.20-n3b7whc
mricron/201909-3s2phrj
msmc2/2.1.4-cuac55f
multiwfn/3.8_1208-qn65pif
mummer4/4.0.0rc1-4llgadq
muscle/3.8.1551-pys2b76
mysql/8.0.29-w7xdbde
nanoflann/1.4.3-u2l24dv
nbo/7.0-phausw3
nccl/2.16.2-1-gjmprw5
ncdu/1.18.1-uofylp6
nco/5.1.5-36hru5t
ncview/2.1.8-nxepxtw
neovim/0.9.4-vygnfr2
netcdf-c/4.9.2-cfggqwi
netcdf-c/4.9.2-4ozokng (D)
netcdf-cxx4/4.3.1-6gcdg5s
netcdf-fortran/4.6.0-kl27oji
netcdf/4.9.2-ar77jpt
netlib-lapack/3.11.0-jdzmstx
netlogo/6.4.0-psm765m
netpbm/10.73.43-m2jdopk
ngc-jax/23.10-paxml-py3-ziphcif
ngc-pytorch/24.03-py3-a6ptiby
ngc-tensorflow/24.03-tf2-py3-lmnuwwg
ninja/1.11.1-k2aq3rl
nlopt/2.7.1-fwj27pk
nnn/4.9-r7kawsv
node-js/18.12.1-qkps4za
nvhpc/23.3-xa4nyqi
nvtop/3.0.1-7r22pjl

octopus-lunter/0.7.4-kkqrfv3
ollama/0.3.14s-a3gonhs
openbabel/3.1.1-nay2mkb
openblas/0.3.23-u6k5fey
opencv/4.6.0s-5z4piup
openexr/3.1.5-6fapou6
openjdk/11.0.17_8-nw5ylvi
openjdk/17.0.5_8-pq2e7ao (D)
openjpeg/2.5.0-iyu5vwb
openmpi/4.1.2-s5wtoqb
openmpi/4.1.4s-smqniuf
openmpi/4.1.5-hkgv3gi
openmpi/4.1.5-kzuexje (D)
openslide/3.4.1-pzjb2kl
openssl/1.1.1t-u2rkdft
or-tools/9.10-k4nov4d
ovito/3.6.0-rile7ax
p7zip/17.05-3xtimiz
pandoc/2.19.2-wawlx5m
pangolin/0.6-vwij3iv
parallel/20220522-5ah2i5h
paraview/5.9.0s-dgv24kr
patchelf/0.17.2-aqmx4qb
paup/4.0a168-cwt24ux
pcre2/10.42-xks64jg
pdftk/2.02-gu7lpeg
pdsh-chaos/23.12-t6ywlrp
perl-dbi/1.643-t74vmeb
perl/5.26.2-o4iq4b4
perl/5.36.0-bt34quz (D)
perl/5.37.9-og4osvm
perl/5.40.0-o7hxci2
picard/2.26.2-qabtyqy
pigz/2.7-zgdlry3
plink/1.9-beta6.27-nvy4vrx
plink/2.00-b6x44xw (D)
popoolation2/1.205s-luyjn2b
postgresql/16.4-vzmexkn
prodigal/2.6.3-vbq7usx
proj/9.2.0-ni5rcfb

protobuf/3.22.2-6hlkkut
py-ase/3.21.0-pyuljod
py-matplotlib/3.7.1-4afsjsz
py-statsmodels/0.13.2-sbdhj4k
py-sympy/1.11.1-gqgr7wu
pycharm-community/2021.3.3-weqrcly
pypy/7.3.13-6a5ma5j
python/3.9.16s-x3wdtvt
python/3.11.0s-ixrhc3q (D)
qgis/3.28.3-5axmqsj
qit/2023-04-04-grwuvgg
qmcpack-mpi/3.16.0s-qp2hymx
qscintilla/2.11.6-dq7zlcq
qt/5.15.9-fb7mjex
qualimap/2.2.1-mybpdoi
quantum-espresso-mpi/7.1-gits-v4bxgtv
quantum-espresso-mpi/7.1s-ia43pjk
quantum-espresso-mpi/7.3s-kydgjwo (D)
r/4.0.0-p7gxu4e
r/4.0.3-pvf2znb
r/4.1.0-bfjsvw5
r/4.2.2-z6qdiis
r/4.3.1-lmofgb4
r/4.4.0-yycctsj (D)
raisd/2.9-svyic22
rclone/1.62.2-o4lkrv6
readline/6.3-rnqups2
root/6.28.04-u7t5ax7
rsem/1.3.3-5obucw6
rstudio/2023.09.1-lsqy746
ruby/3.1.0-gnoxsfm
rust/1.73.0-647r2tw
rust/1.81.0-3qqoayc (D)
sage-container/10.3-avpqipf
sage/9.5-drpqjkh
sage/10.3-nntihfr (D)
salmon/1.9.0-itdua6n
salmon/1.10.1-43ljn7g (D)
samtools/1.12-4v4uiz6
samtools/1.16.1-txuglks (D)

sas/9.4m8-idx4uxl
schmutzi-container/1.5.7-vsorpcq
schmutzi-container/1.5.7-3imwf7h (D)
schrodinger/2023-4-h3kvbn3
schrodinger/2024-1-yqi27-m4qqm46 (D)
scons/4.5.2-housgyw
seqkit/0.10.1-qtiftw4
seqtk/1.3-kbdjwob
shapeit4/4.2.2-3us45un
singular/4.4.0-aeloppr
skewer/0.2.2-nwhklgr
slicer/5.4.0-rb2kk4l
slim/4.0.1-kymgtmu
slim/4.3-u22vcwu (D)
spdlog/1.11.0-qbi24my
splash/2.1.4-unrsfpj
spm/8-77b5myx
spm/12_r7606-prcq7fg (D)
sratoolkit/3.0.0-u4jvgps
stacks/2.65-geg4r7a
star/2.7.10b-fj6kao2
stata/mp17-v7a7uoo
stata/mp18-3wq5b4o (D)
stow/2.4.0-y2q7tsn
stringtie/2.2.1-7uti3ny
sublime-text/4.4143-im3loi3
subread/2.0.2-5agghnd
swig/4.1.1-bq46cxl
synopsys/2023.12-df4a3ab
synopsys/2024.09-q3jwzot (D)
tabix/2013-12-16-d6qvxp7
tcsh/6.24.10-dtqo5ky
tecplot/2022r1-q5cg2zq
tesseract/4.1.1-l2ejycz
tesseract/5.3.3-vq3altt (D)
texlive/20220321-pocclov
texstudio/3.0.1-64vxo64
tmux/3.3a-zyhjvvh
tn93/1.0.12-tcvbyl4
tree/2.1.0-7tlhzo7

trimal/1.4.1-ace7du2
trimgalore/0.6.6-iwfrq4c
trimgalore/0.6.9-hisz5xp (D)
trimmomatic/0.39-w5jnhai
udunits/2.2.28-rycabdx
usearch/11.0.667-wx6utmj
v8/3.14.5-aompxje
vasp-mpi/5.4.4-mdh3hpy
vasp-mpi/6.3.2_avandewa-chn3w3j
vasp-mpi/6.4.2_cfgoldsm_vtst-cff5qmk
vasp-mpi/6.4.2_cfgoldsm-7krhcss
vasp-mpi/6.4.3_yqi27-6uzdgwn (D)
vcftools/0.1.14-syssqsi
vim/9.1.0867-wl3haj7
virtualgl/3.1-yphbrfj
visit-container/3.3.3-qz3dni6
visit-mpi/3.3.3s-lz2dp7m
vmd/1.9.3-oin2dnj
vscode/1.84.2-4tfimgp
wcstools/3.9.7-lo4forb
wxwidgets/3.2.2.1-mk5eiyq
xcrysden/1.5.60-nuxe46i
xeyes/1.2.0-nge56yb
xgboost/1.6.2-fp3ii65
yaml-cpp/0.7.0-6nno2ru
zlib/1.2.13-jv5y5e7
zoxide/0.9.2-ydhigq6
zstd/1.5.5-zokfqsc

Oscar: Sage

Loading and Launching Sage

1. Once authenticated to Oscar, use the following commands at the command line.

2. Start an interactive job by using the `interact` command. This command can take additional parameters to extend the resources and time allotted to the node as well as the partition that the node operates on.

3. The Sage module provides containers. To load them use `module load sage-container/10.3`.

4. To start the container use `apptainer shell /oscar/rt/9.2/software/0.20-generic/0.20.1/opt/spack/linux-rhel9-x86_64_v3/gcc-11.3.1/sage-container-10.3-avpqipfsnbneig726l72jrgdmlrivg4m/sage.sif`

5. Once inside the container's shell use `sage` to launch the Sage console.

Sage on Oscar OnDemand

The easiest way to run Sage on Oscar OcDemand is to run sage in an interactive job via the terminal in your OnDemand session.

Use the interact command with parameters for your specific job to start the interactive session, then load your modules and run the sage binary (steps 2-4 above).

```
interact -n 2 -m 32g -t 04:00:00 -f 'haswell|broadwell|skylake'
```

Using Sage with Batch Scripts

Thanks to Trevor Hyde from Summer@ICERM 2019 for these instructions.

One method for running computations with Sage on Oscar is to write a script and use the slurm batch scheduler to have Oscar run your script. This requires two pieces:

1. A shell script to configure and submit your batch job to the cluster.

2. Your Sage code/program you'd like to run.

Example Batch Script

sage-batch.sh

```
#!/bin/bash

#SBATCH -J test_program
#SBATCH --array=0-9
#SBATCH -t 1:00:00
#SBATCH --mem=8G

#SBATCH -e data/<oscar-username>/test_output/test%a.err
#SBATCH -o data/<oscar-username>/test_output/test%a.out

module load sage-container/10.3

apptainer shell /oscar/rt/9.2/software/0.20-generic/0.20.1/opt/spack/linux-rhel9-x86_64_v3/gcc-11.3.1/sage-container-10.3-avpqipfsnbneig726l72jrgdmlrivg4m/sage.sif

sage test_program.sage $SLURM_ARRAY_TASK_ID
```

- `#!/bin/bash` tells the system this is a bash (shell) script.
- `#SBATCH -J test_program` sets the name of the job which appears when you check the status of your jobs.
- `#SBATCH –array=0-9` is an easy way of doing parallel computations. In this case it says our job will run on 10 different nodes, each node will be passed a parameter and we have specified that the parameters will take the values 0 through 9. You can specify several ranges or even list individual parameters if you prefer.
- `#SBATCH -t 1:00:00` specifies a time limit in `HH:MM:SS` for each node. Once this time runs out your program will stop running on that node. Be careful setting the time limit too high as doing so may make it take a long time for your job to get scheduled to run. Before starting a big computation try to do some smaller tests to see how long you expect to need.
- `#SBATCH –mem=8G` specifies how much memory each node gets. Standard exploratory accounts get 123GB total to use at any one time. So if you allocate too much per job, fewer jobs will run at once. On the other hand, if you allocate too little and a computation needs more than it has, then it will terminate. If this happens an “out of memory” error will show up in the `.err` file for that node.

- `#SBATCH -e data/<ccv-username>/test_output/test%a.err` and `#SBATCH -o data/<ccv-username>/test_output/test%a.out` specify where the error messages and output for each computation should be sent. You should store these files in your user folder, not on the submit node. We each have a folder inside the `data` directory which you can see from the submit node. In this example I have created a folder titled `test_output` where I'm putting both of these files. **You need to make these folders before you run the computation otherwise the output will be dumped into the void!** The `%a` will get replaced with the array parameter. So for example, since we set our array parameters to be `0-9` there will be 10 nodes running and each of them gets a number between 0 and 9; this node corresponding to the parameter 7 will create two files `test7.err` and `test7.out`.
- `module load sage-container/10.3` loads the sage container into the node.
- `apptainer shell /oscar/rt/9.2/software/0.20-generic/0.20.1/opt/spack/linux-rhel9-x86_64_v3/gcc-11.3.1/sage-container-10.3-avpqipfsnbneig726l72jrgdmlrivg4m/sage.sif` initiates the container's Sage console shell.

Everything after this in the script happens as if you typed it yourself onto the command line.

- In our example, we want to run sage code, so the line `sage test_program.sage $SLURM_ARRAY_TASK_ID` runs our example sage program `test_program.sage`.
- The file needs to have the `.sage` extension.
- You should write this file in a text editor, not in a Jupyter notebook (although you can first write and test your program in a Jupyter notebook and then copy and paste it into a new file when it's ready).
- This program is written to accept one input and I have passed it `$SLURM_ARRAY_TASK_ID` which is the array parameter passed to each node. You can use this parameter to select which input parameters to run your program on.

Example Sage Program

test_program.sage

```
import sys


def fun_math(message):
    print message
    sys.stdout.flush()


job_id = int(sys.argv[1])
fun_math('hi this is a test')
fun_math('my job id is' + str(job_id))
```

- In the Sage program, you first define all of your functions and then you include the code you want to run.
- Import `sys` so you can access the array parameter passed to your function from the node. This is accessed in this case by `sys.argv[1]`. Make sure you explicitly coerce to be an integer if you want to use it as an integer; it's a string by default.
- The output of the `print` command is appended to the `.out` file for this node as a new line.
- Notice the line `sys.stdout.flush()` included in the function. This makes the program immediately send whatever output it has to the output file when called. Otherwise the program won't output **anything** until it has completely finished running. If each node is running 100 potentially long computations and it finishes the first 99 but then times out on the 100th computation, and you don't include any `sys.stdout.flush()` commands, everything will be lost when time runs out.

Submitting the Batch Job

- To run this batch program go back to the submit node and type `sbatch <NAME_OF_BATCH_FILE>`. In our example here, our batch file is called `sage-batch.sh`, so we simply type `sbatch sage-batch.sh` Slurm will return a line that tells you your job has been submitted together with a job id number.
- To check the progress of your jobs type `myq` from anywhere on Oscar. This will show you what jobs you have running, how much time they have left, and which jobs are still waiting to run. Be patient, sometimes it takes a minute for things to get started.
- If you realize your code is never going to finish or that you've made some terrible mistake, you can cancel a batch job by typing `scancel <JOB_ID>`. You can specify a single node or just put the general job id for the whole run and cancel everything.

Oscar: MATLAB

Loading and Launching MATLAB

1. Open the Terminal and use the following commands at the command line.

2. `module avail matlab` to list all the available matlab versions.

3. `module load matlab` to load the latest version of matlab (R2023a). Other versions can be specified with the command `module load matlab/R2019a`.

4. `matlab` to launch the MATLAB app.

Installing MATLAB Packages such as YALMIP

MATLAB script packages, such as YALMIP, can be installed directly by the user on their Oscar account.

1. Open the Terminal and connect to Oscar.

2. Navigate to your home folder by typing `cd ~`

3. `mkdir -p MATLAB`

4. `wget -O yalmip.zip https://github.com/yalmip/yalmip/archive/master.zip`

5. `unzip yalmip.zip`

6. In MATLAB, add the YALMIP-master directory to your path.

    1. In the MATLAB file browser, navigate to the MATLAB folder you created in your home folder. `cd ~/MATLAB`
    2. Right click on the YALMIP-master folder.
    3. Select Add to Path > Selected Folders and Subfolders. This adds the YALMIP folders to your path.

7. To save your MATLAB path, use the savepath command in the MATLAB command prompt. `savepath ~/MATLAB/pathdef.m`

YALMIP also requires a solver like SDPT3. The steps below add SDPT3 to MATLAB.

1. Open the Terminal.

2. `cd ~/MATLAB`

3. `wget -O sdpt3.zip https://github.com/sqlp/sdpt3/archive/master.zip`

4. `unzip sdpt3.zip`

5. In MATLAB, add the sdpt3 directory to your path.

   1. In the MATLAB file browser, navigate to the MATLAB folder you created in your home folder. `cd ~/MATLAB`
   2. Right click on the sdpt3-master folder.
   3. Select Add to Path > Selected Folders and Subfolders. This adds the SDPT3 folders to your path.

6. To update/save your MATLAB path, use the savepath command in the MATLAB command prompt. `savepath ~/MATLAB/pathdef.m`

Oscar: Mathematica

Loading and Launching Mathematica

1. Open the Terminal and use the following commands at the command line.

2. `module avail mathematica` to list all the available mathematica versions.

3. `module load mathematica` to load the latest mathematica version.

4. `mathematica` to launch the Mathematica app.