NavLab Software Infrastructure Setup

This document describes how to acquire and build all of the dependencies necessary to setup the Global Trajectory Optimization Lab software infrastructure on the NavLab servers.

All dependencies will be installed to `/archive/Utilities`.

ETD support should there be a technical issue with the NavLab servers: https://aetd-support.ndc.nasa.gov

Developer Support

Contact Donald Ellison (donald.ellison@nasa.gov) or Jacob Englander (jacob.a.englander@nasa.gov) if you find any problems with this guide or the build system itself.

Required dependencies

NOTE: All required software has already been installed on the NavLab machines, but here is a comprehensive list and current version that is present on the machines.

- GCC 7.2.0
    - GNU Multiple Precision Arithmetic Library (GMP) 6.1.2
    - GNU Multiple Precision Complex Library (MPC) 1.1.0
    - GNU Multiple Precision Floating-point Rounding (MPFR) 3.1.4
    - GNU Integer Set Library (ISL) 0.16.1
- GNU Scientific Library 2.4.0
- CMake 3.12.0
- OpenMPI 3.0.0
- SNOPT 7.6
- CSPICE N0066
- Boost 1.63.0

EMTG version 9 **requires** C++17 support, therefore gcc version 7+ MUST be installed.

NavLab machine IP addresses

Server 99: 128.183.251.46 (64 cores)
Server 100: 128.183.251.47 (64 cores, save some for Kenny and only use 60)
Server 101: 128.183.251.48 (64 cores)
Server 4810: 128.183.251.49 (60 cores)

To access these machines with the ssh protocol:

```
shell> ssh username@machineIP
```

GCC 7.2.0

GCC releases site: https://gcc.gnu.org/releases.html

MPC: http://www.multiprecision.org/mpc/download.html

MPFR: http://www.mpfr.org/mpfr-current/#download

GMP: https://gmplib.org/

ISL: https://gcc.gnu.org/pub/gcc/infrastructure/

Extract GCC tarball and rename it:

```
shell> tar -xzf gcc-7.2.0.tar.gz
shell> mv gcc-7.2.0 gcc-7.2.0-src
```

Place the four prerequisite archive packages inside the `gcc-7.2.0-src` directory

Run the prerequisite download script from the `gcc-7.2.0-src` directory:

`shell> ./contrib/download_prerequisites`

Configure GCC to build into a dedicated directory, separate from the src directory:

```
shell> mkdir /archive/Utilities/gcc-7.2.0
shell> cd gcc-7.2.0
shell>../gcc-7.2.0-src/configure --prefix=/archive/Utilities/gcc-7.2.0 --disable-multilib --
program-suffix=-7.2.0 --enable-languages=c,c++,fortran
```

Build and install GCC:

```
shell> make -j 60
shell> make install -j 60
```

Bash environment configuration

STOP: Do not change your bash until you have actually built GCC, otherwise, these changes will redirect everything to a gcc that does not exist yet.

Some of the edits below are preempting the packages that will be built throughout the remainder of this guide.

Your `~/.bashrc` must contain the following aliases to get around the older versions already installed on the NavLab machines:

```
alias gcc='/archive/Utilities/gcc-7.2.0/bin/gcc-7.2.0'
alias g++='/archive/Utilities/gcc-7.2.0/bin/g++-7.2.0'
alias gfortran='/archive/Utilities/gcc-7.2.0/bin/gfortran-7.2.0'
alias c++='/archive/Utilities/gcc-7.2.0/bin/c++-7.2.0'
alias cmake='/archive/Utilities/cmake-3.12.0/bin/cmake'
alias ccmake='/archive/Utilities/cmake-3.12.0/bin/ccmake'
alias mpiexec='/archive/Utilities/openmpi-3.0.0/bin/mpiexec'
alias mpicc='/archive/Utilities/openmpi-3.0.0/bin/mpicc'
alias mpicxx='/archive/Utilities/openmpi-3.0.0/bin/mpicxx'
alias mpifort='/archive/Utilities/openmpi-3.0.0/bin/mpifort'
```

Your `~/.bash_profile` must contain the following modifications:

```
# User specific environment and startup programs
export CC=/archive/Utilities/gcc-7.2.0/bin/gcc-7.2.0
export CXX=/archive/Utilities/gcc-7.2.0/bin/g++-7.2.0
export FC=/archive/Utilities/gcc-7.2.0/bin/gfortran-7.2.0
export MPICC=/archive/Utilities/openmpi-3.0.0/bin/mpicc
export MPICXX=/archive/Utilities/openmpi-3.0.0/bin/mpicxx
export MPIFC=/archive/Utilities/openmpi-3.0.0/bin/mpifort
export TKCOMPILER="gcc"

PATH=/archive/Utilities/gcc-7.2.0/bin:$PATH
PATH=/archive/Utilities/openmpi-3.0.0/bin:$PATH
LD_LIBRARY_PATH=/archive/Utilities/gcc-7.2.0/lib:/archive/Utilities/gcc-
7.2.0/lib64:$LD_LIBRARY_PATH
LD_LIBRARY_PATH=/archive/Utilities/SNOPT-7.5/lib/.libs:$LD_LIBRARY_PATH
LD_LIBRARY_PATH=/archive/Utilities/openmpi-3.0.0/lib:$LD_LIBRARY_PATH
LD_LIBRARY_PATH=/archive/Utilities/boost-1.63.0/stage/lib:$LD_LIBRARY_PATH

export PATH
export LD_LIBRARY_PATH
```

You can also choose to put everything in either `bashrc` or `bash_profile`.

Exit the ssh session and re-enter for the bash settings to take effect (you could use the source command, but restarting is cleaner).

CMake 3.12.0

Cmake downloads page: https://cmake.org/download/

Extract CMake tarball and rename it:

```
shell> tar -xzf cmake-3.12.0.tar.gz
shell> mv cmake-3.12.0 cmake-3.12.0-src
```

Bootstrap CMake to build into a dedicated directory, separate from the src directory:

```
shell> mkdir cmake-3.12.0
shell> cd cmake-3.12.0-src
shell> ./bootstrap --prefix=/archive/Utilities/cmake-3.12.0 --parallel=60
```

Build and install CMake:

```
shell> make -j 60
shell> make install -j 60
shell> cmake --version
```

OpenMPI 3.0.0

Cmake downloads page: https://cmake.org/download/

Extract OpenMPI tarball and rename it:

```
shell> tar -xjf openmpi-3.0.0.tar.bz2
shell> mv openmpi-3.0.0 openmpi-3.0.0-src
```

Bootstrap OpenMPI to build into a dedicated directory, separate from the src directory:

```
shell> mkdir openmpi-3.0.0
shell> cd openmpi-3.0.0-src
shell> ./configure --prefix=/archive/Utilities/openmpi-3.0.0
```

Build and install OpenMPI :

```
shell> make all install
```

GNU Scientific Library 2.4.0

GSL downloads: https://github.com/ampl/gsl/releases

GSL comes with its own CMake build system, if you get it from the AMPL project.

Extract GSL and create a dedicated build directory.

```
shell> tar -xzf gsl-2.4.0.tar.bz
shell> cd gsl-2.4.0
shell> mkdir build
shell> cd build
shell> ccmake ..
```

Configure then generate via the cmake console application.

```
make -j 60
```

SNOPT 7.6

SNOPT installation instructions: http://ccom.ucsd.edu/~optimizers/compile/c/

Extract SNOPT and create a dedicated build directory.

```
shell> unzip snopt7.6.zip
shell> mv snopt7.6 SNOPT-7.6
shell> cd SNOPT-7.6
shell> ./configure --with-cpp
shell> make interface
shell> make install
```

If you want to check the SNOPT build:

```
shell> make examples
shell> make check
shell> cd examples
shell> ./sntoya
```

CSPICE N0066

CSPICE Toolkit: https://naif.jpl.nasa.gov/naif/toolkit_C.html

Extract CSPICE and run the build script:

```
shell> tar -xzf cspice.tar.Z
shell> cd cspice
shell> ./makeall.csh
```

Boost 1.63.0

Boost download: http://www.boost.org/users/history/version_1_63_0.html

Extract Boost and run the bootstrap and build scripts:

```
shell> tar -xjf boost_1_63_0
shell> mv boost_1_63_0 boost-1.63.0
shell> cd boost-1.63.0
shell> ./bootstrap.sh
```

Now, edit the `project-config.jam` file to include `using mpi ;` near the top of the file as well as the exact location of the gcc version that you want to compile Boost with:

```
# Boost.Build Configuration
# Automatically generated by bootstrap.sh

import option ;
import feature ;
using mpi ;

# Compiler configuration. This definition will be used unless
# you already have defined some toolsets in your user-config.jam
# file.
if ! gcc in [ feature.values <toolset> ]
{
    using gcc : 7.2.0 : /archive/Utilities/gcc-7.2.0/bin/gcc-7.2.0 ;
}
```

Now build Boost:

```
shell> ./b2 -j 60
```

Re-run the build script to build the MPI libraries:

```
shell> ./b2 --with-mpi -j 60
```