EMTG Windows Installation

The first part of this guide describes how to install a distributed EMTG executable on Windows for analysis tasks. Developers who also wish to setup the EMTG build environment should complete all of the steps in this guide (both sections) beginning with the first section.

EMTG Installation Required Dependencies

- MS Visual Studio 2017 or better
- MinGW-64
- Python 3.0 or better (we use 3.7)
    - wxPython 4.0.3 or better
        - matplotlib 3.1.2 or better
    - SpiceyPy
    - JPLEphem
    - de423
- SNOPT 7.5 or 7.6
- CMake 3.12.0 or better
- Boost 1.70.0 or better
- GNU Scientific Library 2.4.0 or better
- CSPICE N0066 or better
- Randutils
- Mission-specific CSPICE Kernels

EMTG version 9 **requires** C++17 support, therefore, MSVS 2015 and older are not sufficient

MS C++ Redistributable Libraries

The redistributable libraries may be downloaded from this page.

MinGW-64

MinGW-64 is required in order to provide runtime Fortran libraries to SNOPT. These can also be provided using Intel Visual Fortran, however, MinGW is a free alternative.

The best way to do this is to install msys2 and then use its package manager to install everything else that you need.

Important: Make sure you download the 64 bit version of msys2, not the 32 bit version.

1. Download the 64-bit installer from https://www.msys2.org/.
2. Launch the msys2 bash shell.
3. type "pacman -S mingw-w64-x86_64-toolchain"
4. You will be prompted to select the components that you want. Just hit enter so that it installs everything.

5. Once MinGW has been installed, you must add its `bin` directory to your user and system environment variables. Go to **system->advanced settings->environment variables**.
    - For Windows 8+ use 'winkey+x' and select 'system'
    - For Windows 10 Creator's Update (i.e. May 2017 and onward) you'll ALSO have to then go to 'System Info'
    - For Windows 10, you can also just search for `environment` in the search bar and the utility will pop up
6. In the top half of the screen, under **User Variables** select **Path** and edit it.
7. Set the path to your msys2 bin folder. For example: `C:\msys64\mingw64\bin` depending on what installation path you chose.
8. Edit the system level Path variable with the same minGW path

Python 3.0 or better (we use 3.7)

We recommend using the Anaconda Python distribution as it is well-supported and had comes fully configured with the conda package manager, which makes acquiring the astrodynamics sub-packages straightforward.

1. The latest distribution may be obtained from the Anaconda project download page.
2. Run the installer.
3. There is no need to add Anaconda to the path via the installer (it recommends that you don't). Instead, you can add it manually later on if you would like commands such as `conda`, `ipython` and `python` to be accessible from the Windows command line.
4. Assuming you install Anaconda to `C:\Anaconda3`, adding that directory and `C:\Anaconda3\Scripts` to your **User Variables Path** will give the command line access to those commands.
5. Open a Windows command prompt as an Administrator
6. Install wxPython: `conda install -c anaconda wxpython`
7. Install SpiceyPy: `pip install spiceypy`
8. Install JPLEphem: `pip install jplephem`
9. Install de423: `pip install de423`

SNOPT

SNOPT is commercial nonlinear programming software licensed by Stanford Business Solutions. You will need to procure your own license.

SNOPT Licensing page: https://ccom.ucsd.edu/~optimizers/licensing/
Stanford Business Software SNOPT product page: http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm
Stanford Business Software SNOPT purchase page: http://www.sbsi-sol-optimize.com/asp/sol_snopt.htm

You will need to build SNOPT using mingw-64 unless your computer has Intel Visual Fortran. How to do this is outside the scope of this document.

This distribution of EMTGv9 is known to work with SNOPT 7.5 and 7.6. If you are using a different version then you can modify snoptProblemExtension.h in the src/InnerLoop folder to match the interface of the SNOPT that you have.

Install EMTG

We do not distribute an executable of the open-source EMTG. You will need to build it first as described later in this document, and then you can find the rest of the install instructions here.

1. Build the EMTG executable as described at the bottom of this guide. The executable will be placed in emtg/bin.
2. Place the SNOPT shared library in the EMTG bin directory: `.dll` in `EMTG\bin\libsnopt.dll`
3. Configure `EMTG\PyEMTG\PyEMTG.options` appropriately.

i) Make a copy of `EMTG\PyEMTG\PyEMTG-template.options` and name it `EMTG\PyEMTG\PyEMTG.options`
ii) `EMTG_path` must be set to the location of the `EMTGv9.exe` file
iii) `default_universe_path` is the EMTG Universe parent directory (i.e. not `EMTG\Universe\ephemeris_files`)

4. SPICE ephemeris kernels must be placed in `EMTG\Universe\ephemeris_files`.
5. A desktop shortcut can be created for the PyEMTG GUI as follows:
    1. Create a generic Windows shortcut
    2. Set the **Target** field to:

        ```
        "C:\Program Files\Anaconda3\pythonw.exe"
        <wherever_you_put_emtg>\EMTG\PyEMTG\PyEMTG.py
        ```

        Note the double quotes. The line break in the above command is a space between the two paths.
    3. Set the **Start in** to:

        ```
        <wherever_you_put_emtg>\EMTG\PyEMTG
        ```

    4. Set the icon to PyEMTG/Clemonaut.ico.

CMake

The latest stable CMake Windows installer can be obtained here.

IMPORTANT: If CMake was installed *before* MinGW, then the CMake installation repair will need to be run in order for CMake to recognize the MinGW installation.

Boost

We recommend that you acquire pre-built Boost libraries from https://sourceforge.net/projects/boost/files/boost-binaries/.

GNU GSL

The AMPL development team has created a CMake build system for the GNU Scientific Library. Their distribution can be found on Github here.

1. Download GSL, and create a build directory (e.g. `GSL\build`)
2. Use CMake to generate an MS Visual Studio Solution for GSL
3. Build GSL using MSVS

CSPICE

The download page for the C implementation of the CSPICE tookit is here. The CSPICE toolkit download typically comes pre-built, but building it again is performed by navigating to the CSPICE root directory and running the `makeall` script:

```
shell> .\makeall.bat
```

The generic CSPICE kernels can be found here. Any downloaded CSPICE kernels should be placed in `EMTG\Universe\ephemeris_files`.

EMTG requires that a leap second kernel file (`.tls`) and a frame kernel file (`.pck`) be present in the `ephemeris_files` directory as well as SPICE objects for any planetary bodies included in the optimization problem.

SNOPT

SNOPT is distributed with a pre-built MS Visual Studio Solution that assumes the use of Intel Visual Fortran. If you do not have Intel Visual Fortran, build SNOPT using mingw-64's gfortran. SNOPT comes with an appropriate makefile.

Randutils

Randutils is a third-party random number generator by Melissa O'Neill at MIT. You can find it here:
http://libutl.sourceforge.net/randutils_8hpp_source.html

Place this file in emtg/src/math.

EMTG

The core EMTG codebase is built using CMake. Before doing this, however, a local copy of the file `EMTG-Config-template.cmake` must be created and named `EMTG-Config.cmake`. This new copy must be customized based on your own directory structure.

The following variables must be specified in `EMTG-Config.cmake`

```
CSPICE_DIR
SNOPT_ROOT_DIR
BOOST_ROOT
GSL_PATH
```