EMTGv9 Windows Build Guide

Noble Hatten[*]Edwin Dove[†]Tim Sullivan [‡]

Revision Date	Author	Description of Change
January 10, 2023	Edwin Dove	Created public release version of windows install guide.
January 19, 2023	Noble Hatten	Edits to Edwin Dove's initial draft.
March 8, 2023	Tim Sullivan and Edwin Dove	Added SNOPT compile instructions and image captions.

[*]Aerospace Engineer, NASA Goddard Space Flight Center, Code 595
[†]Aerospace Engineer, NASA Goddard Space Flight Center, Code 595
[‡]Aerospace Engineer, The Aerospace Corporation

Contents

List of Acronyms

EMTG	Evolutionary Mission Trajectory Generator
GSL	GNU Scientific Library
GUI	Graphical User Interface
PEATSA	Python EMTG Automated Trade Study Application
SNOPT	Sparse Nonlinear OPTimizer
NLP	nonlinear program

1 Introduction

The purpose of this document is to describe:

- How to obtain the Evolutionary Mission Trajectory Generator (EMTG) codebase using the public NASA GitHub Repository (Section 3).
- How to obtain the dependencies of EMTG (Sections 2 and 4).
- How to compile EMTG from source on a Windows system (Section 5).

As may be inferred from the first bullet above, the target audience of this document is the public.

Other notes about the contents of this document:

- When listing Evolutionary Mission Trajectory Generator's (EMTG's) dependencies, specific versions of the dependencies are listed. These dependencies are known to work. The use of other versions of dependencies is not guaranteed to work and is not recommended.
- The instructions in this document are intended to be followed in the order in which they are given.

2 EMTG Prerequisites

The instructions indicate specific versions of the EMTG software and dependencies that have been verified to work. Other variations could work but proceed with caution when deviating from the instructions provided in this install guide.

The instructions in this document are intended for a computer running the following operating systems:

- Windows: Windows 10 64-bit installation.

Prerequisites

- EMTG v9 Public Release download (Section 3)
- External Dependencies (Section 4)
  - Python 3.7
  - Mambaforge
    - 22.9.0-2 Windows x86_64
  - MinGW 7.2.0 (variant: 64-bit, posix (threading), and seh (exception handler))
  - Compiler IDE
    - Microsoft Visual Studio 2022 Community Edition
  - CMake 3.23.1
  - Boost 1.79.0
  - GSL 2.4.0
  - CSPICE 64bit
  - SNOPT 7.6
  - randutils rev 2
  - Python packages
    - spiceypy 5.0.1
    - jplephem 2.17
    - matplotlib 3.5.2
    - numpy 1.21.6
    - scipy 1.7.3
    - wxPython 4.1.1
    - astropy 4.3.1

3 Obtaining the EMTG Git Repository

1. Navigate to `https://github.com/nasa/EMTG/`.
2. Verify that the branch selected is master

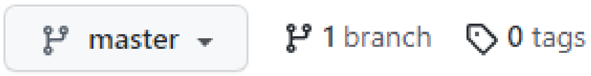
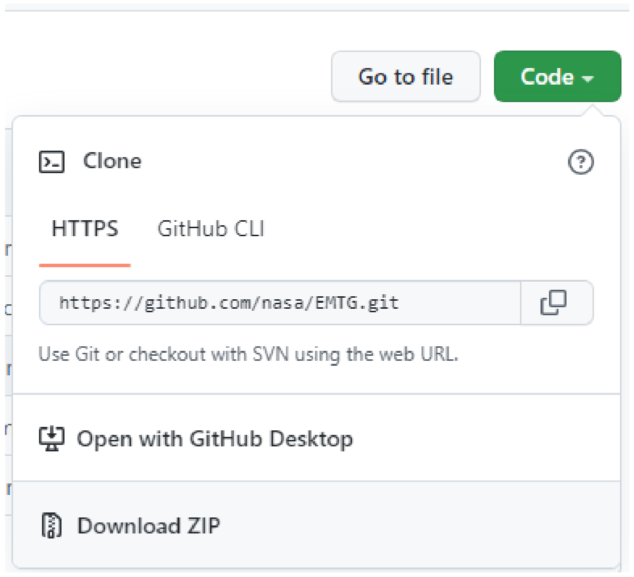


Figure 1: GitHub Master Branch Selection

3. Download the zip file by clicking the 'Code' button, select the 'Download Zip' option, and save to your desired location. Note that cloning the git repository is also an option, but this guide focuses on the simpler method of downloading a static copy of the repository using the 'Download Zip' option.



https://github.com/nasa/EMTG.git

Use Git or checkout with SVN using the web URL.

Figure 2: GitHub Download Prompt

   (a) The user should save this directory to their local machine into a destination with no spaces in the file path and record its location for future use (e.g. C:\SW_Installs\EMTG).

4. Locate the folder that contains the 'EMTG-Config-template.cmake' file.
   For the remainder of this document, this folder will be identified as **<EMTG_root_dir>**
5. Create the **<EMTG_root_dir>**\HardwareModels\ folder
6. Copy the default.emtg_launchvehicleopt and empty.ThrottleTable files from the **<EMTG_root_dir>**\testatron\HardwareModels\to the **<EMTG_root_dir>**\HardwareModels\ folder
7. Create the **<EMTG_root_dir>**\Universe\ folder

8. Create the **<EMTG_root_dir>**\Universe\ephemeris_files\ folder
9. Copy the *.emtg_universe files from the **<EMTG_root_dir>**\testatron\universe\ to the **<EMTG_root_dir>**\Universe\ folder
10. Open the **<EMTG_root_dir>**\Universe\Sun_barycenters.emtg_universe file in a text editor
11. Remove the Bennu line from the file
12. Save the file and close the text editor

4 Installing Dependencies

4.1 Random-Number Utilities

4.1.1 Purpose

EMTG depends on randutils for random number generation.
EMTG is known to work with revision 2 of randutils.

4.1.2 Download Location

The main page for the software distributions is in the following website:
`https://gist.github.com/imneme/540829265469e673d045`

The software revision needed for the EMTG version indicated in this guide can be obtained from the following location:
(In the event the url is no longer active, navigate to the aforementioned software website to find the specific version)
`https://gist.github.com/imneme/540829265469e673d045/8486a610a954a8248c12485fb4cfc390a5f5f854`

4.1.3 Dependency Installation Instructions

1. Click the last link in the 'Download Location' section
2. Click the 'Raw' button
3. Save the file to the <EMTG-root>\emtg\src\Math\ directory

4.2 Python

4.2.1 Purpose

EMTG depends on the Python interpreter and several packages to execute the PyEMTG Graphical User Interface (GUI) and other PyEMTG utilities. PyEMTG is known to be compatible with Python 3.7.[1] Therefore, it is strongly recommended that a Python 3.7 environment be created for PyEMTG. There are multiple ways to get Python but we strongly recommend/support Mambaforge. The intstructions in this guide will show a user how to install Python, a user with a preexisting Python install and knowledge on using the correct environments can utilize their own Python install.

4.2.2 Download Location

The main page for the software distributions is in the following website:
`https://github.com/conda-forge/miniforge/`

The software package needed for the EMTG version indicated in this guide can be obtained from the following location:
(In the event the url is no longer active, navigate to the aforementioned software website to find the specific version)
`https://github.com/conda-forge/miniforge/releases/tag/22.9.0-2/`

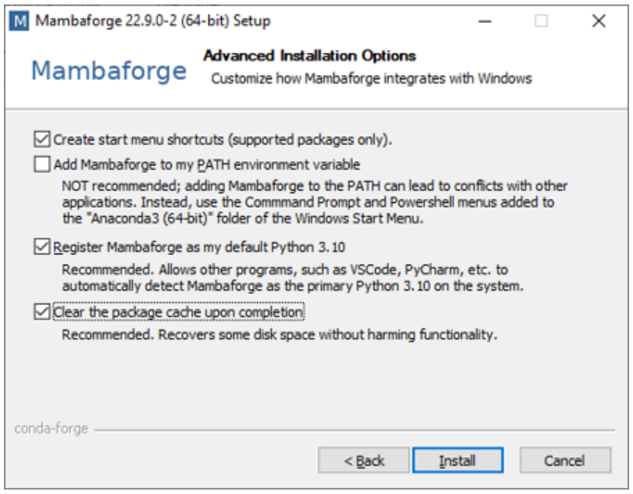
Instructions are given for creating an appropriate Python environment using the Mamba package manager. Information on obtaining and using Mamba is available at `https://mamba.readthedocs.io/en/latest/#`.

4.2.3 Dependency Installation Instructions

1. Download the Mambaforge-22.9.0-2-Windows-x86_64.exe installer and install locally.
   This is the installer with just mamba as the base environment

2. During installation, install mambaforge so it is installed just for the user.
   See the image below for the Windows example of settings to select.

[1]PyEMTG is known to specifically *not* be compatible with Python 3.10 because wxWidgets is not compatible with Python 3.10.



Mambaforge 22.9.0-2 (64-bit) Setup

Mambaforge

Advanced Installation Options

Customize how Mambaforge integrates with Windows

- [✓] Create start menu shortcuts (supported packages only).
- [ ] Add Mambaforge to my PATH environment variable

  NOT recommended; adding Mambaforge to the PATH can lead to conflicts with other applications. Instead, use the Commmand Prompt and Powershell menus added to the "Anaconda3 (64-bit)" folder of the Windows Start Menu.
- [✓] Register Mambaforge as my default Python 3.10

  Recommended. Allows other programs, such as VSCode, PyCharm, etc. to automatically detect Mambaforge as the primary Python 3.10 on the system.
- [✓] Clear the package cache upon completion

  Recommended. Recovers some disk space without harming functionality.

Figure 3: Mambaforge Installation Configuration

3. Open the MiniForge Prompt

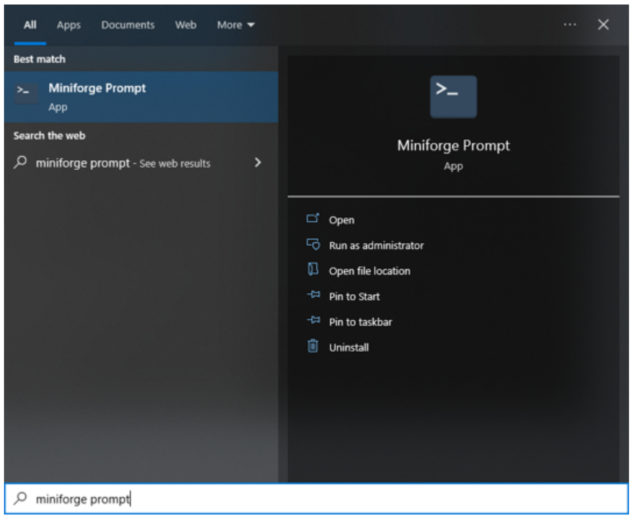


Figure 4: Open Mambaforge Command Prompt

4. Type the following command and press enter to create the specific python environment for EMTG:

```
mamba create -n PyEmtgEnv python=3.7
```

   (a) Accept changes when prompted

5. Type the following command and press enter to switch to the new python environment:

```
mamba activate PyEmtgEnv
```

6. Enter the following to confirm the python version selected is active:

   ```
   python --version
   ```

7. Install the following python packages

   - numpy 1.21.6
   - spiceypy 5.0.1
   - jplephem 2.17
   - scipy 1.7.3
   - matplotlib 3.5.2
   - wxPython 4.1.1
   - astropy 4.3.1

   (a) The following is the command to use to install the various package versions listed above:

   ```
   pip install packageName==#.#.#
   ```

   (b) Here is an example of the command for numpy:

   ```
   pip install numpy==1.21.6
   ```

8. Enter the following to verify the stated versions of the packages were installed:

   ```
   pip list
   ```

4.3 MinGW

4.3.1 Purpose

EMTG depends on the Sparse Nonlinear OPTimizer (SNOPT) nonlinear program (NLP) solver package, which is written in Fortran, and MinGW provides a free Fortran compiler and runtime library for Windows. These instructions assume that MinGW is used to compile the SNOPT library used by EMTG.
EMTG is known to work with MinGW 7.2.0.

4.3.2 Download Location

The main page for the software distributions is in the following website:
`https://sourceforge.net/projects/mingw-w64/files/`

The software package needed for the EMTG version indicated in this guide can be obtained from the following location: *(In the event the url is no longer active, navigate to the aforementioned software website to find the specific version)*
`https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win64/Personal%20Builds/mingw-builds/7.2.0/threads-posix/seh/x86_64-7.2.0-release-posix-seh-rt_v5-rev1.7z/download`

4.3.3 Dependency Installation Instructions

1. Unzip the file downloaded into a desired folder location for later use in an upcoming step.
   (a) Install 7zip if that is not already installed on your machine to extract the *.7z file.
2. Edit the user's Path environment variable to include the bin directory of MinGW so that the EMTG executable can find the Fortran runtime library during execution.
   (a) Press the Windows button.
   (b) Type "Edit environment variables for your account".

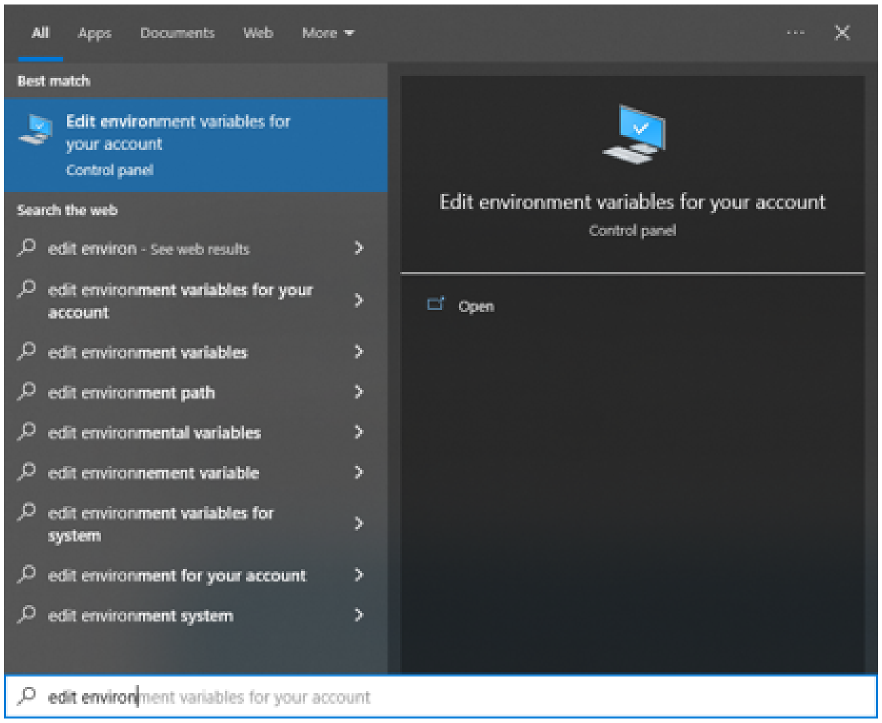
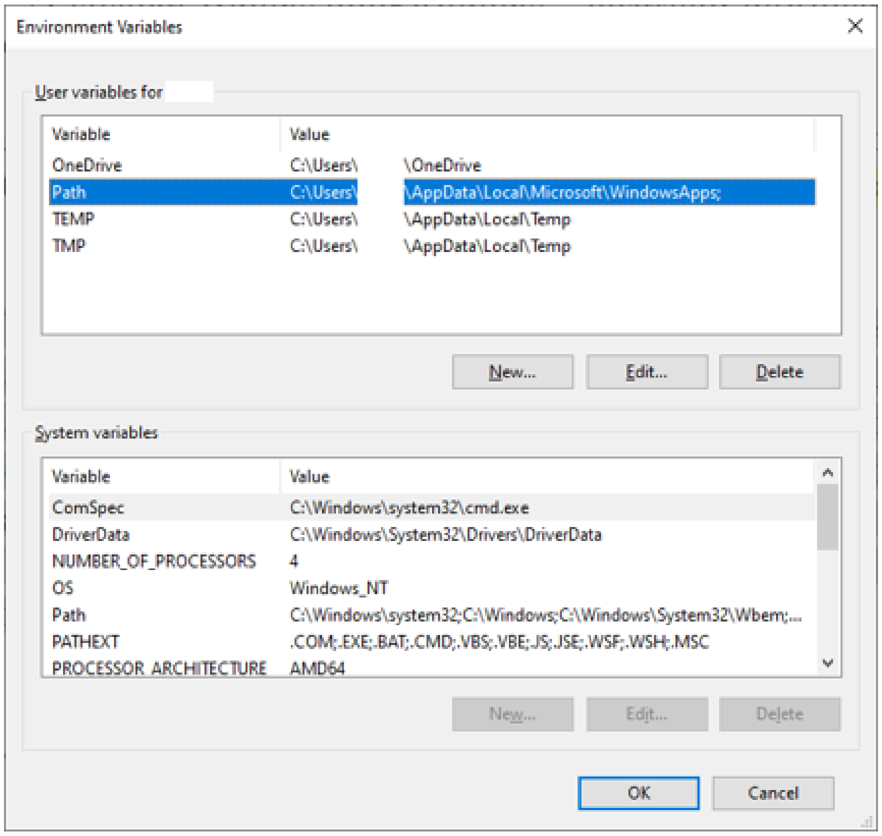

All
Apps
Documents
Web
More
Best match
Edit environment variables for your account
Control panel
Search the web
edit environ - See web results
edit environment variables for your account
edit environment variables
edit environment path
edit environmental variables
edit environnement variable
edit environment variables for system
edit environment for your account
edit environment system
Edit environment variables for your account
Control panel
Open
edit environment variables for your account

Figure 5: Launching Windows Environment Variables Menu

   (c) In the upper block of the new window, labeled "User variables for [User]", click on "Path", then click on the "Edit..." button.

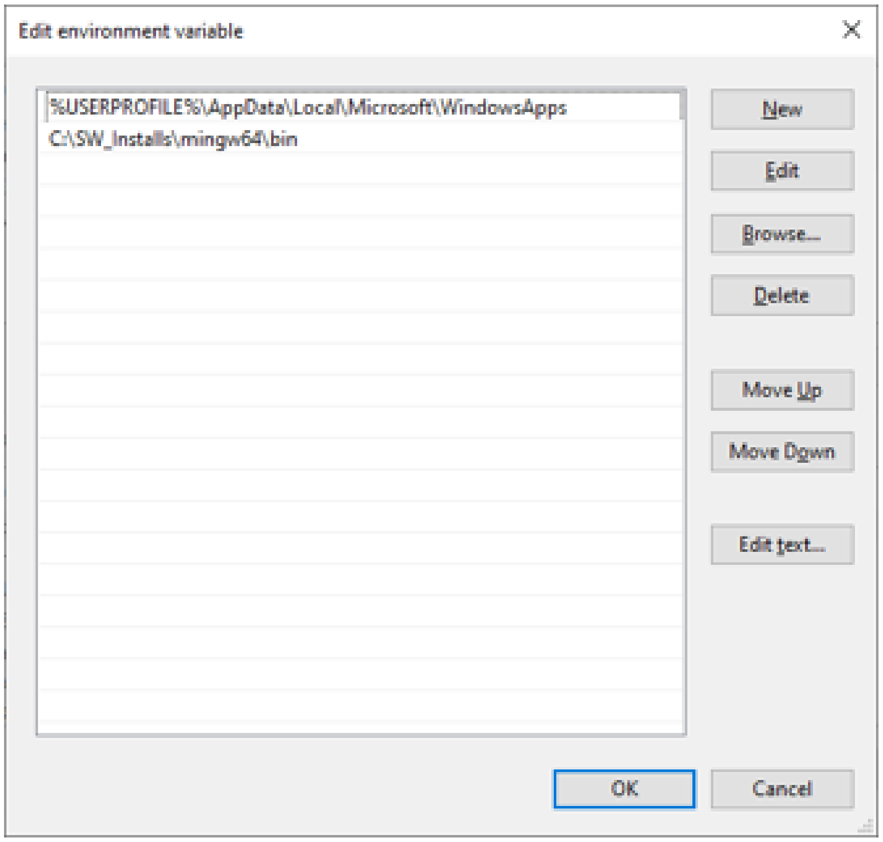


Variable	Value
OneDrive	C:\Users\ \OneDrive
Path	C:\Users\ \AppData\Local\Microsoft\WindowsApps;
TEMP	C:\Users\ \AppData\Local\Temp
TMP	C:\Users\ \AppData\Local\Temp

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	4
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windows\System32\Wbem;...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

Figure 6: Windows Environment Variables Menu

(d) In the popup window, click "New". In the new line, type the full path to the MinGW bin directory.



%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\SW_Installs\mingw64\bin

Figure 7: Windows Edit Environment Variable Menu

An example of the MinGW path would be

`C:\SW_Installs\mingw64\bin`

(e) Click the OK button in both dialog windows to save the edit to the Path.

4.4 Microsoft Visual Studio

4.4.1 Purpose

EMTG depends on a compiler to generate an executable from the source code that will function on a user's machine. EMTG is known to work with Microsoft Visual Studio 2022 Community Edition and this will be utilized throughout this guide. Alternative compilers can be used for users skilled enough to adapt the instructions in this guide for their compiler of choice.

4.4.2 Download Location

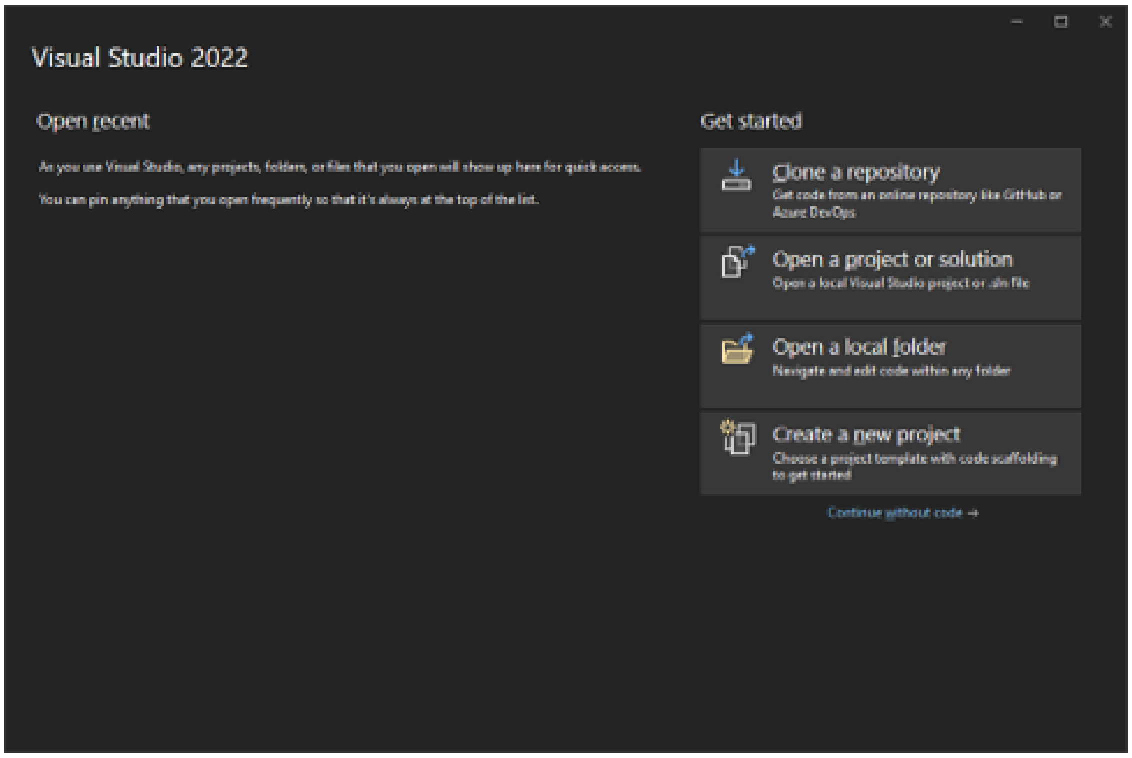
The main page for the software distributions is in the following website:
`https://visualstudio.microsoft.com/vs/community/`

The software package needed for the EMTG version indicated in this guide can be obtained from the following location:
(In the event the url is no longer active, navigate to the aforementioned software website to find the specific version)
`https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&channel=Release&version=VS2022&source=VSFeaturesPage&passive=true&tailored=cplus&cid=2031#cplusplus`.

Executing the Visual Studio installer requires elevated privileges.

4.4.3 Dependency Installation Instructions

1. Execute the downloaded file and proceed through the install instructions utilizing the defaults unless otherwise indicated in the following steps
2. Open Visual Studio once the installer completes
3. Follow the prompts until you reach the 'Get started' prompt

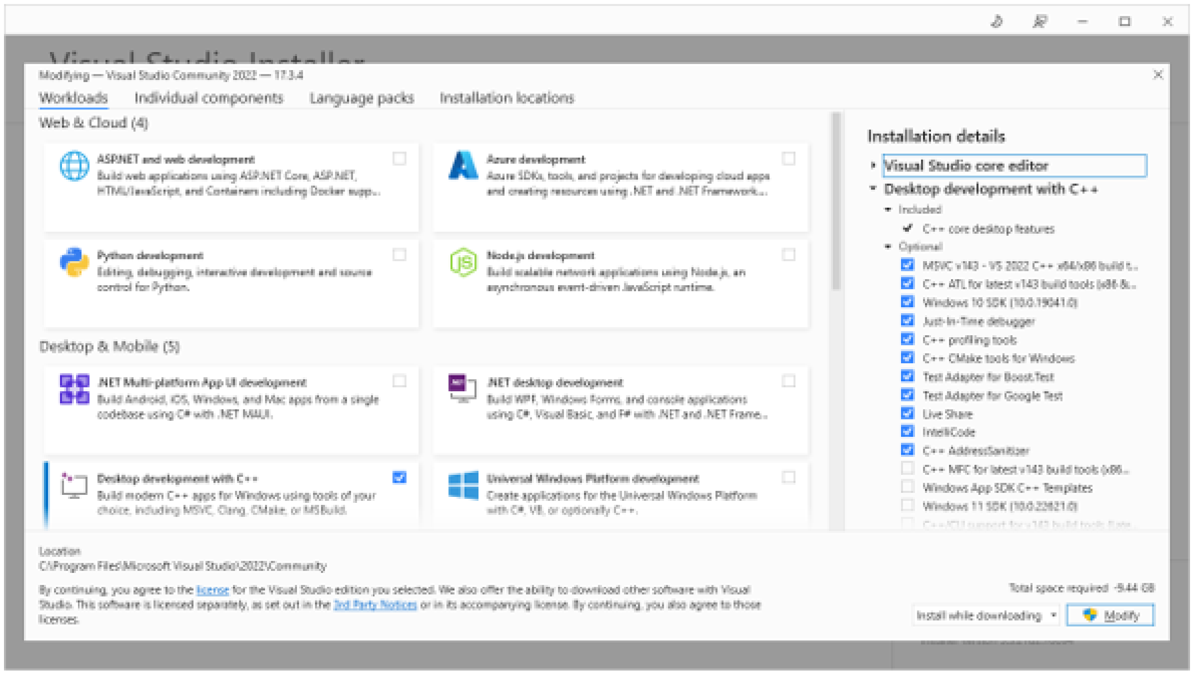


As you use Visual Studio, any projects, folders, or files that you open will show up here for quick access.
You can pin anything that you open frequently so that it's always at the top of the list.
Get code from an online repository like GitHub or Azure DevOps
Open a local Visual Studio project or .sln file
Navigate and edit code within any folder
Choose a project template with code scaffolding to get started

Figure 8: Visual Studios Startup Prompt

4. Click the "Continue without code" link
5. Select 'Tools' ->'Get Tools and Features ...' in the tool menu
6. Select the "Desktop development with C++" Workload

(a) The selection of any other Workload and optional components is at the user's discretion


Visual Studio Installer

Modifying — Visual Studio Community 2022 — 17.3.4

Workloads | Individual components | Language packs | Installation locations

Web & Cloud (4)

- [ ] **ASP.NET and web development** — Build web applications using ASP.NET Core, ASP.NET, HTML/JavaScript, and Containers including Docker supp...
- [ ] **Azure development** — Azure SDKs, tools, and projects for developing cloud apps and creating resources using .NET and .NET Framework...
- [ ] **Python development** — Editing, debugging, interactive development and source control for Python.
- [ ] **Node.js development** — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Desktop & Mobile (5)

- [ ] **.NET Multi-platform App UI development** — Build Android, iOS, Windows, and Mac apps from a single codebase using C# with .NET MAUI.
- [ ] **.NET desktop development** — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F# with .NET and .NET Frame...
- [x] **Desktop development with C++** — Build modern C++ apps for Windows using tools of your choice, including MSVC, Clang, CMake, or MSBuild.
- [ ] **Universal Windows Platform development** — Create applications for the Universal Windows Platform with C#, VB, or optionally C++.

Installation details

- Visual Studio core editor
- Desktop development with C++
  - Included
    - ✓ C++ core desktop features
  - Optional
    - [x] MSVC v143 - VS 2022 C++ x64/x86 build t...
    - [x] C++ ATL for latest v143 build tools (x86 &...
    - [x] Windows 10 SDK (10.0.19041.0)
    - [x] Just-In-Time debugger
    - [x] C++ profiling tools
    - [x] C++ CMake tools for Windows
    - [x] Test Adapter for Boost.Test
    - [x] Test Adapter for Google Test
    - [x] Live Share
    - [x] IntelliCode
    - [x] C++ AddressSanitizer
    - [ ] C++ MFC for latest v143 build tools (x86...
    - [ ] Windows App SDK C++ Templates
    - [ ] Windows 11 SDK (10.0.22621.0)

Location
C:\Program Files\Microsoft Visual Studio\2022\Community

By continuing, you agree to the license for the Visual Studio edition you selected. We also offer the ability to download other software with Visual Studio. This software is licensed separately, as set out in the 3rd Party Notices or in its accompanying license. By continuing, you also agree to those licenses.

Total space required -9.44 GB

Install while downloading | Modify

Figure 9: Visual Studios C++ Workload Selection

7. Click the 'Modify' button, then continue through the prompts to complete the installation options needed for the remainder of the EMTG install steps

4.5 CMake

4.5.1 Purpose

EMTG depends on a CMake-based build system to support the generation of the executable, in addition to a compiler.
EMTG is known to work with CMake 3.23.1.

4.5.2 Download Location

The main page for the software distributions is in the following website:
`https://cmake.org/download/`

The software package needed for the EMTG version indicated in this guide can be obtained from the following location:
(In the event the url is no longer active, navigate to the aforementioned software website to find the specific version)

`https://github.com/Kitware/CMake/releases/download/v3.23.1/cmake-3.23.1-windows-x86_64.zip`

4.5.3 Dependency Installation Instructions

1. Extract the zip into a destination with no spaces in the file path and record its location for future use (e.g. C:\SW_Installs\cmake)
2. Navigate to the cmake '/bin/' directory
3. Right click the "cmake-gui.exe" file and select 'Pin to Start' to easily find the application for later use

4.6 Boost

4.6.1 Purpose

EMTG depends on three components of Boost: filesystem, serialization, and system (and their dependencies). In addition, if a user wishes to build the EMTG PyHardware and Propulator components, then the python component of Boost is required.
EMTG is known to work with Boost 1.79.0.

4.6.2 Download Location

The main page for the software distributions is in the following website:
`https://www.boost.org/users/download/`

The software package needed for the EMTG version indicated in this guide can be obtained from the following location:
(In the event the url is no longer active, navigate to the aforementioned software website to find the specific version)
`https://sourceforge.net/projects/boost/files/boost-binaries/1.79.0/boost_1_79_0-msvc-14.3-64.exe/download/`

Executing the Boost installer requires elevated privileges.

4.6.3 Dependency Installation Instructions

1. Execute the downloaded file and proceed through the install instructions utilizing the defaults unless otherwise indicated in the following steps
   (a) When prompted for the install location, save to a destination with no spaces in the file path (e.g. C:\SW_Installs\boost_1_79_0) and record its location for future use

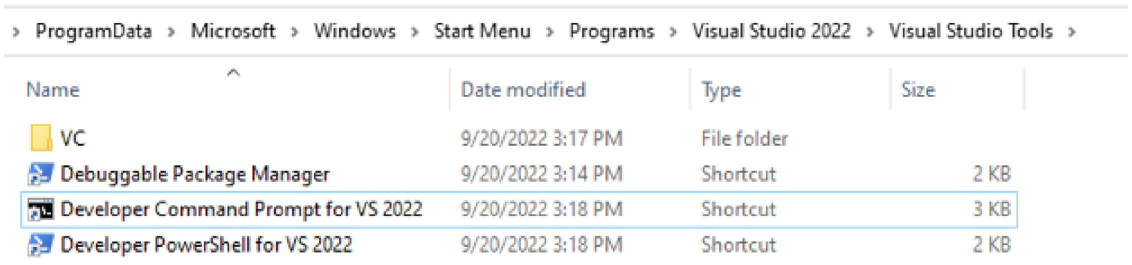
2. Creation of a "user-config.jam" file in your home directory
   (a) Navigate to your user directory (e.g. "C:\Users\MyName\")
   (b) Create a "user-config.jam" file
   (c) Copy the following contents to the file:

   ```
   using python : 3.7 ;
   using python : : C:\\Path\\To\\python.exe ;
   ```

   (d) Replace "`C:\Path\To\python.exe`" with the PyEmtgEnv environment that contains the python 3.7 executable then save the file
   (e.g. `C:\Users\MyName\mambaforge\envs\PyEmtgEnv\python.exe`)

   Note that the white space is important!
   For additional information, see the Boost documentation at `https://www.boost.org/doc/libs/1_79_0/libs/python/doc/html/building/configuring_boost_build.html`.

3. Open a Visual Studio Developer Command Prompt



› ProgramData › Microsoft › Windows › Start Menu › Programs › Visual Studio 2022 › Visual Studio Tools ›

Name	Date modified	Type	Size
VC	9/20/2022 3:17 PM	File folder	
Debuggable Package Manager	9/20/2022 3:14 PM	Shortcut	2 KB
Developer Command Prompt for VS 2022	9/20/2022 3:18 PM	Shortcut	3 KB
Developer PowerShell for VS 2022	9/20/2022 3:18 PM	Shortcut	2 KB

Figure 10: Visual Studio Command Prompt Launch Icon

   (a) Click the Start menu button
   (b) Click on the 'Visual Studio *' folder
   (c) Select the 'Developer Command Prompt *' option

4. Navigate to the root Boost directory (e.g. C: \SW_Installs \boost_1_79_0) in the command window

5. Type the following and hit Enter:

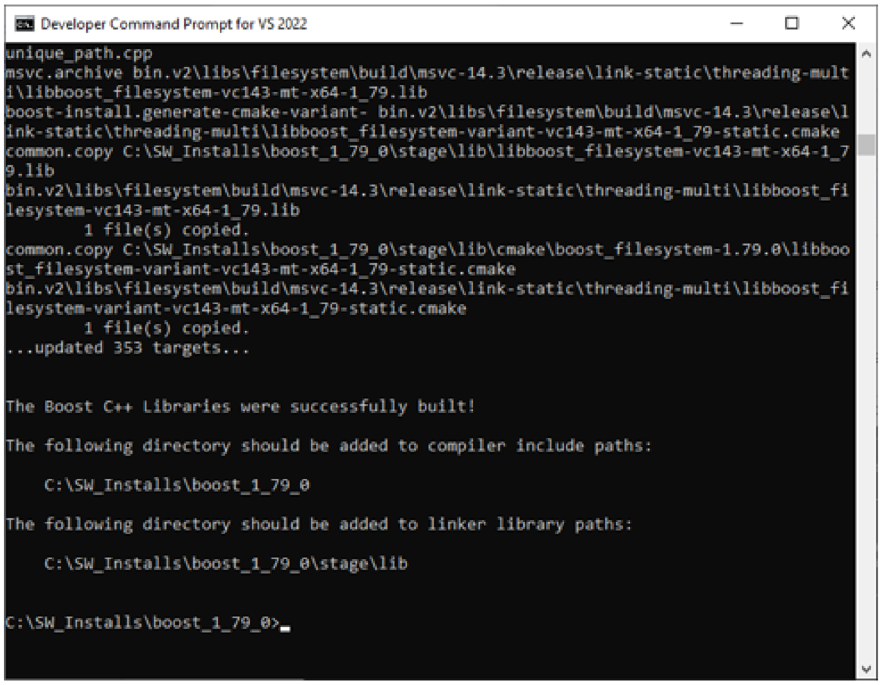
   ```
   bootstrap
   ```

6. Type the following, all on one line, and hit Enter:

   ```
   .\b2 address-model=64 link=static threading=multi runtime-link=shared

   --with-filesystem --with-serialization --with-system --with-python
   ```

If successful, the Boost root directory should now contain the directory path stage\lib\, and there should be several .lib files there.



```
unique_path.cpp
msvc.archive bin.v2\libs\filesystem\build\msvc-14.3\release\link-static\threading-mult
i\libboost_filesystem-vc143-mt-x64-1_79.lib
boost-install.generate-cmake-variant- bin.v2\libs\filesystem\build\msvc-14.3\release\l
ink-static\threading-multi\libboost_filesystem-variant-vc143-mt-x64-1_79-static.cmake
common.copy C:\SW_Installs\boost_1_79_0\stage\lib\libboost_filesystem-vc143-mt-x64-1_7
9.lib
bin.v2\libs\filesystem\build\msvc-14.3\release\link-static\threading-multi\libboost_fi
lesystem-vc143-mt-x64-1_79.lib
        1 file(s) copied.
common.copy C:\SW_Installs\boost_1_79_0\stage\lib\cmake\boost_filesystem-1.79.0\libboo
st_filesystem-variant-vc143-mt-x64-1_79-static.cmake
bin.v2\libs\filesystem\build\msvc-14.3\release\link-static\threading-multi\libboost_fi
lesystem-variant-vc143-mt-x64-1_79-static.cmake
        1 file(s) copied.
...updated 353 targets...


The Boost C++ Libraries were successfully built!

The following directory should be added to compiler include paths:

    C:\SW_Installs\boost_1_79_0

The following directory should be added to linker library paths:

    C:\SW_Installs\boost_1_79_0\stage\lib


C:\SW_Installs\boost_1_79_0>_
```

Figure 11: Boost Successful Build Output

4.7 GSL

4.7.1 Purpose

EMTG depends on GNU Scientific Library (GSL) for cubic-splining utilities.
EMTG is known to work with GSL 2.4.0.

4.7.2 Download Location

The main page for the software distributions is in the following website:

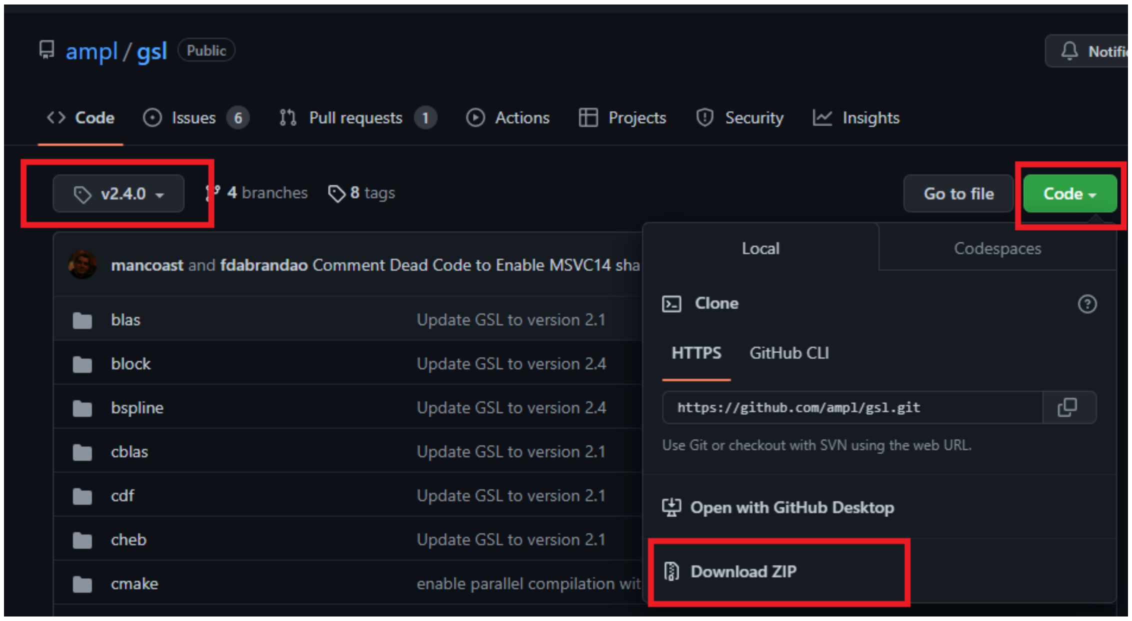
- `https://www.gnu.org/software/gsl/#downloading`
- `https://github.com/ampl/gsl/`

The software revision needed for the EMTG version indicated in this guide can be obtained from the following location:
(In the event the url is no longer active, navigate to the aforementioned software website to find

the specific version)
`https://github.com/ampl/gsl/tree/v2.4.0`

4.7.3 Dependency Installation Instructions

1. Download the zip file from the GitHub page by clicking the 'Code' button, select the 'Download Zip' option, and save to your desired location.



mancoast and fdabrandao Comment Dead Code to Enable MSVC14 sha

blas	Update GSL to version 2.1
block	Update GSL to version 2.4
bspline	Update GSL to version 2.4
cblas	Update GSL to version 2.1
cdf	Update GSL to version 2.1
cheb	Update GSL to version 2.1
cmake	enable parallel compilation wit

Figure 12: GSL Github

   (a) Extract the contents of the zip file into a destination with no spaces in the file path (e.g. C:\SW_Installs\gsl-2.4.0) and record its location for future use.

2. Open the CMake GUI

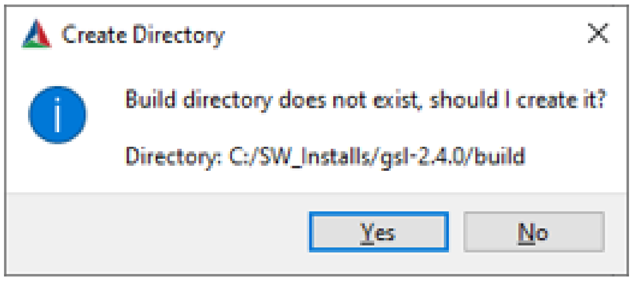
3. Set the CMake settings

   (a) Make sure that the "Advanced" and "Grouped" check boxes are checked.
   This makes it easier to see all options

   (b) In the "Where to build the binaries:" text box, put Path/To/gsl/build.
   It is OK if the "build" directory does not exist yet; CMake will create it.

   (c) In the "Where is the source code:" text box, put Path/To/gsl

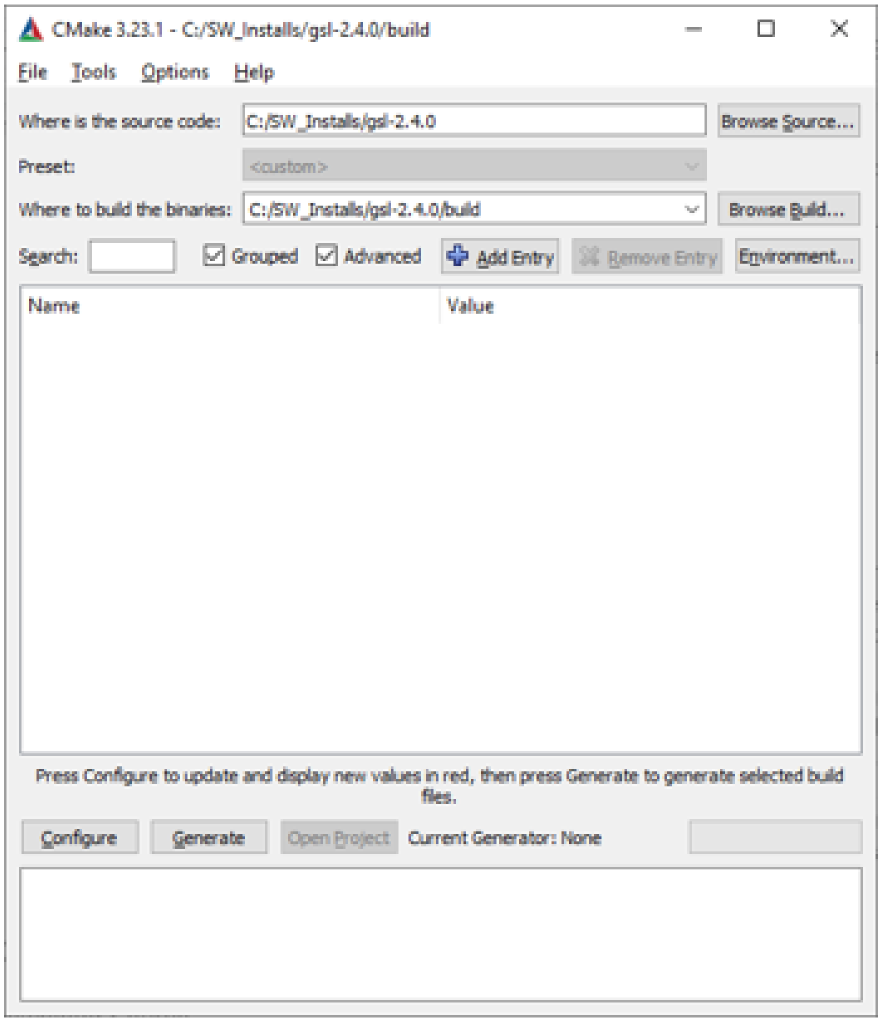
   (d) Click the 'Configure' button and create the build directory if prompted



Build directory does not exist, should I create it?

Directory: C:/SW_Installs/gsl-2.4.0/build

Figure 13: GSL Create New Build Directory



CMake 3.23.1 - C:/SW_Installs/gsl-2.4.0/build

Where is the source code: C:/SW_Installs/gsl-2.4.0

Preset: <custom>

Where to build the binaries: C:/SW_Installs/gsl-2.4.0/build

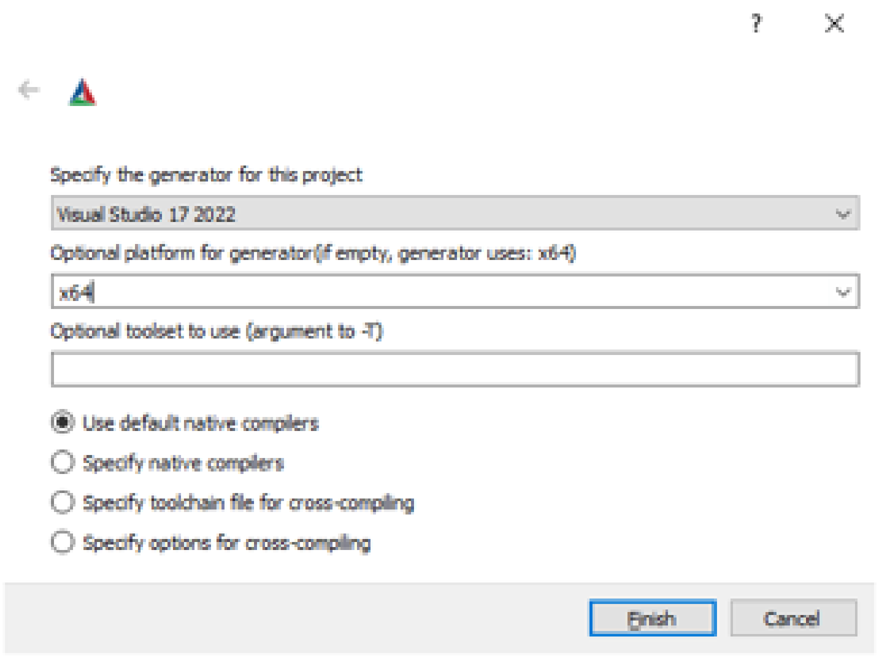
Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

Figure 14: GSL CMake Menu

(e) If this is the first time the 'Configure' button has been clicked since the cache has been cleared, a popup window asking which Toolset to use will appear. Select the correct Visual Studio installation. Make sure that the "x64" toolchain is selected.
This pop-up can also appear on clicking the 'Generate' button if the cache has been cleared.

  i. Verify that the "Configuring done" text is displayed in the log window


Specify the generator for this project

Visual Studio 17 2022

Optional platform for generator(if empty, generator uses: x64)

x64

Optional toolset to use (argument to -T)

(●) Use default native compilers
( ) Specify native compilers
( ) Specify toolchain file for cross-compiling
( ) Specify options for cross-compiling

Figure 15: CMake Visual Studio Settings

4. Click the ‘Generate’ button
   (a) Verify that the “Generating done” text is displayed in the log window
5. Click the ‘Open Project’ button in the CMake GUI to open the project in Visual Studio.
   (a) Right-click on ‘ALL BUILD’ in the Solution Explorer pane and select the ‘Set As Startup Project’ option.

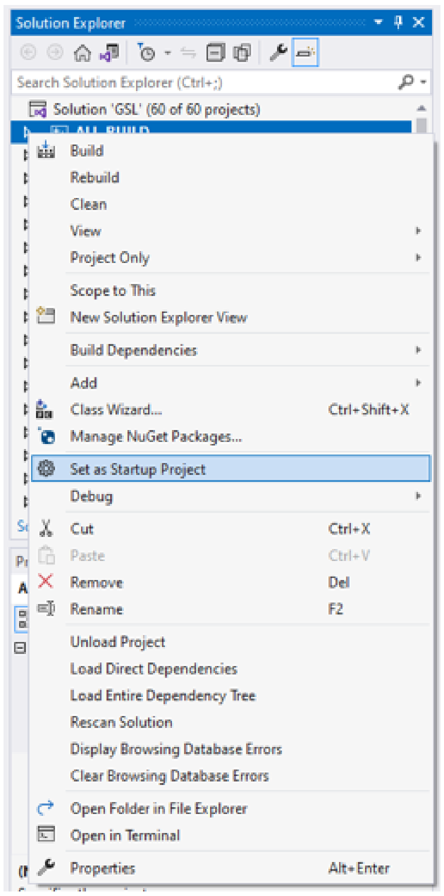


Figure 16: Visual Studio Select Startup Project

(b) In the toolbar, select the ‘Release’ option from the ’Solution Configurations’ dropdown.

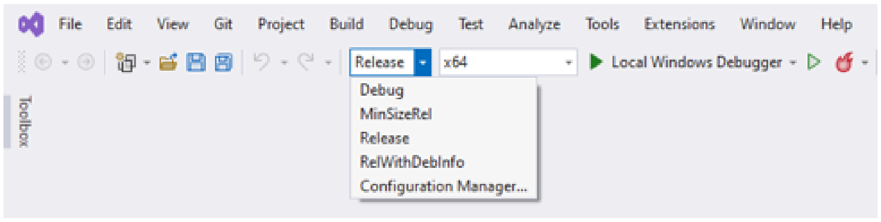
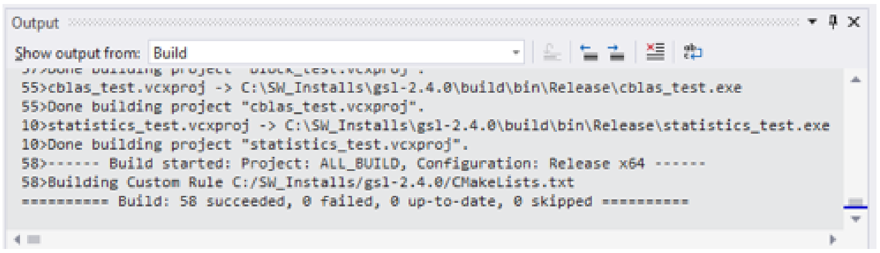


Figure 17: Visual Studio Release Solution Configuration

(c) Right-click on ‘ALL BUILD’ in the Solution Explorer then select the ‘Build’ option.

   i. A successful build will result in a message that says, in part, “0 failed” in the Visual Studio output window.



```
55>cblas_test.vcxproj -> C:\SW_Installs\gsl-2.4.0\build\bin\Release\cblas_test.exe
55>Done building project "cblas_test.vcxproj".
10>statistics_test.vcxproj -> C:\SW_Installs\gsl-2.4.0\build\bin\Release\statistics_test.exe
10>Done building project "statistics_test.vcxproj".
58>------ Build started: Project: ALL_BUILD, Configuration: Release x64 ------
58>Building Custom Rule C:/SW_Installs/gsl-2.4.0/CMakeLists.txt
========== Build: 58 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Figure 18: GSL Visual Studio Success Output

ii. The header files will be in Path/To/gsl/build/gsl.

iii. The library files will be in Path/To/gsl/build/Release.

4.8 CSPICE

4.8.1 Purpose

EMTG depends on CSPICE for ephemeris-lookup utilities.
EMTG is known to work with CSPICE N0067.

4.8.2 Download Location

The main page for the software distributions is in the following website:
`https://naif.jpl.nasa.gov/naif/toolkit_C.html`

The software revision needed for the EMTG version indicated in this guide can be obtained from the following location:
(In the event the url is no longer active, navigate to the aforementioned software website to find the specific version)

4.8.3 Dependency Installation Instructions

1. Once the zip file is downloaded, extract the contents of the zip file to a local folder with no space in the file path (e.g. C:\SW_Installs\cspice) for usage later in these instructions.

2. In addition to the CSPICE toolkit itself, EMTG requires ephemeris files that can be read by CSPICE. In order to get started with EMTG as quickly as possible, it is recommended to download a few commonly required ephemeris files now.

   (a) Navigate to `https://naif.jpl.nasa.gov/pub/naif/generic_kernels/spk/planets/` and download the de430.bsp file to the **<EMTG_root_dir>**\Universe\ephemeris_files\ folder

(b) Navigate to `https://naif.jpl.nasa.gov/pub/naif/generic_kernels/lsk/` and download the naif0012.tls file to the **<EMTG_root_dir>**\Universe\ephemeris_files\ folder

(c) Navigate to `https://naif.jpl.nasa.gov/pub/naif/generic_kernels/pck/` and download the pck00010.tpc and pck00011.tpc files to the **<EMTG_root_dir>**\Universe\ephemeris_files\ folder

4.9 SNOPT

4.9.1 Purpose

EMTG depends on the commercial NLP solver package SNOPT to perform gradient-based optimization. This installation guide is focused on SNOPT 7.6.
EMTG has interfaces known to work for SNOPT versions 7.5 and 7.6. These instructions assume that the user uses MinGW and specifically the gfortran module it is bundled with to compile SNOPT.

4.9.2 Download Location

The main page for the software distributions is in the following website:
For more information on obtaining SNOPT, see `http://www.sbsi-sol-optimize.com/asp/sol_product_snopt.htm`.

The SNOPT source code that is obtained from SBSI can be extracted to a folder on your local system. The root directory of this folder will be called **<SNOPT_root_dir>** for future references.

4.9.3 Dependency Installation Instructions

1. Create an empty `CMakeLists.txt` file and place it in the **<SNOPT_root_dir>** directory
2. Place the following content in the newly created `CMakeLists.txt` file then adjust the content of the file to make sure the #nl lines are no longer word wrapped and add line returns to make it easier to read:

```
cmake_minimum_required(VERSION 3.8.2) #nl

project(winsnopt) #nl

include(CMakeAddFortranSubdirectory) #nl

cmake_add_fortran_subdirectory(src
        PROJECT SNOPT
        LIBRARIES snopt
        LINK_LIBRARIES
        LINK_LIBS snopt
        CMAKE_COMMAND_LINE ""
        NO_EXTERNAL_INSTALL)

include_directories(interfaces/include) #nl

add_library(snopt_interface interfaces/src/snoptProblem.cpp interfaces/include/snoptProblem
    .hpp interfaces/include/snopt.h) #nl
STRING(REGEX REPLACE "/" "\\\\" SNOPT_DLL_ORIGIN "${CMAKE_CURRENT_BINARY_DIR}\\src\\
    libsnopt.dll") #nl
STRING(REGEX REPLACE "/" "\\\\" SNOPT_LIB_ORIGIN "${CMAKE_CURRENT_BINARY_DIR}\\src\\
    libsnopt.lib") #nl
STRING(REGEX REPLACE "/" "\\\\" SNOPT_DLL_DESTINATION "${CMAKE_CURRENT_BINARY_DIR}\\${
    CMAKE_CFG_INTDIR}\\libsnopt.dll") #nl
STRING(REGEX REPLACE "/" "\\\\" SNOPT_LIB_DESTINATION "${CMAKE_CURRENT_BINARY_DIR}\\${
    CMAKE_CFG_INTDIR}\\libsnopt.lib") #nl
message(STATUS ${BUILD_DESTINATION}) #nl
add_custom_command(TARGET snopt_interface POST_BUILD COMMAND copy /Y ${SNOPT_LIB_ORIGIN} ${
    SNOPT_LIB_DESTINATION} WORKING_DIRECTORY ${CMAKE_CURRENT_BINARY_DIR}) #nl
add_custom_command(TARGET snopt_interface POST_BUILD COMMAND copy /Y ${SNOPT_DLL_ORIGIN} ${
    SNOPT_DLL_DESTINATION} WORKING_DIRECTORY ${CMAKE_CURRENT_BINARY_DIR}) #nl
target_link_libraries(snopt_interface ${CMAKE_CURRENT_BINARY_DIR}/${CMAKE_CFG_INTDIR}/
    libsnopt.lib) #nl


link_directories(${CMAKE_CURRENT_BINARY_DIR}/${CMAKE_CFG_INTDIR}) #nl
add_executable(SNOPT_TOY_A ${CMAKE_SOURCE_DIR}/interfaces/cppexamples/sntoya.cpp) #nl
target_link_libraries(SNOPT_TOY_A ${CMAKE_CURRENT_BINARY_DIR}/${CMAKE_CFG_INTDIR}/
    snopt_interface.lib ${CMAKE_CURRENT_BINARY_DIR}/${CMAKE_CFG_INTDIR}/libsnopt.lib) #nl
```

3. Save and close the text file.
4. Create an empty `CMakeLists.txt` file and place it in the **<SNOPT_root_dir>**`\src\` directory
5. Place the following content in the `CMakeLists.txt` file then adjust the content of the file to make sure the #nl lines are no longer word wrapped and add line returns to make it easier to read:

```
cmake_minimum_required(VERSION 3.8.2) #nl

project(snopt Fortran) #nl

#file(GLOB SNOPT_FILES *.f) #nl

set(SNOPT_FILES ${SNOPT_FILES} ../interfaces/src/snopt_wrapper.f90
                               ../interfaces/src/sqopt_wrapper.f90)


set(SNOPT_FILES ${SNOPT_FILES} snblas.f snfilewrapper.f
         sqopt.f    snoptq.f   sq02lib.f  sn03prnt.f sn04wrap.f sn04stats.f sn10mach.f
         sn12ampl.f sn17util.f sn20amat.f sn25bfac.f sn27lu.f
         sn30spec.f sn35mps.f  sn37wrap.f sn40bfil.f sn50lp.f sn55qp.f
         sn56qncg.f sn57qopt.f sn65rmod.f)
```

```
set(SNOPT_FILES ${SNOPT_FILES}
        sqopt.f    snopta.f   snoptb.f   snoptc.f    snoptch.f
        snoptq.f   npopt.f    snctrl.f
        sq02lib.f  sn02lib.f  np02lib.f  sn04wrap.f  sn04stats.f
        sn05wrpa.f sn05wrpb.f sn05wrpc.f sn05wrpch.f sn05wrpn.f sn10mach.f
        sn12ampl.f sn17util.f sn20amat.f sn25bfac.f  sn27lu.f
        sn30spec.f sn35mps.f  sn37wrap.f sn40bfil.f  sn50lp.f    sn55qp.f
        sn56qncg.f sn57qopt.f sn60srch.f sn65rmod.f  sn70nobj.f  sn80ncon.f
        sn82qn.f   sn83bfgs.f sn87sopt.f snfilewrapper.f)

add_library(snopt ${SNOPT_FILES}) #nl
target_link_libraries(snopt gfortran) #nl
```

6. Save and close the text file.
7. Open the `src/sn87sopt.f` file in a text editor
   (a) Navigate to approximately line 2709 and comment (Fortran uses the ! character for comments) out the line that reads

   `primalInf = primalInf/max(xNorm , one)`

   Leaving this line uncommented can incorrectly mark certain solutions as feasible in SNOPT.

   (b) Save and close the file
8. Open the `src/sn70nobj.f` file in a text editor
   (a) Navigate to approximately line 235

   (b) Replace the following line ...

   `call dload ( nF, zero, Elem, 1 )`

   ... with ...

   `call iload ( nF, 0, Elem, 1 )`

   The original line could cause an error when attempting to build SNOPT in Windows using the MinGW console as opposed to using the CMake and Visual Studio method.

   (c) Save and close the file
9. Ensure the Windows Path environment variable contains the path to the MinGW bin directory and save changes. *It is recommended that this be placed at or near the top of the list of directories in path. Refer to the MinGW section for additional detail.*
10. Open the CMake GUI
11. Set "Where is the source code" to the **<SNOPT_root_dir>** directory
12. Set "Where to build the binaries" to **<SNOPT_root_dir>**\build\
    It is OK if the "build" directory does not exist yet; CMake will create it.

13. Click "Configure" and make sure you are using Visual Studio 2022 toolchain and x64 which should be the CMake default.
The MINGW_GFORTRAN CMake variable should be the GFortran executable in your MinGW bin directory. If the Intel Fortran compiler, or something other than GFortran is shown here, the build will likely fail. Windows will often try to default to the Intel Fortran Compiler if installed so ensure MinGW\bin\is the first directory in Path and consider removing the Intel Fortran compiler directories from Path.

14. Click "Generate"

15. Click "Open Project" to open Visual Studio

16. Set build mode to Release.

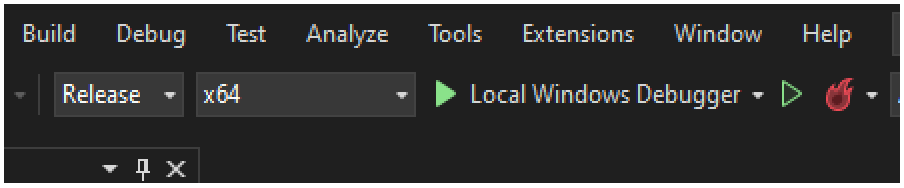


Figure 19: Visual Studio Build Mode

17. Build the projects in the following order:

    - SNOPT_build
    - snopt_interface
    - ALL_BUILD

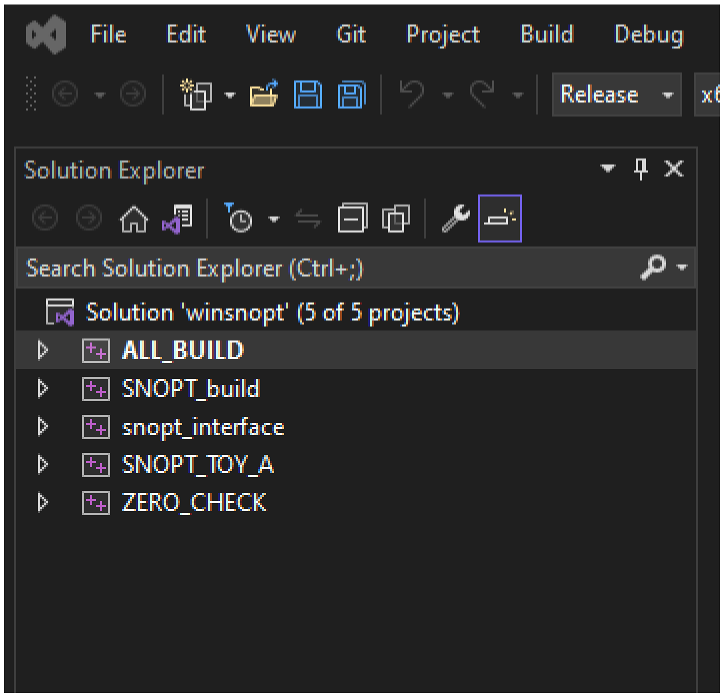
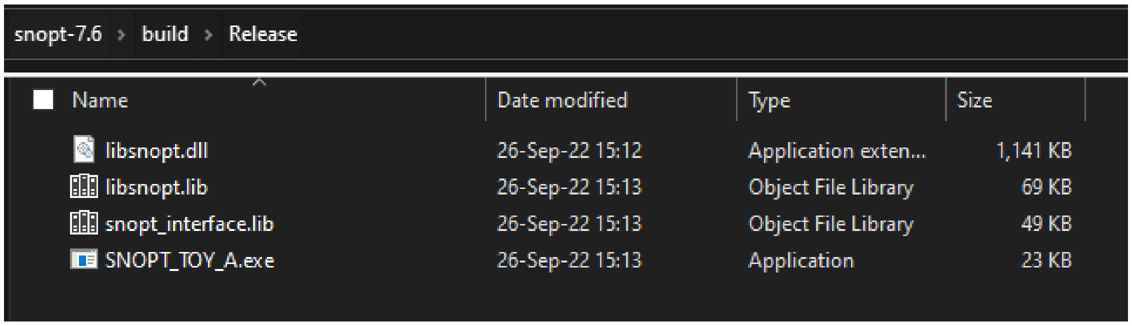


Figure 20: Visual Studio SNOPT Solution Explorer

18. Verify the ouput window message says in part, "0 failed, 0 skipped". The `SNOPT\build\Release` directory should now look like the following image:



```
snopt-7.6 > build > Release

Name                  Date modified     Type                 Size
libsnopt.dll          26-Sep-22 15:12   Application exten...  1,141 KB
libsnopt.lib          26-Sep-22 15:13   Object File Library      69 KB
snopt_interface.lib   26-Sep-22 15:13   Object File Library      49 KB
SNOPT_TOY_A.exe       26-Sep-22 15:13   Application              23 KB
```

Figure 21: SNOPT Release Directory Contents

5 Building EMTG

This section describes how to build the EMTG executable after all dependencies have been acquired and set up.

5.1 The EMTG-Config.cmake File

Follow the following instructions:

1. Navigate to the **<EMTG_root_dir>**\directory
2. Copy the 'EMTG-Config-template.cmake' file and rename to 'EMTG-Config.cmake'
3. Open 'EMTG-Config.cmake' in a text editor
   This file is heavily commented with specific instructions. The user must know where they installed CSPICE, SNOPT, Boost, and GSL.
4. Set the CSPICE DIR variable to the full path of the CSPICE root directory.
   (a) If CSPICE were placed in C:\SW_Installs\cspice, then the user would type in the EMTG-Config.cmake file

   ```
   set(CSPICE_DIR C:/SW_Installs/cspice)
   ```

   Note the use of forward slashes in the file path even though the Windows file system uses backslashes.
5. Set the SNOPT_ROOT_DIR variable to the full path of the SNOPT root directory.

   (a) If SNOPT were placed in C:\SW_Installs\SNOPT76, then the user would type in the EMTG-Config.cmake file

   ```
   set(SNOPT_ROOT_DIR C:/SW_Installs/SNOPT76)
   ```

   Note the use of forward slashes in the file path even though the Windows file system uses backslashes.

6. Set the BOOST_ROOT, BOOST_INCLUDE_DIR, and BOOST_LIBRARY_DIRS variables.

   (a) BOOST_ROOT is the full path to the root directory of Boost.

   (b) BOOST_INCLUDE_DIR is the full path to the directory that contains the Boost header files. For Boost 1.79.0, this is a subdirectory in the Boost root directory called Boost.

   (c) BOOST_LIBRARY_DIRS is the full path to the directory that holds the Boost libraries after they are built. For Boost 1.79.0, this is a subdirectory in the Boost root directory called stage\lib.

   If Boost were placed in C:\SW_Installs\boost_1_79_0, then the user would type in the EMTG-Config.cmake file

   ```
   set(BOOST_ROOT C:/SW_Installs/boost_1_79_0)
   set(BOOST_INCLUDE_DIR ${BOOST_ROOT}/boost)
   set(BOOST_LIBRARY_DIRS ${BOOST_ROOT}/stage/lib)
   ```

   Note the use of forward slashes in the file path even though the Windows file system uses backslashes.

7. Set the GSL_PATH variable to the full path to the GSL build directory (where the libraries are located). For example, if GSL were placed in C:\SW_Installs\gsl-2.4.0, then the user would type in the EMTG-Config.cmake file

   ```
   set(GSL_PATH C:/SW_Installs/gsl-2.4.0/build)
   ```

   Note the use of forward slashes in the file path even though the Windows file system uses backslashes.

8. Save the changes made to the EMTG-Config.cmake file

5.2 Setting CMake Options

The CMake program is used to set additional options for the EMTG build.

1. Open the CMake GUI.

2. Make sure that the "Advanced" and "Grouped" check boxes are checked.
   This makes it easier to see all options.

3. In the "Where to build the binaries:" text box, put **<EMTG_root_dir>**/build.
   It is OK if the "build" directory does not exist yet; CMake will create it.

4. In the "Where is the source code:" text box, put **<EMTG_root_dir>**.

5. Click "Configure" button.

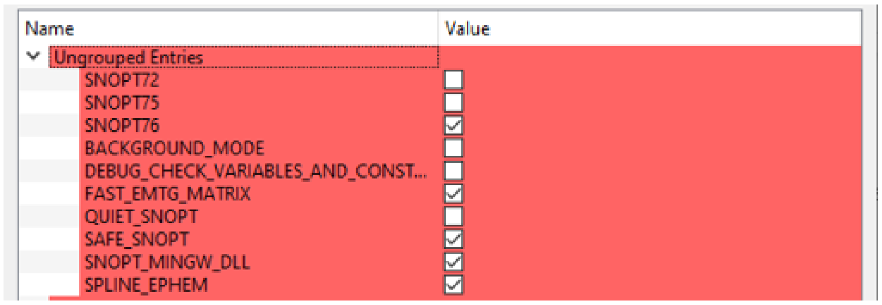
6. Select the options you want for each "Name" in the "Value" column.

   (a) Select and confirm the appropriate SNOPT settings by following the steps below:

      i. Expand the 'Ungrouped Entries' section

      ii. Check that the SNOPT version indicated in the SNOPT section 4.9 is checked and the other versions are unchecked

      iii. Ensure SNOPT_MINGW_DLL is checked



Name	Value
Ungrouped Entries	
SNOPT72	☐
SNOPT75	☐
SNOPT76	✓
BACKGROUND_MODE	☐
DEBUG_CHECK_VARIABLES_AND_CONST...	☐
FAST_EMTG_MATRIX	✓
QUIET_SNOPT	☐
SAFE_SNOPT	✓
SNOPT_MINGW_DLL	✓
SPLINE_EPHEM	✓

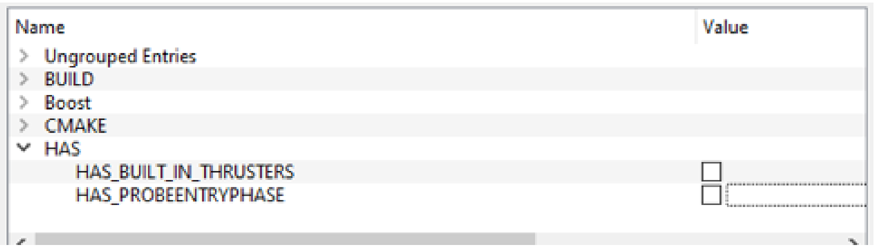
Figure 22: EMTG CMAKE Ungrouped Entries Selection

   (b) Ensure all the options under "HAS" are unchecked:
   Note that for the current public release these options are not fully functional

      i. Expand the "HAS" section

      ii. Uncheck the HAS_BUILT_IN_THRUSTERS option

      iii. Uncheck the HAS_PROBEENTRYPHASE option



Name	Value
Ungrouped Entries	
BUILD	
Boost	
CMAKE	
HAS	
HAS_BUILT_IN_THRUSTERS	☐
HAS_PROBEENTRYPHASE	☐

Figure 23: EMTG CMAKE Has Selection

   (c) Ensure Boost was detected by following the steps below:

      i. Expand the "Boost" section

      ii. Verify the directories have autopopulated to the correct location in which Boost is installed

   (d) Disable the pyhardware and propulator utility by following the steps below:

      i. Expand the "BUILD" section

ii. Uncheck the "BUILD PROPULATOR" option
iii. Uncheck the "BUILD PYHARDWARE" option

7. Click "Configure"

8. Click "Generate"

9. Click the "Open Project" button
   This will launch the Visual Studio application

5.3 Building EMTG in Visual Studio

Following the CMake configure-and-generate steps (Section 5.2), open the project in Visual Studio. This can be done by clicking the "Open Project" button in the CMake GUI. Then, perform the following actions:

1. Right-click on "EMTGv9" in the Solution Explorer then click "Set As Startup Project".

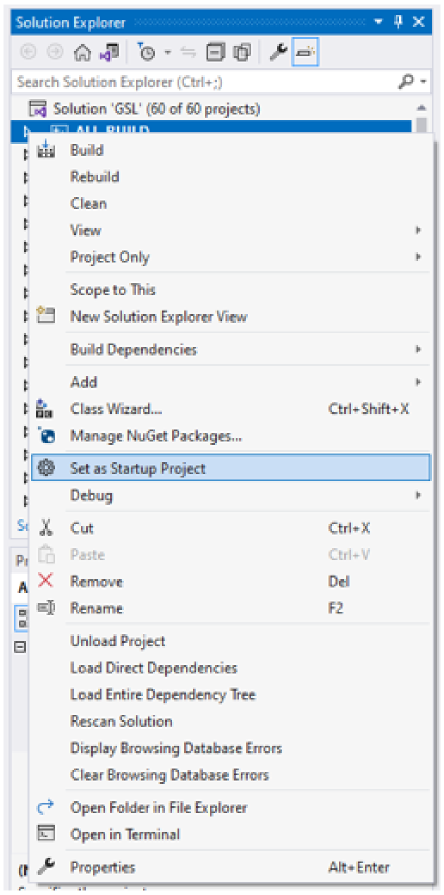


Figure 24: Visual Studio Select Startup Project

2. In the toolbar, select the "Release" option from the "Solution Configurations" dropdown.

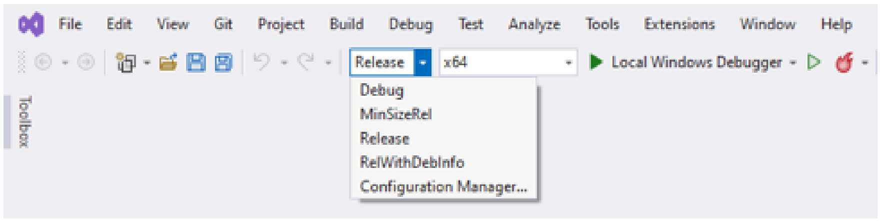


Figure 25: Visual Studio Release Solution Configuration

Do not select "Debug" unless you actually want to debug EMTG!

3. Right-click on "EMTGv9" in the Solution Explorer then click "Build".
4. Verify the build is successful.
   The log output will result in a message that says, in part, "0 failed" in the Visual Studio output window. In addition, Path/To/EMTG/Repo/bin will contain EMTGv9.exe and libsnopt.dll.

6 Executing EMTG

6.1 Using PyEMTG GUI

PyEMTG is a set of Python scripts that provide capabilities such as a GUI for EMTG and automated trade study utilities (i.e., Python EMTG Automated Trade Study Application (PEATSA)). Full PyEMTG documentation is contained in the EMTG repository's /PyEMTG/docs/ directory. In that directory, see PyEMTG_docs_readme.pdf for an overview of available PyEMTG documentation, which includes a reference to the PyEMTG User's Guide.

The following actions are instructions on executing PyEMTG on Windows and running the default mission:

1. Navigate to the **<EMTG_root_dir>**\PyEMTG\ folder
2. Copy the PyEMTG-template.options file to a new file called PyEMTG.options
3. Open the newly copied PyEMTG.options file in a text editor
4. Replace the 'EMTG_path' value with the **<EMTG_root_dir>**/bin/EMTGv9.exe path
   This path needs to have forward slashes (/)
5. Replace the 'default_universe_path' value with the **<EMTG_root_dir>**/Universe path
   This path needs to have forward slashes (/) and do not have a trailing forward slash
6. Save the PyEMTG.options file and close the text editor
7. Navigate to the 'Miniforge Prompt' window where the PyEmtgEnv environment is active
8. Type the following in the prompt, replacing **<EMTG_root_dir>** with your EMTG directory path, and press enter:

```
cd <EMTG_root_dir>\PyEMTG\
```

9. Type the following in the prompt and press enter to Launch PyEMTG:

   ```
   python PyEMTG.py
   ```

10. Click File ->New ->Mission option
    The default Mission values will be loaded.

11. Click on the 'Spacecraft Options' tab

12. Click the 'ellipsis' button associated with the 'Hardware library path' field and select the **<EMTG_root_dir>**\HardwareModels\ folder and click 'Select Folder'

13. Add a '\' to the end of the string in the 'Hardware library path' field if one is not already there

14. Click the 'ellipsis' button associated with the 'Launch vehicle library file' field and select the default.emtg_launchvehicleopt file and click 'Open'

15. Click the 'Launch vehicle' dropdown and select 'ExampleRocket'

16. Click on the 'Physics Options' tab

17. Click the 'Default' button associated with the 'Universe folder' field
    This will populate the field with the value from the PyEMTG.options file.

18. Verify the 'Leap seconds kernal' and 'Frame kernel' filenames are the latest .tls and .tpc files, respectively available, in the **<EMTG_root_dir>**\Universe\ephemeris_files\ folder

19. Click on the 'Journey Options' tab

20. Click the 'ellipsis' button associated with the 'Central body' field

21. Select the Sun_barycenters.emtg_universe and click the 'Open' button

22. Click File ->Run option

23. Rename the mission file to the name of your choosing and click the 'Save' button
    At this point, PyEMTG will execute EMTG and a new command window will open that will display the output generated from EMTG

6.2 Using Command Line

On a local Windows machine, EMTG is most often executed via the PyEMTG GUI. However, it may also be executed from the command line, from a script, or using the Windows Explorer. In the first two instances, EMTG is executed by invoking the EMTGv9.exe command with the name of an EMTG options file ("*.emtgopt") as the only command-line argument. To execute via Windows Explorer, open two Windows Explorer windows. In one, navigate to Path/To/EMTG/Repo/bin. In the other, navigate to the location of the EMTG options file to be executed. Drag and drop the EMTG options file onto EMTGv9.exe to execute.