Building femm-qt on Windows

D. Meeker

May 19, 2026

1 Overview

femm-qt is a cross-platform port of FEMM (Finite Element Method Magnetics) using Qt 6 for the GUI and CMake/Ninja for the build system. This document describes how to build it from source on Windows using the Qt Online Installer and MinGW (no Visual Studio required).

The build produces six executables:

- `femm_gui.exe` — the main GUI application
- `femm_headless.exe` — headless IPC server (no GUI)
- `fkn.exe` — magnetics solver
- `belasolv.exe` — electrostatics solver
- `hsolv.exe` — heat flow solver
- `csolv.exe` — current flow solver

2 Prerequisites

2.1 Qt 6 with MinGW

Download and run the Qt Online Installer from `https://www.qt.io/download-qt-installer-oss`. You will need a free Qt account.

In the installer, select:

- A recent Qt version (e.g. Qt 6.10.3)
- Under that version, check `MinGW 64-bit`
- Under **Developer and Designer Tools**, check:
  - `MinGW 13.1.0 64-bit` (or the version matching your Qt)
  - `CMake`
  - `Ninja`

The default install location is `C:\Qt`. The paths below assume this default; adjust if you installed elsewhere.

After installation, verify the tools are accessible. Open a command prompt and check:

```
C:\Qt\Tools\mingw1310_64\bin\g++ --version
C:\Qt\Tools\CMake_64\bin\cmake --version
C:\Qt\Tools\Ninja\ninja --version
```

2.2 Git (Optional)

If you are cloning from a repository rather than using a source zip, install Git for Windows from `https://git-scm.com/download/win`.

3 Getting the Source

If you have a source zip, extract it:

```
unzip femm-qt-src.zip
cd femm-qt
```

Or clone from a repository:

```
git clone <repository-url> femm-qt
cd femm-qt
```

4 Building

From the `femm-qt` directory, run the build script:

```
win_build.bat
```

This auto-detects Qt, MinGW, CMake and Ninja under `C:\Qt`, configures the project with CMake, and builds it with Ninja. All executables are placed in `build\bin\`.

To wipe the build directory and rebuild from scratch, pass `clean`:

```
win_build.bat clean
```

If the script cannot find your Qt or MinGW installation — for example, if Qt is installed somewhere other than `C:\Qt` — see *Manual configuration* under Troubleshooting.

5 Running

Launch the GUI:

```
bin\femm_gui.exe
```

Or the headless server:

```
bin\femm_headless.exe
```

5.1 Setting FEMM_DIR

The language bindings (OctaveFEMM, MathFEMM, pyFEMM, JuliaFEMM) locate the FEMM executables via the `FEMM_DIR` environment variable. Set it to the `bin` directory:

```
set FEMM_DIR=C:\path\to\femm-qt\build\bin
```

To make this persistent, add it via System Properties → Environment Variables, or in PowerShell:

```
[Environment]::SetEnvironmentVariable("FEMM_DIR",
    "C:\path\to\femm-qt\build\bin", "User")
```

6 Using with MATLAB or Octave

6.1 MATLAB

Set `FEMM_DIR` before launching MATLAB, or set it within MATLAB:

```
setenv('FEMM_DIR', 'C:\path\to\femm-qt\build\bin')
```

Add the `octavefemm` directory to the MATLAB path:

```
addpath('C:\path\to\femm-qt\octavefemm')
```

Then call `openfemm` to launch femm-qt.

6.2 Octave

Octave on Windows requires a modern JDK for socket communication (Adoptium Temurin 25 or later is recommended; MATLAB's bundled Java 8 may segfault with Octave). Set the `JAVA_HOME` environment variable to point at the JDK, then launch Octave.

```
set JAVA_HOME=C:\Program Files\Eclipse Adoptium\jdk-25
```

Add the `octavefemm` directory to the Octave path and call `openfemm`.

7 Troubleshooting

7.1 Manual configuration

`win_build.bat` auto-detects the toolchain under `C:\Qt`. If it picks the wrong Qt version, or your tools are installed elsewhere, set the `QT_DIR` and `MINGW_DIR` variables near the top of the script to the correct paths and run it again.

To configure entirely by hand instead, create the build directory and invoke CMake and Ninja directly:

```
mkdir build
cd build
cmake .. -G Ninja -DCMAKE_BUILD_TYPE=Release ^
    -DCMAKE_PREFIX_PATH="C:/Qt/6.10.3/mingw_64" ^
    -DCMAKE_C_COMPILER="C:/Qt/Tools/mingw1310_64/bin/gcc.exe" ^
    -DCMAKE_CXX_COMPILER="C:/Qt/Tools/mingw1310_64/bin/g++.exe" ^
    -DCMAKE_MAKE_PROGRAM="C:/Qt/Tools/Ninja/ninja.exe"
ninja
```

Adjust the version numbers in the paths to match what you installed (e.g. `6.10.3`, `mingw1310_64`). Use forward slashes in paths, even on Windows.

7.2 CMake cannot find Qt

If CMake reports that it cannot find Qt6, verify that `-DCMAKE_PREFIX_PATH` points to the correct Qt installation. Look for a directory like `C:\Qt\6.10.3\mingw_64` containing `lib\cmake\Qt6`.

7.3 Wrong compiler picked up

If CMake finds a different compiler (e.g. MSVC), make sure you are specifying `-DCMAKE_C_COMPILER` and `-DCMAKE_CXX_COMPILER` explicitly pointing at MinGW's `gcc.exe` and `g++.exe`.

7.4 DLL not found at runtime

If `femm_gui.exe` fails to launch with missing DLL errors, add the Qt and MinGW binary directories to your `PATH`:

```
set PATH=C:\Qt\6.10.3\mingw_64\bin;C:\Qt\Tools\mingw1310_64\bin;%PATH%
bin\femm_gui.exe
```

7.5 Building the standalone mesher

The `tangle` mesher can be built independently without Qt, using just MinGW:

```
cd solvers\common
g++ -O3 -std=c++17 -static -o tangle.exe tangle.cpp float256.cpp -lm
```