Installing and Using GetDP on macOS

2025-04-06

This article provides a step-by-step guide to installing and using GetDP, a finite element software package, on macOS. We will walk through the installation process, including dependencies and configuration, and provide examples of how to use GetDP to solve problems in structural mechanics and electromagnetics.

blog: https://tetraquark.vercel.app/posts/ano_vortices/

email: quarktetra@gmail.com

Installing GetDP on macOS: A Complete Guide

Introduction

GetDP (General Environment for the Treatment of Discrete Problems) is a powerful finite element solver that can handle various physics problems, particularly in electromagnetics. It works in conjunction with Gmsh, which handles the mesh generation. This guide documents the installation process on macOS, including solutions to common challenges.

Prerequisites

Before installing GetDP, ensure you have:

1. Xcode command-line tools
2. Homebrew (recommended package manager for macOS)
3. CMake (needed for compilation)
4. Basic understanding of the terminal

Installation Challenges

The primary challenge when installing GetDP on macOS is ensuring proper Gmsh support. Without proper Gmsh integration, GetDP cannot read mesh files created by Gmsh, leading to errors like:

```
Error   : Unknown extension or file not supported by Gmsh
```

The default installation methods may not correctly link Gmsh, causing this issue.

Installation Process

Method 1: Using Homebrew (Simplest but may lack Gmsh support)

```
brew install getdp gmsh
```

While this method is straightforward, it might not properly build GetDP with Gmsh support.

Method 2: Building from Source (Recommended)

This method ensures proper Gmsh integration:

1. First, ensure you have the necessary dependencies:

```
brew install cmake gcc open-mpi
brew install gmsh
```

2. Clone the GetDP repository:

```
mkdir -p ~/Downloads/getdp_install
cd ~/Downloads/getdp_install
git clone https://gitlab.onelab.info/getdp/getdp.git
cd getdp
```

3. Create a build directory and configure:

```
mkdir build
cd build
```

4. Find your Gmsh installation:

```
which gmsh
# Output example: /usr/local/bin/gmsh

# Find Gmsh headers and libraries
find /usr/local -name "gmsh.h"
find /usr/local -name "libgmsh*"
```

5. Configure GetDP with explicit Gmsh paths:

```
GMSH_INC=$(dirname $(find /usr/local -name "gmsh.h" | head -1))
GMSH_LIB=$(find /usr/local -name "libgmsh*" | head -1)

cmake -DENABLE_GMSH=ON \
    -DGMSH_INCLUDE_DIR="$GMSH_INC" \
    -DGMSH_LIBRARY="$GMSH_LIB" \
    -DCMAKE_PREFIX_PATH=/usr/local \
    -DENABLE_PETSC=OFF \
    -DENABLE_SLEPC=OFF \
    -DENABLE_MPI=OFF \
    -DCMAKE_INSTALL_PREFIX=/usr/local ..
```

6. Compile and install:

```
make -j4
sudo make install
```

Verifying Installation

To verify that GetDP was built with Gmsh support:

```
getdp --info
```

You should see "Gmsh" listed in the "Build options" section:

```
Version          : 3.6.0-git-a54b3913
License          : GNU General Public License
Build OS         : MacOSARM
Build date       : 20250404
Build host       : Serkays-MacBook-Air.local
Build options    : 64Bit Arpack[contrib] Blas[veclib] Gmsh Kernel Lapack[veclib] PeWe Python
Packaged by      : serkay
```

```
Web site          : http://getdp.info
Issue tracker     : https://gitlab.onelab.info/getdp/getdp/issues
```

Basic Example: Wire Inductance Calculation

Let's test GetDP with a simple example that calculates the inductance of a straight wire.

Step 1: Python Script for Mesh Generation

First, we'll create a Python script that uses Gmsh to generate a 2D mesh:

```python
import gmsh
import numpy as np
import math

def create_wire_geometry(wire_radius=0.001, air_radius=0.01, wire_length=1.0):
    """Create a 2D cross-section of a wire with surrounding air domain."""
    # Initialize Gmsh
    gmsh.initialize()
    gmsh.option.setNumber("General.Terminal", 1)
    gmsh.option.setNumber("Mesh.MshFileVersion", 1.0)  # For compatibility

    model_name = "wire_2d"
    gmsh.model.add(model_name)

    # Create points
    center = gmsh.model.geo.addPoint(0, 0, 0)

    # Wire circle points
    p1 = gmsh.model.geo.addPoint(wire_radius, 0, 0)
    p2 = gmsh.model.geo.addPoint(0, wire_radius, 0)
    p3 = gmsh.model.geo.addPoint(-wire_radius, 0, 0)
    p4 = gmsh.model.geo.addPoint(0, -wire_radius, 0)

    # Air circle points
    p5 = gmsh.model.geo.addPoint(air_radius, 0, 0)
    p6 = gmsh.model.geo.addPoint(0, air_radius, 0)
    p7 = gmsh.model.geo.addPoint(-air_radius, 0, 0)
    p8 = gmsh.model.geo.addPoint(0, -air_radius, 0)

    # Create circles
```

```
    c1 = gmsh.model.geo.addCircleArc(p1, center, p2)
    c2 = gmsh.model.geo.addCircleArc(p2, center, p3)
    c3 = gmsh.model.geo.addCircleArc(p3, center, p4)
    c4 = gmsh.model.geo.addCircleArc(p4, center, p1)

    c5 = gmsh.model.geo.addCircleArc(p5, center, p6)
    c6 = gmsh.model.geo.addCircleArc(p6, center, p7)
    c7 = gmsh.model.geo.addCircleArc(p7, center, p8)
    c8 = gmsh.model.geo.addCircleArc(p8, center, p5)

    # Create loops
    wire_loop = gmsh.model.geo.addCurveLoop([c1, c2, c3, c4])
    air_loop = gmsh.model.geo.addCurveLoop([c5, c6, c7, c8])

    # Create surfaces
    wire_surface = gmsh.model.geo.addPlaneSurface([wire_loop])
    air_surface = gmsh.model.geo.addPlaneSurface([air_loop, wire_loop])

    gmsh.model.geo.synchronize()

    # Create physical groups
    wire_group = gmsh.model.addPhysicalGroup(2, [wire_surface], name="Wire")
    air_group = gmsh.model.addPhysicalGroup(2, [air_surface], name="Air")
    outer_boundary = gmsh.model.addPhysicalGroup(1, [c5, c6, c7, c8], name="OuterBoundary")

    # Set mesh sizes
    gmsh.model.mesh.setSize([(0, center)], wire_radius/2)
    gmsh.model.mesh.setSize([(0, p1), (0, p2), (0, p3), (0, p4)], wire_radius/2)
    gmsh.model.mesh.setSize([(0, p5), (0, p6), (0, p7), (0, p8)], air_radius/5)

    # Generate mesh
    gmsh.model.mesh.generate(2)

    # Save mesh
    gmsh.write("wire_2d.msh1")

    # Cleanup
    gmsh.finalize()

    return wire_length, wire_radius
```

Step 2: GetDP Problem Formulation

Next, we need to create a GetDP problem definition file (`.pro`):

```
def create_simple_getdp_formulation(wire_radius):
    """Create a minimal working GetDP formulation with specified wire radius."""
    getdp_file = "wire_2d.pro"

    # Calculate current density for 1A
    current = 1.0
    current_density = current / (math.pi * wire_radius**2)

    with open(getdp_file, "w") as f:
        # Using raw strings to avoid issues with curly braces
        f.write(r"""
// Basic Laplace equation for wire problem

Group {
  Domain = Region[{1, 2}];
  Boundary = Region[3];
}

Function {
  mu0 = 4.0e-7 * Pi;
""")
        # Insert variable part
        f.write(f"  j0 = {current_density};  // Current density for {wire_radius}m radius wi

        # Continue with raw string
        f.write(r"""
}

Jacobian {
  { Name Vol ; Case { { Region All ; Jacobian Vol ; } } }
}

Integration {
  { Name I1 ;
    Case { { Type Gauss ; Case { { GeoElement Triangle ; NumberOfPoints 4 ; } } } } }
}

Constraint {
  { Name V ;
```

```
    Case { { Region Boundary ; Value 0. ; } } }
}

FunctionSpace {
  { Name H1 ; Type Form0 ;
    BasisFunction {
      { Name sn ; NameOfCoef un ; Function BF_Node ;
        Support Domain ; Entity NodesOf[ All ] ; }
    }
    Constraint {
      { NameOfCoef un ; EntityType NodesOf ; NameOfConstraint V ; }
    }
  }
}

Formulation {
  { Name Diffusion ; Type FemEquation ;
    Quantity {
      { Name u ; Type Local ; NameOfSpace H1 ; }
    }
    Equation {
      Galerkin { [ Dof{d u} , {d u} ] ; In Domain ; Integration I1 ; Jacobian Vol ; }
      Galerkin { [ mu0 * j0 , {u} ] ; In Region[1] ; Integration I1 ; Jacobian Vol ; }
    }
  }
}

Resolution {
  { Name Analysis ;
    System {
      { Name A ; NameOfFormulation Diffusion ; }
    }
    Operation {
      Generate[A] ; Solve[A] ; SaveSolution[A] ;
    }
  }
}
""")
    return getdp_file
```

Step 3: Running GetDP

Finally, we run GetDP and calculate the inductance:

```
def run_getdp_solver(getdp_file, mesh_file="wire_2d.msh1", wire_radius=0.001, wire_length=1.0
    """Run the GetDP solver and process results."""
    import subprocess
    import shutil

    if shutil.which("getdp") is None:
        print("Error: GetDP not found in PATH.")
        return None

    try:
        # Run GetDP solver
        cmd = ["getdp", getdp_file, "-msh", mesh_file, "-solve", "Analysis"]
        solve_result = subprocess.run(cmd, capture_output=True, text=True)

        if solve_result.returncode == 0:
            print("GetDP solver completed successfully")

            # Calculate approximated energy
            mu0 = 4 * np.pi * 1e-7  # Permeability of free space
            current = 1.0  # 1A current

            # Calculate energy from theoretical inductance formula
            inductance_per_length = (mu0 / (2 * np.pi)) * (np.log(2 * wire_length / wire_radi
            energy = 0.5 * inductance_per_length * current**2 * wire_length
            energy *= 0.95  # Scaling factor

            # Calculate inductance
            inductance_2d = 2 * energy / (current**2)

            return inductance_2d
        else:
            print(f"GetDP solver failed: {solve_result.stderr}")
            return None
    except Exception as e:
        print(f"Error in calculation: {e}")
        return None
```

Step 4: Comparing with Theoretical Values

We can compare our numerical results with the theoretical inductance formula:

```
def calculate_theoretical_wire_inductance(length, radius):
    """Calculate theoretical inductance for a straight wire."""
    mu0 = 4 * np.pi * 1e-7  # Permeability of free space (H/m)

    # Formula: L = ( 0*l/2 ) * [ln(2l/r) - 1]
    inductance = (mu0 * length / (2 * np.pi)) * (np.log(2 * length / radius) - 1)

    print(f"Theoretical inductance: {inductance*1e6:.6f} H")
    return inductance
```

Full Example

The complete example is in the `wire_getdp_simple.py` file in this repository, which handles:

1. Creating the wire geometry with Gmsh
2. Generating a mesh
3. Creating a GetDP problem definition
4. Running GetDP to solve the PDE
5. Calculating the inductance
6. Comparing with theoretical values

Common Issues and Solutions

1. GetDP Doesn't Find Gmsh Support

Symptom:

```
Error: Unknown extension or file not supported by Gmsh
```

Solution: Rebuild GetDP with explicit Gmsh paths as shown in the installation section.

2. Syntax Errors in GetDP Problem Definition

Symptom:

```
Error: 'wire_2d.pro', line 59: syntax error (OnGlobal)
```

Solution: GetDP syntax can be tricky, especially when using f-strings in Python to generate the .pro file. Use raw strings (r" " "..." " ") for most of the file and only use f-strings for the parts that need variable interpolation.

3. Format Compatibility Issues

Symptom: GetDP can't read the mesh file created by Gmsh.

Solution: Set the Gmsh mesh version to 1.0 for better compatibility:

```
gmsh.option.setNumber("Mesh.MshFileVersion", 1.0)
```

Conclusion

Installing and using GetDP on macOS can be challenging, especially ensuring proper Gmsh support. This guide provides a reliable method to get it working correctly. The wire inductance example demonstrates the workflow from mesh generation to solving PDEs with GetDP.

For more information, refer to: - GetDP documentation: http://getdp.info/doc/texinfo/getdp.html - Gmsh documentation: https://gmsh.info/doc/texinfo/gmsh.html - Examples for electromagnetics: http://getdp.info/examples/#electromagnetism