High-Performance
Computing Center
Stuttgart

Group Assignments Introduction - Finite Element Method (FEM)

Overview

Introduction

Overview: FEM Group Assignments

Three reference implementations are provided as foundational material for the group projects. Based on these, each group defines its own objectives and methodological approach.

- Neuromorphic Computing for FEM — Parametric Study
- Fortran FEM Reference Implementations
  - Implementation 1: Bilinear quadrilateral (Q1) elements on a structured mesh.
  - Implementation 2: Linear triangular (P1) elements on a structured mesh.

FEM – Basics 1

Starting Point: Partial Differential Equations (PDEs)

PDEs describe many physical phenomena but are generally not analytically solvable for complex geometries, motivating numerical approaches such as the FEM.

Linear Elasticity

$$-\nabla \cdot \sigma(\mathbf{u}) = \mathbf{f}$$
$$\sigma = \lambda(\nabla \cdot \mathbf{u})I + 2\mu\,\varepsilon(\mathbf{u}), \qquad \varepsilon(\mathbf{u}) = \tfrac{1}{2}\left(\nabla\mathbf{u} + \nabla\mathbf{u}^\top\right)$$

Heat Equation

$$\frac{\partial^2\phi}{\partial t^2} = \frac{\partial^2\phi}{\partial x^2} + \frac{\partial^2\phi}{\partial y^2} + \frac{\partial^2\phi}{\partial z^2}$$

Objective: Reformulate the PDE as a Linear System

Using standard discretization techniques of the Finite Element Method, the continuous PDE is transformed into a finite-dimensional system of equations, leading to the linear system.

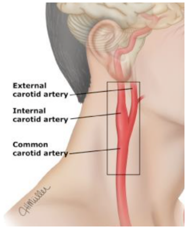
$$\mathbf{K}\mathbf{u} = \mathbf{f},$$

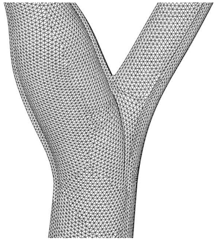
- **K** — global stiffness matrix
- **u** — vector of nodal unknowns
- **f** — global load vector

FEM – Basics 2

Fundamental Steps of the FEM

- **Discretization:** Partition the domain into finite elements (triangles, quadrilaterals, tetrahedra).
- **Shape Functions:** Introduce local basis functions (e.g., P1, Q1) to approximate u on each element.
- **Assembly:** Compute element stiffness matrices and assemble them into the global matrix $\mathbf{K}$ and vector $\mathbf{f}$.
- **Solution:** Apply boundary conditions and solve the resulting linear system (e.g. with the Conjugate Gradient method).





FEM – Basics 3

Advantages of the Finite Element Method

- High geometric flexibility compared to FDM/FVM.
- Enables accuracy improvements through higher-order shape functions.
- Local mesh refinement and adaptive methods.
- Robust handling of complex boundary conditions and multiphysics couplings.

Applications

- Structural mechanics, heat transfer, fluid mechanics.
- Electromagnetics, geophysics, multi-field and coupled problems.

→ For a more detailed theoretical background, the presentation linked below explains how the two-dimensional Poisson equation is solved using the Finite Element Method together with the Conjugate Gradient method. This approach corresponds closely to that adopted in the FEM reference implementations.
→ 2D_Poisson_FEM.pdf

Assignment 1: Optimization of FEM via parameter variation

Setting up your local Host

1. Clone the repo https://github.com/hlrs-fcg/spinnaker2_neurofem.git to yourfolder
2. run the following commands:

Command

```
cd yourfolder/spinnaker2_neurofem/src
git clone --branch neurofem_v6 git@gitlab.com:spinnaker2/py-spinnaker2.git
requirements/spinnaker2/libs
```

3. copy the repo within `libs.zip/s2-sim2lab-app/` to `yourfolder/spinnaker2_neurofem/src/requirements/spinnaker2/src/spinnaker2`
4. Create a virtual env and install the dependencies:

Command

```
python3 -m venv ./venv && source ./venv/bin/activate
pip install -U -r requirements.txt
cp -r requirements/spinnaker2/src/spinnaker2/libs/chip/app-pe/s2app/neurofem_2048/
venv/lib/python3.12/site-packages/spinnaker2/libs/chip/app-pe/s2app/
```

Trouble Shooting

All the information is available under https://github.com/hlrs-fcg/spinnaker2_neurofem/. The repo-structure is illustrated as followed:

```
.
└── src
    ├── neurofem                              # python code for neurofem
    │   ├── simulation                        # main NeuroFEM source code
    │   ├── config                            # configuration classes for SpiNNaker2 and NeuroFEM
    │   ├── problems
    │   │   ├── poisson                       # generation of linear systems for Poisson problems
    │   │   └── utils                         # utility functions for problem generation
    │   └── calc                              # mse, error calculation, etc.
    ├── requirements                          # directory for dependencies
    └── requirements.txt                      # dependencies file
```

All the information is given in **`yourfolder/spinnaker2_neurofem/README.md`**.
Furthermore, check out README.md and TODOS.md in
`yourfolder/spinnaker2_neurofem/src/requirements/spinnaker2/src/spinnaker2/libs`.

Assignment Task

Minimize the relative residual of the spinnaker2-powered FEM-simulation. Thus

1. Perform a parameter sweep with the simulation parameters. Start with varying the parameter gamma and find the gamma value that minimizes the residual. Plot your results in a diagram.
2. Expand task 1 by varying the other parameters in order to determine the parameter set that minimizes the residual. The parameter definitions can be found in `yourfolder/spinnaker2_neurofem/src/neurofem/config.py`.
3. Try to design an algorithm that approximates this parameter set iteratively.
4. Find creative solutions to furtherly improve the framework (e.g to achieve even smaller residuals, speed up the simulation and/or reduce computational cost). Feel engaged to improve the FEM-algorithm itself, you are not restricted to the simulation parameters.
   Further information is given in https://www.nature.com/articles/s42256-025-01143-2.

A simple example can be found in `yourfolder/spinnaker2_neurofem/src/example.py`.
Important: Check out the grading criteria in Lecture 1! Not only your results and methods, but also the structure of your git repo and your presentation will influence your grade.

Assignment 2: Neuromorphic Impl. of Fortran FEM Solvers

FEM Reference Implementations 1 & 2

- Two Fortran-based reference implementations are provided for a model 2D problem on the unit square $[0, 1] \times [0, 1]$.
- Homogeneous Dirichlet boundary conditions are applied.
- Linear systems are solved using the Conjugate Gradient method.
  - Implementation 1: Bilinear quadrilateral (Q1) elements on a structured mesh.
  - Implementation 2: Linear triangular (P1) elements on a structured mesh.
- Each implementation is accompanied by a Jupyter Notebook that provides a quick introduction and visualization utilities for the results.

Assignment Task

Objective: Transfer the provided Fortran reference implementation to the neuromorphic Spinnaker hardware using the NeuroFEM framework.

Steps:

- Select one Fortran reference implementation (Q1 or P1 elements).
    - Both reference implementations are structurally similar, so the choice has minimal impact on the overall task.
- Port the selected implementation to the neuromorphic Spinnaker hardware.
- Validate the new implementation by comparison with the original Fortran reference.

Note:
This group project is **extensive and complex**, requiring significantly more effort compared to the first assignment.

Fortran — GFortran Installation

Fortran is a high-performance, compiled language widely used for numerical computing and scientific applications.

Installing the Fortran compiler **GFortran**:

- macOS:
  - Homebrew: `brew install gcc`
  - Xcode: `xcode-select --install`
- Linux:
  - Debian-based: `sudo apt install gfortran` or `sudo apt-get install gfortran`
  - Arch-based: `sudo pacman -S gcc-fortran`
  - RPM-based (Fedora, RHEL, ...): `sudo yum install gcc-gfortran` or `sudo dnf install gcc-gfortran`
- Windows: use WSL (Ubuntu) and follow the Debian/Ubuntu steps, or follow the installation steps on the website linked below.

Control installation with: `gfortran --version`

Further information: Installing GFortran

Compiling and Running the Fortran FEM Implementation

Compiling the Fortran source code with the GFortran compiler:

```
cd src
gfortran -O2 -std=f2008 mod_mesh.f90 mod_assemble.f90 mod_rhs.f90 mod_solver.f90 mod_io.f90
main.f90 -o main
```

Running the compiled program:

```
# From the project root:
./src/main
```

Alternative: Using provided helper scripts

- Two helper scripts are provided in the `scripts/` directory to simplify build and cleanup: `scripts/compile_fortran.sh` (builds `src/main`) and `scripts/clean_src.sh` (removes `main`, `*.o`, `*.mod`).
- A more detailed description is provided in the Jupyter notebooks accompanying each reference implementation.

Github Repository Workflow

GitHub Workflow - 1

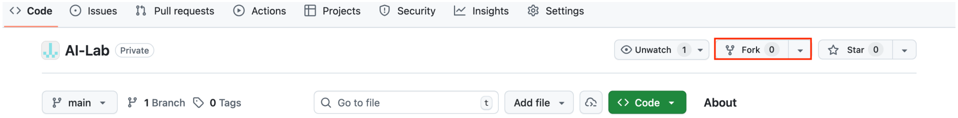
Repository Access

All materials for the FEM group projects are available at: Neuromorphic-Lab

Recommended Git Workflow

Fork a Repository

- Each group should fork the original repository on GitHub, creating an independent copy for the group.
- Other group members then clone this fork locally to start working.



Commands Example:

Clone the group fork locally

```
git clone <your-group-fork-url>
```

GitHub Workflow - 2

Working with Branches in Each Group Repository

- Branches are independent lines of development within the repository.
- Use branches to work on features or tasks without affecting others' work.
- Merge branches back into main or keep them separate once work is complete.

Commands Example:

Create a new branch and switch to it

```
git checkout -b <branch-name>
```

Switch to an existing branch

```
git switch <branch-name>
```

Merge a branch into the target branch

```
git switch <target-branch> # Switch to the branch that should receive the changes
git merge <source-branch> # Merge the other branch into it
```

GitHub Workflow - 3

Synchronize Work within the Group:

- Regularly push and pull changes to/from the group repository to keep all team members updated.

Commands Example:

Push changes to the remote branch

```
git add .
git commit -m "Describe your changes"
git push origin <branch-name>
```

Pull latest changes from the remote branch

```
git pull origin <branch-name>
```

VPN Account Setup Process

SpiNNcloud Account Initialization

1. Receive login credentials for the self-service VPN setup page.
2. Access the VPN portal: `https://vpn.spinncloud.com/vpnportal/webpages/login.html`
3. Enter your username and password, then click **Login**.

Two-Factor Authentication Setup

1. Register your authenticator app by scanning the displayed QR code.
2. Follow the setup instructions inside your authenticator app.
3. Click **Proceed to login** to continue.

VPN Portal Access

1. Return to the VPN Portal for future logins.
2. Use the same username as before.
3. Password format:
   original password + 6-digit authenticator code
   - Example: If your password is `PASSWORD` and the token is `123456`, enter: `PASSWORD123456`
   - Tokens refresh every 30 seconds.
4. Click **Login** to enter the portal.

VPN Client Configuration

Download your personalized VPN configuration file from the portal.

- Choose either a Sophos (Windows) or OpenVPN (Linux) configuration.

Sophos (Windows)

1. Download and install the Sophos connect client from the VPN portal.
2. Import your configuration file into Sophos Connect.
3. Connect using: **username + (password + token)**.

Linux (OpenVPN)

1. Install OpenVPN.
2. Start OpenVPN with your configuration file:

Command

```
sudo openvpn config.ovpn --tun-mtu 1392
```

- Enter your username and password+token when prompted.

Access over JupyterHub

1. After connecting via VPN, visit: `https://10.9.22.11:8000/`

New User Registration:

- Click **Sign Up**.
- Use username format: `COMPANY_USER`
- Set a secure password.
- Submit request for admin approval.

JupyterHub Login

1. Once approved, log in with your credentials.
2. The system automatically launches your working container.
3. You are redirected to the JupyterLab interface.
4. Use terminal or Python sessions as needed.