POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Project Report

Title

Advanced Programming for Scientific Computing

Authors: Andrea Bonifacio, Sara Gazzoni

Advisors: Stefano Pagani

Co-advisor: Mattia Corti

Academic year: 2022-2023

---

1. Introduction

Full 3D blood flow models are important in the study of cardiovascular system since they allow to extract detailed quantities of interest but their actual implementation is limited due to their high computational cost. For this reason, reduced order models are widely used in this fields because of their efficiency. An example is presented in [2], where a one-dimensional reduceed order model is implemented to simulate the blood flow in the aorta using a graph neural network trained on three-dimensional simulations. In this work, we propose a different application, where the graph neural network is used to approximate the solution of different problems. In particular, we consider the heat equation as test case, but the goal of the project is to show the potential extension of this approach to solve more difficult problems with complex geometries, such as the simulations of proteins spreading in the neural system, which are at the basis of neurodegenerative diseases [1]. The main part of this project is the implementation of a library for data generation used to train the graph neural network and the adaptation of the code [di Luca non so come citarlo] to make it suitable for our specific test case. In the following sections, we first present the problem formulation and a detailed description of the code developed, then we show the results obtained and a discussion of the possible further developments and extensions.

2. Problem overview

We consider a general time-dependent variational problem of the form:

$$Lu = f$$

with L a linear operator, f a source term and u the solution. Given a specific geometry Ω and using the finite element method implemented in Fenics, we can solve this problem and obtain the solution u^n at each time step n. From this, we can generate a graph that decribes the geometry of the problem and the solution, storing some values of interest as features of the nodes and the edges. Solving the problem for different geometries and different values of the parameters (e.g. the diffusivity constant) we can generate a dataset that will be used to train the graph neural network. As in [2], the GNN is applied iteratively: at each time step it takes as input the system state Θ^n, which is the set of all the nodes and edges features at that time step, and it predicts an update for the state variables. The prediction is combined with the previous time step to estimate Θ^{n+1}.

2.1. Test case: heat equation

In this work, we consider the heat equation as test case. The mathematical formulation of the problem is the following:

$$\begin{cases} \frac{\partial u}{\partial t} = k\Delta u \quad \text{in } \Omega \subset \mathbb{R}^2, \\ \frac{\partial u}{\partial n} = h \quad \text{on } \partial\Omega_{inlet}, \\ \frac{\partial u}{\partial n} = 0 \quad \text{on } \partial\Omega_{outlet} \cup \partial\Omega_{walls}. \end{cases} \tag{1}$$

where u is the temperature, k is the diffusivity constant, h is the Neumann condition at the inlet boundary. As domain Ω we consider different geometries such as the one shown in Figure : the 2D mesh is composed of 4 trapezoids where the interface between them have different lengths.

We generated 20 different mesh using gmsh. Then we solved the problem in Fenics using Discontinuous Galerkin method and implicit Euler for time discretization, imposing as Neumann condition at inlet $h = 2e^{-(t-2.5)^2}$. From these solutions we generated a dataset of 277 graphs. Each graph has 5 nodes: an inlet node, an outlet node and 3 nodes in correspondence of the interfaces. As descriptor of the the state of the system we consider the heat flux at each time step, which is computed as the integral of the normal derivative of the solution on the interface. The other node features are the thermal diffusivity k, the interface length and the nodal type (inlet, outlet or branch node). As edge features we consider the area of the corresponding trapezoid and the distance between the nodes connected by the edge.

3. Code

3.1. Mesh creation

3.2. Data generation

`GenerateData.py` contains two abstract classes: `Solver` and `DataGenerator`. The first one is used to solve the variational problem, while the second one is designed to store all the quantities of interest which will be used to build the graphs. Each of these parent classes has two child classes: we start describing the solver one.

The abstract base class `Solver` contains the following methods:

- A costructor that takes as input a `MeshLoader` object
- Abstract method `set_parameters`
- Abstract method `solve`
- Abstract method `plot_solution`

All the abstract method are overriden in the child classes `Heat` and `Stokes`. The choice of a parent abstract class for the solver is useful because it allows to use the same code for different problems, implementing child classes that solve different equations, but with the same structure. We focus on the description of the `Heat` class, since it is the one used in the test case, but the `Stokes` class is implemented analogously.

The `Heat` class contains the following methods overriden from the parent class:

- A constructor which uses the `super()` function to inherit the base class constructor. Other problem parameters are passed to the constructor such as function space, the source term, the diffusivity constant, the time step and so on.
- `set_parameters`: function to set different problem parameters
- `solve`: this method solves the Heat equation using Discontinuous Galerkin method and imposing a Neumann conditiona at the inlet boundary. The solution at each time step is stored in a list, as well as the time instants.
- `plot_solution`: it takes as input the solution at a specific time step and it plots it.

3.3. Graph generation

3.4. Graph Neural Network

4. Results

5. Further work

References

[1] Mattia Corti, Francesca Bonizzoni, Luca Dede', Alfio M. Quarteroni, and Paola F. Antonietti. Discontinuous galerkin methods for fisher-kolmogorov equation with application to alpha-synuclein spreading in parkinson's disease. *Computer Methods in Applied Mechanics and Engineering*, 2023.

[2] Luca Pegolotti, Martin R. Pfaller, Natalia L. Rubio, Ke Ding, Rita Brugarolas Brufau, Eric Darve, and Alison L. Marsden. Learning reduced-order models for cardiovascular simulations with graph neural networks. *Computers in Biology and Medicine*, 2023.