Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Exploratory Analysis of Turbulent Flow Data using GNN-based Surrogate Model

Vaishali Ravishankar

Computational Science and Engineering (International Master's Program)

Technische Universität München

Master's Thesis

Exploratory Analysis of Turbulent Flow Data using GNN-based Surrogate Model

Author:	Vaishali Ravishankar
1st examiner:	Univ.-Prof. Dr. Hans Joachim Bungartz
2nd examiner:	Univ.-Prof. Dr. Jochen Garcke
Assistant advisor:	Kislaya Ravi
Submission Date:	April 15th, 2023

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

April 15th, 2023 Vaishali Ravishankar

Acknowledgments

If someone helped you or supported you through your studies, this page is a good place to tell them how thankful you are.

"People sometimes ask me if it is a sin in the Church of Emacs to use vi. Using a free version of vi is not a sin; it is a penance. So happy hacking"

-Richard Stallman

Abstract

Turbulent flows, characterized by their complex and chaotic nature, play a pivotal role in various engineering and natural systems. Understanding and analyzing these phenomena is essential for optimizing design, predicting crucial outcomes and addressing real-world challenges. Therefore, obtaining accurate, efficient and rapid predictions of turbulent behaviors is of utmost importance. Data-driven methods such as Machine Learning algorithms are being increasingly implemented to speed up flow predictions compared to numerical solvers. However, these models tend to have poor generalization capabilities and are often restricted to simple geometries on structured grids. Hence, the development and application of a Graph Neural Network (GNN) based framework is proposed to handle irregular unstructured mesh data from turbulent flow simulations. Moreover, a meta-learning approach can be devised that further empowers the GNN-based surrogate model to adapt and generalize more effectively, even when confronted with out-of-distribution data, with the potential to make turbulent flow predictions more accurate and robust. The underlying goal of this research is to leverage the predictions of the surrogate model to perform a comprehensive exploratory analysis of the parameter space that governs the performance of airfoils operating in turbulent flow conditions.

Contents

Part I.

Introduction and Background Theory

1. Introduction

This document has been created in order to show you some of the capabilities of LaTeX. A great resource for an introduction to LaTeX is Tobias Oetiker's "The Not So Short Introduction to LaTeX 2_ε" [**?**]. Please page through that document before starting with your thesis. Oh, and let's use the mysterious word computer here to give the glossary a reason to appear. A third useful option to reference stuff besides citing or glossarying (?) is using footnotes. Just like this[1] one. And: lists! Lists with bullet points are amazing. I mean, just look at this:

- list
- all
- the
- things!

Anyways your introduction goes here.
Below a few LaTeXexamples are included for beginners

You can also put comments in the margins for you or your advisor

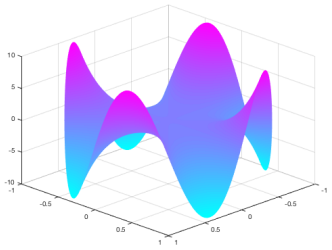


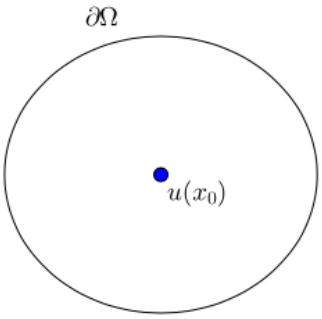
Figure 1.1.: $u(x)$

Equations can also be labeled

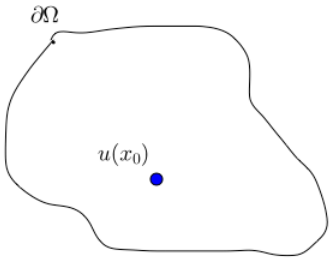
$$\pi = \mathrm{e}^{i \cdot \phi} \tag{1.1}$$

And later referenced. Even in subfigures.

[1] Properly formatted clickable URL: https://www.tum.de/



(a) Equation 2.1



(b) Equation 2.1

1.1. Including code

Code can be using the package Minted.

An exaple of which of can be found below (see Source Code 2.1)

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Source Code 1.1.: My nice listing

Part II.

Body: What Was Done for the Thesis

2. Part2

This document has been created in order to show you some of the capabilities of LaTeX. A great resource for an introduction to LaTeX is Tobias Oetiker's "The Not So Short Introduction to LaTeX 2_ε" [**?**]. Please page through that document before starting with your thesis. Oh, and let's use the mysterious word computer here to give the glossary a reason to appear. A third useful option to reference stuff besides citing or glossarying (?) is using footnotes. Just like this[1] one. And: lists! Lists with bullet points are amazing. I mean, just look at this:

- list
- all
- the
- things!

Anyways your introduction goes here.
Below a few LaTeXexamples are included for beginners

You can also put comments in the margins for you or your advisor

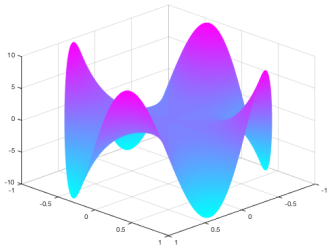


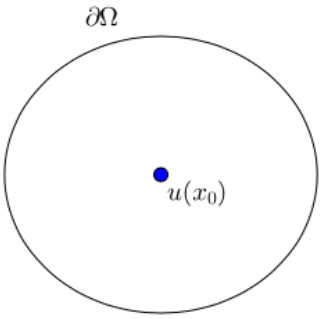
Figure 2.1.: $u(x)$

Equations can also be labeled

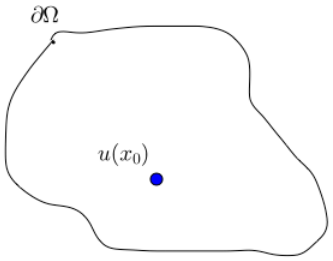
$$\pi = \mathrm{e}^{i \cdot \phi} \tag{2.1}$$

And later referenced. Even in subfigures.

[1] Properly formatted clickable URL: https://www.tum.de/



(a) Equation 2.1



(b) Equation 2.1

2.1. Including code

Code can be using the package Minted.
An exaple of which of can be found below (see Source Code 2.1)

```
import numpy as np

def incmatrix(genl1,genl2):
    m = len(genl1)
    n = len(genl2)
    M = None #to become the incidence matrix
    VT = np.zeros((n*m,1), int)  #dummy variable

    #compute the bitwise xor matrix
    M1 = bitxormatrix(genl1)
    M2 = np.triu(bitxormatrix(genl2),1)

    for i in range(m-1):
        for j in range(i+1, m):
            [r,c] = np.where(M2 == M1[i,j])
            for k in range(len(r)):
                VT[(i)*n + r[k]] = 1;
                VT[(i)*n + c[k]] = 1;
                VT[(j)*n + r[k]] = 1;
                VT[(j)*n + c[k]] = 1;

                if M is None:
                    M = np.copy(VT)
                else:
                    M = np.concatenate((M, VT), 1)

                VT = np.zeros((n*m,1), int)

    return M
```

Source Code 2.1.: My nice listing

Part III.

Results and Conclusion

Appendix

A. Detailed Descriptions

Bibliography