2021-08-01

# Towards Reinforcement Learning Driven Mesh Adaptivity For Second Order Elliptic Problems

Augustine Twumasi
*University of Texas at El Paso*

TOWARDS REINFORCEMENT LEARNING DRIVEN MESH ADAPTIVITY FOR SECOND ORDER ELLIPTIC PROBLEMS

AUGUSTINE TWUMASI

Master's Program in Mathematical Sciences

APPROVED:

Natasha S. Sharma, Ph.D., Chair

Granville Sewell, Ph.D.

Tunna Baruah, Ph.D.

Stephen L. Crites, Jr., Ph.D.
Dean of the Graduate School

*To my*

*FATHER Fosu, MOTHER Dora, WIFE Adelaide, Son Oheneba and my ADVISOR Dr. Sharma*

*with love*

# TOWARDS REINFORCEMENT LEARNING DRIVEN MESH ADAPTIVITY FOR SECOND ORDER ELLIPTIC PROBLEMS

by

AUGUSTINE TWUMASI

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Mathematical Science

THE UNIVERSITY OF TEXAS AT EL PASO

August 2021

# Acknowledgements

This work was only possible thanks to so many giants that made me stand on their shoulders, without whom I would not have been able to complete this journey. To God, in whom I live, move and have my being. I would like to express my sincere gratitude to my mentor Dr. Natasha S. Sharma. With her wisdom, inspiration, and trust, she guided me through my path of pursuing my Masters, and helped me overcome any obstacles on the way. Her support to me is beyond research, and I will forever cherish her guidance. I also extend thanks to my committee members, Prof.Granville Sewell and Prof Tunna Baruah for their valuable suggestions and insightful comments. I have certainly benefited a lot from the discussion with them. I am also very grateful to The University of Texas at El Paso Mathematical Science Department professors and staff for all their hard work and dedication, providing me the means to complete my degree. To the love of my life, Adelaide Awuradwoa Asiamah I cannot imagine my life without her. I love her to the end of the world. To my son Oheneba Yaw Fosu Twumasi, thank you for challenging me, the sight of you propel me to do more. And to my original teachers who thought me the length and breadth of this life; my parents; Dora Okyere and Noah Fosu Manu.

# Abstract

Adaptive mesh refinement techniques have become an indispensable tool in achieving accurate and efficiently computed solutions to problems which require impractically fine uniform meshes to obtain an accurate approximation. The adaptive algorithm involves a recursive application of SOLVE-ESTIMATE-MARK-REFINE steps where in particular the step 'ESTIMATE' involves computing a posteriori error estimator based on only the numerical solution and the data of the problem. Over the years, several a posteriori error estimators have been developed and successfully applied but often times, the choice of estimator is ill suited for the problem at hand. In this research, we present two estimators namely the residual-based estimator and the gradient recovery type estimator and illustrate the suitability of these estimators for different problems at hand. We also provide the foundation for data-driven adaptive mesh refinement strategies based on Reinforcement learning (RL) with a focus on the Q-learning algorithm which is a fundamental learning algorithm in RL.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

Partial differential equations emerge in the mathematical modelling of many physical, chemical and biological phenomena and many diverse subject areas such as fluid dynamics, electromagnetism, material science, astrophysics, economy, financial modelling, etc. Very frequently the equations under consideration are so complicated that finding their solutions in closed form or by purely analytical means is either impossible or impracticable, and one has to resort to seeking numerical approximations to the unknown analytical solution[28]. Over the decades, adaptivity has been a well established tool used to improve the resolution of rough solutions. Since analytic solutions to multiphysics problems are very uncommon, we mostly resort to numerical solution techniques. In order to obtain a decent approximation of the true solution, a rather fine discretization mesh appears to be important. This leads to huge large scale problems on the discrete level which present many difficulties to numerical procedures. The aim of mesh adaptation is to coarsen the mesh in subdomains where the (continuous) solution and data of the problem is calm and to use a fine mesh only in regions where we expect nonlinearity or non smoothness of the significant solution or data. Apparently, one seeks to minimize the number of nodes in a finite element discretization of the problem under the limitation of keeping solution accuracy. For this process, error estimates and local error indicators are required which guide that adaptation process. As a consequence, in recent years automatic adaptivity has become an area of interest in the field of applied mathematics. Therefore, adaptive mesh refinement is fundamentally a sequential decision-making problem in which a sequence of greedy decisions dependent on instantaneous error indicators does not comprise an optimal sequence of decisions for the actual goal of achieving high cumulative or terminal precision. For example, in time-

dependent problems such as advection, an error estimator by itself cannot preemptively refine elements which would encounter complex features in the next time step. Moreover, the numerical error accumulated at the current time step will itself propagate throughout the physical system and determine how the error at future time steps will accumulate. This implies that the optimality of a refinement decision relies on the accuracy of future states and that selecting an element which yields the largest reduction in error at the current time step may not be the optimal decision over the entire simulation. Whether and how optimal AMR strategies can be found by directly optimizing a long-term performance objective are open questions. With the above analogy, we formulate AMR as a Markov decision process(MDP) and propose a reinforcement learning(RL) approach that explicitly trains a mesh refinement policy to optimize a performance metric, such as minimizing a final solution error [39].

## 1.1 Problem Description

In finite element simulations there often arises the need to change the mesh and continue the simulation on a new mesh. Numerical Analysts encounter such an issue when they adaptively refine the mesh to reduce the computational cost, smooth distorted elements to improve system conditioning, or introduce new surfaces and change the domain in simulations of the PDE problem. This adaptive refinement is guided by estimators which is a computable quantity calculated solely in terms of the data of the problem and the numerical solution. Adaptive mesh refinement has been around for the longest time and research about its effeciency in terms of lower bound and reliability in terms of upper bound has been establised over the decades. The challenge arise from the fact that we have several estimators such as residual-based, averaging estimators (gradient recovery type for instance) and estimators based on local problems. Some estimators might not be well suited for certain problems which may be as a result of computational complexity, reliability and efficiency etc.

The problem is not the reliability or the efficiency since it has been well proven, but the issue is how big are the constants involved (in the bounds) and the information they provide us with. For some problems, the estimator is not well suited because the reliability constant might be very large which is just amplifying the estimator but not giving any accurate information about the solution to the problem.
Therefore, depending on the problem at hand, a decision needs to be made about the choice of estimator. But this is a difficult decision in the absence of information about the exact solution, hence there is a need to develop a framework to automate this decision. Recently, machine learning techniques have found applications in deterministic mathematical models particularly adaptive finite element methods for example, recurrent deep neural networks applied to mesh adaptivity in [8] and to *hp*-adaptivity in [23].

## 1.2 Thesis Outline

Having introduced this work, the rest of the thesis is organized as follows. Chapter 2 gives pertinent literature review on adaptive mesh refinement(AMR) and reinforcement learning (RL). In chapter 3, which is the heart of the thesis, we establish the fact that in abscence of the true solution, we can still have an idea about the errors which we encoutered in our approximation technique by defining error estimators and their proofs. We proceed to talk about Markov Decision Process(MDP's) as a decision process in learning by trial and error which forms the basis of our learning procedure, thus, Reinforcement Learning(RL) in chapter 4. We discuss adaptive mesh refinement implementation (AMR) and some algorithm governing the strategy. In chapter 6, we present numerical experiments and illustrate the suitability of these estimators for different problems at hand as well as observe the convergence of the estimators and errors. We also provide the foundation for data-driven adaptive mesh refinement strategies based on Reinforcement learning (RL) with a focus on the Q-learning algorithm which is a fundamental learning algorithm in RL. We conclude this thesis in chapter 7 with significance of our study propose some future studies.

# Chapter 2

# Related Literature

## 2.1 Overview of Adaptive Mesh refinement

### 2.1.1 Mesh

A mesh is a partition of a geometric domain into small simple shapes, such as triangles or quadrilaterals in two dimensions and tetrahedra or hexahedra in three satisfying certain assumptions. Meshes find use in many application areas. In geography and cartography, meshes give compact representations of terrain data. In computer graphics, most objects are ultimately reduced to meshes before rendering. Finally, meshes are almost essential in the numerical solution of partial differential equations arising in physical simulation [6].

### 2.1.2 Types of Meshes

A **structured** mesh is a mesh in which all interior vertices are topologically similar. An extreme restriction of the structured mesh is the condition that it describes a unique coordinate mapping. A mesh is said to be an **unstructured** if the nodes have arbitrarily varying local neighborhoods. A **block-structured or hybrid mesh** is formed by a number of small structured meshes combined in an overall unstructured pattern. In general, structured meshes gives simplicity and easy data access, while unstructured meshes offer more convenient mesh adaptivity(refinement/derefinement based on an initial solution) and better fit for complicated domains. High- qaulity hybrid meshes enjoy the advantages of both approahes, but hybrid meshing is not fully automatic.

### 2.1.3 Mesh Adaptation Procedures

A review of current literature on mesh adaptation techniques demonstrates that a mesh adaptation strategy may include an addition of extra nodes, redistribution of existing nodes, or increasing the order of the interpolating polynomial on an existing mesh. This leads to the following general characterization of commonly used mesh refinement procedures:

- p-adaptation(increasing polynomial order)
- r-adaptation(redistributing existing nodes)
- h-adaptation(adding nodes)

**p-adaptation**

A p-type adaptive strategy assumes that the eIements in the mesh remain firm, both in size and location, but the order of interpolation may change Iocally to match the variation in the solution. The advantage of this method is that coarse meshes can be used. The method was first developed for problems in solid mechanics [31], and later applied to the two-deimensional Navier-Stokes equations by Demkowicz [11]. Schemes making use of *p-adaptation* shows a faster rate of convergence than other adaptive schemes. Szabo et al [31] report shows that for obtaining the same level of accuracy, the p-version reqires only one-fifth to one-tenth the degrees of freedom required in the h-type adaptivity. A significant demerit of the p-type adaptivity is its algorithmic complexity.

**r-adaptation**

The r-type strategy modifies the mesh degree by clustering existing nodes in areas/regions of large solution error. The r-method has been widely utilized with 2-D structured grids [19]. In reality, r-adaptivity can undoubtedly be regarded as one of the earliest (and in a sense, most primitive) grid adaptation techniques. The true potential of adaptive methods was, of course, only realized with unstructured meshes, which became well known due to their remarkable flexibility in modeling complex geometries. Algorithms for r-adaptation of

unstructured triangular meshes often make use of a spring-equilibrium technique to redistribute nodes in the mesh [2]. The utilization of r-adaptivity with unstructured triangular meshes has mostly been accounted for simulating thermal and compressible fluid flows [13]. Since the mesh topology (or connectivity) does not change, the r-type strategy is somewhat simple to implement. Then again, the number of degrees of freedom in the mesh remains fixed during the analysis, and, subsequently, the desired solution accuracy may never be achieved except if the analysis begins with a sufficient number of nodes. Moreover, r-type adaptation often produces highly skewed or stretched element., resulting in solution oscilIations or at least a seriously impaired convergence rate of the solution [24].

**h-adaptation**

The third type of mesh adaptation strategy is the h-type method, which generally involves subdivision of elements based on a set of primitive geometric operations. This technique is both simpIe and efficient. Unstructured meshes, made of triangles and tetrahedra, are particularly suited for h-refinement. A significant benefit of h-refinement is that new elements are completely nested within their forming coarse elements. This nested property of the refined meshes can be effectively exploited for constructing unstructured multigrid algorithms [21]. EIement subdivision, however, results in isotropic refinement, which is not satisfactory for strongly directional flow fields exhibiting extremely thin boundary layers and weak regions [21]. For such flow fields, the r-method is preferred because it can produce directionally stretched elements.

### 2.1.4 Adaptive Mesh Refinement(AMR)

The principle idea of AMR is to enable greater solution accuracy at lower cost, through a more optimal distribution of nodes for each computed solution. The basic steps in an AMR strategy are :

1. Computation of an initial solution.

2. Estimation of local solution error.

3. Modification of mesh based on estimated error.

4. Transferrence of solution onto adapted mesh.

5. Resumption of numerical solution procedure.

AMR is an approach, commonly used to maximize the accuracy or smoothness of the solution and/or minimize the computational cost of the computation. When solutions are calculated numerically, they are often limited to pre-determined quantified grids as in the cartesian plane which constitute the computational grid, or 'mesh'. Many problems in numerical analysis, however, do not require a uniform precision in the numerical grids used for graph plotting or computational simulation, and would be better suited if specific areas of the graph which needed precision could be refined in quantification only in the regions requiring the added precision. Adaptive mesh refinement provides such a dynamic programming environment for adapting the precision of the numerical computation based on the requirements of a computation problem in specific areas of multi-dimensional graphs which need precision while leaving the other regions of the multi-dimensional graphs at lower levels of precision and resolution. This dynamic technique of adapting computation precision to specific requirements has been accredited to **Marsha Berger**, **Joseph Oliger** ,and **Phillip Colella** who developed an algorithm for dynamic gridding called **local adaptive mesh refinement**. [26] A PhD thesis presented to University of Toronto, developed and tested an adaptive solution methodology for 3D incompressible flow simulations in domains of arbitrary complexity, by adopting the Zienkiewicz-Zhu patch recovery error estimator(LPR) to characterize the finite element solution error. [25] addressed the construction of an automatic selection of a suitable mesh refinement threshold in AMR. The method introduced set the threshold parameter automatically to get h-AMR algorithm almost parameterless. What the method does is, it allows in principle, to refine without hand-calibration of the pertinent regions.

[33] argued that immersed boundary methods coupled with adaptive mesh refinement (AMR) is a powerful tool to solve complex viscous incompressible flow problesms, especially in the case where we have moving and deforming boundaries. They gave an overview of ARM tools available currently, with an emphasis on block structured grids that are a natural fit to immersed methods, and discussed the future trends.

[20] investigated and analyzed the grid convergence issues for an adaptive mesh refinement (AMR) code. Their numerical results shows that, the AMR grid may have larger error than those unrefined unifrom grid. Another findings from their work is that, the numerical solution at the coarse–fine interface between different levels of the grid converges only in the first-order accuracy. Consequently, the error close to the coarse–fine interface can rapidly dominate the error in the other regions if the coarse–fine interface is active and not covered by the fine grid.

[1] in their research identified one short coming of all existing literatures on adaptive mesh refinement. They argued that, the techniques are usually applied to tetrahedra and hexahedra. Having this in mind, they used triangular prismatic element as the discretization shape for a Finite Element Method code with adaptivity. They further moved on to use five different marking strategies and compared the results obtained with different parameters.

[38] proposed s method that uses a locally refined mesh in spray region in spray modelling of engine combustion and emissions simulations. H-refinement adaptive method was employed and the results shows that the present scheme can achieve the same level of accuracy in modeling sprays with significantly lower computational cost as compared to a uniformly fine mesh.

[15] presented an extension of the a posteriori error estimation and goal-oriented mesh refinement approach from laminar to turbulent flows, which are governed by the Reynolds-averaged Navier–Stokes and $k-\omega$ turbulence model (RANS-$k-\omega$) equations. Discoutinuous Galerkin discretization of the (RANS-$k-\omega$) equations was the governing idea and was used within an adjoint-based error estimation and adaptive mesh refinement algorithm that targets the reduction of the discretization error in single as well as in multiple aerodynamic

force coefficient.

[9] described the computational algorithm used by parallel multigrid elliptic equation solver with adaptive mesh refinement. Their method and code uses truncation error estimates to adaptivly refine the grid as part of the solution process.

## 2.2 Reinforcement Learning

Reinforcement learning is an area of machine learning concerned with how intelligent agents learns what to-do, how to-do it by mapping situations to actions for the purpose of maximizing a numerical reward signal [3]. The learner (agent) is not informed which actions to take, however rather should determine which actions yield the most reward by performing some trials on them. In most interesting and challenging cases, actions taken by the agent may influence the immediate reward as well as the following situation and, through that, every resulting reward. These two characteristics—trial-and-error search and delayed reward—are the two generally significant distinguishing features of reinforcement learning.

Reinforcement learning is different from *supervised learning*. Supervised learning is task of learning a function that maps an input to an output based on example provided by a knowledgeable external supervisor.

Also, reinforcement learning is different from *unsupervised learning,* which is typically about finding structures hidden in collections of unlabeled data. Often than not, we are tempted to classify reinforcement learning as a kind of unsupervised learning because it does not depend on examples of correct behaviour, but that is not true because reinforcement learning is trying to maximize a reward signal instead of trying to find hidden structures which is in the case of unsupervised learning. Nevertheless, uncovering structure in an agent's experience can certainly be useful in reinforcement learning, but by itself does not address the reinforcement learning problem of maximizing a reward signal, therefore we consider reinforcement learning to be a third machine learning paradigm.

One of the difficulties that emerge in reinforcement learning, but not in other kinds of

learning, is the trade-off between exploration and exploitation. To obtain a lot of reward, a reinforcement learning agent must prefer actions that it has tried in the past and found to be effective in producing reward. However, to find such actions, it needs to try actions that it has not chosen previously. The agent needs to exploit what it has already experienced to acquire reward, however it additionally needs to explore in order to improve action selections later on. The problem is that neither exploration nor exploitation can be sought after without failing at the task. The agent must try a variety of actions and dynamically favor those that appear to be ideal.
Another important element of reinforcement learning is that it explicitly considers the entirety problem of a goal-directed agent interacting with an uncertain environment. This is in contrast to numerous methodologies that consider subproblems without addressing how they might fit into a larger picture.

### 2.2.1 Elements of Reinforcement Learning

Aside the agent and the environment, we can identify four main subelements of a reinforcement learning system: a policy, a reward signal, a value function, and, optionally, a model of the environment.

- **A policy** characterizes the learning agent's way of behaving at a given time. Generally speaking, a policy is a mapping from perceived states of the environment to actions to be taken when in those states. It relates to what in psychology would be called a set of stimulus–response rules or associations. At times the policy may be a simple function whereas in others it may involve extensive computation such as a search process. The policy is the core of a reinforcement learning agent in the sense that it alone is sufficient to determine behavior. In general, policies may be stochastic, specifying probabilities for each action.

- **A reward signal** defines the goal of a reinforcement learning problem. On every time step, the environment sends to the reinforcement learning agent a single number

called the reward. The agent's primary and only objective is to maximize the total reward it receives over the long run. The reward signal thus characterize what are the good and bad events for the agent. In a biological system, we can think of rewards as analogous to the experiences of pleasure or pain. The reward signal is the primary basis for altering the policy; if an action selected by the policy is followed by low reward, then the policy may be changed to select some other action in that situation in the future.

- **Value function** specifies what is good in the long run. In a simplest term, the value of a state is the total amount of reward an agent can expect to accumulate over the future, starting from that state. Whereas rewards determine the immediate, intrinsic desirability of environmental states, values indicate the long-term desirability of states after taking into account the states that are likely to follow and the rewards available in those states. It is much difficult to determine values than it is to determine rewards. Rewards are basically given directly by the environment, but values must be estimated and re-estimated from the sequences of observations an agent makes over its entire lifetime. An established fact is that, the most important component of almost all reinforcement learning algorithms is a method for efficiently estimating values.

- **Model of the environment**: The model of the environment mimics the behaviour of the environment, or more generally, that allows inferences to be made about how the environment will behave. Methods for solving reinforcement learning problems that use models and planning are called model-based methods, as opposed to simpler model-free methods that are explicitly trial-and-error learners—viewed as almost the opposite of planning

### 2.2.2 History of Reinforcement Learning

The early history of reinforcement learning has two primary strings, both long and rich, that were sought after independently prior to intertwining in present day reinforcement learning. One string concerns learning by experimentation, and originated from the psychology of animal learning. This string goes through some of the earliest work in artificial intelligence and what's more, prompted the recovery of reinforcement learning in the mid 1980s. The second string concerns the problem of optimal control and its solution using value functions and dynamic programming. Generally, this string didn't include learning. The two strings were generally independent, yet got interrelated somewhat around a third, less distinct string concerning temporal difference techniques. The string focusing on experimentation learning is the one with which we are most familiar and about which we have the most to say in this short history. Before doing that, however, we briefly discuss the optimal control thread.

The idea "optimal control" came into use in the late 1950s to describe the problem of designing a controller to minimize or maximize a measure of a dynamical system's behavior over time. One of the methods to this problem was developed in the mid-1950s by Richard Bellman and others through extending a nineteenth century theory of Hamilton and Jacobi. This methodology uses the ideas of a dynamical system's state and of a value function, or "optimal return function," to characterize a functional equation, now often called the Bellman equation. The class of methods for solving optimal control problems by solving this equation came to be known as dynamic programming. Bellman also presented the discrete stochastic version of the optimal control problem known as Markov decision processes (MDPs) [4]. Ronald [18] came up with the policy iteration method for MDPs. All of these are essential elements underlying the theory and algorithms of modern reinforcement learning.

Dynamic programming is widely considered the only viable way of solving general stochastic optimal control problems. It suffers from what Bellman called "the curse of dimensionality," implying that its computational requirements grow exponentially with the number of

state variables, however, it is still far more efficient and more widely applicable than any other general method. Connections between optimal control and dynamic programming, on the one hand, and learning, on the other, were slow to be perceived. We cannot make certain about what accounted for this partition, yet its fundamental cause was likely the partition between the disciplines involved and their diverse objectives. Also contributing may have been the common perspective on dynamic programming as an offline computation depending basically on precise system models and analytic solutions to the Bellman equation. Further, the simplest form of dynamic programming is a computation that proceeds backwards in time, making it challenging to see how it could be involved in a learning process that must proceed in a forward direction. Some of the earliest work in dynamic programming, such as that by [5], might now be classified as following a learning approach. [36] work on *An adaptive optimal controller for discrete-time Markov environments* certainly qualifies as a combination of learning and dynamic-programming ideas. [35] argued explicitly for prominent interrelation of dynamic programming and learning methods and for dynamic programming's significance to understanding neural and cognitive mechanisms. This integration of dynamic programming methods with online learning was not in use until the work [34] whose treatment of reinforcement learning utilizing the MDP formalism has been widely embraced.

Since then these relationships have been widely evolved by many researchers, most particularly by [7], who instituted the expression "neurodynamic programming" to allude to the combination of dynamic programming and artificial neural networks. Another term currently in use is "approximate dynamic programming." These different methodologies emphasize different aspects of the subject, yet they all share with reinforcement learning an interest in compassing the classical shortcomings of dynamic programming. On the basis of this analogy, all the work in optimal control can be considered as a work in reinforcement learning. Now, considering the other major string leading to the current field of reinforcement learning, the string based on the idea of trial-and-error learning.

According to American psychologist [37] the idea of experimentation learning goes as far

back as the 1850s to Alexander Bain's discussion of learning by "groping and experiment" and more explicitly to the British ethologist and psychologist Conway Lloyd Morgan's 1894 utilization of the term to describe his observations of animal behavior. Maybe the first to concisely communicate the substance of experimentation learning as a principle of learning was Edward Thorndike:

"The greater the satisfaction or discomfort, the greater the strengthening or weakening of the bond. (Thorndike, 1911, p. 244)" This statement according to Thordike is called the "Law of Effect" because it describes the effect of reinforcing events on the tendency to select actions.

The term "reinforcement" with regards to animal learning came into use well after Thorndike's expression of the Law of Effect, first appearing in this context in the 1927 English translation of Pavlov's monograph on conditioned reflexes. Pavlov described reinforcement as the strengthening of a pattern of behavior due to an animal receiving a stimulus—a reinforcer—in an appropriate temporal relationship with another stimulus or with a response. To be considered reinforcer, the strengthening or weakening should persevere after the reinforcer is withdrawn; a stimulus that merely attracts an animal's attention or that energizes its behavior without producing lasting changes would not be considered a reinforcer.

The idea of implementing trial-and-error learning in a computer appeared among the earliest thoughts about the possibility of artificial intelligence. In a 1948 report, Alan Turing described a design for a "pleasure-pain system" that worked along the lines of the Law of Effect:

"When a configuration is reached for which the action is undetermined, a random choice for the missing data is made and the appropriate entry is made in the description, tentatively, and is applied. When a pain stimulus occurs all tentative entries are cancelled, and when a pleasure stimulus occurs they are all made permanent. (Turing, 1948)". Numerous ingenious electro-mechanical machines were built that demonstrated trial-and- error learning. The earliest may have been a machine built by [32] that was able to find its way through a simple maze and remember the path through the settings of switches. In 1951 W. Grey

Walter built a version of his "mechanical tortoise" (Walter, 1950) capable of a simple form of learning. In 1952 Claude Shannon demonstrated a maze-running mouse named Theseus that used trial and error to discover its way through a maze, with the actual maze recalling the successful directions via magnets and relays under its floor. [14] described a computerized simulation of a neural-network learning machine that learned by experimentation. But their interests soon shifted from experimentation learning to generalization and pattern recognition, that is, from reinforcement learning to supervised learning [10]. This brought about the confusion about the relationship between types of learning. Many researchers believed that they were studying reinforcement learning when they were actually studying supervised learning. For example, pioneers of artificial neural network such as Rosenblatt(1962) and Widrow and Hoff(1960) were clearly motivated by reinforement learning, they used the language of rewards and punishment yet the systems they examined were supervised learning systems suitable for pattern recognition and perceptual learning. Currently, some reseachers blur the distinction between these types of learning. One example is that, some artifial neural network textbooks have used the term"trial-and-error" to describe networks that are learning from training examples. This is a justifiable disarray on the grounds that these networks use error information to update connection weights, yet this misses the essential character of trial-and-error learning as choosing actions on the basis of evaluative feedback that does not depend on knowledge of what the correct action should be.

In the 1960s the terms "reinforcement" and "reinforcement learning" were used in the engineering literature , probably for the first time to describe engineering uses of trial-and-error learning (e.g., Waltz and Fu, 1965; Mendel, 1966; Fu, 1970; Mendel and McClaren, 1970). In particular, influential was Minsky's paper "Steps Toward Artificial Intelligence" [22] which discussed several issues relevant to trial-and-error learning, including prediction, expectation, and what he called the *basic credit-assignment problem for complex reinforcement learning systems*.

# Chapter 3

# A Posteriori Error Estimates for Finite Element Approximations

The a posteriori error estimation of finite element approximations of elliptic boundary value problems have reached some state of maturity. A posteriori error estimates are computable quantities in terms of the discrete solution and data, which are instrumental for adaptive mesh refinement (and coarsening), error control, and equidistribution of the computational effort.

There are different concepts such as

- Residual type a posteriori error estimators
- Gradient recovery(ZZ) type a posteriori error estimators
- Hierarchical type a posteriori error estimators
- Error estimators based on local averaging
- Error estimators based on the goal oriented weighted dual approach

In this research , our focus will be on **residual type estimators** owing to relatively cheap computational cost as compared to other estimators and **gradient recovery(ZZ) estimators**.

## 3.1 A Posteriori Error Estimators

Lets consider the following **model problem**

Let $\Omega$ be bounded simply connected polygonal domain in Euclidean space $\mathbb{R}^2$ with boundary $\Gamma = \Gamma_D \cup \Gamma_N, \Gamma_D \cap \Gamma_N = \emptyset$ and consider the elliptic boundary value problem

$$\begin{aligned} Lu := -\nabla \cdot\ a\ \nabla\ u = f \quad & in\ \Omega \\ u = 0 \quad & on\ \Gamma_D \\ n.a\ \nabla\ u = g \quad & on\ \Gamma_N, \end{aligned} \tag{3.1}$$

where $f \in L^2(\Omega),\ g \in H^{1/2}(\Gamma_N)$ and $a = (a_{ij})_{i,j=1}^2$ is supposed to be a matrix-valued function with entries $a_{ij} \in L^\infty(\Omega)$, that is symmetric, $a_{ij}(x) = a_{ji}(x)$ for almost all x $\in\ \Omega,\ 1 \le i,j \le 2$ and uniformly positive definite in the sense that for almost all $x \in \Omega$

$$\sum_{i,j=1}^{2} a_{i,j}\xi_i\xi_j\ \ge\ \alpha\left|\xi\right|^2,$$

$\xi \in \mathbb{R}^2, \alpha > 0$

The vector $n$ denotes the exterior unit normal vector on $\Gamma_N$. We further set

$$\bar{\alpha}\colon = \max_{1\le i,j\le 2} ||a_{ij}||_\infty$$

Setting $V := \{v \in H^1(\Omega)|\ v|_{\Gamma_D} = 0\}$, the weak formulation of (3.1) is as follows:

Find $u \in V$ such that

$$\begin{aligned} & a(v,u) = l(v), \quad v \in V \\ & where \\ & a(v,u) := \int_\Omega a\ \nabla\ v \cdot \nabla\ u\ dx.\ \ v,u \in V \\ & l(v) := \int_\Omega f\ v dx + \int_{\Gamma_N} g\ v \quad v \in V \end{aligned} \tag{3.2}$$

For a given geometrically conforming simplicial triangulation $\mathcal{T}_h$ of $\Omega$ we denote by $V_h := \{v_h \in V\ |v_h|K \in P_1(K), K \in \mathcal{T}_h\}$ the trial space of continous, piecewise linear finite

elements with respect to $\mathcal{T}_h$. Note that $P_k(K),\ k \geq 0$, denotes the linear space of polynomials of degree $\leq k$ on K.

We will refer to $N_h(D)$ and $\mathcal{E}_h(D),\ D \subseteq \bar{\Omega}$ as the sets of vertices and edges of $\mathcal{T}_h$ on $D$.

The **conforming P1 approximation** of 3.1 reads as follows:

Find $u_h \in V_h$ : such that

$$a(u_h, v_h) = l(v_h),\ v_h \in V_h \tag{3.3}$$

**Definition 3.1.1.** ***Existence and uniqueness of solution.***

*For the existence and uniqueness of a solution to the variational form in 3.2, we define the V-elliptic bilinear form as follows and result to the Lax-Milgram theorem for its establishment.*

*A bilinear form $a(\cdot,\cdot) : V \times V \mapsto \mathbb{R}$ is called, $V-$ elliptic if there exists a constant $\alpha > 0$ such that*

$$|a(u,v)| \leq \alpha ||u||_v^2, \qquad u \in V \tag{3.4}$$

**Theorem 3.1.2.** ***Lax-Milgram Lemma***

*Let $V$ be a Hilbert space with dual $V^*$ and assume that $a(\cdot,\cdot) : V \times V \mapsto \mathbb{R}$ is bounded, $V-$ elliptic bilinear form and $l \in V^*$. Then, the variational equation 3.2 admits a unique solution $u \in V$.*

Now, assuming that $\bar{u_h} \in V_h$ is some **iterative approximation** of $u_h \in V_h$, we are interested in the **total error**

$$e := u - \bar{u_h} \tag{3.5}$$

which is the sum of the **discretization error** $e_d := u - u_h$ and the **iteration error** $e_{it} := u_h - \bar{u_h}$. It is easy to see that the total error $e$ is in $V := {H^1}_{0,\Gamma_D}(\Omega)$ and satisfies the **error equation**

$$a(e,v) = r(v), \qquad v \in V, \tag{3.6}$$

where $r(.)$ stands for the **residual** with respect to the computed approximation $\bar{u_h}$

$$r(v) := \int_\Omega fv \; dx + \int_{\Gamma_N} gv \; d\sigma - a(\bar{u_h}, v), \quad v \in V \tag{3.7}$$

We are interested in cheaply computable **a posteriori error estimator** $\eta$ of the total error $e$ consisting of elementwise error contributions $\eta_K, \; K \in \mathcal{T}_h$ and edgewise error contributions $\eta_E, \; E \in \mathcal{E}_h$ in the sense that

$$\eta^2 = \sum_{K \in \mathcal{T}_h} \eta_K^2 + \sum_{E \in \mathcal{E}_h} \eta_E^2 \tag{3.8}$$

which does provide **a lower and an upper bound** for $e$ according to

$$\gamma \; \eta \leq \; ||e||_{1,\Omega} \; \leq \; \Gamma \eta \tag{3.9}$$

with constants $0 < \gamma \leq \Gamma$ depending only on the ellipticity constants and on the shape regularity of the triangulation $\mathcal{T}_h$.

We may use the local error terms $\eta_K$ and $\eta_E$ as a criterion for **local refinement** of the elements $K \in \mathcal{T}_h$. Among several **refinement strategies**, the so called **mean-value strategy** is as follows:

$$\begin{aligned} \bar{\eta_K} &:= \frac{1}{n_K} \sum_{K \in \mathcal{T}_h} n_K \\ \bar{\eta_E} &:= \frac{1}{n_E} \sum_{E \in \mathcal{E}_h} n_E \end{aligned} \tag{3.10}$$

where $n_K := card \; \mathcal{T}_h$ and $n_E := card \; \mathcal{E}_h$.

Mark an element $K \in \mathcal{T}_h$ and an edge $E \in \mathcal{E}_h$ for refinement, if

$$\begin{aligned} \eta_K &\geq \sigma \; \bar{\eta_K} \\ \eta_E &\geq \sigma \; \bar{\eta_E} \end{aligned} \tag{3.11}$$

where $0 < \sigma \leq 1$ is some appropriate **safety factor,** e.g , $\sigma = 0.9$

**Definition 3.1.3.** ***Efficient and Reliable a posteriori error estimators***

*An a posteriori error estimator $\eta$ satisfying*

$$||e||_{1,\Omega} \leq \Gamma_\eta \tag{3.12}$$

*is called* ***reliable****, since it ensures a sufficient refinement in the sense that the $H^1$-norm of the total error* $\boldsymbol{e}$ *will be bounded by a quantity of the same order of magnitude as a user-prescribed accuracy, if this accuracy is tested by $\eta$.*
*On the other hand, an a posteriori error estimator $\eta$ for which*

$$\gamma\eta \leq ||e||_{1,\Omega} \tag{3.13}$$

*is said to be* ***efficient****, since it underestimates the $H^1$-norm of the total error e and thus prevents too much refinement.*

## 3.2 Residual Based a Posteriori Error Estimators

The residual based a posteriori error estimator can be derived by viewing the residual as an element of the dual space V and evaluating it with respect to the dual norm.

### 3.2.1 Upper Bound for the Total Error

An important tool in the construction of an upper bound for the total error is **Clément's interpolation operator** and **Weighted Clément's interpolation operator** which are defined as follows:

**Definition 3.2.1.** ***Clément's interpolation operator***

*For $p \in N_h(\Omega) \cup N_h(\Gamma_N)$ we denote by $\varphi_p$ the basis function in $V_h$ with supporting point $p$ and we refer to $D_p$ as the set*

$$D_p := \bigcup \{K \in \mathcal{T}_h | p \in N_h(T)\}.0 \tag{3.14}$$

*We refer to $\pi_p$ as the $L^2$-projection onto $P_1(D_p)$, i.e,*

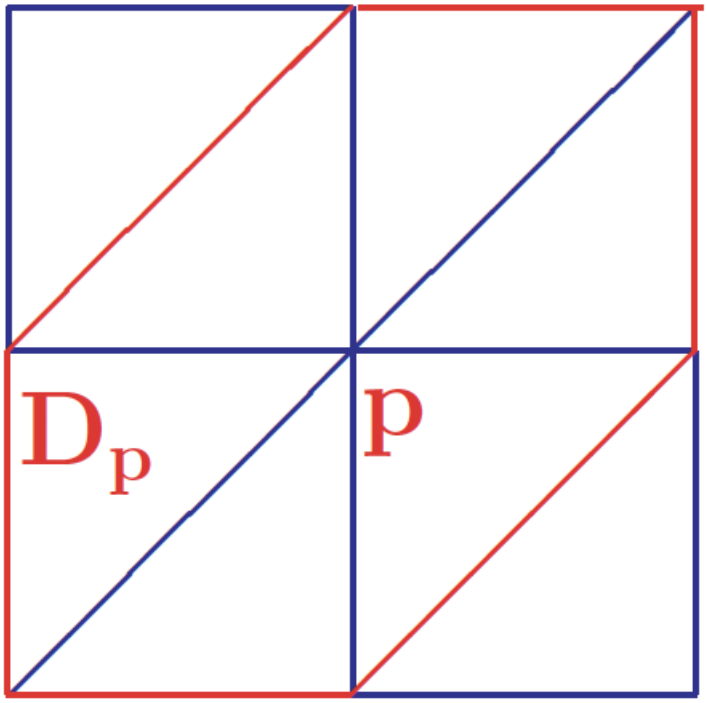



Figure 3.1: Clément's interpolation operator(definition)

$$(\pi(v), w)_{0,D_p} = (v, w)_{0,D_p}, \quad w \in P_1(D_p)$$

*where* $(.,.)_{o,D_p}$ *stands for the* $L^2$*-inner product on* $L^2(D_p) * L^2(D_p)$

*Then,* ***Clément's interpolation operator*** $P_C$ *is defined as follows*

$$\begin{aligned} P_C &: L^2(\Omega) \to V_h \\ P_C v &:= \sum_{p \in N_h(\Omega) \cup N_h(\Gamma_N)} \pi_p(v) \varphi_p \end{aligned} \tag{3.15}$$

*In order to establish* ***local approximation properties*** *of Clément's interpolation operator, for* $K \in \mathcal{T}_h$ *and* $E \in \mathcal{E}_h(\Omega) \cup \mathcal{E}_h(\Gamma_N)$ *we introduce sets*

$$D_K^{(1)} := \bigcup \{K' \in \mathcal{T}_h | N_h(K') \cap N_h(K) \neq \emptyset\} \tag{3.16}$$

$$D_E^{(1)} := \bigcup \{K' \in \mathcal{T}_h | N_h(E) \cap N_h(K') \neq \emptyset\} \tag{3.17}$$

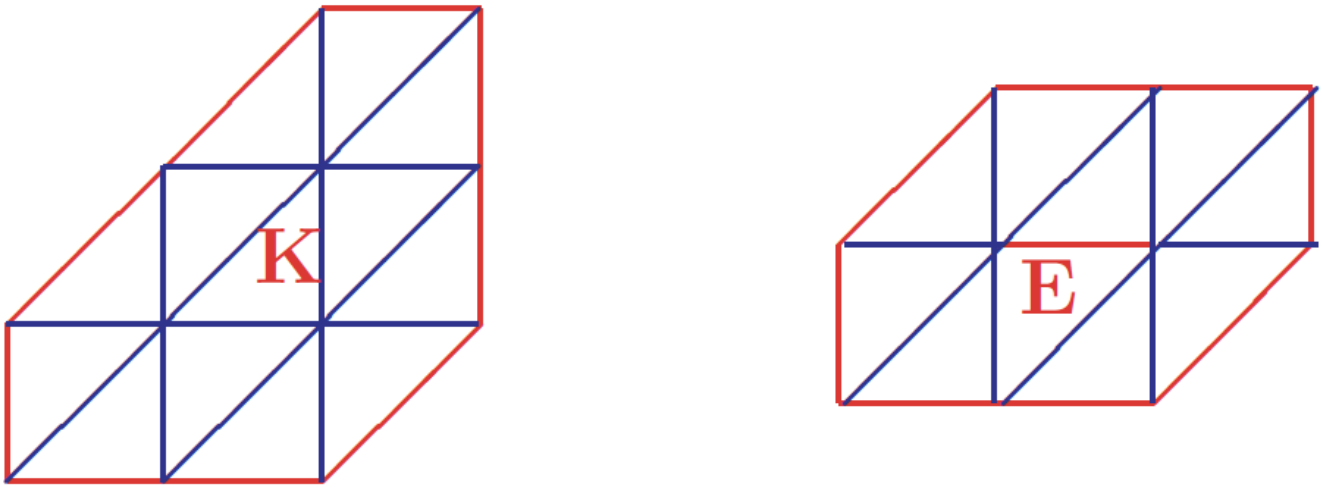



Figure 3.2: Clément's interpolation operator(properties)

**Definition 3.2.2.** ***Weighted Clément-type interpolation*** *Let $\partial^2\mathcal{T}_h$ be the set of nodes, $\Phi_z$ be the basic function of $V_h$ on $z \in \partial^2\mathcal{T}_h, \omega_z = supp\Phi(x)$ and set*

$$\psi_z = \frac{\phi_z}{\psi}, \qquad \psi = \sum_{z\in\Lambda} \phi_z$$

*where $\Lambda = \partial^2\mathcal{T}_h \backslash \partial\Omega$. For $v \in V$, the weighted Clément-type interpolation of $v$ is defined by:*

$$\pi v = \sum_{z\in\Lambda} v_z\phi_z \in V_h, \quad v_z = \frac{(\psi_z, v)}{(\phi_z, 1)}$$

**Theorem 3.2.3.** ***Approximation properties of Clément's interpolation operators.***

*For $K \in \mathcal{T}_h$ and $E \in \mathcal{E}_h(\Omega) \cup \mathcal{E}_h(\Gamma_N)$. Let $D_K^{(1)}$ and $D_E^{(1)}$ be given by (25) and $P_C$ be Clément's interpolation operator as given by (24). Then, there exist constants $C_v > 0, 1 \leq v \leq 5$, depending only on the shape regularity of $\mathcal{T}_h$ such that for all $v \in V_h$:*

$$|P_C\ v||_{0,K} \ \leq C_1\ ||v||_{0,D_K^{(1)}}, \tag{3.18}$$

$$||P_C\ v||_{0,E}\ \leq C_2\ ||v||_{0,D_E^{(1)}}, \tag{3.19}$$

$$||\nabla\ P_C v||_{0,K} \leq C_3\ ||\nabla\ v||_{0,D_K^{(1)}}, \tag{3.20}$$

$$||v - P_C\ v||_{0,K} \leq C_4\ h_K\ ||v||_{1,D_K^{(1)}}, \tag{3.21}$$

$$|v - P_C\ v||_{0,E} \leq C_5\ h_E^{1/2}\ ||v||_{1,D_E^{(1)}} \tag{3.22}$$

*where* $h_K := diam\ K$ *and* $h_E := |E|$.

*Further, there exist the constants* $C_6, C_7 > 0$ *depending only on the shape regularity of* $\mathcal{T}_h$ *such that:*

$$(\sum_{K\in\tau_h} ||v||^2_{\mu,D_K^{(1)}})^{1/2}\ \leq C_6\ ||v||_{\mu,\Omega}, \qquad 0 \leq \mu \leq 1 \tag{3.23}$$

$$(\sum_{E\in\mathcal{E}_h(\Omega)\cup\mathcal{E}_h(\Gamma_N)} ||v||^2_{\mu,D_E^{(1)}})^{1/2}\ \leq\ C_7\ ||v||_{\mu,\Omega}, \quad 0 \leq \mu \leq 1 \tag{3.24}$$

*Furthermore if* $v \in V$, *then the following holds for the weighted Clément's interpolation operator* $\pi$

$$\sum_{K\in\mathcal{T}_h} ||h_K^{-1}(v - \pi v)||^2_{0,K} \leq C|v|^2_{1,\Omega}, \quad \forall v \in V \tag{3.25}$$

$$|\pi v|^2_{1,\Omega} \leq C|v|^2_{1,\Omega} \quad \forall v \in V \tag{3.26}$$

*Again,* $\forall f \in L^2(\Omega)$,

$$\int_\Omega f(v - \pi v) \leq \sum_{z\in\Lambda} \int_{\omega_z} |f - f_z|\ |v - v_z\psi|, \tag{3.27}$$

$$\sum_{z\in\Lambda} \int_{\omega_z} h^{-2}|v - v_z\psi|^2 \leq C|v|^2_{1,\Omega} \tag{3.28}$$

*where* $f_z$ *is a constant on* $\omega_z$ *and* $h_z = max_{K\subset\omega_z} h_K$

*We have now provided all prerequisites to establish an upper bound for the total error* $e$ *measured in the* $H^1$*-norm. For the functions* $v_h \in W_0(\Omega; \mathcal{T}_h)$. *We further refer to* $[V_h]_J$ *as the jump across the common edge* $E \in \mathcal{E}_h(\Omega)$ *of two adjacent elements* $K_1, K_2 \in \mathcal{T}_h$.

$$[v_h]_J := v_h|K_1 - v_h|_{K_2} \tag{3.29}$$

**Theorem 3.2.4.** ***Upper Bound for the Total Error***

*There exist constants* $\Gamma_R, \Gamma_{osc} > 0$ *and* $\Gamma_{it} > 0$ *depending only on the elipticity constants and the shape regularity of* $\mathcal{T}_h$ *such that:*

$$||e||_{1,\Omega} \leq \Gamma_R \eta_R + \Gamma_{osc} osc + \eta_{it} ||e_{it}||_{1,\Omega}, \tag{3.30}$$

*where the* ***element and edge residuals*** *are given by*

$$\begin{aligned}
\eta_R &:= \sum_{v=1}^{3} \eta_R^{(v)} \\
\eta_R^{(1)} &:= (\sum_{K \in \mathcal{T}_h} h_T^2 ||\pi_h f - L\ \bar{u_h}\ ||_{0,K}^2)^{1/2} \\
\eta_R^{(2)} &:= (\sum_{E \in \mathcal{E}_h(\Gamma_N)} h_E ||\pi_h g - n_E.a\ \nabla\ \bar{u_h}\ ||_{0,E}^2)^{1/2} \\
\eta_R^{(3)} &:= (\sum_{E \in \mathcal{E}_h(\Omega)} h_E ||[n_E.a\ \nabla\ \bar{u_h}]_J||_{0,E}^2)^{1/2}
\end{aligned} \tag{3.31}$$

*and Osc stnads for the data oscillations*

$$\begin{aligned}
osc &:= (\sum_{K \in \mathcal{T}_h} osc_K^2 + \sum_{E \in \mathcal{E}_h(\Gamma_N)} osc_E^2)^{1/2} \\
osc_K &:= h_T ||f - \pi_h f||_{0,K}, \qquad osc_E := h_E ||g - \pi_h g||_{0,E},
\end{aligned} \tag{3.32}$$

*Proof.* Setting $v = e$ in 3.6, we obtain

$$\underline{\alpha}\ ||e||_{1,\Omega}^2 \ \leq\ a(e,e) = r(e) = r(P_C e)\ +\ r(e - P_C e) \tag{3.33}$$

Taking advantage of 3.2, for the first term on the right-hand side of 3.33 we get

$$\begin{aligned}
r(P_C e) &= \int_\Omega f\ P_C e\ dx\ + \int_{\Gamma_N} g\ P_C e\ d\sigma - \ a(\bar{u_h}, P_C e) \\
&= \sum_{K \in \mathcal{T}_h} a|_K (u_h - \bar{u_h}, P_C e)
\end{aligned} \tag{3.34}$$

Using 3.21, the Schwarz inequality, and observing 3.24, it follows that

$$\begin{aligned}
r(P_C e)\ &\leq \bar{\alpha}\ C_3 \sum_{K \in \mathcal{T}_h} ||u_h - \bar{u_h}\ ||_{1,D_K^{(1)}} \\
&\leq \bar{\alpha}\ \ C_3 (\sum_{K \in \mathcal{T}_h} ||u_h - \bar{u_h}\ ||_{1,K}^2)^{1/2} (\sum_{K \in \mathcal{T}_h} ||e||_{1,D_K^{(1)}}^2) \\
&\leq \bar{\alpha}\ C_3\ C_6\ ||e_{it}||_{1,\Omega}\ ||e||_{1,\Omega}
\end{aligned} \tag{3.35}$$

On the other hand, for the second term on the right-hand side of 3.33, Green's formula yields

$$\begin{aligned}
r(e-P_Ce) &= \int_\Omega f\ (e-P_Ce)\ dx + \int_{\Gamma_N} g\ (e-P_Ce)\ d\sigma \\
&+ \sum_{K\in\mathcal{T}_h}\int_K \underbrace{\nabla\cdot a\ \nabla\ \bar{u_h}}_{=-L\bar{u_h}}(e-P_Ce)dx \\
&- \sum_{K\in\mathcal{T}_h}\int_{\partial K} n_{\partial K}.a\ \nabla\ \bar{u_h}\ c \\
&= \sum_{K\in\mathcal{T}_h}\int_K (\pi_h f - L\ \bar{u_h})(e-P_Ce)dx \\
&+ \sum_{E\in\mathcal{E}_h(\Omega)}\int_E \Big[\ n_E.\ a\ \nabla\ \bar{u_h}\Big]_j\ (e-P_Ce)\ d\sigma \\
&+ \sum_{E\in\mathcal{E}_h(\Gamma_N)}\int_E (\pi_h g - n_E.\ a\ \nabla\ \bar{u_h})(e-P_Ce)\ d\sigma \\
&+ \sum_{K\in\mathcal{T}_h}\int_K (f-\pi_h f)\ (e-P_Ce)\ dx \\
&+ \sum_{E\in\mathcal{E}_h(\Gamma_N)}\int_E (g-\pi_h g)\ (e-P_Ce)d\sigma
\end{aligned}$$

In view of 3.19,3.20 and 3.24, 3.30 it follows that

$$\begin{aligned}
r(e-P_Ce) \le\ & C_1\ C_6(\sum_{K\mathcal{T}_h} h_K^2||\pi_h f\ - L\ \bar{u_h}\ ||_{0,K}^2)^{1/2}\ ||e||_{1,\Omega} \\
&+ C_2C_7(\sum_{E\in\mathcal{E}_h(\Omega)} h_E||\left[n_E.a\nabla\ \bar{u_h}\right]_J\ ||_{0,E}^2)^{1/2}||e||_{1,\Omega} \\
&+ C_2C_7(\sum_{E\in\mathcal{E}_h(\Gamma_N)} h_E||\pi_h g\ - n_E.a\ \nabla\ \bar{u_h}\ ||_{0,E}^2)^{1/2}||e||_{1,\Omega} \qquad (3.36) \\
&+ C_1\ C_6(\sum_{K\in\mathcal{T}_h} h_K^2||f-\pi_h f||_{0,K}^2)^{1/2}||e||_{1,\Omega} \\
&+ C_2\ C_7(\sum_{E\in\mathcal{E}_h(\Gamma_N)} h_E(||g-\pi_h g||_{0,E}^2)^{1/2}||e||_{1,\Omega}
\end{aligned}$$

□

Using 3.35, 3.36 in 3.33 the assertion follows with

$\Gamma_R = \Gamma_{osc} := \underline{\alpha}^{-1} max(C_1C_6, C_2C_7)$ and $\Gamma_{it} := \underline{\alpha}^{-1}\ \bar{\alpha}\ C_3C_6$

For the construction of a **lower bound**, we will now show that the local contriburions

$\eta_{R,K}^{(v)} :=\ \eta^(v)_R|_K,\ K \in \mathcal{T}_h,\ 1 \leq v \leq 3$

of the residual based error estimator $\eta_R$ do locally provide lower bounds for the total error $e$. For this purpose we need appropriate localized polynomial functions defined on the elements $K$ of the triangulation and the edges $E \in \mathcal{E}_h(\Omega) \cap E \in \mathcal{E}_h(\Gamma_N)$, respectivity. Such functions are given by the **triangle-bubble functions** $\psi_K$ and the **edge-bubble functions** $\psi_E$.

In particular, denoting by $\lambda_i^K,\ 1 \leq i \leq 3$, the **barycentric coordinates** of $K \in \mathcal{T}_h$, then the triangle-bubble function $\psi_K$ is defined by means of

$$\psi_K := 27\lambda_1^K\lambda_2^K\lambda_3^K \tag{3.37}$$

Note that $supp\ \psi_K = K_{int},\ \ i.e,\ \ \psi_K|_{\delta K} = 0,\ K \in \mathcal{T}_h$

On the other hand, for $E \in \mathcal{E}_h(\Omega) \cup \mathcal{E}_h(\Gamma_N)$ and $K \in \mathcal{T}_h$ such that $E \subset \delta K$, and $p_i^k \in \mathcal{N}_h(T),\ 1 \leq i \leq 2,$ we introduce the edge-bubble functions $\psi_E$ according to

$$\psi_E := 4\lambda_1^K\lambda_2^K \tag{3.38}$$

Note that $\psi_E|_{E'} = 0$ for $E' \in \mathcal{E}_h(\Omega),\ E' \neq E$.

The bubble functions $\psi_K$ and $\psi_E$ have the following important properties that can be easily verified taking advantage of the affine equivalence of the elements:

**Lemma 3.2.5.** ***Basic Properties of the Bubble Functions: Part I***

*There exist constants $C_v > 0,\ 8 \leq v \leq 12$, depending only on the shape regularity of the triangulations $\mathcal{T}_h$ such that*

$$||p_h||_{0,K}^2 \quad \leq \quad C_8 \int_K p_h^2\psi_K dx, \qquad p_h \in P_1(K) \tag{3.39}$$

$$||p_h||^2_{0,E} \leq C_9 \int_K p_h^2 \psi_E d\sigma, \qquad p_h \in P_1(E) \tag{3.40}$$

$$||p_h \psi_K||_{1,K} \leq C_{10} h_K^{-1} \ ||p_h||_{0,K}, \qquad p_h \in P_1(K) \tag{3.41}$$

$$||p_h \psi_K||_{0,K} \leq C_{11} ||p_h||_{0,K}, \qquad p_h \in P_1(K) \tag{3.42}$$

$$||p_h \psi_K||_{0,E} \leq C_{12} ||p_h||_{0,E}, \qquad p_h \in P_1(E) \tag{3.43}$$

*For functions* $p_h \in P_1(K), E \in \mathcal{E}_h(\Omega) \cup \mathcal{E}_h(\Gamma_N)$ *we further need an extension* $p_h^E \in L^2(K)$ *where* $K \in \mathcal{T}_h$ *such that* $E \subset \delta K$. *For this purpose we fix some* $E' \subset \delta K, E' \neq E$ *and for* $x \in K$ *denoted by* $x_E$ *that point on* $E$ *such that* $(x - x_E)||E'$ *For* $p_h \in P_1(E)$ *we then set*

$$p_h^E := p_h(x_E) \tag{3.44}$$

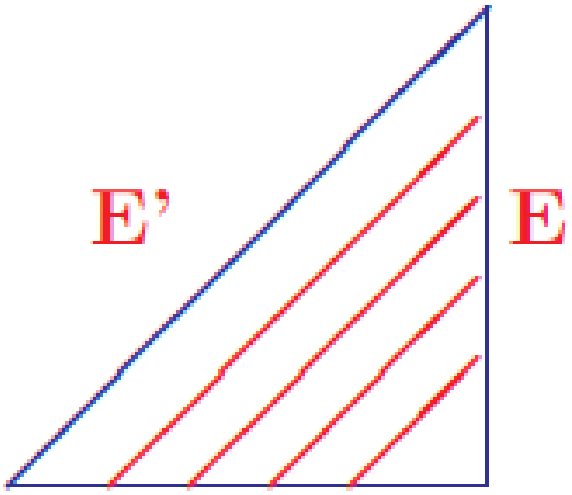



Figure 3.3: Level lines of the extension $p_h^E$

*Further, for* $E \in \mathcal{E}_h(\Omega) \cup \mathcal{E}_h(\Gamma_N)$ *we define* $D_E^{(2)}$ *as the union of elements* $K \in \mathcal{T}_h$ *containing* $E$ *as a common edge.*

$$D_E^{(2)} := \bigcup \{K \in \mathcal{T}_h | E \in \mathcal{E}(K)\} \tag{3.45}$$

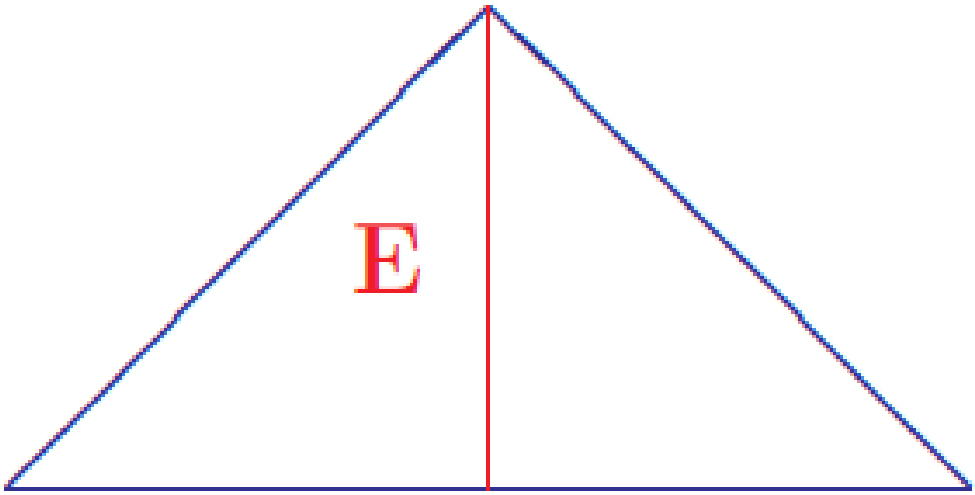



Figure 3.4: The set $D_E^{(2)}$

**Lemma 3.2.6.** ***Basic Properties of the Bubble Functions: Part II***

*There exist constants $C_v > 0,\ 13 \leq v \leq 14$, depending only on the shape regularity of the triangulations $\mathcal{T}_h$ such that*

$$||p_h^E \psi_E||_{1,D_E^{(2)}} \leq C_{13} h_E^{-1/2} ||p_h||_{0,e},\ p_h \in P_1(E) \tag{3.46}$$

$$|p_h^E \psi_E||_{0,D_E^{(2)}} \leq C_{13} h_E^{1/2} ||p_h||_{0,E},\ p_h \in P_1(E) \tag{3.47}$$

*Further there exists a constant $C_15 > 0$ independent of $h_K, h_E$ such that for all $v \in V$ and $\mu = 0.1$*

$$(\sum_{E \in \mathcal{E}_h(\Omega) \cup \mathcal{E}_h(\Gamma_N)} h_E^{1-\mu} ||v||^2_{\mu, D_E^{(2)}})^{1/2} \leq C_{15} (\sum_{K \in \mathcal{T}_h} h_K^{1-\mu} ||v||^2_{\mu,K})^{1/2} \tag{3.48}$$

*We are now able to prove that up to higher order terms the estimator $\eta_R$also does provide a lower bound for the total error $\boldsymbol{e}$:*

**Theorem 3.2.7.** ***Lower bound for the total error***

*There exist constants $\gamma_R, \gamma_E > 0$, depending only on $\bar{\alpha}$ and on the shape regularity of $\mathcal{T}_h$ such that*

$$\gamma_R \eta_R - \gamma_E osc \leq ||e||_{1,\Omega}, \tag{3.49}$$

*where $\eta_R$ and osc are given as in the previous theorem. The theorem can be proved by series of results which establish upper bounds for the local contributions $\eta_{R,T}^{(v)} \leq v \leq 3$ of the estimator $\eta_R$*

**Lemma 3.2.8.** ***Upper bounds for the local contributions***

1. *Let $K \in \mathcal{T}_h$, there there holds :*

$$h_K ||\pi_h f - L\, \bar{u}_h\, ||_{0,K} \;\leq \bar{\alpha}\, C_8\, C_{10} ||e||_{1,K} + C_8\, C_{11} h_K ||f - \pi_h f||_{0,K} \tag{3.50}$$

2. *Let $E \in (\Omega)$. Then there holds:*

$$\begin{aligned} h_E^{1/2} |||[n_E \cdot \nabla\, \bar{u}]_J||_{0,E} \leq & \bar{\alpha}\, C_9 C_{13} ||e||_{1,D_E^{(2)}} \\ + C_9 C_{14} h_E ||f - \pi_h f||_{0,D_E^{(2)}} + C_9 C_{14} h_E & ||\pi_h f - L\, \bar{u}_h\, ||_{0,D_E^{(2)}} \end{aligned} \tag{3.51}$$

3. *Let $E \in \mathcal{E}_h(\Gamma_N)$. Then there holds:*

$$\begin{aligned} h_E^{1/2} ||\pi_h g - n_E \cdot \nabla\, \bar{u}_h\, ||_{0,E} \leq & \\ \bar{\alpha}\, C_9 C_{13} ||e||_{1,D_E^{(2)}} + C_9 C_{12} h^{1/2} & ||g - \pi_h g||_{0,E} \\ + C_9 C_{14} h_E ||f - \pi_h f||_{0,D_E^{(2)}} + C_9 C_{14} h_E & ||\pi_h f - L\, \bar{u}_h\, ||_{0,D_E^{(2)}} \end{aligned} \tag{3.52}$$

*Proof.* 1 Set $p_h := \pi_h f$. Observe that $\psi_K|_{\delta K} = 0$, by **Green's formula**

$$a|_K(\bar{u}_h, p_h, \psi_K) = -\int_K \nabla \cdot a \nabla\, \bar{u}_h\, p_h \psi_K\, dx + \int_{\delta K} \underbrace{n_{\delta K}.a \nabla\, \bar{u}_h\, p_h \psi_K}_{=0} d\sigma \tag{3.53}$$

using 3.40,3.42 and 3.43 and taking advantage of 3.6 and 3.53 it follows that

$$\begin{aligned}
||\pi_h f - L\ \bar{u}_h\ ||^2_{0,K} &\leq \int_K (\pi_h f - L\ \bar{u}_h)\pi_h\psi_K dx \\
&= C_8(\int_K f\pi_h\psi_K\ dx - a|_K(\bar{u}_h\ \pi_h\psi_K) + \int_K (\pi_h f - f)\pi_h\psi_K) \\
&= C_8\big(a|_K(e, \pi_h\psi_K) + \int_K (\pi_h f - f)\pi_h\psi_K dx\big) \\
&\leq C_8 C_{10}\ \bar{\alpha}\ h^{-1}||e||_{1,K}||p_h||_{0,K}\ + C_8 C_{11}||\pi_h f - f||_{0,K}||p_h||_{0,K} \\
&\leq \bar{\alpha}\ C_8\ C_{10}||e||_{1,K} + C_8\ C_{11} h_K||f - \pi_h f||_{0,K}
\end{aligned}$$

Which proofs equation 3.50 □

*Proof.* 2 Set $p_h^E := [n_E.a\nabla\ \bar{u}_h]_J$. From $\psi_E|_{E'} = 0, E' \neq E$, **Green's formula** yields

$$\int_{\partial D_E^{(2)}} n_{\partial D_E^{(2)}}.a\nabla\ \bar{u}_h\ p_h^E\psi_E d\sigma\ =\ a|_{D_E^{(2)}}(\bar{u}_h, p_h^E, \psi_E) + \int_{D_E^{(2)}} \underbrace{\nabla\cdot.a\nabla\ \bar{u}_h}_{=-L\bar{u}_h}\ p_h^E\psi_K dx \qquad (3.54)$$

If we use 3.41, 3.47 and 3.48 and observe 3.6, 3.54 it follows that

$$\begin{aligned}
|||[n_{\partial D_E^{(2)}}.a\nabla\ \bar{u}_h]_J||^2_{0,E} &\leq C_9 \int_E [n_{\partial D_E^{(2)}}.a\nabla\ \bar{u}_h]_J p_h^E\psi_E d\sigma \\
&= \int_{\partial D_E^{(2)}} [n_{\partial D_E^{(2)}}.a\nabla\ \bar{u}_h]_J p_h^E\psi_E d\sigma \\
&= C_9\big(a|_{D_E^{(2)}}(\bar{u}_h, p_h^E, \psi_E) - \int_{D_E^{(2)}} f p_h^E\psi_E dx \\
&+ \int_{D_E^{(2)}} (f - \pi_h f)p^E\psi_E\ dx\ +\ \int_{D_E^{(2)}} (\pi_h f - L\ \bar{u}_h)p^E\psi_E\ dx\big) \\
&= -C_9 a|_{D_E^{(2)}}(e, p_h^E\psi_E) + C_9\big(\int_{D_E^{(2)}} (f - \pi_h f)p_h^E\psi_E dx\ +\ \int_{D_E^{(2)}} (\pi_h f - L\ \bar{u}_h)p^E\psi_E\ dx\big) \\
&\leq C_9 C_{13}\ \bar{\alpha}\ h_E^{-1/2}||e||_{1,D_E^{(2)}}||p_h^E||_{0,E} \\
&+ C_9 C_{13} h_E^{1/2}||f - \pi_h f||_{0,D_E^{(2)}}||p_h^E||_{0,E} + C_9 C_{14} h_E^{1/2}||\pi_h - L\ \bar{u}_h\ ||_{0,D_E^{(2)}}||p_h^E||_{0,E}
\end{aligned}$$

□

from which we readily deduce 3.51

*Proof.* 3 Set $p_h^E := \pi_h g - n_E.a\nabla\ \bar{u}_h$ observing $\psi_E|_{E'} = 0, E' \neq E$, **Green's formula** yields

$$\begin{aligned}\int_E n_E.a\nabla\ \bar{u}_h\ \psi_E\ d\sigma \\ &= \int_{\partial D_E^{(2)}} n_{\partial D_E^{(2)}}.a\nabla\ \bar{u}_h\ \psi_E\ d\sigma \\ &= a|_{D_E^{(2)}}(\bar{u}_h, p_h^E, \psi_E) + \int_{D_E^{(2)}} \underbrace{\nabla \cdot a\nabla\ \bar{u}_h}_{=-L\bar{u}_h} p_h^E \psi_E dx\end{aligned} \tag{3.55}$$

Now, using 3.41, 3.44, 3.47 and 3.48 and taking advantage of 3.6,3.55 we get

$$\begin{aligned}||\pi_h g - n_E.a\nabla\ \bar{u}\ ||_{0,E} \\ &\leq C_9 \int_E (\pi_h g - n_E.a\ \nabla\ \bar{u}\ |) p_h^E \psi_E d\sigma \\ &= C_9\Big( \int_{D_E^{(2)}} f\ p_h^E\ \psi_E\ dx + \int_E g\ p_h^E \psi_E\ d\sigma \\ &- a|_{D_E^{(2)}}(\bar{u}_h, p_h^E, \psi_E) + \int_{D_E^{(2)}} (\pi_h f\ - f)\ p_h^E\ \psi_E\ dx \\ &+ \int_E (\pi_h\ g - \ g)\ f\ p_h^E\ \psi_E d\sigma - \int_{D_E^{(2)}} (\pi_h\ f - \ L\ \bar{u}_h) dx\Big) \\ &= C_9 a|_{D_E^{(2)}}(e, p_h^E, \psi_E) + C_9\Big( \int_{D_E^{(2)}} (\pi_h f\ - f)\ p_h^E\ \psi_E\ dx \\ &+ \int_E (\pi_h\ g - \ g)\ f\ p_h^E\ \psi_E d\sigma - \int_{D_E^{(2)}} (\pi_h\ f - \ L\ \bar{u}_h) dx\Big) \\ &C_9 C_{13}\ \bar{\alpha}\ h_E^{-1/2} ||e||_{1,D_E^{(2)}} |||p_h^E||_{0,E} \\ &+ C_9 C_{14} h_E^{1/2} ||\pi_h f - f||_{0,D_E^{(2)}}\ ||p_h^E||_{0,E} \\ &+ C_9 C_{12} ||\pi_h g - g||_{0,D_E^{(2)}}\ ||p_h^E||_{0,E} \\ &+ C_9 C_{14} h_E^{1/2} ||\pi_h f - L\ \bar{u}_h\ ||_{0,D_E^{(2)}} ||p_h^E||_{0,E}\end{aligned}$$

□

## 3.3 Gradient Recovery Estimator

Gradient recovery type a posteriori error estimate was introduced by Zienkiewicz and Zhu. On the basis of superconvergence analysis, it has been proved that for the piecewise linear

finite element approximation of linear elliptic equation, when the partition is uniform and the solution is smooth enough,

$$\eta = |u - u_h|_{1,\Omega} + o(h) \tag{3.56}$$

where $\eta$ is the gradient recovery type a posteriori error estimate. We derive the gradient recovery type a posteriori error estimate for the finite element approximation of 3.2. The principle results is that:

$$c|u - u_h|_{1,\Omega} - \mathcal{E} \leq \eta \leq c|u - u_h|_{1,\Omega} + \mathcal{E}_* \tag{3.57}$$

**Definition 3.3.1.** *In order to construct a posteriori error estimate, we need to define a gradient recovery operator $gv_h$ on $V_h$ which satisfies the following*

$$gv_h = \sum_{z \in \partial^2 \mathcal{T}_h} gv_h(z)\phi_z, \quad gv_h(z) = \sum_{j=1}^{J_z} \alpha_z^j (\nabla u_h)_{K_z^j} \quad \forall v_h \in V_h$$

### 3.3.1 Upper Bound for the a Posteriori Error Estimates

We give a simple proof of the upper bound of a posteriori error estimates with reference to the gradient recovery, which shares similar properties with residual type a posteriori estimates.

**Theorem 3.3.2.** *Let $u$ and $u_h$ be the solutions of (3.2) and (3.3), respectively. Assume that $f \in L^2(\Omega), a \in L^\infty(\Omega) \cap H^1(\Omega)$, then*

$$|u - u_h|_{1,\Omega}^2 \ \leq \ C\eta^2 + C\mathcal{E}^2 \tag{3.58}$$

*where*

$$\eta^2 = \sum_K \eta_K^2 = \sum_K ||gu_h - \nabla u_h||_{0,k}^2,$$
$$\mathcal{E}^2 = \sum_{z \in \Lambda} \int_{\omega_z} h_z^2 |f - \bar{f}_z|^2 + \sum_{z \in \Lambda} \int_{\omega_z} h_z^2 |(\nabla a - \overline{\nabla a}_z)\nabla u_h|^2$$

*Proof.* Let $e = u - u_h, e_I = \pi e \in V_h$ be the weighted Clément-type interpolation of $e$. It follows from 3.3 and 3.25 that

$$c|u-u_h|^2_{1,\Omega} \leq \int_\Omega a\nabla(u-u_h)\nabla_e = \int_\Omega a\nabla(u-u_h)\nabla(e-e_I)$$

and

$$\int_\Omega a\nabla_u\nabla(e-e_I) = \int_\Omega f(e-e_I).$$

Let $G$ be the recovery operator as defined, then

$$-\int_\Omega a\ g\ u_h\nabla(e-e_i) = \int_\Omega \boldsymbol{\nabla}\cdot(a\ g\ u_h)(e-e_I).$$

Hence,

$$c|u-u_h|^2_{1,\Omega} \leq \int_\Omega f(e-e_I) + \int_\Omega \boldsymbol{\nabla}\cdot(a\ g\ u_h)(e-e_I) + \int_\Omega a(g\ u_h - \nabla u_h)\nabla(e-e_I) \tag{3.59}$$

$$= I_1 + I_2 + I_3 \tag{3.60}$$

It follows from properties 3.25,3.26, 3.27 and 3.28 that

$$I_1 \leq C|e|_{1,\Omega}\Big(\sum_{z\in\Lambda}\int_\Omega h_z^2|f-\overline{f}_z|^2\Big)^{\frac{1}{2}} \leq C\mathcal{E}^2 + \frac{c}{6}|e|^2_{1,\Omega} \tag{3.61}$$

Again,

$$I_2 = \int_\Omega \boldsymbol{\nabla}\cdot(a\ g\ u_h)(e-e_I) \leq \sum_{z\in\Lambda}\int_{\omega_z}\Big|\boldsymbol{\nabla}\cdot(a\ g\ u_h) - \overline{\nabla a}_z g\ u_h(z)\Big||e-e_z\psi|$$

and

$$\sum_{z\in\Lambda}\int_{\omega_z} h_z^{-2}|e-e_z\psi|^2 \leq C|e|^2_{1,\Omega}.$$

Then we have

$$I_2 \leq \ \frac{c}{6}|e|^2_{1,\Omega} + C\sum_{z\in\Lambda}\sum_{K\subset\omega_z} h_z^2\int_K\Big|\boldsymbol{\nabla}\cdot(a\ g\ u_h) - \overline{\nabla a}_h\ g\ u_h(z)\Big|^2.$$

On any element $K$, $\boldsymbol{\nabla}\cdot(a\nabla u_h) = \nabla a \nabla u_h$, hence we can deduce the following,

$$\Big|\boldsymbol{\nabla}\cdot(a\ g\ u_h) - \overline{\nabla a}_h\ g\ u_h(z)\Big| \\ \leq \Big|\boldsymbol{\nabla}\cdot(a\ g\ u_h - a\nabla u_h)\Big| + \Big|(\nabla a - \overline{\nabla a}_z)\nabla u_h\Big| + \Big|\overline{\nabla a}_z\Big(\nabla u_h - g\ u_h(z)\Big)\Big|$$

and

$$\sum_{z\in\Lambda}\sum_{K\subset\omega_z} h_z^2 \int_K \Big|(\nabla a - \overline{\nabla a}_z)\nabla u_h\Big|^2 = \sum_{z\in\Lambda}\int_{\omega_z}\Big|(\nabla a - \overline{\nabla a}_z)\nabla u_h\Big|^2 \leq \mathcal{E}^2.$$

Now, since $g\ u_h, \nabla u_h$ are piecewise polynomials on the element $K$, it follows from inverse inequality property that,

$$\sum_{z\in\Lambda}\sum_{K\subset\omega_z}\int_K h_z^2\Big|\boldsymbol{\nabla}\cdot(a\ g\ u_h - a\nabla u_h)\Big|^2 \\ \leq C\sum_{z\in\Lambda}\sum_{K\subset\omega_z} h_z^2||a||^2_{0,\infty,K}||g\ u_h - \nabla u_h||^2_{1,K} \\ + \leq C\sum_{z\in\Lambda}\sum_{K\subset\omega_z} h_z^2||a||^2_{1,K}||g\ u_h - \nabla u_h||^2_{1,\infty,K} \\ \leq \sum_K |g\ u_h - \nabla u_h||^2_{0,K} \leq C|g\ u_h - \nabla u_h||^2_{0,\Omega} = C\eta^2,$$

and

$$\sum_{z\in\Lambda}\sum_{K\subset\omega_z}\int_K h_z^2\Big|\overline{\nabla a}_z\Big(\nabla u_h - Gu_h(z)\Big)\Big|^2 \\ \leq \sum_{z\in\Lambda}\sum_{K\subset\omega_z} h_z^2\Big||\overline{\nabla a}_h\Big|^2_{0,K}||g\ u_h - \nabla u_h||^2_{0,\infty,K} \\ \leq \sum_K ||g\ u_h - \nabla u_h||^2_{0,K} \leq C||g\ u_h - \nabla u_h||^2_{0,\Omega} = C\eta^2$$

Then we have

$$I_2 \leq C\eta^2 + C\mathcal{E}^2 + \frac{c}{6}|e|^2_{1,\Omega} \tag{3.62}$$

By Schwarz inequality,

$$I_3 \leq C||g\ u_h - \nabla u_h||_{0,\Omega}|e|_{1,\Omega} \leq C\eta^2 + \frac{c}{6}|e|^2_{1,\Omega} \tag{3.63}$$

Which completes the proof. □

### 3.3.2 Lower Bound for the a Posteriori Error Estimates

In this section, a brief presentation of the lower bound for a posteriori error estimates in reference to the gradient recovery and its prrof is given.

**Theorem 3.3.3.** *Let $u$ and $u_h$ be the solutions of (3.2) and (3.3), respectively. Assume that $f \in L^2(\Omega), a \in L^\infty(\Omega) \cap H^1(\Omega)$, then*

$$\eta^2 \leq C|u - u_h|^2_{1,\Omega} + C\mathcal{E}^2_*, \tag{3.64}$$

*where $\eta$ is defined in theorem 3.3.2,*

$$\mathcal{E}^2_* = \sum_K h_z^2 \int_K |f - \bar{f}_z|^2 + \sum_k \int_K h_z^2 |(\nabla a - \overline{\nabla a}_z)\nabla u_h|^2$$

*, where*

*Proof.* $\forall K \in \mathcal{T}_h$, let $S_K = \bigcup_{\overline{K'} \cap \overline{K} \neq \emptyset}$ Using the definition of $g\ u_h(z)$, then on the element $K$,

$$\begin{aligned}
|\nabla u_h - g\ u_h|^2 &= \left| (\nabla u_h)_K - \sum_{z \cap \overline{K} \neq \emptyset} \phi_z \left( \sum_{j=1}^{J_z} \alpha_z^j (\nabla u_h)_{K_z^j} \right) \right|^2 \\
&= \left| \sum_{z \cap \overline{K} \neq \emptyset} \phi_z \left( \sum_{j=1}^{J_z} \alpha_z^j ((\nabla u_h)_K - (\nabla u_h)_{K_z^j}) \right) \right|^2 \\
&\leq C \sum_{K' \subset S_K} \left| (\nabla u_h)_K - (\nabla u_h)_{K'} \right|^2.
\end{aligned}$$

Now, taking into an account, the edge elements $E \in \mathcal{E}_h(\Omega)$, the unit outward normal vector $n$ and the jump term across the common edges $[V_h]_j$, we can deduce following. Note also that $\forall K, K^{'} \subset S_K$, there exist a finite positive integer $m_k$, which is independent of $h$, and elements $K_i \subset S_K, i = 0, 1, ..., m_k$, such that $\overline{K}_{i-1} \cap \overline{K}_i = E_i$, where $E_i \subset S_K$ are edges of elements, $E_i \cap \partial S_K = \emptyset$, and $K = K_0, K^{'} = K_{mk}$. Hence,

$$\left| (\nabla u_h)_K - (\nabla u_h)_{K'} \right| = \left| \sum_{i=1}^{m_K} [\nabla u_h]_{E_i} \right| \leq \left| \sum_{i=1}^{m_K} [\nabla u_h]_{E_i} \right| \leq \sum_{E \subset S_K \setminus \partial S_K} [\nabla u_h]_E.$$

Since $u_h$ is continous on $\Omega$, then $[\frac{a(\partial u_h)}{\partial t}]_E = 0$ if $E \cap \partial\Omega = \emptyset$, where $t$ is the tangent direction of the edge $E$. Hence

$$\Big|[\nabla u_h]_E\Big| \leq C\Big|[a\nabla u_h]_E\Big| = C\left|\left[a\frac{\partial u_h}{\partial n}\right]_E\right| = C\Big|[a\nabla u_h \cdot n]_E\Big|,$$

if $E \cap \partial\Omega = \emptyset$, then

$$\eta^2 = ||g\ u_h - \nabla u_h||^2_{0,\Omega} \leq C\sum_K h_E^2 \sum_{K\subset S_K} \Big|[a\nabla u_h \cdot n]_E\Big|^2 \leq \sum_{E\cap\Omega=\emptyset} h_E \int_E [a\nabla u_h \cdot n]^2_E. \quad (3.65)$$

$$\sum_{E\cap\Omega=\emptyset} h_E \int_E [a\nabla u_h \cdot n]^2_E \leq C|u - u_h|^2_{1,\Omega} + C\sum_K h_E^2 \int_K \big|f + \boldsymbol{\nabla} \cdot (a\nabla u_h)\big|^2, \quad (3.66)$$

and

$$\sum_K h_E^2 \int_K \big|f + \boldsymbol{\nabla} \cdot (a\nabla u_h)\big|^2 \quad (3.67)$$

$$\leq C|u - u_h|^2_{1,\Omega} + C\sum_K h_E^2 \int_K \big|f - \overline{f}\big|^2 + \sum_K h_E^2 \int_K |(\nabla a - \overline{\nabla a}_z)\nabla u_h|^2. \quad (3.68)$$

Therefore from 3.65 - 3.67, we have

$$\eta^2 \leq C|u - u_h|^2_{1,\Omega} + C\mathcal{E}_*^2, \quad (3.69)$$

which completes the proof. □

This chapter was referenced from [16] [12]

# Chapter 4

# Markov Decision Process

In this chapter we introduce the formal problem of Markov decision processes, or MDPs, which we will seek solution to. This problem involves evaluative feedback, and also an associative aspect—choosing different actions in different situations. MDPs are classical formalization of sequential decision making, where actions influence not just immediate rewards, but also subsequent situations, or states, and through those future rewards. Thus MDPs involve delayed reward and the need to tradeoff immediate and delayed reward. In Bandit problems we estimate the value $q_*(a)$ of each action $a$, but in MDPs our focus is to estimate the value $q_*(s, a)$ of each action $a$ in each state $s$, or we estimate the value $v_*(s)$ of each state given optimal action selections. These state-dependent quantities are essential to accurately assigning credit for long-term consequences to individual action selections. MDPs are a mathematically idealized form of the reinforcement learning problem for which precise theoretical statements can be made. We introduce key elements of the problem's mathematical structure, such as returns, value functions, and Bellman equations [29].

## 4.1 The Agent-Environment Interface

MDPs are intended to be a outlining of the problem of learning from interaction to achieve a goal. The learner and decision maker is called the agent. The thing it interacts with, comprising everything outside the agent, is called the environment. These interact continually, the agent selecting actions and the environment responding to these actions and presenting new situations to the agent. The environment also gives rise to rewards, special numerical values that the agent seeks to maximize over time through its choice of actions

[30].

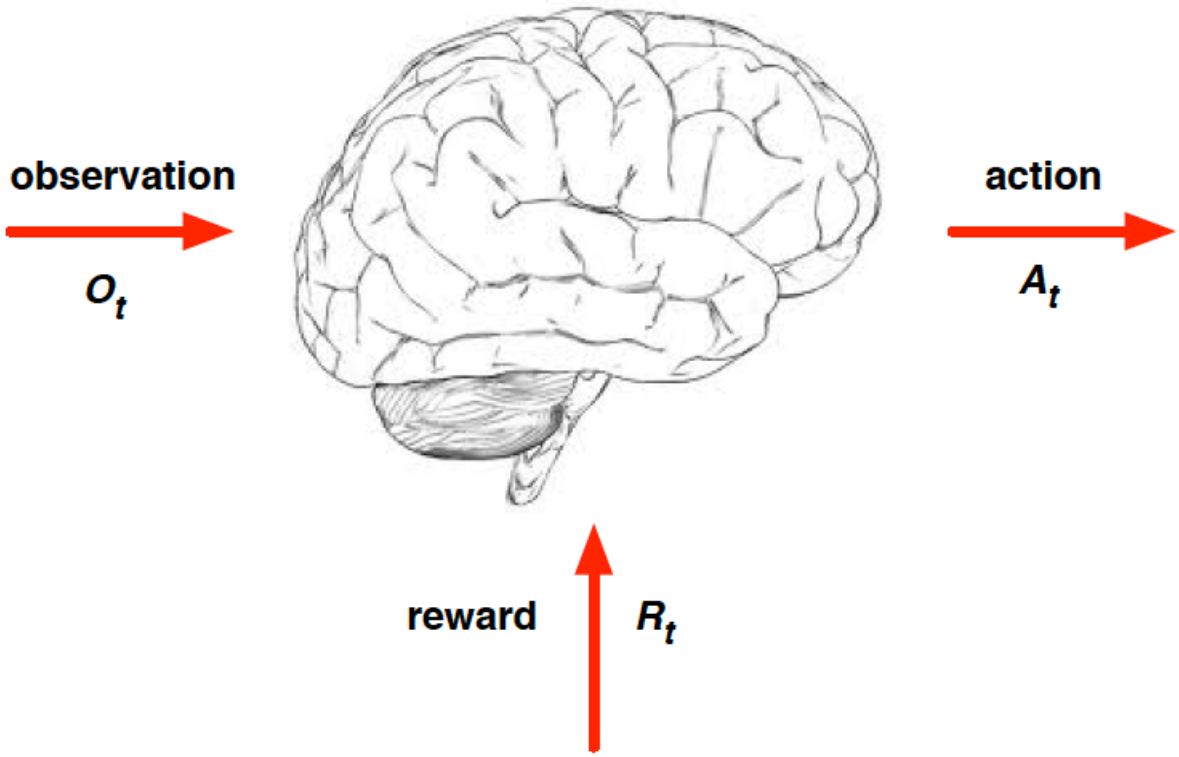



Figure 4.1: Agent and Environment

At each step $t$, the agent

- Executes action $A_t$
- Receives observation $O_t$
- Receives a scalar rewrad $R_t$

The environment

- Receives action $A_t$

- Produce observation $O_{t+1}$
- Produce scalar reward $R_{t+1}$

at every incremental step of $t$

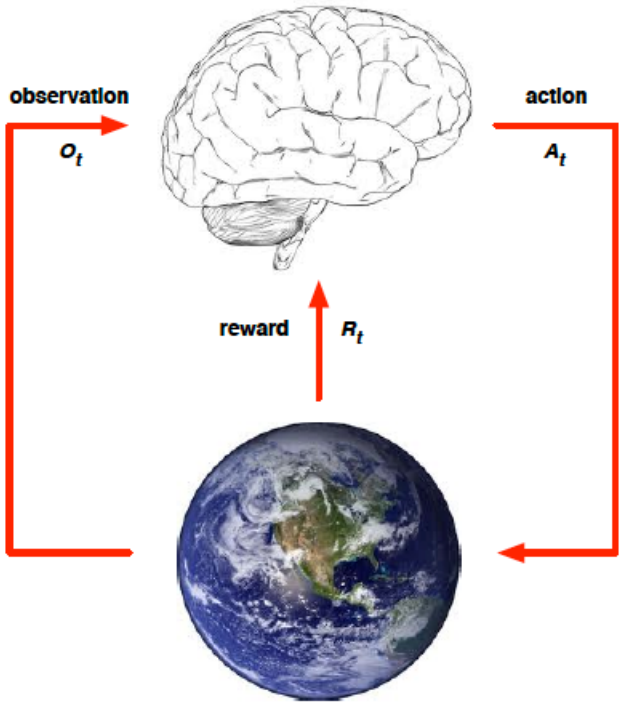



Figure 4.2: Agent and Environment

### 4.1.1 History and State

The **history** is the sequence of observations, actions and rewards. i.e all observable variables up to time $t$. $H_t = O_1, R_1, A_1, ..., A_{t-1}, O_t, R_t$. What happens next depends on the history; the agent select actions and the environment select observations/rewards. **State** is the information used to determine what happens next. Formally speaking, the state is a function of the history: $S_t = f(H_t)$

**Environment State**: The environment state $S_t^e$ is the environment's private representation, i.e whatever data the environments uses to pick the next observation/reward. The environments state is usually not visible to the agent. Even if $S_t^e$ is visible, it may contain irrelevant information.

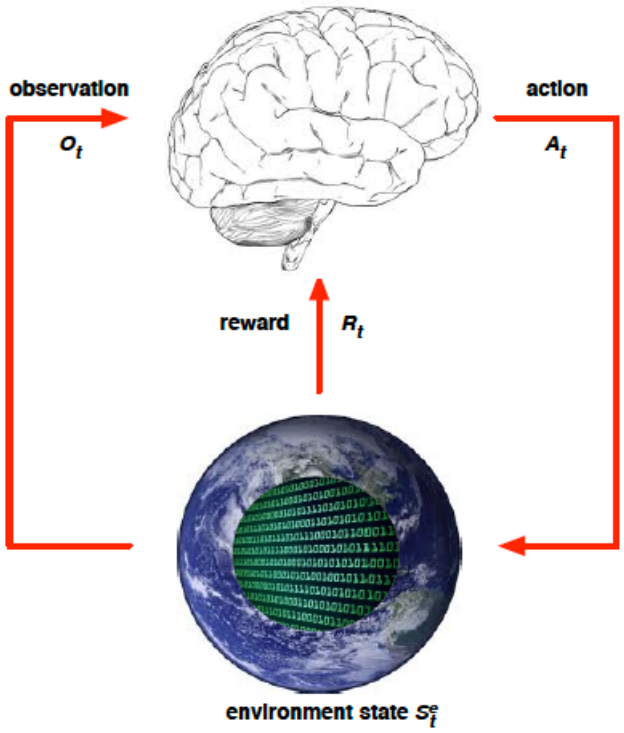



Figure 4.3: Environment State

**Agent State:** The agent state $S_t^a$ is the agent's internal representation, i.e whatever information the agent uses to pick the next action. This is the information used by the reinforcement learning algorithms. It can be any function of history $S_t^a = f(H_t)$

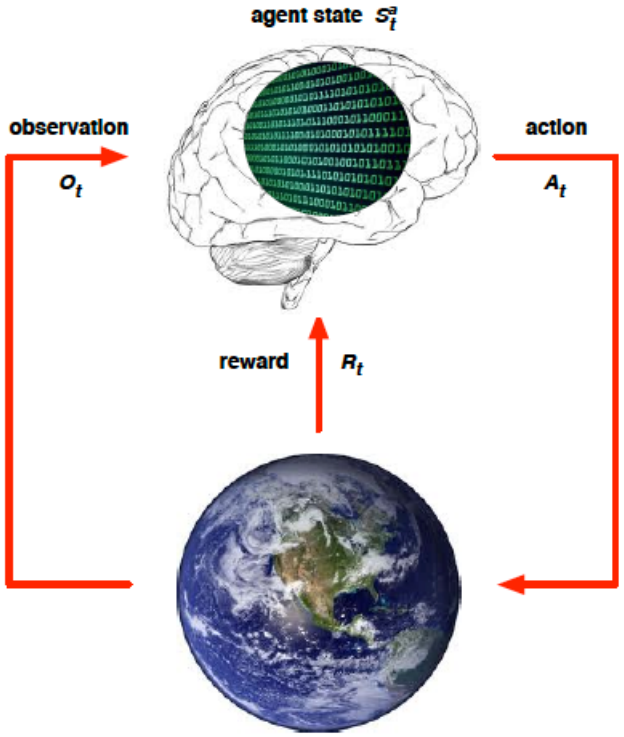



Figure 4.4: Agent State

In general, a Markov decision process of a reinforcement learning agent can be represented in a pictorial form as:

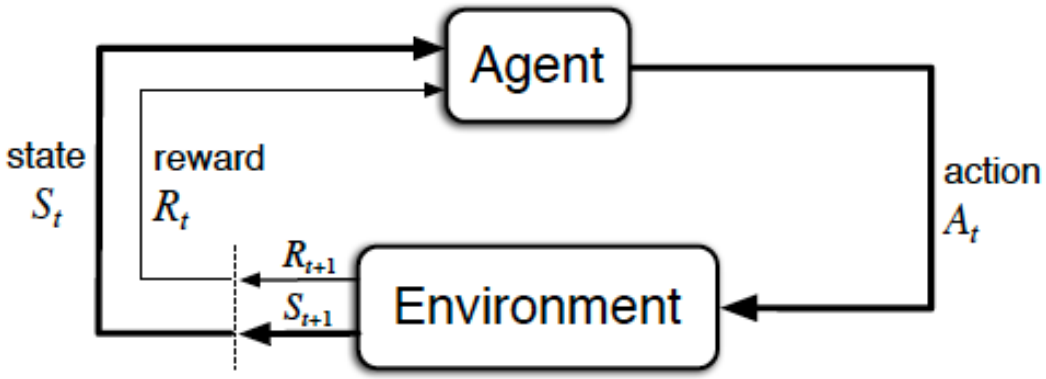



Figure 4.5: The agent-environment interaction in a Markov decision process

## 4.2 Markov Process

Markov decision processes formally describe an environment for reinforcement learning. Where the environment is fully observable, i.e the current state completely characterises the process. Most reinforcement learning problems and infact almost all can be formalised as Markov Decision Processes(MDPs) e.g

- Optimal control primarily deals with continous MDPs.
- Partially observable problems can be converted to MDPs.
- Bandits are MDPs with one state.

### 4.2.1 Markov Property

In a Markov decision process, the probabilities given by $p$ completely characterize the environment's dynamics. That is, the probability of each possible value for $S_t$ and $R_t$ depends only on the immediately preceding state and action, $S_{t-1}$ and $A_{t-1}$, and not at

all on the earlier states and actions. This is best viewed as restriction not on the decision process, but on the state. The state must include information about all aspects of the past agent–environment interaction that make a difference for the future. If it does, then the state is said to have the *Markov property*, which states that:
*"The future is independent of the past given the present"*

**Definition 4.2.1.** *A state $S_t$ is Markov if and only if*

$$\mathbb{P}\left[S_{t+1}|S_t\right] = \mathbb{P}\left[S_{t+1}|S_1, S_2, ...S_t\right] \tag{4.1}$$

*The state captures all relevant information from the history, therefore the state is a sufficient statistic of the future, i.e once the state is known, the history may be thrown away.*

### 4.2.2 State Transition Matrix

For a Markov state $s$ and successor state $s^{'}$, the state transition probability is defined by

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s^{'}|S_t = s\right] \tag{4.2}$$

State transition matrix $\mathcal{P}$ defines transition probabilities from all states s to all successor states $s^{'}$, in essence, the state probabilities $T(s, a, s^{'})$ specify the probability or chance of ending up in state $s^{'}$ if taken an action $a$ in state $s$

$$\mathcal{P} = from \begin{matrix} \\ \\ \\ \\ \\ \end{matrix} \overset{to}{\begin{bmatrix} \mathcal{P}_{11} & ... & \mathcal{P}_{1n} \\ & . & \\ & . & \\ & . & \\ \mathcal{P}_{n1} & ... & \mathcal{P}_{nn} \end{bmatrix}} \tag{4.3}$$

where summation of each row of the matrix is one(1), that is for each state $s$ and action $a$

$$\sum_{s^{'} \in states} T(s, a, s^{'}) = 1 \tag{4.4}$$

such that $T(s, a, s') > 0$

Rows indicate the current state and column indicate the transition.

A Markov process is memoryless random process. i.e a sequence of random states $S_1, S_2, ...$ with the Markov property.

**Definition 4.2.2.** *A Markov Process(or Markov Chain) is a tuple* $\langle \mathcal{S}, \mathcal{P} \rangle$

- $\mathcal{S}$ *is a finite set of states*
- $\mathcal{P}$ *is a state transition probability matrix, given by 4.2*

### 4.2.3 Example:Student Markov Chain

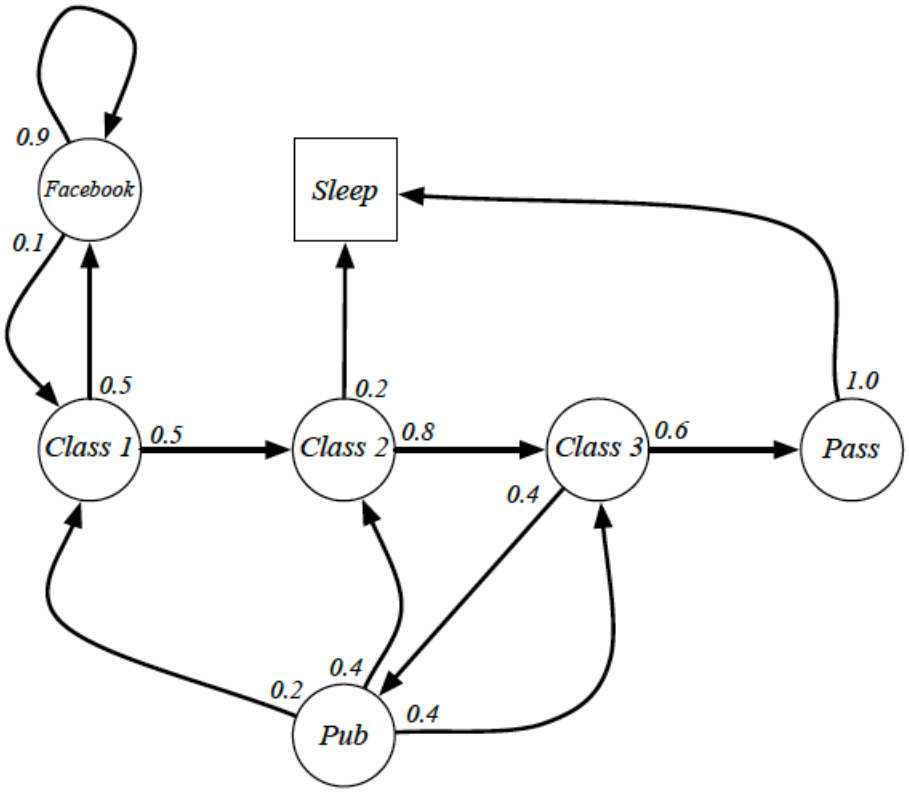



Figure 4.6: Student Markov Chain

From 4.2.3 we can deduce sample episodes starting from $S_1 = C1$. $S_1, S_2, S_T$

- C1 C2 C3 Pass Sleep
- C1 FB FB C1 C2 Sleep
- C1 C2 C3 Pub C2 C3 Pass Sleep
- C1 FB FB C1 C2 C3 Pub C1 FB FB FB C1 C2 C3 Pub C2 Sleep

Student Markov Chain Transition Matrix:

$$P = \begin{array}{c} \\ C_1 \\ C_2 \\ C_3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array} \begin{array}{c} \begin{array}{ccccccc} C_1 & C_2 & C_3 & Pass & Pub & FB & Sleep \end{array} \\ \left( \begin{array}{ccccccc} 0 & 0.5 & 0 & 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0.8 & 0 & 0 & 0 & 0.2 \\ 0 & 0 & 0 & 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \\ 0.2 & 0.4 & 0.4 & 0 & 0 & 0 & 0 \\ 0.1 & 0 & 0 & 0 & 0 & 0.9 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1.0 \end{array} \right) \end{array}$$

## 4.3 Markov Reward Process

In reinforcement learning, the goal of the agent is formalized in terms of a special signal, called the reward, passing from the environment to the agent. A Markov reward process can be defined as a Markov chain with values. At each time step, the reward is a simple number, $R_t \in \Re$. Informally, the agent's goal is to maximize the total amount of reward it receives. This means maximizing not immediate reward, but cumulative reward in the long run.

**Definition 4.3.1.** *A Markov Reward Process is a tuple* $\langle \mathcal{S}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ *is a finite set of states.*
- $\mathcal{P}$ *is a state transition probability matrix, given by 4.2.*
- $\mathcal{R}$ *is a reward function,* $\mathcal{R}_s = \mathbb{E}\left[R_{t+1} \mid S_t = s\right]$.
- $\gamma$ *is a discount factor,* $\gamma \in [0, 1]$.

### 4.3.1 Example: Student Markov Reward Process

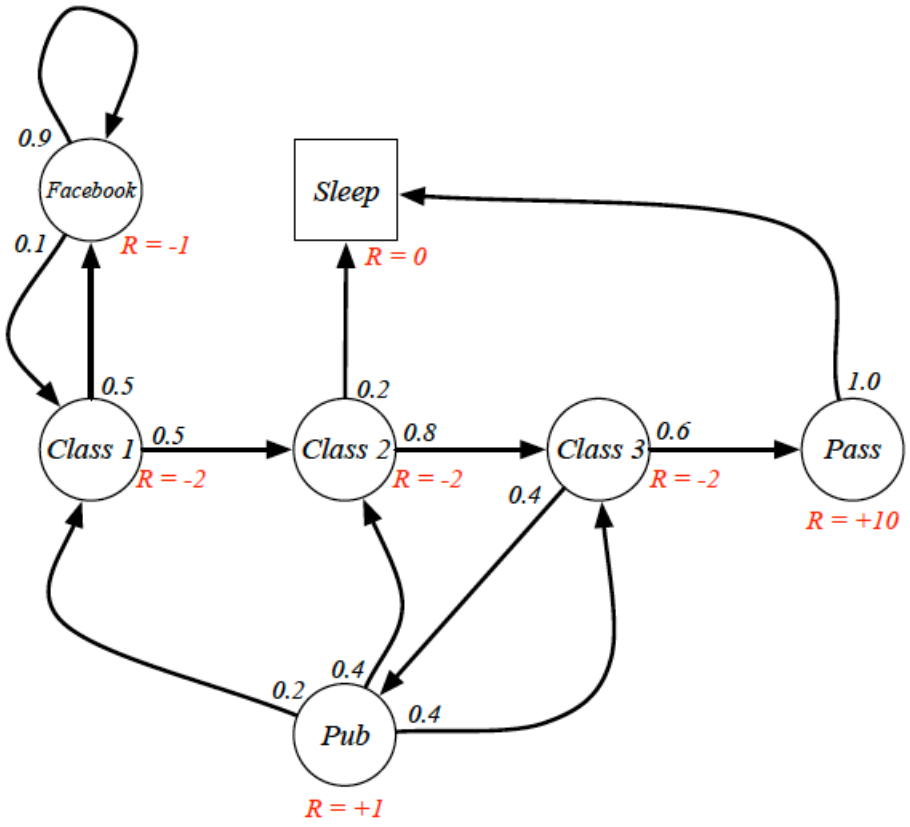



Figure 4.7: Student Markov reward process

**Definition 4.3.2.** ***Return***

*We have stated earlier that the agent's goal is to maximize the cumulative reward it receives in the long run. But the question is how might this be defined formally?. If the sequence of rewards received after time step t is denoted by* $R_{t+1}, R_{t+2}, R_{t+3}, ...,$ *then what precise aspect of this sequence do we wish to maximize?. Generally, we seek to maximize the expected return, where the return, denoted* $G_t$*, is defined as some specific function of the reward sequence. We can therefore define the return* $G_t$ *as the total discounted reward from time-step t.*

$$G_t = R_{t+1} + \gamma R_{t+2} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{4.5}$$

- *The discount* $\gamma \in [0, 1]$ *is the present value of future rewards.*
- *The value of receiving reward* $R$ *after* $k + 1$ *time-steps is* $\gamma^k R$*.*
- *This values immediate reward above delayed reward.*
- $\gamma$ *close to 0 leads to "myopic" evaluation.*
- $\gamma$ *close to 1 leads to "far-sighted" evaluation.*

This approach makes sense in applications in which there is a natural notion of final time step, that is, when the agent–environment interaction breaks naturally into subsequences, which we call episodes. Each episode ends in a special state called the *terminal state,* followed by a reset to a standard starting state or to a sample from a standard distribution of starting states.

Although the return (4.5) is a sum of an infinite number of terms, it is still finite if the reward is nonzero and constant if $\gamma < 1$. For example, if the reward is a constant +1, then the return is:

$$G_t = \sum_{k=0}^{\infty} \gamma^k = \frac{1}{1 - \gamma}. \tag{4.6}$$

### 4.3.2 Why Discounted

Most Markov reward and decision processes are discounted. Some of the possible reasons are listed below;

- Mathematically convenient to discount rewards.
- Avoids infinite returns in cyclic Markov processes.
- Uncertainty about the future may not be fully represented.
- If the reward is financial, immediate rewards may earn more interest than delayed rewards.
- Animal/human behaviour shows preference for immediate reward.
- It is sometimes possible to use undiscounted Markov reward processes (i.e $\gamma = 1$), e.g if all sequences terminate.

**Definition 4.3.3.** ***Value Function***

*The value function $v(s)$ gives the long-term value of state $s$. The state value function $v(s)$ of a Markov Reward Processes(MRP) is the expected return starting from state $s$*

$$v(s) = \mathbb{E}\left[G_t \mid S_t = s\right] \tag{4.7}$$

### 4.3.3 Bellman Equation for MRPs

The value function can be decomposed into two parts

- immediate reward $R_{t+1}$
- discounted value of successor state $\gamma v(S_{t+1})$

$$
\begin{aligned}
v(s) &= \mathbb{E}\left[G_t \mid S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \ \gamma R_{t+2} \ + \gamma^2 R_{t+3} \ + ... | \ S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} \ + \ \gamma(R_{t+2} \ + \gamma R_{t+3} \ + ...) | \ S_t = s\right] \\
&= \mathbb{E}\left[R_{t+1} + \gamma G_{t+1} \ | S_t \ = s\right] \\
v(s) &= \mathbb{E}\left[R_{t+1} + \gamma v(S_{t+1}) \ | S_t \ = s\right]
\end{aligned}
\tag{4.8}
$$

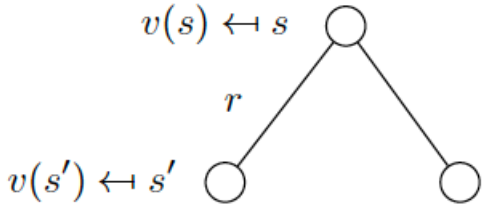



Figure 4.8: Backup diagram for MRPs

$$
v(s) = \mathcal{R}_s + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'} V(S') \tag{4.9}
$$

### 4.3.4 Bellman Equation in Matrix Form

The Bellman equation can be expressed concisely using matrices,

$$
v = \mathcal{R} + \gamma \mathcal{P} v \tag{4.10}
$$

where $v$ is a column vector with one entry per state.

$$
\begin{bmatrix} v(1) \\ . \\ . \\ . \\ v(2) \end{bmatrix} = \begin{bmatrix} \mathcal{R}_1 \\ . \\ . \\ . \\ \mathcal{R}_n \end{bmatrix} + \gamma \begin{bmatrix} \mathcal{P}_{11} & ... & \mathcal{P}_{1n} \\ & . & \\ & . & \\ & . & \\ \mathcal{P}_{n1} & ... & \mathcal{P}_{nn} \end{bmatrix} \begin{bmatrix} v(1) \\ . \\ . \\ . \\ v(n) \end{bmatrix} \tag{4.11}
$$

### 4.3.5 Solving the Bellman Equation

The Bellman equation is a linear equation, therefore it can be solved directly:

$$\begin{aligned} v &= \mathcal{R} + \gamma \mathcal{P} v \\ (I - \gamma \mathcal{P}) v &= \mathcal{R} \\ v &= (I - \gamma \mathcal{P})^{-1} \mathcal{R} \end{aligned} \tag{4.12}$$

But the computational complexity is $O(n^3)$ for $n$ states, therefore direct solution is only possible for small MRPs.
There are many iterative methods for large MRPs, e.g

- Dynamic programming.
- Monte-Carlo evaluation.
- Temporal- Difference learning

## 4.4 Markov Decision Process

**Definition 4.4.1.** *A Markov decision process(MDP) is a Markov reward process with decisions. It is an environment in which all states are Markov.*
*A Markov Decision Process is a tuple* $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$

- $\mathcal{S}$ *is a finite set of states.*
- $\mathcal{A}$ *is a finite set of actions.*
- $\mathcal{P}$ *is a state transition probability matrix*

$$\mathcal{P}^{a}_{ss'} = \mathbb{P}\left[S_{t+1} = s' | S_t = s, A_t = a\right] \tag{4.13}$$

- $\mathcal{R}$ *is reward function,*

$$\mathcal{R}_s^a = \mathbb{E}\left[R_{t+1} = s^{'}|S_t = s, A_t = a\right] \tag{4.14}$$

- $\gamma$ *is a discount factor,* $\gamma \in [0, 1]$.

### 4.4.1 Policies and Value Functions

Almost all reinforcement learning algorithms involve estimating value functions—functions of states (or of state–action pairs) that estimate how good it is for the agent to be in a given state (or how good it is to perform a given action in a given state). The notion of "how good" here is defined in terms of future rewards that can be expected, or, to be precise, in terms of expected return. Of course the rewards the agent can expect to receive in the future depend on what actions it will take. Accordingly, value functions are defined with respect to particular ways of acting, called policies.

**Definition 4.4.2.** ***Policies***

*A policy* $\pi$ *is a distribution over actions given states, Formally, a policy is a mapping from states to probabilities of selecting each possible action. That is,* $\Pi : S \mapsto A(s)$

$$\pi(a|s) = \mathcal{P}\left[A_t = a|S_t = s\right] \tag{4.15}$$

*A policy fully defines the behaviour of an agent. MDP policies depend on the current state not the history, i.e Policies are stationary(time-independent),* $A_t \sim \pi(.|S_t), \forall t > 0$.
*Given an MDP* $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ *and a policy* $\pi$*, the state sequence* $S_1, S_2, ...$ *is a Markov process* $\langle \mathcal{S}, \mathcal{P}^{\pi} \rangle$.
*The state and reward sequence* $S_1, R_2, S_2, ...$ *is a Markov reward process* $\mathcal{M} = \langle \mathcal{S}, \mathcal{P}^{\pi}, \mathcal{R}^{\pi}, \gamma \rangle$ *where*

$$\mathcal{P}_{s,s'}^{\pi} = \sum_{a \in \mathcal{A}} \pi(a|s)\mathcal{P}_{ss'}^{a} \tag{4.16}$$

$$\mathcal{R}_s^{\pi} = \sum_{a \in \mathcal{A}} \pi(a|s)\mathcal{R}_s^{a} \tag{4.17}$$

**Definition 4.4.3.** ***Value Function***

*The* ***state-value function*** $v_\pi(s)$ *of an MDP is the expected return starting from state* $s$*, and then following policy* $\pi$

$$v_\pi(s) = \mathbb{E}_\pi \left[ G_t \mid S_t = s \right] \tag{4.18}$$

*The* ***action-value function*** $q_\pi(s, a)$ *is the expected return starting from stete* $s$*, taking action* $a$*, and then following policy* $\pi$

$$q_\pi(s, a) = \mathbb{E}_\pi \left[ G_t \mid S_t = s, A_t = a \right] \tag{4.19}$$

### 4.4.2 Bellman Expectation Equation for Policy and Value functions

The state-value function can again be decomposed into immediate reward plus discounted value of successor state,

$$v_\pi(s) = \mathbb{E}_\pi \left[ R_{t+1} \ + \ \gamma v_\pi(S_{t+1}) \ \mid S_t \ = s \right] \tag{4.20}$$

The action-value function can similarly be decomposed,

$$q_\pi(s, a) = \mathbb{E}_\pi \left[ R_{t+1} \ + \ \gamma q_\pi(S_{t+1}, A_{t+1}) \ \mid S_t \ = s, A_t = a \right] \tag{4.21}$$

**Definition 4.4.4.** ***Bellman Expectation Equation of V state-value function*** $V^\pi$

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) q_\pi(s, a) \tag{4.22}$$

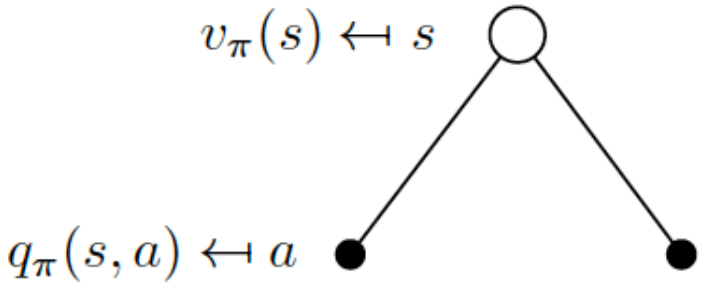



Figure 4.9: Backup diagram for Bellman expectation equation for $V^\pi$

**Definition 4.4.5.** ***Bellman Expectation Equation of Q action-value function*** $Q^\pi$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(S') \tag{4.23}$$

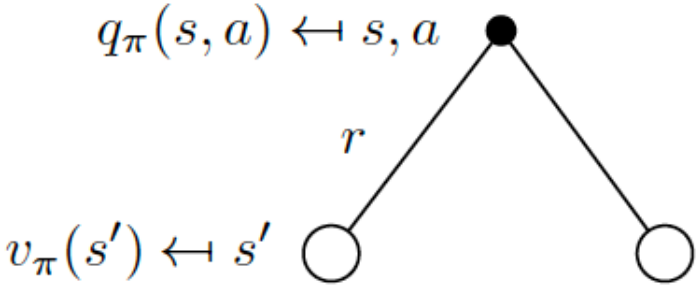



Figure 4.10: Backup diagram for Bellman expectation equation for $Q^\pi$

**Definition 4.4.6.** ***Bellman Expectation Equation for*** $v_\pi$

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a \mid s) \Big( \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_\pi(S') \Big) \tag{4.24}$$

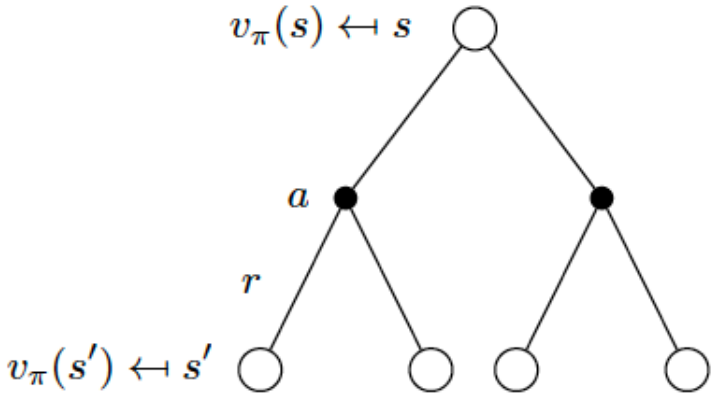



Figure 4.11: Backup diagram for Bellman expectation equation for $v_\pi$

**Definition 4.4.7.** ***Bellman Expectation Equation for*** $q_\pi$

$$q_\pi(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \sum_{a' \in \mathcal{A}} \pi(a' \mid s') q_\pi(s', a') \tag{4.25}$$

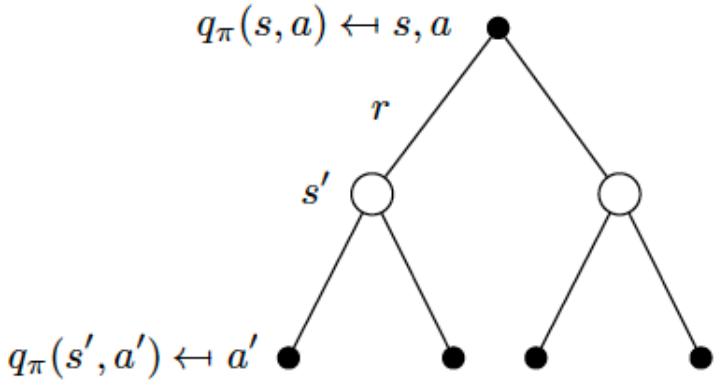



Figure 4.12: Backup diagram for Bellman expectation equation for $q_\pi$

**Definition 4.4.8.** ***Bellman Expectation Equation in Matrix Form***

*The Bellman expectation equation can be expressed concisely using the induced MRP,*

$$v_\pi = \mathcal{R}^\pi + \gamma \mathcal{P}^\pi v_\pi \tag{4.26}$$

*with direct solution*

$$v_\pi = (I - \gamma \mathcal{P}^\pi)^{-1} \mathcal{R}^\pi \tag{4.27}$$

### 4.4.3 Optimal Value Function

**Definition 4.4.9.** *The optimal state-value function $v_*(s)$ id the maximum value function over all policies*

$$v_*(s) = \max_\pi \ v_\pi(s) \tag{4.28}$$

*The optimal action-value function $q_*(s,a)$ is the maximum action-value function over all policies*

$$q_*(s,a) = \max_{\pi} \; q_\pi(s,a) \tag{4.29}$$

*The optimal value function specifies the best possible performance in the MDP. An MDP is "solved" when we know the optimal value function.*

### 4.4.4 Optimal Policy

Define a partial ordering over policies $\pi \geq \pi^{'}$ if $v_\pi(s) \geq v_{\pi'}$, $\forall S$

**Theorem 4.4.10.** *For any Markov Decision Process,*

- *There exists an optimal policy $\pi_*$ that is better than or equal to all other policies, $\pi_* \geq \pi$, $\forall \pi$.*
- *All optimal policies achieve the optimal value function, $v_{\pi*}(s) = v_*(s)$ $\forall s$.*
- *All optimal policies achieve the optimal action-value function, $q_{\pi*}(s,a) = q_*(s,a)$ $\forall s, a$*

## Finding an Optimal Policy

An optimal policy can be found by maximising over $q_*(s,a)$,

$$\pi_*(a \mid s) = \begin{cases} 1 & if \;\; a = \underset{a \in \mathcal{A}}{argmax} \; q_*(s,a) \\ 0 & otherwise \end{cases}$$

There is always a deterministic optimal policy for any MDP. If we know $q_*(s,a)$ we immediately have the optimal policy.

**Definition 4.4.11.** ***Bellman Optimality Equation for*** $V_*$

*The optimal value functions are recursively related by the Bellman optimality equations:*

$$v_*(s) = \max_{a} \; q_*(s,a) \tag{4.30}$$

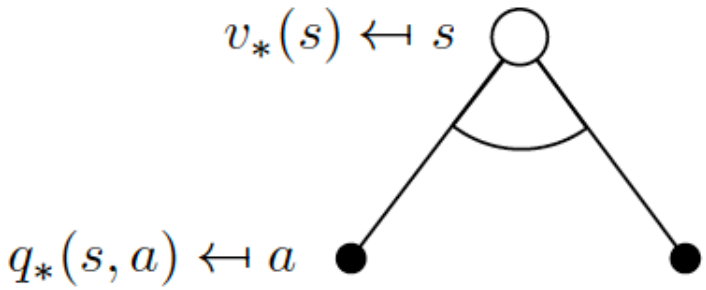



Figure 4.13: Backup diagram for Bellman optimality equation for $V_*$

**Definition 4.4.12.** ***Bellman Optimality Equation for*** $V^*$

$$v_*(s) = \max_a \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(S') \tag{4.31}$$

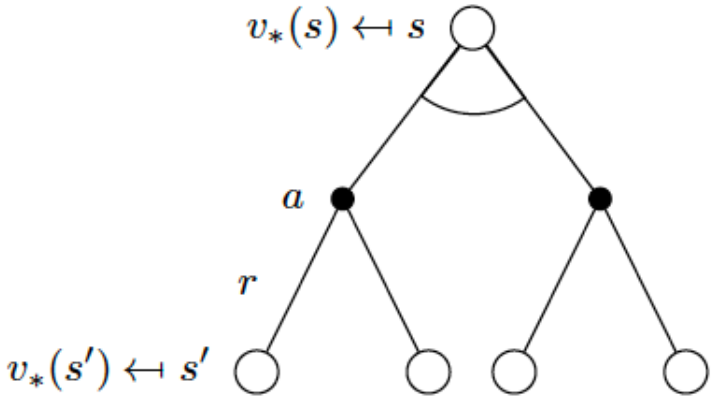



Figure 4.14: Backup diagram for Bellman optimality equation for $V^*$

**Definition 4.4.13.** ***Bellman Optimality Equation for*** $Q^*$

*The Bellman optimality equation states that, for any state-action pair* $(s, a)$ *at the time* $t$*, the expected return from starting in state* $s$*, selecting action* $a$ *and following the optimal policy thereafter will be the expected reward we get from taking action* $a$ *in state* $s$*, which is* $\mathbb{R}_{t+1}$*, plus the maximum expected discounted return that can be achieved from any possible next state-action pair* $(s^{'}, a^{'})$*.*

$$q_*(s,a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a \max_{a'} q_*(s^{'}, a^{'}) \tag{4.32}$$

$$= \mathbb{E}\Big[\mathcal{R}_{t+1} + \gamma \max_{a'} q_*(s', a^{'})\Big] \tag{4.33}$$

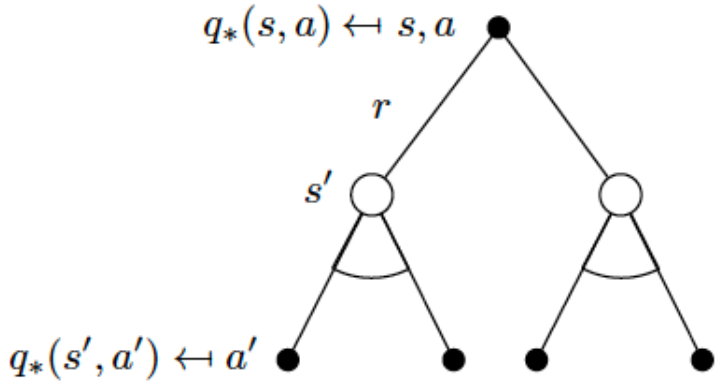



Figure 4.15: Backup diagram for Bellman optimality equation for $Q^*$

The figures in this chapter and the equations were taken from [27].

## 4.5 Learning Algorithm

We start this section by giving a brief history of our reinforcement learning algorithm: Q-learning. A standout amongst other known examples of an effective MDP solving algorithm is Q-learning, which has the great properties of being efficient in both computation and memory and also being able to learn about the optimal policy while following any sufficiently explorative policy. Q-learning was initially introduce in Watkins' Ph.D thesis "Learning from Delayed Rewards", which presented a model of reinforcement learning as incrementally optimizing control of a Markov Decision Process (MDP), and proposed a new algorithm – which was named "Q-learning" – that could on a fundamental level learn optimal control directly without modelling the transition probabilities or expected rewards of the MDP. The first arduous proof of its convergence was done by Watkins and Dayan in 1992. Q-learning is a model -free reinforcement learning algorithm to learn the value of an action in a particular state. It does not need a model of the environment, and it can handle problems with stochastic transitions and rewards without requiring adaptations.

The underlying idea behind this technique is to iteratively update the Q-values for every pair of state and action. The update is based on the Bellman equation (4.33) and is calculated until convergence of the Q-function to the optimal Q-function is achieved. Since the Q-values get updated with each iteration, this iteration technique is called the value iteration.

Below we will describe *how* this algorithm is applied to the problem of picking the right strategy for selecting elements in the AMR algorithm (1) without any prior knowledge about the quality of the solution.

### 4.5.1 The Set Up: Key Elements of RL

- **Environment:** The agent in this environment is the pracitioner who wishes to achieve *accuracy* in the numerical solution computed with *efficiency*.
  *Accuracy* is measured in terms of achieving the optimal convergence rate $\mathcal{O}(h^k)$, $k$ is

the polynomial order of the finite element method.

*Efficiency* refers to achieving the accuracy with the use of minimal number of unknown degrees of freedom (dofs).

It is well known that if the solution is smooth, the expected convergence rate will be achieved and here the "best" choice of degrees of freedom correspond to those arising due to uniform refinement. However, if the solution of the pde is nonsmooth then, the optimal convergence of the solution can only be achieved through adaptive mesh refinement. If the pde solution at hand is nonsmooth due to the presence of a singularity, then adaptive mesh refinement driven by the residual estimator will yield optimally refined meshes and thus, the best dofs. However, if the pde solution is nonsmooth due to a discontinuous diffusion coefficient then, guiding the adaptive mesh refinement using the gradient recovery estimator is a better choice.

- **Challenge:** The challenge in this task arises because the exact solution to practical problems is seldom known *a priori*, hence, in the absence of this knowledge, the agent has to make decisions about the type of mesh refinement strategy to be used. This learning algorithm optimizes this decision.

- **States:** To explain our algorithm, we first assume the state space to be $\{0,\ 1,\ 2\}$ where the numbers represent:

State 0 if the mesh refinement strategy chooses all the cells (triangles or quadrilaterals) i.e., uniform refinement.

State 1 if the mesh refinement strategy selects the cells based on the residual type estimator i.e., adaptive refinement with cell marking based on the residual based estimator.

State 2 if the mesh refinement strategy picks the cells based on the gradient recovery estimator i.e., adaptive refinement with cell marking based on the gradient recovery estimator.

- **Actions:** Analogous to the quantification of the states, we describe the actions taken by the agent as follows.

Action 0 indicates marking all cells for refinement.

Action 1 refers to selecting the cells adaptively based on the residual type estimator.

Action 2 means that the cells are picked adaptively using the gradient recovery estimator.

As an example, if we have a smooth solution, then the optimal set of actions is $\{0,\ 0,\ 0, \cdots\}$ to achieve the optimal convergence rate. We note that the optimal convergence rate for a finite element method using polynomial degree $k$ is $k$.

- **Rewards:** The consequence of each action is a numerical reward which can be thought of as a function $f(\cdot, \cdot)$ defined on the state-action pair as follows:

$$f(S_t, A_t) = R_{t+1} \in \{-1,\ 1\}$$

with

$$R_{t+1} = 1,\ \text{if CR} - \sigma k \geq 0,\ \text{otherwise } -1.$$

Here,

CR: Convergence Rate for the FEM.

$k$: The optimal convergence rate given by the polynomial degree $k$.

$\sigma$: A control parameter whose value is between 0.9 and 1.

- **Episodes** The number of time steps (iterations) of the AMR algorithm we would like to run.

Below, we describe the details of a single time step/ iteration of the Q-Learning algorithm. These steps will recursively occur until the termination of each episode.

### 4.5.2 Q-Learning Algorithm: Storage of Q-values in a Q-Table

At the beginning of the algorithm, the agent has no knowledge about the impact of its actions i.e., whether it would lead to a positive reward or a negative reward. As a consequence, it is unable to assign any specific value to each state-action pair.

Thus, in the absence of any knowledge about the environment, the Q-values for each state-action pair will be initially assumed to be zero or assign random values.

The Q-values will be stored in a tabular form via a Q-table whose rows represent the states and whose columns represent the actions.

| **Actions** | | | |
|---|---|---|---|
| **States** | uniform marking | $\eta_{res}$ based marking | $\eta_{gr}$ based marking |
| uniform mesh | 0 | 0 | 0 |
| adaptive mesh $\eta_{res}$ | 0 | 0 | 0 |
| adaptive mesh $\eta_{gr}$ | 0 | 0 | 0 |

While the above Q-table provides a possible representation of the environment, it provides insufficient information to the agent particularly about the accuracy and efficiency of the solution. Hence our assumption of denoting the states by discrete values needs to be modified to reflect the observed states of degrees of freedom, discretization error and convergence rate.

For assessing the accuracy and efficiency, we require to calculate the convergence rate:

$$CR(t) = 2\,\frac{\log|e_{t-1}/e_t|}{\log|DoF_t/DoF_{t-1}|}. \tag{4.34}$$

Hence, it is natural to instead consider the following quantities as the states:

1. $DoF_t$: DoFs at the current time $t$.

2. $e_t$: Energy norm at the current time $t$.

3. $DoF_{t-1}$: DoFs at the previous time $t-1$.

4. $e_{t-1}$: Energy norm at the previous time $t-1$.

With the above choices, the state space is no longer a finite space and as a consequence, it is impossible to represent the uncountable rows of the Q-table. To circumvent this problem, we discretize the state space into buckets and use these buckets as an entry in the Q-table.

A natural idea to circumvent this problem is, for each digit representing the state real number have a possible combination of digits $0, \cdots, 9$. This results in a prohibitively large Q-table and is not a practical approach. The approach we follow is to discretize the state space into buckets of range values and use these buckets as an entry in the Q-table.
As an example, consider a state space whose continuous values live between $-1.2$ and $0.6$. Using 20 buckets for each range can be computed by partitioning the interval $[-1.2,\ 0.6]$ into 20 sub-intervals of uniform length 0.09 each.

### Initializing Q-values

There are two ways to initializing the Q-table either all the values are set to zero which makes sense since the agent has no information about the worth of each state-action pair. Alternately, this can be initialized with random initial values.

### Updating Q-values

The Q-values of the Q-table are updated according to

$$Q^{new}(s_t, a_t) \leftarrow (1-\alpha) \cdot Q(s_t, a_t) + \alpha \cdot \underbrace{\left(r_t + \gamma \cdot \max_a Q(s_{t+1}, a)\right)}_{\text{learned value}} \tag{4.35}$$

where

- $\alpha$ is the learning rate. It tells us how quickly the agent abandons the current Q-value and how much information to keep.

- $\gamma$ is the discount factor indicating the weight on the future reward versus the immediate reward. It is between 0 and 1 with 1 indicating the greatest emphasis on long terms gains.

- $\max_a Q(s_{t+1}, a)$ is the estimate of optimal future value. It is calculated after the actions has been performed using the newly calculated $s_{t+1}$ see Figure 4.5.

**Epsilon Greedy Strategy: Exploration Vs. Exploitation**

After initializing the Q-values, the obvious question to be asked is:
what action should the agent take next in the absence of any prior information about how good any action is?
This is answered through first, exploring the environment to gain more information about it and then combining the exploring with exploiting the information already obtained about the environment.
Exploitation can be understood as choosing the action with the highest Q-value. Exploration on the other hand, picks its action randomly and learns by the impact of this action. Here, the agent would like to choose the *right* combination of exploring and exploiting the environment. This is because with too much exploiting, the agent might not be able to fully maximize its returns since it is not exploring other actions which might lead to better returns. Achieving a suitable combination of exploitation and exploration is realized by an epsilon greedy strategy.

**Epsilon Greedy Strategy:** This strategy is characterized by the parameter $\epsilon$ which represents the likelihood that the agent will explore the environment rather than exploit it. Initially, $\epsilon$ is set to 1 indicating the 100% certainty of exploring the environment. With each new episode, the agent learns more about the environment and the probability of exploring

the environment rather than exploiting it decreases. Simultaneously, with the information the agent gathers about its environment through exploration, it becomes "greedy" about exploiting the environment.

This is implemented in the Q-learning algorithm by:

1. Generating a random number between 0 and 1.
2. Exploit the environment if the random number exceeds $\epsilon$ otherwise explore.

To summarize, within each episode of the Q-learning algorithm, the following steps are executed:

1. Initialize all the Q-values to zero or a random initial value.
2. For each time step $t$ in each episode:
   (a) Choose an action keeping the exploration-exploitation trade-off.
   (b) Observe the reward and next state $R_{t+1},\ S_{t+1}$.
   (c) Update the Q-value function.

# Chapter 5

# Adaptive Mesh Refinement Implementation

## 5.1 Adaptive Strategy

The implementation of the adaptive algorithm is done according to the cycle:

$$\textbf{SOLVE} \implies \textbf{ESTIMATE} \implies \textbf{MARK} \implies \textbf{REFINE}$$

where:

- **SOLVE**

  This step is seeking a discrete solution $u_h \in V_h$ with respect to a triangulation $\mathcal{T}_h(\Omega)$ of the computational domain $\Omega$ such that

  $$a(u_h, v_h) = \mathcal{L}(v_h) \;\; u_h \in V_h, \; holds.$$

  The solve step is being done by direct solvers using sparse factorization technique with a suitable preconditoner chosen to guide the adaptive process since we are using small value of $\theta$.

- **ESTIMATE**

  The computation of the error estimators, the residual base error estimator $\eta_h$ and the gradient recovery or the so-called Zienkienwicz and Zhu's (ZZ) estimator $\eta_{GR}$.

- **MARK**

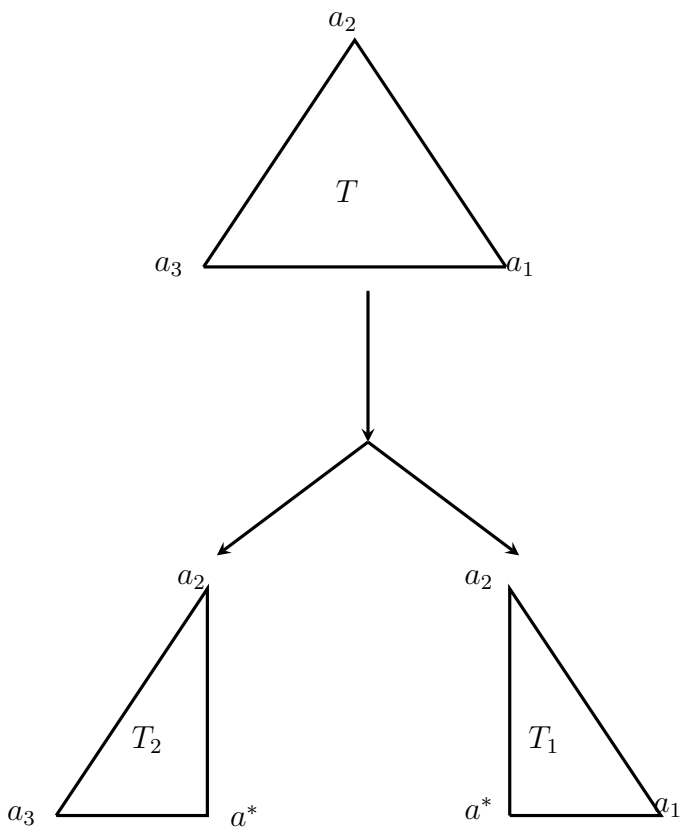
  Based on a given parameter $\theta \in [0, 1]$, we set a threshold and select edges and elements (for refinement) whose indicators exceed this threshold. The selection of the elements and edges for refinement is carried out using the bulk criterion $\theta\ \eta_h \leq \eta_{\mathcal{M}}$, where $\mathcal{M}$ denotes the set of cells and edges in $\mathcal{T}_h$ selected for refinement. There are different ways for selecting this threshold. One way is by marking the elements/edges with indicator of the largest magnitude(Maximum Strategy) which our algorithm depends on or by an averaged indicator(Equidistribution Strategy).

- **REFINEMENT** is realized using either:

  1. The newest vertex bisection in case $\mathcal{T}_h$ consists of triangles otherwise
  2. Quadrilateral refinement where each selected quadrilateral is divided into four quadrilaterals, in case $\mathcal{T}_h$ consists of quadrilaterals.

## 5.2 Refinement by the Newest Vertex Bisection.

The last step, refinement is done by bisection i.e., each marked simplex is divided into atleast two subsimplices. There are several bisection methods which depend either on the geometric structure of the triangulation (longest edge bisection) or are mainly concerned with the underlying topological structure(newest vertex bisection). The newest vertex bisection can be demonstrated as follows. Given a marked triangle $T = span\{a_1, a_2, a_3\}$, we we set a vertex and label it as the newest vertex. The subelements which comes from the $T$ share the new vertex.




Newext vertex bisection assign one of the vertices as the new vertex $a_2$, refinement is then carried on by connecting $a_2$ to the midpoint $a^*$ of the edge connecting $a_1$ and $a_3$. These subelements are orderd as $T_1 = span\{a_1, a^*, a_3\}$ and $T_2 = span\{a_3, a^*, a_2\}$.
The numerical implementation relies on the largest edge bisection. This bisection is a particular case of the newest vertex bisection wherein the newest vertex is fixed as the vertex that faces the longest edge [17].

The adaptive mesh refinement process is driven by ESTIMATE through estimators which could be: residual-type a posteriori error estimators (popular owing to low computational cost) or a wide range of averaging estimators and estimators based on local problems such as gradient recovery estimators. The role of the estimator is in two-fold namely: drive the adaptive process and play the role of the global discretization error. The goal is to automate the selection of estimators in the ESTIMATE step based on the data of the problem and computed solution.

| | | |
|---|---|---|
| $T_7$ | $T_8$ | $T_9$ |
| $T_4$ | $T_5$ | $T_6$ |
| $T_1$ | $T_2$ | $T_3$ |

Figure 5.1: $\Omega = (0,1)^2$

## 5.3 Marking Strategy

Considering the Figure 5.1 above on the domain $\Omega$ as a guide for our refinement strategy, the following can be deduced:

Triangulation: $\mathcal{T}_h = \{T_1, T_2, ..., T_9\}$, in this case considering quadrilaterals.

Estimators: $\eta_{Ti}, i = 1, 2, ..., 9$ and $\eta_{e_j}, e_j$ edges in $\Omega$.

If we consider the set $m = \{T_1, T_4, T_7\}$ to be our selected elements, that is elements with high error indicators.

Existing algorithms use this strategy:

$\sum_{T \in m} \eta_T = \sum_{i \in \{1,4,7\}} \eta_{T_i}$

$\eta_h = \eta_{T1} + ... + \eta_{T_9}$

$\eta_m = \eta_{T1} + \eta_{T_4} + \eta_{T_7}$

$$0 < \frac{\eta_m}{\eta_h} < 1$$

where the largest proportion exceeding $\theta$ is selected for refinement.

That is,

$$\frac{\eta_m}{\eta_h} > \theta.$$

We propose a new marking strategy in this work. The computed estimators: $\eta_{T1}, \eta_{T2}, ..., \eta_{T9}$ is arranged in decreasing order say $\eta_{T1} > \eta_{T4} > \eta_{T7}$.

In this case: $\underset{i=1,...,9}{max}\, \eta_{Ti} = \eta_{T1}$

And loop over all the $i's$ and check the criteria $\frac{\eta_{T1}}{\underset{i=1,...,9}{max}\, \eta_{Ti}} > \theta$ . Such implementation can be achieved by the algorithm below.

**Algorithm 1:** Adaptive Strategy

**Result:** $\eta_{Ti}$

initialization;

Function(f), Boundary condition(BD), Initial condition(IC), Order(polynomial Degree), $\theta$, maximum iteration(N)

Solve:

Compute: $U_h$, the approximate solution of the PDE

Estimation:

Calculate :

$\eta_{res}$ , $\eta_{zz}$

Marking Strategy:

Set $\underset{i=1,...,N}{Max}\, \eta_{Ti} = \rho$

Select $m = \{\eta_{Tj} \geq \ \phi,\ j = 1, ..., n : n \leq N\}$

**for** $i \in m$ **do**

  **if** $\frac{\eta_{Ti}}{\rho} \geq \theta$ **then**

    select $\eta_{Ti}$ for refinement;

  **end**

**end**

Refinement $\eta_{Ti}$ by the newest vertex bisection method.

# Chapter 6

# Numerical Experiments and Results

## 6.1 Benchmark Problems

We present numerical results for some benchmark problems focusing on the performance of the estimators in terms of the convergence rate. The experiment is performed for a suitable choice of parameters for all the three problems. That is :
$\theta = 0.25$ initial mesh size $h = 0.5$ and polynomial degree $k = 2$ in the finite element method.

The rate of convergence for the problems was calculated by the formula:

$$2\frac{\log|e_{n-1}/e_n|}{\log|DoF_n/DoF_{n-1}|}.$$

### 6.1.1 The Smooth Problem

In this example we want to show how an error estimator for the Poisson problem $-\Delta u = f \quad on \quad \Omega$ with $f(x,y) = sin(\pi x)sin(\pi y)$ and $\Omega = (0,1)\times(0,1)$ performs in terms of rate of convergence of the smooth solution. The exact solution of the above problem is:

$$\frac{-sin(\pi x)sin(\pi y)}{2\pi^2}.$$

The energy norm error reduces significantly which gives a faster expected convergence rate because the domain is smooth and the error is not redistributed.

Table 6.1: Smooth Problem: Uniform mesh refinement.

| level | Dofs | $\eta_h$ | $\eta_{GR}$ | H1-Error | RoC |
|---|---|---|---|---|---|
| 1 | 00025 | 0.0068683 | 0.0002060 | 2.945139e-02 | - |
| 2 | 00081 | 8.3440692e-05 | 1.9506340e-06 | 6.257659e-03 | 2.635 |
| 3 | 00289 | 1.890081e-06 | 3.180055-08 | 1.617140e-03 | 2.128 |
| 4 | 01089 | 3.2345233e-08 | 4.9117943e-10 | 4.078479e-04 | 2.077 |
| 5 | 04225 | 5.1695950e-10 | 7.6622648e-12 | 1.021892e-04 | 2.042 |
| 6 | 16641 | 8.1232537e-12 | 1.1969285e-13 | 2.556158e-05 | 2.022 |
| 7 | 66049 | 1.2710717e-13 | 1.8701039e-15 | 6.391289e-06 | 2.011 |
| 8 | 263169 | 1.9867194e-15 | 2.9220016e-17 | 1.597878e-06 | 2.006 |

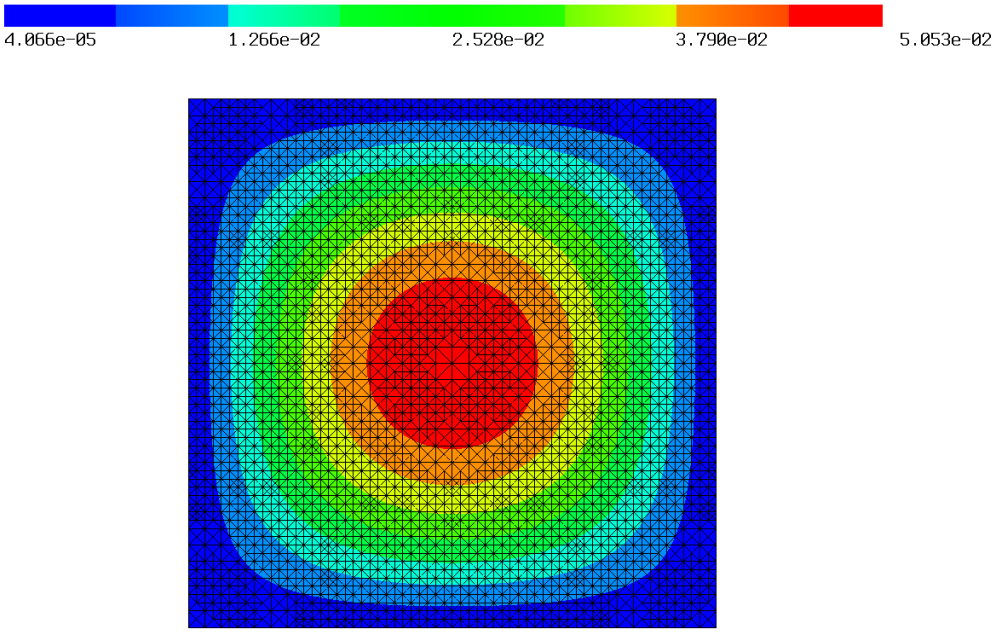



Figure 6.1: Smooth Problem: Uniformly Refined Mesh

Table 6.2: Smooth Problem: Residual error estimate adaptive mesh refinement.

| level | Dofs | $\eta_h$ | $\eta_{GR}$ | H1-Error | RoC |
|---|---|---|---|---|---|
| 1 | 00025 | 0.0068683 | 0.0002060 | 2.945139e-02 | - |
| 2 | 00081 | 8.3440692e-05 | 1.9506340e-06 | 6.257659e-03 | 2.635 |
| 3 | 00289 | 1.890081e-06 | 3.180055-08 | 1.617140e-03 | 2.128 |
| 4 | 00913 | 2.4583505e-07 | 6.5869731e-09 | 5.063194e-04 | 2.019 |
| 5 | 01129 | 5.1149859e-08 | 4.9110490e-10 | 4.052128e-04 | 2.098 |
| 6 | 03553 | 6.0074787e-09 | 9.0659197e-11 | 1.226124e-04 | 2.085 |
| 7 | 03993 | 1.276267e-09 | 8.397540e-12 | 1.053107e-04 | 2.606 |
| 8 | 10233 | 3.0201836e-10 | 5.065032e-12 | 5.512866e-05 | 1.376 |

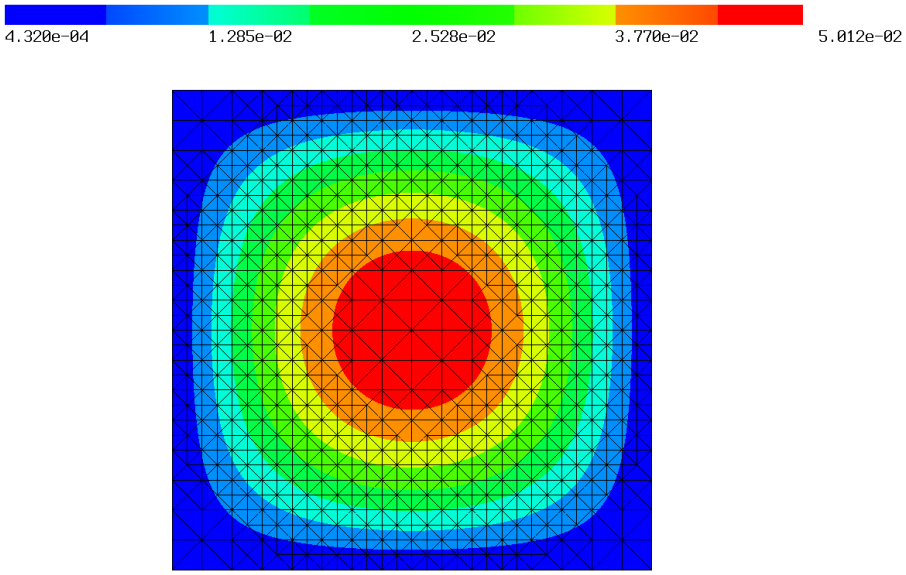



Figure 6.2: Smooth Problem: Residual Based Refined Mesh

Table 6.3: Smooth Problem: Gradient recovery estimate adaptive mesh refinement.

| level | Dofs | $\eta_h$ | $\eta_{GR}$ | H1-Error | RoC |
|---|---|---|---|---|---|
| 1 | 00025 | 0.0068683 | 0.0002060 | 2.945139e-02 | - |
| 2 | 00081 | 8.3440692e-05 | 1.9506340e-06 | 6.257659e-03 | 2.635 |
| 3 | 00289 | 1.890081e-06 | 3.180055-08 | 1.617140e-03 | 2.128 |
| 4 | 00977 | 2.0200548e-07 | 5.4226583e-09 | 4.665650e-04 | 2.041 |
| 5 | 01137 | 6.3905664e-08 | 4.9110263e-10 | 4.086185e-04 | 1.749 |
| 6 | 03753 | 5.8550893e-09 | 8.8769217e-11 | 1.145533e-04 | 2.130 |
| 7 | 04129 | 1.2764033e-09 | 7.662239e-12 | 1.037803e-04 | 2.069 |
| 8 | 14449 | 2.766121e-10 | 1.6127044e-12 | 3.009677e-05 | 1.976 |

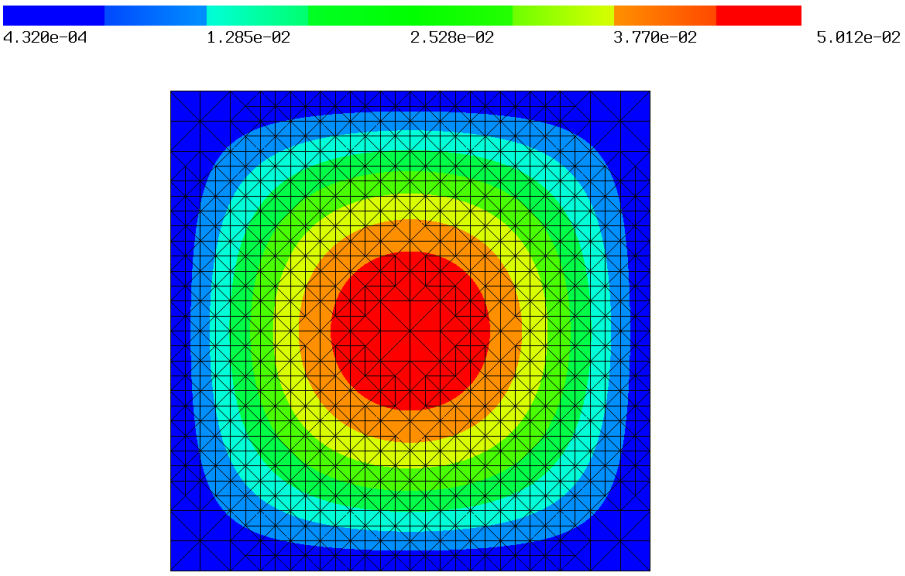



Figure 6.3: Smooth Problem: GR estimatorl Based Refined Mesh

### 6.1.2 The L-Shaped Domain Problem

Let $\Omega \subset \mathbb{R}^2$ be the L-shaped domain

$$\Omega := (-1,0) \times (-1,+1) \cup [0,1) \times (0,1)$$

and consider the Laplace equation

$$-\Delta u = 0 \ \ in \ \ \Omega,$$

$$u = \ \ 0 \ \ on \ \ \Gamma_D,$$

$$n.\nabla u \ = \ \ g \ \ on \ \ \Gamma_N,$$

with the homogeneous Dirichlet boundary conditions on

$$\Gamma_D := \{0\} \times [-1,0] \cup [0,1] \times \{0\}$$

and inhomogeneous Neumann boundary conditions on

$$\Gamma_N := \partial\Omega \setminus \Gamma_D.$$

The inhomogeneous Neumann boundary data $g$ are chosen such that

$$u(r,\psi) = r^{\frac{2}{3}} sin(\frac{2\psi}{3})$$

is the exact solution of the problem in polar coordinates $(r,\psi)$. The solution to the problem is in $H^{1+\frac{2}{3}-\epsilon}(\Omega)$ for any $\epsilon > 0$, and we expect its convergence rate to be approximately $\frac{2}{3}$. The reason for this observation is because the gradient has a singularity at the origin. This matches a theoretical predictions for a domain with a re-entrant corner and an interior angle of $\frac{3\pi}{2}$. Below we present the performance of the finite element method on uniformly refined meshes and on the uniformly refined meshes we expect the convergence rate to be $\frac{2}{3}$. We present similar table for adaptive mesh refinement driven by residual based estimator and gradient recovery estimator as well. Our algorithm selects the residual based estimator for refinement.

Table 6.4: L-Shape Problem: Uniform mesh refinement

| level | Dofs | $\eta_h$ | $\eta_{GR}$ | $H1 - Error$ | RoC |
|---|---|---|---|---|---|
| 1 | 00075 | 0.759915 | 0.2232191 | 2.232191e-01 | - |
| 2 | 00321 | 0.4974086 | 0.142589 | 1.455890-01 | 0.761945 |
| 3 | 01401 | 0.313052 | 0.089682 | 8.968227e-02 | 0.728929 |
| 4 | 05601 | 0.197711 | 0.056463 | 5.646320e-02 | 0.698668 |
| 5 | 26401 | 0.124148 | 0.035562 | 3.556230e-02 | 0.683353 |
| 6 | 131001 | 0.078202 | 0.022401 | 2.240109e-02 | 0.675519 |

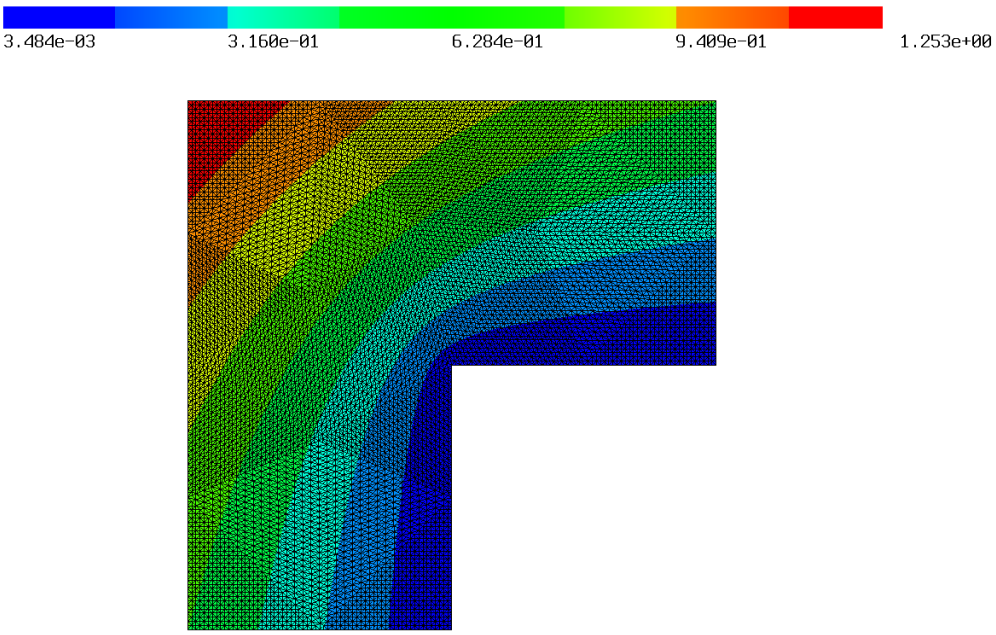



Figure 6.4: L-Shaped Domain: Uniformly Refined Mesh

Table 6.5: L-Shape Problem: Residual error estimate adaptive mesh refinement.

| level | Dofs | $\eta_h$ | $\eta_{GR}$ | $H1 - Error$ | RoC |
|---|---|---|---|---|---|
| 1 | 00075 | 0.7599155 | 0.2232191 | 2.232191e-01 | - |
| 2 | 00145 | 0.5547176 | 0.147487 | 1.474876e-01 | 1.717890 |
| 3 | 00253 | 0.354050 | 0.093685 | 9.368572e-02 | 1.765290 |
| 4 | 00351 | 0.233874 | 0.061012 | 6.101256e-02 | 2.798389 |
| 5 | 00447 | 0.163372 | 0.041477 | 4.147717e-02 | 3.260335 |
| 6 | 00545 | 0.124833 | 0.030432 | 3.043230e-02 | 3.204467 |

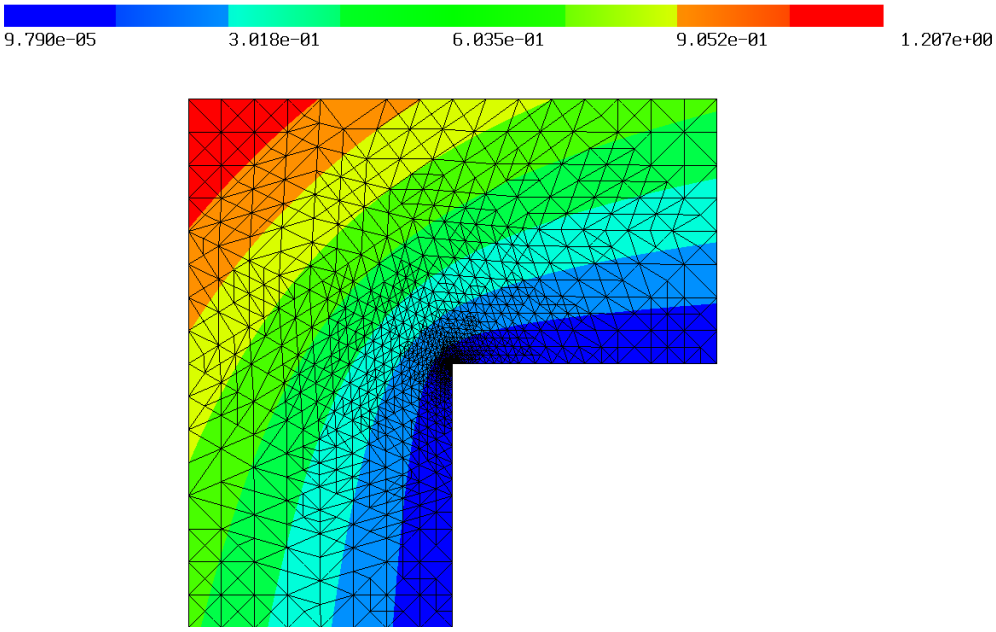



Figure 6.5: L-Shaped Domain: Residual Based Refined Mesh

Table 6.6: L-Shape Problem: Gradient recovery estimate adadptive mesh refinement.

| level | Dofs | $\eta_h$ | $\eta_{GR}$ | $H1 - Error$ | RoC |
|---|---|---|---|---|---|
| 1 | 00075 | 0.7599155 | 0.2232191 | 2.232191e-01 | - |
| 2 | 00145 | 0.5547176 | 0.1474876 | 1.474876e-01 | 1.717890 |
| 3 | 00237 | 0.360361 | 0.0936857 | 9.470750e-02 | 1.996711 |
| 4 | 00325 | 0.246257 | 0.062862 | 6.286266e-02 | 2.823021 |
| 5 | 00413 | 0.182430 | 0.044333 | 4.433312e-02 | 3.072927 |
| 6 | 00501 | 0.149861 | 0.034324 | 3.432473e-02 | 2.737314 |

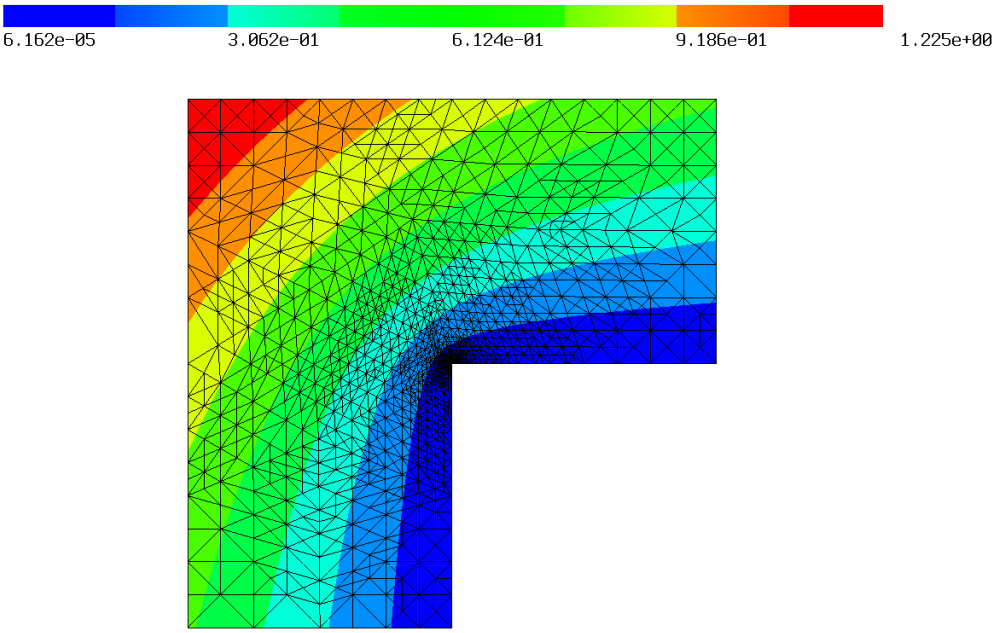



Figure 6.6: L-Shaped Domain: GR estimator Based Refined Mesh

### 6.1.3 Interface Problem (Discontinuous Coefficient Problem)

The following problem represents a domain $\Omega = (-1, 1)^2$ with discontinuous diffusion coefficient $\lambda$ defined as:

$$\lambda = -10 \text{ if } x < 0,$$

$$\lambda = 10 \text{ otherwise.}$$

The problem is to find $u$ in $V$ satisfying

$$\int_\Omega \lambda \nabla u \cdot \nabla v = \int_\Omega fv$$

for all $v \in V$ with homogeneous Dirichlet boundary conditions and $f = 1$.

Here, we do not have any information about the exact solution. However, through the tables and figures below, we can see that the estimators accurately capture the interface of discontinuity although the performance of the GR estimator is better than the residual estimator.

Table 6.7: Interface Problem: Uniform mesh refinement

| level | Dofs | $\eta_h$ | $\eta_{GR}$ | RoC of $\eta_h$ | RoC of $\eta_{GR}$ |
|---|---|---|---|---|---|
| 1 | 00025 | 0.9132653061217371 | 0.003911564625843614 | - | - |
| 2 | 00081 | 6.294394351142967e+56 | 2.9469009829382793e+54 | -222.6575 | -222.809 |
| 3 | 00289 | 2259.0325329307734 | 6.98513191803405 | 193.4967 | 194.1489 |
| 4 | 01089 | 0.339981183032501 | 0.0016699330906267576 | 13.2694 | 12.5717 |
| 5 | 04225 | 0.07367708108499874 | 0.0006148004909422946 | 2.2558 | 1.474068 |
| 6 | 16641 | 0.014637588580174497 | 0.00012054484848106703 | 2.3578 | 2.3770 |
| 7 | 66049 | 2.1970363870689945e-06 | 1.8122566871918527e-08 | 12.7733 | 12.771 |

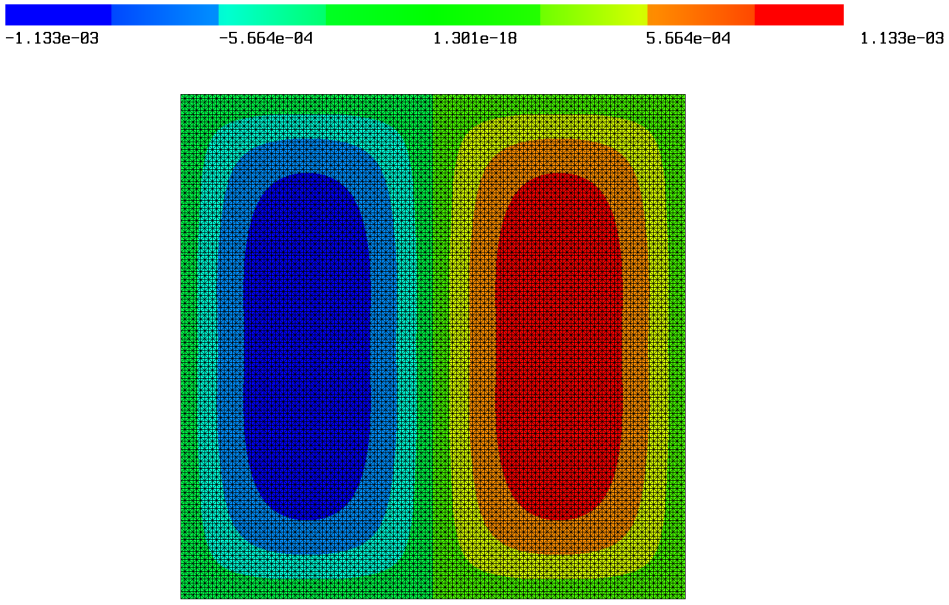



Figure 6.7: Interface Problem: Uniformly Refined Mesh

Table 6.8: Interface Problem: Adaptive-GR mesh refinement

| level | Dofs | $\eta_h$ | $\eta_{GR}$ | RoC of $\eta_h$ | RoC of $\eta_{GR}$ |
|---|---|---|---|---|---|
| 1 | 00025 | 0.9132653061217371 | 0.003911564625843614 | - | - |
| 2 | 00065 | 0.18755740207210225 | 0.0017729387500243139 | 3.31327 | 1.6562 |
| 3 | 00131 | 0.052578749119149136 | 0.000441506073477245 | 3.6294 | 3.9674 |
| 4 | 00249 | 0.02888131151997364 | 0.00011209433301554814 | 1.8656 | 4.2688 |
| 5 | 00545 | 0.028893190979869178 | 2.814221473347642e-05 | -0.0010 | 3.5286 |
| 6 | 01001 | 0.028893151497329468 | 7.042505474137851e-06 | 4.4952e-06 | 4.5571 |
| 7 | 02009 | 0.02889338301881942 | 1.994297748169939e-06 | -2.3004e-05 | 3.6221 |
| 8 | 03909 | 0.0288933856449429 | 6.004738900605176e-07 | -2.7308e-07 | 3.6065 |
| 9 | 07783 | 0.028893385530551062 | 1.6349018542162132e-07 | 1.1497e-08 | 3.7782 |
| 10 | 15389 | 0.028893385577928487 | 4.258039331688478e-08 | -4.8106e-09 | 3.9470 |
| 11 | 30713 | 0.028893385597913615 | 1.0856720204005415e-08 | -2.0018e-09 | 3.9552 |

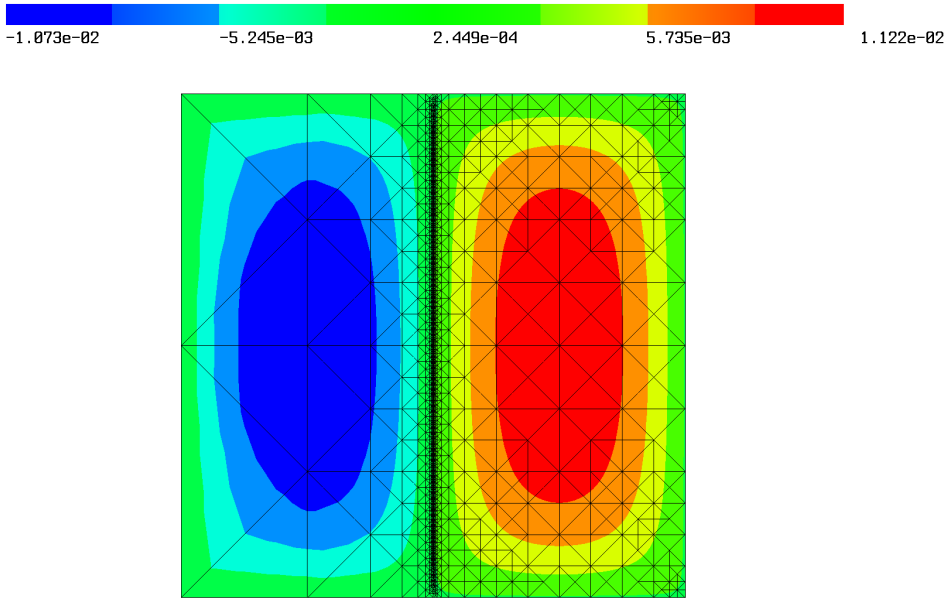



Figure 6.8: Interface Problem: GR estimator Based Refined Mesh

Table 6.9: Interface Problem: Adaptive-Res mesh refinement

| level | Dofs | $\eta_h$ | $\eta_{GR}$ | RoC of $\eta_h$ | RoC of $\eta_{GR}$ |
|---|---|---|---|---|---|
| 1 | 00025 | 0.9132653061217371 | 0.003911564625843614 | - | - |
| 2 | 00078 | 0.18785729188748698 | 0.0017791307231402168 | 2.7795 | 1.3847 |
| 3 | 00142 | 0.052631893883249264 | 0.00044181629538253336 | 4.2474 | 4.650 |
| 4 | 00333 | 0.013437065497157922 | 0.00011208565460063582 | 3.2037 | 3.2185 |
| 5 | 00601 | 0.0033749624479377407 | 2.813064565392496e-05 | 4.6799 | 4.6825 |
| 6 | 01178 | 0.0008444302229777924 | 7.037285215261504e-06 | 4.1174 | 4.1179 |
| 7 | 02358 | 0.0002344376325161656 | 2.0055764313611927e-06 | 3.6930 | 3.6175 |
| 8 | 04453 | 7.193592048480034e-05 | 6.126648738752348e-07 | 3.7165 | 3.7305 |
| 9 | 08869 | 2.265154950257648e-05 | 1.9524113666763506e-07 | 3.3543 | 3.3196 |
| 10 | 17340 | 7.040205691541692e-06 | 5.9007571709311796e-08 | 3.4859 | 3.5694 |
| 11 | 32524 | 2.022375136357052e-06 | 1.75244902772016e-08 | 3.9664 | 3.8605 |

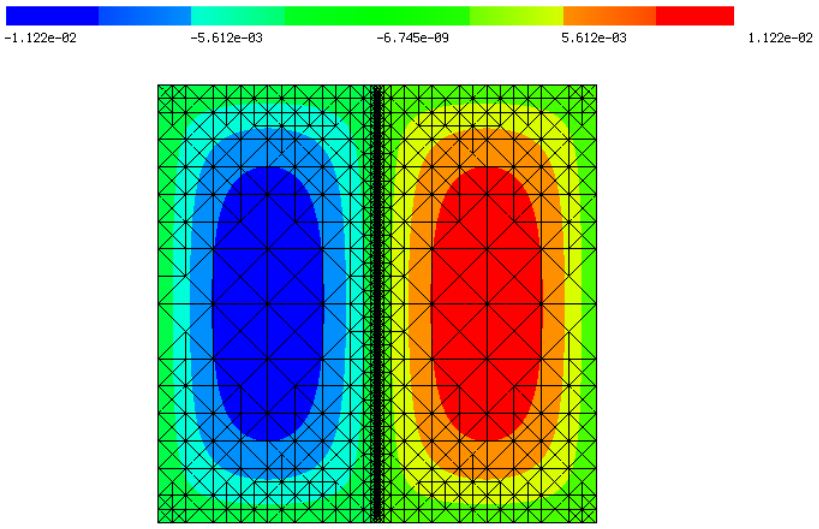



Figure 6.9: Interface Problem: Residual Based Refined Mesh

## 6.2 Applying Q-Learning Algorithm

In this section we provide details of the training of the Q learning algorithm.

Recall the learning algorithm recursively performs the following steps for each episode:

1. Initialize Q-values ($Q(s, a)$) arbitrarily for all state-action pairs.
2. For life or until learning is stopped...
3. Choose an action ($a$) in the current world state ($s$) based on current Q-value estimates ($Q(s, \cdot)$).
4. Take the action ($a$) and observe the the outcome state ($s'$) and reward ($r$).
5. Update $Q(s, a) := Q(s, a) + \alpha\left[r + \gamma \max_{a'} Q(s', a') - Q(s, a)\right]$

1. Initialize all the Q-values to zero or a random initial value.

2. For each time step $t$ in each episode:

   (a) Choose an action keeping the exploration-exploitation trade-off.

   (b) Observe the reward and next state $R_{t+1},\ S_{t+1}$.

   (c) Update the Q-value function.

The computations were carried out for the chip problem using `OpenAIgym` with the following choice of parameters:

1. Episodes = 20 Total training episodes

2. Steps = 3 (Maxsteps per episode)

3. min. alpha = 0.06 (learning rate)

4. min epsilon = 0.01 (exploration rate)

5. gamma = 0.91 (discount factor)

6. Decay epsilon = 3 (decay rate parameter for epsilon)

7. Decay alpha = 23 (decay rate parameter for alpha)

The choice of parameters described above were heuristically determined. We note that the optimal choice of the parameters in the Q-learning algorithm is an open question and will be addressed in the future research.

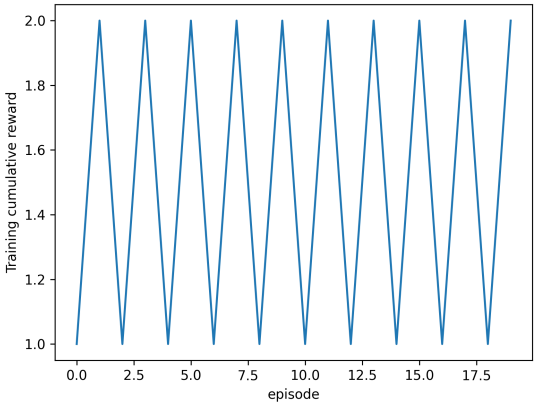



Figure 6.10: Outcome of the training Algorithm

The above figure indicates the cumulative reward earned over the 20 episodes. The total reward return on average is about 2 which is the expected convergence rate.

# Chapter 7

# Conclusion and Future work

## 7.1 Significance of the Results

In this thesis, we present the performance of two error estimators namely residual-based type and gradient recovery type illustrating the suitability of these estimators for different problems at hand. We also provide the foundation for data-driven adaptive mesh refinement strategies based on Reinforcement learning (RL) with a focus on the Q-learning algorithm which is a fundamental learning algorithm in RL.

Data-driven adaptive mesh refinement has been proposed and studied within the neural network framework [8, 23] however, to the best of the author's knowledge, this serves as the first work in the unsupervised learning paradigm.

## 7.2 Future Work

The choice of algorithm parameters in the reinforcement learning algorithm requires a more careful analysis and this is the topic of our future research. Another subject of our future investigation is the applicability of the algorithm to more challenging problems such as time dependent nonlinear higher order pdes (involving the Cahn-Hilliard equations) arising in material science where the dynamics of the problem occur at a very fast time scale and hence there is a need for time adaptivity in addition to spatial adaptivity.

# References


[1] Adrian Amor-Martin and Luis E Garcia-Castillo. Adaptive semi-structured mesh refinement techniques for the finite element method. *Applied Sciences*, 11(8):3683, 2021.

[2] Randolph E Bank and R Kent Smith. Mesh smoothing using a posteriori error estimates. *SIAM Journal on Numerical Analysis*, 34(3):979–997, 1997.

[3] AG Barto, RS Sutton, and CJCH Watkins. Learning and sequential decision making, learning and computational neuroscience: Foundations of adaptive networks, m. gabriel and j. moore, eds, 1990.

[4] R Bellman. Dynamic programming princeton university press princeton. *New Jersey Google Scholar*, 1957.

[5] Richard Bellman and Stuart Dreyfus. Functional approximations and dynamic programming. *Mathematical Tables and Other Aids to Computation*, pages 247–251, 1959.

[6] Marshall W Bern and Paul E Plassmann. Mesh generation. *Handbook of computational geometry*, 38, 2000.

[7] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-dynamic programming*. Athena Scientific, 1996.

[8] Jan Bohn and Michael Feischl. Recurrent neural networks as optimal mesh refinement strategies. *Computers & Mathematics with Applications*, 97:61–76, 2021.

[9] J David Brown and Lisa L Lowe. Multigrid elliptic equation solver with adaptive mesh refinement. *Journal of Computational Physics*, 209(2):582–598, 2005.

[10] WESLEY A Clark and Bernard G Farley. Generalization of pattern recognition in a

self-organizing system. In *Proceedings of the March 1-3, 1955, western joint computer conference*, pages 86–91, 1955.

[11] L Demkowicz, JT Oden, and Tf Strouboulis. An adaptive p-version finite element method for transient flow problems with moving boundaries. *Finite elements in fluids*, 6:291–305, 1985.

[12] Liu Du and Ningning Yan. Gradient recovery type a posteriori error estimate for finite element approximation on non-uniform meshes. *Advances in Computational Mathematics*, 14(2):175–193, 2001.

[13] HA Dwyer. Grid adaption for problems in fluid dynamics. *AIAA journal*, 22(12):1705–1712, 1984.

[14] BWAC Farley and W Clark. Simulation of self-organizing systems by digital computer. *Transactions of the IRE Professional Group on Information Theory*, 4(4):76–84, 1954.

[15] Ralf Hartmann, Joachim Held, and Tobias Leicht. Adjoint-based error estimation and adaptive mesh refinement for the rans and k–$\omega$ turbulence model equations. *Journal of Computational Physics*, 230(11):4268–4284, 2011.

[16] Ronald H.W. Hoppe. Lectures a posteriori error estimates for finite element approximations. URL: `https://www.math.uh.edu/~rohop/spring_05/downloads/Chapter6.pdf`, 2005.

[17] Ronald HW Hoppe and N Sharma. Convergence analysis of an adaptive interior penalty discontinuous galerkin method for the helmholtz equation. *IMA Journal of Numerical Analysis*, 33(3):898–921, 2013.

[18] Ronald A Howard. Dynamic programming and markov processes. 1960.

[19] KD Lee, JM Loellbach, and MS Kim. Adaptive control of grid quality for computational fluid dynamics. *Journal of aircraft*, 28(10):664–669, 1991.

[20] Shengtai Li. Comparison of refinement criteria for structured adaptive mesh refinement. *Journal of computational and applied mathematics*, 233(12):3139–3147, 2010.

[21] Dimitri J Mavriplis. Unstructured mesh generation and adaptivity. Technical report, institute for computer applications in science and engineering hampton va, 1995.

[22] Marvin Minsky. Steps toward artificial intelligence. *Proceedings of the IRE*, 49(1):8–30, 1961.

[23] Maciej Paszyński, Rafał Grzeszczuk, David Pardo, and Leszek Demkowicz. Deep learning driven self-adaptive hp finite element method. In Maciej Paszynski, Dieter Kranzlmüller, Valeria V. Krzhizhanovskaya, Jack J. Dongarra, and Peter M. A. Sloot, editors, *Computational Science – ICCS 2021*, pages 114–121, Cham, 2021. Springer International Publishing.

[24] Darrell W Pepper and David B Carrington. Application of h-adaptation for environmental fluid flow and species transport. *International journal for numerical methods in fluids*, 31(1):275–283, 1999.

[25] Kévin Pons and Mehmet Ersoy. Adaptive mesh refinement method. part 1: Automatic thresholding based on a distribution function. 2019.

[26] Sujata Prakash. Adaptive mesh refinement for finite element flow modeling in complex geometries. *Unpublished PhD Thesis. University of Toronto*, 1999.

[27] David Silver. Lectures on reinforcement learning. URL: `https://www.davidsilver.uk/teaching/`, 2015.

[28] Endre Süli. Lecture notes on finite element methods for partial differential equations. *Mathematical Institute, University of Oxford*, 2012.

[29] Richard S Sutton. Sutton & barto book: Reinforcement learning: An introduction. In *A Bradford Book*. MIT Press Cambridge, MA, 1998.

[30] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[31] Barna A Szabo. Estimation and control error based on p-convergence. Technical report, washington univ st louis mo center for computational mechanics, 1984.

[32] William R Thompson. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294, 1933.

[33] Marcos Vanella, Antonio Posa, and Elias Balaras. Adaptive mesh refinement for immersed boundary methods. *Journal of Fluids Engineering*, 136(4), 2014.

[34] Christopher John Cornish Hellaby Watkins. Learning from delayed rewards. 1989.

[35] Paul J Werbos. Building and understanding adaptive systems: A statistical/numerical approach to factory automation and brain research. *IEEE Transactions on Systems, Man, and Cybernetics*, 17(1):7–20, 1987.

[36] Ian H Witten. An adaptive optimal controller for discrete-time markov environments. *Information and control*, 34(4):286–295, 1977.

[37] Robert S Woodworth and Harold Schlosberg. Experimental psychology. new york: Henry holt and company. 1938.

[38] Qingluan Xue and Song-Charng Kong. Development of adaptive mesh refinement scheme for engine spray simulations. *Computers & fluids*, 38(4):939–949, 2009.

[39] Jiachen Yang, Tarik Dzanic, Brenden Petersen, Jun Kudo, Ketan Mittal, Vladimir Tomov, Jean-Sylvain Camier, Tuo Zhao, Hongyuan Zha, Tzanio Kolev, et al. Reinforcement learning for adaptive mesh refinement. *arXiv preprint arXiv:2103.01342*, 2021.

# Appendix

**Operators and Functional Spaces Operators**

Table 1: Operators and Function Spaces

| Notations | | | |
|---|---|---|---|
| Chapters | S/No | Operators | Function Spaces |
| | 1 | $\Omega \subset R^d; d = 1, 2$ : bounded polygonal domain | $L^2(\Omega)$: Hilbert Space of square integrable functions on $\Omega$ endowed with the norm: $\|f\|p = \left(\int_\Omega |f|^p\right)^{\frac{1}{p}}$ |
| | 2 | $\mathcal{T}_h$ = be a partition of $\bar{\Omega}$ into non-overlapping cells, either triangles or quadrilaterals | |
| | 3 | $h_h$: Diameter of cell $K(h = \underset{k\in\mathcal{T}_h}{max} h_k)$ | ${H^1}_{0,\Gamma_D}(\Omega) = V$ :Sobolev space of order '1' on $\Omega = \Gamma = \Gamma_D \cup \Gamma_N, \Gamma_D \cap \Gamma_N = \emptyset$ |
| | 4 | $\|K\|$ = Area of the cell $K$ | $S_{1,\Gamma_D}(\Omega; \mathcal{T}_h) = V_h : V \in C(\bar{\Omega}) : V|_{ki}$ is a linear polynomial $i = 1, 2$ |
| Chapter 3 | 5 | $\mathcal{P}_N(K)$ =is the space of polynomials on cell $K$ having degree at most $N$ | |
| | 6 | $\mathcal{E}_h^i$ =the set of all the edges of the interior cells in $\mathcal{T}_h$ | |
| | 7 | $\mathcal{E}_h^D$ = the set of all the edges of the cells in$\mathcal{T}_h$ excluding edges which intersect the boundary $\Gamma_N$ | |

Table 2: Operators and Function Spaces

| Notations | | | |
|---|---|---|---|
| Chapters | S/No | Operators | Function Spaces |
| | 7 | $\mathcal{E}_h^D$ = the set of all the edges of the cells in$\mathcal{T}_h$ excluding edges which intersect the boundary $\Gamma_N$ | |
| | 8 | $\mathcal{E}_h^N$ = the set of all the edges of the cells in$\mathcal{T}_h$ which intersect the boundary $\Gamma_N$ | |
| | 9 | $e$ = the edge of a cell | |
| | 10 | $h_e$ = length of an edge. | |
| | 11 | $(u, v)s$ = is the $L^2$inner product of $u$ and $v$ on $S \subset R^d, d = 1; 2..$ | |
| | 12 | $(u, v)_{\mathcal{T}_h} = \sum_{T \in \mathcal{T}_h} (\nabla u, \nabla v)_T$ and $\|\|u\|\|^2_{\mathcal{T}_h}$ | |

Table 3: Operators and Meaning

| Notations | | | |
|---|---|---|---|
| Chapters | S/No | Operator | Meaning |
| | 1 | $s, s^{'}$ | States(Actual State and Successor State respectively) |
| | 2 | $a$ | Action |
| | 3 | $r$ | Reward |
| | 4 | $A_t$ | action at time $t$ |
| Chapter 4 | 5 | $S_t$ | state at time $t$, typically due, stochastically, to $S_{t-1}$ and $A_{t-1}$ |
| | 6 | $R_t$ | Reward at time $t$, typically due, stochastically, to $S_{t-1}$ and $A_{t-1}$ |
| | 7 | $\pi$ | policy (decision-making rule) |
| | 8 | $\pi(s)$ | action taken in state $s$ under deterministic policy $\pi$ |
| | 9 | $\pi(a\|s)$ | probability of taking action $a$ in state $s$ under stochastic policy $\pi$ |
| | 10 | $v_\pi(s)$ | value of state $s$ under policy $\pi$ (expected return) |
| | 11 | $v_*(s)$ | value of state $s$ under the optimal policy |
| | 12 | $q_\pi(s, a)$ | value of taking action $a$ in state $s$ under policy $\pi$ |
| | 13 | $q_*(s, a)$ | value of taking action $a$ in state $s$ under the optimal policy |

# Curriculum Vitae

Augustine Twumasi was born on August 3, 1993 to Noah Fosu Manu and Dora Okyere.

He entered Kwame Nkrumah University of Science and Technology (KNUST) after High school in 2014 for his undergraduate studies in BSc Mathematics. He graduated in 2018 and was privileged to serve as a research and teaching Assistant. While pursuing his degree at KNUST he served at various leadership position including Financial Secretary of Science Students' Association(SCISA) and Treasurer for National Union of Ghana Students' (NUGS).

In pursuit of becoming a research scientist and academician in the field of computational and applied mathematics, he entered the Graduate School at The University of Texas at EL Paso in the fall of 2019 for his masters degree in Mathematical Sciences to lay a foundation for a PhD in Computational Science. He was a regular contributor and participant in a number of workshops and seminars under Mathematics, Computational Science, Data Science and Statistics.

While pursuing a master's degree in Mathematics he worked as a Teaching and Research Assistant. He was elected as Treasurer for the African Student Organization whiles pursuing his graduate studies at UTEP.

His research interests are Partial Differential Equations, Numerical Analysis, Optimization, Scientific Computing, Artificial Intelligence.

Augustine will be pursing his PhD in Computational Science at The University of Texas, El Paso in the fall of 2021. kwabenatwumasi94@gmail.com