Systems-Based Modelling and Optimisation of Fracture Toughness of Metal Alloys

Guangrui Zhang

Department of Automatic Control and Systems Engineering

University of Sheffield

A thesis submitted for the degree of

Doctor of Philosophy

September 2013

Acknowledgements

I would like to thank my supervisor, Prof. Mahdi Mahfouf, for the patient guidance, constructive advices and continuous support during my Ph.D. study.

I wish to thank my parents who supported me financially and mentally, their concern and encouragement were the source of my strength.

I want to thank all my friends and colleagues in the Department of Automatic Control and Systems Engineering, for their friendships and kind help.

Sincere thanks are due to everyone who has helped me in the past four years of my Ph.D. study. Without their support, I would never have completed this thesis.

Abstract

The modelling, prediction and prevention of material failures is the key issue during material design and processing. Finite Element Methods (FEM) combined with physically-based models are a popular approach to modelling fracture characteristics. However, in industrial practice, the interlinked process with high dimensions and complexities could be too complicated to be expressed purely on the principles of physics. Mathematical models via data-driven modelling approaches, were developed to remedy the aforementioned disadvantages of physically-based models. Therefore, this project focuses on developing a hybrid model to assess the toughness of metal alloys, and to improve the current material design techniques through the model-based optimal design.

Firstly, a data-driven model of the crack propagation of compact tension test is constructed; an error compensation strategy is also developed and tested. In order to improve the current material design technique through the model-based optimal design, a multi-objective particle swarm optimisation algorithm is proposed and tested using different benchmark functions. A data-driven model based finite element model structure is then proposed. Finally, the optimisation algorithm is applied together with the finite element analysis to the optimal design of material.

The results show that the constructed model for compact tension test and the error compensation strategy performed well. The proposed multi-objective particle swarm optimisation algorithm outperforms the compared two particle swarm optimisation algorithms; it is also applied successfully to the optimal design of the small punch test.

Publications

- Guangrui Zhang, Mahdi Mahfouf, George Panoutsos, and Shen Wang. A multi-objective particle swarm optimization algorithm with a dynamic hypercube archive, mutation and population competition. In *Evolutionary Computation (CEC), 2012 IEEE Congress on*, pages 1 – 7, Brisbane, Australia, 10–15 June 2012. doi: 10.1109/CEC.2012.6256489

- Guangrui Zhang, Mahdi Mahfouf, Qian Zhang, Sid-Ahmed Gaffour, John Yates, Sabino Ayvar Soberanis, Christophe Pinna, and George Panoutsos. Systems–modelling of compact tension energy in high strength pipeline steel. In *Proceedings of the 18th IFAC World Congress (IFAC WC 2011)*, volume 18(1), Milan, Italy, 28 August – 2 September 2011. doi: 10.3182/20110828-6-IT-1002.01466

- Shen Wang, Mahdi Mahfouf, and Guangrui Zhang. Robust fitness landscape based multi-objective optimisation. In *Proceedings of the 18th IFAC World Congress (IFAC WC 2011)*, volume 18(1), Milan, Italy, 28 August – 2 September 2011. doi: 10.3182/20110828-6-IT-1002.00767

Journal Papers in Preparation

(to be submitted to *APPLIED SOFT COMPUTING*)

- A New Multi-Objective Particle Swarm Optimisation Algorithm with Dynamic Hypercube Archive (mPSO-DHA) with Application to Finite Element Material Design Process.

- Systems-Modelling of Compact Tension Energy in High Strength Pipeline Steel, and Error Compensation using Gaussian Mixture Model.

Contents

List of Figures

Nomenclature

Acronyms

ANN	Artificial Neural Network
BIC	Bayesian Information Criterion
BP	Back Propagation
CAFE	Cellular Automata Finite Element
CI	Computing Intelligence
CMOD	Crack Mouth Opening Displacement
COA	Center of Area
COG	Center of Gravity
CT	Compact Tension
DB	Data Base that contains a set of parameters, scaling functions, and fuzzy membership functions
DHA	Dynamic Hypercube Archive
EA	Evolutionary Algorithm
ES	Evolutionary Strategy
FEM	Finite Element Methods
FIS	Fuzzy Inference System

GA	Genetic Algorithm
GMM	Gaussian Mixture Model
GTN	Gurson-Tvergaard-Needleman
KB	Knowledge Base
MCP	McCulloch-Pitts
MLP	Multi-Layer Perceptron
MOO	Multi-Objective Optimisation
MSE	Mean Square Error
$PAES$	Pareto Archived Evolution Strategy
PDE	Partial Differential Equation
PSO	Particle Swarm Optimisation
RB	Rule Base
RBF	Radial Basis Function
$RMSE$	Root Mean Square Error
RNN	Recurrent Neural Network
SOO	Single-Objective Optimisation
$SPEA(2)$	Strength Pareto Evolutionary Algorithm (2)
SPT	Small Punch Test
SVM	Support Vector Machine

Chapter 1

Introduction

Metallic materials are one of the priorities of the engineering studies. Different types of fractures occur from time to time in almost everywhere both in the industrial applications and in daily life, which result in substantial economic losses. Fig. 1.1 shows a pipeline blowout near Appomattox, Virginia, USA in the year of 2008 [EnergyindustryPhotos, 2008], which is only one of the numerous accidents that happen at all times all over the world, this particular accident having caused five persons injured and many homes were affected. The modelling, prediction and prevention of material failures hence becomes a key issue during the material design and engineering. This project therefore will focus on developing a hybrid model to assess the toughness of metal alloys, and improving the current material design technique through model based optimal design.

Many toughening and analysing methods for engineering materials have been developed, but due to the lack of physical knowledge and the uncertainties in the processing conditions, the actual effects of such methods of engineering are hard to quantify before processing. Therefore, modelling as a subject becomes an irreplaceable part of material science. On the one hand, process models, both physical and mathematical, are efficient tools to predict and analyse the damage of materials in quantitative expressions; on the other hand, engineers wish to find a way for a trade-off between improving material properties and minimizing the costs [Bhadeshia, 2008].

Figure 1.1: Appomattox, Virginia, USA, 2008, after a natural gas pipeline explosion [EnergyindustryPhotos, 2008]

Different modelling approaches have been presented in order to identify the link between process variables, microstructure features and material properties. Some modellers try to build models based on physics principles, which are also identified as white-box models.

Finite Element Methods (FEM) combined with physically based model are a popular approach to modelling fracture characteristics. For instance, a report by [Acharyya and Dhar, 2008] used the Gurson-Tvergaard-Needleman (GTN) model to predict the ductile failure of 22NiMoCr37 and SA-333 Gr-6 Carbon steel, and in [Karabin et al., 2009], the author developed a constitutive model for 7085-T7X (over-aged) aluminum alloy plate samples with controlled micro-structures based on Gurson-Tvergaard (GT) model and Leblond-Perrin-Devaux (LPD) [Leblond et al., 1995] model.

However, in industrial practice, the interlinked process with high dimensions and complexities could be too complicated to be expressed purely on principles of physics, and hence the white-box model may require more extensive tests besides the observed data from industrial process, such a model will be too expensive to be developed [Reed et al., 2009].

Mathematical models via data driven modelling approaches were developed to remedy the aforementioned disadvantages of physically-based models. These modelling methods include Artificial Neural Networks (ANNs), Fuzzy Inference Systems (FISs), Genetic Algorithms (GAs), Support Vector Machines (SVMs), Gaussian process, Bayesian inference, etc. Such data-driven modelling approaches are widely used in the area of material science and engineering. For instance, [Çöl et al., 2007] developed an ANN based model to predict the impact energy of API X65 micro alloyed steel; in [Ozbulut et al., 2007], the hysteretic behaviour of CuAlBe wire was predicted by creating Fuzzy Inference Systems (FISs) from experimental data. The open literature shows many more models used in different industrial processes of material engineering. These modelling methods, combined with physically based models, provide researchers more confidences in modelling newly developed materials.

The traditional modelling methods aim at building a deterministic model to predict certain characteristics with an acceptable level of accuracy. The predictions from this deterministic model can be very accurate when the training data is clean and spreads over the problem space. However, the predictions will not be reliable if the training data is not carefully selected and/or includes noise components. Effects of uncertainties in processing and service conditions represent tough challenges for the modellers. A sufficient way to correlate the prediction results with stochastic models is needed, where a confidence factor will be given as a reference. There is not much work in the literature about stochastic modelling on fracture. [Knott, 2006] presented a probabilistic assessment for the brittle fractures in ferritic steels, and despite the fact that the variability in the experimental results obtained from different steels was analyzed, the author did not present an application to compensate for the error between predicted and actual results.

The optimal design problem for material is one of the key issues in metallurgy. Once an intelligent model for a material model has been developed, it can be a huge advantage if the developed model can be implemented to facilitate the optimal design of metal alloys. The optimisation problems exist everywhere in reality. People always wish to solve problems with the best possible result while trying not to compromise on the costs. Such problems also exist in the processes surrounding metal alloys design and researchers have been devoting time and effort to solve such issues: improving the fracture toughness of the material and at the same time maintaining a certain level of material strength; or reducing the economic costs of production without compromising the integrity of the material. However, more often than not, ensuring the toughness of material and improving material strength can prove to be a hard balancing operation.

The aforementioned problems can be identified as falling within the theme of a Multi-Objective Optimisation (MOO) problem. Different from Single-Objective Optimisation (SOO), a single global solution to the MOO problem cannot be found at most times. It is often necessary to find a set of points that satisfy all the predefined optima, also known as Pareto optimality [Pareto, 1971]. Many efforts were put into solving the MOO problems in the past two decades.

Initially, the weighted-sum method, which converts the MOO problem to a series of SOO problems, was widely used. However, the weighted-sum approach often encounters the difficulty in selecting suitable weights, since the priori selection of weights cannot guarantee acceptable results in all situations, for example, suitable weights are not possible to find if the relationship between different objectives is nonlinear. Moreover, the weighted-sum method cannot obtain solutions which are on the non-convex portions of the Pareto optimal set in the criterion space [Messac et al., 2000]. Another disadvantage is that varying the weights consistently and continuously may not result in an even distribution of Pareto optimal points and an accurate, complete representation of the Pareto optimal set [Das and Dennis, 1997].

There exist several algorithms that can solve MOO problems by converting them into SOO problems. However, the requirement of gradient information, which must be calculated via objective functions and constraints, increases the computational costs and limits the application of such approaches. It is therefore necessary to develop new algorithms which can solve the MOO problems directly.

Thus, bio-inspired algorithms, such as Genetic Algorithms (GA) [Holland, 1975] and Particle Swarm Optimisation (PSO) [Kennedy and Eberhart, 1995] were tailored to solve the MOO problems.

There exist a number of reported applications which use bio-inspired optimisation techniques for alloy design. For example, Zhang and Mahfouf applied a modified PSO [Zhang and Mahfouf, 2009] and a new reduced space searching algorithm [Zhang and Mahfouf, 2010] to the design of alloy steels. The work aims at determining the optimal heat treatment regimes and the required weight percentages for chemical composites to obtain the desired mechanical properties of steel hence minimising production costs and achieving the overarching aim of right-first-time production of metals.

Those bio-inspired algorithms are global optimisation techniques, whose advantage is the ability to locate the global optimum rather than the local solution; it is also worth noting that these techniques have the capacity to combine any single-objective global technique as an optimisation engine [Leu and Yang, 1999] and the ability to converge to the Pareto optimal set as a whole.

1.1 Research Aims

The aim of this research is to develop a systematic optimal design approach of metal alloys.

This project is composed of two parts: 1) Fracture Modelling - the development of a model of material fracture and an error compensation method and 2) Process Optimisation - analyses and optimisation of metal alloy design process.

The modelling part in this project consists of a) developing data-driven models based on the existing data of steel crack propagation. An effective error compensating method is also required to compensate for the error and give a confident band of model predictions; b) eliciting a model of material, which combines the classical FEM with data-driven modelling techniques. An multi-objective optimisation algorithm is to be designed. The developed optimisation algorithm, together with the FEM, will be used for the systematic optimal design of metal alloys.

1.2 Contribution

A new multi-objective particle swarm optimisation algorithm is proposed. The integration of dynamic hyper-cube archive, weight adaptation, mutation, pool selection and modified global best selection techniques enhances the multi-objective optimisation ability of traditional particle swarm optimisation algorithms. Comparisons with other multi-objective optimisation algorithms show that the algorithm is effective in optimising various problems.

Models with Gaussian mixture model error compensation structure of the crack propagation are established. The Gaussian mixture error compensation structure shows the ability of providing the confidence band and a probable error for the predicted results from both fuzzy and neural network model. The predicting error is reduced through embedding the error compensation structure into the previously built model.

A synergistic model combining the data-driven model and FEM is developed. The constitutive equation model in the FEM is replaced by a neural network model. The simulation based on a simple element model using the synergistic approach leads to a good agreement with the results from the model using the constitutive equation model only. This synergistic modelling approach extends the FEM to model the material when the physical expertise of the material is missing.

The systematic optimal design approach of metal alloys based on FEM and the proposed new multi-objective PSO is successfully applied in the simulation of the small punch test. The optimal parameters for the material are estimated for different desired damage propagation process. The proposed optimal design approach introduces the idea of user-oriented design into the material design procedure.

1.3 Thesis Structure

The chapters will be structured as follows:

Chapter 2 introduces the background knowledge relating to the project. The introduction to Genetic Algorithms [Holland, 1975], Evolutionary Strategies [Rechenberg, 1973; Schwefel, 1977] and Particle Swarm Optimisations [Kennedy and Eberhart, 1995] are first presented, based on the single objective optimisation methods. The algorithms for Multi-Objective Optimisation are also introduced. Artificial Neural Networks, Fuzzy Logic and Fuzzy Systems are reviewed as they are widely used intelligent modelling methods. The Finite Elements Method, which is developed to model the materials through the "white-model" approach, is introduced briefly as well.

In Chapter 3, a multi-objective particle swarm optimisation algorithm (mPSO-DHA) is described. The algorithm extends the original particle swarm optimisation using dynamic hyper-cube archiving, weight adaptation, mutation operator, pool selection and modified global best selection. In order to be of certain the best parameter settings for the algorithm, various settings of parameters are explored, and an appropriate setting of parameters is selected. The application of mPSO-DHA to the well-known multi-objective benchmark functions, including ZDT [Zitzler et al., 2000] series and DTLZ [Deb et al., 2002] series, is analysed and compared with other multi-objective PSO algorithms, in order to assess the performance of the new algorithm. Comparisons with other evolutionary algorithms, such as PAES [Knowles and Corne, 1999], SPEA [Zitzler and Thiele, 1998] and NSGA2 [Deb et al., 2000], are concluded.

Chapter 4 reports on the modelling of steel crack propagation using fuzzy and neural-network, using the data acquired from previous work of [Ayvar et al., 2005]. Data analyses are carried out initially. The fuzzy modelling for crack propagation is detailed, using the method which has been used in [Zhang and Mahfouf, 2008, 2011] with hierarchical clustering initialization and gradient decent learning. After the fuzzy modelling, the double-loop neural network [Yang et al., 2003] is selected to modelling the same data. A comparison between the fuzzy model's and neural network's performances are also conducted. Moreover, and in order to reduce the error of models, an error compensation approach is proposed and examined, the approach uses Gaussian Mixture Model [McLachlan and Peel, 2004] to predict and to compensate for the errors.

Chapter 5 shows a data-driven model embedded within a finite element modelling approach (DMFEM). The idea of this approach is to replace the constitutive equations in the finite element model by a data-driven model. The neural network model for an isotropic hardening elastic-plastic simple element model is firstly constructed using MATLAB to verify the feasibility of the approach. Both the standard implicit user subroutine UMAT and the explicit user subroutine VUMAT are modelled. The constructed data-driven model is then transfered and embedded into ABAQUS environment in Fortran. The feasibility of the proposed DMFEM in modelling and analyzing material is validated.

In Chapter 6, an optimisation process of material design based on mPSO-DHA and finite element analysis is proposed and evaluated. The mPSO-DHA is applied to the finite element models which relate to the material failures in order to find the optimal material properties which will lead to the desired damage propagation process. The optimisations on different situations and design objectives for the small punch test are carried out, where the optimal values of elastic and plastic characteristics are estimated for minimization, maximization and specific value of the maximum load and corresponding displacement.

Finally, Chapter 7 includes a detailed conclusion of the project, and the future research directions will also be discussed.

Chapter 2

A Background to Optimisation, Intelligent Systems, and Finite Element Method

2.1 Nature-inspired Optimisation

Nature-inspired optimisation algorithms are inspired by natural phenomena, such as the biological mechanisms of evolution, the social behaviours of humans and animals, and other experiences of human activities. These techniques are all population-based iterative algorithms, the most successful and widely-used algorithms in the recent researches are Evolutionary Algorithms (EA).

Evolutionary algorithms are generic population based meta-heuristic optimisation algorithms. An EA uses the mechanisms inspired by biological evolution, which includes reproduction, mutation, recombination, and selection. EA covers many algorithms – the Genetic Algorithms (GA) [Holland, 1975], the Evolution Strategy (ES) [Rechenberg, 1973; Schwefel, 1977], Particle Swarm Optimisation (PSO) [Kennedy and Eberhart, 1995], etc. – these algorithms may differ in the implementation or the nature of the particular applied problem, however they all show the generality of searching the global optimum rather than local optimal points. Moreover, EA has the advantage of optimising complex or partial known problems as a non-gradient algorithm.

2.1.1 Genetic Algorithms

One of the most well known types of evolutionary algorithm is genetic algorithm. GA has been used as an efficient optimisation method ever since [Holland, 1975] published his work. In a genetic algorithm, a population of chromosomes, aka genotypes or genomes, evolves towards better solutions through a process of competition or controlled variation.

Although there exist several variants of GAs, it is widely accepted that GAs should include the following components:

- Genetic representation of the potential solutions;
- Fitness evaluation algorithm;
- The genetic operators, i.e. crossover operator and mutation operator, should be used for the reproduction process;
- A selection mechanism, which introduces competition and pressure to individuals;

The prototype GA uses binary coded chromosomes to encode the candidate solutions. The crossover for binary coded chromosomes is realized through picking two chromosomes randomly and exchanging a portion of the chromosomes. The single-point and two-point crossover [Goldberg, 1989] are mostly used in binary coded GA. However, other encoding methods such as real value coding, which can reduce the computational cost of binary coding based methods, are also possible and necessary when addressing different problems. Simulated Binary Crossover (SBX) was proposed by [Deb and Agrawal, 1995] for real valued GA. More details about GA are available in [Back, 1996].

2.1.2 Evolutionary Strategies

Evolution Strategies (ESs) are joint development of Bienert, Rechenberg and Schwefel in the 1970s at the Technical University of Berlin (TUB) in Germany [Rechenberg, 1973; Schwefel, 1977]. It was first implemented as an experimental procedure to deal with hydrodynamical problems such as shape optimisation of a bended pipe or a structure optimisation of a two-phase flashing nozzle [Back, 1996]. Schwefel first simulated a two membered ES on the first available computer, which now commonly has the name of (1+1)-ES. $(\mu+\lambda)$-ES and (μ, λ)-ES, were then introduced by [Schwefel, 1981] to incorporate the principle of a population. The selection scheme in this work is called multi-membered ES. Thereafter, different variations of ES were developed, such as recombinative ES and self-adaptive ES.

The two differences of ES from GA are: a) there is no crossover operator in ESs while GAs have crossover operators; and b) the step size of mutation operator in ES can adjust itself during the optimisation process, which provides ES with the self-adaption capability that GAs do not normally have.

2.1.3 Particle Swarm Optimisation

The particle swarm optimisation algorithm was proposed originally by [Kennedy and Eberhart, 1995]. It was then modified by [Shi and Eberhart, 1998a]. PSO is a search and optimisation method which is inspired by the social behaviours of animals such as bird flocking and fish schooling.

PSO shares many similarities with GA; it is initialized with a population of potential solutions, which are called particles, and then search for the optima by iterations. The particles will fly through the search space, where the velocity and direction are adjusted by their own and their neighbours' historical behaviours.

Various modifications of PSO have been developed in order to ensure convergence to the optimum, . One of the most focused part of these variants is the design of operator. There are mainly three ways of constructing the operator:

- Exploration – the flight direction and velocity of particles are affected by the global best solution of current swarm [Kennedy and Eberhart, 1995];
- Exploitation – the movement of particles is affected by the local best solution of all its neighbourhoods [Eberhart et al., 1996];
- Balanced – the trend of particles is affected by both the global and local optimums of current stage [Kennedy, 1999].

In comparison to GA, PSO has the advantages that it is easy to implement and the number of tuning parameters is less than that associated with GA. Particles in PSO are updated by internal velocity rather than the mutation and crossover which are used in GA. The simplicity and fast searching speed made PSO a more popular algorithm in recent years.

More related work and details about PSO can be found in [Eberhart et al., 2001].

2.2 Multi-Objective Optimisation Algorithms

Multi-objective optimisation is the process of optimising two or more conflicting objectives simultaneously, subject (or not) to certain constraints. Mathematically, a multi-objective optimisation problem can be written in the following form (minimization):

$$\begin{aligned} \text{Find a vector } \boldsymbol{x} \in X, \quad & \text{which minimizes:} \\ \boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), & f_2(\boldsymbol{x}), \cdots, f_k(\boldsymbol{x}))^T, \\ \text{subject to:} \quad & g_1(\boldsymbol{x}) \leq 0, \cdots, g_m(\boldsymbol{x}) \leq 0, \\ \text{and/or} \quad & h_1(\boldsymbol{x}) = 0, \cdots, h_n(\boldsymbol{x}) = 0. \end{aligned}$$

where X is the feasible space of decision vectors, $\boldsymbol{x}$ is the vector of decision variables, $k \geq 2$ is the number of objectives, and m, n are the number of inequality and equality constraints respectively.

The solution to a multi-objective optimisation problem is not simply optimising each objective to its extreme simultaneously. There usually exist multiple solutions which lead to such a situation: if any one of the objectives was optimised closer to its extreme, other objective(s) would suffer as the result. In order to identify the multiple solutions, the Pareto domination relationship was defined [Deb, 2008; Sawaragi et al., 1985] (minimization):

> One solution $\boldsymbol{x}_1 \in X$ dominates another solution $\boldsymbol{x}_2 \in X$, if $f_i(\boldsymbol{x}_1) \leq f_i(\boldsymbol{x}_2), \forall i \in k$, and $f_i(\boldsymbol{x}_1) < f_i(\boldsymbol{x}_2)$ for at least one $i \in k$, where k is the number of objectives and X is the feasible space of decision vectors.

Pareto-optimal solutions, also known as non-dominated solutions, were then defined as follows [Deb, 2008; Sawaragi et al., 1985] (minimization):

> One solution $\boldsymbol{x}^* \in X$ is Pareto optimal if for every $\boldsymbol{x} \in X$, $f_i(\boldsymbol{x}) \geq f_i(\boldsymbol{x}^*), \forall i \in k$, where k is the number of objectives and X is the feasible space of decision vectors.

Multi-objective optimisation is widely used in most areas of economics and engineering, which involve multiple objectives problems in control and decision making. Several well-known Multi-Objective Evolutionary Algorithms (MOEAs), which have been developed in recent years, will be reviewed in the following sections.

2.2.1 Pareto Archived Evolution Strategy

The Pareto Archived Evolution Strategy (PAES) was introduced by [Knowles and Corne, 1999] as a modification of ES. In the simplest form of PAES, the algorithm consists of a (1+1) evolution strategy, where a single parent generates a single offspring in each iteration. The elitist mechanism of PAES is a historical archive which is used to maintain some of the previously found non-dominated solutions. The mutated individuals are compared with the solutions in the historical archive. In order to maintain the diversity of solutions, PAES includes a crowding procedure that recursively divides the searching space. Each individual is located in a certain grid according to its fitness value. The number of individuals that reside in each grid is recorded in an external memory set. The external memory is then used as a reference in the selection and archiving process.

The advantages of PAES are: a) the computational cost is lower than some multi-objective GAs; and b) it is adaptive and does not need the critical setting of other extra parameters beside the number of divisions of the objective space.

2.2.2 Strength Pareto Evolutionary Algorithm

The Strength Pareto Evolutionary Algorithm (SPEA) was proposed by [Zitzler and Thiele, 1998]. SPEA introduces an elitism by storing the previously found non-dominated solutions into an archive, which is called the 'external non-dominated set'. In each iteration, the newly found Pareto solutions are compared with the solutions in the external set. Then the best solutions are copied into the external population. Each individual in the external set will be assigned with a strength value, which is computed based on the proportion to the number of solutions that the individual dominates. The fitness of each solution in the current population is calculated according to the strength of all the external individuals that dominate it. A clustering technique called 'average linkage method' is introduced in order to maintain the diversity of the solutions.

Based on SPEA, a second version [Zitzler et al., 2002] was proposed as Strength Pareto Evolutionary Algorithm 2 (SPEA2). The improvements in SPEA2 are:

- The fitness assignment procedure is not only based on how many individuals that certain solution dominates, the number of how many individuals dominate it, is also taken into account;
- A nearest neighbourhood density estimation scheme is added in order to give more precise guidance for the search process;

- The archiving truncation method is enhanced to guarantee the preservation of boundary solutions.

2.2.3 Non-dominated Sorting Genetic Algorithm II

The Non-dominated Sorting Genetic Algorithm II (NSGA-II) was proposed by [Deb et al., 2000]. NSGA-II is a more efficient revised version of the the Non-dominated Sorting Genetic Algorithm (NSGA) [Srinivas and Deb, 1994]. NSGA-II employs a crowded tournament selection operator which is designed to keep the diversity of the solutions. Unlike the case of other algorithms, there is neither an external memory nor any extra niching parameters in NAGA-II. The elitist procedure is realized through combining the best parents with the best offsprings. NSGA-II has then become the most popular multi-objective optimisation algorithm because of its parameter-less niching approach.

2.3 Artificial Neural Networks

The artificial neural networks (ANNs) are inspired by the biological nervous systems. They are also referred to as just neural networks. ANN is defined as a network implemented by a number of interconnected artificial neurons. A typical neuron is shown in Fig. 2.1.

The artificial neuron shown in Fig. 2.1 has no learning process on its own. The activation function, weights and threshold are predefined. x_0 is a bias input usually set to be 1, with $\omega_0 = b$ in which b is a threshold introduced to adjust the weighted sum of inputs. x_i is the ith input. ω_i is the weight for x_i. y is the output computed from $\varphi(.)$ which is the activation function. The activation function varies in different applications of ANN. The unit step function was used in the work of [McCulloch and Pitts, 1943]. By using unit step function as activation function, the neuron in Fig. 2.1 becomes a McCulloch-Pitts (MCP) unit.

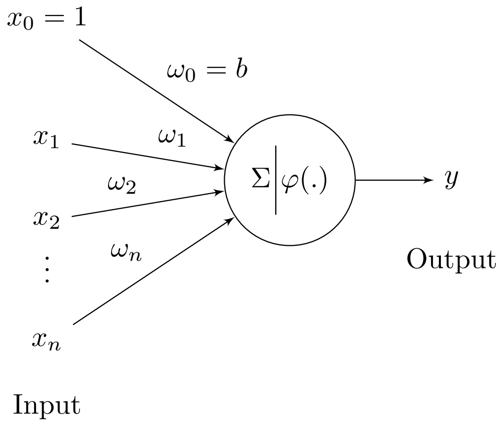


Figure 2.1: A typical artificial neuron

An ANN constructed by MCP neurons can simulate any kind of logical function [McCulloch and Pitts, 1943]. MCP units are useful in binary classifications. Other activation functions, such as a linear combination of all the inputs or the sigmoid function, can also be selected to fit the requirements of specific problems.

If the activation function is linear combination and the inputs and outputs are in a linear relationship, it is simple to set the weights by using Least Squares approximations. An ANN based on linear neurons (which use linear combination as activation functions) actually performs a linear transformation of the input vector. Non-linear relationships in linear neurons can only be approximated by polynomial expansion. Theoretically, any continuous function can be approximated as accurately as desired by a polynomial function. Therefore, a wide range of complex functions can be approximated by linear neurons.

However, the computational cost of approximating non-linear relationships using linear neuron shows no benefit compared to other algorithms. A non-linear activation function (sigmoid function [Cybenko, 1989]) was proposed as an alternative solution. By using a sigmoid function, the accuracy of ANN no longer depends on the degree of polynomial expansion, but depends on the training of weights.

The Least Squares approximation can not be applied to the process of weights training in case of using sigmoid activation functions. In this situation, the gradient descent algorithm is an efficient method to determine the weights, since derivative of the sigmoid function, such as logistic function, is easy to calculate.

$$\boldsymbol{\omega}(t+1) = \boldsymbol{\omega}(t) - \gamma_t \frac{\partial J}{\partial \boldsymbol{\omega}(t)} \tag{2.1}$$

The gradient descent method (shown in Eq. 2.1) computes the derivative of the Mean Squared Error (MSE) with respect to the changes of weights, and then changes the weights in every iteration so that the weights always lead the MSE to its extreme, where $\boldsymbol{\omega}$ represents the weight vector, t the iteration number, γ_t a small enough positive number as the learning rate, and J the cost function, which is MSE in this case.

Multi-Layer Perceptron (MLP) can then be constructed based on a network of sigmoid units. MLP is a feed-forward network that consists of multiple layers of neurons - one input layer, several hidden layers and one output layer. Each layer is fully connected to the next one. Back-Propagation (BP) [Rumelhart et al., 1986] is adopted as the weights training algorithm. The BP technique is in fact a superposition of two gradient descent processes. It firstly computes the changes of weights for output layer, then the changes of weights for hidden layer(s) can be determined according to the new weights of output layer, so that the algorithm is named “back propagation”.

It is proved in [Cybenko, 1989] that when using only one sigmoid hidden-layer and one linear output, MLP becomes a universal approximator that can approximate any well-behaved function to arbitrary accuracy as long as there are appropriate number of hidden neurons. The hidden layer in such a situation can be seen as a pre-processor while the pre-processing function is adapted by learning technique rather than priori determined. Compared to the polynomial expansion approach, MLP can fit a more compact model.

The performance of ANN varies since many types of ANN were developed and applied in different research areas. The feed-forward network (e.g. MCP and MLP) is commonly used because of its simple structure. Radial Basis Function

(RBF) networks [Broomhead and Lowe, 1988], in which radial basis functions such as Gaussian are usually selected as the activation function for hidden layer(s), have the advantage of not suffering from local minima. RBF networks also proved efficient in solving classification problems. Recurrent Neural Networks (RNNs) [Elman, 1990] use a directed cycle as the connection between units, which gives the network an internal memory space. RNN performs well in unsegmented connected handwriting recognition [Graves et al., 2009]. The above networks only reflect a small portion of ANNs. There are still lots of ongoing cutting-edge researches in ANNs, such as those related to biophysical models and nano-devices, which could lead to neural computing.

2.4 Fuzzy Logic and Fuzzy Systems

Fuzzy sets were first proposed by [Zadeh, 1965] of the University of California. The basic ideas and foundations of fuzzy system were developed based on Zadehs theory in the late 1960s. Fuzzy modelling is a systems approach which describes the process under investigation using fuzzy quantities, such as fuzzy sets, fuzzy rules, and linguistic labels. A fuzzy system is known to be good at universal approximation; it has the ability of modelling complex, nonlinear or partial known systems. The main advantage of the fuzzy modelling technique is its ability to represent nonlinear complex systems via simple modelling structures; the linguistic “if-then” rules are easy to understand by non-experts, and this can improve the model’s transparency [Wang, 1997].

2.4.1 Fuzzy Sets

Fuzzy set theory allows the elements have degrees of membership to a fuzzy set, while in a classical set, the membership of elements is defined in binary – an element either belongs or does not belong to the set. The membership of elements is described with the aid of a membership function valued in the real unit interval $[0, 1]$. The classical sets can therefore be seen as special conditions of the fuzzy sets – if the membership functions of a fuzzy set only take the values 0 or 1. According to [Zadeh, 1965], the definition of a fuzzy set is:

> *"Let X be a space of points (objects), with a generic element of X denoted by x. Thus, $X = \{x\}$. Then a fuzzy set (class) A in X is characterized by a membership (characteristic) function $f_A(x)$ which associates with each point in X a real number in the interval $[0, 1]$, with the value of $f_A(x)$ at x representing the 'grade of membership' of x in A."*

For example, a fuzzy set using Gaussian function as the membership function can be written as (A, μ_A), where $A = \{x_1, \cdots, x_n\}$ is a classical set, $\mu_A(x_i)$ is the degree of membership of x_i. The fuzzy set (A, μ_A) can also be denoted as $\{\mu_A(x_1)/x_1, \cdots, \mu_A(x_n)/x_n\}$.

Some of the popular used membership functions are shown in Fig. 2.2, including Gaussian functions, triangular-shape functions and trapezoidal-shape functions.

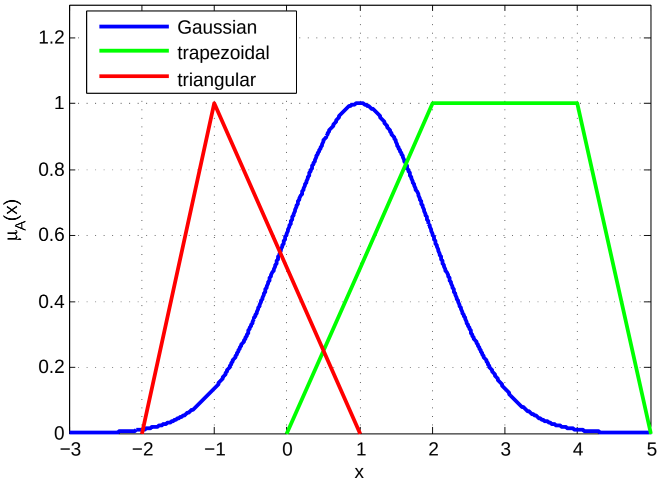


Figure 2.2: Several membership functions

2.4.2 Fuzzy Systems

A fuzzy system as shown in Fig. 2.3 generally consists of four parts: a fuzzifier, a fuzzy rule base, a fuzzy inference engine and a defuzzifier.

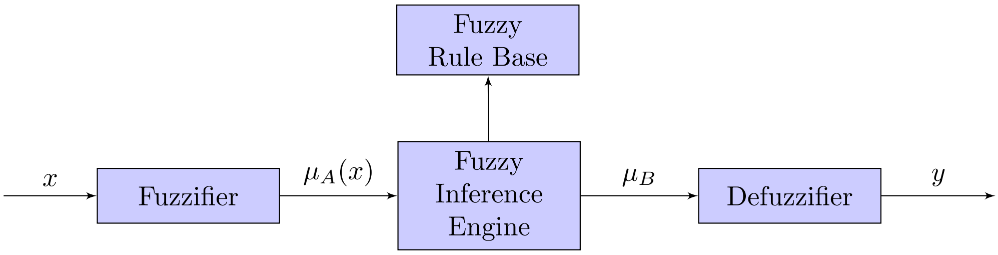


Figure 2.3: A Typical Structure of Fuzzy Systems

The fuzzifier is a component that maps the real valued input variable x to a fuzzy set A; it is simply a set of membership functions. Three types of fuzzifiers are frequently used as mentioned in Fig. 2.2. The defuzzifer, in contrast to fuzzifier, is implemented to specify a point y that best represents a fuzzy set B in the output space. The mostly used defuzzifier techniques are centre of gravity (COG), centre of area (COA), centre average, and maximum defuzzifier.

The fuzzy rule-base is the core of a fuzzy system, which consists of the fuzzy If-Then rules. The expression of a fuzzy rule can be different, there are, however, two types of fuzzy rule bases are popular: Mamdani-type [Mamdani, 1974] and Sugeno-type (TSK) [Sugeno and Yasukawa, 1993]. A Mamdani fuzzy If-Then rule is a statement in the following form:

$Rule^m$: **IF** x_1 is A_1^m **AND** $\cdots$ **AND** x_n is A_n^m **THEN** y^m is B^m

where m is the number of ruels, A_n^m and B^m are fuzzy sets in the input space $U_n \subset R$ and $V \subset R$ respectively, $x_n \in U_n$ and $y^m \in V$ are the input and output variables of the fuzzy system respectively.

A TSK If-Then rule is different from Mamdani rule in the way that outputs are determined. It can be described as follow:

$Rule^m$: **IF** x_1 is A_1^m **AND** $\cdots$ **AND** x_n is A_n^m **THEN** $y^m = g^m(x_1, \cdots, x_n)$

where the premise part is the same as Mandani rule, however, the consequence part is quantitative. g^m can be a linear or quadratic function. The defuzzifier is not used in this case.

The fuzzy inference engine is a component in which the fuzzy logic operations are processed, the fuzzy “If-Then” rules are combined in order to build a map from the fuzzy inputs to the output fuzzy set. The fuzzy operators include Containment, Union, Intersection, Complement and Cartesian product. These operators and inference process are detailed in [Wang, 1997].

2.4.3 Combining Fuzzy Systems with Other Approaches

Classical fuzzy modelling methods (mostly Mamdani-types) define the fuzzy systems based on the expert knowledge and experience, which will lead to a limitation of fuzzy modelling applications: when the expert knowledge is lacking or not available, traditional expertise-based fuzzy model cannot be generated. An efficient machine learning approach can prove to be a good solution to this situation.

There are successful approaches to expanding the capability of classical fuzzy systems by introducing learning mechanisms: neuro-fuzzy systems and genetic fuzzy systems. Such hybrid systems combine the approximately reasoning ability from fuzzy systems with the learning ability from neural networks and genetic algorithms, the scope of fuzzy systems application can then be expanded.

The neuro-fuzzy system was first proposed by [Jang, 1993], where a framework of Adaptive-Network-based Fuzzy Inference System (ANFIS) was introduced. The structure of ANFIS is a combination of ANNs and fuzzy systems. It is flexible and easy to modify. Many variations of ANFIS were then developed in order to address different problems [Nauck et al., 1997]. Chen and Linkens have used a hierarchical neural network to generate fuzzy model, identify the fuzzy model structure, and optimise the parameters for fuzzy model [Chen and Linkens, 2000]. The authors also proposed a succesful approach to rule-base self-generation and simplification for fuzzy models [Chen and Linkens, 2004].

The genetic fuzzy system is another technique for enhancing the learning ability of classical fuzzy systems. The learning process in genetic fuzzy system is regarded as an optimisation process of the Knowledge Base (KB). Genetic algorithm can be applied to different levels in a fuzzy inference system. The simplest case is genetic tuning, which uses a GA to obtain a set of associated parameters of the scaling functions and the fuzzy membership functions (DB). In a mediate level, the application of GAs is a genetic rule base learning, which applies GA in searching for a suitable and complete rule-base (RB). Three approaches are available for genetic rule-base learning: the Michigan approach [Holland and Reitman, 1977], the Pittsburgh approach [Smith, 1980], and the iterative rule learning approach [Venturini, 1993]. The first two aforementioned applications of GA can be combined together in order to optimise the DB and RB at the same time. However, this method leads to increase in computational costs as the complexity of searching space increases. The three approaches for genetic rule-base learning are also available for this combined method.

The fuzzy system can also be combined with the classical control techniques. Babuska proposed a fuzzy self-tuning PI controller to control the pH value in fermentation, the results of simulations proved the capacity of the proposed controller [Babuska et al., 2002].

It is worth noting that maintaining interpretability is important in the design of neuro-fuzzy systems and genetic fuzzy systems. Since it is contradictory that improving the accuracy and in the meantime simplifying the rule base structure, multi-objective optimisation algorithms are used to determine how many rules are included in the rule base and how many inputs are involved in each rule [Babuska and Verbruggen, 2003; Gonzlez et al., 2007; Wang et al., 2005].

2.5 Finite Element Method

Finite Element Method (FEM), also known as Finite Element Analysis (FEA), is an engineering analysis tool to find approximate solution to complicated systems, by solving partial differential equations (PDE), as well as integral equations. The idea of FEM as priorly explored until [Clough, 1960] coined the phrase 'Finite Element Method'. FEM divides a very complicated problem into small elements that can be solved in relation to each other, while in each element, the problem can be solved easily by reducing the partial differential equations into a set of algebra equations or ordinary differential equations.

FEM is a continuously developing method in a wide variety of engineering disciplines such as electromagnetism, fluid dynamics, civil and aeronautical engineering, material and mechanical engineering. In mechanical engineering, FEM is widely used in the simulation of the damage process of material. A series of models were developed for different types and conditions of damages. These include numerous texts in the literature ranging from the late 1960s until now.

There are several models which were developed to analyze ductile fractures, some of the commonly known models are, Rice-Tracey model [Rice and Tracey, 1969], Gurson model [Gurson, 1975], Rousellier model [Rousselier, 1987], and Argon-Im-Safoglu model [Argon et al., 1975].

2.6 Summary

In this chapter, some background knowledge relating to this project has been introduced. The basic concepts of nature-inspired optimisation have been reviewed and some of the popular optimisation algorithms have been discussed, which include Genetic Algorithms, Evolutionary Strategies, Particle Swarm Optimisation, Pareto Archived Evolution Strategy, Strength Pareto Evolutionary Algorithm, and Nondominated Sorting Genetic Algorithm II. The basics of artificial neural networks, fuzzy sets, fuzzy systems have then been briefly described. Finally, the finite element method has been introduced together with some of the well-known models of ductile fractures.

The next chapter will report on a multi-objective particle swarm optimisation algorithm with applications to multiple objective problems.

Chapter 3

A Multi-Objective Particle Swarm Optimisation Algorithm with Applications to Multiple Objective Problems

3.1 Basics of Particle Swarm Optimisation

The Particle Swarm Optimisation (PSO) algorithm received significant attention since [Kennedy and Eberhart, 1995] first introduced it in 1995. This powerful optimisation algorithm was inspired by the social behaviour of animals, such as birds flocking and fish schooling.

The algorithm starts with a group of random solutions, which are called particles. These particles then fly through the objective space based on both the historical behaviour of themselves and the behaviour of other particles. Similar to the birds flocking activities, the particles are expected to fly towards the optimal solutions in the optimisation process.

The original velocity and position updating mechanism uses the following equations [Shi and Eberhart, 1998a]:

$$\begin{cases} V_i^n = w \times V_i^{n-1} + C_1 \times R_1 \times (X_{Pb_i} - X_i) + C_2 \times R_2 \times (X_{Gb_i} - X_i) \\ X_i^n = X_i^{n-1} + V_i^n \end{cases} \tag{3.1}$$

where n is the index of the iteration, w is the inertia weight, $C_1, C_2 > 0$ are acceleration coefficients, $R_1, R_2 \in [0, 1]$ are uniformly distributed random variables. X_i and V_i are the position and velocity for the ith particle in the population, X_{Pb_i} is the best position found by X_i in the previous iterations, and X_{Gb_i} is the best position found so far by the whole population of particles.

It can be seen that the velocity of one particle is determined by the previous velocity (first component of Eq. 3.1), the “personal view” of the particle (second component of Eq. 3.1), and the “cooperation” from other particles (third component of Eq. 3.1).

The first component represents the inertia that the particle tends to keep the same direction as it traveled in the previous iterations. The inertia weight w determines how much the previous velocity will effect the particle’s behaviour. [Eberhart and Shi, 2001, 2000; Shi and Eberhart, 1999, 1998b] analysed the effects of the inertia weight using fixed and dynamically changing weights. In their study, it is concluded that: a larger inertia weight lets the particles move more freely, so that the global optima will be found faster; and a smaller value of w enhances the particle’s ability of exploring the neighboring space, which encourages the particle to converge the local optima.

The second component of the equation is the personal thinking of particle, which makes the particle to fly towards the best position in its own experience. The third component, which attracts the particle to move towards the best position found so far by all particles, represents the cooperative effect from the whole population and the historical search. Although [Eberhart and Shi, 2001] suggested that $C_1, C_2 = 2$ are supposed to perform well in most problems, the acceleration coefficients in these two components do not have to be necessarily equal, the values of acceleration can vary depending on the problem to be solved [del Valle et al., 2008].

In order to maintain the particles flying within the search space, some limitations are applied in Eq. 3.2:

$$\begin{cases} V^{max} = (X^{uplim} - X^{lowlim})/N, \\ \\ \text{IF } V_i > V^{max}, \text{THEN } V_i = V^{max}, \\ \text{ELSEIF } V_i < -V^{max}, \text{THEN } V_i = -V^{max}, \\ \\ \text{IF } X_i^n > X^{uplim}, \text{THEN } X_i^n = X^{uplim}, \\ \text{ELSEIF } X_i^n < X^{lowlim}, \text{THEN } X_i^n = X^{lowlim}, \end{cases} \tag{3.2}$$

where N is a positive number in $[1, 10]$, X^{uplim}, X^{lowlim} are the upper and lower limits of the search space, V_{max} is the maximum velocity of particles, V_i is the velocity of ith particle, X_i^n is the position of ith particle in the nth iteration.

The selection of N must be carefully considered; a too large or small V^{max} will cause the particle to move either erratically or restrictedly. It is worth noting that V^{max} can be dynamically changing in the work of [Fan and Shi, 2001], which may improve the performance of the algorithm. In addition, other methods were proposed to handle the boundary problem in [Huang and Mohan, 2005; Robinson and Rahmat-Samii, 2004].

In general, the PSO algorithm optimisation procedure can be summarised as follows:

1. Initialisation, randomly generate a swarm of particles within the searching space;
2. Evaluation, assign the fitness values to the particles according to the predefined objective functions;
3. Memorization, for each particle, compare current fitness value with its personal best position, store the better one as the new personal best position;

4. Navigation, compare the fitness values of all particles, locate the particle that has the best fitness value, and store it as the global best position;

5. Acceleration, update the velocity of all particles using Eq. 3.1;

6. Swarming, limit the velocities and update the position of all particles using Eq. 3.2;

7. Iterating, repeat from Step 2 until the maximum number of iterations is reached or a satisfying result is achieved.

3.2 Extending Particle Swarm Optimisation into Multi-Objectives Optimisation

It has been shown that PSO is an efficient algorithm for solving single objective optimisation problems [Poli, 2008]. PSO has the advantages of a simpler structure and a lower computation cost, as well as a fast convergence speed. In order to introduce these advantages into the optimisation of multi-objective problems, many studies have been published. Coello proposed an algorithm [Coello Coello and Lechuga, 2002] that extended PSO to handle multi-objective problems using a non-dominated and hypercube archiving technique, and Raquel presented a multi-objective PSO [Raquel and Naval, 2005] using a crowding distance ranking technique. There are many different variations of multi-objective PSO each introducing different techniques for tackling specific problems [Coello Coello and Lechuga, 2002; Hu et al., 2003; Raquel and Naval, 2005; Zhang and Mahfouf, 2006].

However, when modifying PSO to solve multi-objective problems, researchers must face the trade-off between keeping the advantages of PSO, such as a lower computation cost and a fast convergence speed, and ensuring the ability of finding the true optimal front. Each algorithm may perform quite differently for different dimensional problems. For example, MOPSO [Coello Coello and Lechuga, 2002] and MOPSO-CD [Raquel and Naval, 2005] are easily trapped in multi-modal problems. [Hu et al., 2003] showed interesting results in two-dimensional problems (ZDT series), but they did not run the simulation for higher dimensional problems, [Zhang and Mahfouf, 2006] proposed a modified PSO called nMPSO which extends PSO using a 'momentum term' and a dynamic weighted aggregation, and the simulation results are very competitive in Zhang's work. However, by using the weighted aggregation ranking, the algorithm 'shifts' the multi-objective problem into a single objective one.

MOPSO [Coello Coello and Lechuga, 2002] was proposed based on the storing of a grid-structure (hypercube) archive. However, in MOPSO, the hypercubes are simply used to assign the particles with density information for ranking purposes. The dividing bounds of objective space are not adjustable, and the approach helps to maintain the diversity of solutions but has no benefit for the convergence properties of the algorithm.

In this chapter, a dynamic hypercube archive (DHA) will be designed to exploit the hypercubes more efficiently. DHA participates in the whole process of the optimisation, and it is structured to save not only the density status, but also the distance information. The density and distance information will be used for global best selection and population competition (pool selection), via which the DHA can improve the algorithm's performance in terms of the convergence rate and the diversity of solutions.

An extended multi-objective PSO algorithm mPSO-DHA is proposed using dynamic hypercube archiving (DHA), mutation operator, weight adapting mechanism, enhanced global selection strategy and pool selection technique. This algorithm is designed to improve the performance of PSO in solving multi-objective problems. The dynamic hypercube archive is used via an enhanced global selection mechanism, which includes the ability of keeping the diversity of solutions, and DHA is also structured to gather the distance information for population competitions. A mutation operator is designed in order to increase the chances of jumping out from local Pareto-fronts, and weight adaptation boosts the local searching process of the algorithm. A pool selection procedure is integrated in order to increase the selection pressure. The simulation results using 2-D and

3-D benchmark problemsare presented in order to show the generality of the algorithm. In addition, comparative studies with other PSOs and evolutionary algorithms are carried out, which determines whether mPSO-DHA is competitive with other popular multi-objective optimisation algorithms.

3.3 A Multi-objective Particle Swarm Optimisation Algorithm with a Dynamic Hyper-cube Archive, Mutation and Population Competition

The proposed algorithm extends the single objective PSO algorithm to solve multi-objective problems. The new approach introduces a dynamic hyper-cube archive (DHA) to keep the diversity of the solutions, and includes a simple mutation operator to enhance the algorithm's global search ability. A weight adapting and pool selection technique are integrated to enhance the performance of finding local optimal solutions. The algorithm is described in the following section.

3.3.1 Algorithm Structure

The algorithm structure is similar to the original PSO. Before the process starts: initialise the position of all particles X_i in the population POP randomly in the search space; and initialise all velocities of particles V_i to zero. The following procedure is implemented:

1. Locate the position of particles in the population POP according to the hyper-cubes; the hyper-cube generation process will be proposed later in Section 3.3.2. Save POP with density and distance information, and save the current generation of POP to POP$_{old}$.

2. Update the velocity of ith particle using the following equation:

$$V_i^n = \begin{cases} w_i \times R_1 \times V_i^{n-1} + \\ \alpha \times [C_1 \times R_2 \times (X_{Pb_i} - X_i) + \\ C_2 \times R_3 \times (X_{Gb_i} - X_i)], \texttt{ if } n \leq 100 \\ \\ w_i \times R_1 \times V_i^{n-1} + C_1 \times R_2 \times (X_{Pb_i} - X_i) \\ + C_2 \times R_3 \times (X_{Gb_i} - X_i), \texttt{ if } n > 100 \end{cases} \tag{3.3}$$

the equations are modified from the basic PSO [Shi and Eberhart, 1998a], where w_i is the inertia weight, which applies the effect to the particles from its recent behaviour. R_1, R_2, $R_3 \in [0, 1]$ are random values, α is a control factor, C_1, C_2 are learning factors, n is the number of current generation, i is the number of current particle, and N is the maximum iterations. The global best point X_{Gb_i} is chosen from the archive. The global best point is randomly chosen from those cubes that have shorter distance to the origin-cube than the cube that X_i is currently in. The Pseudo-code for global best selection will be shown in Section 3.3.6.

The velocity limitation V^{max} is set to $0.2 \cdot (X^{uplim} - X^{lowlim})$, where X^{uplim}, X^{lowlim} are the upper and lower bound of the objective values. When $n \leq 100$, the use of $\alpha = n/100$ is to keep the "Personality" of particles, which at the very beginning of the optimisation weakens the effects from other particles, then the accelerate factor increases so that the particle can be guided in the later stage of the optimisation. Also, it should be noticed that the inertia weight w_i is dynamically changing through the whole process. The mechanism of weight changing will be discussed in Section 3.3.3.

3. Update the position of all particles in POP using the following equation:

$$\begin{cases} X_i^n = X_i^{n-1} + V_i^n, \\ \\ \text{IF } X_i^n > X^{uplim}, \text{THEN } X_i^n = X^{uplim}, V_i^n = -V_i^n, \\ \text{ELSEIF } X_i^n < X^{lowlim}, \text{THEN } X_i^n = X^{lowlim}, V_i^n = -V_i^n. \end{cases} \tag{3.4}$$

4. Carry-out the mutation process with designed mechanism in Section 3.3.4 and a predefined mutation probability P_m;

5. Evaluate the objective values of all particles;

6. For each particle, update the particle best point if the old one is dominated by the current result;

7. Store all non-dominated particles to a global result storage G_{store}, delete dominated ones, and replace points in the higher dense cubes when the size of G_{store} reaches the store limitation - the Pseudo-code for archive replacing will be discussed later in Section 3.3.2.

8. Merge POP and POP$_{old}$ to a new set of particles pool. Decide whether the pool selection should be started based on the mechanism in Section 3.3.5, if yes, choose particles from pool until generate a new POP.

9. Go to step 1 for the next iteration if $n < N$.

3.3.2 The Dynamic Hyper-cube Archiving and Its Modifications

In order to identify the density of solutions in the objective space, and to make the ranking process simpler, the hyper-cube archiving technique is used in this algorithm. The objective space will be divided into a series of hyper-cubes, the procedure of hyper-cube generation is as follows:

1. Divide every dimension of the explored objective space equally into M parts, where M is a predefined value. The whole objective will then be divided into M^d hyper-cubes, d is the number of dimensions in objective space. The upper and lower bounds of the explored objective space are updated through the following mechanism: for every dimension of objective value, set the upper limit to the maximum of both the population and archive, then change the lower bound to the minimum of population only when the minimum of population is less than the current lower bound.

2. For each one of the particles, identify the hyper-cube that contains the particle.

3. For those hyper-cubes which are not empty, define *Density* as the number of particles it contains. Then, assign a coordinate that represents its position, for example, in a 2-D problem where the space is divided into 6^2 hyper-cubes, the coordinate for hyper-cube A in Fig. 3.1 is (3,1), and hyper-cube B's coordinate is (0,0). Calculate the euclidean distance *Distance* to the origin-cube according to the current coordinate system.

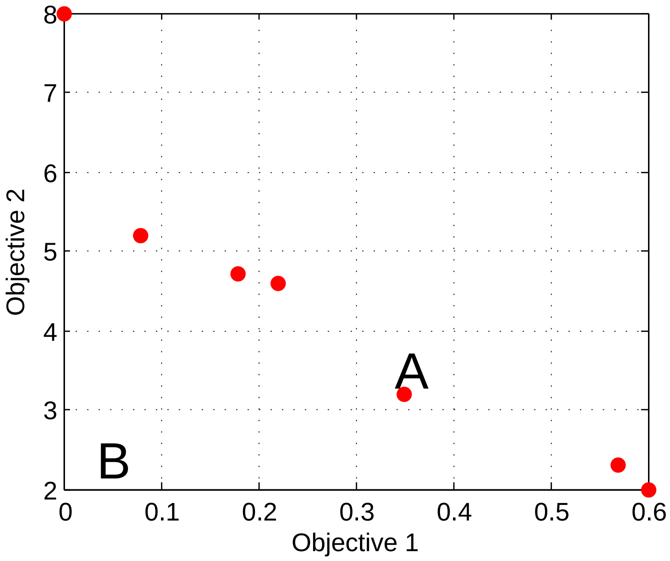


Figure 3.1: Hyper-cube Coordinating

4. Structure the archive storage as (*Particles*, *Density*, *Distance*), save the particles with the corresponding density and distance information.

When the archive reaches the predefined size limit, the new non-dominated particles will replace the particles in the archive through Algorithm 3.1, where RAND(A) means selecting a particle randomly from particle set A.

Algorithm 3.1 Pseudo-code of archive maintaining

```
$P_{\text{new}}$ = new non-dominated particle
$G_{\text{dens}}$ = particles in $G_{store}$ with largest $Density$
$G_{\text{far}}$ = particles in $G_{store}$ with largest $Distance$
$Density_{\max} = \max(Density)$
if $G_{\text{dens}} \cap G_{\text{far}} \neq \emptyset$ then
  $\text{RAND}(G_{\text{dens}} \cap G_{\text{far}}) = P_{\text{new}}$
else
  if $Density_{\max} = 1$ then
    $\text{RAND}(G_{\text{far}}) = P_{\text{new}}$
  else
    $\text{RAND}(G_{\text{dens}}) = P_{\text{new}}$
  end if
end if
```

3.3.3 The Weight Adaptation

The purpose of designing the weight adaptation mechanism is to enhance the particle's local searching ability. During the non-dominated sorting and hypercube archiving process, every particle in the population will be given a flag F, the flag is an indicator of whether the particle explored better solutions. Assuming that $F = 1$ represents the better solution found, $F = 0$ means no particle is dominated by the current particle and the current particle is not dominated by any particle in the archive, and $F = -1$ shows current particle is dominated by particles in the archive. The weight changing mechanism can be defined as follows:

$$w_i^n = \begin{cases} \min(2, w_i^{n-1} \times 2), & \text{if } F = 1 \\ w_i^{n-1}, & \text{if } F = 0 \\ \max(0.125, w_i^{n-1}/2), & \text{if } F = -1 \end{cases} \tag{3.5}$$

If the current particle dominated any of the particles in the archive, other better solutions may be found in the future in the direction of the current particle's velocity. To keep the direction of the current state, the inertia weight w_i is multiplied by 2. Vice-versa, when the current particle is dominated by any particle in the archive, it is undesirable for the particle to move towards the current direction, hence, w_i will be decreased. As the initial w_i is set to 0.5, the scaling range of w_i is limited to 4 times bigger or smaller of the initial value, which will prevent the particles from moving erratically.

3.3.4 The Mutation Operator

A mutation operator is used to maintain the global search ability. The mutation rate is independent from the iteration number so that the algorithm can jump out of the local optima all the time in multi-modal problems. When the changes of the whole archive tend to decrease, the mutation process will begin.

The degree of changes Q^n in the archive is defined as $Q^n = \hat{S}^{n-1}/S^{n-1}$. Where n is the index of iterations; S^n is the average shortest distance from current archive to the previous one, this distance is computed after the shortest distance from every point to the points in previous archive is determined; $\hat{S}^n$ is the standard deviation of S^n in the last five generations. $\hat{S}^n$ and S^n can be calculated using the following equations, where i and j mean the ith and jth particle in the archive, and G is the particle in the archive.

$$\begin{aligned} S^n &= \text{mean}_j(\min_i \sum_d (G_j^d - G_i^d)^2) \\ \hat{S}^n &= \text{STD}_{n-5}^n S^n \end{aligned} \tag{3.6}$$

When $Q^n < 0.2$, which means that the change of the archive tends to be small, the mutation will start. For every particle in the population, if Rand(1) $< P_m$, X being the objective value of current particle in the population, and d being randomly chosen from all the variable dimensions, the mutation may be defined as follows:

$$X_d^n = X_d^{n-1} + (X_d^{uplim} - X_d^{lowlim}) \times 0.25 \times (0.5 - \text{Rand}(1)) \tag{3.7}$$

3.3.5 Pool Selection

A pool selection technique is introduced to increase the selection pressure of the original PSO. It is a simple competition mechanism between the particles in the current population and the old population. Similar to the shortest average distance from archive to archive in the mutation operation, the shortest average distance from population to population is used here to determine when to start the pool selection process. Assume n being the index of the iterations, N being the maximum number of iterations, and Q_p^n being the variation of population, when $n < N - 200$ and $Q_p^n > 0.1$, which means the change of particle positions tends to be large, the whole population may be misleading and may run randomly in the search space. In this case, the current population POP will be merged with the previous population POP_{old}, and the particles with the shorter distance to the origin-cube (the origin-cube is defined through hyper-cube generation, not the regular zero point) will be selected as the new population. The condition $n < N - 200$ keeps the diversity of solutions at the end stage of the optimisation. The variation of population Q_p^n is defined as $Q_p^n = \hat{S}_p^{n-1} / S_p^{n-1}$, where,

$$\begin{aligned} S_p^n &= \text{mean}_j (\min_i \sum_d (X_j^d - X_i^d)^2) \\ \hat{S}_p^n &= \text{STD}_{n-5}^n S_p^n \end{aligned} \tag{3.8}$$

3.3.6 Global Best Selection

Based on the cube density and distance information which can be acquired from hyper-cube generation process, the global best guide selection algorithm for each particle will be selected through the following pseudo-code in Algorithm-3.2.

It should be noted that in the last 200 generations of the optimisation, the global best particle will be selected both from the nearest and farthest particles, this is to keep the diversity of the solutions.

Algorithm 3.2 Pseudo-code of Global Best Particle Selection

```
$n$ is the index of the iterations, and $N$ is the maximum number of iterations.
$X_{Gb_i}$ = global best point for $X_i$
$G_s$ = particles in $G_{store}$ with smaller $Distance$ than $X_i$
$G_{\text{near}}$ = particles in $G_{store}$ with smallest $Distance$
$G_{\text{far}}$ = particles in $G_{store}$ with largest $Distance$
$G_{\text{sparse}}$ = particles in $G_{store}$ with smallest $Density$
if $G_s = \emptyset$ then
    $X_{Gb_i} = \text{RAND}(G_{sparse})$
else
    if $n > N - 200$ then
        $X_{Gb_i} = \text{RAND}(G_{\text{near}} \cup G_{\text{far}})$
    else
        $X_{Gb_i} = \text{RAND}(G_{\text{near}})$
    end if
end if
```

3.4 Experimental Studies using mPSO-DHA

In this part, the modifications and improvements are tested using different parameter settings in order to determine the best parameters and performances. The analyses are based on the experiments using different benchmark functions under different settings of population sizes, cube numbers and mutation rates. Two performance metrics GD and Δ [Deb, 2008] are calculated to provide an intuitive image of the effects of different parameters.

3.4.1 Benchmark Functions

The mPSO-DHA algorithm is developed to improve the multi-objective optimisation performance of the original PSO. A series of multi-objective benchmark functions are selected as they are designed to test the algorithm in different situations, such as multi-modality and separability. The well-known benchmark functions ZDT series (2-dimensional, listed Eqs. 3.9 - 3.12) and DTLZ series (3-dimensional, listed Eqs. 3.13 - 3.16) in [Deb et al., 2002; Zitzler et al., 2000] are used in the experimental studies, and Figs. 3.2 and 3.3 plot the Pareto-fronts of the benchmark functions.

1. Benchmark function – ZDT1 (concave)

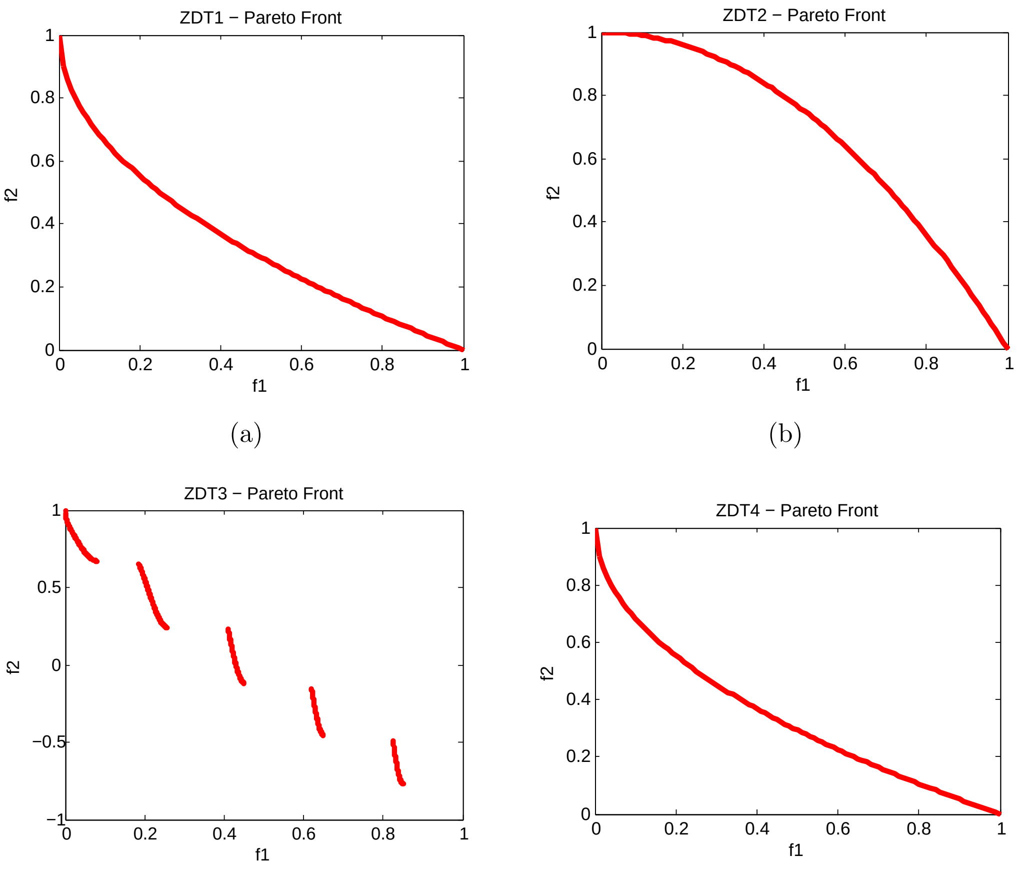
$$\begin{aligned}
&\text{Minimize } (f_1, f_2)\text{, where:}\\
&f_1(\mathbf{x}) = x_1\\
&f_2(\mathbf{x}) = g(\mathbf{x}) \cdot (1.0 - \sqrt{x_1/g(\mathbf{x})})\\
&g(\mathbf{x}) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i\\
&\text{subject to } 0 \leq x_i \leq 1, \text{ for } i = 1, \cdots, n
\end{aligned} \tag{3.9}$$

2. Benchmark function – ZDT2 (convex)

$$\begin{aligned}
&\text{Minimize } (f_1, f_2)\text{, where:}\\
&f_1(\mathbf{x}) = x_1\\
&f_2(\mathbf{x}) = g(\mathbf{x}) \cdot (1.0 - (x_1/g(\mathbf{x}))^2)\\
&g(\mathbf{x}) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i\\
&\text{subject to } 0 \leq x_i \leq 1, \text{ for } i = 1, \cdots, n
\end{aligned} \tag{3.10}$$

3. Benchmark function – ZDT3 (disconnected)

$$\begin{aligned}
&\text{Minimize } (f_1, f_2)\text{, where:}\\
&f_1(\mathbf{x}) = x_1\\
&f_2(\mathbf{x}) = g(\mathbf{x}) \cdot (1.0 - \sqrt{x_1/g(\mathbf{x})} - (x_1/g(\mathbf{x}))\sin(10\pi x_1))\\
&g(\mathbf{x}) = 1.0 + \frac{9}{n-1}\sum_{i=2}^{n} x_i\\
&\text{subject to } 0 \leq x_i \leq 1, \text{ for } i = 1, \cdots, n
\end{aligned} \tag{3.11}$$



(a)

(b)

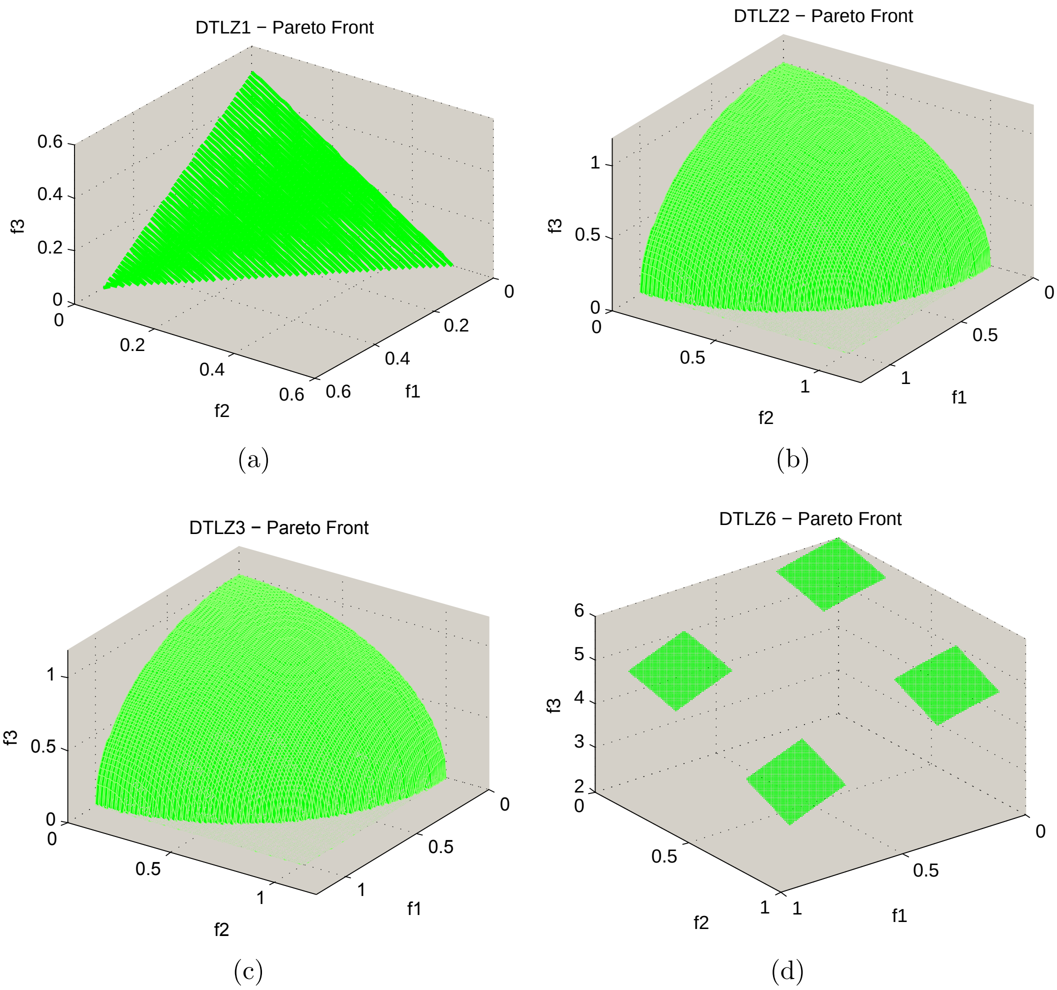
(c)

(d)

Figure 3.2: Pareto-front for ZDT series

4. Benchmark function – ZDT4 (multi-modal)

$$
\begin{aligned}
&\text{Minimize } (f_1, f_2), \text{ where:}\\
&f_1(\mathbf{x}) = x_1\\
&f_2(\mathbf{x}) = g(\mathbf{x}) \cdot (1.0 - (x_1/g(\mathbf{x})^2))\\
&g(\mathbf{x}) = 1.0 + 10(n-1) + \sum_{i=2}^{n}(x_i^2 - 10\cos(4\pi x_i))\\
&\text{subject to } 0 \le x_1 \le 1, -5 \le x_i \le 5, \;\text{ for } i = 1, \cdots, n
\end{aligned}
\tag{3.12}
$$



(a)

(b)

(c)

(d)

Figure 3.3: Pareto-front for DTLZ series

5. Benchmark function – DTLZ1 (multi-modal)

$$
\begin{aligned}
&\text{Minimize } (f_1, f_2, f_3), \text{ where:}\\
&f_1(\mathbf{x}) = \tfrac{1}{2}x_1x_2(1+g(\mathbf{x}_M))\\
&f_2(\mathbf{x}) = \tfrac{1}{2}x_1(1-x_2)(1+g(\mathbf{x}_M))\\
&f_3(\mathbf{x}) = \tfrac{1}{2}(1-x_1)(1+g(\mathbf{x}_M))\\
&\mathbf{x}_M = (x_3, \cdots, x_n)\\
&g(\mathbf{x}_M) = 100(|\mathbf{x}_M| + \sum_{x_i\in\mathbf{x}_M}(x_i-0.5) - \cos(20\pi(x_i-0.5)))\\
&\text{subject to } 0 \le x_i \le 1,\ \text{ for } i = 1, \cdots, n
\end{aligned}
\tag{3.13}
$$

6. Benchmark function – DTLZ2 (spherical)

$$
\begin{aligned}
&\text{Minimize } (f_1, f_2, f_3)\text{, where:}\\
&f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\\
&f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M))\cos(x_1\pi/2)\sin(x_2\pi/2)\\
&f_3(\mathbf{x}) = (1 + g(\mathbf{x}_M))\sin(x_1\pi/2)\\
&\mathbf{x}_M = (x_3, \cdots, x_n)\\
&g(\mathbf{x}_M) = \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5)^2\\
&\text{subject to } 0 \le x_i \le 1,\ \text{ for } i = 1, \cdots, n
\end{aligned}
\tag{3.14}
$$

7. Benchmark function – DTLZ3 (complex, multi-modal)

$$
\begin{aligned}
&\text{Minimize } (f_1, f_2, f_3)\text{, where:}\\
&f_1(\mathbf{x}) = (1 + g(\mathbf{x}_M))\cos(x_1\pi/2)\cos(x_2\pi/2)\\
&f_2(\mathbf{x}) = (1 + g(\mathbf{x}_M))\cos(x_1\pi/2)\sin(x_2\pi/2)\\
&f_3(\mathbf{x}) = (1 + g(\mathbf{x}_M))\sin(x_1\pi/2)\\
&\mathbf{x}_M = (x_3, \cdots, x_n)\\
&g(\mathbf{x}_M) = 100(|\mathbf{x}_M| + \sum_{x_i \in \mathbf{x}_M} (x_i - 0.5) - \cos(20\pi(x_i - 0.5)))\\
&\text{subject to } 0 \le x_i \le 1,\ \text{ for } i = 1, \cdots, n
\end{aligned}
\tag{3.15}
$$

8. Benchmark function – DTLZ6 (disconnected)

$$
\begin{aligned}
&\text{Minimize } (f_1, f_2, f_3)\text{, where:}\\
&f_1(\mathbf{x}) = x_1, f_2(\mathbf{x}) = x_2\\
&f_3(\mathbf{x}) = h(\mathbf{x}_M)(1 + h(\mathbf{f}, g))\\
&\mathbf{x}_M = (x_3, \cdots, x_n)\\
&g(\mathbf{x}_M) = 1 + \tfrac{9}{n-2} \sum_{x_i \in \mathbf{x}_M} (x_i)\\
&h(\mathbf{f}, g) = 3 - \sum_{i=1}^{2} (\tfrac{f_i}{1+g}(1 + \sin(3\pi f_i)))\\
&\text{subject to } 0 \le x_i \le 1,\ \text{ for } i = 1, \cdots, n
\end{aligned}
\tag{3.16}
$$

3.4.2 The Metrics of Algorithm Performance

Also, it is necessary to compare the performance of mPSO-DHA to other multi-objective optimisation algorithms. Hence, GD and Δ in [Deb, 2008] provide the ideal approach to summarise the features of different algorithms both on convergence and diversity.

The metric GD is a value that measures the distance from the solutions to the true Pareto-front of the problem. With this metric, the characteristic of converging speed can be easily obtained. GD is defined as follows [Deb, 2008]:

$$GD = \frac{(\sum_{i=1}^{|Q|} d_i^m)^{1/m}}{|Q|} \tag{3.17}$$

where m is the number of objectives, Q is the obtained Pareto solution set by the algorithm, d_i is the Euclidean distance from the solution $i \in Q$ to the nearest member of the true Pareto set P, a set of $|P| = 500$ uniformly distributed Pareto-optimal solutions are selected.

Δ is a metric that describes how well is the distribution of solutions, so that the diversity of solutions can be measured. Δ is defined as follows [Deb, 2008]:

$$\Delta = \frac{\sum_{m=1}^{M} d_m^\varepsilon + \sum_{i=1}^{|Q|} \left| d_i - \bar{d} \right|}{\sum_{m=1}^{M} d_m^\varepsilon + |Q| \, \bar{d}} \tag{3.18}$$

where d_i is the distance between the neighbouring solutions in the Pareto solution set Q. $\bar{d}$ is the mean value of all d_i. d_m^ε is the distance from the extreme solutions of P and Q along the mth objective.

3.4.3 The Effect of Swarm Sizes

An experiment is designed to test the mPSO-DHA algorithm under different settings of swam sizes (5, 10, 25, 50, 100) using the ZDT series benchmark functions in [Zitzler et al., 2000]. The other parameter are set as in Table 3.1.

The results in Table 3.2 are generated from an average of 10 independent runs under every different population settings. It is shown that for ZDT1, ZDT2 and ZDT3, the convergence measurement GD reaches the best (highlighted in red color) when the swam size is set to 25. For ZDT4, all settings have failed to converge. Swam sizes 5, 10 and 25 result in the same level of GD, but the value 10 performs slightly better than other values of swam sizes.

The diversity measure Δ becomes the smallest when the swam size is set to 25 for the problem ZDT2, ZDT3 and ZDT4. The Δ's for ZDT1 are very similar that they are with little variance, and the best Δ for ZDT1 is obtained using the swam size of 10.

Table 3.1: Parameter settings - effect of swam sizes

Archive size	M hypercubes	w_i	C_1	C_2	α	P_m	Evaluation times
100	30	Adaptive	1	2	0.5	0.5	25000

3.4.4 The Effect of Cube Numbers

This part provided the results of the experiment using different values of cube numbers, in order to analyse the effect of cube number settings in mPSO-DHA. Cube numbers (10, 30, 50) are used in the experiments, and other parameters are set as in Table 3.3. All the results are calculated based on the results of 10 independent runs for each cube number setting.

It can be concluded from the results in Table 3.4 that for simpler problems ZDT1, ZDT2 and ZDT3, the cube number 30 leads to better results in the comparison of the convergence measures GD, and cube numbers 30 and 50 lead to similar results in convergence measure.

Table 3.2: Test results under different swam size settings

Swam size		5	10	25	50	100
ZDT1	GD	1.184E-3 (3.925E-4)	8.103E-4 (4.584E-5)	7.722E-4 (5.927E-5)	1.126E-3 (2.579E-4)	3.240E-3 (3.262E-4)
	Δ	4.970E-1 (2.495E-2)	4.929E-1 (1.973E-2)	5.271E-1 (3.760E-2)	4.937E-1 (2.521E-2)	5.244E-1 (3.254E-2)
ZDT2	GD	2.584E-2 (6.216E-4)	7.124E-2 (9.188E-2)	3.088E-4 (4.249E-5)	3.960E-4 (8.173E-5)	5.318E-4 (3.467E-4)
	Δ	6.111E-1 (3.797E-2)	5.227E-1 (3.195E-2)	5.091E-1 (1.850E-2)	5.099E-1 (4.190E-2)	7.585E-1 (2.234E-1)
ZDT3	GD	1.099E-2 (2.056E-2)	6.267E-4 (2.472E-4)	5.807E-4 (2.542E-5)	8.231E-4 (1.025E-4)	2.160E-3 (4.800E-4)
	Δ	9.107E-1 (1.770E-1)	9.491E-1 (1.189E-1)	7.078E-1 (4.574E-3)	8.456E-1 (8.476E-2)	8.207E-1 (2.328E-2)
ZDT4	GD	2.934E-1 (3.380E-1)	1.030E-1 (1.248E-1)	2.059E-1 (1.523E-1)	1.026 (8.908E-1)	2.462 (2.077)
	Δ	1.470 (2.353E-1)	1.280 (4.149E-1)	9.548E-1 (4.876E-1)	1.629 (1.658E-1)	1.194 (2.527E-1)

Table 3.3: Parameter settings - effect of cube numbers

Archive size	Swam size	w_i	C_1	C_2	α	P_m	Evaluation times
100	25	Adaptive	1	2	0.5	0.5	25000

The results for ZDT1 and ZDT2 plot a trend suggests that a bigger cube number leads to a better ability to keep the diversity of the solutions. However, in the tests for ZDT3 and ZDT4, this trend is not proved. It is shown that better performances for ZDT3 and ZDT4 require a cube number that is neither too big nor too large. The reasons for this are, that for ZDT3, the Pareto-front is disconnected. A too big or small value of cube number may cause the algorithm to exert more efforts on the disconnected part of the objective space; for ZDT4, the objective space are large scale and the Pareto-front is multi-modal, hence a cube number value that is too big or small will lead to the outcome that the solutions spread over different local optimal fronts.

Table 3.4: Test results using different cube numbers

Cube Number		10*10	30*30	50*50
ZDT1	*GD*	8.509E-4 (2.123E-4)	7.722E-4 (5.927E-5)	8.952E-4 (1.427E-4)
	Δ	6.304E-1 (1.515E-2)	5.271E-1 (3.760E-2)	4.429E-1 (1.528E-2)
ZDT2	*GD*	3.838E-4 (1.098E-4)	3.088E-4 (4.249E-5)	3.865E-4 (4.822E-5)
	Δ	6.703E-1 (4.755E-2)	5.091E-1 (1.850E-2)	4.653E-1 (4.884E-2)
ZDT3	*GD*	6.473E-4 (1.192E-4)	5.807E-4 (2.542E-5)	5.914E-4 (1.693E-4)
	Δ	9.949E-1 (1.718E-1)	7.078E-1 (4.574E-3)	8.707E-1 (2.846E-2)
ZDT4	*GD*	7.172E-1 (9.513E-1)	2.059E-1 (1.523E-1)	2.045E-1 (1.784E-1)
	Δ	1.374 (3.054E-1)	9.548E-1 (4.876E-1)	1.670 (2.231E-1)

3.4.5 The Effect of Mutation Rates

Another experiment is carried out in this part to determine the effect of different mutation rates. Mutation rates (0.1, 0.3, 0.5, 0.7, 0.9) are used in the experiments, and other parameters are set as in Table 3.5. All results are calculated based on the results of 10 independent runs after changing to different mutation rates.

Table 3.6 shows the test results using different mutation rates. From the results of ZDT1 and ZDT2, it can be seen that *GD* increases when the mutation rate becomes large. The Δ measure, in contrast, is decreasing at the same time.

Table 3.5: Parameter settings - effect of mutation rates

Archive size	Swam size	w_i	C_1	C_2	α	M	Evaluation times
100	25	Adaptive	1	2	0.5	30	25000

Table 3.6: Test results using different mutation rates

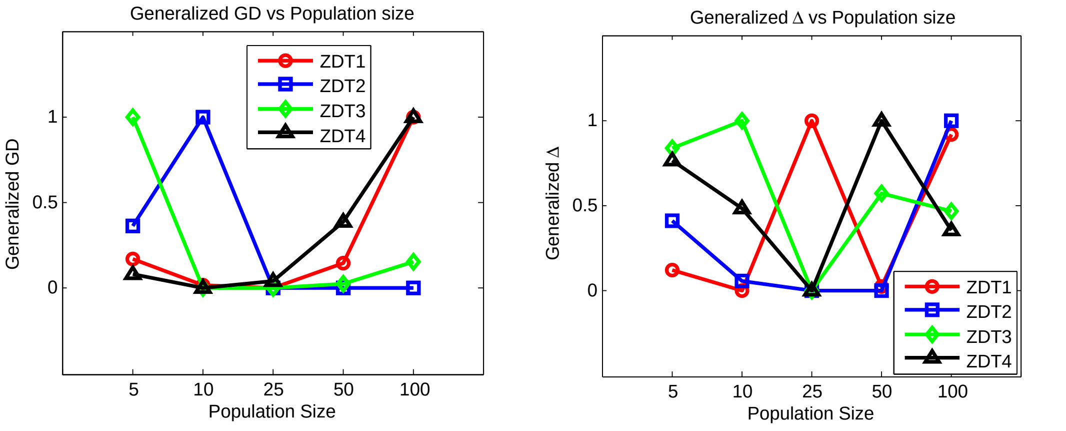
Mutation rate		0.1	0.3	0.5	0.7	0.9
ZDT1	GD	4.503E-4 (1.136E-4)	7.330E-4 (1.027E-4)	7.722E-4 (5.927E-5)	1.134E-3 (2.009E-4)	1.580E-3 (2.040E-4)
	Δ	5.504E-1 (3.748E-2)	5.291E-1 (3.685E-2)	5.271E-1 (3.760E-2)	4.853E-1 (3.593E-2)	4.901E-1 (2.143E-2)
ZDT2	GD	1.783E-4 (3.331E-5)	2.416E-4 (3.974E-5)	3.088E-4 (4.249E-5)	4.505E-4 (4.029E-5)	6.990E-4 (1.395E-4)
	Δ	6.179E-1 (4.066E-2)	5.744E-1 (1.374E-2)	5.091E-1 (1.850E-2)	5.142E-1 (2.345E-2)	4.890E-1 (1.549E-2)
ZDT3	GD	5.102E-4 (3.396E-5)	6.101E-4 (9.101E-5)	5.807E-4 (2.542E-5)	7.408E-4 (1.599E-4)	7.627E-4 (2.652E-5)
	Δ	8.587E-1 (2.645E-2)	8.733E-1 (7.289E-2)	7.078E-1 (4.574E-3)	8.609E-1 (4.753E-2)	9.025E-1 (1.627E-1)
ZDT4	GD	2.083E-1 (1.108E-1)	2.321E-1 3.458E-1	2.059E-1 (1.523E-1)	4.804E-1 (1.934E-1)	2.172E-1 (2.809E-1)
	Δ	1.552 (3.193E-1)	1.463 (2.886E-1)	9.548E-1 (4.876E-1)	1.779 (1.250E-2)	1.513 (2.935E-1)

It can be concluded that for simpler problems, a bigger mutation rate will lead to a slower convergence speed and a better distribution of the solutions.

However, the aforementioned conclusion does not apply to the problems ZDT3 and ZDT4, the results of which indicate that the solutions for ZDT3 are closest to the Pareto-front when the mutation rate is set to 0.1, and have a better distribution when the rate is set to 0.5. The best solutions for ZDT4 are found with the mutation rate of 0.5.

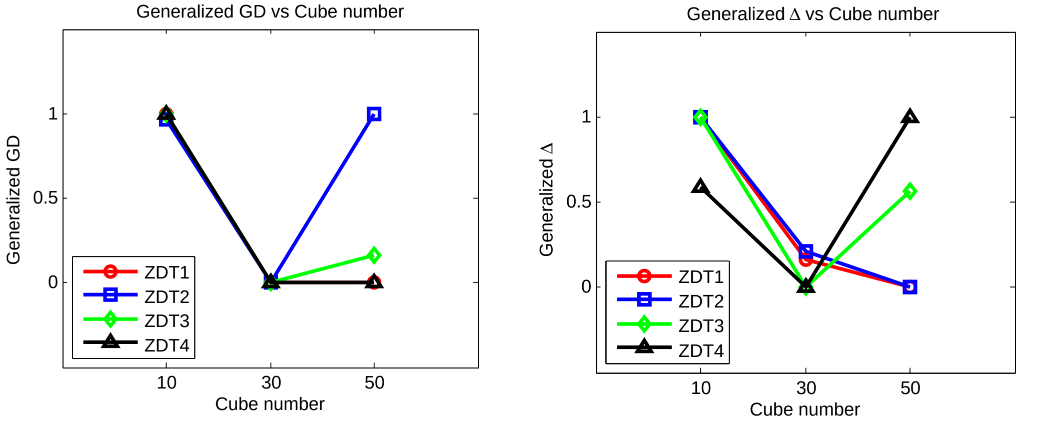
3.4.6 Analysis of Experimental Results

The final tuned parameters can be set at last after the experimental studies. Fig. 3.4 plots the curve of the GD and Δ changing using different population settings. The effect of cube numbers is shown in Fig. 3.5, and Fig. 3.6 draws the curve when the mutation rate is changing.



(a) (b)

Figure 3.4: Generalized GD and Δ changes using different population settings



(a) (b)

Figure 3.5: Generalized GD and Δ changes using different cube nubmers

It can be clearly concluded that the swam size 25 performed better in most of the experiments. In the tests of cube numbers, the value 30 outperformed

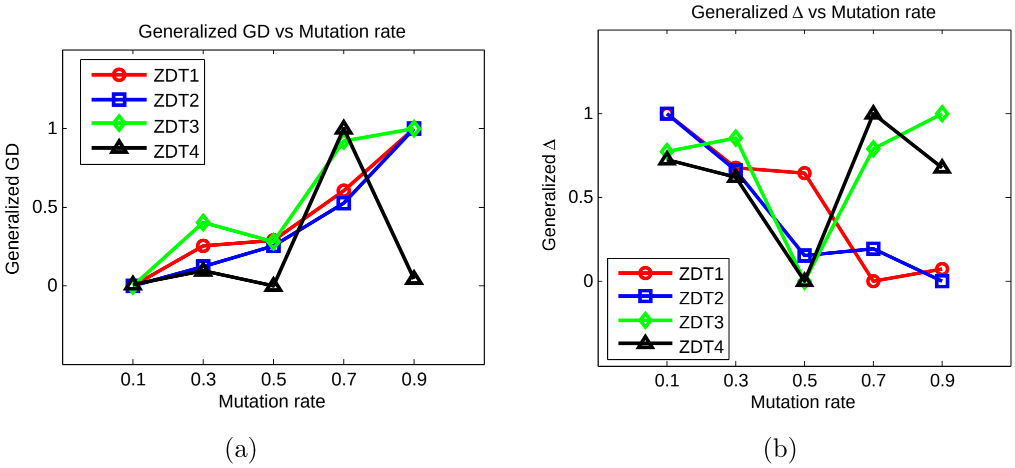


Figure 3.6: Generalized GD and Δ changes using different mutation rates

the others. Setting the mutation rate to 0.5 can keep the balance between the convergence speed of the algorithm and the diversity of the solutions.

These values, i.e. swam size 25, cube number 30, and mutation rate 0.5, will be used in the following comparative studies with other algorithms.

3.5 Comparisons with Other Multi-objective Particle Swarm Optimisation Algorithms

The initial design intent of mPSO-DHA is to ehance the multi-objective optimisation performence of PSO, hence, a comparitive study with other multi-objective PSO algorithms is necessary. Therefore, two widely used multi-objective PSO algorithms MOPSO [Coello Coello and Lechuga, 2002] and MOPSO-CD [Raquel and Naval, 2005] are compared with mPSO-DHA in this section.

3.5.1 Performance in Two-objective Benchmark Functions

The tests were first carried out on the well-known test functions presented by [Zitzler et al., 2000] (ZDT Series).

3.5.1.1 Test function - ZDT1

ZDT1 is a simple concave test function. In this experiment, the number of variables n was set to 30, and the parameters are set as in Table 3.7. The parameters for MOPSO and MOPSO-CD are the same as they were in [Coello Coello and Lechuga, 2002] and [Raquel and Naval, 2005]. The maximum generation is adjusted so that the maximum evaluation times for the three algorithms are all set to 25000. Also, it should be noted that the mutation operator in MOPSO-CD is different from the one in mPSO-DHA.

Table 3.7: Parameter settings for ZDT1

	MOPSO	MOPSO-CD	mPSO-DHA
Swam size	40	100	25
Archive size	200	500	100
M hypercubes	30	N/A	30
w_i	0.4	0.4	Adaptive
C_1	1	1	1
C_2	1	1	2
α	N/A	N/A	0.5
P_m	N/A	0.5	0.5
N	625	250	1000

The simulation results being shown in Fig. 3.7, it can be seen that the solutions found by mPSO-DHA are better than both the other two PSOs. MOPSO, and MOPSO-CD even failed to find the true Pareto-front after 25000 evaluation times.

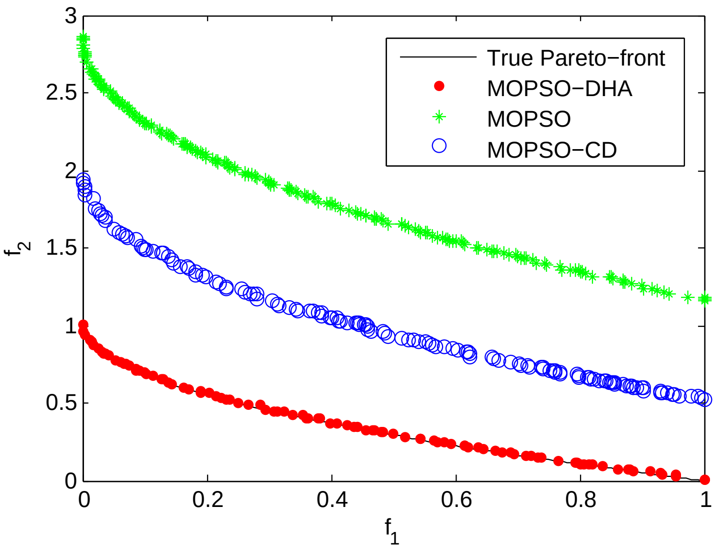


Figure 3.7: Simulation Results For ZDT1

3.5.1.2 Test function - ZDT2

ZDT2 is a simple convex test function, and the number of variables is set to $n = 30$. The parameters for this simulation are similar to those for ZDT1. The maximum evaluation time is set to 25000 as well. The solutions found by the three algorithms are shown in Fig. 3.8, where it was found that mPSO-DHA outperformed the other two algorithms. MOPSO is not able to keep the diversity of the solutions and only one point (0,3.2) is shown in Fig. 3.8.

3.5.1.3 Test function - ZDT3

The difficulty of ZDT3 is that it has disconnected Pareto-fronts, and here $n = 30$ was chosen. The parameters for this simulation are similar to those of ZDT1. The maximum evaluation time is still no more than 25000. According to the simulation results presented in Fig. 3.9. mPSO-DHA found the true five parts of Pareto-front while the other two algorithms did not find the real optimal fronts.

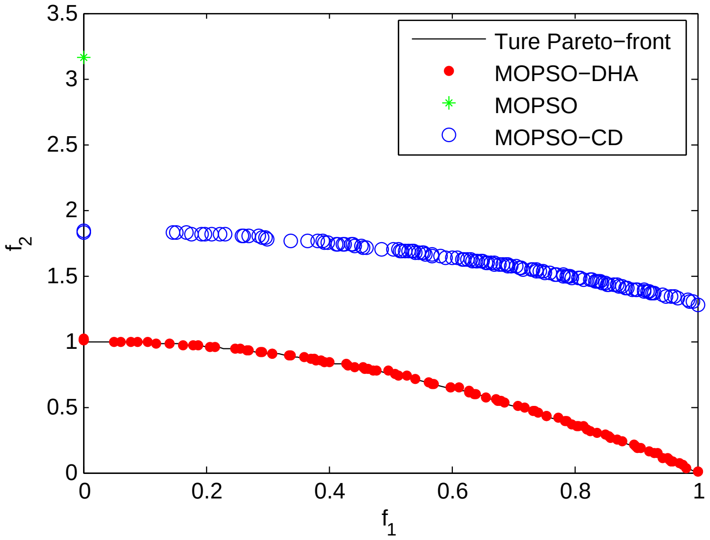


Figure 3.8: Simulation Results For ZDT2

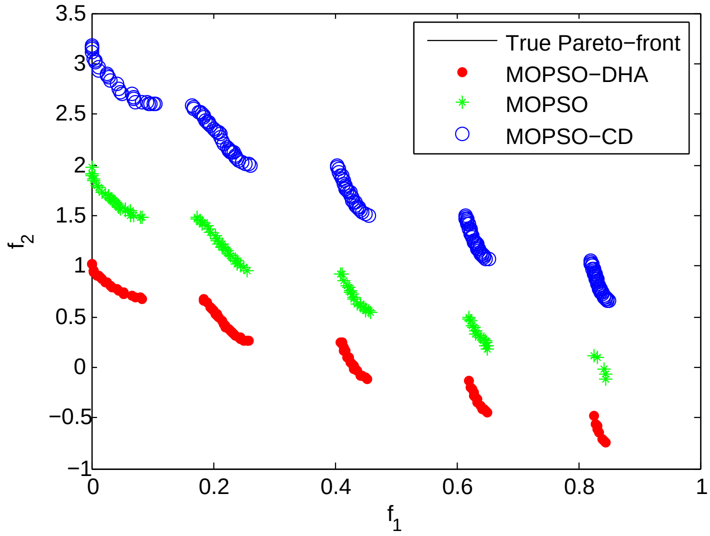


Figure 3.9: Simulation Results For ZDT3

3.5.1.4 Test function - ZDT4

ZDT4 has 21^9 local Pareto-fronts. These local optima represent a real challenge to the algorithms' ability of solving a multi-modal problem; $n = 10$ is chosen in this problem. The parameters for this simulation are similar to those of ZDT1, except

N is set to 1875 for MOPSO, 750 for MOPSO-CD and 3000 for mPSO-DHA, so that the maximum evaluation time is 75000 for all the three algorithms. The results are plotted in Fig. 3.10; it can be seen that mPSO-DHA found the true Pareto-front, and the other two algorithms failed to find the optimal-solutions. MOPSO has even converged to only one solution (0,7). This result shows that mPSO-DHA has the ability to solve multi-modal problems and can avoid local Pareto-fronts.

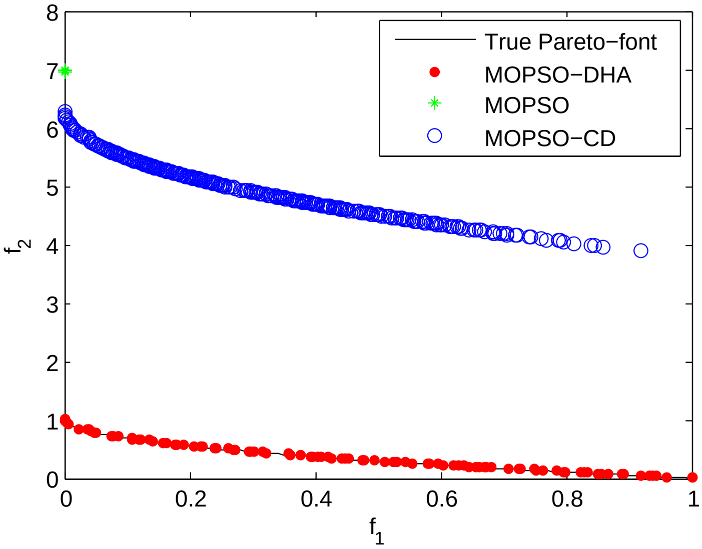


Figure 3.10: Simulation Results For ZDT4

3.5.2 Performance Assessment using Three-objective Benchmark Functions

Because problems in the real world would almost certainly not always be of a low dimensional nature, it is important to test the algorithm under more challenging conditions which involve more objectives to be optimised. [Deb et al., 2002] proposed a series of multi-dimensional problems called DTLZ Series in 2002. Simulations under a 3-D situation will be conducted in the following parts:

3.5.2.1 Test function - DTLZ1

DTLZ1 is a multi-modal test function that has $(11^5 - 1)$ local Pareto-fronts, and the number of variables $n = 7$ is chosen in this problem as suggested in [Deb et al., 2002]. The parameters for this simulation is set to be the same as those for ZDT4, and the maximum evaluation time is 75000 for all three algorithms. As shown in Fig. 3.11, mPSO-DHA found the true Pareto-front, while the other two algorithms failed to find the optimal-solutions. This result shows that mPSO-DHA has the ability of solving multi-dimensional optimising problems and the performance is better among the three modified PSOs. The scaled results for mPSO-DHA can be found in Fig. 3.12.

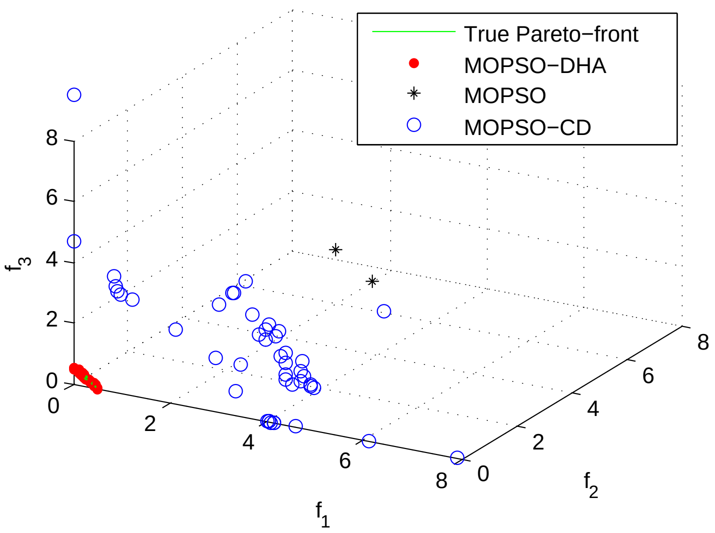


Figure 3.11: Simulation Results For DTLZ1

3.5.2.2 Test function - DTLZ2

DTLZ2 has a spherical Pareto-optimal front, and $n = 7$ is chosen in this problem similarly to what it was in [Deb et al., 2002]. The parameters for this simulation are similar to those for ZDT1, and the maximum evaluation time is 25000 for all three algorithms. The results in Fig. 3.13 show that all three algorithms found the true Pareto-front, however, MOPSO-CD did not converge to all the Pareto-

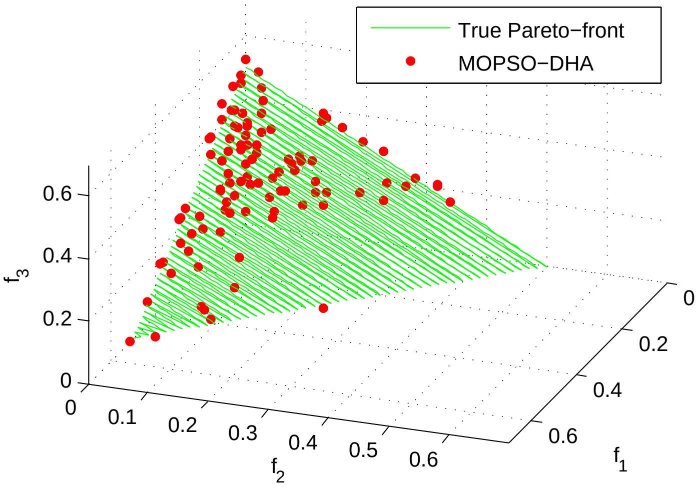


Figure 3.12: Scaled Results for DTLZ1

fronts. The solutions found by MOPSO and mPSO-DHA spread evenly across the Pareto-front surface.

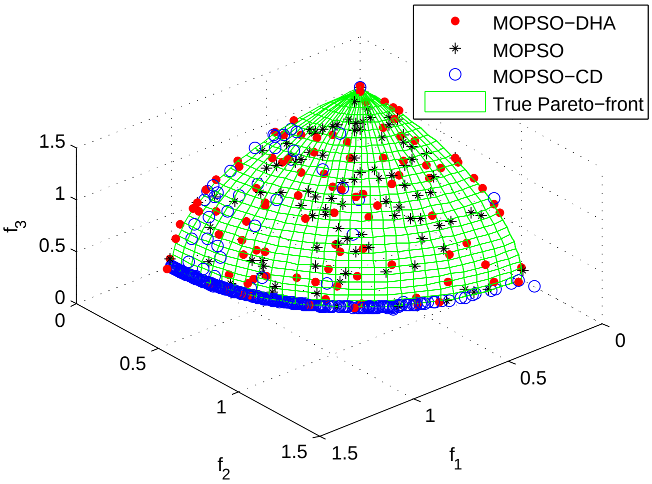


Figure 3.13: Simulation Results For DTLZ2

3.5.2.3 Test function - DTLZ3

Problem DTLZ3 is also a multi-modal test function and it is defined by replacing the $g(\mathbf{x}_M)$ function in DTLZ2 with the $g(\mathbf{x}_M)$ that has been used in DTLZ1.

Again, as suggested in [Deb et al., 2002], $n = 12$ is adopted. The parameters for this simulation are similar to those for ZDT1, N is set as 2500 for MOPSO, 1000 for MOPSO-CD and 4000 for mPSO-DHA, i.e. the maximum number of evaluation times is set to 100000 for all the three algorithms. Fig. 3.14 presents the simulation results where it can be seen that mPSO-DHA converged to the Pareto-front while MOPSO and MOPSO-CD have failed to locate the global optima. The scaled results for mPSO-DHA can be found in Fig. 3.15.

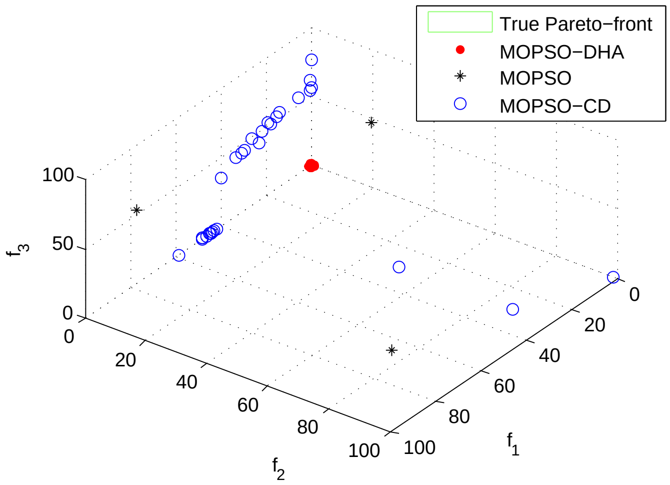


Figure 3.14: Simulation Results For DTLZ3

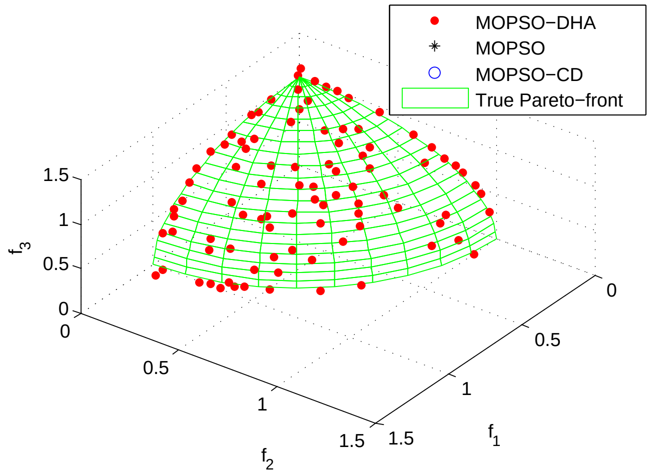


Figure 3.15: Scaled Results for DTLZ3

3.5.2.4 Test function - DTLZ6

DTLZ6 has disconnected Pareto-fronts and a dimensional factor n is set to 22 as suggested in the original paper [Deb et al., 2002]. The parameters for this simulation are similar to those of ZDT1; the maximum evaluation time is 25000 for all the algorithms. Fig. 3.16 shows the results of the simulations, where it can be seen that mPSO-DHA performance was superior compared to the other two algorithms. mPSO-DHA has found the Pareto-front and the diversity of the solutions is good, MOPSO spreads well but has failed to locate the Pareto-front. MOPSO-CD has neither found the real optima nor has it distributed the solutions evenly in the objective space.

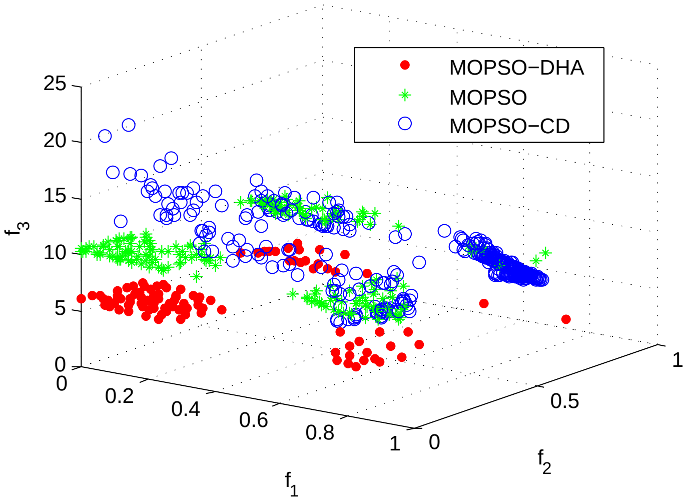


Figure 3.16: Simulation Results For DTLZ6

3.5.3 Measures of the Solutions

In order to provide a brief idea of the performance of mPSO-DHA, the measurement GD for convergence and Δ for diversity are calculated for the ZDT test functions in Table 3.8 using all three algorithms. The values are computed from the average of 10 independent runs, where the maximum evaluation time is set to 25000.

Table 3.8: GD and Δ for ZDT Series Test Functions

Function	Algorithm	Mean value of GD (variance σ)	Mean value of Δ (variance σ)
ZDT1	MOPSO	7.616E-2 (4.368E-3)	7.823E-1 (5.689E-3)
	MOPSO-CD	7.074E-2 (2.020E-2)	9.929E-1 (1.614E-1)
	mPSO-DHA	7.222E-4 (5.927E-5)	5.271E-1 (3.760E-2)
ZDT2	MOPSO	9.372E-1 (1.507E-1)	1.000 (0.000)
	MOPSO-CD	9.436E-2 (8.684E-2)	9.142E-1 (1.669E-1)
	mPSO-DHA	3.088E-4 (4.249E-5)	5.091E-1 (1.850E-2)
ZDT3	MOPSO	5.872E-2 (1.279E-2)	9.001E-1 (3.110E-2)
	MOPSO-CD	5.568E-2 (1.894E-2)	9.860E-1 (7.794E-2)
	mPSO-DHA	5.807E-4 (2.542E-5)	7.078E-1 (4.574E-3)
ZDT4	MOPSO	7.885E-1 (3.756E-1)	1.211 (3.650E-1)
	MOPSO-CD	4.671E-1 (1.624E-1)	1.901 (3.524E-1)
	mPSO-DHA	2.059E-1 (1.523E-1)	9.548E-1 (4.876E-1)

As shown in Table 3.8, mPSO-DHA outperformed MOPSO and MOPSO-CD in both convergence and diversity. GD and Δ for the test with ZDT1 are ideal in that small values represent a good convergence rate and a good diversity of the solutions. In the simulation of ZDT2, GD is even smaller than it is for ZDT1, which means that mPSO-DHA converged closer to the real Pareto-front. The result of ZDT3 shows a larger value for Δ, and this is because of the disconnected Pareto-front and the computation of Δ is from the whole set of solutions. The simulation of ZDT4 includes lager values of GD and Δ, this is to say that mPSO-DHA failed to converge to the Pareto-front in only 25000 evaluation times, but 2.059E-1 with a standard deviation of 1.523E-1 show that the solutions are very close to the true Pareto-front. As shown in Section 3.5.1.4, mPSO-DHA can find the real optima in less than 75000 evaluation times.

3.6 Comparison with Other Evolutionary Algorithms

The algorithm mPSO-DHA is tested in comparison with other evolutionary algorithms in this section. The compared multi-objective optimisation algorithms are selected as the Pareto Archived Evolution Strategy (PAES) [Knowles and Corne, 1999], the Strength Pareto Evolutionary Algorithm (SPEA) [Zitzler and Thiele, 1998] and the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [Deb et al., 2000]. The maximum number of evaluation time is set to 25000 for all algorithms in the following experiments.

The parameters of mPSO-DHA are set the same as in Table 3.7, the settings for other algorithms are listed as follows:

PAES: Population size 100, maximum generation 250, crossover probability 0.9 and mutation probability 0.01.

SPEA: Population size 80, external population size 20, maximum generation 250, crossover probability 0.9 and mutation probability 1/80.

NSGA-II (binary-coded): Population size 100, maximum generation 250, crossover probability 0.9 and mutation probability 1/(string-length). 30 bits were used to code each variable.

Table 3.9 lists the results for all the compared algorithms, the compared results for PAES, SPEA, and NSGA-II is obtained from [Deb, 2008].

It is shown that for all problems, the metric GD is the smallest for mPSO-DHA, which means that mPSO-DHA provides the fastest convergence speed among all four algorithms. The comparison of metric Δ suggests that NSGA-II leads to a better distribution of solutions. For the simpler problems ZDT1 and ZDT2, Δ for mPSO-DHA and NSGA-II are at the same level. However, the solutions for ZDT3 and ZDT4 found by NSGA-II are much farther from the Pareto-front than the results of mPSO-DHA, which means there is no advantage for NSGA-II in disconnected and multi-model problems.

Table 3.9: Comparison between mPSO-DHA and other evolutionary algorithms

Algorithm		mPSO-DHA	PAES	SPEA	NSGA-II
ZDT1	GD	7.722E-4 (5.927E-5)	8.210E-2 (8.680E-3)	1.250E-3 (0.000)	8.940E-4 (0.000)
	Δ	5.271E-1 (3.760E-2)	1.230 (4.840E-3)	7.300E-1 (9.070E-3)	4.630E-1 (4.160E-2)
ZDT2	GD	3.088E-4 (4.249E-5)	1.260E-1 (3.690E-2)	3.040E-3 (2.000E-5)	8.240E-4 (0.000)
	Δ	5.091E-1 (1.850E-2)	1.170 (7.680E-3)	6.780E-1 (4.480E-3)	4.350E-1 (2.460E-2)
ZDT3	GD	5.807E-4 (2.542E-5)	2.390E-2 (1.000E-5)	4.420E-2 (1.900E-5)	4.340E-2 (4.200E-5)
	Δ	7.078E-1 (4.574E-3)	7.900E-1 (1.650E-3)	6.660E-1 (6.660E-4)	5.760E-1 (5.080E-3)
ZDT4	GD	2.059E-1 (1.523E-1)	8.550E-1 (5.270E-1)	9.510 (1.130E+1)	3.230 (7.310)
	Δ	9.548E-1 (4.876E-1)	8.700E-1 (1.010E-1)	7.320E-1 (1.130E-2)	4.795E-1 (9.840E-3)

From the above comparison, it can be concluded that mPSO-DHA is competitive with other popular evolutionary algorithms, and mPSO-DHA has the advantage of solving complex multi-objective optimisation problems such as ZDT3 and ZDT4.

3.7 Summary

In this chapter, a multi-objective PSO called mPSO-DHA has been proposed, which uses dynamic hypercube archiving, a mutation operator, a weight adapting mechanism, an enhanced global selection strategy and a pool selection technique. It has then been tested via a series of well-known benchmark functions.

The effects of the variation of parameters have then been studied and the final parameters have been determined based on the experiments using ZDT series functions.

From the experiments in different test functions and comparisons to MOPSO and MOPSO-CD, the modified mPSO-DHA has proved to be an efficient multi-objective optimisation algorithm both in convergence and diversity. The proposed algorithm has performed very well in all the tested problems.

The search speed and the diversity have been significantly improved in the new mPSO-DHA as it is able to find the Pareto-fronts while the other two multi-objective PSO algorithms MOPSO and MOPSO-CD did not in ZDT1-3 and DTLZ6. According to the simulation results with ZDT4, DTLZ1 and DTLZ3, mPSO-DHA has also outperformed the other two PSOs in solving multi-modal problems. The new global selection strategy has helped to maintain the diversity of solutions when keeping the ability of finding the true Pareto-front; this has been proved in the tests with ZDT2 and DTLZ2.

Comparing with other evolutionary algorithms, mPSO-DHA has outperformed PAES and SPEA. It has provided competitive results as compared to NSGA-II in the simpler problems ZDT1 and ZDT2. The advantage of solving complex problems has led to better results than NSGA-II in solving ZDT3 and ZDT4.

In the next chapter, modelling on steel crack propagation using fuzzy and neural-network will be described, an error compensation structure using Gaussian mixture models will also be introduced.

Chapter 4

Modelling Steel Crack Propagation using Fuzzy and Neural-Network, and Error Compensation via Gaussian Mixture Models

4.1 Modelling of Material Behaviour

The modelling of material behaviour can generally be categorized into two types: physically-based modelling (e.g. Rice-Tracey model [Rice and Tracey, 1969], Gurson model [Gurson, 1975], or Rousellier model [Rousselier, 1987] combined with finite element analysis), and data-driven approaches, such as fuzzy modelling, neural networks etc., also known as Computational Intelligence (CI)-based models. Through the investigation in material modelling, the physically-based models and finite element analyses are well-established techniques which proved their success for a wide spectrum of material properties.

In most of the physically-based models, the material specific parameters need to be determined using extensive experimental work or via expert's knowledge. Therefore, the user needs to find a different set of parameters in order to apply the model on a different material. The cost of establishing the right material properties in the manufacturing of long distance gas pipelines will remain high unless a deeper understanding of the material's strength is achieved and this knowledge is applied during the design phase of the project.

The variability of material properties under certain conditions, which may also be called scatter, also becomes an issue with the physically-based models. This may be due to the lack of knowledge between the processing conditions and the material strength.

Data-driven approaches may have certain advantages in modelling multi-scale process. These computationally powerful tools can be used to model the relationships among the fracture characteristics, micro-structure data, and process conditions. The data-driven approach may also be combined with physically-based models as a part of a hybrid model. For example, the parameters in physically-based model can be estimated using suitably trained Artificial Neural Networks (ANN).

However, the training data in a data-driven modelling approach must be selected carefully, which means that the data must spread over the problem space in order to assist the complex function fitting process. The method of experimental design [Brownlee, 1984] can prove to be very helpful in providing a sufficient quantity and quality of data, so that the model's reliability can be improved.

A specific problem is chosen in this chapter to study the performance of data-driven approaches in modelling material behaviours.

4.2 Modelling of Steel Crack Propagation

High strength steel has proved to be a popular material in recent years, especially with gas companies when constructing long distance gas pipelines, as it provides more benefits in resistant high service pressure while the pipe wall thickness remains unchanged. By using high strength steel in the production of gas pipelines, the costs of fabrication and transportation can be lowered, this in turn increases the financial benefits. However, the calibrated empirical method, which is used to assess the fracture toughness of low toughness material, is no longer able to provide accurate assessment for the modern tough pipeline steels. Using the Charpy upper shelf energy predicted by the old application will lead to a large error in the characterisation of the pipeline fracture's resistance [Buzzichelli and Scopesi, 2000].

In previous research work [Ayvar et al., 2005], it was suggested that the fracture energy in the Charpy impact test, which relates to the fracture propagation, can be divided into two parts. One part is related to the flat fracture at the centre of the typical Charpy fracture surface, and the other part is associated with the slant fracture at the edges. The latter part is the most important component of fracture energy, due to the fact that the dominant failure mechanism in gas pipeline usually fast propagates a ductile shear, which can be reasonably attributed to the real failure mode. Therefore, for a comprehensive failure analysis, test specimens with different flat and slant fracture characteristics are required.

The Compact Tension (CT) test is an effective way to measure material toughness. It has been widely used in previous modelling efforts in order to identify the flat fracture characteristic of the material. The previous report from The Department of Mechanical Engineering, The University of Sheffield [Soberanis, 2007] analyses the CT test results using the 3D-Cellular Automata Finite Element (CAFE) approach. This method showed a good performance in simulating the fracture process, it has however a poor capacity of generalising from one material to another.

Based on the experiments from [Soberanis, 2007], this chapter presents and compares two data-driven approaches in order to model the same problem: a) a neural-fuzzy approach using hierarchical clustering and b) a Back-Propagation (BP) neural network with a double loop training procedure. The two approaches are used to model the relationship between the parameters of the steel crack propagation process and released flat fracture energy in the Compact Tension

(CT) experiments of X100 gas line pipe steel. The performances of these two data-driven models will be investigated and compared in order to find an efficient and accurate method of material modelling.

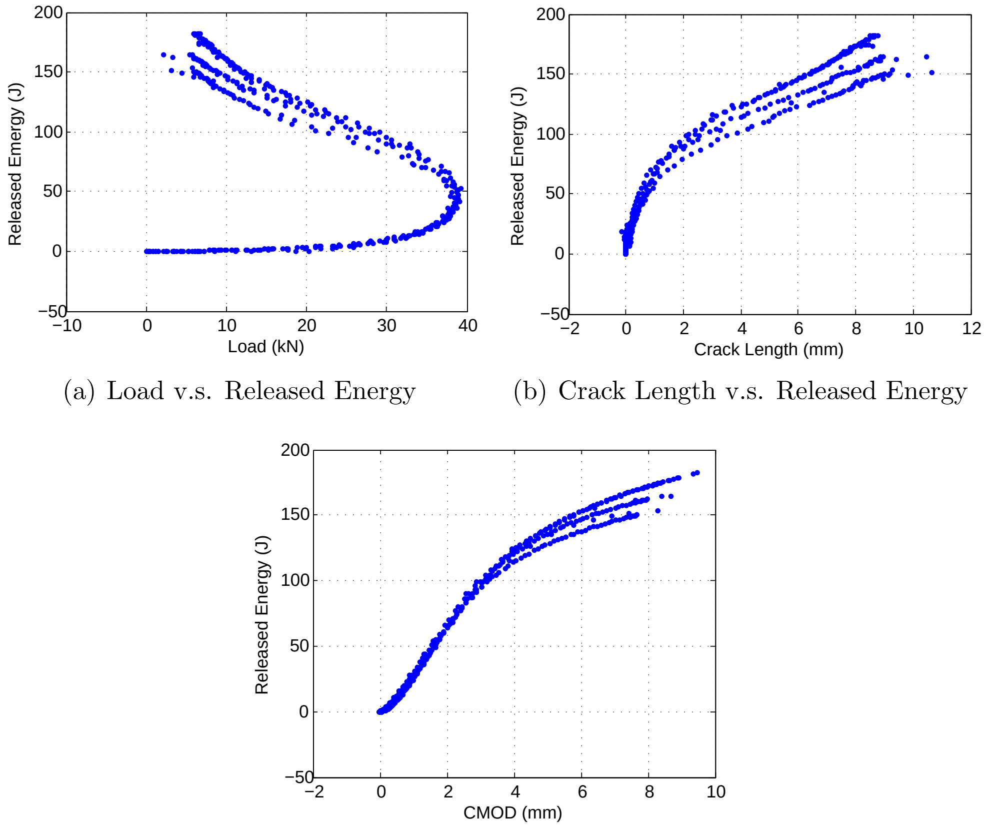
All experimental data originate from the work carried out in The Department of Mechanical Engineering, The University of Sheffield [Soberanis, 2007]. Six compact tension specimens were tested at room temperature, and the specimens were extracted with the initial crack along the longitudinal direction from the pipeline. This is the direction of a fast running shear fracture in real structures in the cases of burst pipelines. The test pieces were machined according to the specification of ASTM E-1820 [ASTM E 1820–01, 2001].All test samples were side grooved on each side up to 20% of the specimen original thickness to reduce shear lip formation and ensure a straight crack front. The specimen thickness was 15mm and the relation of initial crack length to specimen width was 0.5. The tests were carried out under displacement control at a low displacement rate of 0.01mm/s.

The composition of the X100 pipeline steel used in the experiments are shown in Table 4.1.

Table 4.1: Composition of X100 pipeline steel used in CT experiments

Element	C	Si	Mn	P	S	Cu
Wt%	0.06	0.18	1.84	0.008	0.001	0.31
Element	Ni	Cr	Mo	Nb	Ti	Al
Wt%	0.5	0.03	0.25	0.05	0.018	0.036

The data sets from tests include load, CMOD, crack length and the measurements of released flat fracture energy, a total of 432 data which come from six test data sets were used to develop the prediction model. In this work, 70% of the data are used for training the model, and 30% are used for model validation. Data distributions are shown in Fig. 4.1, which describe the relationships of the load versus the released energy, the CMOD versus the released energy, and finally the crack length versus the released energy.



(a) Load v.s. Released Energy

(b) Crack Length v.s. Released Energy

(c) CMOD v.s. Released Energy

Figure 4.1: Data Distributions

4.2.1 Correlation Coefficient Analysis

A statistical correlation coefficient analysis was carried out to identify the magnitude of the effect of different inputs to the output. The correlation coefficients among all variables are also summarised in Table 4.2.

From Table 4.2, it can be concluded that the correlation of CMOD against energy and crack length against energy is high, which means these two inputs affect fracture characteristics more than the load does. The coefficient of load against energy is negative, which may be caused by the decreasing of load in the middle stage of fracture process, due to the crack speed controlling procedure; the correlation coefficient between CMOD and the crack length is high, this agrees with

Table 4.2: Correlation coefficients among inputs and output

Variables	Load	CMOD	Crack Length	Released Energy
Load	1	-4.865E-1	-5.721E-1	-4.015E-1
CMOD	-4.865E-1	1	9.785E-1	9.725E-1
Crack Length	-5.721E-1	9.785E-1	1	9.552E-1
Released Energy	-4.015E-1	9.725E-1	9.552E-1	1

the fact that CMOD and crack length increase simultaneously during fracture propagation.

4.3 Fuzzy Modelling on Compact Tension Energy

Fuzzy modelling is a systematic approach which describes the process under investigation using fuzzy quantities, such as fuzzy sets, fuzzy rules, and linguistic labels. This approach is frequently used in the modelling of material properties and process design [Chen et al., 2004]. The main advantage of the fuzzy modelling technique is its ability of representing nonlinear complex systems using simple modelling structures, which in materials research can be related to good prediction performance and good generalisation properties. The linguistics “if-then” rules are easy to understand by non-experts, which improve the models transparency.

However, expert knowledge is required to build a fuzzy model, an efficient way of automatic rule generation or training can be achieved using various data-driven algorithms. Several improved adaptive fuzzy modelling approaches have so far been developed to generate the fuzzy rules automatically using fuzzy neural network training, also known as neural-fuzzy modelling.

One effective method [Zhang and Mahfouf, 2008, 2011] will be described, whereby hierarchical clustering is used to determine the number of fuzzy rules and generate an initial fuzzy rule-base from the data. A gradient decent algorithm is then applied to optimise the parameters of the fuzzy rule-base.

4.3.1 Fuzzy Modelling using Hierarchical Clustering and Gradient Decent

Data clustering is considered to be an effective method of generating the initial fuzzy rule-base. The clustering methods can traditionally be divided into two categories: hierarchical clustering and partitional clustering. Partitional clustering methods are either trying to associate data in predefined clusters or finding areas with higher data density. The advantage of this method is its fast clustering speed. However, most partitional clustering algorithms come with a degree of inaccuracy, e. g. they cannot lead to the same clustering result across various runs. Hierarchical clustering tries to build a tree structure which divides all the data by levels of similarity and builds parent-child relationships between different levels, this method yet suffers from high computational costs due to the high structure complexity [Jain et al., 1999].

Various clustering methods have been developed in the past. An improved hierarchical clustering algorithm [Zhang and Mahfouf, 2008, 2011] was developed to avoid the inaccuracy of partitional clustering and the high computational complexity of traditional hierarchical clustering method. The methodology of this approach can be summarised as follows:

1. A desired number of clusters N_c and threshold N_{max} are chosen, the threshold is a measurement of whether the data set is too large for computation, it should be set as $N_{max} \geq N^{\frac{1}{2}}$.
2. If $N \leq N_{max}$, begin the normal agglomerative complete-link clustering algorithm [Jain et al., 1999] to classify the data into N_c clusters, then end the clustering stage; if $N > N_{max}$, go to next step.
3. Separate the data equally and randomly into i groups, where $i = \text{Ceil}(N/N_{max})$, $\text{Ceil}(x)$ is a function that returns the smallest integral value that is not less

than x.

4. Classify the data in every group into j sub-clusters using the normal agglomerative complete-link clustering algorithm, where $j = \text{Floor}(N/i)$, $\text{Floor}(x)$ is a function that returns the smallest integral value that is not larger than x.

5. Select the representative data from every sub-cluster, the representative data is the data point which is closest to the centre of every sub-cluster.

6. Construct a representative data set which includes all the $i \times j < N_{max}$ data points.

7. Cluster the representative data set into N_c clusters using the normal agglomerative complete-link clustering algorithm.

8. Replace every the representative data point with the original data set in its corresponding sub-cluster.

From the clustering process, the information about clusters is provided, which can then be used to construct an initial fuzzy model.

Assume a $(D + 1)$-dimensional modelling problem, where D is the number of inputs, and the problem has one output.

Based on the information given by clustering, the rule base is composed and constructed by N_c fuzzy rules, let C_n represent the nth cluster, DN_n is the number of data points in C_n, the fuzzy rule corresponding to C_n can be presented as the following form:

$$R_n : \textbf{IF } x_1 \text{ is } A_1^n \textbf{ AND } x_2 \text{ is } A_2^n \textbf{ AND } x_D \text{ is } A_D^n \textbf{ THEN } y \text{ is } Z_n. \tag{4.1}$$

where $n = 1, 2, \cdots, N_c$; $x = [x_1, x_2, \cdots, x_D]$ are the input linguistic variables; A_i^n, $(i = 1, 2, \cdots, D)$ is the antecedent fuzzy sets; y is the output linguistic variables; and Z_n is the consequent fuzzy set.

The Gaussian function is selected as the membership function in this method, for every fuzzy set A_i^n, the centre of membership function c_i^n being the centre of the corresponding dimension, and the width of membership function σ_i^n can be calculated from solving Eq. 4.2,

$$\min_j(\mu_{A_i^n}(x_i^{nj})) = \min_j(\exp(-\frac{(x_i^{nj} - c_i^n)^2}{(\sigma_i^n)^2})) = Th \tag{4.2}$$

where $j = 1, 2, \cdots, DN_n$. By setting a suitable threshold $Th = 0.5$, the generality of the membership function can be guaranteed.

An initial fuzzy model can be constructed once the rule-base generation terminates. A parameter learning or optimising process should then be applied in order to fine-tune the prediction model. There are several methods which can be used for training fuzzy models. In this procedure, a gradient-descent optimization algorithm is adopted for tuning the membership function parameters, which are c_i^n and σ_i^n, where the Root Mean Square Error (RMSE) is calculated as the performance index. The parameter learning progress in kth iteration can be derived as in Eq. 4.3.

$$\begin{gathered}
\Delta c_i^n = \lambda_c \cdot (y_k - y_{dk}) \cdot (Z_n - y_{dk}) \cdot \tfrac{x_i^{nj} - c_i^n}{(\sigma_i^n)^2} \cdot \tfrac{\mu_n}{\sum \mu_n} \\
\Delta \sigma_i^n = \lambda_\sigma \cdot (y_k - y_{dk}) \cdot (Z_n - y_{dk}) \cdot \tfrac{x_i^{nj} - c_i^n}{(\sigma_i^n)^3} \cdot \tfrac{\mu_n}{\sum \mu_n} \\
\mu_n = \exp(-\tfrac{(x^n - c^n)^2}{(\sigma^n)^2})
\end{gathered} \tag{4.3}$$

where λ_c and λ_σ are the learning rate.

After the parameter training process is completed, the final fuzzy model is obtained. The fuzzy modelling results will be discussed next.

4.3.2 Modelling Results

The prediction results of a model with 15 linguistic rules are shown in Fig. 4.2. As a result of the hierarchical clustering method in the modelling process, the

RMSE index of predictions remained the same in repeated runs, whose value is 3.0864. The green and red lines are the upper and lower bound of the 10% error band. It can be seen that the model has predicted the released flat fracture energy very well in the crack beginning stage while the released energy is low; prediction is also accurate enough during the fracture propagation, i.e. the released energy is at the intermediate level; however, at the end of failure, where the released energy is high, the prediction becomes less accurate.

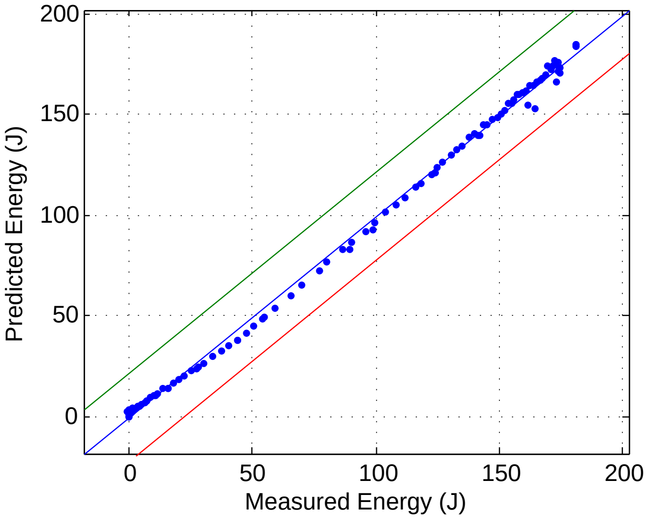
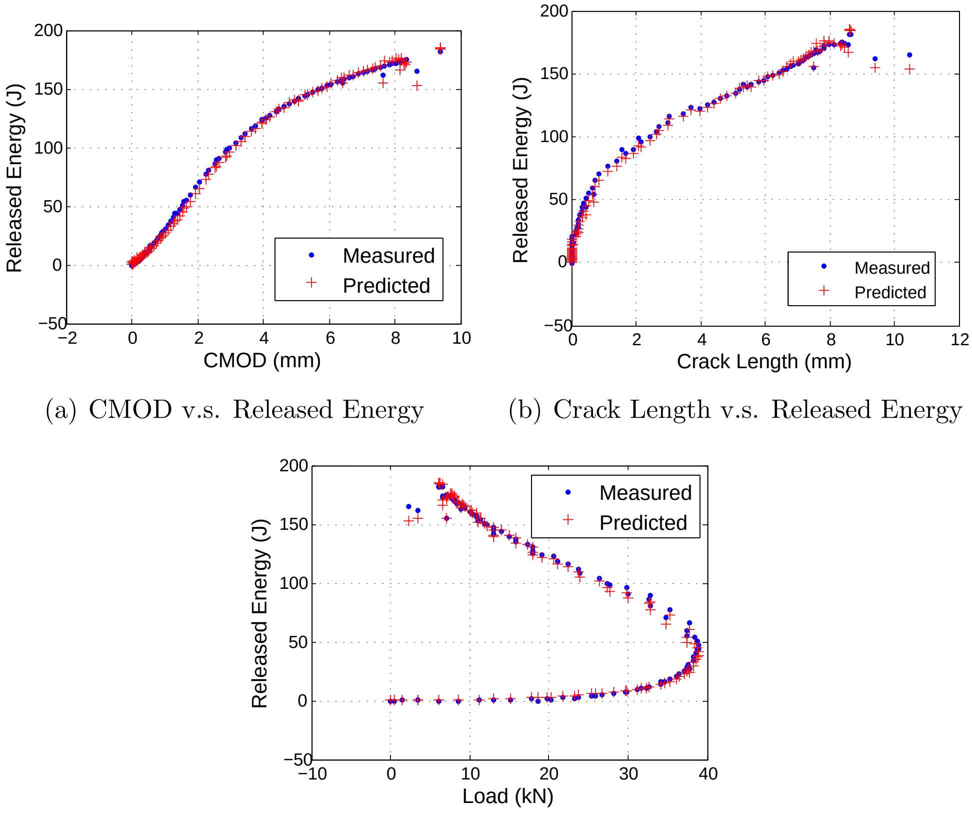


Figure 4.2: The Predicted Results for the Fuzzy Model

Fig. 4.3(a) shows the predicted curve, where the released energy is a function of CMOD. A good agreement between the predicted results and the measurement is observed. The agreement can also be seen in Fig. 4.3(b), which describes the released energy as a function of the crack length.

The prediction error in the fracture completing process may be caused by the inaccurate measurement in the experimental data due to the fast shape change at the end of the fracture propagation.

The curve in Fig. 4.3(c) shows the released energy as a function of the applied load. The failure in this figure starts from the bottom left corner, continues to the right side, and ends at the top left corner.



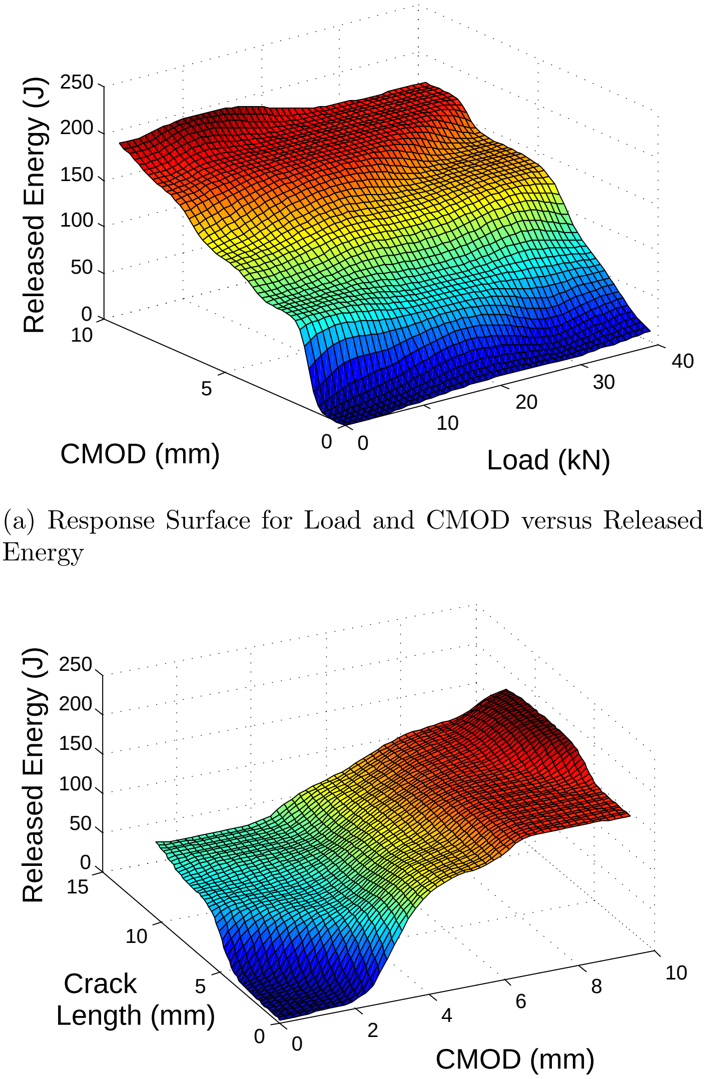
(a) CMOD v.s. Released Energy

(b) Crack Length v.s. Released Energy

(c) Load v.s. Released Energy

Figure 4.3: Predicted Results of Fuzzy Model

Using the fuzzy model, the response surfaces of the process are plotted in Figs. 4.4. The surfaces provide a crude idea about the interactions among inputs and output. The energy released rate is slow when the crack length and CMOD are both very small. The released energy increases as long as the crack length and CMOD grow. The increase in the load does not come with a high energy release rate because of the fixed crack propagation speed. Crack length increases with a fixed CMOD will not lead to the peak energy as there cannot be too much released energy without large shape changes of specimen. It can be concluded that the energy released rate changes slightly in the middle stage of fracture, and increases at the beginning and end of the crack propagation.



(a) Response Surface for Load and CMOD versus Released Energy

(b) Response Surface for CMOD and Crack Length versus Released Energy

Figure 4.4: Response Surfaces of Fuzzy Model

4.4 Neural Network Modelling on Compact Tension Energy

The fuzzy model constructed above shows a good performance in predictions before the end stage of fracture, however, the scatter in the high energy region can not be predicted well enough. The ANN approach has proved to be an efficient method in approximating continuous nonlinear functions. It is therefore considered to model the same relationship in order to reduce the scatter.

4.4.1 Double-Loop Neural Network Training Procedure

The validated network is a typical three-layer back-propagation (BP) network. Fig. 4.5 shows a three-layer BP network where there are an input layer, a hidden layer and an output layer.

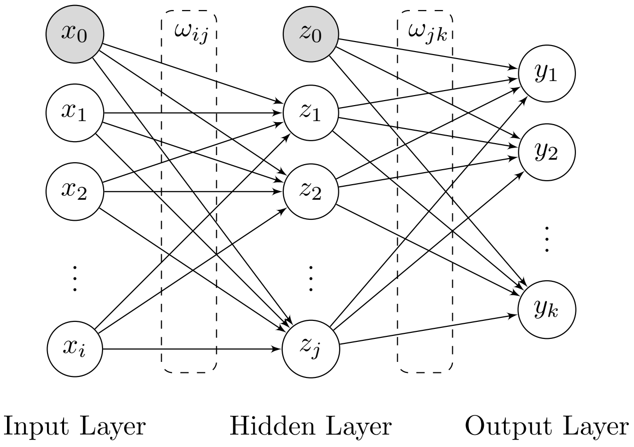


Figure 4.5: Three-Layer Neural Network

Typically, the training of a BP network consists of three steps:

1. Initialisation: x_0 and z_0 are fixed to 1, and all weights ω_{ij} and ω_{jk} are randomlized;
2. Forward process: calculate the network outputs according to the input values, the forward process follows Eq. 4.4.

$$\begin{cases} z_j(n) = f_j(\sum\limits_i \omega_{ij} x_i(n) + b_j) \\ y_k(n) = f_k(\sum\limits_j \omega_{jk} z_j(n) + b_k) \end{cases} \tag{4.4}$$

where ω_{ij} is the weighted connection from the ith input neuron to the jth hidden neuron, ω_{jk} is the weighted connection from the jth hidden neuron to the kth output neuron, z_j is the output of the jth hidden neuron, f_j and f_k are the activation functions of the respective hidden and output neurons, and y_k is the output of the kth output neuron.

In this problem, sigmoid function is selected as the activation function for hidden neurons, and linear activation is chosen for output neurons.

3. Backward process: adapt the network weights according to the error performance, the training algorithm is based on Levenberg-Marquardt optimisation, which proved to converge very fast and in most cases can deliver a good prediction accuracy.

In this part, a BP trained NN network with 8 hidden neurons was adopted. A BP neural network implemented via a double loop training procedure [Yang et al., 2003] is described in Fig. 4.6. The inner loop epochs iMax = 10 and outer loop iterations jMax = 50 were selected for this problem.

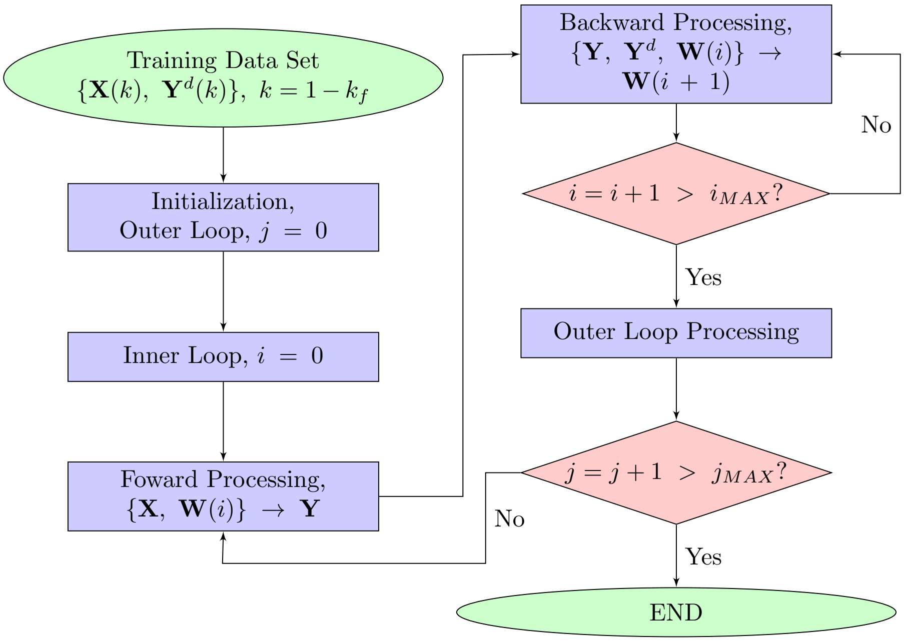


Figure 4.6: Double-Loop Training Procedure

The double loop training procedure has the advantages of monitoring the training progress and recording the optimal network against the predefined performance criteria.

4.4.2 Modelling Results

During the modelling process, data is divided into three parts: training data (60%), validation data (25%), and test data (15%). The training subset is used for updating the network weights and biases and computing the gradient; the validation subset is used for preventing the network from over-fitting. When the error on the validation data set begins to rise, the training process should be terminated.

Due to the variability of neural network training, several trials were performed in order to find an optimal model. The RMSEs of 5 runs using neural network model are summarised in Table 4.3. It can be seen that there is a variation of RMSE values from 6.9796 to 5.0736 in only five trials. The 2nd run with the error band shown in Fig. 4.7, yield a good performance, which is similar to the result obtained via the fuzzy modelling approach. The green and red lines in Fig. 4.7 are the upper and lower bound of the 10% error band. The prediction results of the validated neural network is shown in Fig. 4.8.

Table 4.3: RMSEs of neural network modelling

Trials	1	2	3	4	5	Average
RMSE	5.619	5.073	5.804	6.979	6.305	5.956

Fig. 4.9 plots the response surfaces of the neural network. It can be seen that the surfaces are not as smooth as the response surfaces of fuzzy model. There are hardly any information about the process that can be extracted from the response surfaces. In Fig. 4.9(a), a trend can still be seen that the released energy increases as long as the load and CMOD increase, although there is a wrinkled part in the middle of the surface, which is caused by the decreasing of load during the later stage of crack propagation. In Fig. 4.9(b), only the diagonal direction of the

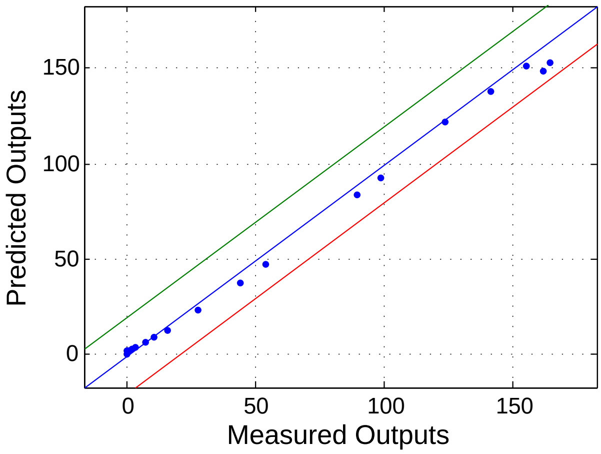
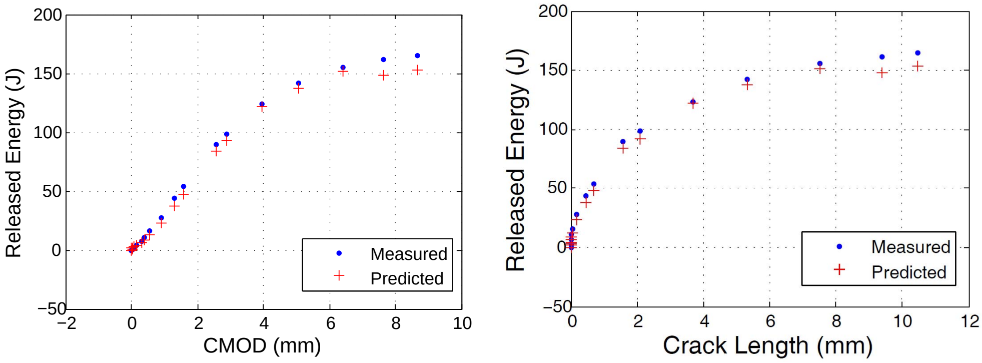
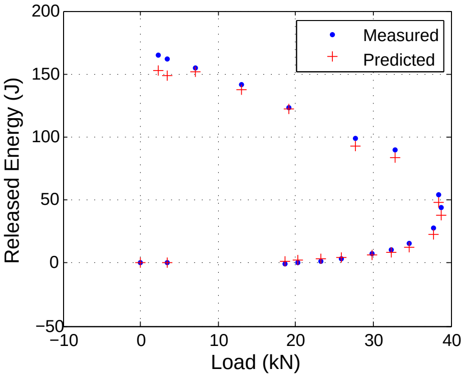


Figure 4.7: The Predicted Results for Neural Network



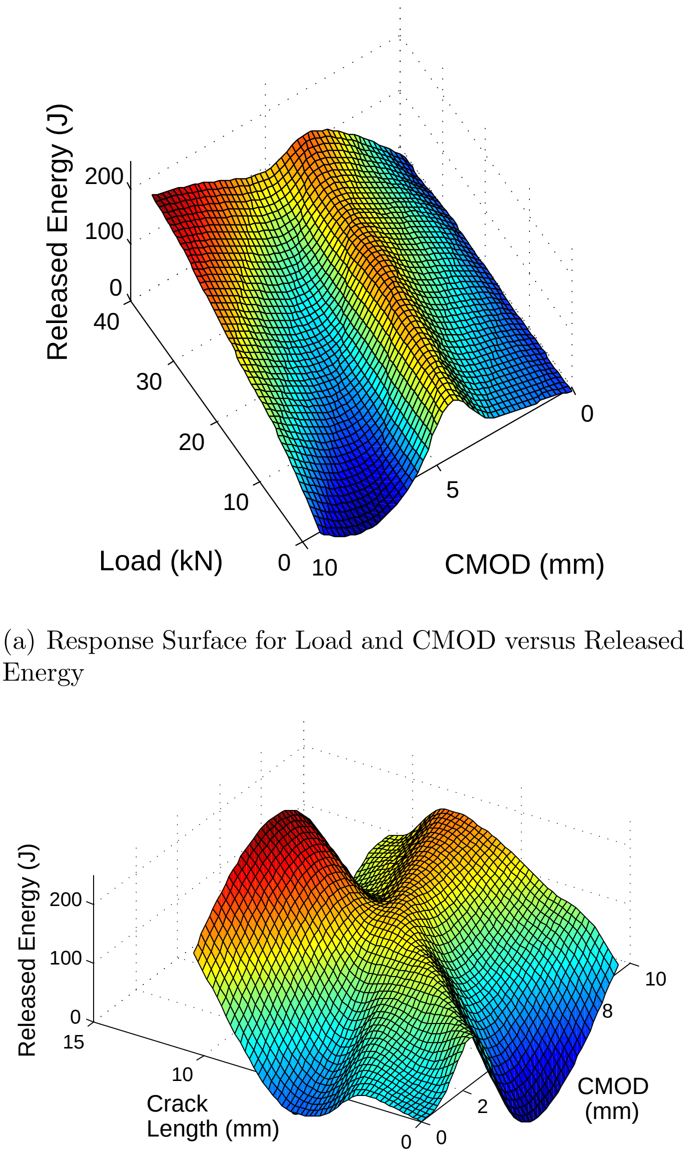
(a) CMOD v.s. Released Energy

(b) Crack Length v.s. Released Energy



(c) Load v.s. Released Energy

Figure 4.8: Predicted Results of Neural Network



(a) Response Surface for Load and CMOD versus Released Energy

(b) Response Surface for CMOD and Crack Length versus Released Energy

Figure 4.9: Response Surfaces of Neural Network

surface shows a logical inference of the propagating of crack: the released energy increases simultaneously with CMOD and the crack length, other part of Fig. 4.9(b) could not provide any useful information about the crack propagation, the reason is that the training data is mainly located on the diagonal direction of the surface, so that the network was only fitted into that direction. The differences from the response surfaces of fuzzy model are caused by the different modelling ideas between neural network and fuzzy model, where neural network is purely fitting a function according to the given training data, while fuzzy model can provide limited inferences for the unseen part of data because of the fuzzification process of data.

4.5 Comparison of the modelling results between Fuzzy Model and Neural Network

In order to compare the performance of the two modelling methods, the RMSE of fuzzy model was calculated using the same validation data set. However, no significant improvement was observed from neural network, since the NN modelling RMSE of 5.0736 is even larger than the RMSE of the fuzzy modelling results which is 4.1747. For the performance comparison in the high energy region, the RMSEs in that specific region were calculated. The RMSE is 7.5278 for the fuzzy model and 10.3836 for the neural network model, which means that the NN shows no improvements in the high energy region.

The two proposed modelling methods proved to be efficient and accurate in assessing the flat fracture energy and it is possible to use this approach to model new test data under different process conditions, such as different temperatures, compositions or micro-structures. However, the large errors (scatter) in the high energy region still exist in the predicted results of both models, therefore developing an efficient method in order to avoid this situation becomes a key issue.

An error compensation structure will be discussed next.

4.6 Error Compensation using Gaussian Mixture Model

There may be errors when embedding validated data-driven models in the real applications, since the data which is used for model training is from historical collections and statistics, the model may not be accuracy when new factors or changes, which may lead to errors, are found in the application. Moreover, when an established model is moved into a new environment, the slightly differences of environment may lead to the lost of model accuracy. An error compensation component – where the component is only based on data analysis and therefore can save the efforts of developing a completely new model – is an ideal solution to this situation.

The Gaussian Mixture Model (GMM) method [McLachlan and Peel, 2004] is considered to be a mature clustering and density estimation method. By applying it to the modelling process, one can monitor the error distributions of the predicted result. From the observed error distributions, a compensation procedure can be combined into the model validation stage. A stochastic compensated model structure, which is drawn in Fig. 4.10, is proposed to isolate the error distribution mode, and then to establish the correlations with the predictions from deterministic models.

After building a deterministic model - such as a fuzzy inference system or a neural network - a GMM will then be used as a stochastic based prediction reference. The GMM will analyse the error distribution in the training data space and provide a confidence band for the predictions emanating from the deterministic model.

4.6.1 Construction of a GMM

The data used to develop the GMM can be the same data set that has been used for training the deterministic model, it can also be new data set collected after the model is established. The data set $X^e = (x^e_1, x^e_2, \cdots, x^e_n)$ should consist of inputs

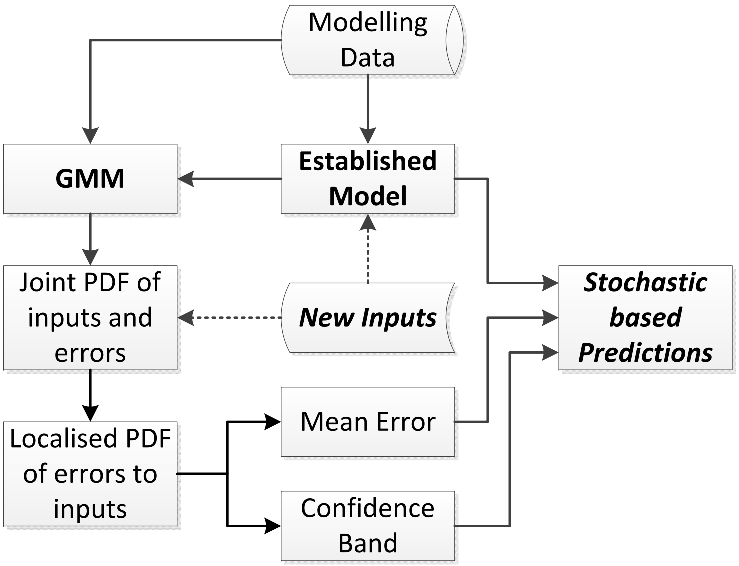

Modelling Data
GMM
Established Model
Joint PDF of inputs and errors
New Inputs
Stochastic based Predictions
Mean Error
Localised PDF of errors to inputs
Confidence Band

Figure 4.10: GMM Compensated Model Structure

$X = (x_1, x_2, \cdots, x_n)$ and errors $E = (e_1, e_2, \cdots, e_n)$. The Expectation Maximisation (EM) method will then be carried out in order to tune the parametres of GMM.

1. Initialize a GMM with randomly chosen parameters. The number of Gaussian componenets will be decided according to the analyses of Bayesian Information Criterion (BIC), which will be described in Step 6. k-means clustering is used to generate the initial parameters (ω_k, μ_k, σ_k), where i represents the kth component, ω_k is the mixing coefficient (weight) for the corresponding component, μ_k is the mean (centre) of the component k, and σ_k is the covariance matrix (radius) of the component, there GMM can be therefore described in Eq. 4.5.

$$P(x_n^e|\boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \sum_{k=1}^{K} \omega_k g(x_n^e|\mu_k, \sigma_k) \tag{4.5}$$

where $P(x_n^e|\boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ is the probability of that data point x_n^e exists, $g(x_n^e|\mu_k, \sigma_k)$ is the value of the kth Gaussian component for the data point x_n^e, $(\boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ are weight, mean, and radius vectors of all components, and K is the number of components.

2. Introduce a variable $Z_k(x_n^e)$, which describes the probability of that the kth Gaussian component generated the nth data point x_n^e, the equation for computing $Z_k(x_n^e)$ is shown in Eq. 4.6;

$$Z_k(x_n^e) = \frac{\omega_i g(x_n^e|\mu_i, \sigma_i)}{\sum\limits_{k=1}^{K} \omega_k g(x_n^e|\mu_k, \sigma_k)} \tag{4.6}$$

3. According to the Gaussian parametres and $Z_k(x_n)$ that are already calculated, the statistical weight, mean, and radius ($\bar{\omega}_k$, $\bar{\mu}_k$, $\bar{\sigma}_k$) can be computed for all components using Eq. 4.7.

$$\begin{cases} \bar{\omega}_k = \frac{1}{N} \sum\limits_{n=1}^{N} Z_k(x_n^e) \\ \bar{\mu}_k = \frac{\sum\limits_{n=1}^{N} Z_k(x_n^e) x_n^e}{\sum\limits_{n=1}^{N} Z_k(x_n^e)} \\ \bar{\sigma}_k = \frac{\sum\limits_{n=1}^{N} Z_k(x_n^e)(x_n^e - \mu_k)(x_n^e - \mu_k)^T}{\sum\limits_{n=1}^{N} Z_k(x_n^e)} \end{cases} \tag{4.7}$$

where N is the total number of data points.

4. Compute the likelihood using Eq. 4.9.

$$P(X^e|\boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\sigma}) = \prod_n \sum_k Z_k(x_n^e) \tag{4.8}$$

where $P(X^e|\boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\sigma})$ is the probability of that the whole data set X^e belongs to current GMM.

5. Set $(\bar{\boldsymbol{\omega}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\sigma}})$ from step 3 as the parameters for next iteration, iterate step 3 and step 4 until the following condition in Eq. 4.9 is satisfied or the predefined max iteration times are reached.

$$\ln P(X^e|\bar{\boldsymbol{\omega}}, \bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\sigma}}) - \ln P(X^e|\boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\sigma}) < \epsilon \tag{4.9}$$

where ϵ should be a small positive number, it is set to 1.0×10^{-4} in this problem.

6. The final number of Gaussian components can be decided after analysing the Bayesian Information Criterion (BIC) of GMMs fitted using different number of components. The BIC is defined in Eq. 4.10.

$$BIC = -2 \log P(X^e | \boldsymbol{\omega}, \boldsymbol{\mu}, \boldsymbol{\sigma}) + K \log N \tag{4.10}$$

   It is obvious that more Gaussian components will fit a better model, which gives smaller BIC value. However, increasing of components will greatly aggravate the computational burden. Therefore, a number of components which leads to a relatively small BIC shall be chosen.

The priori probability $P(e|x_i)$, which gives the probability that new input x_i get an error e, can be computed after the GMM is tuned. According to Bayes' Theorem [Pillai, 2002], $P(e|x_i)$ can be derived in Eq. 4.11.

$$\begin{aligned} P(e|x_i) &= \frac{P(x_i,e)}{P(x_i)} \\ &= \frac{P(x_i,e)}{\int P(x_i,\xi)d\xi} \\ &= \frac{P(x_i,e)}{\int \sum_{k=1}^{K} \omega_k g(x_i,\xi|\mu_k,\sigma_k)d\xi} \\ &= \frac{P(x_i,e)}{\sum_{k=1}^{K} \omega_k \int g(x_i,\xi|\mu_k,\sigma_k)d\xi} \end{aligned} \tag{4.11}$$

Hence, the estimated error can be computed using Eq. 4.12:

$$\bar{e}(x_i) = \int e \cdot P(e|x_i)de \tag{4.12}$$

The estimated error can either be given as a compensating inference or be used to compute a confidence band for the prediction results from the model, the confidence band (the expecting error standard deviation) can be computed as in Eq. 4.13.

$$Std(e(x_i)) = \sqrt{\int (e - \bar{e})^2 \cdot P(e|x_i)de} \tag{4.13}$$

Therefore, the error compensated output can be computed using Eq. 4.14.

$$y_i^c = y_i - \bar{e}(x_i) \tag{4.14}$$

where y_i^c is the compensated output for the input x_i, y_i is the output without compensation for the input x_i.

Finally, the error compensated outputs y^c with variances $Std(e(x))$ can be predicted through the proposed structure.

Experiments are carried out with both the fuzzy model and the neural network in order to examine the performance of the proposed error compensation structure. The models which were developed in 4.3 and 4.4 are used in the following parts.

4.6.2 A GMM-Compensated Fuzzy Model

Firstly, the same data set that was used for model training is imported into the construction of GMM. The output error distribution, where the training data is the input, is shown in Fig. 4.11. In order to measure the reliability of the fitted GMM_{f1}, Fig. 4.12 shows the curves for $P(e|x_{s1})$ and $P(e|x_{s2})$,which is the error probability for given inputs x_{s1} and x_{s2}.

The parameters of fitted GMM_{f1} are given in Table 4.4. The BIC analysis of components number is shown in Fig. 4.13, where $k = 5$ is chosen. It should be noted that only crack length and CMOD have been chosen to fit the GMM, because the input load shows not so relative to the output in section 4.2.1. The reduction of load proved efficiency in the experiment, in this case of fuzzy model compensation, the compensated RMSE for data set x_4 after reducing of load is 3.4944, yet the RMSE before the reducing of load is 4.3500, which is even larger that it is before applying the GMM. This is because that the changing of load is not linear.

The input x_{s1} is chosen for that it leads to an medium error 1.8622 through all the inputs, the corresponding output $y_{s1} = 162.4622$. It can be seen in Fig. 4.12

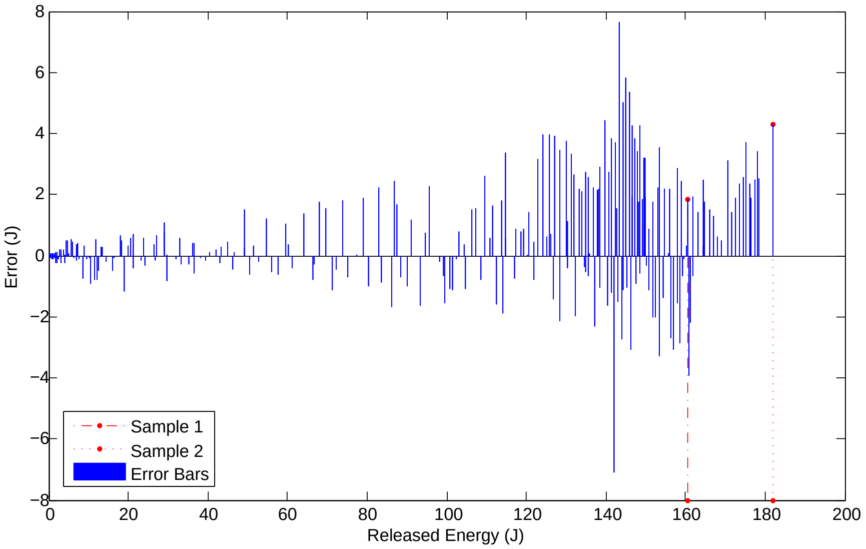


Figure 4.11: Output Error Distribution for Training Data

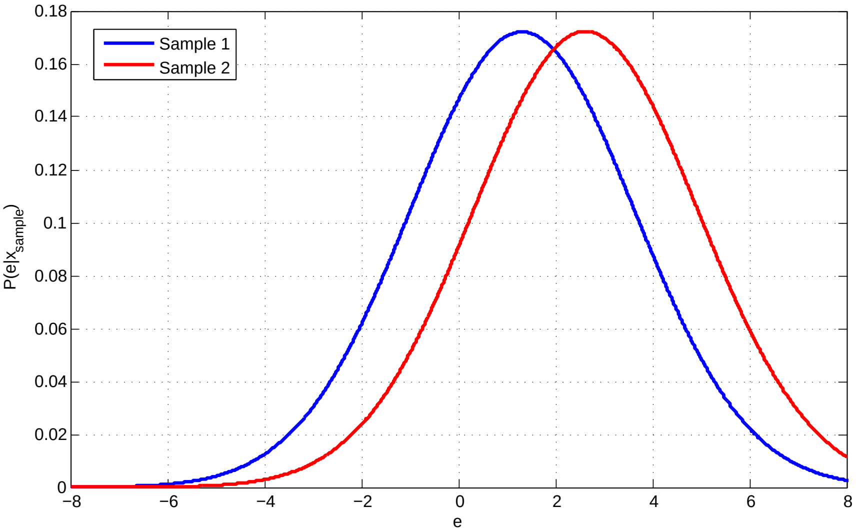


Figure 4.12: Error Probability Distribution for Sample Inputs

that the error probability curve reaches its maximum around point $(1.36, 0.1721)$, the most probable error 1.36 is not far from the actual value 1.8622. Using the error estimating approach shown in Eq. 4.12, an estimated error of the input x_{s1} can be computed, the resulting value is $\bar{e}(x_{s1}) = 1.3159$, which is 70% of the actual error. The expecting error variance can be calculated as in Eq. 4.13, the

Table 4.4: Fitted Parameters of GMM_{f1}

k		5				
ω		2.734E-1	2.297E-1	2.293E-1	1.072E-1	1.604E-1
μ	CMOD (mm)	4.020	1.445E-1	1.069	8.037	6.743
	Crack Length (mm)	4.505	0.000	3.163	8.592	7.550
	e (mm)	6.931E-1	1.800E-3	3.850E-2	3.212	-9.230E-2
σ	CMOD (mm)	1.546E-1	1.400E-2	2.474E-1	9.689E-1	9.001E-1
	Crack Length (mm)	5.083	1.000E-3	1.000E-1	1.320E-1	7.009E-1
	e (mm)	2.793	7.700E-3	2.577E-1	1.890	4.510

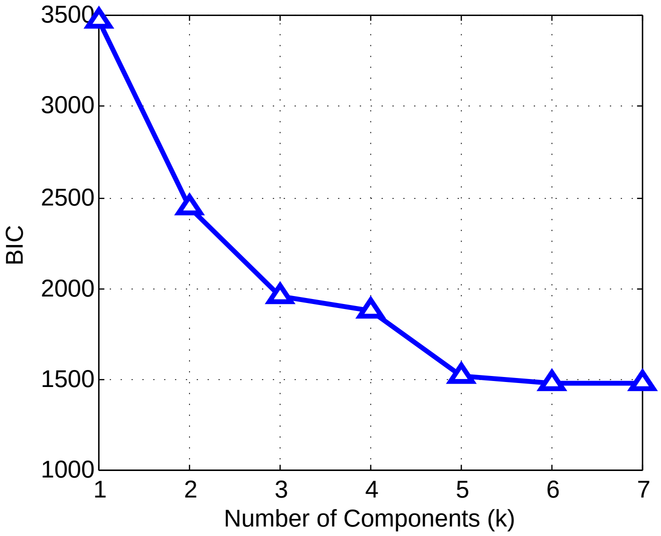


Figure 4.13: BIC analysis for GMM_{f1}

result is $Std(e(x_{s1})) = 2.3180$. These values $(\bar{e}(x_{s1}), Std(e(x_{s1})))$ are therefore predicting an compensated output $y^c_{s1} = 161.1463$ with a variance of 2.3180.

Another input x_{s2} is then selected since it is associated with a greater error 4.3064, where the output without error compensation is $y_{s2} = 186.4264$. It is shown in Fig. 4.12 that the the highest error probability is around point $(2.64, 0.1723)$, the most probable error is 2.64 comparing to the actual value of 4.3064. Following the same procedure as it has been done for x_{s1}, $(\bar{e}(x_{s2}) = 2.6233, Std(e(x_{s2})) = 2.3194)$ can be computed, the estimated error 2.6233 is

60.9% of the actual error. The final compensated output is $y^c_{s2} = 183.8031$ with a variance of 2.3194.

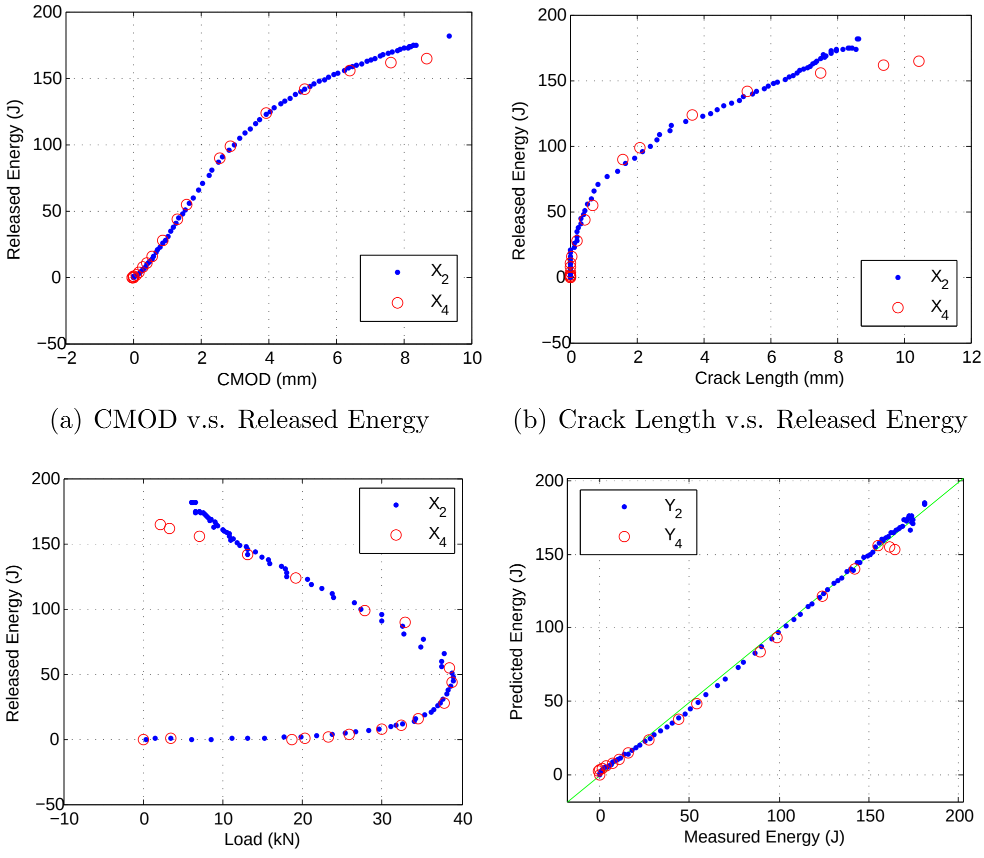
From the calculation of data point x_{s1} and x_{s2}, the fitted GMM_{f1} can generate estimated error for given inputs, and the compensated outputs are closer to the real results. It can be concluded that the GMM is well fitted to the training data, therefore, the GMM are now integrated with the fuzzy model for error compensation and the outputs' confidence analysis.

Two new data sets are prepared for the experiment, the distribution of data sets X_2 and X_4, together with the the outputs without error compensation Y_2 and Y_4, are shown in Fig. 4.14. The RMSE of Y_2 and Y_4 are 2.7823 and 4.1747 respectively.

Fig. 4.15 shows the compensated outputs for X_2, and Fig. 4.16 is the compensated result for X_4. It can be seen that the compensated outputs reduced the error, especially in the high energy region. These error reductions are verified through the computation of RMSE. The RMSE for Y^c_2 is 2.7348 (98.29% of the RMSE before compensation), and 3.4944 for Y^c_4 (83.70% of the RMSE before compensation).

4.6.3 A GMM-Compensated Neural Network

The model used in this part is the same as the network which was trained in 4.4. In order to construct the GMM, the training data is used to generate an error distribution. However, the fitted GMM_{n1} cannot predict acceptable error compensation, the RMSE after compensation (0.9914) is even lager than it was (0.9904) before the compensation. Fig. 4.17 shows the performance of the applying GMM_{n1}. Obviously, GMM_{n1} is not fitted well using the training data. The bad fitting of GMM_{n1} is because that the mean error is too small for the training data set, where it can be seen that the errors spread evenly above and below zero in all the data range, so that the GMM can not generate meaningful error compensations.



(a) CMOD v.s. Released Energy

(b) Crack Length v.s. Released Energy

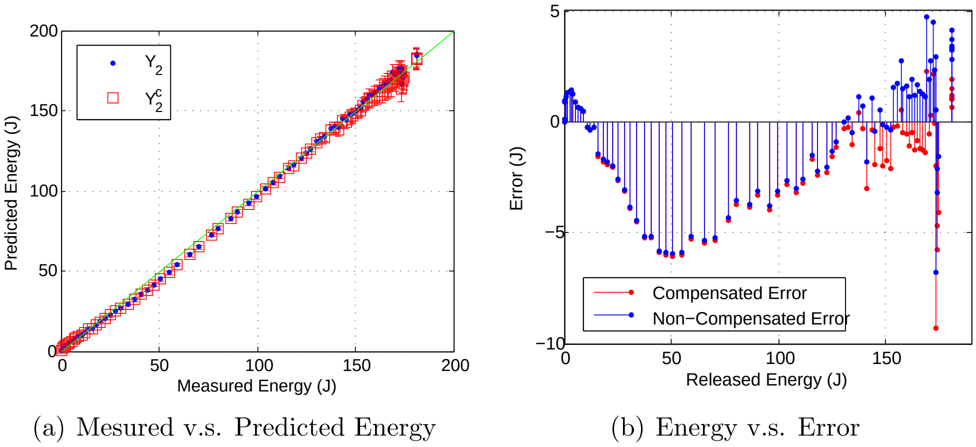
(c) Load v.s. Released Energy

(d) Measured v.s. Predicted Energy

Figure 4.14: Distribution of X_2, X_4, Y_2 and Y_4

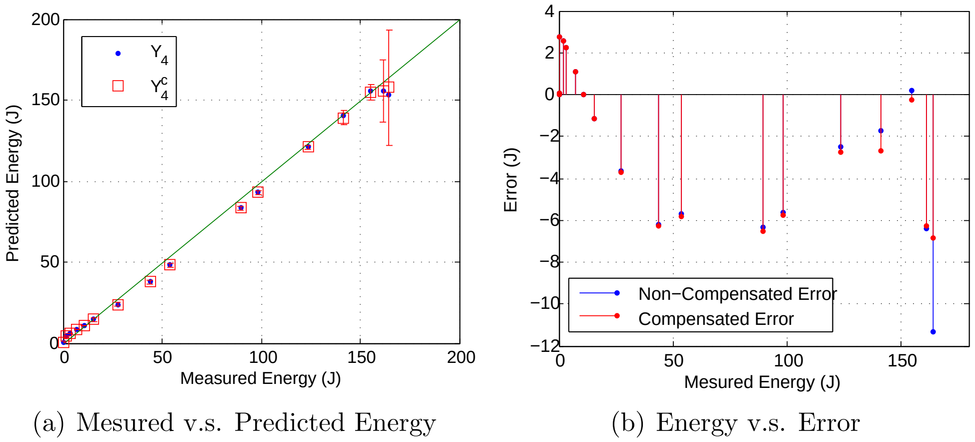
Data set X_2 as in 4.6.2, together with X_1, which is part of the training data, are introduced to construct an acceptable GMM, the reconstructed GMM is represented by GMM_{n2}. Error distributions before and after the application of GMM_{n2} for X_1 and X_2 is shown in Fig. 4.18, where the RMSE is 1.8473 without compensation and 1.4003 with compensation. It can be seen that, in lower energy region, GMM_{n2} provides better error compensations when the errors are negative; in higher energy region, it predicts acceptable error compensation no matter what the errors are.

Same test data set X_4 as in 4.6.2 is used to evaluate the performance of GMM_{n2}. The results are shown in Fig. 4.19, where the RMSE before compensation is 5.0736, and 4.2013 (82.81%) after the compensation. It can be seen that



(a) Mesured v.s. Predicted Energy (b) Energy v.s. Error

Figure 4.15: Error Distributions for Y_2 and Y_2^c (with GMM_{f1})



(a) Mesured v.s. Predicted Energy (b) Energy v.s. Error

Figure 4.16: Error Distributions for Y_4 and Y_4^c (with GMM_{f1})

the errors are compensated in all range.

The fitted parameters for GMM_{n2} is listed in Fig. 4.5. The BIC analysis of components number is shown in Fig. 4.20, where $k = 4$ is chosen.

4.6.4 Analysis of The Experimental Results

From the experiments using different models and different data sets, the proposed error compensation structure proved to be an efficient method in reducing the

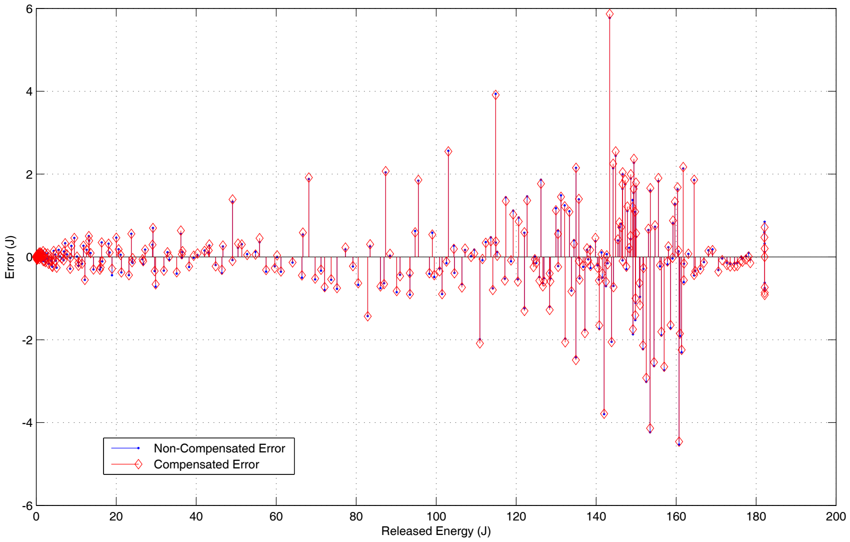


Figure 4.17: Traning Set - Error Distribution Before and After Applying GMM_{n1}

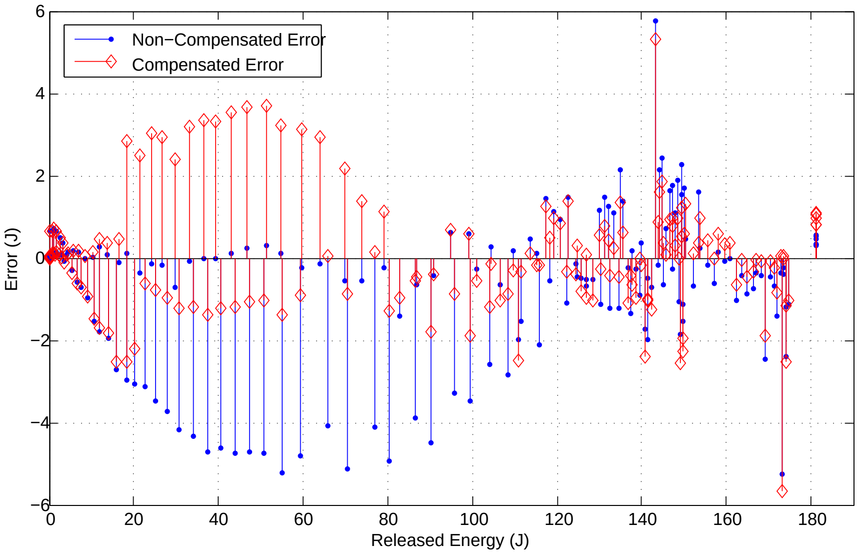
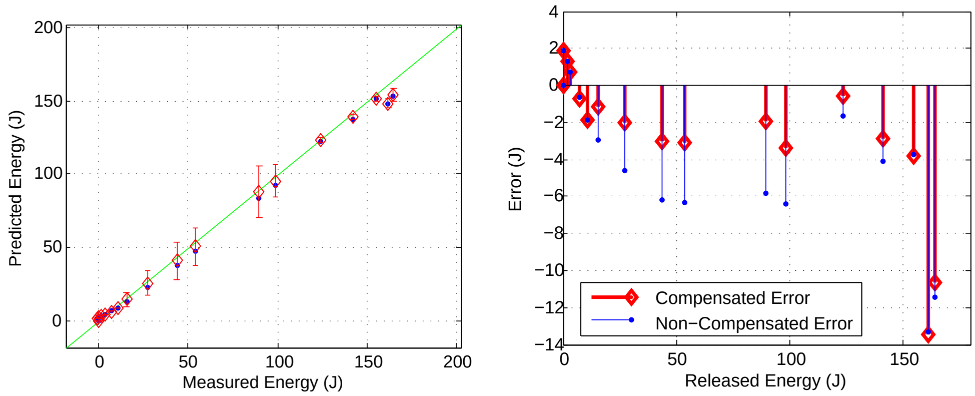


Figure 4.18: X_1 and X_2 - Error Distribution Before and After Applying GMM_{n2}

errors of predicted results.

The GMM used for fuzzy model is fitted using the training data of the model, where the error distribution for the training data is biased. The GMM achieved a good fitting and provided acceptable error compensations. The latter experiment



(a) Mesured v.s. Predicted Energy

(b) Energy v.s. Error

Figure 4.19: Error Distributions for Y_4 and Y_4^c (with GMM_{n2})

Table 4.5: Fitted Parameters of GMM_{n2}

k		4			
ω		2.123E-1	3.277E-1	1.587E-1	1.300E-2
μ	CMOD (mm)	2.252E-1	7.067	1.615	3.646
	Crack Length (mm)	3.213E-9	7.768	7.939E-1	3.874
	e (mm)	9.840E-2	1.712E-1	-3.725	-3.725E-1
σ	CMOD (mm)	4.060E-2	1.862	7.565E-1	3.329
	Crack Length (mm)	1.000E-3	8.143E-1	7.738E-1	7.1025
	e (mm)	1.214E-1	2.353	1.119	6.054E-1

in neural network, where a new biased data set are imported for GMM fitting, lead to the similar results that the GMM provides good error compensations.

From the initial experiment in neural network, where the training data were used to fit the GMM, it can be seen that the error distribution for the training data set is more like to be unbiased, this leads to a GMM that cannot provide good error compensation – the GMM cannot extract reliable information about the error distribution patterns from the fitting data.

In the latter experiment in neural network, it can be seen from Figs. 4.15 and 4.16 that the compensation from GMM is lager in the high energy region, smaller

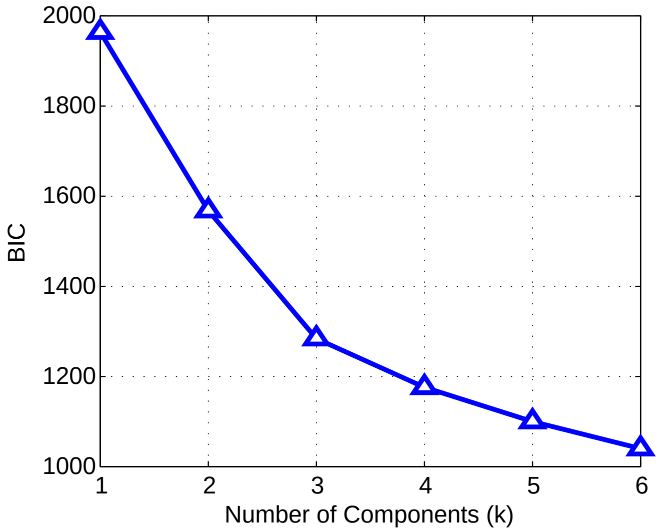


Figure 4.20: BIC analysis for GMM_{n2}

in lower energy region, where the range of error compensation is the same as the error changing range of the training data, which can also be seen in Fig. 4.11.

The same pattern can be found in the experiment using neural network, where the error compensations and confidence band of the outputs in Fig. 4.19 are lager in the lower energy region, smaller in the higher energy region. It can be found the same error changing trend in Fig. 4.18 that the data set used to fit the GMM has lager errors in the lower energy region while the errors are smaller in the higher energy region.

From the above analyses, it can be clearly concluded that the fitted GMM will reflect the error distribution according to the data which is used to create the GMM. Therefore, the more comprehensive data sets are collected after the model application, the more accuracy GMM can be fitted. Once a GMM is fitted, it will provide useful information about error compensations and confidence bands.

4.7 Summary

In this chapter, two data-driven models were successfully trained to model the crack propagation process of X100 gas line pipe steel. The two selected model structures are a) a neural-fuzzy approach using hierarchical clustering and b) a Back-Propagation (BP) neural network with a double loop training procedure. The performances of these two data-driven models are investigated and compared. The neural network is found to be inferior by comparison of the RMSE for this specific modelling problem. Moreover, the response surfaces of neural network gave no idea about the process, where in contrast the response surfaces from fuzzy model drew clear maps among different variables.

A Gaussian Mixture Model (GMM) error compensation structure is then proposed and validated, where GMM is used as a stochastic based prediction reference for the developed models. This structure is designed in order to avoid the efforts on rebuilding a completely new model when the error is caused by slightly changes of the environment or newly found error-relating factors. Experiments showed that the errors of the predicted results from both models were decreased after applying the proposed structure. It can be concluded that GMM can give reliable error compensations and confidence bands.

In the next chapter, a data-driven model embedded finite element modelling approach will be introduced.

Chapter 5

A Synergistic Modelling Approach Combining Data-driven Model and Finite Elements Model

5.1 Finite Element Method and Constitutive Equations in Material Area

The finite element method (FEM) (also known as finite element analysis) is a well-known and comprehensive method which provide a numerical approach to finding approximate solutions to partial differential/integral equations or systems. It has been introduced in Section 2.5 and several damage models based on FEM were listed as well.

In general, a finite element model in material area should include two key factors:

1. A geometrical model of the specimen to be analysed in which the geometry

is divide into a number of discrete sub regions ('elements', or often called 'meshes'), which are usually formed by the connection of several nodes. Nodes are assigned at a certain density throughout the material depending on the anticipated stress levels of a particular area.

2. The constitutive equations which describe the relationship between applied stresses or forces, and strains or deformations. The damage models mentioned in Section 2.5 are defined for different damage conditions.

The geometrical model varies according to different shapes of the specimen. The density of meshes is predefined depending on the area that to be analysed. There are different categories of elements, where the use of different kind of elements depends on the damage type. Both of the element type and geometrical model needs to be defined before the analysis, so that the geometrical model can be considered as the objective conditions of the FEM.

The other part of the FEM is the selection and the adjusting of the constitutive equations and the parameter settings for the equations under different situations. The damage models introduced in Section 2.5 relate to the construction of constitutive equations under different damage situations. For example, the Gurson-Tvergaard-Needleman (GTN) damage model [Tvergaard and Needleman, 1984] is defined in the form of the constitutive equations as follows [Tvergaard and Needleman, 1984]:

$$\begin{gathered}\Phi = (\tfrac{\sigma_{eq}}{\sigma_Y})^2 + 2q_1 f^* \cosh(q_2 \tfrac{3p}{2\sigma_Y}) - (1 + q_3(f^*)^2) = 0 \\ f^*(f) = \begin{cases} f, & f \le f_c \\ f_c - \frac{f_u^* - f_c}{f_F - f_c}(f - f_c), & f > f_c \end{cases}\end{gathered} \tag{5.1}$$

where σ_{eq} is the von Mises equivalent stress, σ_Y is the yield strength of the material, p is the hydrostatic stress, q_1, q_2, and q_3 are fitting parameters which depend on the material, f_c is the critical value of void volume fraction, f_F is void volume fraction at final fracture and $f_u^* = 1/q_1$.

The procedure of constructing a finite element model is shown in Fig. 5.1. The figure shows the procedure of constructing a finite element model for a simple bar as an example. The importance of idealization and discretization is to identify the different connecting structures or stressing areas of the modelling target. Depending on the structure that to be analysed, the type of elements can be varied, different types of the elements will lead to very different results due to the theories behind those elements. Fig. 5.1 only shows two types of elements while there are dozens of different types of elements have been developed in the open literature. The critical part of the geometrical plot is to determine that how the elements should distribute. At last, the constitutive equations of the elements is defined. The parameters in the constitutive equations may be determined through experiments.

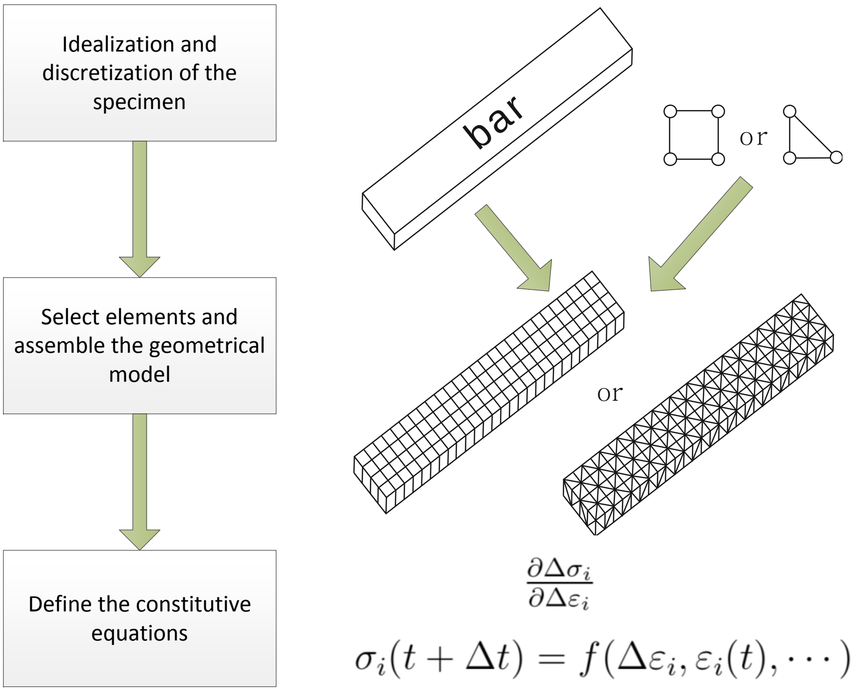

Idealization and discretization of the specimen
Select elements and assemble the geometrical model
Define the constitutive equations
bar
or
or
$\frac{\partial \Delta \sigma_i}{\partial \Delta \varepsilon_i}$
$\sigma_i(t+\Delta t) = f(\Delta\varepsilon_i, \varepsilon_i(t), \cdots)$

Figure 5.1: Finite Element Model Construction Procedure, where σ_i is the stress of the ith calculation point, ε_i is the strain of the ith calculation point, t is time.

The analyses of the modeled specimen can then be carried out based on the finite element model. There are several softwares that can be used for finite element analyses in the market, one of the most popular FEM softwares is ABAQUS, which is used in this project.

The basic analysing process of ABAQUS is described in Fig. 5.2. In the analyses, ABAQUS will firstly allocate the element strain increment for each element, which is calculated from the preset situation. The iterative computation will be carried out during the analyse. The strain increment will then be passed to the user sub-routine, in which the constitutional equations are defined. Together with other user-defined variables, the stress increment at each Gauss point for current element will be computed. Then the internal force in the element and the element tangent matrix can be generated. Hence, the global equilibrium equations can be calculated in order to reflect the whole state of the specimen. Knowing the new state of the specimen and the preset force and boundary condition, the strain increment for next iteration can be computed. The iterative analyse will continue until the simulation time runs out.

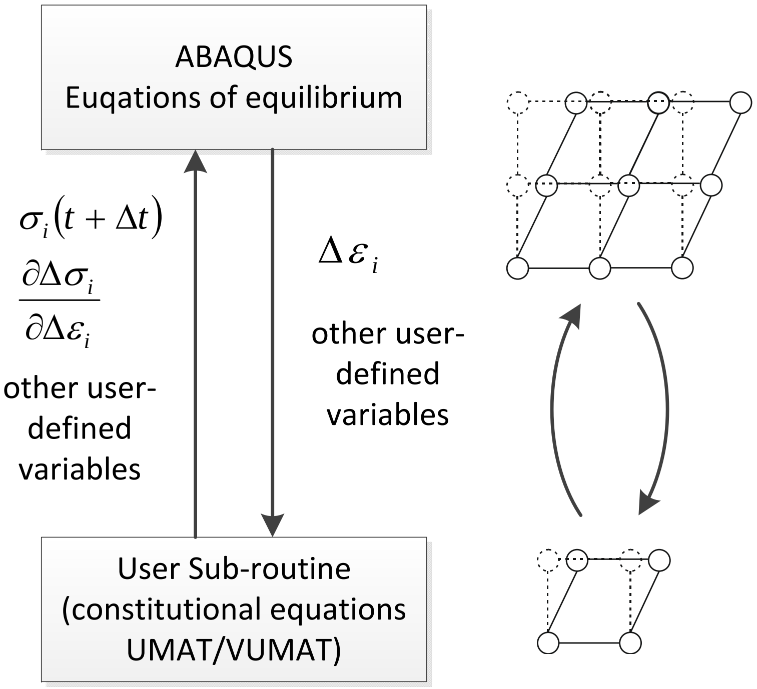

ABAQUS
Euqations of equilibrium
$\sigma_i(t+\Delta t)$
$\frac{\partial \Delta\sigma_i}{\partial \Delta\varepsilon_i}$
other user-defined variables
$\Delta\varepsilon_i$
other user-defined variables
User Sub-routine
(constitutional equations
UMAT/VUMAT)

Figure 5.2: ABAQUS finite element analysing process

It is worth noting that there are two modes for one experiment to be analysed: the explicit/dynamic method, and the implicit/standard method. The implicit method is often used to analyse static equilibrium structures. The choice of element types for implicit method is wide, and the time increment for the simulation can be large. However, the implicit method requires iterative and convergence checking, since the unknown values in this mode are obtained from current information, which may not be the known value but the calculated result. The explicit method are used to solve large, highly discontinuous or high-speed dynamic problems. No iterative or convergence checking are required in this mode, so that the explicit method is more efficient in solving very large problems. But the time increment for explicit mode must be kept small in order to maintain the accuracy.

5.2 Limitations of Constitutive Equations

Being the core part of finite element analysis, damage models based on the constitutive equations are well-developed. People can select different damage models based on the type of failure and the characteristics to be analysed. For instance, the Gurson model [Gurson, 1975] proposed a methodology for obtaining an approximate yield surface for material containing voids, while the Rice-Tracey model [Rice and Tracey, 1969] analyzed a case of dilatational growth of a single spherical void in a material under uniform stress state applied at infinity.

However, due to the lack of knowledge in the micro-structure level damage development, current models can only reflect part of the material nature. There are no constitutive model that can completely explain the whole process of ductile damage. Moreover, the aforementioned damage models in Section 2.5 have to be calibrated for a particular material before any of them can be used to predict the fracture behaviour of a structural component. The parameters of the constitutive equations can be very different for different materials, scales and specimens. The trail-and-error method is usually used for the parameter calibrations, which highly depends on the user expertise.

In [Corigliano et al., 2000], the extended Kalman filter was used in order to identify the parameters for the isotropic GT yield criterion for progressively cavitating ductile materials. The result showed that the identified parameters fitted the experimental result curve very well in the filtered space, however, the method still needs the prior knowledge in order to choose the type of the damage model that to be identified.

In conclusion, the constitutive models for material has the limitations of:

1. Expertise in material is required to determine the type of damage model to be selected.
2. The parameters of the constitutive equations need to be calibrated before application.
3. The calibrated parameters are limited only to certain scale, material and specimen.

5.3 Introduction of Data-driven Models to FEM

With the limitations of the constitutive models which were described in the previous section, a data-driven approach that can enhance or replace the constitutive damage models can be very useful in improving the generality and transferability of the finite element model.

In this section, a data-driven combined finite element model (DMFEM) structure is proposed. The aim of this work is to include a data-driven model in the user sub-routine instead of constitutional equations, where σ and $\Delta\varepsilon$ and other user-defined variables are defined as the inputs, and the outputs are the stress, the yield condition, and other variables for the next time step. Prospectively, this approach may extend the generality and transferability of the traditional finite element method.

A brief structure of the proposed model structure in comparison with the traditional finite element model in ABAQUS will be shown in Fig. 5.3, where the constitutive equations in the user-subroutine will be replaced by a fitted data-driven model which can be trained from any experimental data without the expertise of the damage types.

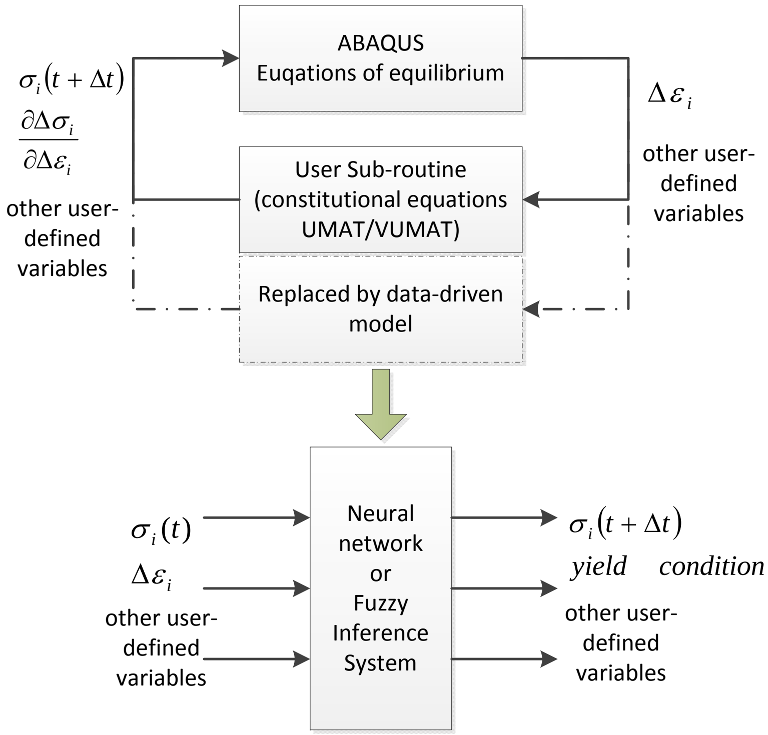

ABAQUS
Euqations of equilibrium
$\sigma_i(t+\Delta t)$
$\frac{\partial \Delta\sigma_i}{\partial \Delta\varepsilon_i}$
other user-defined variables
$\Delta\varepsilon_i$
other user-defined variables
User Sub-routine (constitutional equations UMAT/VUMAT)
Replaced by data-driven model
$\sigma_i(t)$
$\Delta\varepsilon_i$
other user-defined variables
Neural network or Fuzzy Inference System
$\sigma_i(t+\Delta t)$
yield condition
other user-defined variables

Figure 5.3: Proposed DMFEM structure, where σ_i is the stress of the ith calculation point, ε_i is the strain of the ith calculation point, t is time.

5.4 Experiments and Analyses

Experiments are designed in this part in order to test the feasibility of the proposed model structure. The experimental studies for the proposed DMFEM will firstly be carried out by replacing the simple element model. Then the trained model will be transfered from MATLAB into ABAQUS v6.10-2 environment.

5.4.1 A Simple Element Model

The experimental work starts from a simple element model. A simple element is the basic component of the finite element model, where the properties of any proposed model structure can be easily tested through it. The target simple element model is implemented based on the theory of isotropic hardening plasticity.

The constitutive equations for the isotropic hardening plasticity is shown in Eq. 5.2.

$$
\begin{aligned}
&f = \sigma_e - r - \sigma_y = (\tfrac{3}{2}\sigma' : \sigma')^{1/2} - r - \sigma_y \\
&d\lambda = \begin{cases} \frac{n \cdot C d\varepsilon}{n \cdot Cn + h} & f > 0 \\ 0 & f \leq 0 \end{cases} \\
&d\sigma = Cd\varepsilon^e = C(d\varepsilon - d\lambda n)dr = hdp = hd\lambda \\
&\sigma_{t+\Delta t} = \sigma + d\sigma \\
&\varepsilon^p_{t+\Delta t} = \varepsilon^p + d\varepsilon^p \\
&r_{t+\Delta t} = r + dr \\
&J = \frac{\partial d\sigma}{\partial d\varepsilon} = C - \frac{Cn \otimes Cn}{n \cdot Cn + h}
\end{aligned}
\tag{5.2}
$$

where f is the yield function, $d\lambda$ is the plastic multiplier, σ is the stress of current calculation point, ε is the strain of current calculation point, r is the isotropic hardening variable, p is the effective plastic strain, h is the hardening coefficient, J is the continuum Jacobian matrix for UMAT/Implicit implementation, ε^e and ε^p are elastic strain and plastic strain respectively.

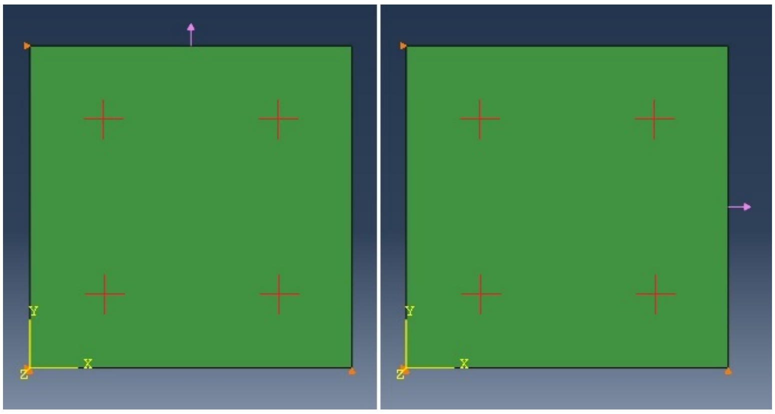


Figure 5.4: A Simple element model used in the experiment

The geometric model of the simple element used in this experiment is shown in Fig. 5.4, the red crosses (which can not actually be seen in the ABAQUS environment) in the figure represent the Gaussian integration points, where the computations are carried out during the simulation. The left edge of the model is locked in y-axis and the bottom edge of the model is locked in x-axis. The locking of the left and the bottom edges is to set the boundary conditions so that the changes of the model can be easily measured from the directions of x and y axes. The material properties are defined in Table 5.1. In order to train the neural network model, different loads will be applied through different directions, the stresses, the changes of strains, and the yield conditions for the integration points will be exported from ABAQUS.

Table 5.1: Material properties used in the experiments

Young's Modulus E (GPa)	Poisson Ratio ν	Yield Strength σ_0 (MPa)	Hardening h (MPa)
210	0.3	240	1206

5.4.1.1 Modelling using UMAT - Implicit Subroutine

The modelling approach will be established firstly through ABAQUS/Standard, where the subroutine UMAT will be used. The source code of the UMAT for isotropic hardening material which is used for data exporting was extracted from [Dunne and Petrinic, 2005]. The element type using in this part is CAX4, which is a regular type of element for the modelling of linear axisymmetric stresses. As the equilibrium and convergence checks are needed in the implicit analysis, the exported data must be cleaned first, the iterations that are calculated for equilibrium checks are deleted after being exported; this is because the equilibrium checking data sets are almost the same as in normal iterations. The exported time versus stress curves for integration point 1 with different top loads are shown in Fig. 5.5, the strain increments versus time curves for the same element are shown in Fig. 5.6. It should be noted that the curves under 100N can only be seen in the diagram of σ_{22}, they were covered by the curves under 200N in other diagrams, that is because of that both loads does not exceed the yield strength which has been set in Table 5.1, so that the changes are still in the elastic linear region. We can see from the figures that with uniform loads and fixed boundaries in the left and bottom, the nonlinear characteristics of plasticity can only be observed in the anisotropic stresses.

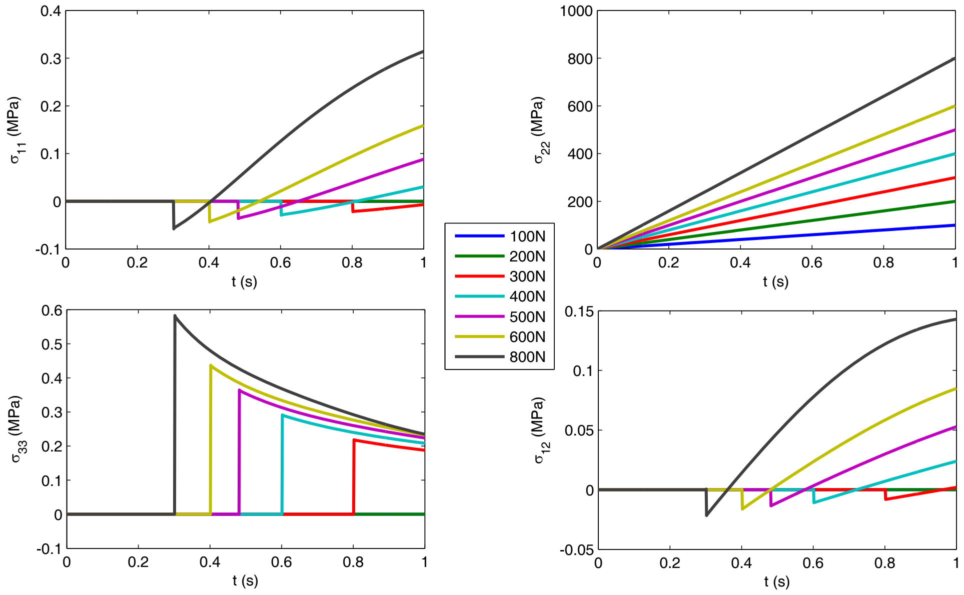


Figure 5.5: The stress vs time curves for integration point 1 of a elastic-plastic simple element model with different loads from the top

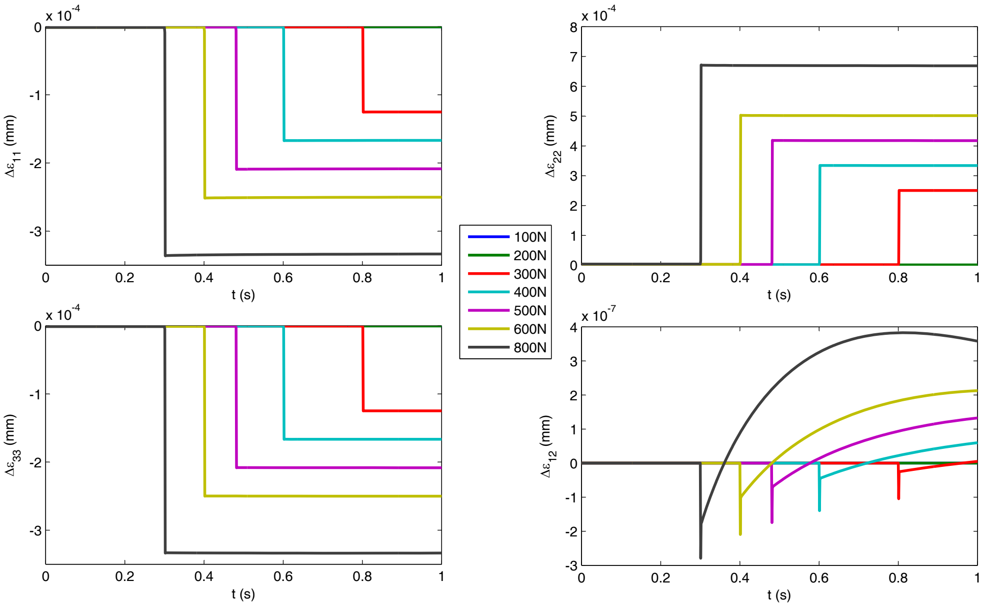


Figure 5.6: The strain increments vs time curves for integration point 1 of an elastic-plastic simple element model with different loads from the top

In the first experiment, the inputs for the model are the stress σ_{ij}^t and the strain increment $\Delta\varepsilon_{ij}$, the output is the stress for the next time increment $\sigma_{ij}^{t+\Delta t}$. The final training data is collected through sampling of the results under different loads. The data exported under 100N, 200N, 400N, 600N, 800N top load was used for training and the data under 300N top load was selected as the validation data. The 500N top load data is selected as the test data.

In the training stage, the number of inner cycles for the double loop training is set to 10, while the number of out cycles is 50, the training goal of mean square error is 1E-5, the sampling rate of data is set to 20Hz.

The modelling results are shown in Figs. 5.7 and 5.8. The red and green lines in Fig. 5.7 are the 10% upper and lower error bands respectively. It can be seen in Fig. 5.8 that the predicted results of the stresses fit very well comparing to the test data. The nonlinear plastic characteristic in the late stage of the simulation is predicted accurately. The RMSE of the stresses are listed as follows: $RMSE_{\sigma_{11}} = 0.0036$, $RMSE_{\sigma_{22}} = 2.7079$, $RMSE_{\sigma_{33}} = 0.0019$, $RMSE_{\sigma_{12}} = 0.0045$. The results shows that the trained neural network can predict accurate results with the information from ABAQUS.

Another information $J = \frac{\partial d\sigma}{\partial d\varepsilon}$, which is also known as the Jacobian matrix, is requested by ABAQUS when applying any user defined models in the form of UMAT subroutine. By exporting the Jacobian matrix from the gathered commercial codes, another experiment can be carried out with an addition output $\frac{\partial d\sigma}{\partial d\varepsilon}$. The Jacobian matrix is a 4×4 matrix, it is symmetric in the simulated problem. Fig. 5.9 and Fig. 5.10 shows the distribution of the value of Jacobian matrix.

The modelling results using the same training, validation and test data set is shown in Figs. 5.11 and 5.12. It can be seen that although the Jacobian matrix is predicted with a decent accuracy, the stress curves do not fit the testing data. This is because the additional outputs of the Jacobian matrix increased the model complexity. The model accuracy can be improved by adjusting the structure and training method of the neural network. However, the Jacobian matrix is only a state variable which is required by static analysis of finite element model, the

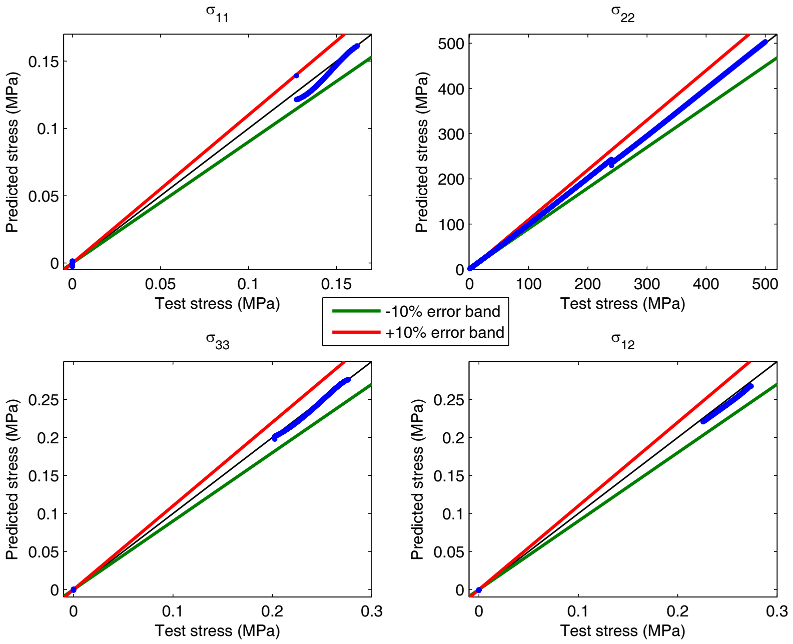


Figure 5.7: The predicted results vs actual results for integration point 3 of the simple element model with 500N load from the top

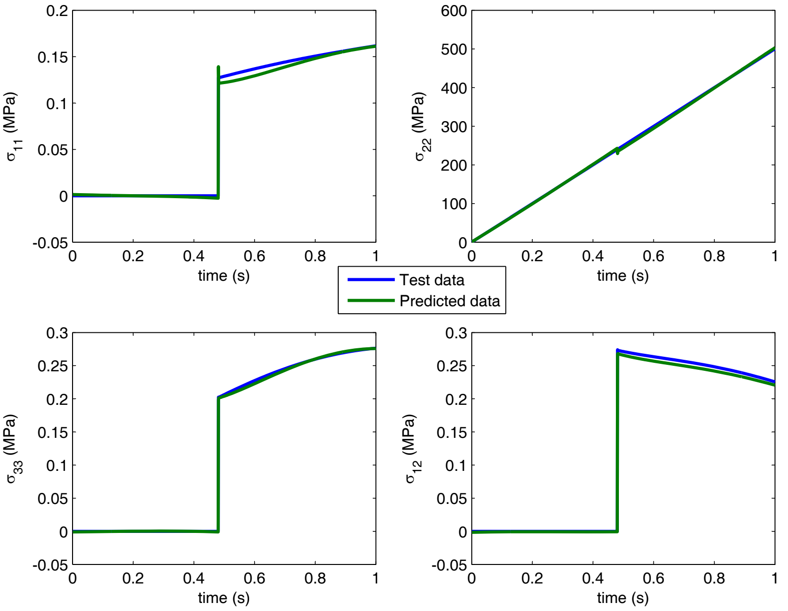


Figure 5.8: The predicted and actual results vs time for integration point 3 of the simple element model with 500N load from the top

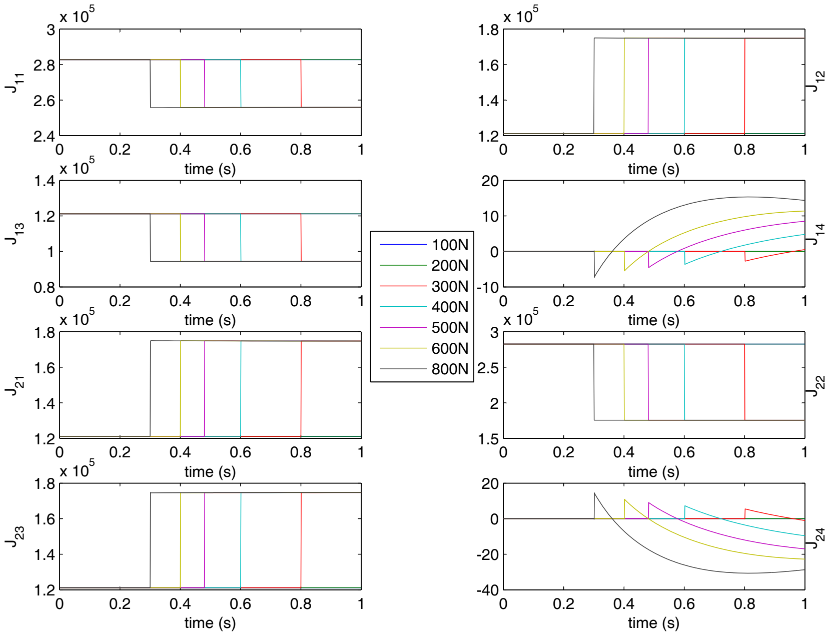


Figure 5.9: The value of Jacobian matrix for the simulated simple element model (J_{11} to J_{24})

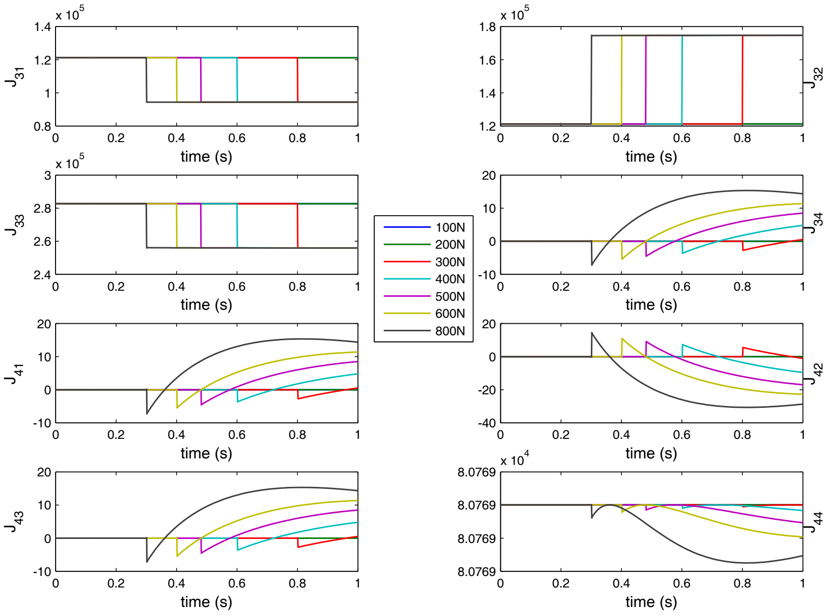


Figure 5.10: The value of Jacobian matrix for the simulated simple element model (J_{31} to J_{44})

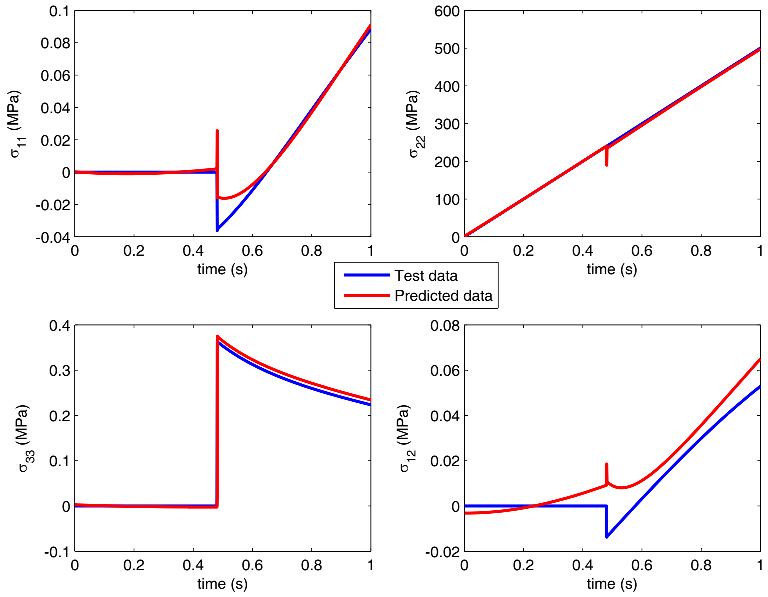


Figure 5.11: The actual and predicted stress vs time curves for integration point 1

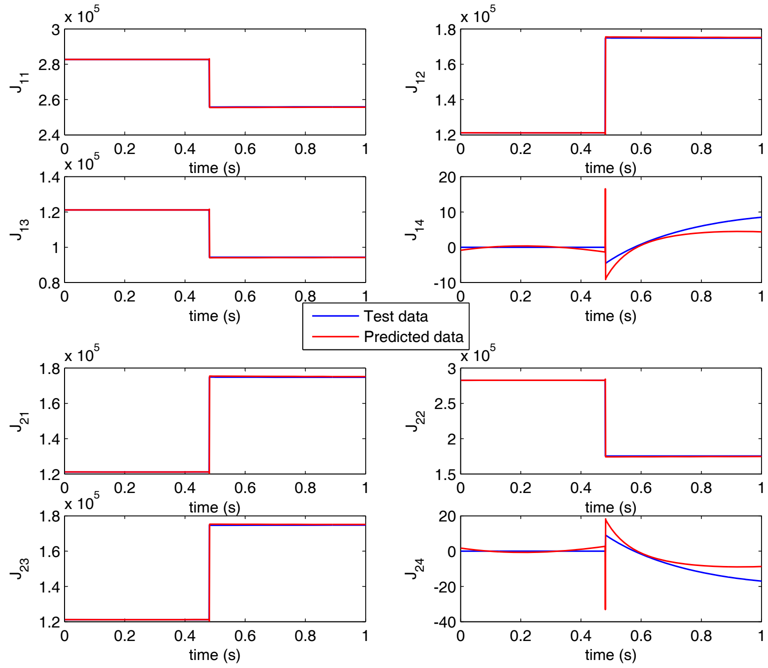


Figure 5.12: The actual and predicted values of the Jacobian matrix (J_{11} to J_{24})

values of Jacobian matrix can not be directly determined from physical tests of the material. Hence, the Jacobian matrix should not be included in the training of a data-driven model. In order to remove the Jacobian matrix from the modelling process, the modelling using dynamic explicit analysis, in which the Jacobian matrix is not required, will then be experimented.

5.4.1.2 Modelling using VUMAT - Explicit Subroutine

The ABAQUS/Dynamic explicit analysis is widely used for the damage analysis as it can easily adjust the density, bulking, mass scaling and other user defined variables during the analysis. It has the ability of deleting and changing the elements through the whole process of analysis. The explicit approach changes the status of the model element by element, so that the global stiffness of the model, which is calculated based on the Jacobian matrix of all elements, is not required in the explicit analysis. However, the accuracy of ABAQUS/Dynamic analysis highly depends on the size of time increment, the time increment must be small enough in order to make sure the result can be converged. The same simple element model as shown in Fig. 5.4, continues to be used in the experiment. The element type used in this part is CAX4R, the CAX4 element with reduced integration, this is because that the selection of element types is limited in the dynamic mode. The material density, linear and quadratic bulk viscosity parameters, and time increments are defined in Table 5.2, the values are selected in order to keep the balance between accuracy and analyzing time. The code of the sample VUMAT is recoded from the UMAT implementation in the previous part.

Table 5.2: Parameters setting in ABAQUS/Dynamic

Density (ton/mm^3)	Linear bulk viscosity	Quadratic bulk viscosity	Time increment (s)
0.01	0.06	1.2	0.0001

The exported data from ABAQUS/Explicit is shown in Fig. 5.13. The figure only shows the curves under positive top loads, the actual collected data includes the data under negative top loads and positive/negative loads from the right side. It can be seen that the stresses and strain increments are changing drastically especially in the late stage of the simulation, where is the nonlinear plastic region. The waving data is caused by the rarefaction waves that is determined by the density and bulking parameters. The waving is considered to be acceptable as the amplitudes of the waving increased after the yielding of material, which proves the results are still trustworthy.

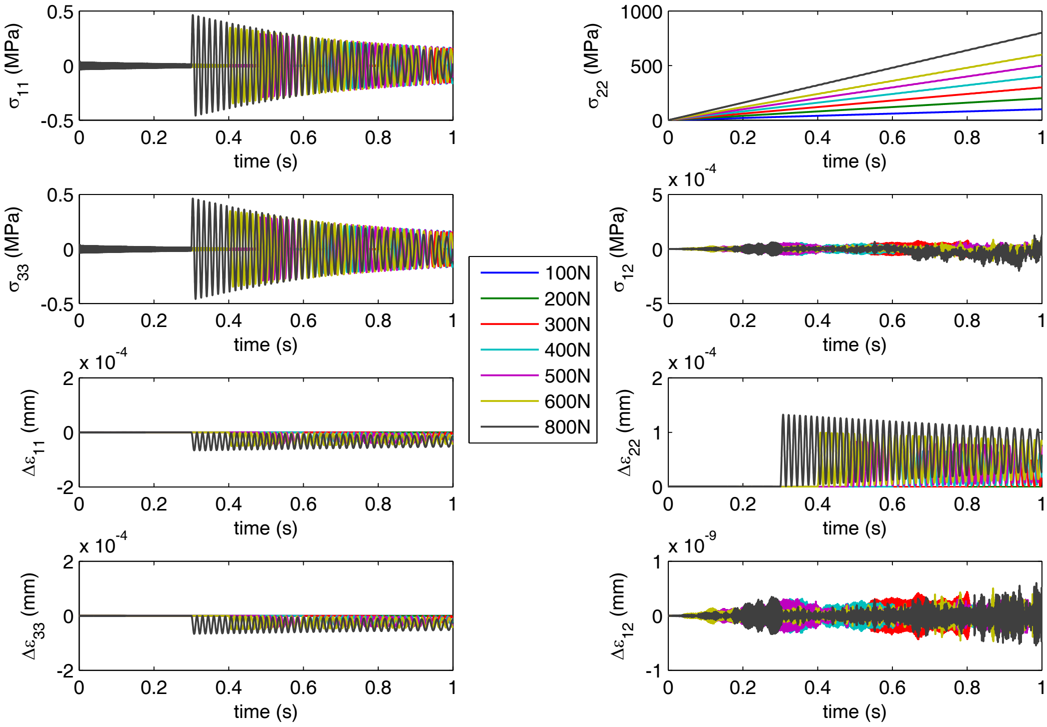


Figure 5.13: The test data exported from ABAQUS/Explicit

The training procedure is carried out using the data under ±100N, ±200N, ±400N, ±600N, ±800N top load for training and the data under ±300N top load was selected as the validation data. The +500N top load data is selected as the test data. The modelling results are shown in Fig. 5.14. It can be seen that the model performed very well in predicting all the stresses.

The trained model provided good results without the Jacobian matrix in the explicit modelling experiment. In the next section, a neural network material

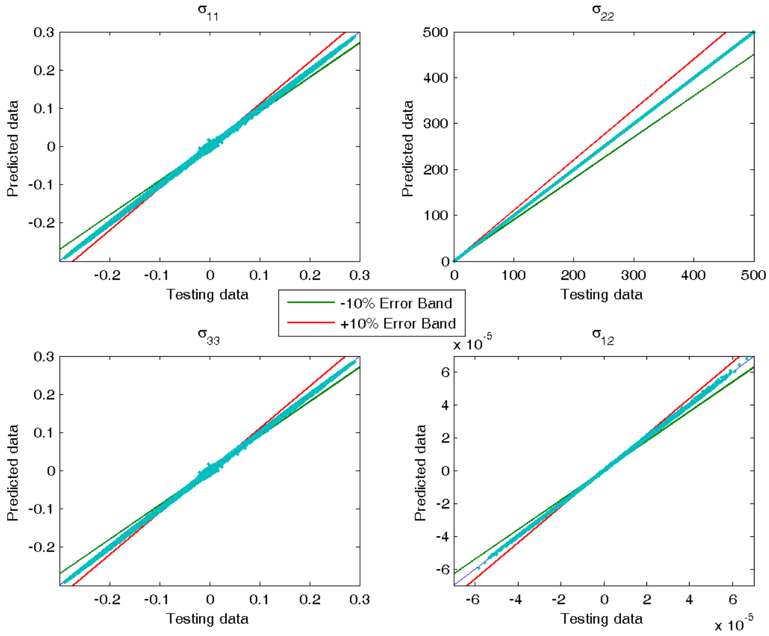


Figure 5.14: The results of modelling using ABAQUS/Explicit

definition model will be embedded into the ABAQUS environment in order to replace the constitutive equations.

5.4.2 Transfer data driven model into ABAQUS environment

The constitutive equations which define the material reactions under different situations are realized through the user subroutines, where the Fortran language are used. Hence, the model obtained in last section must be exported from MATLAB and then recoded into Fortran. The parameters of the neural network obtained in the last section is shown in Table 5.3. In order to analyse the relationship between stress and strain increments, the output in this experiment is selected as the stress increments, different from the stresses in the previous section. The predicting results of stress increments under +500N top load is shown in Fig. 5.15.

Table 5.3: Parameters setting of the neural network

Inputs	8
Hidden layer neurons	8
Outputs	4
Pre-process method	Min-Max
Hidden layer transfer function	Logarithmic Sigmoid
Output layer transfer function	Pure linear

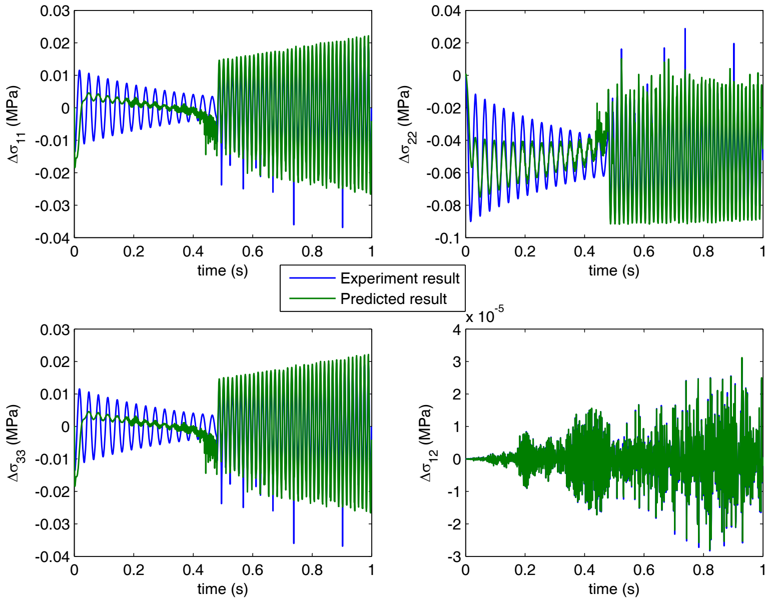


Figure 5.15: The predicting results of stress increments under +500N top load

It can be seen in Fig. 5.15 that the predicted results given by the neural network are even better than the results exported from ABAQUS. The waving amplitude of stress increments in the earlier stage of the simulation is smaller than that of the exported data. From the predicted results in the elastic linear region, the model performed very well.

By writing the input/output weights and biases and other model parameters into a data file and then reading the data file from ABAQUS Fortran subroutine, the neural network is successfully transfered into ABAQUS environment. The simulations using both the neural network and the targeted elastic-plastic isotropic hardening material model can then be carried out.

The main issue in the recoding process is that the recoded Fortran subroutine can hardly pass the data check step in the ABAQUS analysis if the training data is not selected careful, due to estimating process of time and strain increments. The increments estimating process is based on the Newton method, ABAQUS estimates and scales the time and strain increments iteratively and tries to find the increments that can converge to the given load/displacement. The model must be able to provide rational results in different scales of strain increments, a badly trained model may lead to the situation that ABAQUS fails initially in the data check process and the analysis can not be continued.

The simulation results of the ABAQUS/Explicit using user-defined constitutive equations and ABAQUS/Explicit using transfered neural network model are shown in Figs. 5.16, 5.17. In the simulation, the simple element receives a linear increasing top stretching load where the max load is 500N.

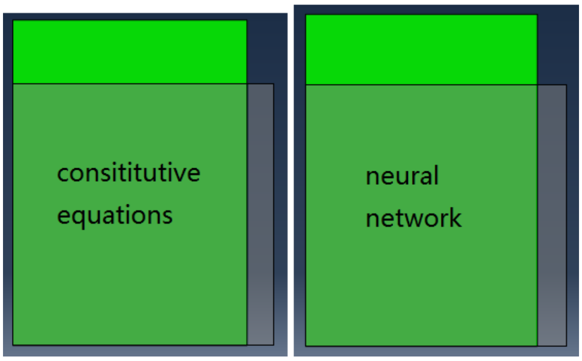


Figure 5.16: The change of 1 mm x 1 mm element shape during the simulations

It can be seen in Fig. 5.16 that the final deformation of the element is similar, where the shadowed blocks are the initial shape of the element, and the green rectangles are the deformed shape of the element. It is shown that based on the neural network, ABAQUS can obtain the correct information about the deformation directions and increments. The strain curves in Fig. 5.17 show that the embedded neural network can provide information about the elastic-plastic change, there are a turning point in ε_{11} and ε_{22} in the later propagation near 0.6s. However, the turning point obtained from the neural network material model is

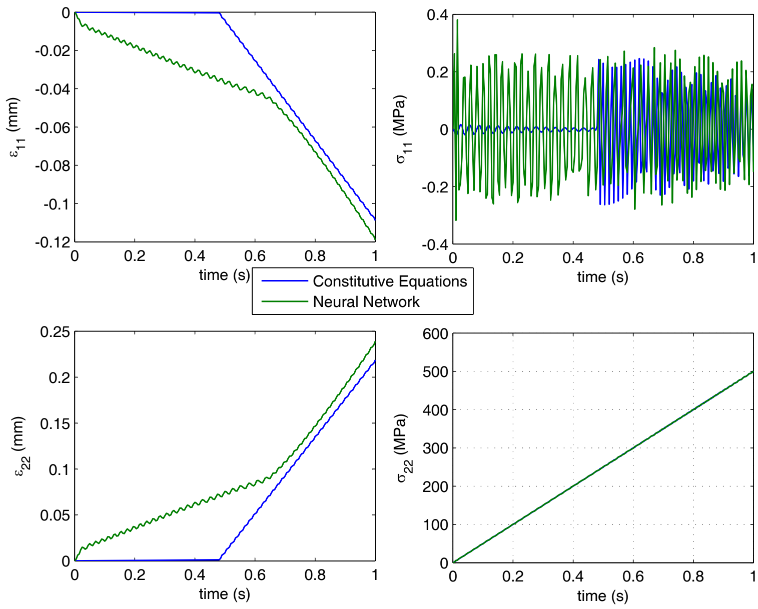


Figure 5.17: The simulation results of ABAQUS/Explicit using user subroutines

not the same as that from the constitutive equation, the elastic region of the neural network predicts much lager strain increments than that of the constitutive equation. This is because that the subroutine using constitutive equations has a yield function that can measure whether the material has reached the yield condition.

In Fig. 5.17, the stress curves on the y-axis direction σ_{22} are similar from both material models, the applied load has been simulated accurately. However, in both material models, ABAQUS cannot provide the correct information on the stresses across the other directions, the stresses in other directions are waving around zero, while the actual stress should be zero before the yield of material, and a small increasing value after the yield. The inaccuracy in the other stress directions are not caused by the material models, it is because of the less number of elements and the increments estimating algorithm in ABAUS/Explicit. The model based on constitutive equations is more stable in the elastic region compared to the neural network model.

In conclusion, the neural network model in the previous simulations can provide useful information on the deformation process. However, the performance in the linear elastic region is affected by the nonlinear plastic region during training process, which leads to a lower accuracy in the prediction of the elastic region.

In order to avoid the negative affects from the nonlinear plastic region during the model establishment, a yield measurement logic input can be added into the neural network if the yield strength of material is known. The additional input can be calculated from the comparison between the effective stress σ_e,which can be easily computed from the stresses, and the yield strength σ_0:

$$f_y = \begin{cases} 0, & \sigma_e \leq \sigma_0 \\ 1, & otherwise. \end{cases} \tag{5.3}$$

The results of the simulation using the new model which was trained with the additional input f_y are shown in Figs. 5.18, 5.19. From Fig. 5.18, it can be seen the neural network model provides the correct deformation shape. The strain and stress curves in Fig. 5.19 shows that the plastic nonlinear region now begins in the same time as the result from the constitutive equations. The new model shows the ability of providing correct material response after embedded into VUMAT.

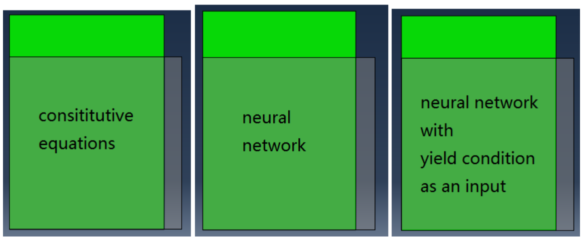


Figure 5.18: The change of element shape during the simulations

5.5 Summary

In this chapter, a data-driven model embedded finite element modelling approach (DMFEM) has been introduced. The neural network models of the isotropic

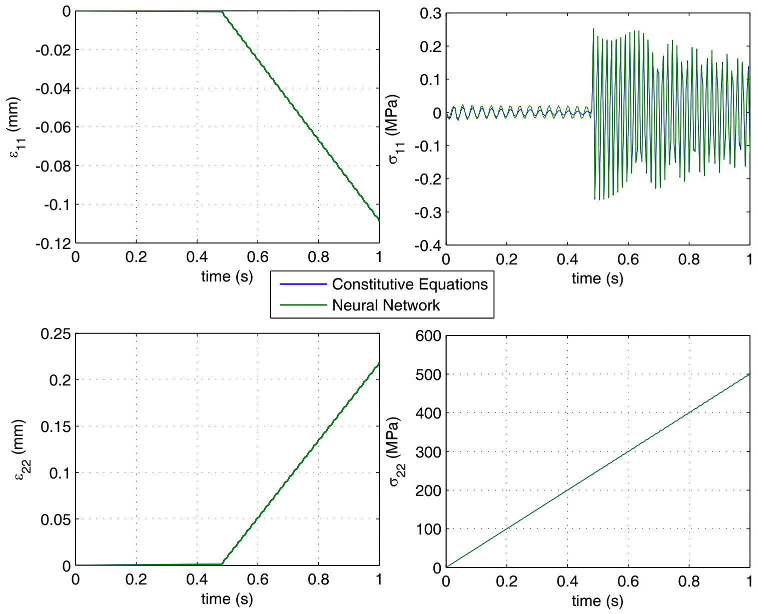


Figure 5.19: The simulation results of ABAQUS/Explicit, neural network with additional input

hardening elastic-plastic simple element model using both UMAT and VUMAT were constructed using MATLAB, the neural network models in MATLAB have performed well using the exported data.

The constructed neural network material model has then been successfully embedded into ABAQUS using the VUMAT user-subroutine in Fortran code. By adding a new input, the trained model has successfully predicted the deformation process under different load and displacement situations under ABAQUS environment. The neural network material model has proved to be able to provide the material responses under different loads.

By embedding the neural network material model in ABAQUS, the proposed DMFEM approach has been realized for the isotropic hardening elastic-plastic material. The DMFEM approach can be extended by adjusting the training data from different types of material and conditions in the future.

The next chapter will implement an optimisation process of material design based on mPSO-DHA and Finite Element Analysis.

Chapter 6

An Optimisation Process of Material Design based on mPSO-DHA and Finite Element Analysis

In Chapter 5, a data-driven approach to replacing the constitutive equations in the finite element model has been introduced. However, there are still material and geometrical parameters which are required to be identified before any simulations can be carried out. The material parameters are changing due to service loading, aging, irradiations and other conditions. In order to determine the material parameters for different materials, several techniques of finding and estimating the material parameters in certain damage conditions has been proposed in the open library.

In [Corigliano et al., 2000], an extended Kalman filter has been constructed in order to identify the suitable parameters in limited number of simulation runs. A neural network model has been introduced in [Abendroth and Kuna, 2006] to model the relationship between the process conditions and the material parameters in the simulation of a small punch test which used the GTN model for the analysis. However, these research studies were only focusing on the parameter identification for one specific material and in a fixed damage situation, the objectives of the identifications are to fit the experiment results, not to locate the optimal material parameters a priori under several given constraints.

In the following sections, the multi-objective optimisation problem in the finite element analyses will be described.

6.1 Multi-objective Optimisation and Finite Element Model

Fig. 6.1 describes the process of material design. Following the blue arrows, the general material design process as follows:

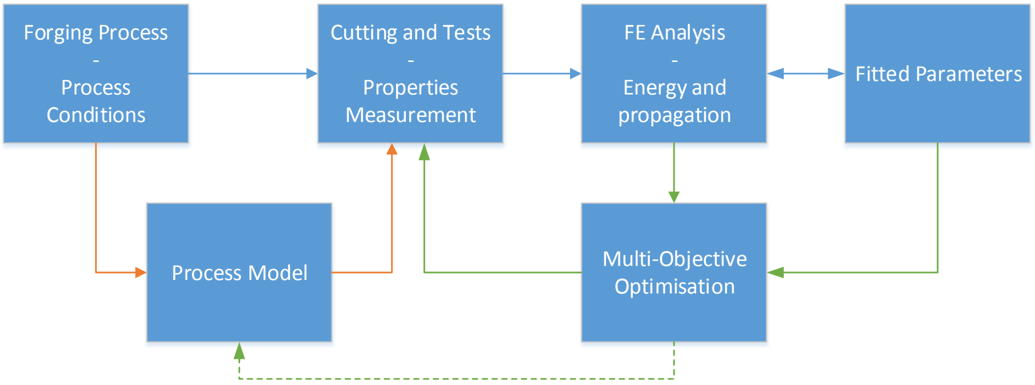


Figure 6.1: The schematic process of applying multi-objective optimisation into material design

1. Forging the material, the process conditions are defined and the hardening treatments are selected according to the expertise and the physics which are known a priori.

2. Cutting the materials into different specimens that will be used in different tests, then the material properties will be measured and estimated through series of tests.

3. Find the parameters which fit the finite element model (constitutive model) through various methods, then, the simulations on different types of damages can be carried out.

Some previous works have been published on building the models between the process conditions and the material properties, and the red arrows in Fig. 6.1 can conclude these researches. However, the finite element model has been seen as only as an analysis tool which analyzes the behaviour of an existing material, describes the changes of a failure that has already occurred, or predicts the properties according to the fitted material parameters [Balaraman et al., 2006]. [Haslinger and Neittaanmäki, 1988] used the finite element model as a reference in the designing the shapes of the material, however, the optimal design (not identification) of material characteristics using finite element models has not been found in the open libraries.

It is common that before the design of new materials, the model of the relationship between process conditions and compositions has been built in advance (red arrows in Fig. 6.1). The material parameters are then easily predicted before any specimen of the material is produced in reality. However, damage experiments are required in order to ascertain that how the failure would be propagated under certain conditions. It would be a great advantage if one can find out the material parameters which would lead to the appropriate damage process, hence, the experiments are only required for a reduced number of specimens that has the relative material parameters. Moreover, the number of experiments can be reduced and the cost of design can be lowered.

For example, for a series of material which is developed based on the same elements but different ratios, the Poisson's ratio ν and the Young's modulus E (elastic parameters) can be determined or computed before the design of material. If the finite element model is based on the Gurson model, the parameters for the damage model can be identified through several simple tests such as tensile test or small punch test (SPT). Thus, the remaining parameters to be determined are the hardening parameters, i.e. plastic characteristics, in the Gurson model case, these parameters are the initial yield stress σ_0, and changing of yield strength above σ_0.

In the above description, the elastic parameters and the damage model parameters can be transfered into different scenarios of damages and specimens [Cuamatzi-Melendez and Yates, 2009]. However, the hardening parameters can be different in different temperatures, or after certain strengthening process. Hence, one can only identify the parameters for the material if different hardening process was carried out, moreover, the damage propagation cannot be known until new experiments are set up and processed.

Figure 6.2 shows the sketch map of two different load-displacement curves under a small punch test using different hardening processes, where the red and blue strain-stress curves in the left graph lead to the red and blue load-displacement curves respectively. It can be seen different hardening properties will lead to different damage propagations. Although the two specimens have the same initial yield stress (360MPa), the failure process, especially the maximum load, is affected by different hardening behaviors.

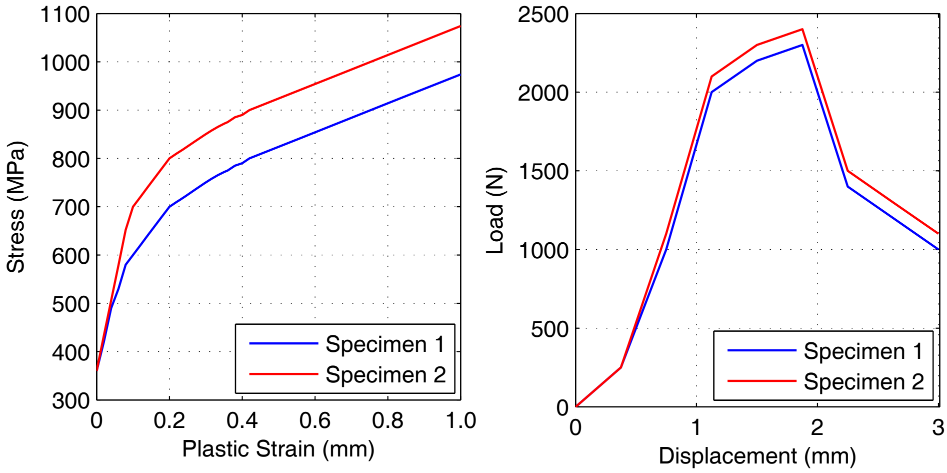


Figure 6.2: Different hardening properties will lead to different damage propagations

In the real world, people would want to know the characteristics of a material a prior before using it. It is necessary to develop a method that can provide the suitable hardening parameters that will lead to the expected damage behaviors under certain situations. Therefore, one can treat the material on purpose using necessary strengthening methods which will generate certain hardening behavior (stress-strain behavior). Since the models between strengthening process and the hardening behavior can be constructed in many cases [Kim et al., 2012] and the tests of determining the material hardening parameters are well developed, the cost of setting new experiments especially for the target damage situation can be reduced.

The procedure of finding suitable material parameters which can lead to the appropriate failure can be deemed as an optimisation process (green arrows in Fig. 6.1). There could be several objectives that the user will expect from the damage process, for example, the target material may be expected to have a certain yield strength and in the meantime it will crack only with certain amount of deformation. Therefore, the process can be a multi-objective optimisation problem.

Fig. 6.3 shows the process of finding the parameters that would lead to the appropriate damage propagation, when the material will yield after a certain value of strength and the crack will only occur after a minimum amount of deformation and before a maximum amount of deformation. It can be seen that the procedure has the same components as an optimisation problem: the estimation of parameters, the evaluation process, the measure of performance, and predefined goals and constraints.

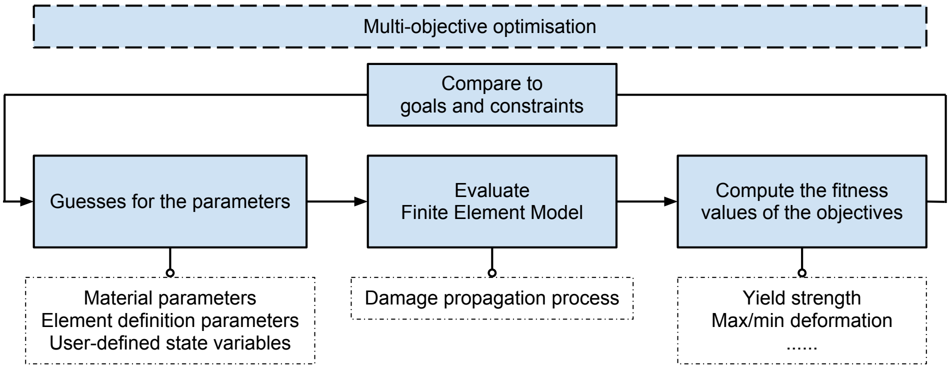

Multi-objective optimisation
Compare to goals and constraints
Guesses for the parameters
Evaluate Finite Element Model
Compute the fitness values of the objectives
Material parameters
Element definition parameters
User-defined state variables
Damage propagation process
Yield strength
Max/min deformation
......

Figure 6.3: Multi-objective optimisation problem in finite element analysis

To apply the multi-objective optimisation algorithm into the material design procedure, a small punch test finite element model will be modeled and analyzed together with the mPSO-DHA algorithm which is described in Chapter 3.

The procedure of the optimisation will combine ABAQUS and MATLAB together. For each evaluation of the objectives, the estimated solutions will firstly be written into the input files of ABAQUS, ABAQUS will run in command mode and record the results of the simulation into text files through the Python script. The mPSO-DHA will load the results from the text file and continue the iterating process.

6.2 Optimisation of Failure Behaviour using mPSO-DHA

6.2.1 The small punch test

A small punch test is a test designed to measure and analyse the properties of new and in-service components. The typical specimen for a small punch test is a disk like specimen (diameter D=8 mm and thickness t=0.5 mm) being deformed in a miniaturized deep drawing experiment. The specimen is clamped between a die (bore diameter d=4 mm, die edge radius r=0.5 mm) and a down-holder, and centrically deformed by the punch with a spherical head (radius R=1.25 mm), as shown in Fig. 6.4. The measurements are commented in Fig. 6.4, it should be noted that the dashed lines are a sketch map of the deformed shape, not the actual deformation.

The finite element model for the small punch test includes only half of the specimen since the test specimen and performance are axisymmetric. Fig. 6.5 shows the finite element model of the small punch test. The size of the 2D mesh of the specimen is set as the same as the sketch map, where the elements have a size of 0.1×0.1 mm.

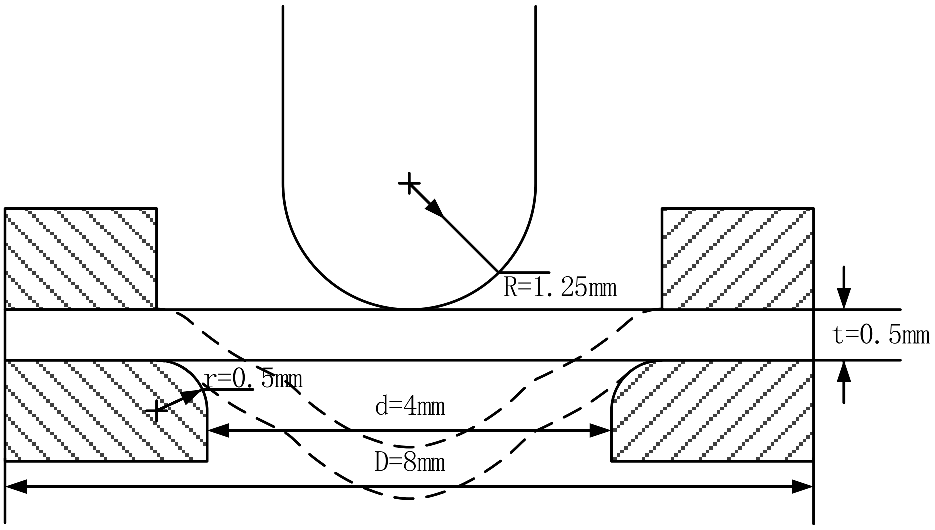


Figure 6.4: A sketch plot of the small punch test

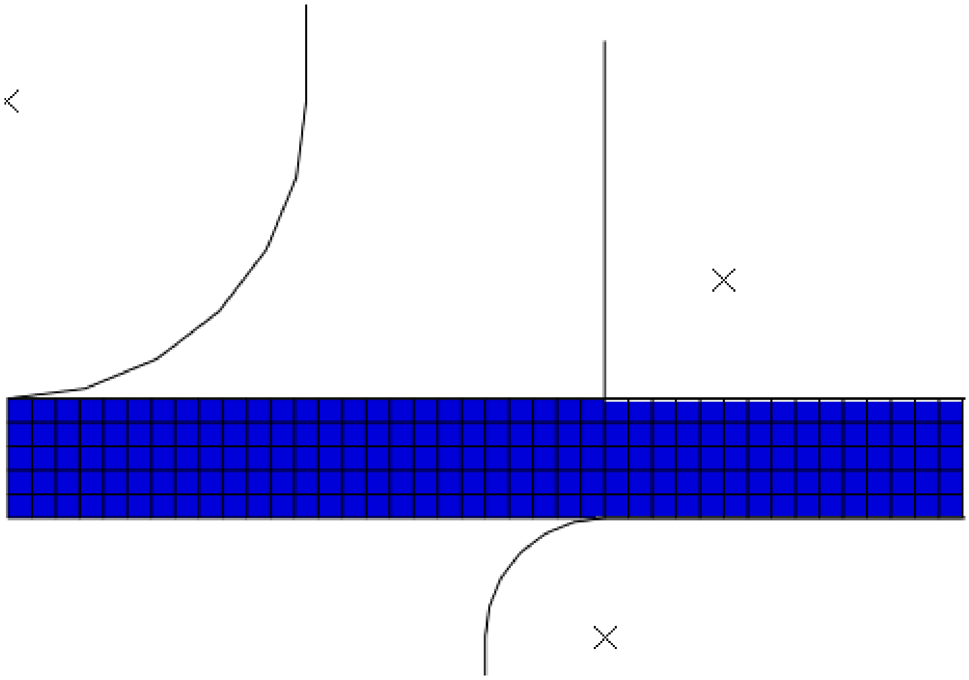
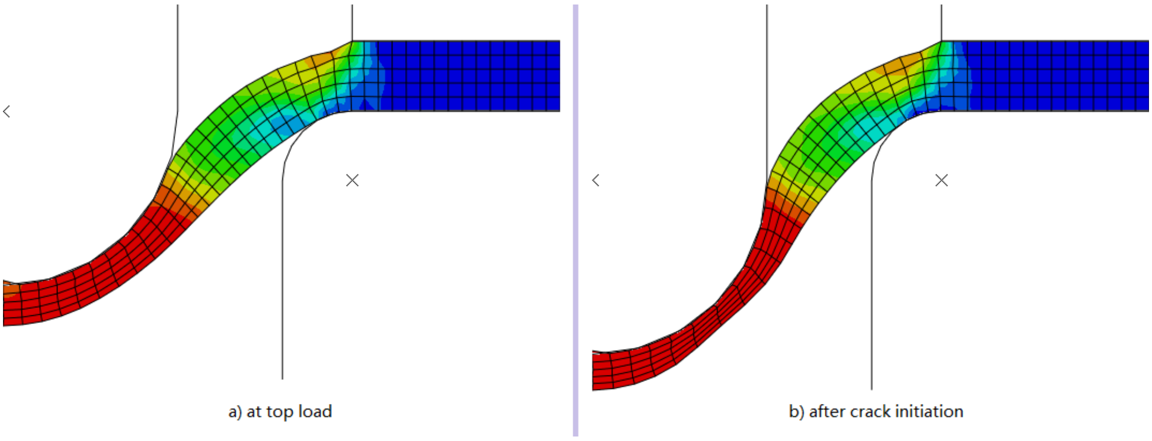


Figure 6.5: The finite element model for the small punch test

In Fig. 6.6, two states of the specimen during the simulation are shown, where the left part is the state at maximum load, and the right is after the crack initiation. The necking area of the specimen during damage propagation can be



a) at top load

b) after crack initiation

Figure 6.6: The finite element simulation of a small punch test

easily observed. During the simulation, the load displacement curve can be calculated, from which one can extract the information about the material properties and how the damage will happen under certain situations. For example, [Eto et al., 1993] found that the maximum force is correlated with the ultimate tensile strength.

There are several parameters which will affect the results of the simulation, the elastic parameters E (Young's Modulus) and ν (Poisson's Ratio), the plastic parameters σ_0 (initial yield stress) and σ^* (max yield stress when plastic strain is 1), and the constitutive parameters q_1, q_2, q_3, f_c and f_F for the GTN constitutive model. By adjusting the material parameters, the final results of when and what is the maximum load can be changed.

For example, using the parameters set in Table 6.1, the load-displacement curve of the simulation is shown in Fig. 6.7, where the red cross is the maximum load point which in this run is (1.8158,1980.5). This means the material can provide up to 1980.5 N resistant load and the crack will initiate after the displacement reaches 1.8158 mm.

The maximum load and the corresponding displacement reflect the ductility and the strength to some extent. A simulation that leads to a lower maximum load and a larger displacement at the maximum load point means the material

Table 6.1: Example of finite element model parameters setting

E (GPa)	ν	σ_0 (MPa)	σ^* (MPa)	q_1	q_2	q_3	f_c	f_F
199	0.3	380	1000	1.5	1.05	2.25	0.0017	0.005

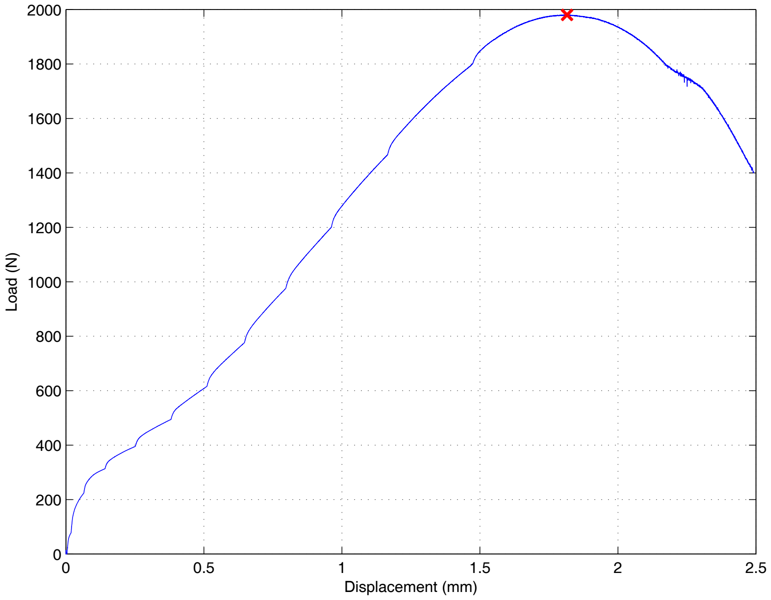


Figure 6.7: The load-displacement curve of the small punch test

has good ductility, and vice versa. In the design of new materials, maintaining the ductility and improving the strength may be conflict with each other, in which case, the maximum load and corresponding displacement can be set as the objectives of the optimisation in order to find the material that fits the desired purpose.

In this part, the final maximum load and the corresponding displacement will be optimised based on different elastic/plastic parameters using mPSO-DHA. For the convenience of description, F_m and u_c will be used to represent the maximum load and the corresponding displacement, respectively.

6.2.2 Minimize F_m and u_c by varying elastic parameters

The approximate elastic parameters for most of the mainstream materials are open and known, so that one can easily identify which material to be used if the elastic parameters are given. The first experiment is focused on finding the optimal elastic parameters that will minimize both the maximum load F_m and the corresponding displacement u_c. This resulting parameters of this experiment would lead to a very fragile material. In this case, the optimisation problem is very simple, that is:

$$\begin{cases} Minimize\ F_m \\ Minimize\ u_c \end{cases} \qquad (6.1)$$
$$\text{subject to } E \in [150, 400], \nu \in [0.2, 0.4]$$

The mPSO-DHA is set to have 10 of populations and 40 generations, the grid number in each dimension is set to 10. The other parameters for mPSO-DHA is set the same as it is in Chapter 3. $E \in [150, 400]$ and $\nu \in [0.2, 0.4]$ are set as the varying range of the elastic parameters. The other parameters are set as the same as in Table 6.1.

The refined non-dominated results of 3 independent optimisation runs are shown in Fig. 6.8, the parameters are listed in Table 6.2. It can be seen that in the simulation of small punch test, the changing of the elastic parameters has no great affect on F_m, which is mostly determined by the plastic property of the material. The displacement at the top load, u_c, can be slightly lowered by changing the elastic parameters.

From the results of Fig. 6.8, the optimal E and ν can be selected as (300.80, 0.3398) that will lead to the smallest $u_c = 1.6946$ mm and decent top load $F_m = 1981.87$ N. The load-displacement curve of the chosen E and ν is shown in Fig. 6.9. The known material that has the closest elastic property is Molybdenum. It is known that Molybdenum's E is around 329 GPa and ν is approximately 0.31, thus, one can then looking for the suitable material in the variations of

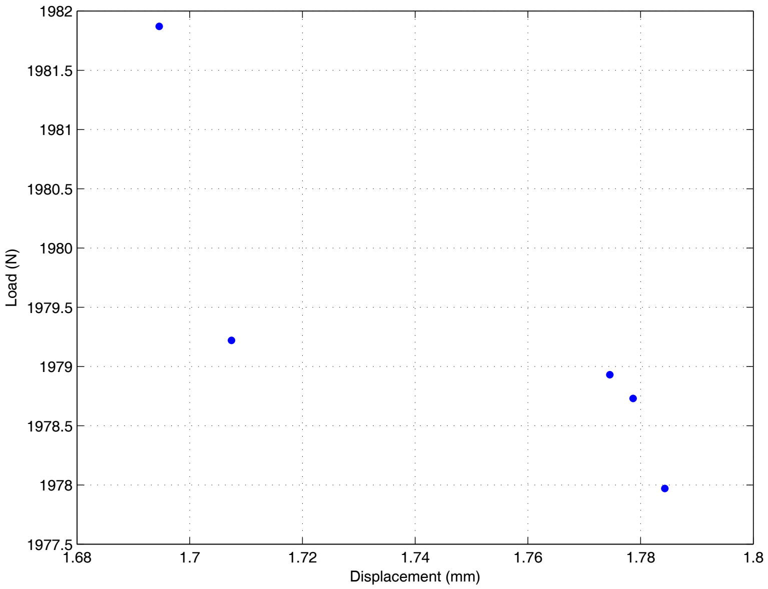


Figure 6.8: The optimal results of minimizing F_m and u_c by changing E and ν

Table 6.2: The optimal values of elastic parameters that minimizes F_m and u_c

Solutions	E (GPa)	ν	u_c (mm)	F_m (N)
1	317.86	0.33	1.77	1978.73
2	300.80	0.33	1.69	1981.87
3	278.46	0.37	1.78	1977.97
4	278.69	0.36	1.70	1979.22
5	284.76	0.36	1.77	1978.93

Molybdenum alloys.

6.2.3 Maximize F_m and u_c by varying plastic parameters

In some situations of the pipeline or metal components design, only a few types of steel or other kinds of metal are available, hence, the elastic parameters can not be changed by the engineers. In this case, the designer may use different heat-treatment or other hardening techniques that will change the material's plastic characteristic, e.g. σ_0 the initial yield stress and σ^* the yield stress,so that

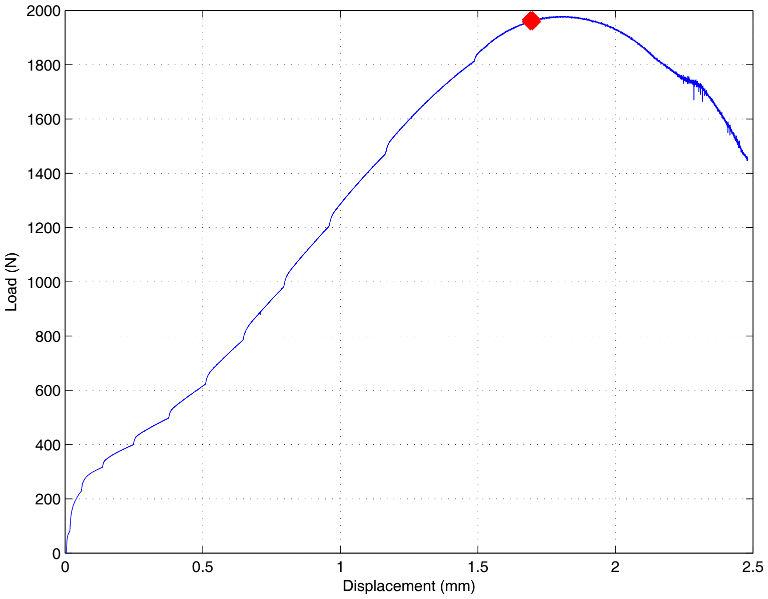


Figure 6.9: The load-displacement curve for $E = 300.80$ and $\nu = 0.3398$

the plastic properties can be changed to give different strain-stress curves. This section aims to find the optimal plastic parameters that will maximize the top load F_m and the corresponding displacement u_c. The objectives are given as:

$$\begin{cases} Maximize\ F_m \\ Maximize\ u_c \end{cases} \tag{6.2}$$
$$\text{subject to } \sigma_0 \in [200, 700], \sigma^* \in [800, 1100]$$

The mPSO-DHA is set to have 5 of populations and 40 generations, the grid number in each dimension is set to 10. The other parameters for mPSO-DHA is set the same as it is in Chapter 3. $E = 200\ GPa$ and $\nu = 0.3$ are set, which is the standard values for steels. $\sigma_0 \in [200, 700]$ and $\sigma^* \in [800, 1100]$ are defined as the variation range of the plastic parameters.

The refined non-dominated results of 3 independent optimisation runs are shown in Fig. 6.10. Part of the optimal parameters is listed in Table 6.3. It can be seen that in the simulation of small punch test, the changing of the plastic parameters may lead to vastly different damage propagations. The value of F_m

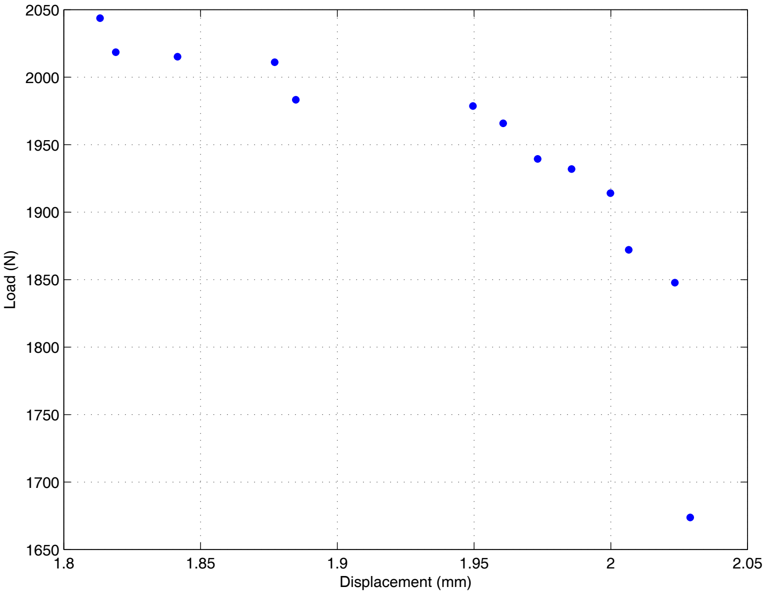


Figure 6.10: The optimal results of maximizing F_m and u_c by changing σ_0 and σ^*

Table 6.3: The optimal values of plastic parameters that maximizes F_m and u_c

Solutions	σ_0 (MPa)	σ^* (MPa)	u_c (mm)	F_m (N)
1	656.69	1087.31	1.81	2043.70
2	609.90	1088.02	1.81	2018.41
3	627.36	1076.82	1.84	2015.22
4	561.15	1100.01	1.87	2011.03
5	544.98	1087.65	1.88	1983.34
6	491.71	1100.07	1.94	1978.65
7	473.76	1097.03	1.96	1965.98
8	430.89	1091.49	1.97	1939.42
9	424.04	1088.23	1.98	1931.90
10	364.47	1090.32	1.99	1914.03
11	340.02	1070.67	2.00	1872.17
12	322.02	1058.99	2.02	1847.89
13	263.49	964.06	2.02	1673.82

ranges from 1673.8 to 2043.7 N which has a increase of 22.1%, the value of u_c is from 1.8133 to 2.0291 mm (11.9% larger than 1.8133). From the listed parameters, it is observed that the lower initial yield stress will lead to the smaller F_m and the larger u_c, while the σ^* rarely affects the result.

According to the optimal yield strengths σ_0 and σ^*, the user can identify the solutions that suits theirs need and choose the necessary hardening technique that will be applied onto the material.

6.2.4 Maximize/minimize F_m and u_c by varying elastic and plastic parameters

When designing a component, one may want to know how different the component will behave in certain circumstances by using different materials. In this part, both the maximization and the minimization processes will be carried out simultaneously in order to find the feasible space of the small punch test damage behavior. The elastic and plastic parameters are changing at the same time in the optimisation. The objectives of the problems are defined as:

$$
\begin{aligned}
&Maximize\ /\ Minimize\ (F_m, u_c),\\
&\text{subject to} \begin{cases} E \in [150, 400] \\ \nu \in [0.2, 0.4] \\ \sigma_0 \in [200, 700] \\ \sigma^* \in [800, 1100] \end{cases}
\end{aligned}
\tag{6.3}
$$

Once again, the mPSO-DHA is set to have 5 of populations and 40 generations, the grid number in each dimension is set to 10. The other parameters for mPSO-DHA is set the same as it is in Chapter 3. $E \in [200, 400]$, $\nu \in [0.2, 0.4]$, $\sigma_0 \in [200, 700]$ and $\sigma^* \in [800, 1100]$ are defined as the variation range of the elastic and plastic parameters.

The refined non-dominated results of 3 independent optimisation runs for both

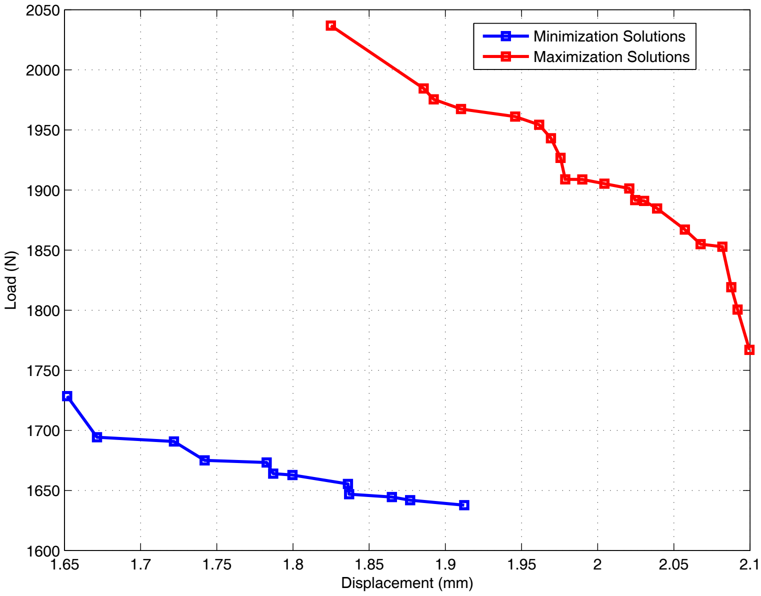


Figure 6.11: The optimal results of maximizing and minimizing F_m and u_c simultaneously by changing elastic and plastic parameters

Table 6.4: Part of the optimal values of material parameters that minimizes/-maximizes F_m and u_c

Solutions	E (GPa)	ν	σ_0 (MPa)	σ^* (MPa)	u_c (mm)	F_m (N)
1 (min)	267.93	0.34	618.61	895.04	1.65	1728.55
2 (min)	264.74	0.31	561.03	892.06	1.72	1690.83
3 (min)	281.17	0.30	532.02	893.27	1.78	1673.34
4 (min)	276.95	0.30	486.19	893.00	1.83	1646.96
5 (min)	274.87	0.28	413.66	910.06	1.91	1637.83
6 (max)	198.95	0.29	655.09	1083.92	1.82	2036.88
7 (max)	226.91	0.31	516.32	1086.91	1.91	1967.40
8 (max)	206.82	0.29	486.94	1086.62	1.96	1954.33
9 (max)	238.86	0.29	415.81	1075.93	1.97	1980.94
10 (max)	255.81	0.30	343.65	1085.85	2.02	1901.36
11 (max)	266.17	0.30	292.00	1065.25	2.08	1852.84
12 (max)	197.27	0.28	200.00	1031.43	2.09	1767.03

maximization and minimization are shown in Fig. 6.11, the optimal solutions are listed in Table. 6.4. It can be seen that a feasible region of the possible F_m and u_c could be estimated from the solutions which has been found: the maximum load will be from 1600 N to 2050 N, the crack will initiate at the displacement from 1.6mm to 2.1mm, approximately.

It should be noted that the feasible region in Fig. 6.11 could be extended and completed by increasing the maximum generation number of mPSO-DHA, which will lead to a longer optimisation time and a more specific feasible region.

6.2.5 Finding the optimal material parameters for desired F_m and u_c

$$\begin{aligned} &Minimize \begin{cases} J_1 = (\frac{F_m - F_m^*}{F_m^*})^2 \\ J_2 = (\frac{u_c - u_c^*}{u_c^*})^2 \end{cases} \\ &\text{subject to} \begin{cases} E \in [150, 400] \\ \nu \in [0.2, 0.4] \\ \sigma_0 \in [200, 700] \\ \sigma^* \in [800, 1100] \end{cases} \end{aligned} \tag{6.4}$$

Based on the results that were found in the previous section, the feasible region of the crack propagation is known for the small punch test simulation. Now, people may select the F_m and u_c that suit their needs, and use the same optimisation process to find the correspond optimal material parameters. Hence, the objectives of the new problem can be described as in Eq. 6.4, where F_m^* and u_c^* are the targeted values of F_m and u_c, J_1 and J_2 are two objective functions that represents the proximity of the solutions.

Fig. 6.12 shows the solutions that are found by the mPSO-DHA when the target F_m^* is set to 1800N and the u_c^* is 1.9mm. The results are the refined non-dominated solutions of 3 independent optimisation runs, where the population of particles is 5 and the generation number is 40, the hypercubes are 10x10. The corresponding material parameters are listed in Table 6.5. The results show that the algorithm found the optimal material parameters with no great errors: the solutions are all located by the x and y axises, and the solution near the origin point (2.194E-07, 4.694E-08) leads to very small values for both J_1 and J_2.

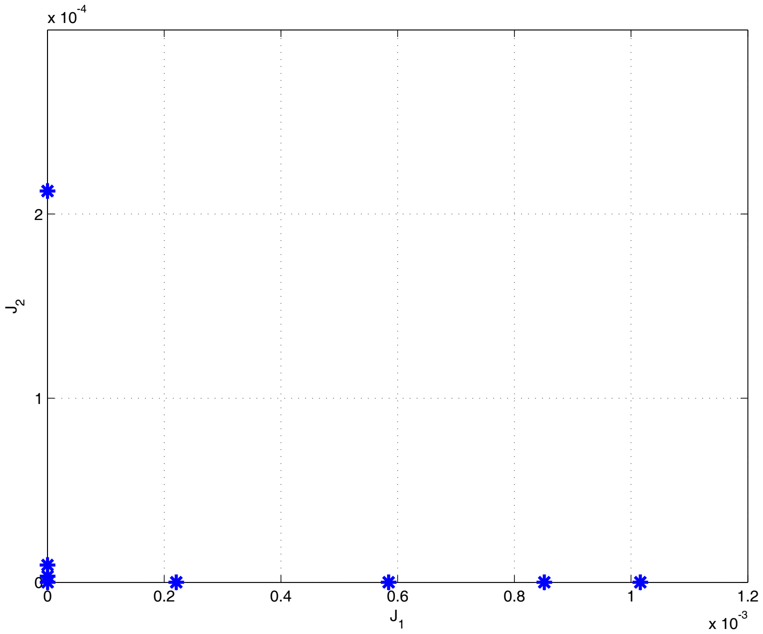


Figure 6.12: The optimal J_1 and J_2 for the desired $F_m^* = 1800$ N and $u_c^* = 1.9$ mm

Table 6.5: The optimal values of material parameters for $F_m^* = 1800$ N and $u_c^* = 1.9mm$

Solutions	E (GPa)	ν	σ_0 (MPa)	σ^* (MPa)	J_1	J_2
1	364.03	0.34	412.11	1009.91	1.862E-07	3.280E-06
2	364.11	0.33	412.11	1009.91	8.514E-04	1.975E-09
3	364.52	0.34	412.45	1009.91	5.846E-04	1.000E-08
4	368.86	0.33	414.73	1011.05	1.000E-03	2.777E-10
5	325.66	0.34	491.37	1003.39	3.202E-08	2.125E-04
6	323.21	0.35	472.63	994.66	2.207E-04	2.777E-08
7	326.52	0.34	477.02	989.58	5.121E-08	9.404E-06
8	300.07	0.36	487.40	987.24	2.194E-07	4.694E-08

From the optimal parameters in Fig. 6.12, the users can choose which kind of material is to be used and treated for the desired component. The final value of the material parameters can be selected directly from the solutions, or a set of computed parameters, for example, the average of all the solutions.

Fig. 6.13 shows the load-displacement curve for the finite element simulation of the small punch test when the eighth solution listed in Table 6.5 is decided as the final solution. It can be seen that the $F_m = 1799.6$ N and $u_c = 1.8991$ mm have only 0.02% and 0.05% differences from the targeted value $F_m^* = 1800$ N and $u_c = 1.9$ mm. Therefore, the user may look for the material which has the elastic properties $E = 300.07$ GPa and $\nu = 0.36$. These elastic parameters may be found in some of the Aluminum alloys and Copper alloys, once the material is forged, necessary treatments are needed in order to achieve the target plastic properties: $\sigma_0 = 487.40$ MPa and $\sigma^* = 987.24$ MPa.

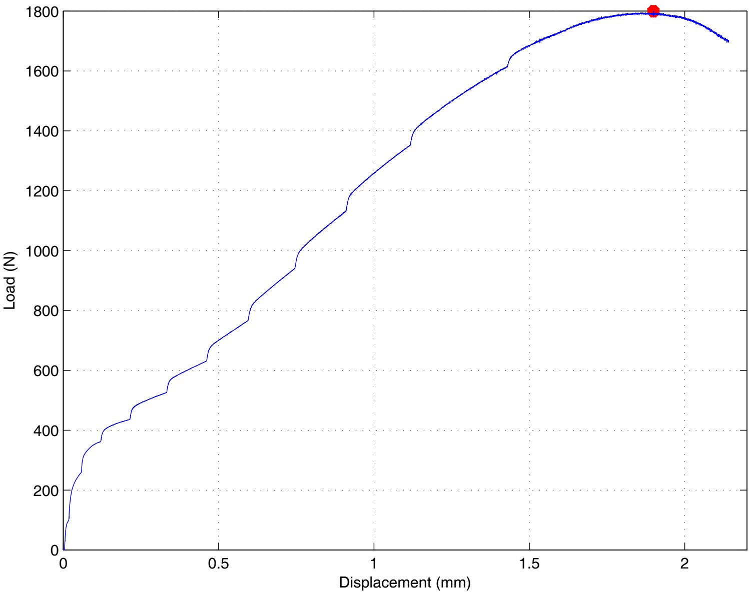


Figure 6.13: The load-displacement curve using the eighth optimal parameter set

6.3 Summary

In this Chapter, the mPSO-DHA has been successfully applied into the optimal design of small punch test.

Several different optimisations were carried out for the small punch test in order to find the optimal solutions for different situations of material design. The optimisation process was applied for: a) minimization of F_m and u_c by only varying the elastic parameters; b) maximization on F_m and u_c by only varying plastic parameters; c) minimizing and maximizing (F_m, u_c) simultaneously by varying elastic and plastic parameters; and d) finding the optimal material parameters for a desired damage behavior. In Section 6.2.5, the proposed method of finding optimal material parameters that would lead to the specific damage behavior proved to be efficient.

The next chapter will conclude the thesis, and the future work of this project will be discussed as well.

Chapter 7

Conclusions and Future Work

7.1 Summary of Thesis

In this project, a multi-objective particle swarm optimisation algorithm (mPSO-DHA) has been proposed; the modelling of steel crack propagation has been carried out using both fuzzy and neural-network; an error compensation scheme based on Gaussian mixture model has been developed together with the model of steel crack propagation; a data-driven model embedded finite element modelling (DMFEM) approach has been introduced; a tool for designing new materials has been assembled based on mPSO-DHA and DMFEM.

The first Chapter targeted the research aims of this project, it also listed the structure of this thesis and main contents of each chapter.

Chapter 2 introduced the background knowledge relating to the project. The introduction for Genetic Algorithms [Holland, 1975], Evolutionary Strategies [Rechenberg, 1973; Schwefel, 1977] and Particle Swarm Optimisations [Kennedy and Eberhart, 1995] were presented, based on the single objective optimisation methods, the algorithms for Multi-Objective Optimisation were also introduced. Then, Artificial Neural Networks, Fuzzy Logic and Fuzzy Systems were reviewed as they are widely used intelligent modelling methods. The Finite Elements Method, which is developed to model the materials through the “white-model” approach, was briefed as well.

In Chapter 3, a multi-objective particle swarm optimisation algorithm (mPSO-DHA) has been described. The algorithm successfully extended the original particle swarm optimisation using dynamic hyper-cube archiving, weight adaptation, mutation operator, pool selection and modified global best selection. The best parameter settings for the algorithm were found through experiments. The application of mPSO-DHA to the well-known multi-objective benchmark functions, including ZDT [Zitzler et al., 2000] series and DTLZ [Deb et al., 2002] series, were analysed and compared with other multi-objective PSO algorithms. The performance of the proposed algorithm proved to be superior to the compared multi-objective PSOs. The comparisons with other evolutionary algorithm, such as PAES [Knowles and Corne, 1999], SPEA [Zitzler and Thiele, 1998] and NSGA2 [Deb et al., 2000], showed that mPSO-DHA has the advantage of solving complex multi-objective optimisation problems.

Chapter 4 reported on the modelling of steel crack propagation using fuzzy and neural-network, using the data acquired from previous work of [Ayvar et al., 2005]. A data analysis was carried out firstly. The fuzzy modelling for crack propagation was then detailed, using the method which has been used in [Zhang and Mahfouf, 2008, 2011] with hierarchical clustering initialization and gradient decent learning. After the fuzzy modelling, the double-loop neural network [Yang et al., 2003] has been selected to modelling the same data. A comparison between the fuzzy model’s and neural network’s performance led to the conclusion that both models can predict the crack in decent accuracy, the fuzzy model performed slightly better in the high energy region. In order to reduce the error of models, an error compensation approach was then proposed and examined, the experiments showed that the proposed GMM [McLachlan and Peel, 2004] error compensation method provided useful information about error compensations and confidence bands.

Chapter 5 has introduced a data-driven model embedded finite element modelling approach (DMFEM). The idea of this approach was to replace the constitutive equations in the finite element model by a data-driven model. The neural network models of the isotropic hardening elastic-plastic simple element model using both UMAT and VUMAT were constructed using MATLAB, and the experiments using the simple element model verified the feasibility of the approach. The constructed data-driven model was then successfully embedded into ABAQUS using the user-subroutine in Fortran code. The proposed DMFEM has proved to be able to provide material response under different loads.

In Chapter 6, an optimisation process of material design based on mPSO-DHA and finite element analysis has been proposed and evaluated. The mPSO-DHA was applied on finite element models on material failures in order to find optimal material properties which will lead to the desired damage propagation process. The optimisation based on different scenarios and design objectives for the small punch test has been carried out, where the optimal values of elastic and plastic characteristics has been estimated for minimization, maximization and specific value of the maximum load and corresponding displacement. The feasible space of how the damage will behave under small punch test was located. Moreover, the optimal material parameters for a specific damage behavior were successfully identified.

7.2 Conclusion

In this research, a new multi-objective particle swarm optimisation algorithm mPSO-DHA has been proposed. The integration of dynamic hyper-cube archive, weight adaptation, mutation, pool selection and modified global best selection techniques has enhanced the multi-objective optimisation ability of traditional particle swarm optimisation algorithms. The comparison with other multi-objective optimisation algorithms has showed the algorithm is effective in optimising various problems.

However, in order to assure the generality of mPSO-DHA, it is neccessary to test mPSO-DHA using higher-dimension problems and newly developed benchmark functions. The application in optimising the structure of data-driven models has not been explored in this project.

Models with GMM error compensation structure of the crack propagation have been established. The Gaussian mixture error compensation structure has showed the ability of providing the confidence band and a probable error for the predicted results from both fuzzy and neural network model. The prediction error has been reduced through embedding the error compensation structure into previously built model without rebuilding the whole model.

It should be commented that the GMM error compensator can only provide reliable error information when the training data for GMM reflects the true error distribution. The compensator would not provide good error prediction while the error is unbiased.

A synergistic model combining the data-driven model and FEM has been developed. The constitutive equation model in the FEM has successfully been replaced by a neural network model. The simulation based on a simple element model using the synergistic approach has led to a good agreement with the results from the model using constitutive equation model. This synergistic modelling approach has extended the FEM to model the material when the physical expertise of the material is missing.

Yet, the synergistic model has only been built based on the simulation data of a single element. Further experiements are needed to collect real data which could be used to build new models and measure the generation properties of the proposed approach in the case of larger scale structures. The model of 3D element has not been studied in the project.

The systematic optimal design approach of metal alloys based on FEM and the proposed new multi-objective PSO has been successful applied to the simulations of a small punch test. The optimal parameters for the material have been estimated for different desired damage propagation process. The proposed optimal design approach has introduced the idea of user-oriented design into the material design procedure.

The link between micro-structure and material parameters has not been considered in this study. The reverse engineering aspect related to the approach – fitting the FEM parameters to the known experiment results – has not been studied.

7.3 Future Works

Base on the studies described in the thesis, some future works are suggested:

1. The proposed mPSO-DHA proves to be an efficient multi-objective optimisation algorithm in solving the well-known ZDT and DTLZ series benchmark functions, comparing to other state-of-art multi-objective optimisation algorithms. Other newly developed benchmark functions and real world problems is still needed in order to test the applicability of mPSO-DHA in different types of situations and/or constraints.

2. With the applications into the crack propagations of the X100 pipeline steel, the data-driven modelling approach has been successfully integrated with the Gaussian Mixture Model error compensation scheme. The error compensation method using Gaussian Mixture Model provides a stochastic adaptive process in dealing with the error of existing models. Future works should be focused on applying the method into other real world problems and existing models.

3. The data-driven model embedded within the finite element modelling approach can be extended to other materials and element types using other data-driven modelling methods and more industrial collected data. In addition, the modelling of the 3D situation should be considered in the future in order to improve the performance of DMFEM.

4. The proposed optimisation procedure of material design should not only be used in finding the optimal parameters for desired material behavior, but to also be applied into fitting the constitutive parameters of finite element model into the observed data. By setting the objectives of the optimisation

to several sample points of the load-displacement curve, or the mean error between the predicted curve and the actual experiment data, the user can find all the required parameters without tuning the constitutive parameters manually.

5. It is possible to correlate between the industrial process of forging material and the material characteristics which need to be defined priorly in the finite element model. Therefore, instead of optimising the material characteristics that will lead to a desired failure behaviour, the process conditions of forging the material can be directly optimised. The cost of design and experiments can be further reduced.

References

Martin Abendroth and Meinhard Kuna. Identification of ductile damage and fracture parameters from the small punch test using neural networks. *Engineering fracture mechanics*, 73(6):710–725, 2006.

S Acharyya and S Dhar. A complete gtn model for prediction of ductile failure of pipe. *Journal of Materials Science*, 43(6):1897–1909, 2008.

A.S. Argon, J. Im, and R. Safoglu. Cavity formation from inclusions in ductile fracture. *Metallurgical Transactions A*, 6:825–837, 1975. ISSN 0360-2133. doi: 10.1007/BF02672306.

ASTM E 1820–01. Standard Test Method for Measurement of Fracture Toughness. Technical report, West Conshohocken, PA, 2001.

S.S. Ayvar, S.H. Hashemi, I.C. Howard, and J.R. Yates. An experimental and finite element study of the ductile tearing characteristics of high-toughness gas pipeline steel. *Applied Mechanics and Materials*, 3-4:259–266, 2005. doi: 10.4028/www.scientific.net/AMM.3-4.259.

R. Babuska and H. Verbruggen. Neuro-fuzzy methods for nonlinear system identification. *Annual Reviews in Control*, 27(1):73–85, 2003. doi: doi:10.1016/S1367-5788(03)00009-9. URL http://www.ingentaconnect.com/content/els/13675788/2003/00000027/00000001/art00009.

R. Babuska, J. Oosterhoff, A. Oudshoorn, and P.M. Bruijn. Fuzzy self-tuning {PI} control of ph in fermentation. *Engineering Applications of Artificial Intelligence*, 15(1):3 – 15, 2002. ISSN 0952-1976. doi: http://dx.doi.

org/10.1016/S0952-1976(02)00003-9. URL http://www.sciencedirect.com/science/article/pii/S0952197602000039.

C.T. Back. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996. ISBN 9780195099713. URL http://books.google.co.uk/books?id=EaN7kvl5coYC.

Karthikeyan Balaraman, Sudipto Mukherjee, Anoop Chawla, and Rajesh Malhotra. Inverse finite element characterization of soft tissues using impact experiments and taguchi methods. *SAE Paper*, pages 01–0252, 2006.

HKDH Bhadeshia. Mathematical models in materials science. *Materials Science and Technology*, 24(2):128–136, 2008.

D.S. Broomhead and D. Lowe. Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

K.A. Brownlee. *Statistical theory and methodology in science and engineering*. Krieger Pub. Co., 1984. ISBN 9780898747485.

G. Buzzichelli and L. Scopesi. Fracture propagation control in very high strength gas pipelines. *Revue de Mtallurgie*, 97(11):1409–1416, 2000. doi: 10.1051/metal:2000119. URL http://dx.doi.org/10.1051/metal:2000119.

M.-Y. Chen, D. A. Linkens, D. J. Howarth, and J. H. Beynon. Fuzzy model-based charpy impact toughness assessment for ship steels. *ISIJ International*, 44(6): 1108–1113, jun 2004. ISSN 09151559. doi: 10.2355/isijinternational.44.1108.

Min-You Chen and Derek A. Linkens. A fuzzy modelling approach using hierarchical neural networks. *Neural Computing and Applications*, 9(1):44–49, 2000.

Min-You Chen and Derek A. Linkens. Rule-base self-generation and simplification for data-driven fuzzy models. *Fuzzy Sets and Systems*, 142(2):243–265, 2004.

R.W. Clough. *The Finite Element Method in Plane Stress Analysis*. American Society of Civil Engineers, 1960.

C.A. Coello Coello and M.S. Lechuga. Mopso: a proposal for multiple objective particle swarm optimization. In *Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on*, volume 2, pages 1051 –1056, 2002. doi: 10.1109/CEC.2002.1004388.

M Çöl, HM Ertunc, and M Yılmaz. An artificial neural network model for toughness properties in microalloyed steel in consideration of industrial production conditions. *Materials & design*, 28(2):488–495, 2007.

Alberto Corigliano, Stefano Mariani, and Barbara Orsatti. Identification of gurson–tvergaard material model parameters via kalman filtering technique. i. theory. *International journal of fracture*, 104(4):349–373, 2000.

Ruben Cuamatzi-Melendez and JR Yates. Transferability of the gurson damage model parameters with charpy and tensile tests with different constrain level. *Advanced Materials Research*, 65:19–31, 2009.

G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989. ISSN 0932-4194. doi: 10.1007/BF02551274.

Indraneel Das and JE Dennis. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural optimization*, 14(1):63–69, 1997.

K. Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In Marc Schoenauer, K. Deb, G. Rudolph, Xin Yao, Evelyne Lutton, Juan Julian Merelo, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature – PPSN VI*, pages 849–858, Berlin, 2000. Springer.

K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable Multi-Objective Optimization Test Problems. In *Congress on Evolutionary Computation (CEC 2002)*, pages 825–830. IEEE Press, 2002.

Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*. John Wiley & Sons, Chichester ; New York, 2008.

Kalyanmoy Deb and Ram B. Agrawal. Simulated Binary Crossover for Continuous Search Space. *Complex Systems*, 9:115–148, 1995.

Y. del Valle, Ganesh K. Venayagamoorthy, Salman Mohagheghi, J.-C. Hernandez, and Ronald G. Harley. Particle swarm optimization: Basic concepts, variants and applications in power systems. *IEEE Trans. Evolutionary Computation*, pages 171–195, 2008.

Fionn Dunne and Nik Petrinic. *Introduction to computational plasticity*. Oxford University Press New York, 2005.

Eberhart and Yuhui Shi. Particle swarm optimization: developments, applications and resources. In *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, volume 1, pages 81 –86 vol. 1, 2001. doi: 10.1109/CEC.2001.934374.

R.C. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, volume 1, pages 84 –88 vol.1, 2000. doi: 10.1109/CEC.2000.870279.

R.C. Eberhart, P.K. Simpson, R. Dobbins, and R.W. Dobbins. *Computational intelligence PC tools*. Mathematical computing. AP Professional, 1996.

R.C. Eberhart, Y. Shi, and J. Kennedy. *Swarm Intelligence*. The Morgan Kaufmann Series in Evolutionary Computation. Elsevier Science, 2001. ISBN 9781558605954.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

EnergyindustryPhotos. Gas pipeline blowout photos. http://www.energyindustryphotos.com/pipeline_blowout_photos_and_natu.htm, 2008.

Motokuni Eto, Hideaki Takahashi, Toshihei Misawa, Masahide Suzuki, Yutaka Nishiyama, Kiyoshi Fukaya, and Shiro Jitsukawa. Development of a miniaturized bulge test (small punch test) for post-irradiation mechanical property evaluation. *ASTM Special Technical Publication*, 1204:241–241, 1993.

H. Y. Fan and Y. Shi. Study of Vmax of the particle swarm optimization algorithm. In *Proc. of the Workshop on Particle Swarm Optimization*, Indianapolis, IN: Purdue School of Engineering and Technology, April 2001.

David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1st edition, 1989. ISBN 0201157675.

Jess Gonzlez, Ignacio Rojas, Hctor Pomares, Luis J. Herrera, Alberto Guilln, Jos M. Palomares, and Fernando Rojas. Improving the accuracy while preserving the interpretability of fuzzy function approximators by means of multi-objective evolutionary algorithms. *International Journal of Approximate Reasoning*, 44(1):32 – 44, 2007. ISSN 0888-613X. doi: 10.1016/j.ijar.2006.02.006.

A. Graves, M. Liwicki, S. Fernandez, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(5):855 –868, may 2009. ISSN 0162-8828. doi: 10.1109/TPAMI.2008.137.

Arthur L. Gurson. *Continuum theory of ductile rupture by void nucleation and growth. Part I, Yield criteria and flow rules for porous ductile media*. Division of Engineering Brown University, Providence, R.I., 1975. ”September 1975”.

J. Haslinger and P. Neittaanmäki. *Finite element approximation for optimal shape design: theory and applications*. Wiley, 1988. ISBN 9780471920793. URL http://books.google.co.uk/books?id=eYNRAAAAMAAJ.

J.H. Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975. ISBN 9780472084609.

John H. Holland and Judith S. Reitman. Cognitive systems based on adaptive algorithms. *SIGART Bull.*, (63):49–49, June 1977. ISSN 0163-5719. doi: 10.1145/1045343.1045373.

Xiaohui Hu, R.C. Eberhart, and Yuhui Shi. Particle swarm with extended memory for multiobjective optimization. In *Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE*, pages 193 – 197, april 2003. doi: 10.1109/SIS.2003.1202267.

T. Huang and A.S. Mohan. A hybrid boundary condition for robust particle swarm optimization. *Antennas and Wireless Propagation Letters, IEEE*, 4:112 – 117, 2005. ISSN 1536-1225. doi: 10.1109/LAWP.2005.846166.

A K Jain, M N Murty, and P. J. Flynn. Data clustering: A review, 1999.

J. S. R. Jang. ANFIS: adaptive-network-based fuzzy inference system, May 1993.

ME Karabin, F Barlat, and RT Shuey. Finite element modeling of plane strain toughness for 7085 aluminum alloy. *Metallurgical and Materials Transactions A*, 40(2):354–364, 2009.

J. Kennedy. Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance. In *Congress on Evolutionary Computation*, volume 3, pages 1931–1938, 1999. doi: 10.1109/CEC.1999.785509.

J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942 –1948 vol.4, nov/dec 1995. doi: 10.1109/ICNN.1995.488968.

JH Kim, MG Lee, F Barlat, and D Kim. Prediction of nonlinear kinematic hardening parameters for dp steels by crystal plasticity-based micromechanical analysis. *Advanced Science Letters*, 13(1):224–227, 2012.

JF Knott. Deterministic and probabilistic modelling of brittle fracture mechanisms in ferritic steels. *Fatigue & Fracture of Engineering Materials & Structures*, 29(9-10):714–724, 2006.

J. Knowles and D. Corne. The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, pages 3 vol. (xxxvii+2348), 1999. doi: 10.1109/CEC.1999.781913.

J-B Leblond, G Perrin, and J Devaux. An improved gurson-type model for hardenable ductile metals. *European journal of mechanics. A. Solids*, 14(4): 499–527, 1995.

Sou-Sen Leu and Chung-Huei Yang. Ga-based multicriteria optimal model for construction scheduling. *Journal of Construction Engineering and Management*, 125(6):420–427, 1999.

E. H. Mamdani. Application of fuzzy algorithms for control of simple dynamic plant. *Procedings of IEEE*, 121(12):1585–1588, 1974.

WarrenS. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5:115–133, 1943. ISSN 0007-4985. doi: 10.1007/BF02478259. URL http://dx.doi.org/10.1007/BF02478259.

G. McLachlan and D. Peel. *Finite Mixture Models*. Wiley series in probability and statistics: Applied probability and statistics. John Wiley & Sons, 2004. ISBN 9780471654063.

Achille Messac, Glynn J Sundararaj, Ravindra V Tappeta, and John E Renaud. Ability of objective functions to generate points on nonconvex pareto frontiers. *AIAA journal*, 38(6):1084–1091, 2000.

Detlef Nauck, Frank Klawonn, and Rudolf Kruse. *Foundations of Neuro-Fuzzy Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1997. ISBN 0471971510.

OE Ozbulut, C Mir, MO Moroni, M Sarrazin, and PN Roschke. A fuzzy model of superelastic shape memory alloys for vibration control in civil engineering applications. *Smart materials and structures*, 16(3):818, 2007.

V. Pareto. *Manual of political economy*. Scholars Book Shelf, 1971. URL http://books.google.co.uk/books?id=qAC8AAAAIAAJ.

S.U. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill Education (India) Pvt Limited, 2002. ISBN 9780070486584. URL http://books.google.co.in/books?id=g6eUoWOlcQMC.

Riccardo Poli. Analysis of the publications on the applications of particle swarm optimisation. *J. Artif. Evol. App.*, 2008:3:1–3:10, January 2008. ISSN 1687-6229. doi: http://dx.doi.org/10.1155/2008/685175. URL http://dx.doi.org/10.1155/2008/685175.

Carlo R. Raquel and Prospero C. Naval. An effective use of crowding distance in multiobjective particle swarm optimization. In *In Proc. of Genetic and and Evolutionary Conference (GECCO 2005),25764, ACM*. Press, 2005.

I. Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Problemata, 15. Frommann-Holzboog, 1973. ISBN 9783772803734.

PAS Reed, MJ Starink, SR Gunn, and I Sinclair. Invited review: Adaptive numerical modelling and hybrid physically based anm approaches in materials engineering–a survey. *Materials Science and Technology*, 25(4):488–503, 2009.

J.R. Rice and D.M. Tracey. On the ductile enlargement of voids in triaxial stress fields. *Journal of the Mechanics and Physics of Solids*, 17(3):201 – 217, 1969. ISSN 0022-5096. doi: 10.1016/0022-5096(69)90033-7. URL http://www.sciencedirect.com/science/article/pii/0022509669900337.

J. Robinson and Y. Rahmat-Samii. Particle swarm optimization in electromagnetics. *Antennas and Propagation, IEEE Transactions on*, 52(2):397 – 407, feb. 2004. ISSN 0018-926X. doi: 10.1109/TAP.2004.823969.

G. Rousselier. Ductile fracture models and their potential in local approach of fracture. *Nuclear Engineering and Design*, 105(1):97 – 111, 1987. ISSN 0029-5493. doi: 10.1016/0029-5493(87)90234-2. URL http://www.sciencedirect.com/science/article/pii/0029549387902342.

D.E. Rumelhart, G.E. Hintont, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Y. Sawaragi, Hirotaka Nakayama, and Tetsuzo. Tanino. *Theory of multiobjective optimization*. Academic Press, Orlando :, 1985. ISBN 0126203709.

Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 1981. ISBN 0471099880.

H.P. Schwefel. *Numerische Optimierung Von Computer-Modellen Mittels Der Evolutionsstrategie: Mit Einer Vergleichenden Einführung in Die Hill-Climbing-und Zufallsstrategie*. Birkhäuser, 1977.

Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 69 –73, may 1998a. doi: 10.1109/ICEC.1998.699146.

Y. Shi and R.C. Eberhart. Empirical study of particle swarm optimization. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 3, pages 3 vol. (xxxvii+2348), 1999. doi: 10.1109/CEC.1999.785511.

Yuhui Shi and Russell Eberhart. Parameter selection in particle swarm optimization. In V. Porto, N. Saravanan, D. Waagen, and A. Eiben, editors, *Evolutionary Programming VII*, volume 1447 of *Lecture Notes in Computer Science*, pages 591–600. Springer Berlin / Heidelberg, 1998b. ISBN 978-3-540-64891-8. URL http://dx.doi.org/10.1007/BFb0040810. 10.1007/BFb0040810.

Stephen Frederick Smith. *A learning system based on genetic adaptive algorithms*. PhD thesis, Pittsburgh, PA, USA, 1980.

S. A. Soberanis. *3D CAFE modelling of ductile fracture in gas pipeline steel*. PhD thesis, The University of Sheffield, United Kingdom, 2007.

N. Srinivas and Kalyanmoy Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. volume 2, pages 221–248, 1994.

M. Sugeno and T. Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *Fuzzy Systems, IEEE Transactions on*, 1(1):7+, February 1993. ISSN 1063-6706.

V Tvergaard and A Needleman. Analysis of the cup-cone fracture in a round tensile bar. *Acta metallurgica*, 32(1):157–169, 1984.

Gilles Venturini. Sia: A supervised inductive algorithm with genetic search for learning attributes based concepts. In *Proceedings of the European Conference on Machine Learning*, ECML '93, pages 280–296, London, UK, UK, 1993. Springer-Verlag. ISBN 3-540-56602-3.

Hanli Wang, Sam Kwong, Yaochu Jin, Wei Wei, and K.F. Man. Multi-objective hierarchical genetic algorithm for interpretable fuzzy rule-based knowledge extraction. *Fuzzy Sets and Systems*, 149(1):149 – 186, 2005. ISSN 0165-0114. doi: 10.1016/j.fss.2004.07.013.

Li-Xin Wang. *A course in fuzzy systems and control*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997. ISBN 0-13-540882-2.

Y.Y Yang, D.A. Linkens, M. Mahfouf, and A.J. Rose. Grain growth modelling for continuous reheating process - a neural network-based approach. *ISIJ Int (Iron Steel Inst Jpn)*, 43(7):1040–1049, 2003. ISSN 0915-1559.

Lofti A. Zadeh. Fuzzy sets. *Information and Control*, 8:338–353, 1965.

Qian Zhang and M. Mahfouf. A new structure for particle swarm optimization (npso) applicable to single objective and multiobjective problems. In *Intelligent Systems, 2006 3rd International IEEE Conference on*, pages 176 –181, sept. 2006. doi: 10.1109/IS.2006.348413.

Qian Zhang and Mahdi Mahfouf. Mamdani-type fuzzy modelling via hierarchical clustering and multi-objective particle swarm optimisation (fm-hcpso). *International Journal of Computational Intelligence Research*, 4 (4):314328, 2008.

Qian Zhang and Mahdi Mahfouf. A modified pso with a dynamically varying population and its application to the multi-objective optimal design of alloy steels. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on*, pages 3241–3248. IEEE, 2009.

Qian Zhang and Mahdi Mahfouf. A nature-inspired multi-objective optimisation strategy based on a new reduced space searching algorithm for the design of alloy steels. *Engineering applications of artificial intelligence*, 23(5):660–675, 2010.

Qian Zhang and Mahdi Mahfouf. A hierarchical mamdani-type fuzzy modelling approach with new training data selection and multi-objective optimisation mechanisms: A special application for the prediction of mechanical properties of alloy steels. *Applied Soft Computing*, 11(2):2419 – 2443, 2011. ISSN 1568-4946. doi: 10.1016/j.asoc.2010.09.004.

E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. In K.C. Giannakoglou et al., editors, *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, pages 95–100. International Center for Numerical Methods in Engineering (CIMNE), 2002.

Eckart Zitzler and Lothar Thiele. An evolutionary algorithm for multiobjective optimization: The strength pareto approach, 1998.

Appendix A

The Fortran code of the user-subroutine used in Chapter 5:

```
      SUBROUTINE VUMAT(
C Read only -
     1 nblock, ndir, nshr, nstatev, nfieldv, nprops, lanneal,
     2 stepTime, totalTime, dt, cmname, coordMp, charLength,
     3 props, density, strainInc, relSpinInc,
     4 tempOld, stretchOld, defgradOld, fieldOld,
     3 stressOld, stateOld, enerInternOld, enerInelasOld,
     6 tempNew, stretchNew, defgradNew, fieldNew,
C Write only -
     5 stressNew, stateNew, enerInternNew, enerInelasNew )
C
      include 'vaba_param.inc'
C
C All arrays dimensioned by (*) are not used in this algorithm
      dimension props(nprops), density(nblock),
     1 coordMp(nblock,*),
     2 charLength(*), strainInc(nblock,ndir+nshr),
     3 relSpinInc(*), tempOld(*),
     4 stretchOld(*), defgradOld(*),
     5 fieldOld(*), stressOld(nblock,ndir+nshr),
     6 stateOld(nblock,nstatev), enerInternOld(*),
     7 enerInelasOld(*), tempNew(*),
     8 stretchNew(*), defgradNew(*), fieldNew(*),
     9 stressNew(nblock,ndir+nshr), stateNew(nblock,nstatev),
     1 enerInternNew(*), enerInelasNew(*)
C
      character*80 cmname
```

```
C
      PARAMETER (M=3,N=3,ID=3,ZERO=0.D0,ONE=1.D0,TWO=2.D0,THREE
         =3.D0,
     +           SIX=6.D0, NINE=9.D0, TOLER=0.D-6,IN=9,IO=4,IHD=8)
C
      DIMENSION XIDEN(M,N),XNV(4),DPSTRAN(4), XNVE(4),
     +           DESTRAN(4), DSTRESS(4), DSTRAN(4),
     +           STR(M,N),DSTR(M,N), XNDIR(M,N), DYPROD(4,4),
     +           DV(4), DDS(4,4), DPROD(4), DLAM(4)
C
      DIMENSION dinput(IN),output(IO),
     +           dinputweight(IHD,IN),dinputbias(IHD),dinputxmax(
        IN),
     +           dinputxmin(IN),dinputxrange(IN),dinputgain(IHD),
     +           outputweight(IO,IHD),outputbias(IO),outputxmax(IO
        ),
     +           outputxmin(IO),outputxrange(IO),outputgain(IO),
     +           dinputgen(IN),hidden(IHD),outputgen(IO)
C
      E = props(1)
      XNUE = props(2)
      SIGY0 = props(3)
      h = props(4)
      NTENS = ndir+nshr
C
      EBULK3 = E/(ONE-TWO*XNUE)
      EG2 = E/(ONE+XNUE)
      EG = EG2/TWO
      ELAM = (EBULK3-EG2)/THREE
C
      OPEN(122,
     1 FILE='C:\Users\uos\SkyDrive\PhD\MATLAB\DMFEM\NNPara.txt')
      READ(122,*) ((dinputweight(i,j), j = 1,IN), i=1,IHD)
      READ(122,*) (dinputbias(i), i=1,IHD)
      READ(122,*) (dinputxmax(i), i=1,IN)
      READ(122,*) (dinputxmin(i), i=1,IN)
      READ(122,*) (dinputxrange(i), i=1,IN)
      READ(122,*) dinputymax
      READ(122,*) dinputymin
      READ(122,*) dinputyrange
      READ(122,*) ((outputweight(i,j), j=1,IHD), i=1,IO)
      READ(122,*) (outputbias(i), i=1,IO)
      READ(122,*) (outputxmax(i), i=1,IO)
      READ(122,*) (outputxmin(i), i=1,IO)
      READ(122,*) (outputxrange(i), i=1,IO)
      READ(122,*) outputymax
      READ(122,*) outputymin
      READ(122,*) outputyrange
      close(122)
C
C the base of the natural logarithm
      ee=2.71828182845904523536
C
      OPEN(121,
     1 FILE='C:\Users\uos\SkyDrive\PhD\ABAQUS\DDSDDE.txt')
C
      do 100 ei = 1,nblock
C
           DO K=1,4
             DSTRAN(K) = strainInc(ei,K)
           END DO
C
           DO I=1,M
                 DO J=1,N
               STR(I,J)=0.0
             END DO
           END DO
C
           DO 50 I=1,M
             DO 50 J=1,N
               IF(I .EQ. J) THEN
                 XIDEN(I,J)=1.0D0
               ELSE
                 XIDEN(I,J)=0.0D0
               END IF
  50       CONTINUE
C
           DO K = 1,3
             STR(K,K) = stressOld(ei,K)
           END DO
           STR(1,2) = stressOld(ei,4)
           STR(2,1) = stressOld(ei,4)
C
           CALL KDEVIA(STR,XIDEN,DSTR)
C
           CALL KEFFP(DSTR,PJ)
C
         qq=PJ-SIGY0
         IF (qq.LE.0.) THEN
           vnl = 0.
         ELSE
           vnl = 1.
         END IF
C
C
           DO K = 1,4
             dinput(K) = stressOld(ei,K)
           END DO
           DO K = 1,4
             dinput(K+4) = strainInc(ei,K)
           END DO
             dinput(IN) = vnl
C
           DO K = 1,4
             WRITE(121,*) stressOld(ei,K)
           END DO
           DO K = 1,4
             WRITE(121,*) strainInc(ei,K)
           END DO
           WRITE(121,*) vnl
C
         DO K = 1,IN
C           dinputgen(K) = (dinput(K)-dinputxmin(K))
C     1     /(dinputxmax(K)-dinputxmin(K))*dinputyrange+
   dinputymin
            dinputgain(K)= dinputyrange/dinputxrange(K)
            IF (dinputxrange(K).EQ.0.) THEN
              dinputgain(K)=1.
              dinputxmin(K)=dinputymin
            END IF
            dinputgen(K)=(dinput(K)-dinputxmin(K))*dinputgain(K)
     1                   +dinputymin
         END DO
C
         DO i = 1,IHD
           dntemp = 0
           DO j = 1,IN
             dntemp = dntemp+ dinputweight(i,j)*dinputgen(j)
           END DO
           dntemp = dntemp + dinputbias(i)
           hidden(i) = 1/(1+ee**(-dntemp))
         END DO
C
         DO i = 1,IO
           dntemp = 0
           DO j = 1,IHD
             dntemp = dntemp + outputweight(i,j)*hidden(j)
           END DO
           dntemp = dntemp + outputbias(i)
           outputgen(i)=dntemp
         END DO
C
         DO K = 1,IO
C           output(K) = (outputgen(K)-outputymin)
C     1     /(outputymax-outputymin)*outputxrange(K)+outputxmin(K
   )
            outputgain(K)=outputxrange(K)/outputyrange
            IF (outputxrange(K).EQ.0.) THEN
               outputgain(K) = 1
               outputxmin(K) = outputymin
            END IF
            output(K)=((outputgen(K)-outputymin)
     1      *outputgain(K)+outputxmin(K))
         END DO
C
           DO K = 1,4
             stressNew(ei,K) = stressOld(ei,K)+(output(K))
           END DO
C
         DO K = 1,4
```

```
           WRITE(121,*) stressNew(ei,K)
         END DO
C
C
  100 continue
C
      return
      end

**
************************************************
**           UTILITY    SUBROUTINES            *
************************************************
**
**
****************************************
**          EFFECTIVE STRESS           *
**   (CONTRACTED MATRIX CALCULATION)  *
****************************************
*USER SUBROUTINE
      SUBROUTINE KEFFP(EFF1,VAL1)
C
      INCLUDE 'VABA_PARAM.INC'
C
      PARAMETER (M=3,N=3)
      DIMENSION EFF1(M,N)
C
      X=0.0
      DO 10 I=1,M
      DO 10 J=1,N
       X=X+EFF1(I,J)*EFF1(I,J)
10    CONTINUE
      IF(X .LE. 0.0) GO TO 20
      VAL1=SQRT((3.0/2.0)*X)
20    RETURN
      END
**
**
******************************************************
**   DEVIATORIC STRESS CALCULATION    *
```

```
*****************************************************
*USER SUBROUTINE
      SUBROUTINE KDEVIA(STRSS,XIDENTY,DEVITO)
C
      INCLUDE 'VABA_PARAM.INC'
C
      PARAMETER (M=3,N=3)
      DIMENSION STRSS(M,N),XIDENTY(M,N),DEVITO(M,N)
C
      X=0.0
      DO 10 I=1,M
      DO 10 J=1,N
      IF(I .EQ. J) THEN
      X=X+STRSS(I,J)
      ELSE
      END IF
10    CONTINUE
C
      DO 20 I=1,M
      DO 20 J=1,N
      IF(I .EQ. J) THEN
        DEVITO(I,J)=STRSS(I,J)-((1./3.)*X*XIDENTY(I,J))
      ELSE
        DEVITO(I,J)=STRSS(I,J)
      END IF
20    CONTINUE
      RETURN
      END
**
**
```

Appendix B

The input file of small punch test simulation for ABAQUS used in Chapter 6:

```
*Heading
SmallPunchTest
*Preprint, echo=NO, model=NO, history=NO, contact=NO
*INCLUDE, INPUT=sptmesh.inp
*Amplitude, name=Amp-1, time=TOTAL TIME
 0., 0., 1., 1.
**
** MATERIALS
**
*Material, name=Material-1
*Elastic
*INCLUDE, INPUT=spt_ela_set.inp
*Plastic
*INCLUDE, INPUT=spt_pla_set.inp
*Porous Metal Plasticity
 1.5, 1.05, 2.25
*Porous Failure Criteria
 0.005, 0.0017
*Density
 7.87E-09
**
** INTERACTION PROPERTIES
**
*Surface Interaction, name=IntProp-1
*Friction
 0.25
*Surface Behavior, no separation, pressure-overclosure=HARD
```

```
**
** BOUNDARY CONDITIONS
**
** Name: BC-1 Type: Displacement/Rotation
*Boundary
 HolderN, 1, 1
 HolderN, 2, 2
 HolderN, 6, 6
 DieN, 1, 1
 DieN, 2, 2
 DieN, 6, 6
** ----------------------------------------------------------------
**
** STEP: Step-1
**
*Step, name=Step-1
*Dynamic, Explicit
, 1.
*Bulk Viscosity
0.06, 1.2
** Mass Scaling: Semi-Automatic
**               Whole Model
*Variable Mass Scaling, dt=1e-06, type=below min, frequency=10
**
** BOUNDARY CONDITIONS
**
** Name: BC-2 Type: Displacement/Rotation
*Boundary, amplitude=Amp-1
 PunchN, 1, 1
 PunchN, 2, 2, -2.5
 PunchN, 6, 6
**
** INTERACTIONS
**
** Interaction: Int-1
*Contact Pair, interaction=IntProp-1, mechanical constraint=
   KINEMATIC, cpset=Int-1
HolderI.HolderSurf, Top
** Interaction: Int-2
*Contact Pair, interaction=IntProp-1, mechanical constraint=
   KINEMATIC, cpset=Int-2
DieI.DieSurf, Bottom
** Interaction: Int-3
*Contact Pair, interaction=IntProp-1, mechanical constraint=
   KINEMATIC, cpset=Int-3
PunchI.PunchSurf, Top
**
** OUTPUT REQUESTS
**
*Restart, write, number interval=1, time marks=NO
**
** FIELD OUTPUT: F-Output-1
**
*Output, field, variable=PRESELECT
**
** HISTORY OUTPUT: H-Output-1
**
*Output, history, frequency=100
*NODE OUTPU, NSET=PunchI.MP
 U2, RF2
*End Step
```