CRANFIELD UNIVERSITY

SCHOOL OF APPLIED SCIENCES

MSC BY RESEARCH THESIS

Academic Year 2009—2010

ILJA ORLOVS

Micro–Electro–Mechanical System design synthesis and optimisation by the means of Genetic Algorithms

Supervisors: Dr Ashutosh Tiwari and Dr Elhadj Benkhelifa

October 2010

This thesis is submitted in partial fulfilment of the requirements
for the degree of Master of Science by Research.

Abstract

Micro–Electro–Mechanical Systems (MEMSs) are microscopic (1 to 100 μm in size) electro–mechanical systems that are widely used in a variety of commercial applications owing to their small size, energy consumption and, in some cases, high precision. The conventional approaches to design optimisation of MEMS is complex, time consuming and requires a multidisciplinary team with considerable expertise. Hence, exploring the utility of other unconventional paradigms becomes desirable. This thesis focuses on the MEMS design synthesis and optimisation by means of evolutionary algorithms.

As the creation of a MEMS design by conventional means is a complex process that takes substantial amount of time and requires attention of multitude of experts, design optimisation of MEMS through multiobjective Evolutionary approaches has the potential to shorten the design time by providing the designer with sets of automatically generated alternative designs to select from. This approach can handle ill–defined problems without a priori knowledge which minimises the need for expert engineers. It has potential to uncover radically new designs that are beyond the human capabilities. Research in this area is still in its infancy and limited only to a handful of researchers around the globe. Although the relevant literature has already reported results which demonstrate the potentials of the approach, there is still a significant amount of research to be done for this field to reach maturity. One thing that will always remain questionable in such applications is 'how to increase the efficiency and quality of MEMS designs?'. This thesis proposes a number of enhancements towards faster and human competitive MEMS designs.

Proposed enhancements introduce several enhancements to the process of automated MEMS design synthesis. First enhancement increases efficiency of Genetic Algorithm (GA) by lowering number of required evaluations while allowing for an algorithm to collect information on dependability of various

MEMS elements by the means of linkage learning. Second enhancement introduces automated collection and re–usage of MEMS components, a process that opens way for fully automatic MEMS design synthesis, valuable layout feature discovery and re–usage of discovered features in successive syntheses of designs. And the last third enhancement introduces tolerance towards errors of production sequence for arbitrary MEMS designs.

1. A circular chromosome, a novel data structure is used in the GA. Given data structure possesses the linkage learning capabilities, incorporates redundancy and shows increased performance of the GA when compared with previously published state–of–the–art results.

2. Based on the linkage–learning capabilities of circular chromosome, a component–based GA implementation is presented and explored, showing noticeable increase in the GA performance when compared with results of traditional MEMS design synthesis by the means of GA.

3. A process–induced error tolerant MEMS layout synthesis is proposed. Because currently deployed micromachining processes have low relative precision, it is desirable to introduce automated synthesis method that is able to evolve MEMS designs with less sensitivity towards production errors. The approach for evolutionary synthesis of such MEMS designs while exclusively using numerical simulation is proposed and is displayed to yield error–tolerant MEMS designs when compared to the results of classic non–tolerant optimisation.

Contents

List of Figures

List of Acronyms

AI	Artificial Intelligence
AMS	Analog & Mixed–Signal
BB	Building Block
BEM	Boundary–Element Methods
CAD	Computer–Aided Design
CBR	Case–Based Reasoning
DB	Database
DNA	Deoxyribonucleic Acid
DOE	Design of Experiment
EA	Evolutionary Algorithm
EC	Evolutionary Computation
EHD	Energy Harvesting Devices
EO	Evolutionary Optimisation
EPE–2	Extended Probabilistic Expression 2
FEA	Finite Element Analysis
FEM	Finite Element Method

FE Finite Element

GA Genetic Algorithm

GPS Global Positioning System

HDL Hardware Description Language

IC Integrated Circuit

IHC Interactive Hybrid Computation

IT Information Technology

LLGA Linkage learning Genetic Algorithm

MEMS Micro–Electro–Mechanical System

MOGA Multi–objective genetic algorithm

NSGA–II Non–dominated Sorting Genetic Algorithm II

PDE Partial Differential Equation

PE Probabilistic Expression

R/W Read/Write

R&D Research and Development

SA Simulated Annealing

SOC System–On–a–Chip

UML Unified Modelling Language

Chapter 1

Introduction

Micro–Electro–Mechanical System (MEMS) is a microscopic mechanical system that often has an embedded electrical circuit and is designed for a specific function. MEMS tend to incorporate elements of various physical domains – electronics, fluidics, biological et al. In the light of multi–domain nature of MEMS it is not uncommon to see active interaction and experience exchange between various disciplines of Science on given field. While being active field of research for the last few decades, MEMS design and production has grown to multi–billion industry with its production being actively deployed in the long list of industries and wide array of day–to–day items(Liu 2006).

As for now MEMS production process is based on fabrication processes originally developed for semiconductor–based Integrated Circuit (IC) industry with attempts of expanding IC production technology with MEMS–specific functionality are constantly being made.

When 'MEMS' term was created, it was considered that MEMS machines incorporate only mechanical and electrical elements as is reflected in its name, but in present time they commonly include magnetic, optical, chemical, thermal, and biological elements as well(Bao & Wang 1996). As MEMS devices are intrinsically smaller than their macroscopic analogues, they tend to have smaller mass and faster response time. While exploiting material properties found on microscopic level can lead to noticeable increase in precision and sensitivity of a MEMS device comparing to its macroscopic analogues.

It is widely accepted point of view that micro engineering holds a promise of enabling major advances in almost all aspects of automotive, transportation, medicine, biomedicine, pharmaceutics, health, telecommunication, energy, Information Technology (IT), security, informatics and environmental monitoring – just to mention few. In (Gilleo 2005), author asserts that "MEMS is the ultimate enabling technology for the integration of almost any physical, chemical and biological phenomena that include motion, light, sound, chemistry, biochemistry, radio waves and computation, but all on a single chip". Hence, as range of applications of given technology grows, the importance of research and development of given branch of technology is increasing.

Yet another fields actively deploying and greatly benefiting of MEMS is biomedicine and medicine. With MEMS deployed in these field being named as BioMEMS(Grayson, Shawgo, Johnson, Flynn, Li, CIMA & Langer 2005). While much work is underway, advances in BioMEMS are remarkable, given a short time. BioMEMS can be categorised into two types of devices – sensing and action devices. This BioMEMS classification correlates with general MEMS classification that commonly distinguishes two major classes of MEMS – sensors and actuators and one additional actively researched class with no noticeable commercial applications yet – an energy generator, namely an Energy Harvesting Devices (EHD) technology.

As BioMEMS, sensor MEMS can be used to measure pressure (e.g. blood or spinal), temperature, glucose, force (e.g. tissue tension or muscular) and electrical impulse (e.g. nerve, brain or heart).

It is possible to deploy MEMS sensors for detection of various chemical substances(Bedair & Fedder 2005), for instance, for oxygen detection or to develop sensors for monitoring purposes, for example gas flow monitoring.

As for the actuation class, BioMEMS applications include creation of a micro–fluidic pump for drug delivery and DNA separators used for DNA analysis. Another medicine field that is actively deploy array of MEMS devices is surgery(Rebello 2005). Amongst non–biological applications of MEMS here are actuators that are used in bomb triggers, as one example, microswitches and micro–resonators.

As MEMS field is actively expanding and is deployed in number of industries, a number of computer–aided tools for MEMS design exists to help designers to cope with increasing demand for new MEMS designs. As for now, the major focus of given computer–aided tools is analysis of design requirements and specifications at various levels of granularity of model. Although highly advanced and useful in the process of designing and evaluation of novel devices due to advanced visualisation and analysis techniques, its capabilities of automated design optimisation are currently quite rudimental.

As for now, the great share of optimisation is done manually by a slow process of model design, model evaluation and number of successive design tweaks and re–evaluation loops that often incorporate a number of software tools.

The number and complexity of tools used in MEMS design process is dictated by the fact that MEMS is an interdisciplinary field that combines elements of electronics, mechanical and electrical engineering, chemistry, optics, and fluid mechanics. The multi–domain nature of MEMS is another reason MEMS technology enables a vast variety of applications.

As the MEMS industry is still of a relatively young age and as it involves large amount of disciplines of Science, there is no integral 'Science of MEMS' yet developed as well as the there is still no 'Engineering of MEMS design' with error–free approach for creation of MEMS designs(Clark, Bindel, Kao, Zhu, Kuo, Zhou, Nie, Demmel, Bai, Govindjee et al. 2002).

Establishing set of methodologies for the design of a MEMS is one of big objectives of current MEMS research. Development of whole loop of MEMS design beginning with specification of the desired function and resulting with a design optimised for fabrication would greatly benefit industry.(Silva, Giasolli, Cunningham & DeRoo 2002) As current design process flow device layout synthesis and all other elements of MEMS device development purely on MEMS experience of experts and their prior experience in MEMS design and knowledge of similar devices is commonly perceived to have room for improvement. In general, the process of MEMS design is complex and challenging task as the structure of MEMS commonly is coupled, heterogeneous and intrinsically three–dimensional.

The common MEMS design workflow deployed in the industry is based on trial–and–error approach. With given approach requiring several iterations and number of prototypes built and tested to destruction before design requirements are matched. It is believed by experts that this methodology has room for improvement as it requires noticeable financial investments for the production of prototypes as well as it accounts of certain time share of products' life span(Fan, Wang, Achiche, Goodman & Rosenberg 2008).

Although precision of computer–aided simulation has increased by a great degree during last years, the process is still limited. For example, currently existing tools face noticeable difficulties emulating multi–domain external environment.

Common optimisation methods (such as Newton's method and gradient descent) are suitable for problems of low dimensionality, but have their performance limited in case of high–dimensional problems and as in the field of MEMS design most problems require high–dimensional optimisation process to cope with growing number of design specifications and variables.

Yet another downside of using traditional methods is that commonly used methods like Newton's method or gradient descent are meant for finding local optima by 'rolling down the hill' from the starting point given to optimisation method. As traditional methods are deterministic in their nature, they yield deterministic results that are dependant on the choice of start point. This means that such a method would be unable to explore completely new MEMS designs by remaining restricted to the local solution space. This downside makes traditional methods unsuitable for the aim of the MEMS design synthesis.

Although MEMS optimisation process is currently dominated by traditional methods, an alternative route exists, looking to overcome some of the difficulties faced by normal optimisation methods.

The alternative approaches are usually based on stochastic algorithms belonging to the Evolutionary Optimisation (EO) that is subdomain of Evolutionary Algorithm (EA). All EO methods are stochastic algorithms based on the principles of Darwinian evolution, where potential solutions undergo evolution process inspired by Nature: selection, reproduction and mutation.

One of the most popular EO algorithm classes used for MEMS design synthesis and optimisation are GA, that have been shown to perform well on a variety of the device optimisation applications including MEMS design optimisation.

GA is class of stochastic algorithms inspired by natural evolution used for black box optimisation. Just as natural evolution, GA uses mutation, crossover, selection and inheritance to find optimal solution set. With solutions closer to optimum being considered to be of higher fitness and are given higher chances of transmitting their genes to successive generations.

Due to multi–domain nature and complexity of MEMS, the performance of GAs is still limited.

The designs of MEMS devices usually have complex topology, and the GAs are commonly developed using more simple theoretical problems, thus their potential performance remains partially unexplored because developed sub–elements of given algorithms and data structures were built and evaluated on the much simpler kind of problems with predictable and controlled set of constraints.

1.1 Aim and objectives

During the literature review a variety of problems of MEMS automated design synthesis by the means of GA process were highlighted:

1. large number of objective function evaluations
2. high computational complexity of each function evaluation
3. inefficient MEMS structure representation in GA
4. GA inability to reuse past MEMS designs and learn from past experience
5. inability of GA to automatically learn and preserve good MEMS design elements
6. lack of classification of MEMS devices and their design elements

7. problem of synthesis of MEMS designs that are tolerant towards production errors

8. relatively low precision of response prediction for current MEMS software evaluation packages

9. low precision of MEMS production

It would be impossible to address all of problems listed above, so a subset of problems was chosen for current research.

It was decided to address further aims:

1. large number of objective function evaluations

2. inefficient MEMS structure representation in GA

3. GA inability to re–use past MEMS designs and learn from past experience

4. inability of GA to automatically learn and preserve good MEMS design elements

5. automated synthesis of production error tolerant MEMS designs by the means of GA

The aim of this research is to develop new Genetic Algorithm based approaches for the optimisation of MEMS designs. To achieve this aim, a number of research objectives were identified:

1. To propose circular chromosome with linkage learning capabilities to address the issues associated with large number of objective function evaluations and inefficient MEMS structure representation in GA.

2. To develop automatic Building Block (BB) detection and re-use approach to address the inability of GA in automatically re–using past MEMS designs and learning and preserving good MEMS design elements.

3. To propose error–tolerant MEMS layout synthesis to enable automated synthesis of production error tolerant MEMS designs by the means of GA.

1.2 Methodology

For current research, the inductive approach is assumed to be the most suitable, as MEMS is a relatively recently–established field of research, which restricts amount of MEMS–specific theoretical knowledge available on system design.

As the inductive approach dictates, the first thing done is an accumulation of observations made by fellow researchers and available in the literature. Some general patterns are spotted for which a hypothesis is formed and tested for feasibility and the theoretical basis for acquired results is drawn.

To address large number of objective function evaluations and inefficient MEMS structure representation in GA, the structure of circular chromosome was proposed amongst with adaptation of novel mutation operator that was originally developed and used for other types of optimisation problems. Resulting algorithm was used to performs number of experiments and the results of given experiments were compared with results of previous state–of–the–art research.

To address inefficient MEMS structure representation in GA, GA inability to re–use past MEMS designs and learn from past experience and inability of GA to automatically learn and preserve good MEMS design elements, the approach for automatic Building Block (BB) detection and re–usage based on linkage learning capabilities of circular chromosome was proposed. As given approach was implemented, the several modifications of general approach were tested and compared against results of common GA evolution.

To address automated synthesis of production error tolerant MEMS designs by the means of GA, the literature on nature and statistical properties of MEMS production errors was reviewed, proposing approach for simulation of given production errors in the virtual environment and the adaptation of previously described MEMS robust layout synthesis to the domain of numerical

simulation was done. As the first attempt of adaptation displayed inability to evolve MEMS layouts with tolerance in case of multiple response evolution (as opposed to single response case previously addressed in literature), a number of algorithm modifications were tested addressing found inconstancies. Results acquired from various algorithms were compared against each other and basis for observed performance of each modification was drawn.

1.3 Thesis structure

The thesis consists of an introduction chapter, followed by the literature review chapter describing the global state of MEMS research and development direction with critical discussion.

The latter chapters contain research done – the initial observation, hypothesis made, experiments conducted and the theoretical basis given for obtained results with critical discussion of viability of previously given hypothesis in the light of obtained results.

In the last chapter, conclusions are aggregated and summarised, research limitations are exposed and direction for future research is drawn.

1.4 Contributions

Attempting to help for GA to address the complexity of the MEMS design, a new chromosome representation was proposed, named 'circular chromosome'. The main advantage of circular chromosome in comparison to the commonly used plain chromosome is that it introduces linkage learning and redundancy(Chi & Lin 2008, Harik 1997) to the process of MEMS design optimisation.

The redundancy is expected to allow for GA to 'remember' parts of good designs(Levenick 1991) even if they currently aren't very beneficial. The memorisation would be expected to happen when mutation in GA causes for the sequence of the genes in chromosome to become disabled, unexpressed in the actual device design. And the process of re-introduction of such disabled

genes would require for GA to trigger an opposite mutation – a mutation that would enable sequence of disabled genes to be expressed to the MEMS device design(Wineberg & Oppacher 1996).

The circular chromosome was compared with results with the state–of–the–art GA proposed by a variety of researches and it was shown that circular chromosome–enabled GA is able to find better solutions with fewer function evaluations (section 3.5).

Next problem addressed in this thesis in chapter 4 is inability of GA to re–use experience from previous runs, leaving it to perform 'cold start' by generating a number of random MEMS designs each time optimisation process is initiated. This might be undesirable, as it is a computationally demanding task to generate enough random combinations of genes to accumulate needed number of feasible MEMS designs to emerge. It is also unknown what the distribution of solutions in objective space will be.

To address this problem, the component–based GA was introduced. The given component incorporated noticeable number of MEMS components in the database and with ability to re–use the given components in the GA evolution.

To increase efficiency of MEMS layout synthesis, GA decision space needs to be reduced. In case of component–based MEMS layout synthesis this is done by dividing set of variables describing each GA component to a two subsets – the first set where GA is allowed to change values and second set that contains bound variables, forbidden to be changed by GA in the process of optimisation.

The various combinations of the variables for the component–based GA were tested and it was demonstrated that some of combinations of variables result in superior solutions when comparing with the reference state–of–the–are GA.

The component–based MEMS design optimisation has demonstrated its vitality and ability to generate better Pareto front than other state–of–the–art GA.

Third problem researched in current thesis is automatic generation of MEMS designs having tolerance towards geometrical errors introduced by

production process.

To address this problem, a robust optimisation–inspired approach was proposed and undertaken. A number of enhancements of initial approach were proposed and tested. The ability of GA to evolve error–tolerant designs of MEMS devices of arbitrary topology exclusively by the means of numerical simulation was demonstrated. Please refer to chapter 5 for detailed research undertaken.

Chapter 2

A Review of Literature: MEMS design optimisation

2.1 Introduction

Field of MEMS is an multi–disciplinary field incorporating elements of various areas of science and engineering. Being actively researched for the last few decades, this field greatly enhanced and became much more mature throughout these years. The MEMS has successively emerged into commercial technology with total market volume measured in billions and a wide array of commercial applications. The potential of MEMS was known long before the advance in science and technology actually enabled it – as an example, Richard Feynman's 1959 lecture "There's Plenty of Room at the Bottom"(Feynman 1999) has anticipated possibility of creation of a micro–scale machines.

As one particular technology that enabled creation of MEMS one can name construction of transistor in 1947 at Bell Labs(Bardeen & Brattain 1948). This moment can be viewed as the start of the long race for miniaturisation we're observing nowadays, as it has initiated a fast paced IC industry. The IC technology enabled a variety of methods that were later adapted for production of MEMS. Continuous attempts of development of MEMS–specific production precesses that would meet all needs of given industry are

attempted.

According to (Hsu 2001), for device to be considered a MEMS device, it must incorporate two types of systems – Mechanical and Electrical (Figure 2.1). This claim was illustrated by author using the example of vehicle airbag, as in the case of accident, the airbag detects acceleration using embedded micro–sensor and inflates.

The principal structure of micro–system can be seen in Figure 2.1.

However contrary to its original meaning the term MEMS is now includes variety of other types of elements – magnetic, optical, chemical, thermal and biological. Antennas can be included to MEMS as well. In this context, the integrated circuits are the components encoding the behaviour of the MEMS.

MEMS can be designed to integrate some or even all of micro structures required for any particular task or function to acquire ability to register and process information acquired from physical stimuli of any domain and act according to designed program – output an electrical or digital signal, emit optical signal or perform any other possible action in compliance with their design and available functionality (Lyshevski 2002).

It is possible to deploy multiple combinations of the components from various domains – electrical, mechanical as sensors for a variety of environmental parameters – velocity, pressure, temperature, effectively making MEMS of an unprecedented devices for compound systems such as the micro–heat–engines, robots, System–On–a–Chip (SOC) and heat pumps (Gad-el-Hak 2006).

2.1.1 Current and future applications of MEMS

According to market review published by Nexus(Tischulena 2004) and judging by the fact that volume of research done in field of MEMS is steadily increasing (according to (Google Inc. 2010), the amount of publications on the MEMS field during period of 1990—1995 years and period of 2005—2010 are 17400 and 18200 respectively), the micro–systems industry is currently booming and has potential to revolutionise our current lifestyles as much as the computer has.

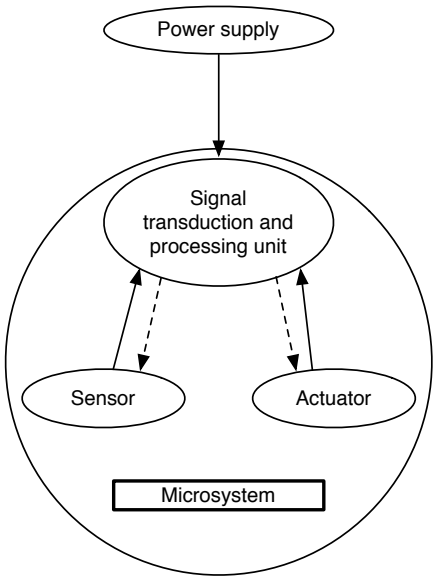


Figure 2.1: Components of a MEMS (Hsu 2001)

However, the aim of this section is not to discuss process of advancement of MEMS design or any particular state–of–the art design, but rather to display the desirability of Research and Development (R&D) of MEMS and the research that one might expect to happen in a way that is similar to one that computer industry has undergone – the majority of research and advancement in technology being made not because of governmental needs but because of commercial investments to the R&D. This point of view is supported by the Nexus's third technology market analysis report (Tischulena 2004) as it promises for MEMS market to double from $12 billion in 2004 to $25 billion in 2009.

According to data published by Nexus(Tischulena 2004), the IT sector is responsible for the biggest share of used MEMS and such state is expected to remain unchanged in the near future even as the MEMS market is constantly expanding and shares of other industries are starting to increase. The IT industry consumes the majority of MEMS technology as a variety of IT–related machinery incorporates MEMS elements, for example inkjet printer heads and Read/Write (R/W) heads in most of tape drives.

As for the consumer market, the variety of peripherals, digital cameras,

personal navigation Global Positioning System (GPS)–enabled devices all usually incorporate elements produced by the MEMS technology.

MEMS products have also fount variety of applications in medicine and biomedicine fields[1] (Gilleo 2005, Maluf, Gee, Petersen & Kovacs 1995). While much issues remain to be solved, advancement done by BioMEMS is remarkable, especially taking into account the relatively small age of given technology.

2.1.2 Designing a MEMS device

As the number and sophistication of MEMS designs increases with time, there is an increasing interest in standardisation of MEMS design practices.

Although nominal strategy and general outline for MEMS device design process is invented, no standard methodologies for achieving any particular desired design specifications are developed(Antonsson, Cavallaro, Mahadevan, Maher & Maseeh 1997). A design flow, that is similar with the design flow defined for devices of electronics domain, commonly followed during MEMS design process was identified and developed by Fedder (Fedder, Iyer & Mukherjee 1997).

As one might expect, the MEMS design process starts on the level of highest abstraction, where overall system specification (or in other words functional requirements) is defined. The second step, according to that design flow, is creation of a high–level description of the future MEMS (or 'system description', as it is called in the related literature). As defined by Fedder in (Vudathu & Laur 2007) in the MEMS design flow, during MEMS design process designs of two abstract representation of MEMS are manipulated: micro–mechanical and physical device layout. And as the final stage in the MEMS design process, the 'fabrication and testing' phase is defined. Usually the micro–mechanical components of the system are fabricated using micro–machining computable processes (the most widespread process is etching – a process of removal of desired elements of silicon wafer leaving structure of desired design) that forms MEMS devices, but the electronic components in

[1]This class of applications is known as BioMEMS

MEMS are usually fabricated by same process that is used for production of process of IC. The MEMS design flow similar to one that was reported by Fedder, was also defined by Antonsson in (Antonsson et al. 1997).

2.2 Conventional optimisation of MEMS design

For any particular separate MEMS device (not system of MEMS devices), the computer aided modelling process can be represented as the three–stage process: creation of the computer model of the device, simulation of computer model of a device to obtain estimate of its physical behaviour and properties and the actual analysis of simulation.

In (Senturia 2001), Senturia proposed a hierarchy that is comparable with previously discussed MEMS design flow according to which the MEMS design synthesis consists of four design levels:

1. 'System' level – level describing integration of the device or devices with electronic components used for Input/Output of data

2. 'Device' level – level where a function model of device is defined

3. 'Physical' level – level where a physical model of device is created, for example by using techniques of Finite Element (FE) modelling

4. 'Process' level – level where device fabrication defined e.g. by defining process information and mask layout

The Levels are listed in order of reduction of order of abstraction (with 'System' level being most abstract and the 'Process' level being least abstract).

The conventional MEMS optimisation is usually manual trial–and–error process that is ether performed using 'build and break' process (process of building a MEMS prototype and testing it to destruction) or using Computer–Aided Design (CAD) tools for building the model of MEMS and running simulation upon it to gain estimates of the devices' properties.

The precision of CAD tools have increased by a great degree during the last few years as did the amount of available computational powers. But it is still largely considered impossible to produce a working MEMS without building at least few prototypes(Markus 2002, Schr\öpfer, King, Kennedy & McNie 2005).

Conventional optimisation workflow is heavily dependant on tools available. Hence, deployment of CAD tools in the design process makes MEMS designers to rely on their CAD tools in their day–to–day lives, and as focus of this work does not include the processes of MEMS production, it makes sense to discuss computer–aided simulation techniques available up to current time.

Development of software that could be used for MEMS design and simulation began in the 80's. There is a wide variety of tools available and a huge amount of accumulated information. Current work covers only major and most noticeable moments in research on MEMS simulation. For the brief timeline of the development of MEMS simulation software please refer to Table 2.1.

Several approaches for simulating a behaviour of a MEMS device are currently known:

1. Finite Element Method (FEM)
2. Analytical modelling
3. Macro–modelling
4. Nodal modelling
5. Reduced Order Modelling

2.2.1 Computer–aided modelling

Finite Element Method is the most widespread MEMS modeling technique, it is a numerical modelling that is able to find approximate solutions with one of boundary element, finite element or finite difference method for the Partial Differential Equations (PDEs).

Date	Contributors	Methods	Description
1973	L., Nagel (Quarles, Pederson, Newton, Sangiovanni-Vincentelli & Wayne 2010)	'SPICE' Behavioural Simulator	SPICE is a integrated circuit simulation package that can be used for simulating various MEMS devices
1982	Lee, K. W., and Wise, K. D., (Lee & Wise 1982)	Simulation Analysis, Finite Difference method	Simulation tool for calculating the output response of piezoresistive or capacitive pressure sensors. Coined 'SENSIM'
1984	Verilog–AMS (*Verilog-AMS* 2010)	Hardware Description Language (HDL) Simulator	Verilog–AMS is a hardware description language tool.
1987	Bin, T. Y., and Huang, R. S.	Analytical	A Simulation tool for pressure sensor analysis and geometrical design. Coined 'CAPSS'
1989	Puers, B., et al., (Puers, Peeters & Sansen 1989)	FEA	Device level modelling and analysis package 'CAPSIM'
1989	Koppelman, G, K., (Koppelman 1989)	Process Modeller, Solid Modelling, Process Description Language	A process and fabrication simulator based on integrated circuit fabrication techniques. Also solid modelling is present. Coined 'OYSTER'
1990, 1991	Crary, S., and Zhang, Y., (Crary & Zhang 1990, Crary, Juma & Zhang 1991)	Process Modeller, FEM/ FEA	Combining process modelling, mesh creation and finite element analysis into a CAD environment coined 'CAE-MEMS'.
1992	Senturia, et al., (Senturia, Harris, Johnson, Kim, Nabors, Shulman & White 1992)	Process Modeller, FEM, BEM and hybrid FEM–BEM analysis.	A package combining process modelling and simulation along with simulation and analysis tools. Included electrostatic analysis. Coined 'MEMCAD'
1997	Vandemeer, J., et al., (Vandemeer, Kranz & Fedder 1997)	Behavioural Simulator	A behavioural simulator that combines lower element building block behaviours into higher ordered structures for a system level analysis.
1998	Zhou, et al., (Zhou, Clark & Pister 1998)	Behavioural Simulator	An approach for higher level modelling by abstracting behaviour of elements for the construction of planar devices. Coined 'SUGAR'

Table from (Benkhelifa, Farnsworth, Tiwari, Bandi & Zhu 2010, Tables 1–4)

Table 2.1: A chronological review of major cornerstones in R&D of MEMS field

The Finite Element approach in solid modelling is implemented by discretising the structure of modelled device to a set of discrete atomic elements described by finite number of formulas (such as PDE)(Gyimesi, Avdeev & Ostergaard 2004, Hung, Yang & Senturia 2002).

Estimated responses of a whole model are calculated using information obtained from simulation of set of all descrete elements that are given model consists of, taking into account each elements' properties such as stiffness, electrical conductivity, mass and others. To aid design synthesis of MEMS on a 'Physical' (level where a physical model of device is created) and 'Device' (level where a function model of device is defined) levels, a number of FE applications have been developed to incorporate physical layout creation for example, (L-Edit n.d.).

Analytical modelling is an approach to describing the behaviour of a MEMS device using set of mathematical formulas.

MEMS device model can be developed for describing the behaviour of a whole device by using differential equations. Analytical models allow for designer to get insight on device performance just by computing set of equations. Analytical models usually have few inputs – the most common is to have only device geometric parameters and material properties as inputs. As analytical models are simple and computationally effective when compared with FE modelling process, as FE modelling process can take for several days to complete. The analytical models can contribute to increase designers insight to a device and its behaviour that he or she is building(Senturia 2001).

However, construction of analytical model usually is extremely time consuming and complex task. Also, it is usually impossible to use analytical model of a simpler device that is being part of a supermodel, e.g. using analytical model of a MEMS resonator to create a multi–stage MEMS resonator that consists of multitude of resonators (Mukherjee, Fedder, Ramaswamy & White 2000).

Given limitations make the field of application of analytical models very ad–hoc and not as popular as one might want or expect them to be.

Macro–modelling is a combination of an analytical modelling and a numerical method for analysing device behaviour. Usually given models are effectively defined as low–order differential equation or in the form of circuit (Crary & Zhang 1990, White 2004).

The extraction of lumped elements by coupling of numerical simulations can help in overcoming fragility that analytical methods may have but usage of given methods can be costly.

The weak spot of macro–modelling is the process of creation of macro–models as creation of such models requires vast amounts of time investments and large number of personnel, as creation of these models require attention of MEMS experts.

Nodal modelling is an approach that is based on the idea that a system can be decomposed into a set of N–terminal devices each being represented by an ordinary differential equation. N–terminal devices can be interconnected with the coupled differential equations being solved by nodal analysis.

It is common for nodal modelling software to provide basic system building blocks, such as beams, gaps and anchors that can be assembled into a multitude of different planar electro–mechanical structures that can be compared and analysed using nodal behavioural analysis. The hierarchical structures maintain a connection between physical implementation of the device and its behaviour.

The validity of results obtained using nodal modelling is usually done by comparing against traditional finite element analysis techniques by (Clark, Zhou & Pister 1998, Zienkiewicz & Taylor 2006) and by measuring responses of actually built MEMS devices. It was demonstrated that nodal modelling approach is feasible and less computationally expensive than FEM.

Reduced order modelling is a technique of creation of MEMS models namely 'reduced order models' that require less computational power and are dependant on smaller amount of variables by using a number of approaches such as quadratic techniques (Bechtold, Rudnyi & Korvink 2003, Yang & Yu 2004, Chaturantabut & Sorensen 2010) or an Arnoldi algorithm (Bai

2002, Freund 2004) or the Karhunen-Loeve/Galerkin method.

As most reduced model have fixed device layout and face difficulties when simulating nonlinear systems that is MEMS, the application of Reduced order modelling to the domain of MEMS remains limited. Due to fixed device layout, reduced order modelling yield little insight on effect of various elements of MEMS design on its performance, but their deployment is beneficial for an iterative design synthesis approach on 'system' and 'device' levels.

2.2.2 Automated MEMS design optimisation

The multi–domain nature of MEMS makes it likely that MEMS optimisation process will require knowledge of several fields such as electrostatic, fluidic and electronic and therefore it is likely that a number of experts in a variety of fields will be required for a task of designing single MEMS device.

Because multitude of software for MEMS design and simulation lately became available and precision of simulation results started to reach a level when it could partially replace old 'build and break' process of designing MEMS, it was just a matter of time until research on the optimisation of a simulated MEMS device is undertaken.

The automated optimisation usually uses computer simulation for evaluation of MEMS performance, the real–world success of it is limited by precision of design parameter estimates provided by software and the limitations of a MEMS production process.

Conventional automated optimisation techniques are tightened to look for nearest optimum point (also referred to as 'local optima'); therefore they require an input MEMS model that needs to be optimised. This means that conventional optimisation still relies on a MEMS expert (or experts) to develop the model that needs to be optimised.

Because of that, the common aim of conventional optimisation is to optimise a design that is worth of being optimised, saving valuable time of MEMS experts without requiring them to evaluate multitude of slightly different designs.

Usually, the conventional optimisation process requires two parameters

to be configured. The first parameter is the so–called 'cost function' – an integral function used to compare one design of device to another (with design that is having lesser 'cost' to be considered superior). The second parameter is an actual model of a MEMS, that can be simulated using software of choice. After defining these two sets of parameters, the only thing left to do is an invocation of optimisation algorithm, an algorithm that will find such combination of input parameters with respect to constraints that minimise cost function.

The most common approach is making an optimisation algorithm that simply loops through all possible and allowed combinations of parameters and finds the combination of parameters that results in minimal cost function. The obvious downside of this is that such an approach requires extreme amounts of computational power, especially if different design parameters have different levels of impact to the cost function.

Yet another problem of such an approach is that the cost function might not be continuous (according to (Brenner, Lang, Li, Qiu & Slocum 2002)) and any optimisation undertaken (to make search faster by not evaluating every combination of parameters) might lead to lost solutions. To address this issue, the model reduction technique is usually applied to the MEMS design. As the creation of reduced model requires a lot of FE simulations of original MEMS design, the process of reduction itself can become troublesome.

Another well–known and widely used approach is gradient–based search or alternatively, 'gradient descent', that does not perform check of all possible parameter combinations, it rather 'slides down' the hill of cost function down to the local minima. The usage of such a method expects that the problem that is being optimised does not contain areas where cost function is undefined; so gradient–based search cannot be applied to any MEMS optimisation problem, but rather to a subset of given problems that possess required properties.

Because computational complexity of gradient–based search increases with increase in dimensionality of search space (that is, with increase in number of input parameters to find combination of the given MEMS design problem), the application of gradient–based search to the design optimisation of

complex MEMS designs becomes a burden for an optimising system. This problem was addressed by (Brenner et al. 2002, Abdalla, Reddy, Faris & G\ürdal 2005) where researchers attempted to invent a gradient search insensitive to the number of parameters.

The list of algorithms used for conventional MEMS design optimisation of MEMS is not limited to the brute force optimisation and the gradient–based search.

Amongst methods used for MEMS optimisation there is a well–known algorithm called 'Simulated Annealing'(Ongkodjojo & Tay 2002, Engesser, Franke, Maute, Meisel & Korvink 2010) and hybrid methods such as 'Simulated annealing–Gradient search' or 'Simulated annealing–Generalized Reduced Gradient (GRG)'(Parkinson, Jensen & Roach 2000, Parkinson & Wilson 1988).

2.3 Analysis of conventional approach

Although there is a remarkable progress in development of MEMS modelling and simulation tools with steady increase of precision in simulated responses, usability of MEMS designing software, range of simulated environments and domains, analysis and MEMS design visualisation techniques, the research volume on automated MEMS design remains limited. Therefore, most commercial MEMS built up to today are usually undergoing slow and high in cost hybrid CAD–assisted initial design phase and 'build and break' evaluation and re–evaluation.

Since MEMS is a relatively young branch of technology and because MEMS use multi–domain devices that incorporate elements of mechanics, electronics, optics, chemistry and other domains, there is no global approach developed for designing a MEMS device. As there is no such approach, there are no design methodologies on how to develop a MEMS that matches particular design objective, leaving the whole process to the personal experience of the experts performing the actual design of the fabricated MEMS(Achiche, Fan & Bolognini 2007, Xin, Guangyi, Liang & Guizhang 2006).

The multi–domain nature of MEMS often requires for experts to build

multiple representations modelling various parts of MEMS for various domains. That leads to wasted effort and the unnecessary burden and complexity in assembling resulting device design. Therefore, there is a need for unified cross–domain representation of MEMS designs, for MEMS experts to be able to share various parts of designs between simulation environments and simulation abstraction levels (currently, transferring a device design to a different simulation level usually requires creation of new design representing given MEMS on this level).

According to Zha and Du (Zha & Du 2003) and as confirmed by a variety of other researchers (Calis & Desmulliez 2006, Fan et al. 2008), the MEMS design process up to date is generally performed by trial–and–error, what requires multiple design iterations before the design requirements for designed system are fulfilled. This is considered an inefficient, slow and costly design methodology. Also, as MEMS production techniques have relatively low precision (when comparing with the actual size of devices produced), their actual design process ofter requires insight to the fabrication process to be used and because of limitations of a production process, the functional design of a MEMS system is often influenced by the restrictions of fabrication process. Also, according to Calis et al. (Calis & Desmulliez 2006), there is a lack of design 'weak spot'[2] detection in currently available software tools and the lack of ability for current simulation environments to predict the outcome of complex internal or external influence.

The widely used conventional optimisation techniques (such as exhaustive search[3], gradient descent or Newton's method) have proven to be applicable to the field of MEMS design optimisation but all of them have certain downsides.

The exhaustive search requires enormous computational effort, as it evaluates all possible design modifications and number of such designs grows exponentially with parameter state increase and parametrically as number of states increases, as number of design combinations is $\prod^{k} p_k$ where k is number of design variables and p_k is number of states the kth design variable

[2] detection of spots that are likely to cause problems during exploitation of actual device

[3] Also named as 'brute force search'

can be in.

The computational complexity of gradient and Newton's methods greatly increases with growth of number of design variables to be searched. Another weakness of these methods is that they find local optima. Therefore they require a good starting point (an initial MEMS design created by expert to be optimised) and that their performance is dependant on this starting point and the result is deterministic, meaning that these methods will not be able to 'climb out' of local optima and return another, better solution.

Another downside of using conventional optimisation techniques is that for design to be optimised, the designer needs to define an objective function(s) that need to be optimised, the functions to obtain response estimates from simulation environment, and mapping of given responses to the objective functions. Plus, a set of constraints usually needs to be defined for optimisation not to produce invalid or irrelevant solutions (i.e. designs that cannot be produced with selected production technique). All of this requires in–depth knowledge of simulation environment, the MEMS production process and the multitude of other ad–hoc features that cannot be easily standardised. Plus, all given things need to be re–done when the initial design to be optimised is altered; this increases work burden on MEMS designers and removes them from their primary duties – namely, designing a MEMS devices.

2.4 MEMS design synthesis by the means of EC

The Evolutionary Computation (EC) is a subfield of Artificial Intelligence field of Computer Science (for more detailed position of EC in the taxonomy of Computer Science please refer to Figure 2.2).

The EC uses darwinian principles for finding solutions – a growth or development in a population–based random guided search.
The field of EC consists of four major branches:

1. Genetic Algorithms (GA) (Holland 1992, Goldberg 1989)

Figure 2.2: Place of EC and GA in Computer Science

according to ACM taxonomy(ACM n.d.)

2. Evolutionary Strategies (ES) (Rechenberg 1994)

3. Evolutionary Programming (EP) (Fogel 1999)

4. Genetic Programming (GP) (Koza 1992)

This thesis concentrates on the usage of GA for the MEMS design synthesis as it is one of the mostly adopted on the field for the MEMS design optimisation by the means of EC. Another reason for picking the GA is that Genetic Algorithms were successfully applied to a variety of device optimisation problems(Weile, Michielssen & Goldberg 1996, Lu & Kota 2003).

The GA is search heuristic that is inspired by the process of natural evolution. This heuristic is routinely used to generate good solutions to optimisation and search problems.

The word 'synthesis' is defined as "the combination of components or elements to form a connected whole"(Oxford Dictionaries 2010). Therefore, the term 'design synthesis' would be a process of finding such combination of design elements that would meet all design requirements and maximise desired objectives such as power output, sensitivity or energy conversion efficiency.

The process of MEMS automated design optimisation and synthesis would be a process that takes specifications of the desired MEMS device – such as volume requirements and resonance frequency and would output the MEMS device design that will match given requirements when produced.

The implementation of system possessing such MEMS design synthesis capabilities would allow greatly enhance speed of production of MEMS devices, and shorten MEMS time to market that will benefit the whole industry of MEMS. Also, such a system can change current process of designing a MEMS as it currently heavily relies on knowledge and experience of a designer and such a system would relieve precious human resources from the drag of designing every element of a MEMS system to a more abstract level of design.

The stochastic EC methods and in particular GA with its ability to perform optimisation of multi–dimensional problems and variety of developed

metrics for measuring performance and methods for applying constraints to the search space makes GAs a very strong candidate for performing MEMS design optimisation and synthesis. The desirability of EC (and especially GA) application to MEMS design optimisation and synthesis problem is supported by the fact that GAs have been proven to be robust and reliable in other device optimisation problems(Weile et al. 1996, Lu & Kota 2003).

Since in GA each solution is represented with genotype and actual evaluation is performed on the phenotype (in case of MEMS, the phenotype is actual model of a device and the genotype is data structure (chromosome), in the GA that describes the building algorithm for this particular model), the way of representing MEMS design holds the uttermost importance – a good design should meet two objectives:

1. Minimise the decision space[4] dimensionality.

2. Maximise the search space[5] dimensionality.

For the correct understanding of the research done in current work, understanding of GA is required; the major aspects of GA are discussed further.

2.4.1 Structure of Genetic Algorithms

The core concept of the GA is a concept of population. Population can be represented as a set of strings which encode alternative solutions (also commonly named 'individuals' or 'phenotypes') to an optimised problem.

In the GA, population is evolved towards better individuals where better individuals are the individuals that represent solutions that have more optimal properties with the passage of time. The time is represented as a sequence of discrete steps with each step having distinct and separate population (that is set of solutions). As the sequence of populations is one–dimensional and ordered by time, the populations can be referred to by the

[4]Decision space is space in which GA undertakes decisions. Usually decisions are represented by mutation and/or crossover of particular phenotypes.

[5]The search space is space in which GA tries to find set of Pareto optimal solutions

number of moment of time they belong to and such case they are called *generations*. So, saying 'ith generation' is the shortcut for saying 'the population of solutions existing at the ith moment of time'.

As every solution in GA usually has two properties – its genotype and its phenotype, there are two distinct spaces GA exists in. A *decision space* is a space where solutions' genotype exists, a space where GA is able to undertake decisions on how to change genotype, what genotype to prefer over another, etc. The second important space GA exists in is an *objective space*, a space where solutions' phenotype exists. The importance of objective space is based on the fact that its parameters are actually evaluated and measured and the direct optimisation process exists in objective space rather than in decision space. GA is commonly expected not to have any knowledge of the structure of decision space or any knowledge of how objective space maps to decision space and the only way for GA to interact with objective space usually is solution evaluation process (a process of expressing phenotype from genotype, also commonly referred to as 'objective function evaluation', or when context is obvious, 'functional evaluation' and the distance of an individual from the point of optimum is called 'fitness').

In general, the only knowledge on objective space GA possesses is the current generation of solutions, so the amount of solutions in current generation (namely 'population size') defines the amount of information on the optimised problem available to GA at the any moment of time.

To generate the next generation from previous generation, GA evaluates fitness of all individuals and stochastically selects a subset of them for the purpose of reproduction. The reproduction process commonly includes the recombination of the genes (also referred to as 'crossover') and the random mutation of genotypes of some individuals. The set of new individuals generated by the crossover and mutation is later used for the construction of the next generation (another common source of solutions for the next generation is *elitism* – a process of copying good individual from the previous generation to the next generation without introduction of changes of any kind, first application of the elitist strategy to the field of GA dates back to the year 1986 as in (Grefenstette 1986) and the process of injection of newly cre-

1	1	0	0	0	0	1	0	1	1	1	1

(a) Binary encoding

0.42	0	0.4	0.5	0.33	0.24	0.4	0.2	1	0.99	0.57	0.73

(b) Real value encoding

Figure 2.3: GA chromosome encoding examples

ated random solutions namely *random immigrants* (introduced by (Forrest & States. 1993)).

Usually GA terminates on two occasions – ether maximum generation count is reached or the solution with the satisfactory fitness level is found. If termination of the algorithm is caused by the limit on the number of generations, it is not sure if a satisfactory solution is found.

To find problems' optimum, a GA requires:

1. way to represent (encode) solution
2. way to evaluate solution

Solution encoding

The encoding is a problem–specific part of any GA, but as usual, there are several most widespread GA encodings that are widely used. The first one is binary encoding (Figure 2.3(a)) and the second is real value encoding (Figure 2.3(b)).

In case of binary encoding, the chromosome is represented by an array (as in computer terminology, or in mathematical terminology, a vector) of bits – data elements where each element can take one of two possible states – 1 (one, on) or 0 (zero, off).

Most common usage for bit–encoded values is conversion of fixed code (i.e. mapping first X bits to first arguments' value, the successive Y bits to second arguments' value, etc.) that is later evaluated in objective function. As an example of fixed code, one may refer to Figure 2.4 as example of

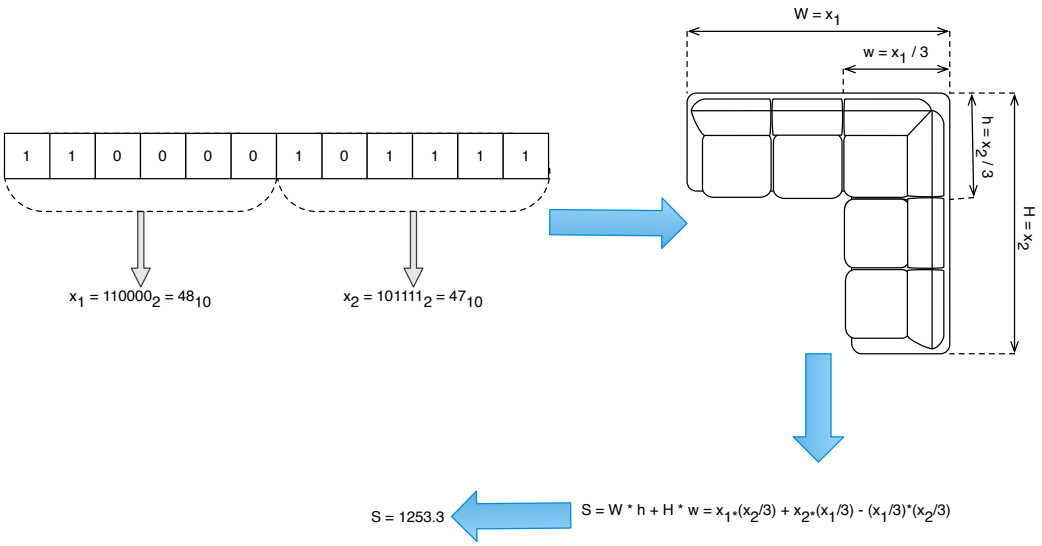


Figure 2.4: Example of mapping of a chromosome to objective function

binary–encoded chromosome mapping to the dimensions of the sofa with the objective function of sofas' area.

Other encodings are also possible – for example, an integer value encoding.

The major weakness of using binary encoding for such mapping is the so–called 'Huffman peak problem', namely there are some bit combinations that are extremely hard to rollover. For example, if GA changes chromosome value only by flipping every bit with probability of 20%, chances of number $15_{10} = 001111_2$ mutating to $16_{10} = 010000_2$ are $80\% * 20\%^5 = 0.0256\%$ as there are 5 bits that need to be simultaneously flipped and one bit that needs to be preserved. While the chances of $2_{10} = 000010_2$ being changed to $3_{10} = 000011_2$ are $80\%^5 * 20\% = 6.55\%$, so in each case the actual evaluated parameter difference equals to 1, but the chances of 2–to–3 mutation happening over 15–to–16 differ $\frac{6.55\%}{0.0256\%} = 256$ times.

Also the binary chromosome is often used for encoding of combinatorial tasks – e.g. when element on position N is 1, then the object $\#N$ is present in the evaluated set, if Nth bit is 0 then the object is absent in the evaluated set.

An alternative widespread approach is to have sequence of real values as chromosome (see Figure 2.3(b)), but this kind of chromosome requires

a development of a special crossover operator, as the common one–point or more general many–point crossovers perform suboptimally when used on real–value chromosome(Agrawal & Deb 1994, Deep & Thakur 2007).

Solution evaluation

The process of solution evaluation is a process of mapping solutions' genotype to the values of objective functions. This process can be simple, as computation of an area of sofa (see Figure 2.4) or extremly complex such as performing FE simulation of an airplane or a MEMS device. In some cases when no simulation software is available or the precision of softwares' output is insufficient, the actual physical prototype can be built and evaluated.

The outcome of solution evaluation is a vector of so–called 'objective functions' – the functions that are important for GA as they are ones driving optimisation process. The vector of objective functions is generated by GA by mapping the required solution properties (such as area, stiffness or the power output level) to the function of the evaluation environment – usually simulation software, but sometimes it can be responses of actually built and tested prototypes.

Pareto front and Pareto optimum

One of the cornerstone concepts of multiobjective GA is the concept of Pareto dominance, Pareto front and Pareto optimum[6]. The concepts of Pareto front and Pareto optimum are defined for minimisation problem.

Pareto dominance is a major concept that Pareto front and Pareto optimum concepts are based on. It was introduced by Vilfredo Federico Damaso Pareto.

The vector of values $\vec{a} = (a_1, \ldots, a_N)$ is said to dominate vector $\vec{b} = (b_1, \ldots, b_N)$ if and only if $\vec{a}$ is partially less than $\vec{b}$, i.e.

$$(\forall i \in 1, \ldots, N : a_i \leq b_i) \wedge (\exists j \in 1, \ldots, N : a_i < b_i)$$

[6]Pareto optimum sometimes is referred as 'Edgeworth–Pareto optimum', but usage of the 'Pareto front' term is much more widespread and is adopted in the further parts of given work.

As the shorthand, the '$\vec{b}$ is Pareto dominated by $\vec{a}$' or '$\vec{b}$ is more Pareto efficient than $\vec{a}$' is also denoted as $\vec{a} \preceq \vec{b}$.

Pareto set In the set of vectors $\Re$, the pareto front is a subset of nondominated vectors $\Re^*$ such that

$$\forall \vec{a} \in \Re^* : \nexists \vec{b} \in \Re : \vec{b} \preceq \vec{a}$$

The Pareto set is an important concept for GA, as GA usually computes set of solutions for the current generations and it is common for GA to give highest chances for offspring generation (via crossover and mutation) for solutions belonging to a given set or to attempt to preserve such solutions by making such solutions an elitist solutions and copying them to the next generation without any modification at all.

Pareto front Pareto front is the plot of the nondominated set of vectors $\Re^*$; for an example of two–dimensional Pareto front please refer to Figure 2.5.

In Figure 2.5, solutions belonging to nondominated set $\Re^*$ are depicted with green diamonds and those dominated are depicted as white diamonds. For each nondominated solution, the area it dominates (for GA that would be objective space) is depicted with black semi–transparent rectangle.

Pareto optimum The Pareto optimal vector (also referred to as 'nondominated vector' or 'Pareto front' as well) is such a vector that is not Pareto dominated by any other vector, in the other words, the pareto optimal vector $\vec{o}$ for the universal set of objective vector values $\Re$ is such that

$$\forall \vec{x} \in \Re : \vec{o} \preceq \vec{x}$$

It is common for multiply vectors to satisfy given condition, and the set of such vectors that satisfy $\vec{o}$ condition is referred to as 'Pareto optimal set'.

Usually, the Pareto optimum is known only for the artificial problems, a problem that was created by researchers for the measurement of the GA

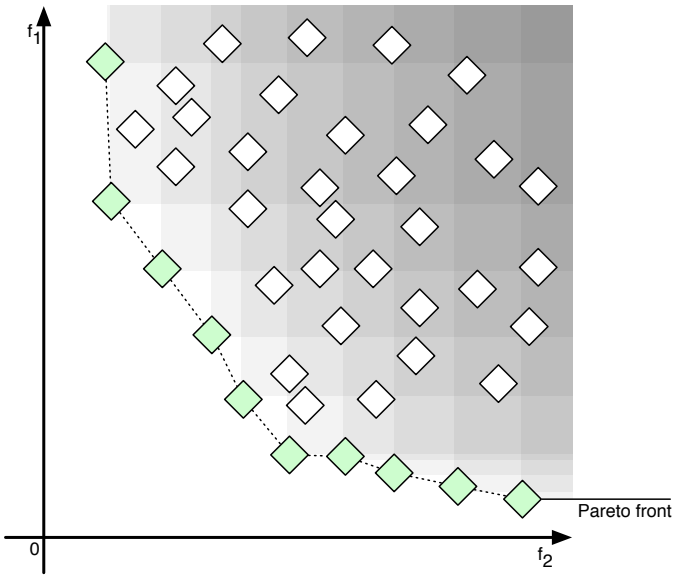


Figure 2.5: Pareto front example

performance. In case of real–world problems (such as MEMS design optimisation and synthesis), the Pareto optimum is not known.

It is easy to see that in case of one–dimensional problems the Pareto dominance concept degenerates into plain 'less than' relation.

2.4.2 Elements of Genetic Algorithms

As one might see from general GA flowchart (Figure 2.6), the GA execution consists of a number of stages:

1. Initial population generation
2. Check for termination criteria
3. Computation of fitness for solutions in generation
4. Selection
5. Crossover
6. Mutation

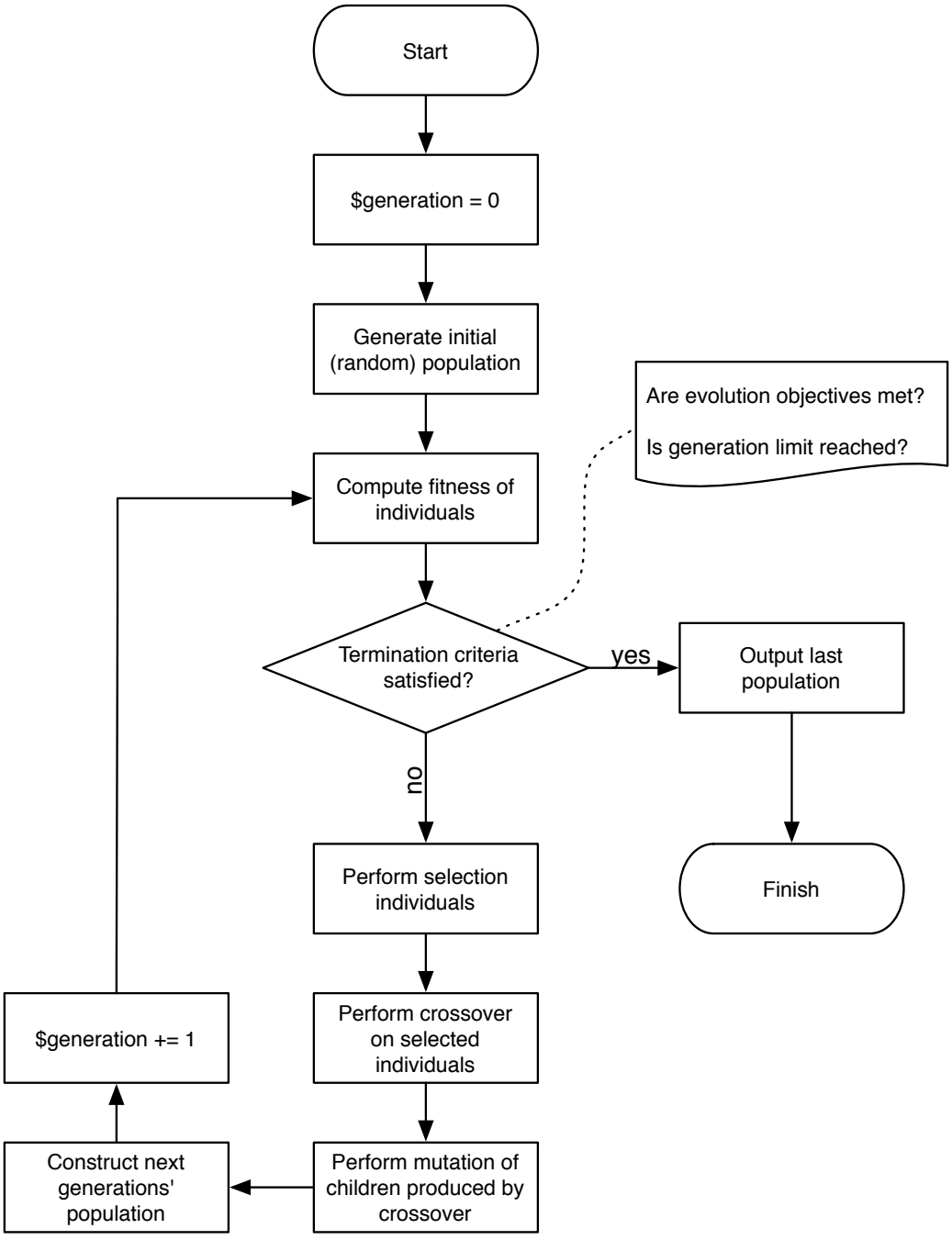


Figure 2.6: Generic GA flow

A number of simple of stages (computation of fitness, initial population generation and termination criteria check) were discussed previously. The later stages were omitted from previous overview discussion as they require the understanding of general GA techniques.

Selection

In GA, 'selection' is an operator that chooses the solution from current generation for further reproduction (that is usually done by performing crossover on the subset of the set of selected solutions and applying mutation to the result of crossover).

Usually selection operators favour solutions that belong to current Pareto front (although there is rarely a 'random' selection operator that just selects solutions with uniform probability and does not depend on the fitness of a solution).

The most popular selection operators are:

- Roulette wheel selection
- Tournament selection
- Top percent selection
- Pareto front selection

Roulette wheel selection is a selection algorithm that selects solutions with probability promotional to its fitness (or rank of the given solution in the current generation when ordered by the count of solutions it is Pareto dominated by).

The commonsense analogy for the roulette wheel selection would be spinning a roulette wheel with the length of an arc assigned for each solution proportional to its fitness.

Tournament selection is a selection algorithm to select one solution. It constructs subset of the solutions (usually using roulette wheel selection) and then selects best solution in the constructed subset.

When compared to roulette wheel selection, this method has an additional selection pressure as the solutions with lower fitness have smaller probability to get selected for reproduction.

Top percent selection is a selection algorithm that randomly choses solution from the top N percent of generation (as generation is ordered by solution rank or fitness).

Pareto front selection is a selection algorithm that randomly choses solution from the current pareto front.

Crossover

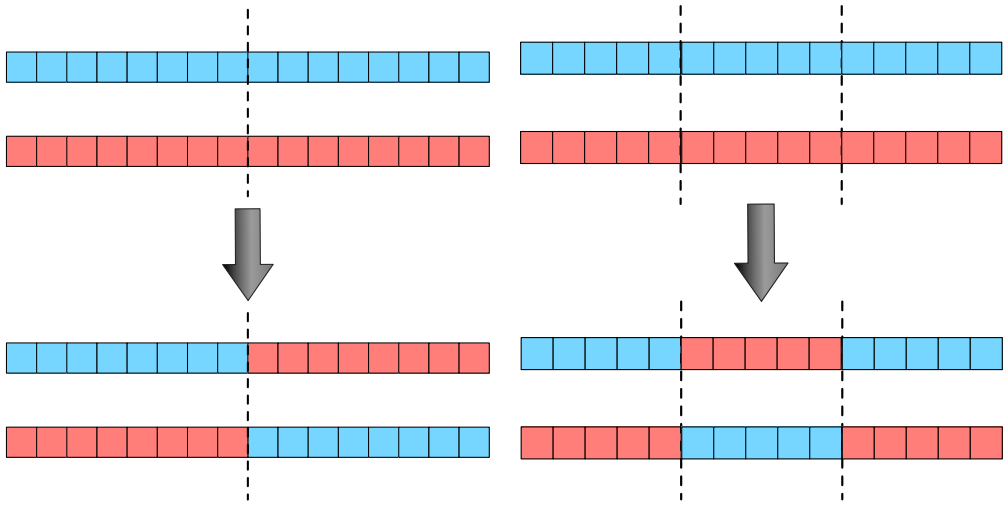
In GA, crossover is a genetic operator that combines two solutions (also referred as 'parents') to produce new solutions (referred as 'offspring'). The crossover is inspired by role of reproduction in course of natural evolution.

Crossover is supposed to spread the good traits of previous generation of solutions to the next generation. In the ideal case, crossover should produce offspring solution with predictable solution (as the random decision and objective space shuffling is performed by mutation operator), dependent on parents position in decision space.

Usually crossover has a user–defined probability of happening. This means that when parents (there are usually two of them) are passed to the crossover operator, there is $N\%$ probability of crossover operator returning generated offsprings and $1 - N\%$ probability of crossover operator returning parent solutions unchanged. The ability of crossover to return unchanged parents provides semi–elitism (as this is not elitism in its original form – returned parents are still subject to mutation and the selection of preserved parents has stochastic elements).

Amongst variety of possible crossover algorithms, three algorithms are most popular:

- One–point crossover
- Two–point crossover



(a) One–point crossover (b) Two–point crossover

Figure 2.7: Crossover operations used in GA

- Simulated Binary crossover

One–point crossover is a crossover algorithm that selects a single crossover point on both parents' chromosomes and swaps the data beyond that point between chromosomes. For graphical illustration please refer to Figure 2.7(a).

Two–point crossover is a crossover algorithm that selects two crossover points on both parents' chromosomes and swaps the chromosome data between given points. For graphical illustration please refer to Figure 2.7(b).

SBX crossover or the 'Simulated Binary Crossover' in full was introduced by Kalyanmoy Deb et al. in (Agrawal & Deb 1994). It is a crossover algorithm created especially for real–encoded values to preserve important trait of one–point crossover executed on the binary chromosome that encoded an integer (please refer to Figure 2.8 for graphical explanation).

In SBX crossover there is a so–called 'spread factor' (denoted as β in formulas) that is either controlled by user (by passing a constant parameter to the GA) or is computed from other properties of the current generation.

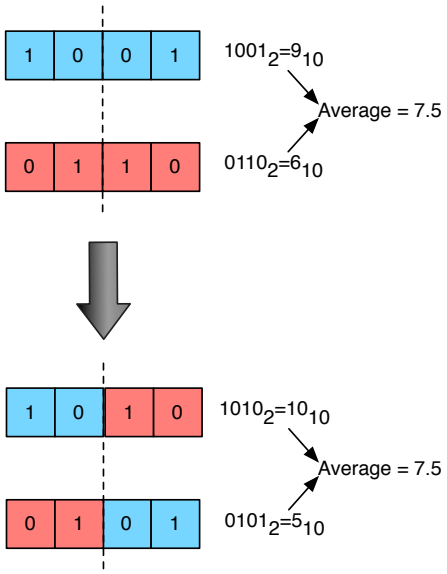


Figure 2.8: Preservation of an average value during binary crossover

The values for each children chromosome (there are two parent solutions participating in SBX crossover process and two children solutions are generated using given parent solutions) are defined as equation 2.1.

$$c_1 = \bar{x} - \frac{1}{2}\beta(p_2 - p_1) c_2 = \bar{x} + \frac{1}{2}\beta(p_2 - p_1) \tag{2.1}$$

Where

— $\bar{x} = \frac{p_2+p_1}{2}$

— p_1 and p_2 are the genes of parent chromosomes

— c_1 and c_2 are the genes of offspring chromosomes

As can be easily seen, the SBX crossover preserves average value ($\bar{x}$) of appropriate genes of every pair of parent and offsprings.

Similar to the case of mutation, the β coefficient is a function using variety of current generations' parameters.

Mutation

Just as in the natural world, mutation is a random error found in genetic code. In GA mutation, an operator stochastically changes one or more gene (gene is one chromosome element) from its initial state to some new, random state. This operation ensures that GA always explores new parts of objective space (as the decision space gets randomly disturbed).

As the mutation disturbs decision space of the GA, it helps GA not to get stuck in local optima. The probability of mutation is usually set by user and is usually set for a low value, like 7%, as the higher levels of mutation would alter solution chromosomes too much making GA to lose information provided by solution parents making GA nothing more than random search.

The list of available mutation operators depend on the encoding used in the GA. For example, when bit encoding is used, the mutation operator is usually either bit inversion operator (replacing value of bit to the opposite one with defined probability) or the bit replacement operator (replacing the bit with new randomly generated one – can be either different from original value or equal to original value).

As for the real value and integer encodings, the most common mutation operator is a uniform mutation that replaces the mutated value with new a random uniformly distributed value in allowed boundaries. A variety of non–uniform mutation operators are used as well. In case of non–uniform mutation, the random value is distributed not uniformly in given bounds, but rather with some more sophisticated probability distribution such as Gaussian or normal distribution.

Some mutation operators do not use constant mutation probability and attempt to define mutation probability as some function dependent on generation number, average fitness of generation or the average distance between solutions in objective space (to increase mutation when solutions are starting to crowd in some small area of objective space).

2.4.3 MEMS design optimisation

As was previously discussed, conventional design optimisation tools (such as gradient–based search) are really meant to find a local optima, meaning that such tools are not powerful enough to perform a full-featured automated MEMS design. The success of evolutionary approaches (and especially GA) in other device optimisation(Weile et al. 1996, Lu & Kota 2003) lead to the decision to apply GA techniques to the field of MEMS design synthesis.

Although the application of EC to MEMS design problem is relatively young field of research and there still are a lot of problems to solve, it is seen that judging by publicised results in EC for MEMS (Zhang, Kamalian, Agogino & Séquin 2005, Zhang 2006) the overall approach holds a big potential as EC algorithms have demonstrated an ability to outperform traditional methods of optimisation.

As most of MEMS naturally have multiple optimisation objectives – for example, in case of meandering resonator, a resonant frequency, per–axis stiffness and the minimisation of surface area, the multi–objective GAs (Genetic Algorithms that are using Pareto front approach, as opposed to the other class of single–objective algorithms) naturally fit the requirements for automoted MEMS optimisation algorithms.

This point of view is further supported by the fact that even the oldest multi–objective genetic algorithm known ('Multi-Objective Genetic Algorithm' (MOGA)(Goldberg 1989)) is able to outperform one of the conventional optimisation methods, such as Simulated Annealing (SA), proving that MEMS design optimisation is truly a multi–objective problem that is best solved using multi–objective optimisation algorithms, according to Kamalian et al. (Kamalian, Zhou & Agogino 2002).

As the usage of GA is computationally expensive and since every act of solution evaluation in GA during MEMS design optimisation process actually results in simulation run, additional constraints should be defined. Introduction of constraints will reduce the freedom for GA to choose combination of particular variables, what will lead to faster convergence(Kamalian, Agogino & Takagi 2004).

Having GA converging faster has the danger of less of an objective space to be probed, therefore reducing chances of finding good solutions. But taking into account reduced computational time and effort, adding constraints to GA process might be worth the risk.

The most common constraints in MEMS design optimisation are defining partial (x–axis or y–axis) or full device symmetry. Also constraining the MEMS device to manhattan geometry is quite common[7]. Introduction of constraints while preserving the number of evaluations used in the GA run can also be performed to make GA to probe bigger volume of a search space, constraining it from concentrating on one cluster of similar phenotypes.

It is common to accomplish GA run with usage of some classical local optimisation technique, such as gradient method. This is done because, the result of GA is likely to be not an optimum, but rather some approximation of optimum. In a way, these two algorithms (GA and conventional optimisation method such as gradient optimisation) are supplementary of each other, as GA is able to probe a wide space of possible designs, but can face difficulties performing precise local optimisation and the conventional local optimisation is unable to find a good starting point for itself, but it is able to find fine local optima.

There are two widely adopted approaches for initial population generation for MEMS design optimisation by the means of GA – a fully random generation or initial population set by the designer (usually from designs collected by scanning through appropriate publications). An interesting alternative approach was proposed in (Cobb, Zhang & Agogino 2007a) of making initial population by reusing results of previous simulations, which would speed up the actual algorithm and potentially increase quality of GA outcome. In a nutshell, this approach proposes usage of a special database for storage of good MEMS designs and their retrieval when needed. With database being enriched by new designs, every time GA simulation run ends, the last

[7]It has been noted that human designers tend to design MEMS with manhattan topology, assuming that such designs are more robust and/or easier in terms of production.
However, this is not true in case of MEMS, as most of the production processes do not depend on the complexity or uniformity of angles used and the precision of production is not usually dependant on the device design.

Pareto front is saved back. This approach also has elements of Case–Based Reasoning (CBR), as designs that satisfy certain set of conditions such as resonance frequency (in case of evolving the MEMS resonator) are chosen from database.

2.5 Work related to circular chromosome

According to (Senturia 2001), the arbitrary MEMS–enabled system can be usually described at different abstraction layers: 'system', 'device', 'physical' and 'process' with each higher level increasing level of abstraction from actual physical device.

This research concentrates on 'physical' abstraction level – level that is one step higher in the abstraction than 'process' level (one that describes actual ways of production of MEMS devices such as etching process).

The 'physical' abstraction level is not taking into account problems of integration of a MEMS device to a MEMS–enabled system (given problem is reviewed on 'system' and 'device' abstraction levels), although it seems to be possible to apply EC to every level of MEMS synthesis(Li & Antonsson 1998).

Due to the development and availability of a wide variety of tools for MEMS device simulation and evaluation using virtual environment (that is without building the actual device design), the automated design techniques such as EC have obtained means for performing design evaluation in fully automated experimental environment rather than execution of experiments by the means of using human–assisted development and response measurement cycle.

Naturally, EC techniques can be applied to any level of MEMS design problem as only requirement for implementation of fully automated EC optimisation is presence of virtual evaluation environment for layer of abstraction that design is optimised in.

Since a number of simulation software by any level of abstraction of MEMS system are currently created and being developed, it can be assumed that it is possible to perform design optimisation at any level of abstraction ('system', 'device', 'physical' or 'process'). As it was previously stated, only

'device' level optimisation will be explored in current work.

First attempts of automated MEMS design were undertaken using traditional methods of optimisation (e.g. gradient–based optimisation) with the help of tools embedded into MEMS simulation packages(Haronian 1995, Brenner et al. 2002, Ye & Mukherjee 1998, Mukherjee, Iyer & Fedder 1998, Fedder et al. 1997). Although GA–assisted design optimisation has demonstrated to routinely yield better results than traditional methods, an impact that can be attributed to discontinuity of and multi–objective nature of MEMS design optimisation problem and GA being able to cope better with such problems than traditional methods.

The problem of MEMS design optimisation has attracted attention of various researchers (Achiche et al. 2007, Lohn, Kraus & Hornby 2008, Zhang 2006, Kamalian 2004). An effect not that surprising if one takes into account importance of MEMS development and size of MEMS industry.

The earliest paper available to author that directly relates to automated design of MEMS at a 'device' level is by Ningning Zhou(Zhou 2002), addressing the most basic conceptual problems such as principles of encoding of a MEMS device to GA chromosome, initial population generation and GA operators (mutation and crossover).

One of the influential innovations described in previously mentioned paper(Zhou 2002), one, that strongly influenced majority of papers related to same area of interest, is an approach of encoding a MEMS device as rooted acyclic tree of nodes i.e. the graph–like structure of interconnected nodes with one node chosen as reference node.

This paper has introduced an idea of matrix encoding for one–dimensional structure of MEMS– an encoding that was later deployed by majority of researchers.

An alternative approach to encoding of a MEMS device was introduced in (Lohn et al. 2008). As the original Zhou's encoding was basically one–dimensional sequence of nodes (with each node having three properties – length, width and angle), the Lohn's paper (Lohn et al. 2008) introduced an alternative encoding scheme with ability of representation of tree of nodes as sequence of commands similar to 'turtle graphics' approach with commands:

'go straight', 'set pen width', 'turn left', 'turn right' and introducing branching (via 'branch start' and 'branch end' commands) and loop structures ('loop start' and 'loop end' commands). But because no intersection detection was used during the evaluation of given algorithm, most MEMS designs created by it are unfeasible. Disregarding unfeasible designs, the author states that 'turtle graphics' approach (Lohn et al. 2008) allowed to reduce amount of computations (both by reducing population size and generations) required to reach design with responses faster than state–of–the–art results published in (Kamalian 2004). But given results can be criticised, as topology of objective space and decision space might have been altered due to allowance of collisions in designs.

Nevertheless, encoding scheme proposed in (Lohn et al. 2008) was successfully put to stricter test in (Hornby, Kraus & Lohn 2008) showing that 'turtle graphics' encoding scheme does indeed lead to increased speed of finding the optimum.

But there are some problems left unsolved by all of known publications:

- Initial population generation

  mentioned in (Zhou 2002), implicitly mentioned in (Lohn et al. 2008) (by assigning custom probabilities to each command during generation).

- Mutation

  the standard mutation operators seems to be not as effective as one would like them to be – even when mutations do now suffer from Huffman peak problems (as in (Lohn et al. 2008)), plain mutations seem to be rather ineffective in GA application to the problem of MEMS design synthesis.

Execution of experiments with same settings and algorithms as ones published by (Zhou 2002) have highlighted yet another problem in applying GA techniques to the MEMS design problem: the premature convergence of device phenotypes – as each good design in the last population of GA seemed to be present multiple times in all performed experiment runs, which is actually

a bad thing as each design copy is stored and evaluated independently, thus wasting computational resources and reducing decision space explored since population size in GA is constant.

2.6 Work related to component–based MEMS synthesis

Generally, the idea of BB usage is so widespread that the whole MEMS design industry seems to be related to it. The MEMS design and simulation software packages like SUGAR, IntelliSense(IntelliSense Corp 2010), NODAS(*NODAS* 2010) and others are shipped with pre–compiled libraries of MEMS elements, or in other words, Building Blocks.

In the Research environment most widely deployed BB simulation package is SUGAR, an open–source software package that runs in MATLAB (a numerical computing environment and fourth–generation programming language). Its development is done by Berkley Sensor and Actuator Center.

The features of SUGAR to which can be attributed its popularity are: a flexible and compact SPICE–like 'netlist' language for defining MEMS design, high simulation speed, powerful visualisation capabilities, extensible architecture and possibilities for integration to wide scope of third–party MATLAB modules.

SUGAR inherits its name and philosophy from SPICE. A MEMS designer can describe a device in a compact netlist format, and very quickly simulate the devices behavior. Using simple simulations in SUGAR, a designer can quickly find problems in a design or try out new ideas. Later in the design process, a designer might run more detailed simulations to check for subtle second-order effects; early in the design, a quick approximate solution is key. SUGAR provides that quick solution.

In software packages, the BBs are included for the convenience of the designer, so that the designer does not need to use design primitives (such as nodes) to re–create all elements of MEMS design from scratch.

Another reason for inclusion of BBs to the software package, (basing on

the analysis of source codes of SUGAR done by author) is the increase in the precision of simulated response values, as there might be more precise behaviour models created for particular MEMS elements than results yielded by the simulation of a set of interconnected atomic elements as nodes in case of SUGAR.

Also, a variety of researchers such as Zhang et al.(Zhang et al. 2005, Zhang 2006), Raffi Roupi Kamalian (Kamalian 2004) and Zhou (Zhou et al. 2002, Zhou 2002) have used MEMS encoding that incorporated the ability to handle BBs (or 'clusters', according to the terminology used by these researchers).

The usage of BB was taken even further by Cobb et al.(Cobb et al. 2007a, Cobb, Zhang, Agogino & Mangold 2008a, Cobb, Zhang & Agogino 2007b) when they introduced the hierarchical classification system for MEMS and their components.

The widespread usage of BBs in the domain of MEMS design illustrates the desirability of the creation of MEMS BB libraries (this was mentioned throughout automated MEMS design synthesis literature as well). And lack of the diversity of BBs used in the research on MEMS automated design synthesis highlights the need for exploration of automated BB database creation.

2.7 Work related to process–induced error tolerant layout synthesis

The major share of research on MEMS error tolerance is performed using the analytical models of MEMS.

A variety of techniques were applied and developed by a multitude of researchers addressing the given problem with optimisation of analytical models – topology optimisation (Maute & Frangopol 2003), and robust design synthesis by conventional approaches (Sedivec 2002, Liu, Paden & Turner 2002, Ma & Antonsson 2001), Simulated Annealing (SA) (Hwang, Lee, Park, Lee, Cho & Lee 2003) and GA (Fan et al. 2007, Fan, Liu, Sorensen &

Wang 2009).

As this work focuses on MEMS evolutionary synthesis by means of GA, it would be of interest to review the error–tolerant MEMS layout synthesis and the state–of–the–art research on robust optimisation performed by Zhun Fan et al. (Fan et al. 2007, Fan et al. 2009).

The proposed approach is based on the technique of designing error–tolerant MEMS by the means of robust optimisation as described for the first time by Peter Josef Sedivec in (Sedivec 2002).

In these works, actual MEMS design is denoted as $\vec{x}$, production error vector is denoted as $\vec{\delta}$ and the overall (including disturbance introduced by $\vec{\delta}$) designs' response estimate is denoted as $f(\vec{x}, \vec{\delta})$.

As dictated by robust optimisation approach, the $f(\vec{x}, \vec{\delta})$ is approximated by first–order Taylor series expansion (equation 2.2).

$$f(\vec{x}, \vec{\delta}) \cong f(\vec{x}, 0) + \nabla_{\delta} f(\vec{x}, 0)\delta \tag{2.2}$$

With $\nabla_{\delta} f(\vec{x}, 0)$ being gradient of $f(\vec{x}, \vec{\delta})$ with respect to δ.

Further mathematical operations that were initially introduced and can be seen in (Sedivec 2002) lead to composite robustness–enabled function that was later used as GA objective in (Fan et al. 2007):

$$\min_{x}(N(\vec{x}) + D(\vec{x}, \Omega)) \tag{2.3}$$

Where $N(\vec{x})$ and $D(\vec{x}, \Omega))$ are defined as

$$N(\vec{x}) \equiv \left(\frac{f(\vec{x}, 0) - \vec{f}}{\vec{f}} \right)^2 \tag{2.4}$$

$$D(\vec{x}, \Omega) \equiv \frac{1}{\vec{f}^2}(\nabla_{\delta} f(\vec{x}, 0)\Omega\nabla_{\delta}^T f(\vec{x}, 0)) \tag{2.5}$$

With $f(\vec{x}, 0)$ being designs' $\vec{x}$ response with zero error vector, or, as it is denoted in current work, an 'ideal design' response.

It can be seen that equation 2.3 is defined as conjunction of two functions – $N(\vec{x})$ and $D(\vec{x}, \Omega))$ (where Ω is a pre–defined matrix of production errors applied to 'ideal design' $\vec{x}$). With $N(\vec{x})$ function is the measure of the

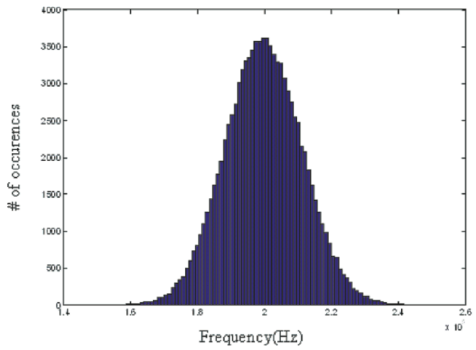


Image from (Fan et al. 2007, p. 530)

$\mu = 199.9kHz$

$\sigma = 11.3kHz$

(a) Non–robust design

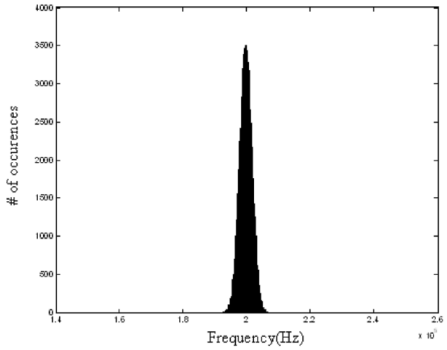


Image from (Fan et al. 2007, p. 531)

$\mu = 199.7kHz$

$\sigma = 1.95kHz$

(b) Robust design

Figure 2.9: Histograms of the natural frequencies of designs of crab–leg resonator acquired by (Fan et al. 2007)

designs' performance towards target response $\vec{f}$ and $D(\vec{x}, \Omega)$ is the measure of $\vec{x}$ tolerance towards production errors.

As displayed in (Fan et al. 2007), the robust optimisation approach by the means of GA has indeed increased the robustness of the resulting MEMS designs while preserving the mean value of target response (please refer to Figure 2.9 for exact values of results).

2.8 Research gap

In given thesis a number of research gaps was identified and addressed.

First research gap identified is inefficient representation of MEMS in chromosomal structures deployed for MEMS layout synthesis, big evaluation count and inability for GA to remember and reuse desirable features of designs throughout single design synthesis run.

This problem is addressed in chapter 3 of this thesis by introduction of circular chromosome. Its probabilistic expression properties and linkage learning capabilities allows for GA to preserve elements of MEMS designs that were previously used but rejected allowing for evolutionary optimisation to have more freedom while synthesising MEMS layout design.

Second research gap is inability for GA to automatically collect and reuse desirable MEMS design elements is some form of global memory bank. Although desirability of having such good design memory bank is demonstrated in the literature (Cobb & Agogino 2006, Cobb et al. 2007a, Cobb et al. 2007b, Cobb et al. 2008a, Cobb, Zhang, Agogino & Mangold 2008b), all of previous research have manually collected data for populating such memory bank. To unlock full potential of automated MEMS design synthesis process of acquiring knowledge by GA should be automated just as all other steps of automated design synthesis are.

Linkage learning capabilities of circular chromosome provide good starting point for implementing automated design element collection, extraction and re–usage techniques. Problem left unaddressed by deployment of circular chromosome is process of automated conversion of good design elements into design elements that can be reused in future GA runs. This problem was addressed in chapter 4.

Third and last problem addressed is instability of properties of automatically generated MEMS designs when subject to process of MEMS production and its errors. Previous research on given topic is done only in case of one objective design synthesis using analytical models for acquiring estimated MEMS responses (Fan et al. 2007, Sedivec 2002). The multi–domain nature of MEMS and general lack of analytical models limits usability of developed techniques.

In the chapter 5 given problem was addressed by attempting to adapt previously published techniques to the domain of multiobjective production error tolerance have yielded suboptimal results, several enhancements were proposed, tested and and having their performance compared. The justification for performance of each approach was proposed.

2.9 Summary

The application of EC techniques to the field of MEMS currently yields promising results. Optimisation still has a lot of limitations and behaviour of evolutionary optimisation technique in the field of MEMS design synthesis

Date	Contributors	Methods	Description
1998	H Li and E. K. Antonsson (Li & Antonsson 1998)	GA	Showing that MOGA can be used to synthesise mask for production
1999	H. Li and E. K. Antonsson (Li & Antonsson 1999)	MOGA	Simple mask shapes framework for the mask synthesis process
2001	L. Ma, E. K. Antonsson (Ma & Antonsson 2001)	MOGA + noise	Simple mask shapes noise to phenotype to create robust design
2001	N. Zhou, B. Zhu, A. M. Agogino and K. Pister (Zhou, Zhu, Agogino & Pister 2001)	MOGA	Meandering springs MOGA is able to do the device level design optimisation
2002	N. Zhou, A. M. Agogino and K. S. Pister (Zhou et al. 2002)	MOGA	Introduction of CBR
2004	R. H. Kamalian, A. M. Agogino and H. Takagi (Kamalian et al. 2004)	MOGA + constraints	Constraints can increase search speed and thus give better results
2005	Y. Zhang, R. Kamalian, A. M. Agogino and C. Sequin (Zhang et al. 2005)	MOGA + gradient-descent optimisation	Local optimisation can increase result quality, but needs MOGA to find the global one
2005	Z. Fan, J. Wang, and E. D. Goodman (Fan et al. 2007)	MOGA	Robustness can be introduced to the MOGA process
2006	C. Cobb, Y. Zhang and A. M. Agogino (Cobb et al. 2007a)	MOGA + CBR	MEMS resonator CBR can eliminate need for starting designs
2008	G. S. Hornby, W. F. Kraus and J. D. Lohn (Hornby et al. 2008)	GP + noise	Location noise and pre–stress can ensure robustness

Table from (Benkhelifa et al. 2010, page 37)

Table 2.2: Major points in research of MEMS design synthesis by GA

remains largely unexplored. As is evident from Table 2.2, the techniques of evolutionary optimisation of the MEMS devices are progressing fast, but as given field is still in its infancy, there are a lot of problems that remain to be addressed and resolved.

For instance, the GA is currently allowed to act only on the set of atomic building blocks provided by designer without having any possibility to change them or to create any new building blocks, which might handicap performance of the GA, as designers might dismiss some building block variants just basing on their personal perception.

Another unexplored way of enhancing the MEMS design synthesis might

lie in the domain of enhancing the GA itself by introduction of more intelligent GA operators, such operators that would behave more intelligently allowing for GA to behave more intelligently. Therefore, reducing the needed evaluation count.

Chapter 3

Circular chromosome

3.1 Introduction

The problem of automated design synthesis by the means of GA always faces two conflicting targets – to evolve more sophisticated designs matching desired set of properties more precisely and to evolve good designs in shorter time by performing less functional evaluations.

As design time constraints become more strict and non–forgiving with each successive problem, every effort in optimisation has to be made. As choosing appropriate encoding is one of the problems that lies at the heart of the GA, it strongly affects performance of the GA for particular application.

Commonly, GAs incorporate various stochastic elements such as selection, mutation and crossover operators[1] (not necessarily all of them have to be stochastic, but at least one of them has to as stochastic behaviour is found in nature and GA are considered a biologically–inspired algorithm).

The problem of encoding used by GA to solve particular problem is commonly perceived to be an important decision that has to be made. It is commonly perceived that perfect GA encoding would not be inter–linked, meaning that each piece of information (bits in case of boolean values, real number in case of real–coded values, etc.) control only one aspect of phenotype and therefore, changing one gene in the chromosome would ideally

[1] Common GA algorithm flow can be seen in Figure 2.6

result in change of only one aspect of phenotype properties and not impact other properties that are supposedly controlled by other genes.

As a matter of fact, all current GA encodings deployed to date for MEMS design optimisation problem have inter–linked genes. And there is little chance for encoding without linked genes, as MEMS usually have complex topology, that can only be described by multiple inter–connected components. As GA–assisted design process requires some level of abstraction to be able to represent as big as possible number of MEMS designs in desired format (i.e. 3D models, nodal representation as in SUGAR input files or even at higher abstraction level such as component–based designs used by a number of previous researchers) to be reflected in the data stored in the genes of each solution.

The actual MEMS design is whole and indivisible in terms of its measured or simulated responses (such as resonance frequency). Since each element of a MEMS design influences integral device responses and thus design performance and variety of other properties, such as design correctness – as in case of working with SUGAR(a MEMS simulation package) nodal representation, changing angle of a junction of one node pair might result in collision is some other, distant, set of nodes (as can be seen in Figure 3.1).

The task of developing near–ideal universal encoding for MEMS device design is a problem extremely hard to solve. And as all GA encodings applied to problem of MEMS device synthesis seem to suffer from design element interconnection problem (the problem of one element influencing properties of nearby elements or of a whole design). This problem cannot be referred to as imperfection in particular GA encoding, but rather is a natural effect of MEMS design synthesis. To address the problem of single MEMS element influencing global MEMS responses, an introduction of GA algorithm with linkage learning capabilities for the MEMS design optimisation problem seems to be desirable.

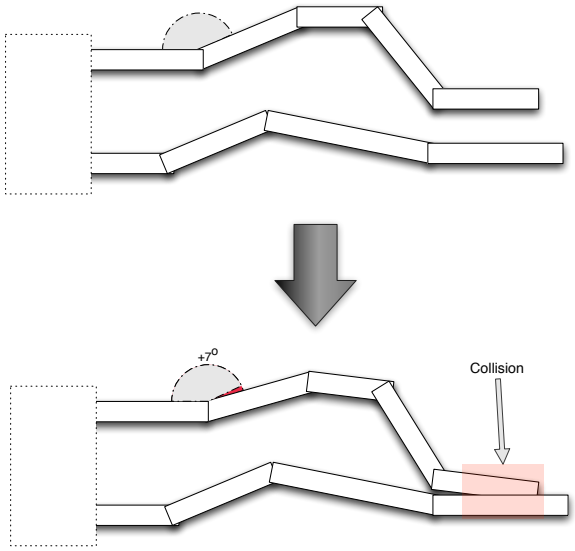


Figure 3.1: Example of mutation resulting in distant collision

3.2 Proposed enhancements

Two major problems of MEMS design optimisation by the means of GA were left without proper attention – the effects of mutation on the performance of GA and the problem of generation of initial population. The logical question to ask would be if the proper GA flavour was chosen for the MEMS design optimisation problem at all, as the non–locality of effects of mutation might limit performance of GA–enabled MEMS optimisation process.

The inter–dependance of MEMS components and hence the non–locality of mutation has led to the decision to introduce Linkage learning Genetic Algorithm (LLGA)–like algorithm to the MEMS design optimisation problem with ability to detect linkage between nodes of the MEMS device and to be able to perform better than common GA used by the previous researchers.

Just as most of scientific progress during last decades was inspired by nature, the data structure proposed was inspired by structure of the bacterial Deoxyribonucleic Acid (DNA). A 'circular chromosome' was proposed in attempt to introduce linage learning behaviour to the MEMS design optimising GAs. The proposed chromosome structure introduces introns (noncoding gene sequences) to the GA for allowing the algorithm to preserve bits of de-

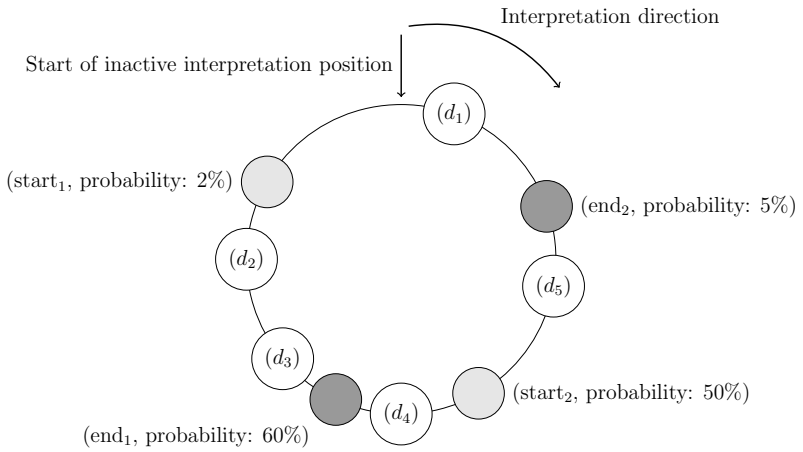


Figure 3.2: Proposed chromosome structure

signs even if given design parts were 'thrown away' by the current generation of solutions.

As can be easily concluded from schematic depiction of the circular chromosome in Figure 3.2, the major difference between 'plain' chromosome commonly used in GA and proposed 'circular' chromosome is that circular chromosome has no distinctive begin or end. But since actual MEMS design building process needs to start somewhere and to end somewhere as well, used encoding scheme (that is not restricted to any particular scheme, but in current work the encoding used by Zhou(Zhou et al. 2001, Zhou et al. 2002) and other Berkley researchers(Zhou et al. 2002, Clark et al. 1998, Kamalian 2004, Zhang 2006) is used) is extended with two additional genes – one 'start' gene and one 'end' gene, so the sequence of chromosome genes actually expressed would be one that starts with 'start' and ends with 'end' genes.

To address diversity issue spotted previously during the runs of experiments with same settings as described by (Zhou 2002), probabilities of evaluation to the 'start' and 'end' commands are used by allowing multiple 'start' and 'end' commands in single chromosome.

The common process of expressing genotype to phenotype would be the start of interpretation process on the random position of the circular chromosome, and initially all read genes are not expressed while the interpretation position is moved according to the direction of interpretation until the

'start' gene is not met. When 'start' gene is faced, the interpretation process switches to the active phase with probability that is equal to the value of parameter assigned to given gene. If interpretation process switches to active phase, then all data genes faced during the successive interpretation process are expressed to the phenotype until the stop gene is faced. When the given gene is faced, the interpretation stops with probability of argument value that is attached to the given stop gene. Since it is possible for one stop gene in the gene to exist and that 'stop' gene can have execution probability smaller than 100%, the active phase of interpretation process has infinite mathematical expectation. This potential problem was addressed by introduction of artificial limit for length of expressed genotype[2]. The actual implementation selecting start position is different from theoretical one due to performance reasons. In the implemented realisation, start position was chosen from all available start positions where probability of starting in any given start position n was equal to $\mathbb{P}_n = \frac{p_n}{\sum_{\forall n}^{i} p_i}$, where p_x is probability of starting in position x.

For example, with initial state of interpretation process as depicted in Figure 3.2, there are two possible start positions – start_1 and start_2 with probabilities of 2 and 50 percent, respectively. The actual probability for starting interpretation on the start_1 is $\frac{0.02}{0.02+0.5} = \frac{0.02}{0.52} \approx 3.85\%$ and for the start_2 it is $\frac{0.5}{0.02+0.5} = \frac{0.5}{0.52} \approx 96.16\%$.

In case start_1 got lucky and interpretation started from that position, the gene sequence, that will be used for the generation of phenotype will be $[\underbrace{d_1, \overbrace{d_5, d_4, d_3, d_2, d_1}^{Ntimes}}_{length \leq 1000}]$ (where d_n is nth gene in the chromosome) where $N \in \mathbb{N}$ in case if interpretation ends with end_2. Or alternatively, in case if it ends with end_1, the output sequence will be $[\underbrace{d_1, d_5, d_4, \overbrace{d_3, d_2, d_1, d_5, d_4}^{N\ times}}_{length \leq 1000}]$ where $N \in \mathbb{N}$ (in given example the maximum interpretation length is set to 1000).

Adaptation of given probabilistic expression technique is also desirable

[2]In the setup of experiments done during given research, the maximum interpretation length was set to 1000.

in the sense that MEMS device design cannot be easily broken down to the positionally–independent BB. Given a GA way to explore same sequences of genes in various neighbourhoods (as the prefix and postfix gene sequences change because of stochastic properties of gene expression algorithm), property that might be desirable if one attempts to tackle the automated BB generation from phenotypes. This approach is backed up by the fact that it has been shown in (Chen & Goldberg 2002), that the introduction of start expression gene increases ability of LLGA to identify and separate BB in comparison with 'classical' approach where interpretation of gene starts on the static position of chromosome.

Interestingly, the Extended Probabilistic Expression 2 (EPE–2) chromosome was proposed by Georges Raif Harik in (Harik 1997), a structure that was developed for solving combinatorial problems and has special gene structure for that particular purpose – in it each gene (a chromosome element) is composed of two sub–elements – position and data (as displayed in Figure 3.3). When chromosome is interpreted, for any given position only data thaw read last has effect on actual phenotype (e.g. in Figure 3.3 example, the d_2 is effective data for position 0 and d_3 is effective data for position 1, whereas any previous genes containing data are overridden during process of interpretation). The EPE–2 chromosomal structure is similar to circular chromosome proposed in this work, as it was created for different purpose and its emergence was dictated by Georges Raif Harik as linkage–learning crossover operator.

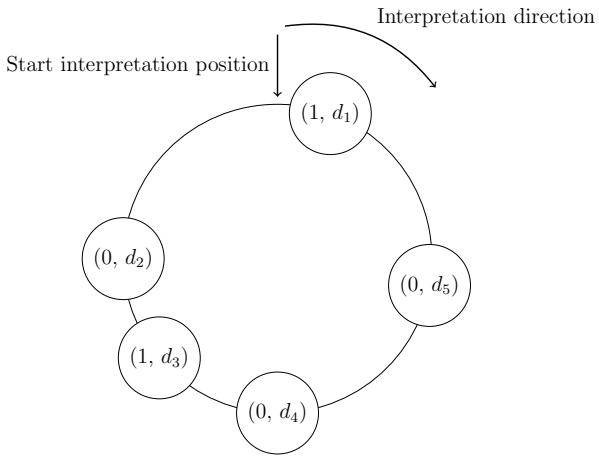


Figure 3.3: EPE–2 chromosome structure

3.3 Description of proposed approach

The MEMS design synthesis by the means of GA was actively researched in Berkley and the earliest most influential publication by the Berkley researchers in given field is (Zhou et al. 2002).

In (Zhou et al. 2002), the authors have introduced the decomposition of MEMS devices to a set of building blocks – e.g. in case of SUGAR (a MEMS numerical simulation program developed in Berkley), each device is represented as a set of interconnected nodes, but since GA usually requires tradeoff between search space exploration and performance. The bigger space it is searching, the more variables and variable configurations it needs to check, the more function evaluations it needs to perform, creating some pre–built groups of nodes, called 'blocks' is desired, as it allows introduction of complex components (for example, serpentine string) without introducing too many new variables.

In case of meandering resonator, the device would be decomposed to a set of two kinds of blocks – a block that represents central mass and four blocks that represent design of each of four legs of meandering resonator.

The mass was described by two parameters – length and width of beams (given mass is composed of four beams with given length and width arranged a square pattern). Then, each of four corners of resulting mass would be

Type	*Connected to*	*Design parameters*
mass	[1 2 3 4]	[L W]
block #1	[1]	[(l_1 w_1 θ_1) (l_2 w_2 θ_2) (l_3 w_3 θ_3) ...]
block #2	[2]	[(l_1 w_1 θ_1) (l_2 w_2 θ_2) (l_3 w_3 θ_3) ...]
block #3	[3]	[(l_1 w_1 θ_1) (l_2 w_2 θ_2) (l_3 w_3 θ_3) ...]
block #4	[4]	[(l_1 w_1 θ_1) (l_2 w_2 θ_2) (l_3 w_3 θ_3) ...]

Figure 3.4: Meandering resonator GA representation according to (Zhou et al. 2002)

attached to the legs that were described as sequence of three-element tuples, where each tuple represents single node and each element in appropriate tuple describes one of three encoded (and thus, variable) parameters of given node – width, length or angle.

The structure used for device syntheses in (Zhou et al. 2002) is depicted in Table 3.4 and it can be seen that the main limitation of it is that there is no proper coding for structures that contain branches and it can only encode sequence of blocks without cycles. As there is no possibility to evolve cycles during GA run, any element containing cycles of nodes needs to be hard-coded into phenotype–to–genotype expression algorithm as separate block type, therefore limiting GA and possibly leaving good solutions unexplored (for example using proposed coding, there is no possibility of exploring resonator design with triangular mass and three legs similar to one depicted in Figure 3.5).

Given representation of MEMS was further extended in (Zhang 2006). The approach proposed in this work is similar to encapsulation principle in computer science – there is primitive object (a node – the rectangle with arbitrary length, width and angle) and there are various objects that can contain other objects. For visual Unified Modelling Language (UML) representation of approach taken in (Zhang 2006), see Figure 3.6.

For example, the MEMS resonator depicted in Figure 3.7(a) according to (Zhang 2006) is represented as Figure 3.7(c) (with subcomponents as defined in figures 3.7(e) and 3.7(f)) when using component expression of Table 3.7(b). But not all of components used in this figure are resolved directly to sequence of nodes. Some of them (such as I–shaped centre mass or polyline spring)

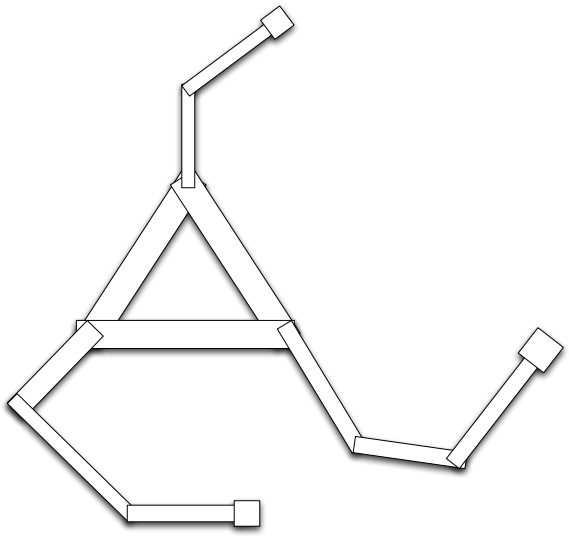


Figure 3.5: Triangular mass resonator

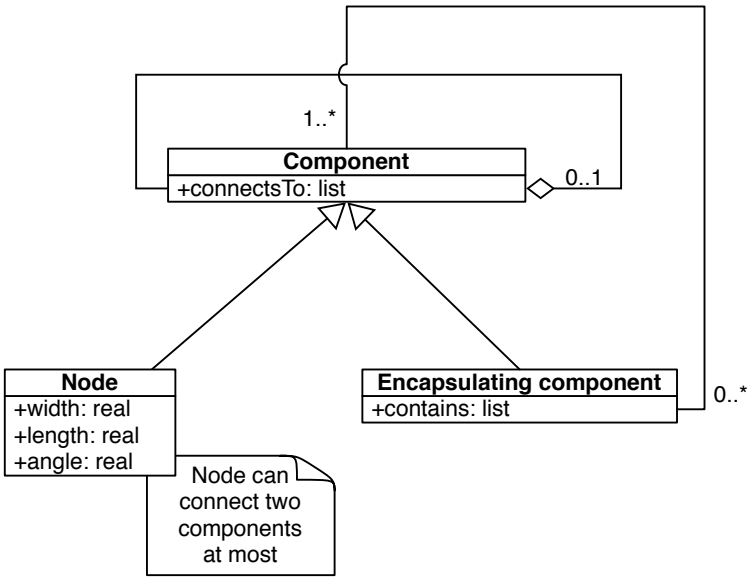


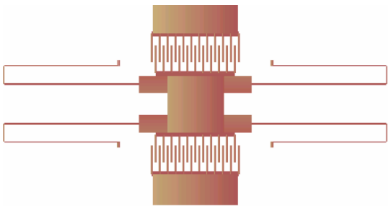
Figure 3.6: Objective MEMS structure UML representation according to (Zhang 2006)

are composite. These elements are assigned some high–level properties (e.g. width, length and others – as can be seen in Figure 3.7(d) with incapsulated data structure displayed in Figure 3.7(e)) which are subject to GA influence. These high-level elements are later resolved to lower-level elements, that have representation in actual MEMS schematics, are assembled to SUGAR package and are sent for simulation.

This attempted encapsulation has disadvantage of being restrictive to the possible phenotypes of the device – once design element is defined by human, it can change only in a certain way and nothing can change phenotype in a way that is not intended by human. For example, for the component for meandering flexure (see Figure 3.8) there is no way for GA to add or delete a node or change one of flexures' internal angles. Also, there is no way for GA to inject one component inside other component (in example with meandering flexures that would be adding one flexure with start somewhere in the middle of other, already existing, flexure).

Although the GA representation used by Zhang does allow to mimic previously described flexure injection if there happen to be two flexures initially created by GA. For GA it would be possible to add one extra component to the end of first flexure before second flexure, so a component seems to be inside of given composite flexure[3]. But this scenario has extremely small chance of happening.

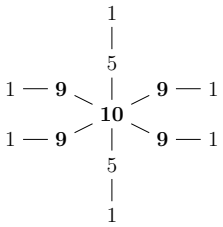
[3]Here, the 'composite flexure' means that a single visible flexure on the phenotype is actually two separate flexures in the genotype, that happen to be arranged in such a manner.



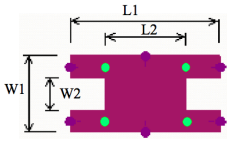
(a) Resonator

Gene type	MEMS design component
10	I-shaped center mass
9	Polyline spring
7	Beam for I-shaped center mass
5	Comb drive
4	Mass plate
1	anchor
0	beam

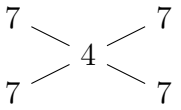
(b) Resonator component table



(c) Complete resonator design



(d) I–shaped centre mass schematics



(e) Type 10 (I–shaped centre mass) subtree

0 — 0 — 0

(f) Type 9 (Polyline spring) subtree

Figure 3.7: Component design for resonator example (Zhang 2006, pages 49—50)

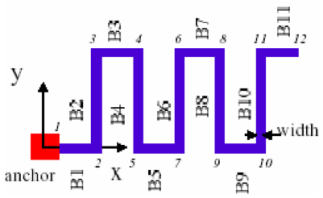


Figure 3.8: Meandering flexure by (Zhang 2006, page 20)

3.4 Motivation

The work of (Zhang 2006, Kamalian 2004) on Interactive Hybrid Computation (IHC) has demonstrated that there are such phenotype elements that might be non–beneficial for current design, but may lead to far better results when given properties are preserved over generations and allowed to evolve.

The problem of currently non-beneficial (or less beneficial than some of current combinations of properties) mutations with good potential when search space is big and solution population is small (and this is the case with MEMS automated design as well) was tackled and resulted in showing that introduction of introns[4](Levenick 1991) is beneficial for such problems.

The MEMS automated design has been implicitly shown to be struggling with the problem of linked genes as both of (Zhang 2006, Kamalian 2004) proposed the MEMS design automation program with database of pre-computed good MEMS building blocks.

In the context of biological genetics, genetic linkage is a tendency for certain genes to be passed from parents to children together[5], whereas in GA context learning linkage means learning what genes should be inherited together, similar to biological systems. This type of linkage was closely inspected in (Harik 1997). (Harik & Goldberg 1997) introduced two mechanisms that any LLGA should implement – linkage skew and linkage shift. Linkage skew is mechanism that ensures increased probability of linked chromosome element survival during crossover process and linkage shift is a tendency of LLGA to form so–called deceptive building blocks – building blocks

[4]An intron is a gene element that does not express itself into phenotype

[5]In the case of biological systems this is done by placing genes physically close together, so crossover would have low probability of separating given genes.

that encode locally optimal phenotype instead of building blocks that encode globally optimal phenotype.

The case with building block–oriented GA chromosome representation is a clear demonstration of linkage implemented by designer (as opposed to learning by GA itself) – defining a set of linked components. In simple case, this includes a sequence of linked nodes that describes some MEMS component, as every component has the definition of variables that GA is allowed to change in optimisation process and number of fixed variables that GA is not allowed to change, e.g. in case of (Zhang 2006, Kamalian 2004), for all building blocks with the exception of resonator legs, the GA is not allowed to delete, add or otherwise modify the single nodes that given blocks contain. The resulting dimensionality of decision space is greatly reduced with no proof given to justify which particular decision leads to the development of a particular BB.

The Probabilistic Expression (PE) introduced in (Harik 1997) and EPE (Extended PE) introduced later in (Harik & Goldberg 2000) by same author have been widely used in linkage–learning GAs and have shown to be beneficial in linkage–learning GAs.

Since MEMS automated design seems to benefit from the exchange of MEMS elements between solutions (in case of (Zhang 2006, Kamalian 2004) via database of MEMS elements, or as in the case of linkage–learning GAs by crossover), the circular chromosome structure with linkage–learning genetic operators should be beneficial for MEMS automated design process.

Because EPE–2 chromosome structure has been shown to be good in linkage learning GAs, the introns incorporated in given chromosome structure could address promising mutation propagation (as described in (Levenick 1991)) addressing the problem (Zhang 2006, Kamalian 2004) addressed by using IHC.

3.5 Experimental setup

3.5.1 Meandering resonator

For the first set of experiments, the EC design synthesis of meandering resonator was chosen as the meandering resonator is the simplest and well known device to be simulated in SUGAR package. The settings used in (Zhou et al. 2002) (see Table 3.1) were taken as base with only necessary changes applied – the population size reduced to highlight the performance gain because of deployment of circular chromosome and defining terms specific to circular chromosome, such as maximum interpretation length and the chromosome size constraint (as in case of circular chromosome, the size of chromosome is not equal to the size of data read and parsed into phenotype).

The settings, that were changed or added to original settings by Zhou, for circular chromosome can be seen in Table 3.2.

As can be seen, from data provided by (Zhou et al. 2002) (Table 3.3), the average error in resonance frequency is 1424.16 $rad \cdot s^{-1}$, the average error in stiffness by X axis (K_x) is 0.065 $^N/_m$ and average error in stiffness by Y axis (K_y) is 0.02 $^N/_m$.

In case of circular chromosome, by the end of 50th generation, there were 49 solutions, or 4.9 solutions per experiment with resonance frequency error strictly smaller than 1424.16 $rad \cdot s^{-1}$ and minimum error of 4.723 $rad \cdot s^{-1}$ on the pareto front of 50th generation, 6 solutions with strictly smaller error in stiffness by X axis (with minimum error of 0.014 $^N/_m$ on the pareto front of 50th generation) and no solutions with smaller error in stiffness by Y axis (the minimum error by the end of 50th generation was 0.137 $^N/_m$ and global minimum 0.04 $^N/_m$).

By the end of 50th generation, there were 15.9 solutions on the pareto front with 14.9 solutions being on Pareto front in average during whole history of GA optimisation; one might conclude that given population is far

[6]Originally in (Zhou et al. 2002), it reads "Resonant frequency (Hz)", but according to (Lohn et al. 2008) this is a misprint and should be read as "$rad \cdot s^{-1}$". This is also supported by the fact that in same paper Zhou states that target resonant frequency is 93723 $rad \cdot s^{-1}$(Zhou et al. 2002, page 5).

Meandering resonator design experiment settings		
Algorithm settings		
Algorithm		MOGA
Elitism rate		5%
Population		400
Generation count		30
Crossover probability		0.7
Mutation probability		0.1
Selection operator		Roulette wheel
Resonator design parameters		
Central mass		Four 100x20x2 μm beams laid out in a square.
Node width	min	2μm
	max	20μm
Node length	min	–
	max	400μm
Node count	min	–
	max	6
Objectives		
Lowest natural frequency		14.916 kHz (93723 $rad \cdot s^{-1}$)
X stiffness		1.90 $^{N}/_{m}$
Y stiffness		0.56 $^{N}/_{m}$

Table 3.1: Meandering resonator experiment settings according to (Zhou et al. 2002)

Circular chromosome settings		
Maximum interpretation length		1000
Population		200
Experiment count		10
Chromosome length	min	3
	max	16

Table 3.2: Settings for experiments with circular chromosome

Resonant frequency $(rad \cdot s^{-1})^6$	K_x (N/m)	K_y (N/m)
93746	1.80	0.567
93859	1.82	0.602
92632	2.00	0.559
93350	1.95	0.557
94290	1.84	0.59
87368	1.90	0.52

Table 3.3: Resulting design parameter examples according to (Zhou et al. 2002, figure 7)

from converging to the local minima.

3.5.2 Flexure resonator

The flexure resonator experiments were performed by Ying Zhang and Raffi Roupen Kamalian, each of whom have performed separate set of experiments with same GA settings (settings used by these researchers can be seen in Table 3.4). In circular chromosome experiments, these settings were used with the changes seen in Table 3.2.

The stiffness ratio constraint was introduced in (Kamalian 2004) to force GA to producing designs that would oscillate only in one axis, as flexure resonator incorporates two comb drives, that would restrict movement in one axis, thus, rendering any results obtained without such constraint infeasible.

Both there researchers performed set of experiments for various device constraint cases. The constraint cases are defined as following.

1. Unconstrained leg angles, asymmetric legs
2. Unconstrained leg angles, vertically symmetric legs
3. Unconstrained leg angles, symmetric legs
4. Manhattan angles, symmetric legs

In both cases, only the pareto front solutions with resonant frequency within 5% of target frequency were analysed.

Flexure resonator design experiment settings		
Algorithm settings		
Run count		25
Population		400
Generations		50
Other settings		Same as in Section 3.5.1
Resonator design parameters		
Central mass		100x100μm plate with 4 30x50μmbeams attached. Plus mass of the comb drives (11 fingers, 50μm long by 4μm thick with 3μm gap, plus a 4μm thick spine)
Node length	min	10μm
	max	100μm
Node width	min	2μm
	max	10μm
Node count	min	1
	max	7
Stiffness		$\frac{stiffness_x}{stiffness_y} > 10$
Objectives		
Device resonance frequency		10 kHz
Device area		0 (minimize)

Table 3.4: Flexure resonator evolution settings according to (Kamalian 2004)

Source	Resonant frequency (Hz)	Area m^2
Raffi Roupen Kamalian	–	–
Ying Zhang	10440	1.633×10^{-7}
Circular chromosome	10002	2.484×10^{-8}

Table 3.5: Constraint case 1: Best solutions after 50 generations

Source	Resonant frequency (Hz)	Area m^2
Raffi Roupen Kamalian	10325	1.711×10^{-7}
Ying Zhang	10388	1.625×10^{-7}
Circular chromosome	10004	7.158×10^{-8}

Table 3.6: Constraint case 2: Best solutions after 50 generations

Constraint case 1

Comparison of best results yielded by circular chromosome to previous researchers can be seen in Table 3.5. With the circular chromosome approach, the algorithm was able to generate 6.96 unique pareto front solutions and 8.6 non–pareto front unique strictly better [7] solutions per single experiment than solutions provided by Ying Zhang.

Constraint case 2

Comparison of best results yielded by circular chromosome to previous researchers can be seen in Table 3.6. With the circular chromosome approach, the algorithm was able to generate 8.52 unique pareto front solutions and 10.4 non–pareto front unique solutions strictly better than both Ying Zhang and Raffi Roupen Kamalian simultaneously, per single experiment.

Constraint case 3

Comparison of best results yielded by circular chromosome to previous researchers can be seen in Table 3.7. With the circular chromosome approach, the algorithm was able to generate 3.12 unique pareto front solutions and 3.88 non–pareto front unique solutions, strictly better than both Ying Zhang and Raffi Roupen Kamalian simultaneously, per single experiment.

[7] strictly smaller area and strictly smaller frequency error

Source	Resonant frequency (Hz)	Area m^2
Raffi Roupen Kamalian	10463	1.55×10^{-7}
Ying Zhang	10344	1.463×10^{-7}
Circular chromosome	10016	5.964×10^{-8}

Table 3.7: Constraint case 3: Best solutions after 50 generations

Source	Resonant frequency (Hz)	Area m^2
Raffi Roupen Kamalian	10380	1.703×10^{-7}
Ying Zhang	10493	1.599×10^{-7}
Circular chromosome	10036	5.779×10^{-8}

Table 3.8: Constraint case 4: Best solutions after 50 generations

Constraint case 4

Comparison of best results yielded by circular chromosome to previous researchers can be seen in Table 3.8. With the circular chromosome approach, the algorithm was able to generate 2.2 unique pareto front solutions and 3.28 non–pareto front unique solutions strictly better than both Ying Zhang and Raffi Roupen Kamalian simultaneously, per single experiment.

3.6 Discussion of results

As can be seen, the circular chromosome performed better in most of the experiments requiring two times less function evaluations due to reduction of population size from 400 to 200 solutions with same generation count.

The performance gain in case of circular chromosome usage can be attributed to two properties introduced by circular chromosome approach to MEMS synthesis process:

1. redundancy

2. linkage learning

As the performance gain is attributed to these two properties of circular chromosome, it makes sense to discuss them in detail.

3.6.1 Linkage learning

The importance of the correct choice of BB and importance of choosing correct encoding for genetic algorithms was regarded as critical for GA's performance and was widely addressed by a number of researchers, whereas linkage learning properties of GAs were overlooked until beginning of last decade, according to (Thierens & Goldberg 1993).

In GA, linked building blocks (in most of the cases, the BB can be declared same thing as a gene in chromosome of single GA solution) is such a BB, presence of which, under certain circumstances (e.g. the right block sequence and properties) results in enhanced solution fitness when compared to other possible BB setups.

For example, in the case of meandering resonator modelled in SUGAR, the genes describing resonators' leg are actually linked, since changing one node in leg by rotating it by some angle with high probability changes optimal properties for all successive leg nodes.

Another evidence highlighting importance of linkage learning is that the concept of BB is related to linkage learning. The 'Building Blocks' proposed by (Zhou et al. 2002) and later adapted by a variety of other researchers, such as Raffi Roupen Kamalian (Kamalian 2004) and Ying Zhang (Zhang 2006) are nothing more that manual realisation of hand–picked linked genes for the MEMS design synthesis. For an example please refer to Figure 3.9 in which an example of handpicked linked genes provided by MEMS designer is demonstrated. As can be seen, same building blocks are reused throughout multitude of designs by multitude of researchers. Given building blocks are a way of defining linked genes by MEMS designers.

As can be demonstrated on the example of MEMS flexures (Figure 3.8), the flexure models are restricted to have alternating pattern of angles between beams using which the flexure is constructed. In an ideal world, MEMS optimisation process would be able to automatically extract such flexure as building block through detection of linked genes in it.

The, current approaches are unlikely to succeed in such a task as the appearance of flexure–like structure of noticeable length by the means of

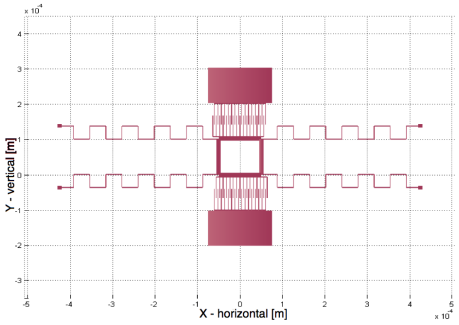


Image from (Cobb et al. 2007b, page2)

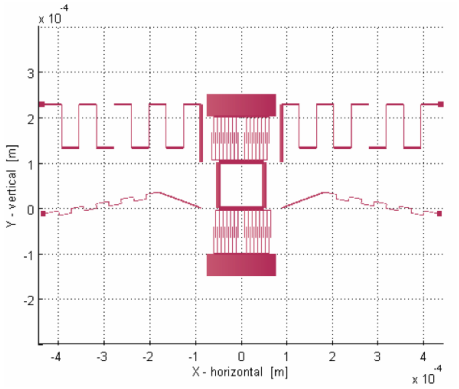


Image from (Cobb et al. 2008b, page 471)

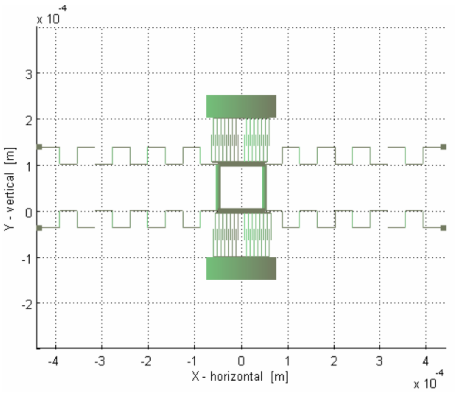


Image from (Cobb et al. 2008b, page 471)

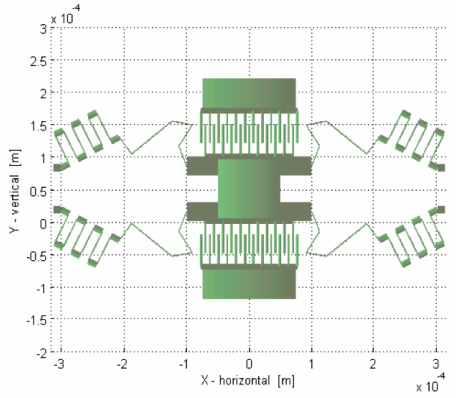


Image from (Zhang 2006, page 91)

Figure 3.9: Example of linked genes in MEMS

random evolution in the GA is unlikely, but this is a kind of problem to be addressed in future. And, as the author sees it, not only MEMS design process will benefit from advancements in research on linkage learning GAs, but also any design process will benefit that incorporates creation of models with complex design topologies.

There are two logical approaches to linked BB management – one is to create such chromosome encoding that linked properties would be obeyed and preserved and the alternative way of development of such a GA would be to automatically detect linked blocks and preserve them.

In the groundbreaking publication (Thierens & Goldberg 1993) it was shown that linkage learning is essential for GAs that attempt to solve complex problems.

Both options have their weak and strong sides. The first option (BB–

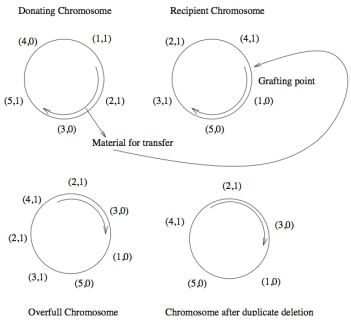


Image from (Harik 1997, page 39)
(a) GA crossover operator

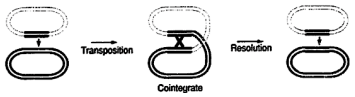


Image from (Chen 1996, page 194)
(b) Bacterial chromosome crossover process

Figure 3.10: Chromosome crossover process

preserving GA encoding developed by designer) does not require any extra computational power from GA, does not require any noticeable changes in generic GA, but what it does require is manual labour and deep understanding and analysis of optimised problem, which might be extremely time–consuming.

It can be expected for the automatic BB linkage detecting GA approach to require more computational power, since designer is making computer to do one extra thing to all other computations performed during GA run, but because the development of such a GA would require less human time, as deep analysis of the optimised problem will be no longer be required.

Since the problem of MEMS EC synthesis is multi–dimensional, the development of universal MEMS encoding seems to be an extremely complex task to address. This point of view is supported by the results obtained in Section 3.5 – the application of simple linkage–learning GA algorithm to the problem that was addressed by the whole research group yielded better results while using half of function evaluations.

In the experiment performed in Section 3.5, the linkage learning was introduced by using crossover operator that was introduced in (Harik 1997) resulting from theoretical research.

Interestingly, as can be seen in Figure 3.10 crossover operator introduced by this publication matches the chromosome crossover mechanism used in biological entities.

The success of this approach shows that developed encoding by previous

researchers is suboptimal as linkage–learning GAs that are in relatively early stages of development yielded better results than previous state–of–the–art research shows that better encodings with higher effectiveness for MEMS devices might be developed.

The sub–optimality of MEMS encoding proposed by Zhou (Zhou et al. 2002) and later used in majority of research on MEMS design synthesis is demonstrated by the fact that designs with similar responses were synthesised by other researchers such as (Lohn et al. 2008) using a completely different encoding scheme.

3.6.2 Redundancy

Redundancy is an ability of chromosome structure to contain *non–coding segments* (that are also often referred to as *introns* by the name of their biological analogs).

The non–coding segment in the context of GA stands for such GA chromosome subsequence that is not affecting properties of evaluated solution (also referred to as *phenotype*) in any way. The non-coding DNA segments are found in biological entities as well. In fact, it is estimated that 95% of human DNA are actually non–coding sequences(Fedorova & Fedorov 2005).

The intron–incorporating GA was first introduced in (Levenick 1991) and later was applied to a variety of problems showing an improvement in algorithm efficiency and results obtained (Langdon 2000, Miller & Smith 2006, Wineberg & Oppacher 1994).

The linkage–learning GA introduced by (Harik 1997) incorporates introns as well, and according to (Harik 1997) it seems that removing introns form EPE–2 chromosome does reduce overall performance of GA (Goldberg, Lobo, Deb, Harik & Wang 1998).

For the given MEMS problem, redundancy was introduced by allowing chromosome to contain more data that was actually expressed in the device design (e.g. allowing circular chromosome 16 element in length, with idea that at least two elements must be used for ‘start’ and ‘stop’ commands and allowing for beam coding data to take up to two times allowed to be

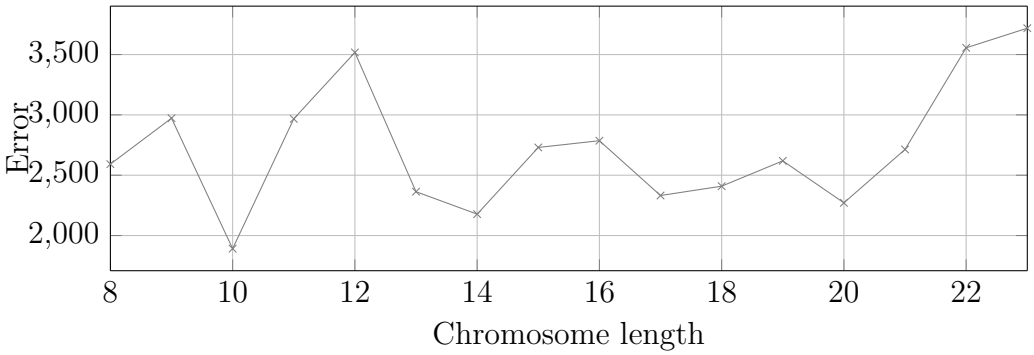
expressed in the actual device). Having chromosome length greater than maximum design size is desirable.

To illustrate desirability of introduction of non–coding genes to the MEMS design problem, a set of experiments was performed. Each group of experiments was performed with same GA configuration as experiments executed for testing of circular chromosome with flexure resonator (please refer to subsection 3.5.2 for further description).

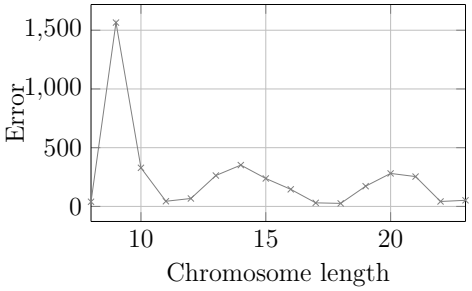
For each chromosome length, a set of ten experiments with same chromosome length was performed. Chromosome length varied from eight genes in length (six genes for encoding of phenotype plus two genes for 'start' and 'stop' commands, so taking into account that maximum allowed node count per resonator leg is six there is very low chance of introns to emerge) to 21 genes. With maximum node count equal to six (per leg), there are at least 13 introns per chromosome (or 61.9% of the chromosome length).

Only solutions with less than 20% error in resonance frequency were evaluated. This was done because the frequency objective is of uttermost importance for MEMS resonator, as the resonator with incorrect resonance frequency is unusable disregarding any achievements in other objectives. The ±20% boundary was chosen based on the fact that SUGAR simulation environment usually demonstrates 80—95% precision in MEMS device response estimation (according to (Cobb & Agogino 2006)). The average error was later calculated for each objective using the set of remaining solutions.

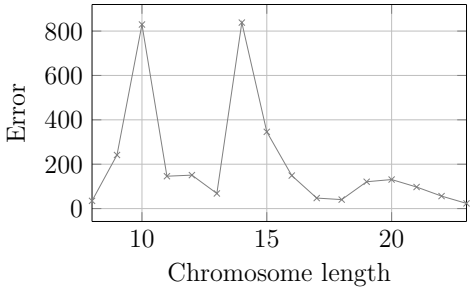
As can be seen from the figures 3.11(a), 3.11(b), 3.11(c) and 3.11(d), the increase of redundancy is indeed likely to decrease the error ratios for all of the objectives as the chromosome length increases. Of course, given results contain noticeable levels of noise, as only ten GA runs were performed for each chromosome length, but it is clearly visible, especially on the graphs 3.11(b) and 3.11(c) that as chromosome length increases the average error decreases, what can attributed to the redundancy as other research (e.g. (Chi, Hsu & Lin 2007, Yilmaz & Wu 2005, Raich & Ghaboussi 1997)) has detected similar effect of redundancy on the performance of the GA.



(a) Frequency error



(b) Stiffness by X error



(c) Stiffness by Y error

Chromosome length	Frequency error	Stiffness X error	Stiffness Y error
8	2591.9195	39.6431	34.8655
9	2972.4787	1565.1098	241.5263
10	1891.3068	328.6591	829.5640
11	2967.1109	43.8768	145.8523
12	3517.5517	67.4779	150.7930
13	2363.4413	263.0008	68.1478
14	2177.3233	351.8587	838.2706
15	2730.0140	237.9620	345.9123
16	2785.9940	145.4572	148.8646
17	2331.7949	30.3855	47.3555
18	2409.1924	25.7031	40.4451
19	2618.9051	172.1261	120.9453
20	2271.2403	280.8241	130.8893
21	2713.7015	254.0002	97.1498
22	3555.7765	41.9543	56.2688
23	3718.4291	51.9638	23.3950

(d) Tabular results

Figure 3.11: Chromosome length effect on levels of objective error

3.7 Summary

The only downside of this approach seems to be the inability to evolve solutions with extremely low error ratio, just as in the case of reaching Y stiffness objective in case of meandering resonator. It is speculated by the author that the inability to surpass the results provided by Multi–objective genetic algorithm (MOGA) with classic chromosome can be attributed to the probabilistic properties of evolution of phenotypes (in this particular case, designs of Meandering resonator), and thus, inability of population encoded by circular chromosome, to converge and perform local search for local minima.

The inability of circular chromosome to perform local search could be addressed by the introduction of a process similar to positive reinforcement loop, for example, by manual increase of probabilities for start/stop interpretation commands, that were used to produce given MEMS designs, in the solutions belonging to the Pareto front.

Alternatively, the conventional means of optimisation, such as the gradient–based method could be used to find solutions belonging to local optima. This approach is more desirable from the strategical point of view, as making GA to converge by reduction of probabilities would limit the size of search space, and that is not desirable, while optimising output designs with local optimisation technique will allow for GA to be less precise in exchange for being more explorative.

The effects of redundancy and linkage introduced by the deployment of circular chromosome need more exploration and research, as the currently introduced technique seems to improve the GA performance. The deployed data structure and crossover operator were developed to act on simple one–dimensional looped chromosome, while de–facto used chromosome has more complex structure as each gene in it contains three values – length, width and angle of the encoded node. As GAs are applied to increasingly complex tasks, it would be desirable to invest into research of the crossover operators for complex chromosomal structure.

Chapter 4

Component–based MEMS synthesis

4.1 Introduction

Due to the success of circular chromosome described in previous chapter, it was decided to exploit one of the most important properties introduced by circular chromosome – linkage–learning for Building Block (BB) identification.

In case of MEMS, the BB identification would enable automated extraction of good MEMS designs or sub–elements of good designs such as flexures to be stored in MEMS design repository for later re–usage by GA in future in MEMS design synthesis processes.

The automated extraction of BBs is desirable as it would allow a future designer to re–use good design elements without having to re–invent all design elements from scratch. The desirability of usage of BBs in MEMS is further supported by the fact that many MEMS design packages such as IntelliSense(IntelliSense Corp 2010) and SUGAR are already shipped with various common design elements such as comb drives and flexures included to their libraries.

MEMS encoding introduced by Zhou (Zhou et al. 2002) and adopted in many successive works including current one incorporates BB ideology. The

importance of BB approach was highlighted by Cobb et al. in (Cobb et al. 2008a, Cobb & Agogino 2006, Cobb et al. 2008b, Cobb et al. 2007b) where the biologically–inspired hierarchy for both MEMS and MEMS elements (BBs) was proposed.

However, the process of creation of database of BBs for GA to work with is purely manual to the moment, using human designer to introduce the BB to the GA. This is hard and troublesome process that limits amount of BBs available to the GA.

The alternative route of collecting the BBs is by implementing some sort of automated system to find the good bits of MEMS designs and add them to design library.

4.2 Description of the proposed approach

As there are two major problems that are needed to be resolved for the implementation of automated creation of BB database: BB identification and extraction and BB creation, as when a desirable sequence of nodes is actually found, it is nothing more than a set of interconnected nodes. And the set of interconnected nodes contains large number of variables that are controlled by GA most of which must be kept linked with other BB variables to preserve the actual existence of BB.

In the case of making decision on what BB variables should be exposed to the GA and what should remain bound inside of a BB thereby preserving its existence, there is no particular hint to take from human designers, as they tend to perform algorithmic compression of the BB along with deciding the exposed variables. For example, a spring might have 'a number of crenulations' (Kamalian 2004, page 98) that would regulate the length of synthesised spring. A technique of performing the collection of BB samples, their matching, extrapolation of design traits and building an algorithm to be able to perform synthesis of similar BBs sharing same traits is currently unavailable.

As the simplest approach is often the best, it was decided to take an alternative approach to the problem of the creation of BB (from the set of

nodes considered to be a BB). The alternative approach is to export a static list of variables from the given set, incapsulating any other variable inside the BB. Therefore, there is a need to define such a set of variables that could be extracted from any sequence of nodes. As will be discussed later, several possible approaches were identified and tested.

The problem of BB identification is quite complex, as it might invoke a variety of techniques used to find similarities in data. As this research has already stepped into the field of unknown by attempting to define a universal BB extraction procedure, it was decided not to introduce additional uncertainty in results and to take the simplest possible approach. Thus, it was decided to split the MEMS device models used as examples for current experiments to a number of static parts that later can be recombined as a new design of MEMS.

As the simulation software of choice for this work was SUGAR, is was natural to use already built and tested flexure resonator for the GA runs that were used to populate the BB database. The reason to prefer flexure resonator over simpler meandering resonator is that evolving flexure resonator is a more complex task as it contains comb drives and, hence, a restriction on oscillation direction. The motivation to use BBs is that they ease the creation of MEMS designs. One might also expect added constraints to add validity to the results obtained.

To accumulate enough designs for the population of BB database, 38 runs of the circular chromosome–enabled GA design synthesis were performed with same evolution settings and constraints as in circular chromosome experiments (for detailed description of settings please refer to Table 4.1).

As the simplest possible approach to BB extraction from resonator designs would be to allow for GA to reconstruct flexure resonator designs with similar properties. It was decided to extract all four legs from each resonator to be saved to the database as BBs.

Because resonator designs were evolved without symmetry constraint, it was decided to preserve information regarding which corner of the central mass given leg was attached to. Due to the expectation for legs to evolve with respect to the stiffness ratio constraint, the maximum stiffness ratio

Flexure resonator design experiment settings		
Algorithm settings		
Run count		38
Population		200
Generations		50
Algorithm		NSGA–II
Other settings		Same as in Section 3.4
Resonator design parameters		
Central mass		100x100μm plate with 4 30x50μm beams attached. Plus mass of the comb drives (11 fingers, 50μm long by 4μm thick with 3μm gap, plus a 4μm thick spine)
Node length	min	10μm
	max	100μm
Node width	min	2μm
	max	10μm
Node count	min	1
	max	6
Stiffness		$\frac{stiffness_x}{stiffness_y} > 10$
Objectives		
Lowest natural frequency		14.916 kHz (93723 $rad \cdot s^{-1}$)
X stiffness		1.90 $^N/_m$
Y stiffness		0.56 $^N/_m$

Table 4.1: Flexure resonator evolution settings used for CBR database initialisation

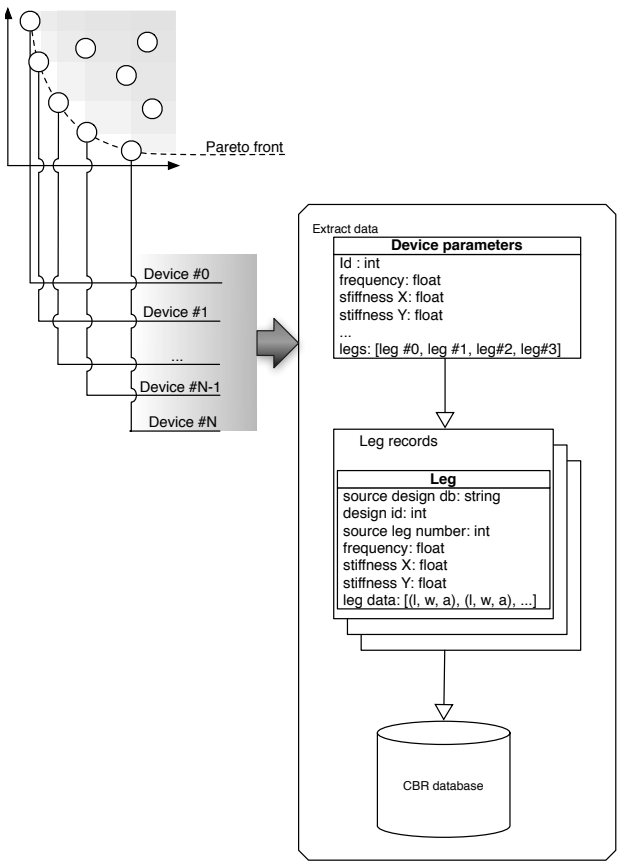


Figure 4.1: Process of generation of first version of CBR DB

for any leg was oriented with respect to defined constraints and the relative position of the resonators' central mass. Therefore, it could be beneficial to preserve the position of the leg in relation to its connection to the central mass.

The whole process of populating the BB database, as can be seen in Figure 4.1, was implemented in a straight–forward manner: as GA simulation ended, solutions of the Pareto front of final generation were acquired and their properties extracted – legs, important design parameters (such as frequency and per axis stiffness) as well as some miscellaneous information – name of the database (as all experiment runs produced separate SQLite database containing full list of designs produced by the GA run) that given solution was extracted from, the leg positional information – namely the identifier

denoting which part of the resonators' mass given leg was attached to – upper left, upper right, lower left or lower right.

The positional information was meant to be re–used during GA run as GA was allowed to chose only appropriate legs for each position.

4.2.1 Comparison methods

To compare the BB–enabled GA to the original GA, the metrics to be used needed to be defined. As the Pareto optimum is not known for a MEMS design synthesis problem, the metrics commonly used for GA comparison (such as distance of last Pareto front from Pareto optimum) are not applicable. It was decided that GAs must be compared by actual results they produce – by their Pareto fronts on the last generation. As the GAs can be compared by:

- optimisation targets
- diversity
- function evaluation count
- number of pareto solutions

A number of metrics to measure these parameters needed to be defined.

Comparison by optimisation targets is not a straight–forward task because output of multi–objective GA is not one solution, but rather a Pareto front of solutions. It was decided to use hypervolume metrics introduced in (Fleischer & Fleischer 2003) applied to the Pareto front design responses from desired values of objectives.

A hypervolume metric shows overall "size of the space covered" (as was denoted in (Zitzler & Thiele 1999, Zitzler & Thiele 1998)) from the zeroth point (point with all coordinates equal to 0). A lower hypervolume value means that Pareto front is closer to that point, and the values of objective errors are used as coordinates for points of hull whose hypervolume is measured. A lower hypervolume value means lower error and therefore a pareto front that is closer to the desired objectives.

The Diversity of solutions is a measurement of response variation between solutions belonging to the same Pareto front.

In this work, the GAs that are having high value of spacing metric (that will be defined later) are considered diverse, as the spacing metric is an average deviation of distances of solutions from average of the distance between solutions. For more information, please refer to the subsection 'spacing metric' below.

Function evaluation count is a simple metric that equals to the number of solutions per generation multiplied by the number of generations.

Number of pareto solutions is defined as the average number of solutions on the pareto front.

Spacing metric

Spacing metric (alternative depiction Δ') has been introduced by Schott in (Schott 1995). The purpose of this metric is to gauge haw evenly the points in the approximation set are distributed in the objective space. This metric is given by:

$$\Delta' = \sqrt{\frac{1}{n-1}\sum_{i+1} n(\overline{d} - d_i)^2} \tag{4.1}$$

where

$$d_i = \min_j(|f_1^i(\vec{x}) - f_1^j(\vec{x})| + |f_2^i(\vec{x}) - f_2^j(\vec{x})|) \tag{4.2}$$

where

$$i, j = 1, 2, \ldots, n \tag{4.3}$$

$\overline{d}$ is the mean of all d_i and n is the size of the known Pareto front. If this metric is used in conjunction with other metrics it may provide information about the distribution of vectors obtained. It has low computational

overhead. The metric can be generalised to more than two dimensions by extending the definition of distance d_i to cover more objectives.

It can be replaced with

$$d_i = \min_j (\sum_{d=1}^{k} |f_d^i(\vec{x}) - f_d^j(\vec{x})|) \tag{4.4}$$

Where k is number of objective vectors.

In case of multiple experiments, the average of each experiments' metrics will be used.

A lower metric value means that the solutions are evenly spaced.

4.3 Motivation

For the BB–enabled experiments, the ModeFrontier(ESTECO s.r.l 2009) visual programming environment was chosen, as it provided simple and error–proof way of defining experimental programs' workflow. This property of ModeFrontier(ESTECO s.r.l 2009) is perceived important as a variety of different approaches to the variables exported from BB were considered.

As the ModeFrontier(ESTECO s.r.l 2009) currently allows only the subset of known GAs to be used, the choice had to be made. It was opted to use NSGA–II as an industry standard in world of Genetic Algorithms (GAs) deploying settings displayed in Table 4.2. Although most parameters are set to their default values for current version of ModeFrontier(ESTECO s.r.l 2009), it was decided that explicitly listing them could benefit future researchers to be able to precisely repeat performed experiments if needed.

The only non–default parameters set in Non–dominated Sorting Genetic Algorithm II (NSGA–II) settings (Table 4.2) are the 'Number of generations' and 'Crossover probability'. In case of 'number of generations', the actual number was chosen taking into account the total amount of time devoted to an experiment and the fact that in previous experiments, the better solution occurrence significantly slowed down after 25—30 generations, thus the fifty generation limit should be enough for monitoring algorithms' performance.

ModeFrontier(ESTECO s.r.l 2009)NSGA–II settings	
Algorithm settings	
Experiment count	25
Scheduler properties[1]	
Parameters	
Number of generations	50
Crossover probability	0.9
Mutation probability for real–coded vectors	1.0[2]
Mutation probability for binary strings	1.0[2]
Advanced parameters	
Distribution index for real–coded crossover	20.0[2]
Distribution index for real–coded mutation	20.0[2]
Crossover type for binary–coded variables	Simple[2]
Random generator seed	1[2]
Category parameters	
Categorise generations	Off[2]

Table 4.2: ModeFrontier(ESTECO s.r.l 2009) GA settings

Notes

[1] 'Scheduler' is a component in ModeFrontier(ESTECO s.r.l 2009)responsible for setting of global GA options.

[2] Default value

For the 'Crossover probability' parameter, the value that was chosen, was set rather high (0.9 or 90%) to promote genetic code mixing, thus, to make the behaviour of genetic algorithm to be more explorative.

Since randomly generated designs are much likely to be self–intersecting than not, ModeFrontier(ESTECO s.r.l 2009) tools for generation of initial population were insufficient for the generation of collision–free initial population of MEMS, as randomly generated population of two hundred designs is likely to have only about a handful of non–intersecting designs. To address this problem, a helper program was implemented that generated initial population to be later loaded as ModeFrontier(ESTECO s.r.l 2009) user Design of Experiment (DOE) sequence.

To reduce measurement noise and uncertainty in GA performance and its results, the GA targets (such as resonance frequency, per axis stiffness and constraints) were kept same as targets used in GA runs that provided cases

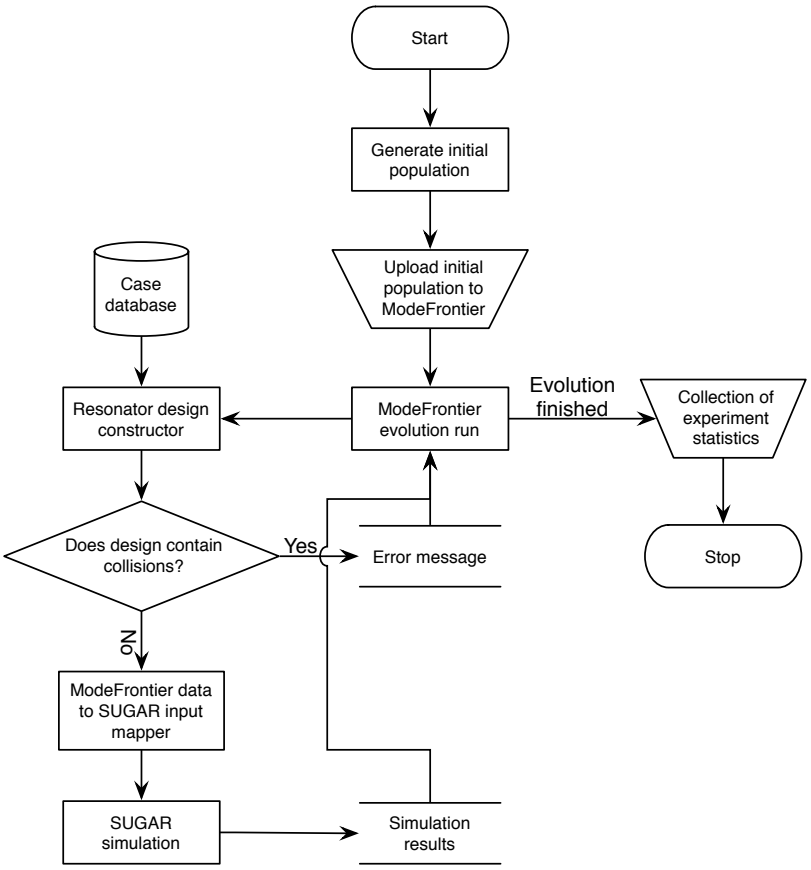


Figure 4.2: ModeFrontier(ESTECO s.r.l 2009)–based CBR–assisted evolution prototype workflow

for case Database (DB) used in given experiments.

The overall ModeFrontier(ESTECO s.r.l 2009)–based CBR process information workflow is depicted on Figure 4.2.

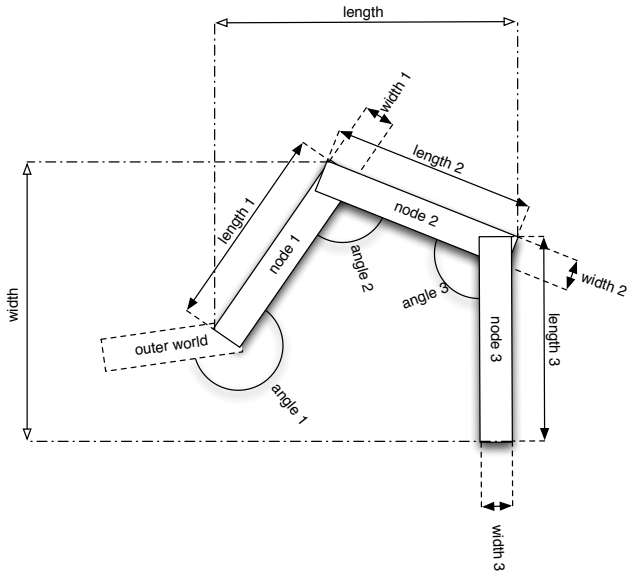
Also ModeFrontier(ESTECO s.r.l 2009) has two other properties that are important for obtaining correct perspective on the results produced by it: it does not re–generate or attempts to fix in any way invalid designs[1] and the designs that have broken constraints (such as stiffness ratio in given experiments).

As is was decided to perceive BBs as indivisible entities with a list of exported variables. The definition of particular variable to be exported as

[1] e.g. designs containing intersections

$$[\ldots, \underbrace{(length_1, width_1, angle_1)}_{beam_1}, \underbrace{(length_2, width_2, angle_2)}_{beam_2}, \underbrace{(length_3, width_3, angle_3)}_{beam_3}]$$

(a) Encoded



(b) Expressed

Figure 4.3: Example of a sequence of encoded values and expressed MEMS structure

static universal relation without any reference to the number of components in the BB, number of nodes or the relative positions of the nodes.

To understand the motivation of choosing one algorithm over another, one needs to consider the actual properties that are possessed by every set of interconnected nodes.

Due to deployed encoding, an encoding that was introduced by Zhou (Zhou et al. 2002), the sequence of nodes is represented as one–dimensional vector of triples with each triple describing one node and sequence of given triples describing the sequence of nodes where each successive node is connected to one preceding it.

For example, a sequence of beams encodes a node configuration as displayed in Figure 4.3.

As can be seen, every sequence of nodes, where each node has three properties (for list of properties assigned to an a beam please refer to equation 4.5)

has an outer angle, that is used for connecting given sequence of nodes to the outer world, and a total length and width.

$$\underbrace{(length, width, angle)}_{beam} \tag{4.5}$$

The first and simplest export from BB variable configuration is one variable, that is connection angle. Hence, the first BB–assisted GA configuration is 'changing base angle' GA modification. 'Changing' means that GA evolves not a completely new $angle_1$ (in the context of Figure 4.3), but a δ_1 that is used to produce a new $angle'_1$ using the formulae $angle'_1 = angle_1 + \delta_1$.

As the legs (that are used as simple BBs in current experiment) were evolved in the environment requiring stiffness constraint, it is probably undesirable to attempt to scale the BB without respect to the aspect ratio it has evolved. Therefore, a BB should be linearly scaled preserving relation between its total width and height. Also, as the base angle modification is desirable, it can help the GA to 'fix' a leg's vector of maximum stiffness, that should be needed as each leg has evolved in a neighbourhood of other three resonator legs. Therefore it is reasonable to consider that its direction of maximum stiffness has evolved with respect to the direction of maximum stiffness of other legs in the same resonator. As the combination of legs assembled by GA to the new resonator will most probably be different, one should allow the GA to tune the legs' base angle. Hence, the second GA configuration to check is 'Changing base angle and scaling of whole leg'.

Preserving aspect ratio while scaling a BB has ben justified, and yet another approach could be undertaken is preserving scaling of not all GA, but rather scaling of each separate node while preserving its aspect ratio. This could allow the GA to tune a particular beam of the node, if needed. Therefore, third configuration of GA is 'per–beam scaling and changing the base angle'.

As one might attempt to evolve the wholly new layout of BB while preserving the original node scales, the most impact on the stiffness of a leg (and hence a resonance frequency of a resonator) is expected to be done by the angles between nodes. Hence it might be beneficial to evolve angles to

match stiffness of a leg to its new neighbours. Hence, a 'per–beam scaling and changing all angles' GA configuration is tested.

To reduce the amount of GA variables, while preserving the aspect ratio of beams, a 'changing all angles between nodes' experiment configuration is performed.

And to check if the changing angles by evolving δ's is any better than completely replacing them, a 'replacing all angles between nodes' GA configuration ought to be checked.

4.4 Experimental setup

For each experiment a separate ModeFrontier(ESTECO s.r.l 2009)workflow was constructed and ten initial populations generated using a separate program. When using the random initial population generation, the percentage of designs having node intersections (and hence, are invalid) is so great that for randomly generated population of 300 solutions, there are only 1–3 solutions that have no intersections. This was considered inappropriate as starting from valid and diverse solution set is vital to the success of the GA.

As the actual node count in each BB is not known, but it is known from the parameters used to initialise the database (please refer to Table 4.1 for full list of parameters) that legs are 1 to 6 nodes in size, the maximum allowed number of variables was evolved, in appropriate configurations. Later, a custom software performed the actual configuration evaluation, and ignored extra variables, when the actual node count became known as leg (BB) was fetched from the database.

4.4.1 Reference NSGA–II evolution

For the actual ModeFrontier(ESTECO s.r.l 2009)workflow, please refer to the appendix for Figure A.1 (page 165).

As the ModeFrontier(ESTECO s.r.l 2009) does not provide variable length chromosome data structure, an alternative approach mimicking variable chromosome length needed to be implemented. This problem was solved by the creation of a matrix capable of holding maximum number of nodes per leg (that equals to six in this particular group experiments) and adding one additional variable that denotes actual number of nodes expressed in the device design with minimum value of 1 and maximum value of 6. As can seen on ModeFrontier(ESTECO s.r.l 2009) workflow, this functionality was implemented by the creation of 18 node–related variables for each leg (since each node had three properties – length, width and angle) and one additional variable denoting number of expressed phenotype nodes.

For the results of reference NSGA–II evolution please see Table 4.3.

4.4.2 Changing base angle

For the actual ModeFrontier(ESTECO s.r.l 2009)workflow, please refer to the appendix for Figure A.2 (page 166).

For the results of changing base angle experiment please see Table 4.4.

4.4.3 Changing base angle and scaling of whole leg

For the actual ModeFrontier(ESTECO s.r.l 2009)workflow, please refer to the appendix for Figure A.3 (page 167).

For the results of changing base angle and scaling of whole leg experiment please see Table 4.5.

4.4.4 Per–beam scaling and changing the base angle

For the actual ModeFrontier(ESTECO s.r.l 2009)workflow, please refer to the appendix for Figure A.4 (page 168).

Run results	
Population stats	
Errorous design share	12%
Pareto solution share	33%
Dominated solution share	55%
Number of pareto solutions	24.71
Function evaluation count	2500
Δ' (Spacing) metric	20509.1900
S–metrics	25754402011.5224
Properties of last Pareto front	
Frequency objective	
Maximum	89404.8277
75th percentile	69516.7967
Median	21327.0495
25th percentile	1891.2296
Minimum	7.5869
Standard deviation	31503.5383
Mean	32793.1934
Stiffness by X axis objective	
Maximum	358.9156
75th percentile	7.5652
Median	0.8130
25th percentile	0.3579
Minimum	0.0001
Standard deviation	48.1937
Mean	12.7553
Stiffness by Y axis objective	
Maximum	5982.6567
75th percentile	126.7689
Median	31.4900
25th percentile	19.0250
Minimum	6.1656
Standard deviation	829.3495
Mean	259.1829

Table 4.3: Results for reference NSGA–II evolution

Run results	
Population stats	
Errorous design share	42%
Pareto solution share	23%
Dominated solution share	35%
Number of pareto solutions	13.43
Function evaluation count	2500
Δ' (Spacing) metric	48900.5307
S–metrics	221913492613.5949
Properties of last Pareto front	
Frequency objective	
Maximum	292324.2877
75th percentile	64372.6903
Median	34804.1916
25th percentile	19356.2376
Minimum	63.7388
Standard deviation	37557.2804
Mean	42085.2297
Stiffness by X axis objective	
Maximum	103.9512
75th percentile	1.2359
Median	0.7538
25th percentile	0.3100
Minimum	0.0042
Standard deviation	20.8013
Mean	5.4242
Stiffness by Y axis objective	
Maximum	65893.0955
75th percentile	26.6954
Median	15.8811
25th percentile	10.7588
Minimum	2.7049
Standard deviation	6790.5703
Mean	924.4585

Table 4.4: Results for changing base angle experiment

Run results	
Population stats	
Errorous design share	35%
Pareto solution share	23%
Dominated solution share	42%
Number of pareto solutions	16.14
Function evaluation count	2500
Δ' (Spacing) metric	29962.6939
S–metrics	121742909109.1729
Properties of last Pareto front	
Frequency objective	
Maximum	472255.7898
75th percentile	53693.5440
Median	25474.2262
25th percentile	3375.1942
Minimum	4.6142
Standard deviation	69515.0623
Mean	40377.9230
Stiffness by X axis objective	
Maximum	107.7024
75th percentile	1.1292
Median	0.3939
25th percentile	0.1484
Minimum	0.0002
Standard deviation	16.2722
Mean	3.8545
Stiffness by Y axis objective	
Maximum	77345.6031
75th percentile	28.4096
Median	18.1246
25th percentile	12.7518
Minimum	3.9310
Standard deviation	9090.1152
Mean	1541.6474

Table 4.5: Results for changing base angle and scaling of whole leg experiment

For the results of per–beam scaling and changing the base angle experiment please see Table 4.6.

4.4.5 Per–beam scaling and changing all angles

For the actual ModeFrontier(ESTECO s.r.l 2009)workflow, please refer to the appendix for Figure A.5 (page 169).

For the results of per–beam scaling and changing the all angles experiment please see Table 4.7.

4.4.6 Changing all angles between nodes

For the actual ModeFrontier(ESTECO s.r.l 2009)workflow, please refer to the appendix for Figure A.6 (page 170).

For the results of changing all angles between nodes experiment please see Table 4.8.

4.4.7 Replacing all angles between nodes

For the actual ModeFrontier(ESTECO s.r.l 2009)workflow, please refer to the appendix for Figure A.7 (page 171).

For the results of replacing all angles between nodes please see Table 4.9.

4.4.8 Collation

The major metrics are collated in Table 4.10 for convenience of discussion. As evaluation count is equal to 2500 in all experiments, it is omitted from table.

Run results	
Population stats	
Errorous design share	32%
Pareto solution share	33%
Dominated solution share	34%
Number of pareto solutions	21.50
Function evaluation count	2500
Δ' (Spacing) metric	52451.1533
S–metrics	167366859420.9907
Properties of last Pareto front	
Frequency objective	
Maximum	333087.7145
75th percentile	68467.6854
Median	37509.5645
25th percentile	16259.1300
Minimum	70.4444
Standard deviation	46206.7352
Mean	47087.2797
Stiffness by X axis objective	
Maximum	472.5060
75th percentile	1.0875
Median	0.5587
25th percentile	0.1962
Minimum	0.0001
Standard deviation	56.3200
Mean	8.6054
Stiffness by Y axis objective	
Maximum	10085.2629
75th percentile	26.4832
Median	18.9527
25th percentile	13.1819
Minimum	0.5513
Standard deviation	1010.8805
Mean	195.7774

Table 4.6: Results for per–beam scaling and changing the base angle experiment

Run results	
Population stats	
Errorous design share	38%
Pareto solution share	30%
Dominated solution share	32%
Number of pareto solutions	19.90
Function evaluation count	2500
Δ' (Spacing) metric	25973.1162
S–metrics	107114474449.4576
Properties of last Pareto front	
Frequency objective	
Maximum	279767.4906
75th percentile	49049.3383
Median	20678.1663
25th percentile	1941.0241
Minimum	0.4572
Standard deviation	32504.2043
Mean	30138.6653
Stiffness by X axis objective	
Maximum	296.1763
75th percentile	5.6520
Median	0.5964
25th percentile	0.1562
Minimum	0.0002
Standard deviation	39.3460
Mean	11.4884
Stiffness by Y axis objective	
Maximum	40590.4449
75th percentile	106.8100
Median	30.8180
25th percentile	22.0883
Minimum	5.0460
Standard deviation	3646.9945
Mean	595.5632

Table 4.7: Results for per–beam scaling and changing the all angles experiment

Run results	
Population stats	
Errorous design share	32%
Pareto solution share	33%
Dominated solution share	35%
Number of pareto solutions	26.71
Function evaluation count	2500
Δ' (Spacing) metric	19449.3947
S–metrics	4631204309.7647
Properties of last Pareto front	
Frequency objective	
Maximum	158669.7808
75th percentile	48637.4469
Median	24097.9087
25th percentile	3546.2850
Minimum	5.7304
Standard deviation	32488.0002
Mean	30767.7679
Stiffness by X axis objective	
Maximum	53.6428
75th percentile	1.3273
Median	0.8002
25th percentile	0.2535
Minimum	0.0002
Standard deviation	7.0756
Mean	2.8132
Stiffness by Y axis objective	
Maximum	8389.2718
75th percentile	80.2388
Median	21.3252
25th percentile	15.8814
Minimum	8.2203
Standard deviation	619.6769
Mean	118.4929

Table 4.8: Results for changing all angles between nodes experiment

Run results	
Population stats	
Errorous solution share	21%
Pareto solution share	37%
Dominated solution share	43%
Number of pareto solutions	23.71
Function evaluation count	2500
Δ' (Spacing) metric	30366.9812
S–metrics	624565898.0879
Properties of last Pareto front	
Frequency objective	
Maximum	645635.5937
75th percentile	50122.2591
Median	35252.8632
25th percentile	13180.6523
Minimum	12.4803
Standard deviation	104341.8431
Mean	53671.2827
Stiffness by X axis objective	
Maximum	23.1937
75th percentile	1.2207
Median	0.6638
25th percentile	0.3081
Minimum	0.0007
Standard deviation	3.1016
Mean	1.3454
Stiffness by Y axis objective	
Maximum	1262.6997
75th percentile	22.3718
Median	15.9082
25th percentile	11.4927
Minimum	3.9166
Standard deviation	119.5638
Mean	41.6398

Table 4.9: Results for replacing all angles between nodes experiment

Experiment name	Number of pareto solutions	Δ' (Spacing) metric	S–metrics
Reference NSGA–II evolution	24.71	20509.1900	25754402011.5224
Changing base angle	13.43	48900.5307	221913492613.5949
Changing base angle and scaling of whole leg	16.14	29962.6939	121742909109.1729
Per–beam scaling and changing the base angle	21.50	52451.1533	167366859420.9907
Per–beam scaling and changing all angles	19.90	25973.1162	107114474449.4576
Changing all angles between nodes	26.71	19449.3947	4631204309.7647
Replacing all angles between nodes	23.71	30366.9812	624565898.0879

Table 4.10: Collated table of metrics

4.5 Discussion of results

Based on to the values of chosen metrics, the two BB approaches have performed better than reference NSGA–II runs. The leaders are 'changing all angles between nodes' and 'replacing all angles between nodes' GA configurations.

In the case of 'changing all angles between nodes', the average number of solutions belonging to Pareto front is 8% greater than number of solutions belonging to the Pareto front of the reference run, the Δ' metric is 5.4% lower than Δ' metric of reference population and the S–metric is 5.56 times smaller than reference population, meaning that Pareto front of 'changing all angles between nodes' was five times as close to finding ideal solution – a solution with objective error of zero.

In the case of 'replacing all angles between nodes' GA configuration, the average number of pareto solutions was smaller by 1.04% than reference GA, the Δ' metric has shown 1.48–times increase compared with the reference and the S–metric is 41.24 times lower when compared with the reference GA.

In both cases, a noticeable decrease in the hypervolume metric is visible, meaning that these approaches generate better Pareto fronts than standard

NSGA–II GA.

Therefore, one might assume that it is undesirable to implement scaling functionality to the GA that uses automatically generated BBs, as all results yielded by algorithms that incorporate scaling are worse. This is probably because scaling result has unforeseen effects on the designs. Such as, the mutation that performs scaling changes stiffness constraint because the beam geometric parameters were changed, but the angles between beams were left unchanged.

Also, according to metric 'changing base angle', the GA flavour is undesirable. This is likely because as each leg in the database has evolved with respect to other three legs in design, changing of just one base angle is insufficient to make leg to 'adapt' to a new neighbouring leg.

The improvement made by 'changing/replacing all angles between nodes' demonstrated the need for each leg to 'tune' to its surroundings, in the other words, to tune for a different combination of neighbouring legs (as this is the only thing that changes because central mass is kept static through out all experiments). And, as the 'replacing all angles between nodes' experiments' S–metric is 7.41 times lower than the 'changing all angles between nodes', it can be assumed that preserving an old BB information is not always as desirable as one might expect.

The results of current experiments display that automated BB generation is a not an easy task to address and that this research direction would greatly benefit from additional attention by researchers. The results show that in the case of automated BB generation, the amount of information preserved in the BB needs to be considered along with of the amount of information that GA is allowed to override.

Chapter 5

Process–induced error tolerant layout synthesis

5.1 Introduction

The emergence and fast growth of MEMS technology can be contributed to the semiconductor industry, as MEMS production is commonly performed by fabrication techniques borrowed from that industry.

While production imperfection and outcome uncertainty is present in all known production techniques, it is an inherent concern in the case of MEMS, as the production errors directly influence the performance of built MEMS, making notable share of production to be rejected decreasing outcome yield. With the MEMS production precision in order of the tenths of microns (μm) and the minimum feature size of 2 μm, it is widely accepted that the relative production uncertainty can reach 5% or even 10% as reported by (Hong, Lee & Kim 2000).

In this work, the automated creation of MEMS designs that are tolerant towards production errors, or as shorthand 'error–tolerant MEMS' will be researched as it is receiving only a little attention in spite of likely impact error–tolerant MEMS designs would have on the yield of MEMS production due to the decrease in the share of defects.

The multi–domain nature of MEMS introduces certain problems for achiev-

ing error tolerance for MEMS designs, as it is expected for the designs to be stable in their responses in all of its properties each of which can belong to a separate domain, hence possibly requiring a special simulation environment.

5.2 Description of the proposed method

All previous research has focused on the creation of designs using analytical models that are tolerant to the errors induced by manufacture process in one of its responses. But as MEMS are complex multi–domain machines, the error tolerance in one of the systems' responses is not enough, as for example in the case of MEMS resonator, the frequency objective is indeed important, as the resonator whose resonance frequency does not match the specification is useless. But the stiffness objectives have to be considered, as the resonators' movement is commonly sensed using electrostatic effect on the comb drives that are attached to resonators' body and incorrect stiffness in one of the axes is likely to cause a foul in given comb drives and therefore premature breakdown of all of the MEMS–enabled system.

The most noticeable results in error–tolerant MEMS design synthesis were achieved by Fan et al. in (Fan et al. 2007, Fan et al. 2009). It was chosen to project the results of single–objective robust optimisation approach used in (Fan et al. 2007) to the problem of multiobjective optimisation of numerical MEMS design creating a new methodology for the synthesis of multiobjective error–tolerant arbitrary MEMS designs without the requirement of the presence of analytical MEMS model.

In this work, it is assumed that the errors are introduced by MEMS etching fabrication process. Because etching is a chemical process of removal of extra material from the surface of a wafer, the small imperfections in the environmental conditions can cause either too much or too little of material to be removed from the wafer therefore causing ether overetch or underetch as illustrated in Figure 5.1.

The Monte–Carlo simulation was performed to mimic etching errors, as it is considered as a natural instrument for simulation of such dimensional variations. The simulation was performed by the creation of a certain num-

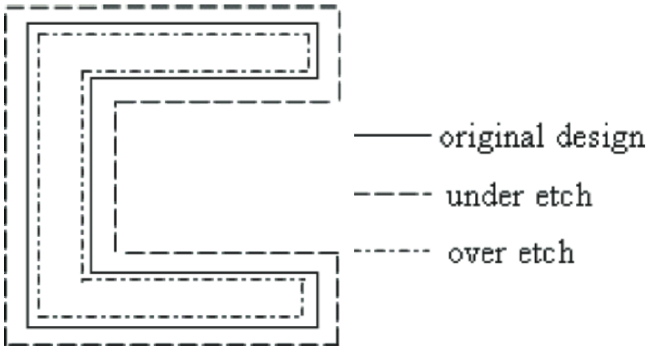


Image from (Fan et al. 2007, page 522)

Figure 5.1: Under– and over–etch of a MEMS structure

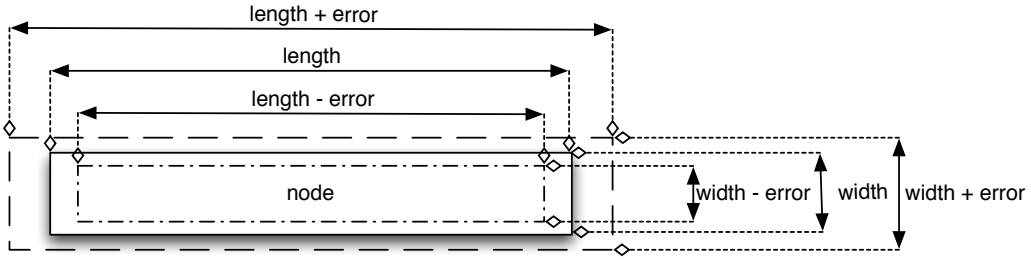


Figure 5.2: Etching error representation in used encoding

ber of designs $\vec{x}^*$ from an ideal design $\vec{x}$ by introducing random number of perturbations of appropriate scale into random number of variables of $\vec{x}$.

The MEMS design encoding developed by Zhou et al. in (Zhou et al. 2002) was deployed in the given experiment, as it naturally allows the simulation of under– and over–etch through changing of the widths and heights the SUGAR nodes.

Because every node is represented as triple $(length, width, angle)$ and given random variable $\delta_{err} \in [-\max(\text{overetch error}), \max(\text{underetch error})]$, production errors can be modelled by mapping original beam description to $(length + \delta_{err}, width + \delta_{err}, angle)$ with expected result that is illustrated in Figure 5.2.

The objective functions were designed by the inspiration taken from equations that emerged from robust design theory, equations 2.3—2.5 as defined

in (Fan et al. 2007). The objective function[1] deployed in current work is

$$Objective(r, \vec{d}) \equiv T(r, \vec{d}) + E(\vec{d}) \tag{5.1}$$

$$T(r, \vec{d}) \equiv |r - Response(\vec{d})| \tag{5.2}$$

$$E(\vec{d}) \equiv \langle i \in N : Response(Err_{prod}(\vec{d})) - Response(\vec{d}) \rangle \tag{5.3}$$

$$Err_{prod}(\{d_1, d_2, \ldots, d_K\}) \equiv (d_1 + \xi, d_2 + \xi, \ldots, d_K + \xi) \tag{5.4}$$

Where r is desired response value, $\vec{d}$ is evaluated design, N is number of design simulations with applied production error and ξ is a random variable with uniform distribution whose boundaries match production errors for chosen MEMS production process (this is considered to be etching in this work).

It can be seen, both equations 5.2 and 5.3 are sharing response measurement units (that are undefined for general function, as units are objective–dependent). This allows them to be summed in equation 5.1.

Because this work is exploring the viability of design synthesis of MEMS that are tolerant to the production–induced errors in multitude of responses, as opposed to only one response that was previously researched in literature, the list of responses to be made error–tolerant must be defined.

It was decided to do proof of concept experiment with the simplest MEMS topology possible, hence Meandering resonator was chosen. To be consistent with previous research, it was opted to use same objectives as in previous experiments in both this work (Table 3.1) and previous MEMS research (Zhou et al. (Zhou et al. 2002)).

For comparison, a control group using same experiment configuration was run without robustness objective component for GA objectives, efficiently making control group to evolve MEMS resonators matching certain responses.

[1] It is assumed that GA is minimising objective functions.

5.3 Motivation

Because calculation of gradient $\nabla_\delta f(\vec{x}, 0)$ is possible only for analytical model of MEMS, its application to the real–world problems is limited, hence limiting the area of application of robust design methodology for MEMS designs.

Also, the multi–domain nature of MEMS requires for designs to be robust towards multitude of objectives rather than one single objective, as demonstrated by Fan et al. in (Fan et al. 2007, Fan et al. 2009).

To address these two limitations, the robust optimisation approach, as described by the set of equations 2.3—2.5, is projected to the domain of numerical MEMS models resulting in a GA with objectives inspired by robust design approach and extended to the multiobjective MEMS synthesis problem.

5.4 Experimental setup

To research the possibility of multi–tolerant MEMS design synthesis inspired by robust optimisation approach, the design synthesis of multi–tolerant meandering resonator was performed. For gaining statistical feasibility in the light of stochastic nature of GA, the given experiment was repeated ten times as also the control experiment for the evolution of non–tolerant meandering resonator. The GA settings for both experiments can be found in Table 5.2 and Table 5.3.

As it can be seen, the global settings for current set of experiments (Table 5.1) are heavily based on settings used in chapter 3 (Table 3.1), the settings that were previously used by Zhou in (Zhou et al. 2002).

After the experiments were completed, the Pareto fronts of the last population of solutions (where each 'solution' is one possible MEMS design) were extracted for each experiment run and the evaluation of every design was performed for both the 'ideal design' (design without any errors introduced) plus two hundred evaluations of the errorous designs generated from 'ideal design' by introduction of random errors in $(0\mu m, 1\mu m)$ bounds.

To illustrate the result, a number of histograms were generated with

Error–tolerant meandering resonator design synthesis		
Algorithm settings		
Algorithm		NSGA–II
Population		200[1]
Generation count		200[1]
Crossover probability		0.9
Mutation probability		0.1
Selection operator		Roulette wheel
Resonator design parameters		
Central mass		Four 100x20x2 μm beams laid out in a square.
Resonator constraints		Unconstrained asymmetric
Node width	min	2μm
	max	20μm
Node length	min	10μm
	max	400μm
Node count		4
Target values		
Lowest natural frequency		14.916 kHz (93723 $rad \cdot s^{-1}$)
X stiffness		1.90 $^N/_m$
Y stiffness		0.56 $^N/_m$

Notes

[1] The population size and number of generations were set to match ones used in (Fan et al. 2007)

Table 5.1: Global settings used for all experiments on error tolerance

Error–tolerant meandering resonator design synthesis[1]		
Resonator design parameters		
Simulated layouts with errors per design		10
Error boundaries	min	0 μm
	max	1 μm
Objectives		
Frequency Objective		$T_1(93723, \vec{d}) + E_1(\vec{d})$
X stiffness objective		$T_2(1.90, \vec{d}) + E_2(\vec{d})$
Y stiffness objective		$T_3(0.56, \vec{d}) + E_3(\vec{d})$

Notes
[1] These settings are extension to the GA settings defined in Table 5.1

Table 5.2: Settings for design synthesis of multi–tolerant meandering resonator

Non error–tolerant meandering resonator design synthesis[1]	
Objectives	
Frequency Objective	$T_1(93723, \vec{d})$
X stiffness objective	$T_2(1.90, \vec{d})$
Y stiffness objective	$T_3(0.56, \vec{d})$

Notes
[1] These settings are extension to the GA settings defined in Table 5.1

Table 5.3: Settings for design synthesis of non–tolerant meandering resonator

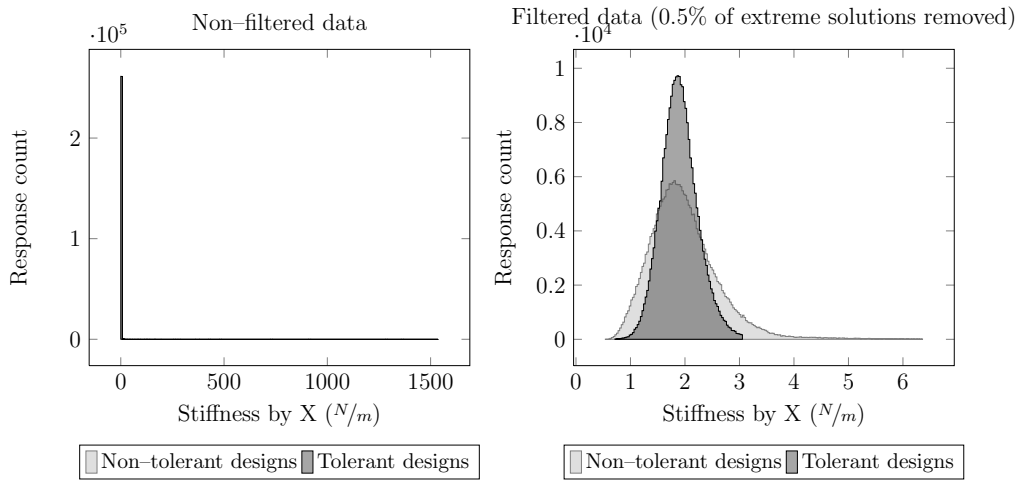


Figure 5.3: Degeneration of histogram due to extreme values

99.5% of the design responses mapped to each of the histogram. Removal of 0.5% of farthest from the target response objective was performed because mapping of single farthest responses moved histograms' boundaries so much that the part of the maximum response density degenerated to one column in some cases (as can be seen in Figure 5.3).

As can be seen in plots of the cumulative population responses (Figure 5.4), the proposed approach to multiobjective tolerance has yielded some interesting results, as the error–enabled design response estimates for the stiffness by X axis (Figure 5.4(d)) and by Y axis (Figure 5.4(f)) resemble a normal distribution with a normal probability density function (as in equation 5.5) (just as stated in (Fan et al. 2007)), while the distribution of resonance frequency for designs with introduced production errors (Figure 5.4(b)) display no such visible resemblance.

$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(x-\mu)^2}{2\sigma^2}} \tag{5.5}$$

This can be explained by the fact that each 'ideal' design, with response value on the left column histograms (Figure 5.4(a), Figure 5.4(c) and Figure 5.4(e)), is actually expected to have its own distribution of error–introduced modifications that are distributed according to normal probability

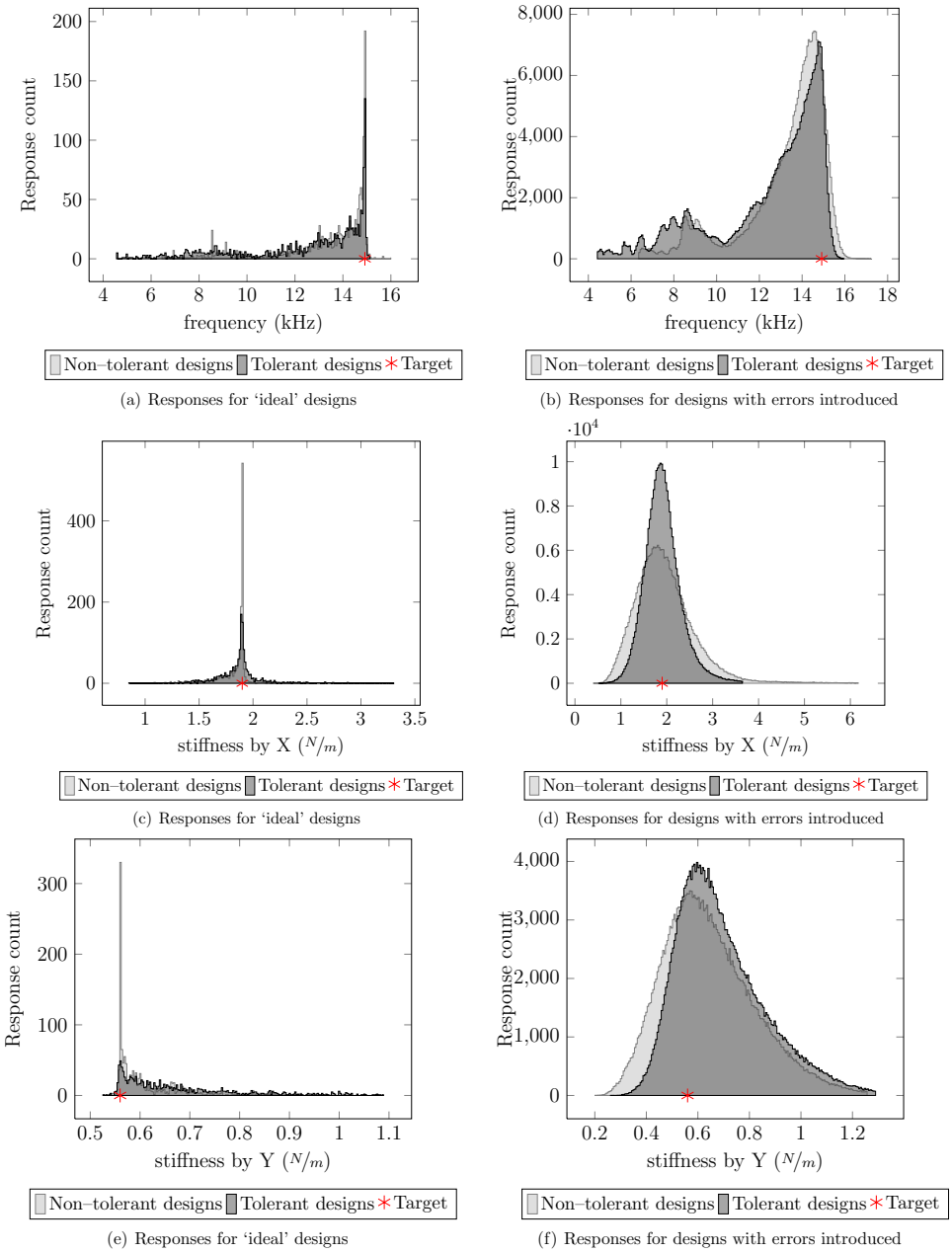


Figure 5.4: GA objectives inspired by robust optimisation: full population results

distribution (equation 5.5). And multitude of 'ideal' designs scattered by the large area of responses (as in Figure 5.4(b)) would cause an interference between distributions of the responses of error–introduced designs. This point of view is further supported by the fact that (as can be seen in Figure 5.4(f)) the distribution of the error–enabled responses for the stiffness by Y is extended to the $y \mapsto \infty$ side with the 'ideal' design distribution having similar trail of designs towards the same side.

The hypothesis of each 'ideal' design $\vec{d}$ with response value $Response(\vec{d})$ having responses for the error-enabled versions $\vec{d}^*$ distributed according to normal distribution (equation 5.5) with the $\mu \approx Response(\vec{d})$ is further supported by the fact that in cases of Figure 5.4(a) and Figure 5.4(e) it looks like the error–introduced non–tolerant designs seem to match the objective with higher probability. But if the given hypothesis is correct, this effect can be attributed to the definition of the tolerant design GA objective (depicted in equation 5.1) as the minimised composite objective is composed of two sub–objectives: $T(r, \vec{d})$ (equation 5.2), that drives GA towards target response value, therefore shifting the μ of the probability distribution of the error-enabled designs and second sub–objective $E(\vec{d})$ (equation 5.3) that drives GA towards designs having smaller response variance (that is represented by σ^2 in equation 5.5) when subject to process–introduced errors.

One might expect that the proposed approach works well when it is easy to minimise $T(r, \vec{d})$ sub–objective, as then GA will concentrate on minimisation of $E(\vec{d}, X)$ sub–objective. Therefore, the resulting design would have high probability of $T(r, \vec{d}) \approx 0$ making $\mu \approx r$ and, as r has been reached, the major cause for GA to prefer one solution over another is for that solution to have smaller $E(\vec{d})$, therefore minimising σ^2 for current μ^2.

But when $T(r, \vec{d})$ sub–objective is hard to minimise, the major cause for GA to prefer one solution over another would be the minimisation of $E(\vec{d})$ sub–objective that would cause the designs to have smaller variance (σ^2) when subject to errors. But mean (μ) is far from target response value,

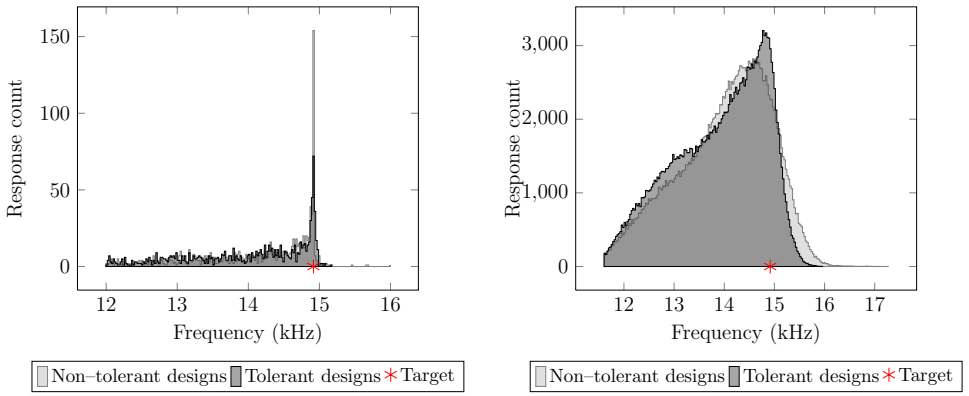
[2]It seems that 'stiffness by X' is easy to reach objective as the distribution of responses for 'ideal' designs (Figure 5.4(c)) for the tolerant and non–tolerant designs is similar, while tolerant designs have much smaller distribution in case of error–introduced designs (Figure 5.4(d)).

causing the design to be further from target with smaller variance. In this case, the definition of minimisation objective (equation 5.2) would reduce the pressure for GA to reach desired response target in favour of having more error–tolerant solutions (a behaviour exhibited in the case of 'frequency' and 'stiffness by Y' objectives).

In case the given hypothesis is correct, it would be expected for error–tolerant designs with 'ideal' response being close to the target response value to have smaller variance of responses for error–introduced designs. This can be tested by plotting the histograms for designs with 'ideal' responses within certain bounds of target response. The bounds of 20%, 4% and 1% were chosen to check this hypothesis.

As clearly illustrated by the sequence of figures for the objectives that previously exhibited no clear natural distribution of error–introduced design responses (that is 'frequency' (Figures 5.5, 5.6 and 5.7) and limited 'stiffness by Y' (Figures 5.11, 5.12 and 5.13) while 'stiffness by X' (Figures 5.8, 5.9 and 5.10) exhibit no such behaviour), the more evenly the 'ideal' designs are distributed around target, the more natural distribution–like error–introduced design response histogram appears, which can be accounted for the $\mu \approx Response(\vec{d})$ hypothesis. Also, the error–introduced response histograms exhibit smaller variance for the error–tolerant designs when compared to non–tolerant designs.

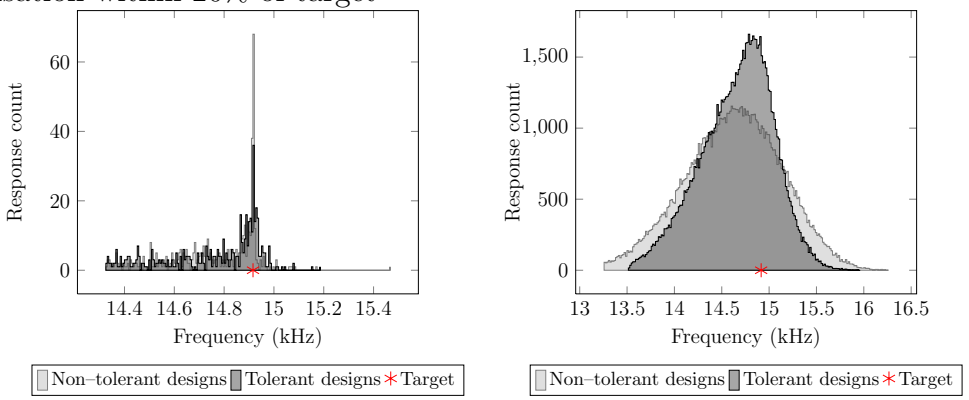
Based on the results, it was decided that the practice of merging variance minimisation sub–objective $T(r, \vec{d})$ and the objective approach sub–objective $E(\vec{d})$ should not be expected to perform well in multi–dimensional complex problems, as it allows for GA to trade objective approach for variance minimisation, therefore allowing creation of designs that are far from objective target but have small variance.



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

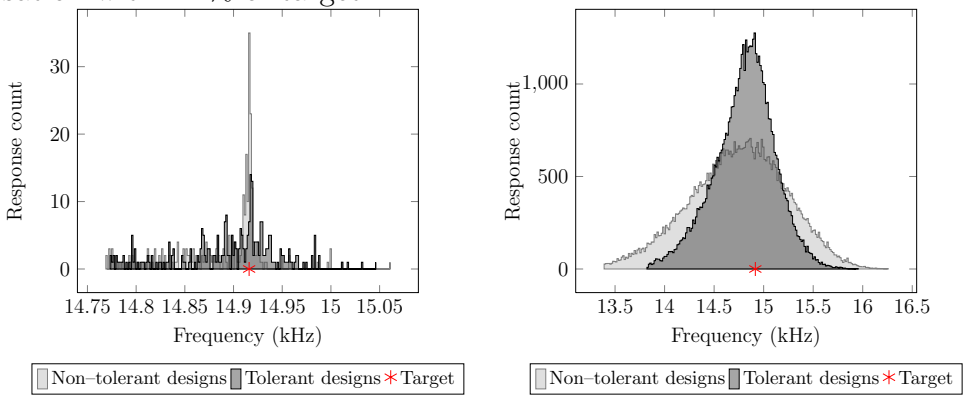
Figure 5.5: Designs for frequency of GA objectives inspired by robust optimisation within 20% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

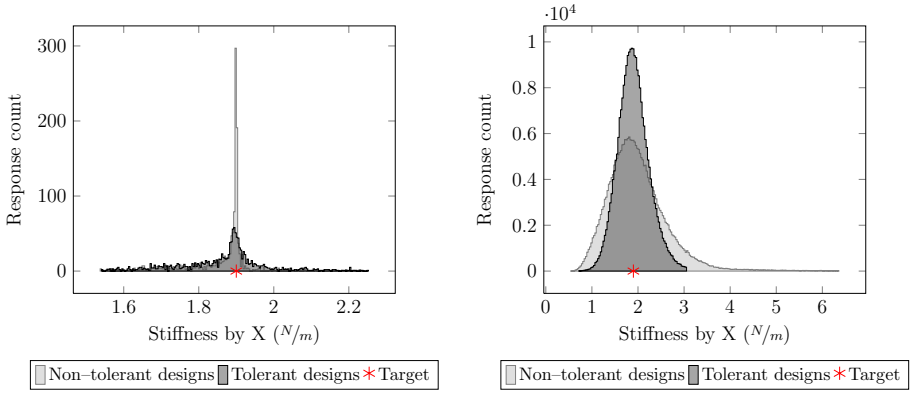
Figure 5.6: Designs for frequency of GA objectives inspired by robust optimisation within 4% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

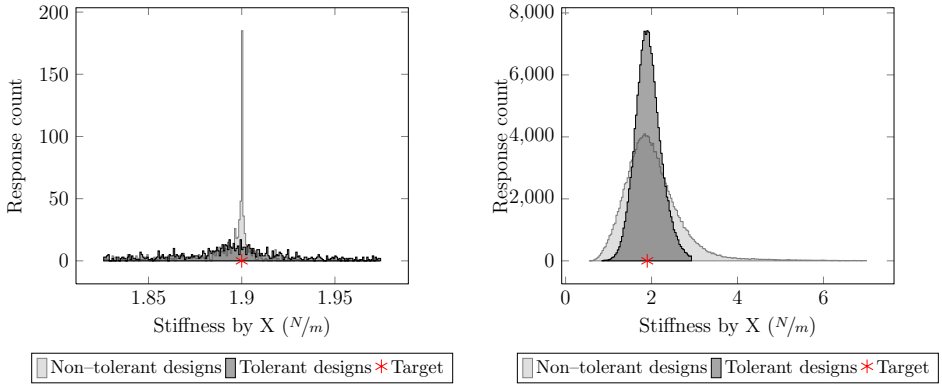
Figure 5.7: Designs for frequency of GA objectives inspired by robust optimisation within 1% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

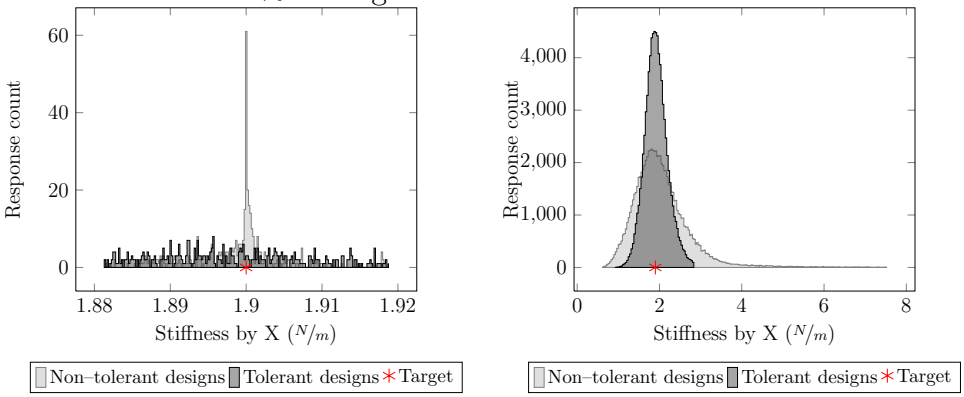
Figure 5.8: Designs for stiffness by X of GA objectives inspired by robust optimisation within 20% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

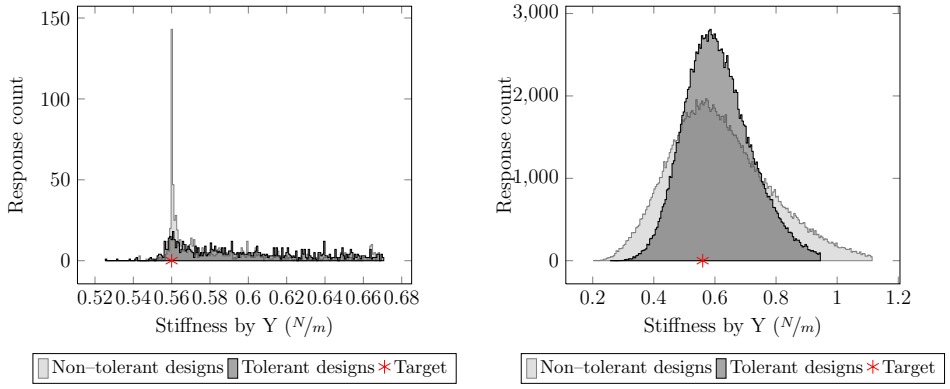
Figure 5.9: Designs for stiffness by X of GA objectives inspired by robust optimisation within 4% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

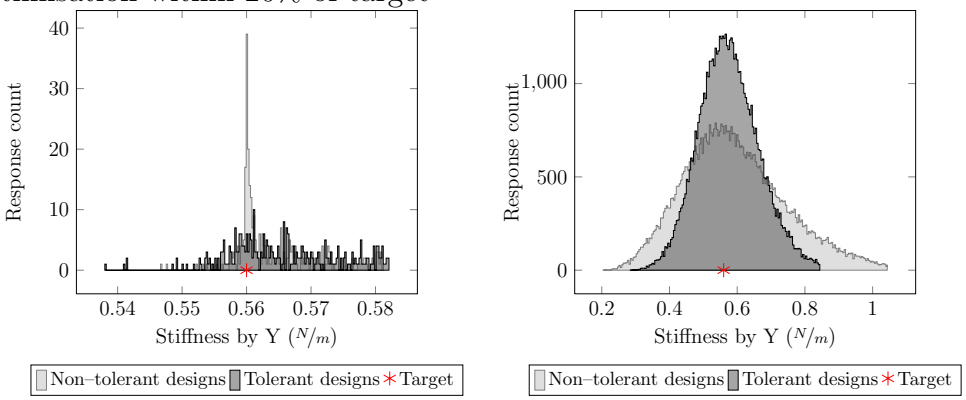
Figure 5.10: Designs for stiffness by X of GA objectives inspired by robust optimisation within 1% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

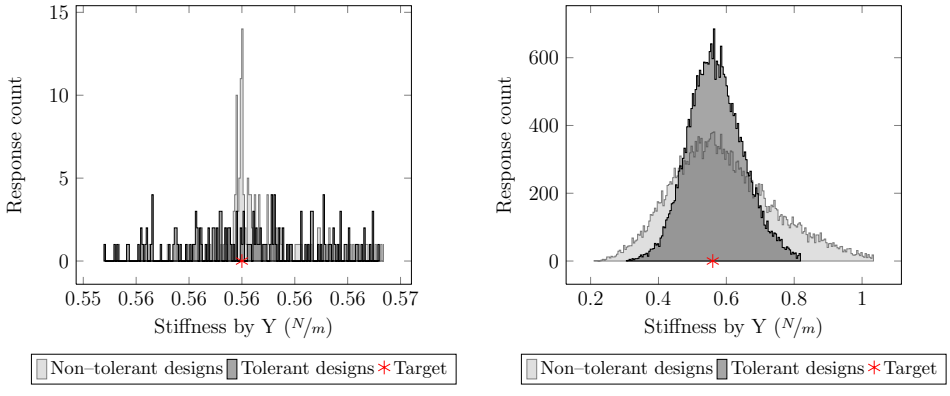
Figure 5.11: Designs for stiffness by Y of GA objectives inspired by robust optimisation within 20% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

Figure 5.12: Designs for stiffness by Y of GA objectives inspired by robust optimisation within 4% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

Figure 5.13: Designs for stiffness by Y of GA objectives inspired by robust optimisation within 1% of target

5.5 Modifications to initial approach

Simple application of GA objective inspired by robust optimisation has shown its inability to evolve multiobjective error–tolerant MEMS designs as it allows the GA to exchange the approach towards desired response with the robustness of design.

To address this issue two alternative modifications to the original approach of evolution of error–tolerant MEMS designs were proposed.

1. Making error tolerance metric target response–dependent

   It was identified that the initial robust optimisation–inspired metric is ineffective in the case of evolving multiobjective error tolerant designs because of GA exchanging sub–objective of approaching towards target for sub–objective of evolving error tolerant design.

   If only designs that are within certain neighbourhood of target response are allowed to evolve error tolerance, this should allow the GA to approach target response at first (hence achieving reasonable μ for error design response distribution) and later, when design response is within reasonable proximity of target, to concentrate on evolving as error tolerant design.

2. Separating error tolerance metric to a separate dimension

   This approach should allow the GA to perform evolution of error tolerant designs within separate objective space dimension, hence not allowing compromises in dimensions meant for achieving target response value.

5.5.1 Response–dependent error tolerance

It was identified that the initial robust optimisation–inspired metric is ineffective in the case of evolving multiobjective error tolerant designs because of GA exchanging sub–objective of approaching towards target for sub–objective of evolving error tolerant design.

If only designs that are within certain neighbourhood of target response are allowed to evolve error tolerance, this should allow the GA to approach target response at first (hence achieving reasonable μ for error design response distribution) and later, when design response is within reasonable proximity of target, to concentrate on evolving an error tolerant design.

To achieve this target, the original error tolerance objective (equation 5.3) was modified as denoted in equation 5.6 (where r is the desired design's response value).

$$E_{obj}(\vec{d}, r) \equiv \langle i \in N : Response(Err_{prod}(\vec{d})) - r \rangle \tag{5.6}$$

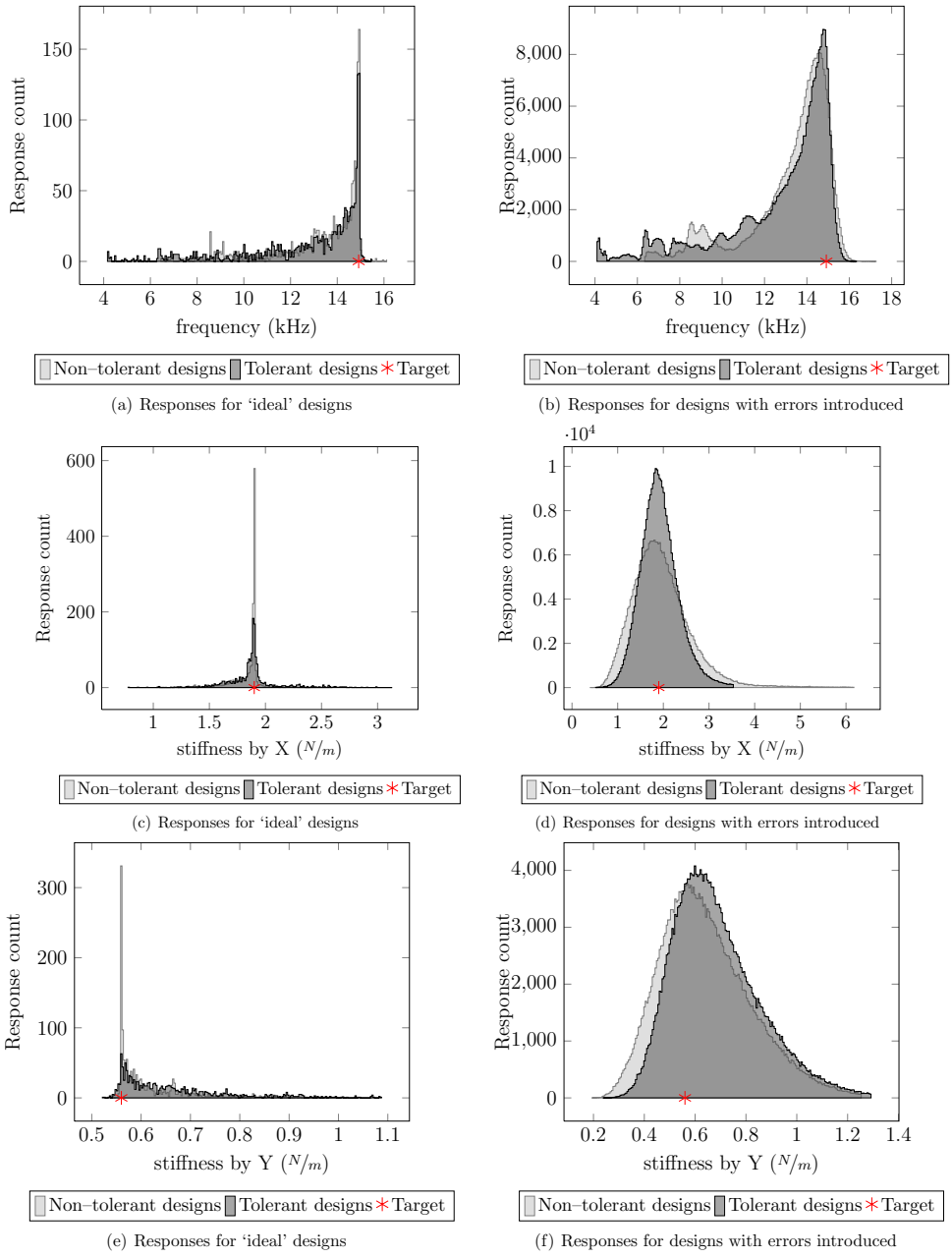
This change will make the objective to evolve smaller error tolerance relative to the desired response value r rather than its current 'ideal response' value $Response(\vec{d})$ (μ of distribution of devices' responses when 'ideal design' is subject to product errors).

To verify the proposed objective function (equation 5.6), a set of experiments were performed with same settings as previously attempted evolution of error tolerant MEMS designs inspired by robust design theory (please refer to Table 5.2 for detailed description). For the sake of clarity, the GA configuration for response–dependent error tolerant designs is listed in Table 5.4.

As shown by acquired results (Figures 5.14, 5.15, 5.16, 5.17, 5.18, 5.19, 5.20, 5.21, 5.22 and 5.23), the response–dependent error tolerance exhibited expected properties of error tolerance while bringing the overall population closer to target response.

5.5.2 Error tolerance as a separate objective

Another possible approach is to bring error tolerance objective as a separate GA objective dimension, therefore not allowing compromises in dimensions meant for achieving target response value. Because the extension of GA search space with multitude of error tolerance objectives (one objective for each GA response target) is undesirable due to the exponential increase in search space, it was considered that the creation of unified dimensionless error tolerance is more practical.



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

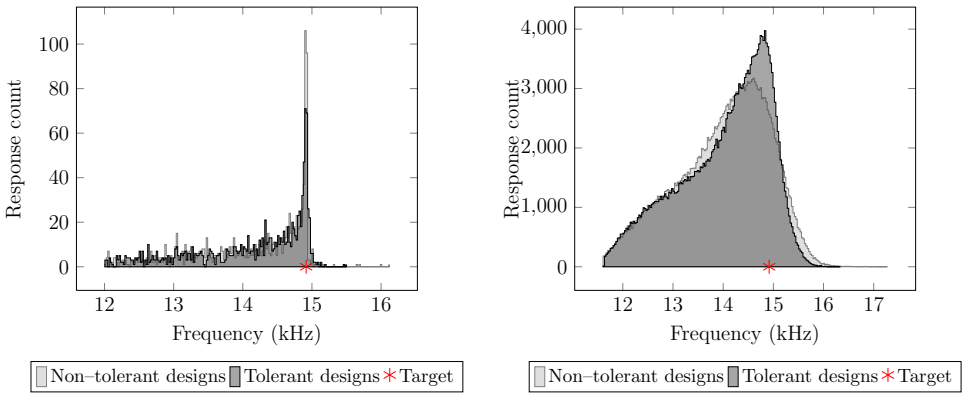
(c) Responses for 'ideal' designs

(d) Responses for designs with errors introduced

(e) Responses for 'ideal' designs

(f) Responses for designs with errors introduced

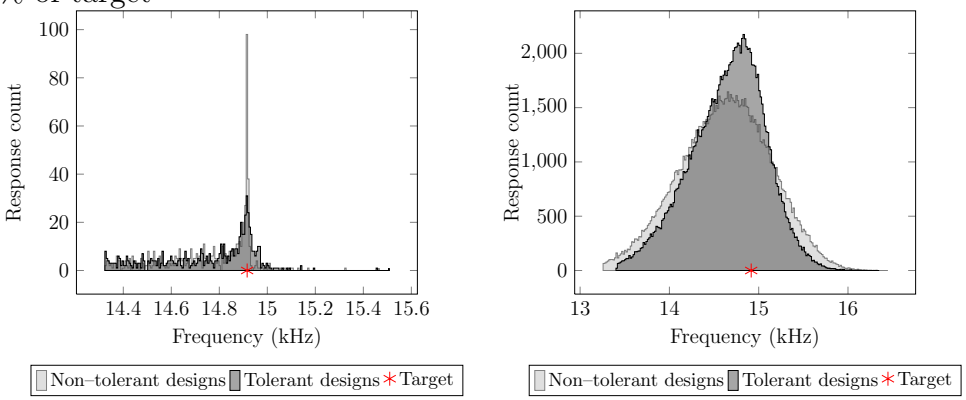
Figure 5.14: Response–relative error tolerance: full population results



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

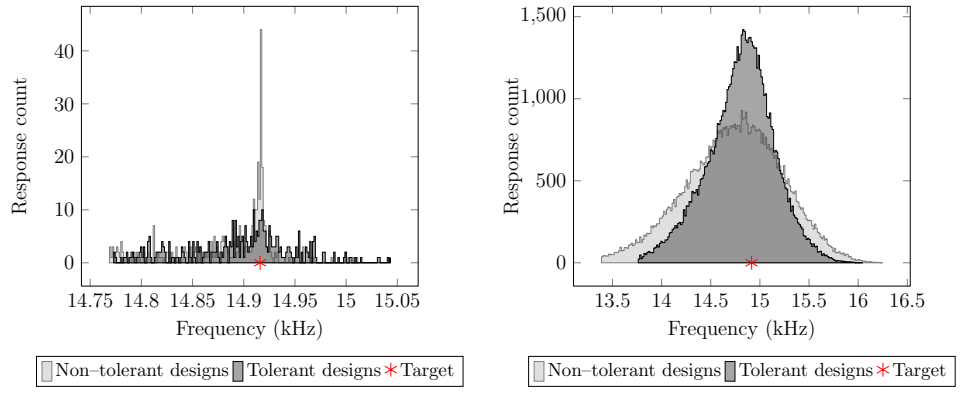
Figure 5.15: Designs for frequency of Response–relative error tolerance within 20% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

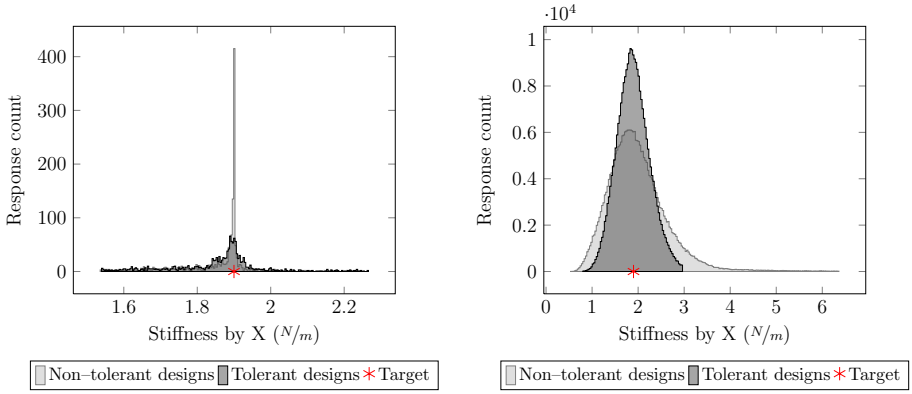
Figure 5.16: Designs for frequency of Response–relative error tolerance within 4% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

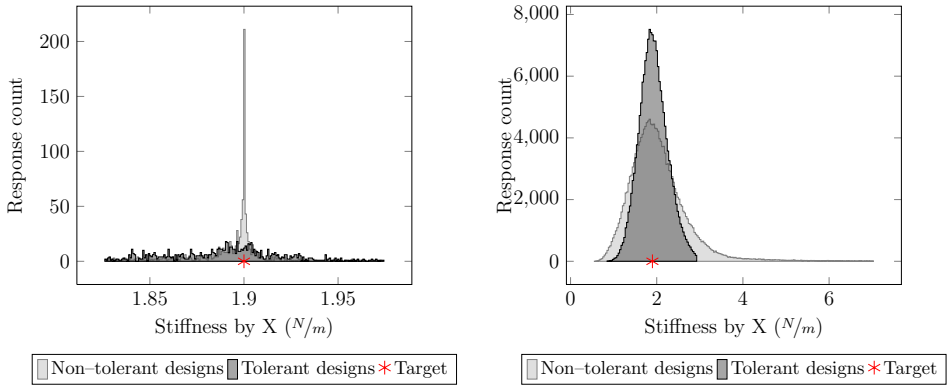
Figure 5.17: Designs for frequency of Response–relative error tolerance within 1% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

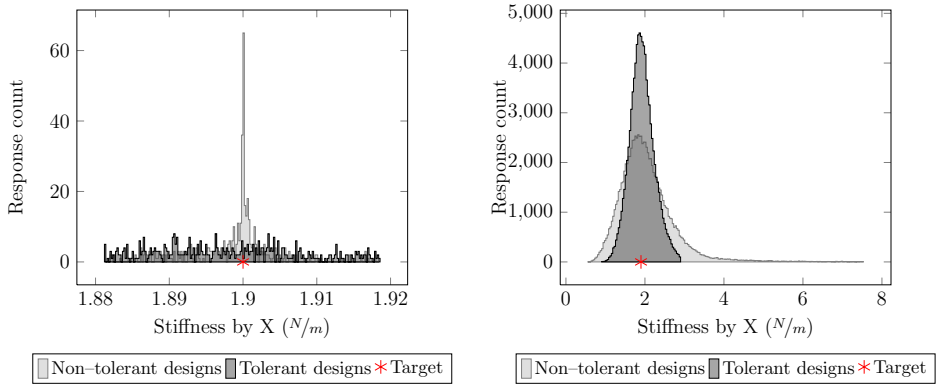
Figure 5.18: Designs for stiffness by X of Response–relative error tolerance within 20% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

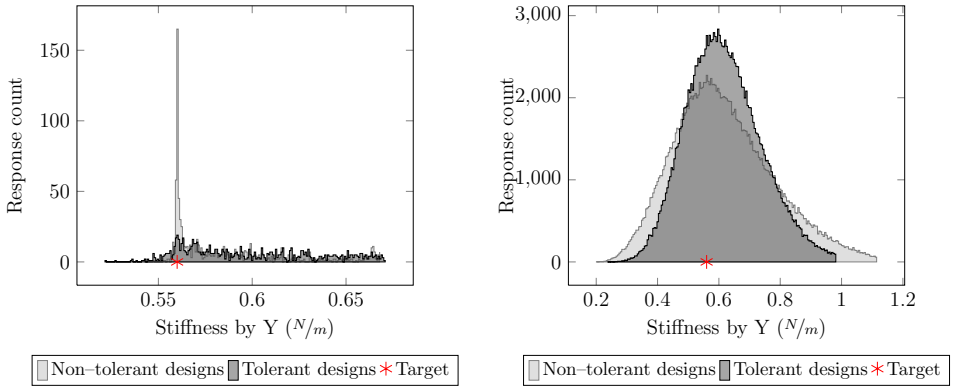
Figure 5.19: Designs for stiffness by X of Response–relative error tolerance within 4% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

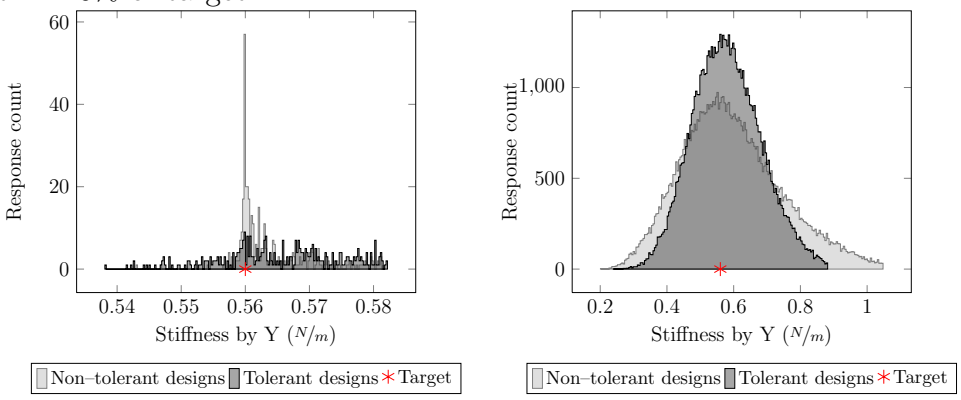
Figure 5.20: Designs for stiffness by X of Response–relative error tolerance within 1% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

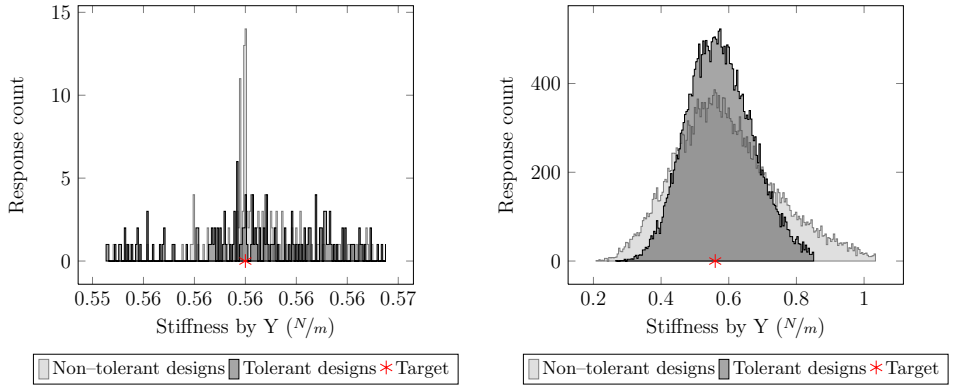
Figure 5.21: Designs for stiffness by Y of Response–relative error tolerance within 20% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

Figure 5.22: Designs for stiffness by Y of Response–relative error tolerance within 4% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

Figure 5.23: Designs for stiffness by Y of Response–relative error tolerance within 1% of target

Meandering resonator design synthesis with error tolerance towards objective value[1]		
Resonator design parameters		
Simulated layouts with errors per design		10
Error boundaries	min	0 μm
	max	1 μm
Objectives		
Frequency Objective		$T_1(93723, \vec{d}) + E_{obj_1}(\vec{d}, 93723)$
X stiffness objective		$T_2(1.90, \vec{d}) + E_{obj_2}(\vec{d}, 1.90)$
Y stiffness objective		$T_3(0.56, \vec{d}) + E_{obj_3}(\vec{d}, 0.56)$

Notes
[1] These settings are extension to GA settings defined in Table 5.1

Table 5.4: Settings for design synthesis of multi–tolerant meandering resonator

The introduced objective is based on normalised $E(\vec{d})$ modification as defined in equation 5.7, that can be extended as equation 5.8.

$$E_{norm}(\vec{d}) \equiv \frac{E_(\vec{d})}{Response(\vec{d})} \tag{5.7}$$

$$E_{norm}(\vec{d}) \equiv \frac{\langle i \in N : Response(Err_{prod}(\vec{d})) - Response(\vec{d}) \rangle}{Response(\vec{d})} \tag{5.8}$$

As self–evident, the $E_{norm}(\vec{d})$ is dimensionless and therefore $Error_{norm}(\vec{d})$ for responses of various domains can be summed. This is exploited when cumulative error tolerance objective is defined as shown in equation 5.9 (where B is the set of all response dimensions. For the meandering resonator problem, this would contain responses 'frequency', 'stiffness by X' and 'stiffness by Y').

$$Variance(\vec{d}) \equiv \langle \forall b \in B : Error^b_{norm}(\vec{d}) \rangle \tag{5.9}$$

Therefore, an alternative approach to defining objectives for error–tolerant GA is defined in equation 5.10 (it is implied that GA is minimising objecitves).

Error–tolerant meandering resonator design synthesis with error [1]		
Resonator design parameters		
Simulated layouts with errors per design		10
Error boundaries	min	0 μm
	max	1 μm
Objectives		
Frequency Objective		$T_1(93723, \vec{d})$
X stiffness objective		$T_2(1.90, \vec{d})$
Y stiffness objective		$T_3(0.56, \vec{d})$
Error tolerance objective		$\sum_3^{i=1} E_{norm_i}(\vec{d})$

Notes
[1] These settings are extension to GA settings defined in Table 5.1

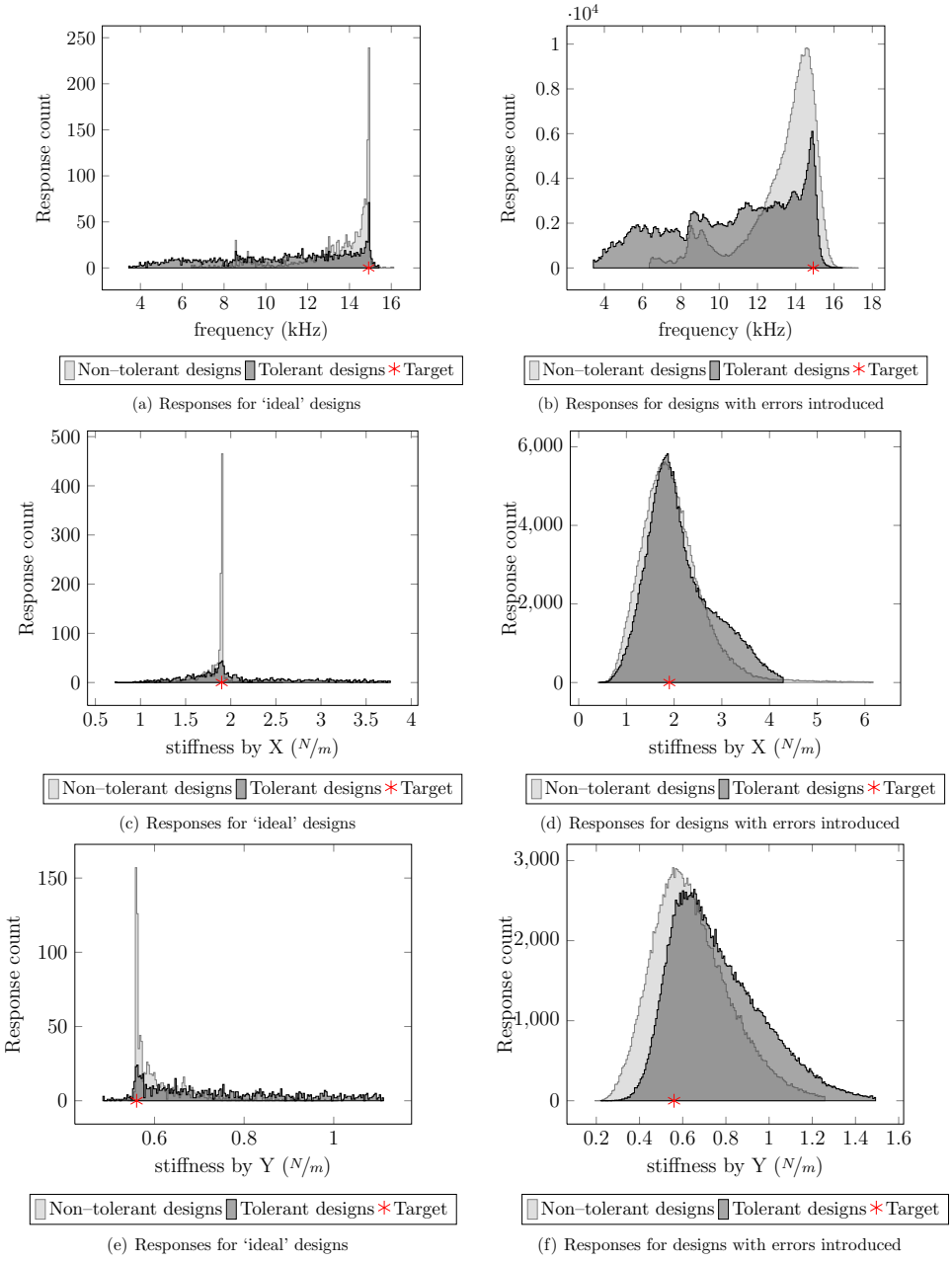
Table 5.5: Settings for design synthesis of multi–tolerant meandering resonator

$$objectives = \begin{bmatrix} T_0(r_0, \vec{d}) \\ T_1(r_1, \vec{d}) \\ \dots \\ T_N(r_N, \vec{d}) \\ Variance(\vec{d}) \end{bmatrix} \tag{5.10}$$

To verify the proposed objective function (equation 5.10), a set of experiments were performed with same settings as previously attempted evolution of error tolerant MEMS designs inspired by robust design theory (please refer to Table 5.5 for detailed description).

As can be seen from the displayed graphs of full populations (Figure 5.24), the distribution of designs is more uniform when using given GA configuration. This is because of extra dimension making more designs to be located on Pareto front, therefore limiting the overall movement of population towards desired responses in objective functions (that is frequency and per–axis stiffness).

The downside of this effect is that the means of distributions of MEMS designs are likely to be located further from target values (this idea is sup-



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

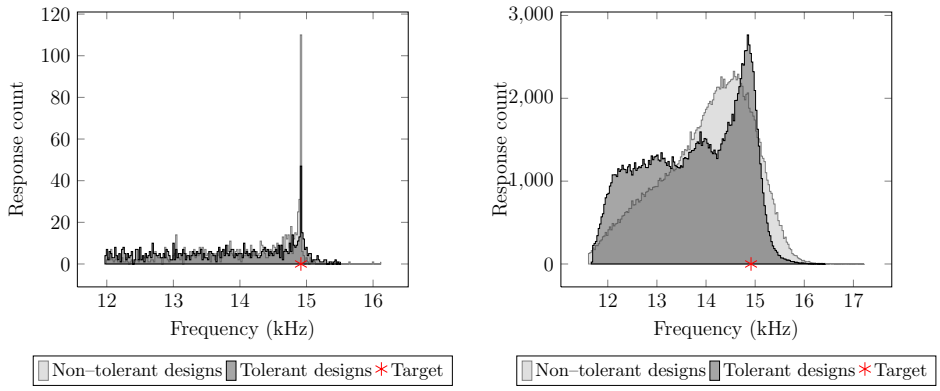
(c) Responses for 'ideal' designs

(d) Responses for designs with errors introduced

(e) Responses for 'ideal' designs

(f) Responses for designs with errors introduced

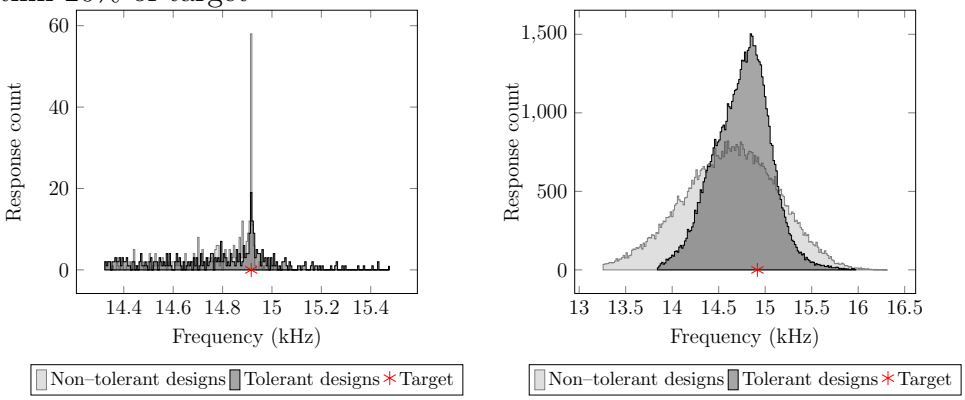
Figure 5.24: Error tolerance in separate objective: full population results



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

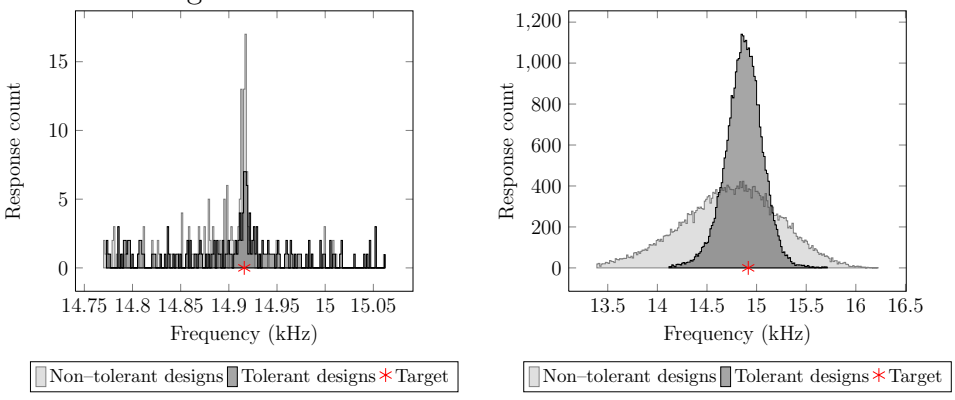
Figure 5.25: Designs for frequency of Error tolerance in separate objective within 20% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

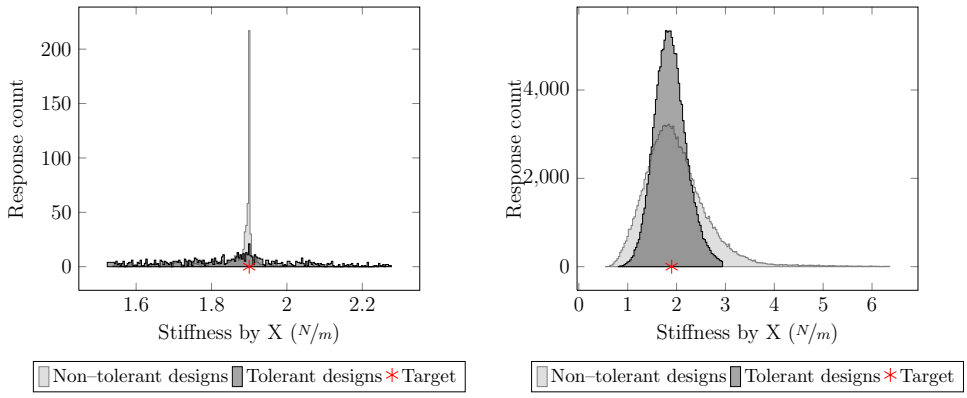
Figure 5.26: Designs for frequency of Error tolerance in separate objective within 4% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

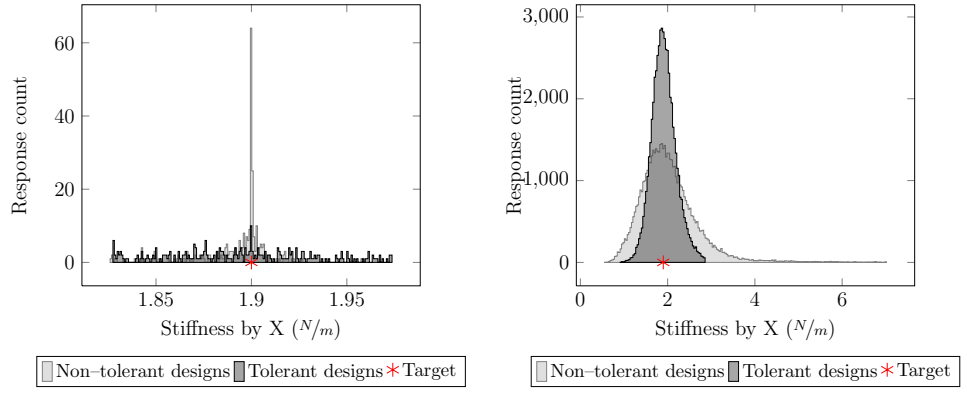
Figure 5.27: Designs for frequency of Error tolerance in separate objective within 1% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

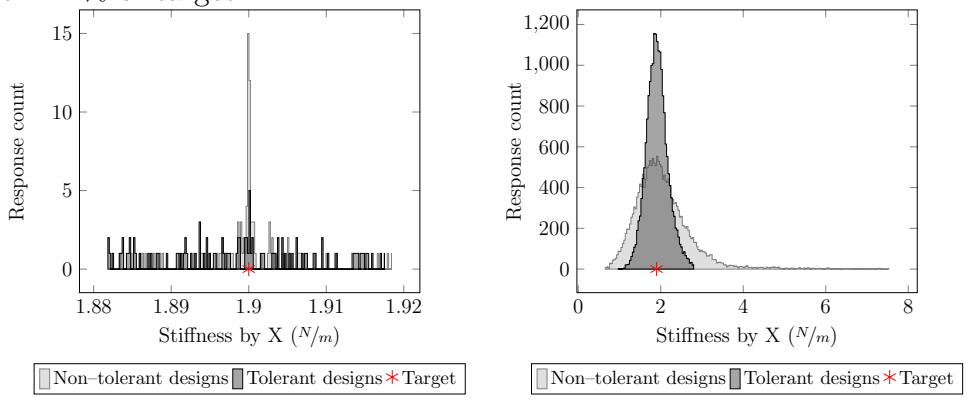
Figure 5.28: Designs for stiffness by X of Error tolerance in separate objective within 20% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

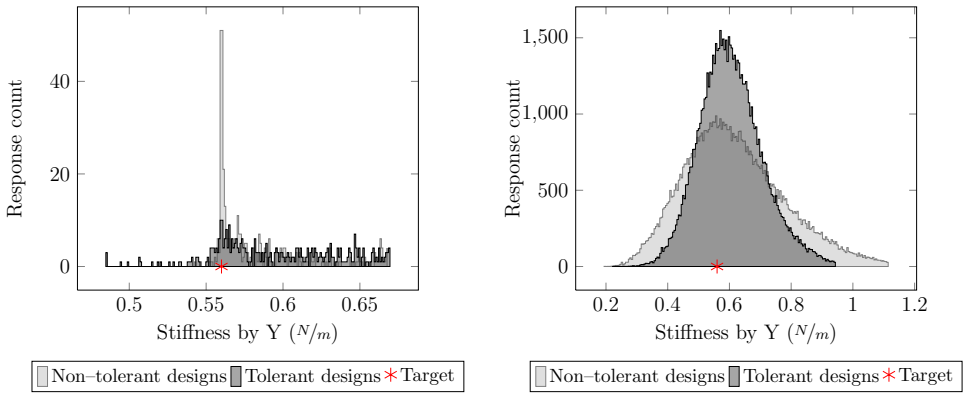
Figure 5.29: Designs for stiffness by X of Error tolerance in separate objective within 4% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

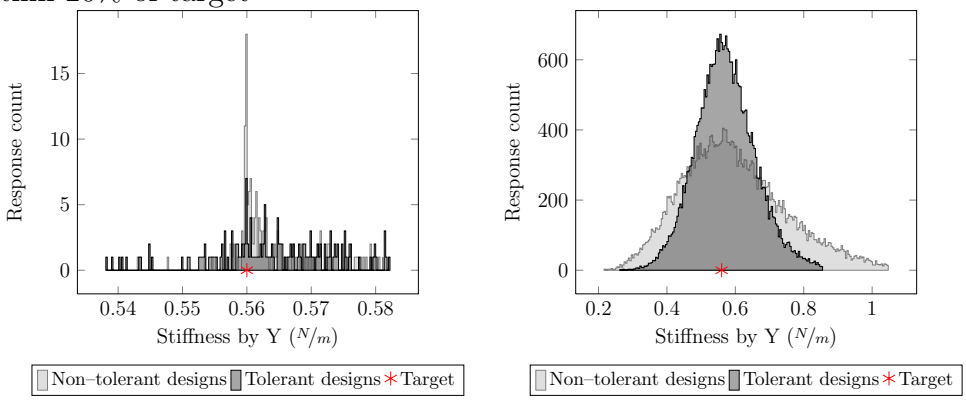
Figure 5.30: Designs for stiffness by X of Error tolerance in separate objective within 1% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

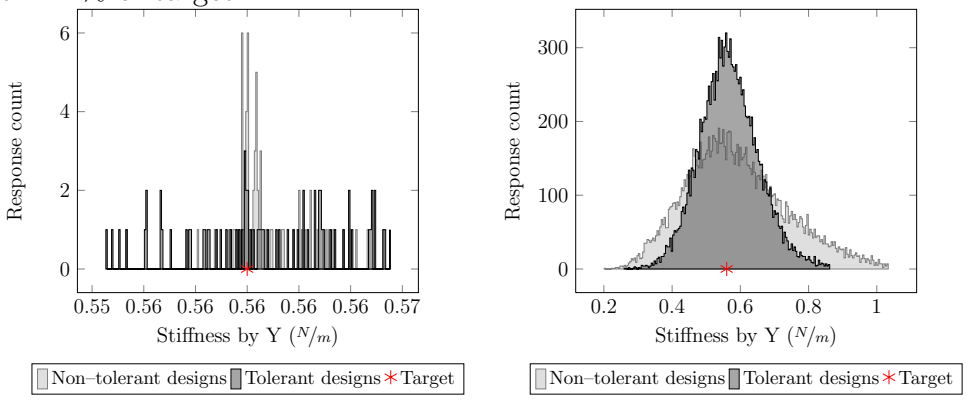
Figure 5.31: Designs for stiffness by Y of Error tolerance in separate objective within 20% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

Figure 5.32: Designs for stiffness by Y of Error tolerance in separate objective within 4% of target



(a) Responses for 'ideal' designs

(b) Responses for designs with errors introduced

Figure 5.33: Designs for stiffness by Y of Error tolerance in separate objective within 1% of target

ported by the plots of 'ideal' designs in full population graphs – Figure 5.24 and by distribution of 'ideal' designs in subsets of population in Figures 5.25, 5.26, 5.27, 5.28, 5.29, 5.30, 5.31, 5.32 and 5.33). Judging by the graphs, it seems that the proposed approach towards the evolution of error tolerance is undesirable, but overall conclusions will be drawn later, during comparative analysis of results yielded by all tested GA configurations.

5.6 Discussion of results

Because the performance of GA should be measured by the ability of GA to achieve desired outcome, a set of designs needs to be selected to represent the achievements of each of the three modifications of GApresented in sections 5.4, 5.5.1 and 5.5.2 plus the control non–tolerant designs evolved with settings described in Table 5.3.

The selection of best performing designs is made by ordering a unified set of all elements found on all pareto fronts of all runs for each configuration. The ordering of elements in given set is carried out based on metric M_Σ.

$$M_\Sigma \equiv \sqrt{\sum_{3}^{n=1} \left(1 - \frac{Response_n(\vec{x})}{r_n}\right)^2} \tag{5.11}$$

With r_n is target value for nth response parameter, and $Response_n(\vec{x})$ is evaluation of nth response parameter of a design $\vec{x}$ GA performs the evolution of MEMS meandering resonator with the aim of evolving designs having three particular responses (frequency, stiffness by X and stiffness by Y). It is evident that M_Σ is normalised euclidian distance for point of target responses.

After the elements were ordered, 21 designs with smallest M_Σ metrics were chosen for detailed performance evaluation. This evaluation was made by the creation of 10,000 MEMS simulations with production errors introduced for each of the 21 chosen designs.

The resulting 10,000 values for each response were fitted to normal distribution with Matlab '`normfit`' function and output pair of (μ, σ) saved with respect to 'ideal' MEMS design that produced it. The resulting 21 pairs of

Frequency (Hz)		Stiffness by X ($^N/_m$)		Stiffness by Y ($^N/_m$)	
μ	σ	μ	σ	μ	σ
Control (non–tolerant) designs					
14605	521.35	2.0509	0.5771	0.6098	0.1784
Error dependency based on robust optimisation theorem					
14203	366.24	1.9824	0.4072	0.602	0.1195
Response–dependent error tolerance					
14174	451.68	1.9442	0.3689	0.5916	0.1071
Error tolerance in separate objective					
11249	214.48	1.6941	0.3485	0.6683	0.1342

Table 5.6: Generalised average parameters of best results

Frequency		Stiffness by X		Stiffness by Y	
$\left\|1-\frac{\mu}{14916}\right\|$	$\frac{\sigma}{14916}$	$\left\|1-\frac{\mu}{1.90}\right\|$	$\frac{\sigma}{1.90}$	$\left\|1-\frac{\mu}{0.56}\right\|$	$\frac{\sigma}{0.56}$
Control (non–tolerant) designs					
0.02085	0.03495	0.07942	0.30374	0.08893	0.31857
Error dependency based on robust optimisation theorem					
0.0478	0.02455	0.04337	0.21432	0.075	0.21339
Response–dependent error tolerance					
0.04975	0.03028	0.02326	0.19416	0.05643	0.19125
Error tolerance in separate objective					
0.24584	0.01438	0.10837	0.18342	0.19339	0.23964

Table 5.7: Generalised relative average parameters of best results

(μ, σ) for each setup of an experiment are available in appendix B.

As can be seen from Table 5.6 and from appendix B, the M_Σ metric selects designs only based on their relative performance without respect to the relative importance of objectives (i.e. frequency objective in current set of results is more important than stiffness–related objectives). But this is not of concern at the current stage of research as the overall behaviour of the proposed algorithm is more important.

For better demonstration of relative performance, Table 5.7 is created to illustrate the actual relative errors in means $\left|1-\frac{\mu}{r_n}\right|$ and relative standard deviation $\frac{\sigma}{r_n}$ of resulting designs.

As the M_Σ metric is not making any distinction between different objectives and is based on designs' relative performance, it makes sense to discuss

$\langle \forall n : \left\lvert 1 - \frac{\mu_n}{r_n} \right\rvert \rangle$	$\langle \forall n : \frac{\sigma_n}{r_n} \rangle$
Control (non–tolerant) designs	
0.0631	0.2191
Error dependency based on robust optimisation theorem	
0.0554	0.1508
Response–dependent error tolerance	
0.0432	0.13856
Error tolerance in separate objective	
0.1825	0.14581

Table 5.8: Average design performance

average designs' performance in all objective dimensions. For better visual perception, the average for all response means (μ) in all objective dimensions ($\langle \forall n : \left\lvert 1 - \frac{\mu}{r_n} \right\rvert \rangle$) and average for all response dispersions (σ) ($\langle \forall n : \frac{\sigma}{r_n} \rangle$) were calculated for all sets of experiments and are displayed in Table 5.8.

It can be seen that the worst performance is demonstrated by 'error tolerance as separate objective' approach, as it failed to evolve designs with average relative mean response rate within 18% of targets and as high as 24.6% for frequency objective (see Table 5.7), which is higher than SUGAR's commonly expected response prediction precision (that is 20%). This makes 'error tolerance in separate objective' algorithm to be an outlier as it is effectively failing to achieve desired response parameters. This illustrates that the idea of adding one extra dimension is not viable, as with the increase in the dimensionality of objective space the overall performance of GA noticeably decreases.

By comparing results for three other experiment configurations (control, robust optimisation–inspired approach and response–dependent error tolerances), it is easy to see that the best results in both average approach of means of responses (μ) and error–caused dispersions of errors (σ). The best results are shown by 'response–dependent error tolerance' experiment configuration, that shows $\frac{0.0631}{0.0432} = 68.5\%$ more precise results in terms of average of response distribution means and $\frac{0.2191}{0.13856} = 158\%$ smaller average error dispersion than non–tolerant designs.

Chapter 6

Conclusions and Future Research

6.1 Summary

MEMS designs are generally created by build–and–break process, a process that takes long time to complete – starting with half a year and more.

To address the problem of extremely long design phase for MEMS design synthesis, a number of computational tools for MEMS computer–assisted design evaluation such as ANSYS(ANSYS Inc. 2009) of SUGAR were introduced. This will allow the designers to build and evaluate virtual models, reducing the amount of required build–and–break cycles for the development of MEMS designs.

But as the MEMS market is steadily increasing due to steady increase in MEMS deployment in medicine, consumer electronics, automative and other markets, the need for automated MEMS layout design synthesis methodology is arising. The deployment of black–box optimisation Artificial Intelligence (AI) techniques such as GA allows for human designers to shorten the initial phase of layout synthesis, as GA is capable of performing black–box optimisation for problems with a size of search space that would be un–processable using conventional optimisation techniques such as exhaustive search.

MEMS models come in two classes – (i) analytical, that is basically a

set of formulas describing the behaviour of particular topologies of MEMS designs (ii) and numerical – a model that is using FE simulation toolboxes such as ANSYS(ANSYS Inc. 2009) or SUGAR for estimation of MEMS design parameter responses. The decision of MEMS design class to be optimised therefore needs to be done. Due to the limitations and general lack of analytical models coupled with noticeable difficulties in the creation of new analytical models, the choice was made in favour of numerical models with SUGAR MEMS simulation environment.

The ability of multiobjective GA to return a set of alternative solutions without making sacrifices to one MEMS response over another was considered valuable in the case of MEMS layout design synthesis, as the multi–domain nature of MEMS makes it extremely hard to prioritise one design's response over another (which possibly belongs to a completely different domain of physics).

In this thesis, a new efficient GA circular chromosomal structure was introduced. It was demonstrated that the proposed chromosome reduces time and number of evaluations required to perform MEMS design synthesis by the means of GA. In previous work, which addressed different areas of GA, a special crossover operator was developed. This operator was incorporated to the workflow of GA. It was illustrated by the previous research that such a circular chromosomal structure has redundancy and is expected to possess linkage learning functionality – an effect that could be beneficial for MEMS evolutionary synthesis. It was demonstrated that this has a positive influence on MEMS evolutionary synthesis performance.

The promising results from the successful introduction of circular chromosome to the problem of MEMS evolutionary synthesis with its linkage–learning functionality led to the research on automated Building Block (BB) generation for the problem of MEMS design synthesis. Several approaches for automated BB handling were explored. Several promising realisations were identified and the basis for successful realisations to exhibit observed performance was proposed.

Yet another big and complex problem faced by the MEMS industry was addressed in successive chapter. This chapter focused on the creation of

MEMS designs that are tolerant to production errors as the production errors are of concern when designing MEMS devices due to the relatively low precision of etching technology commonly used. With the introduction of error tolerance–evolving GA objectives that were inspired by the robust optimisation approach found in literature, the attempts to introduce several enhancements to the initial approach were successful. An approach was introduced for the evolution of error–tolerant MEMS designs with no analytical models, only with numerical simulation.

Although volume of case studies used in each chapter for obtaining results is somehow limited, as the attempted approaches are not tuned for particular problems, it is reasonable to expect that results presented in this work are applicable to much more general set of problems than they were tested on in current work.

The MEMS resonator problem, although simple, possesses properties important and common for major part of MEMS design problems – the design itself is sensitive to small changes in devices' layout, the volume of incorrect layouts (such as self–intersecting designs) is much greater than volume of producible designs and optimisation problem is multiobjective in its nature, just as it is in general case of creating MEMS designs.

6.2 Research limitations

More research should be done on the influence of linkage and redundancy on MEMS layout synthesis optimisation. The general motivation for GA to preserve the good design elements in the redundant parts of genes seem to be supported by experimental results. But more research on the behaviour of linkage–learning capabilities of circular chromosome in the case of MEMS layout design synthesis is required. The effects and statistical evidence of the re–usage of design elements by GA is desirable, but not researched in current thesis due to the time constraints.

Regarding the automated MEMS BB synthesis, a major research limitation is the simplicity of chosen approach, as the application of proposed component–based MEMS design synthesis for meandering resonator is an

application of potentially complex decision support system to a greatly simplified problem. The proposed layout synthesis approach using automatically generated BBs should be applied to a more complex problem, a problem that can have actual building blocks (subsets of nodes) evolved and extracted rather than the usage of a big design element such as the whole leg of a meandering resonator.

The research on error–tolerant layout synthesis was greatly limited due to the time constraints. Because of this, no research on the effect of the number of evaluations of error–introduced designs on the level of error tolerance of resulting designs was made. This limitation of research is both desirable and relatively simple to overcome.

6.3 Conclusions

It was demonstrated that the circular chromosome that was proposed in chapter 3 outperforms currently published state–of–the–art algorithms when compared using yielded results. It is considered worthwhile to continue research on this topic in the future.

In the case of research on component–based MEMS synthesis (Chapter 4), it was demonstrated that automated component generation is possible, but considerable difficulties can emerge when choosing particular approach towards extraction and re–usage of Building Blocks (BBs). The introduction of automatically generated BBs in to the process of automated MEMS layout synthesis is highly desirable and successful implementation of given approach would allow for GA to use the libraries of automatically generated MEMS components. The future research in given direction is considered important and desirable.

The design of error–tolerant MEMS layouts researched in Chapter 5 demonstrated the ability of GA to evolve error–tolerant MEMS designs using numerical simulation rather than using analytical models of MEMS designs.

A number of research objectives were listed in section 1.1. Results and outcomes of pursuing these objectives are given below.

1. To propose circular chromosome with linkage learning capabilities to address the issues associated with large number of objective function evaluations and inefficient MEMS structure representation in GA.

   In Chapter 3, the novel circular chromosome data structure was introduced to the field of MEMS design layout synthesis and optimisation. This chromosome allowed to synthesise MEMS layout designs that are superior to ones provided by previous state–of–the-art research using only half of functional evaluations when compared to previous research.

2. To develop automatic BB detection and re-use approach to address the inability of GA in automatically re–using past MEMS designs and learning and preserving good MEMS design elements.

   In Chapter 4, the automated BB construction, storage and re–usage approach was proposed and explored. As the BB were generated using MEMS designs produced by circular chromosome algorithm, the linkage learning capabilities of this chromosome were deployed in given procedure.

   Compared to classic GA, the attempted approach was demonstrated its ability to produce MEMS layout designs that with responses closer to desired values.

3. To propose error–tolerant MEMS layout synthesis to enable automated synthesis of production error tolerant MEMS designs by the means of GA.

   In Chapter 5, the problem of automated MEMS error–tolerant layout synthesis was addressed using several approaches inspired by robust optimisation methodology.

   Results yielded by attempted approaches demonstrated ability for GA to evolve MEMS error–tolerant layouts with tolerance in various responses by using numerical simulation.

This thesis has contributed to the field of MEMS design synthesis and optimisation by the means of GA in several ways.

The introduction of circular chromosome has proposed a way for increasing efficiency of MEMS design layout synthesis. Furthermore, circular chromosome has introduced linkage learning to the MEMS design layout synthesis, and as the linkage learning might allow for GA to build, identify and re–use blocks of linked genes as a BB in the future, this contribution might have noticeable impact on the performance and efficiency of the automated MEMS layout synthesis.

The idea of creation MEMS BB library for the automated design synthesis methods to deploy knowledge previously gathered is widely discussed in the literature, but general lack of interest contributing to such library amongst with big volume of BB needed to be gathered, does not allow for this library to be brought to reality. The introduction of fully automated BB acquire, storage and re–usage cycle pursued in Chapter 4 allows to explore alternative path for construction of such library. Rather than filling it with human–generated designs, current approach allows for library to be automatically populated using BB generated by automatic methods. This could allow to enhance performance and to gather new designs produced by automated MEMS layout synthesis with no human contribution.

Problem of instability of MEMS responses when subject to production errors was tackled in Chapter 5. The ability of GA to synthesise error–tolerant designs was demonstrated, potentially increasing yield of MEMS production by lowering defect ratio.

6.4 Future Research

6.4.1 Circular chromosome

The results of the experiments showed that the representation of genetic information as circular chromosome increases GA efficiency in finding the good MEMS designs in global decision space. But it is inefficient when it comes to fine–tuning of designs found.

The future research in this direction might concentrate on this problem and there are a few approaches that seem promising.

1. Introduction of a process similar to the positive reinforcement loop, for example, the manual increase of probabilities for start/stop interpretation commands, which were used to produce given MEMS designs in the solutions belonging to the Pareto front.

2. Introduction of the conventional means of optimisation, such as the gradient–based method, to find solutions belonging to local optima.

As the circular chromosome had introduced linkage learning and redundancy to the MEMS evolutionary synthesis process, furtherer research on the benefits and possibilities of these properties of GA chromosome for the domain of MEMS design synthesis is desirable.

The development of novel crossover operator for MEMS should be of benefit for the problem of MEMS synthesis, as the MEMS encodings commonly expect for each gene in the chromosome to encode several parameters at once (such as node length, width and angle), making each gene to be constructed of several independent values. The common GA crossover operators act on a chromosome as if it is a vector of primitive values.

According to currently deployed concept of crossover of the circular chromosome, the linkage learning happens purely stochastically, while it might be possible to introduce some kind of artificial intelligence to learn and extract the actual building blocks from device phenotypes.

6.4.2 Component–based MEMS synthesis

One might attempt addressing the problem of automated BB recognition and automative recognition of properties needed to be preserved in certain BB types. This could be a good place for application of a variety of techniques from the domain of image recognition, as the process of creation of image recognition algorithms is somehow similar – an algorithm is given a number of particular images containing and not containing certain object and the algorithm needs to distill the traits of the presence of the desired object. The set of the resulting traits could be considered as an 'abstract model' of the searched model. Similar techniques could be used for the extraction of

an 'abstract element' of the MEMS design with such, as abstract element to be used as BB later.

6.4.3 Process–induced error tolerant layout synthesis

Research on the influence of the count of errorous evaluations on error tolerance

The Current set of experiments was performed with GA doing ten evaluations of error–introduced designs for every generation's design (or, as these designs are sometimes called in this chapter 'ideal designs').

The increase in the count of simulations of error–introduced designs might influence the overall level of error tolerance, as Fan et al. in (Fan et al. 2007) presented results $\sigma_{robust} = 1.95kHz$ and $\sigma_{non\_robust} = 11.3kHz$ for crab–leg (or as it referred in to given paper – 'meandering' resonator). Since this publication displays $\frac{11.3}{1.95} = 579.5\%$ decrease in robust designs' standard deviation, it is likely that there is a room for improvement of best result achieved in this work (by 'response–dependent error tolerance' approach) by 158% and it is probable that the given improvement can be achieved through more excessive probing of error tolerance of designs during their evolution.

Development of specialised metrics

The M_{Σ} metric used in current work is a simple method for selection of 'best' (or most representative) designs from resulting Pareto front. More sophisticated approaches with better relevance towards reality (i.e. ones that addresses that frequency objective is more important than stiffness objectives) probably could and certainly should be developed.

Yet another metrics whose development should be researched is one that the dynamic properties of MEMS designs and helps to select error–tolerant designs.

Research on the influence of error tolerance towards the speed of matching objectives

It is probable that the solutions with higher error tolerance (ones with smaller σ) evolve towards desired responses slower than non–tolerant solutions, as the error tolerance is a measurement of how stable designs' response is when designs are subject to errors. Since GA is evolving designs with crossover and mutation and the mutation is a process of introducing random errors into designs, the designs that are more tolerant to the errors could have weaker reaction when subject to mutation.

In case this issue is confirmed, the research on 'triggered error tolerance' evolution could be done. With this approach, designs should be subject to optimisation for error tolerance only when certain condition are met, i.e. when design is within certain neighbourhood of target response.

References

Abdalla, M. M., Reddy, C. K., Faris, W. F. & G\ürdal, Z. (2005). Optimal design of an electrostatically actuated microbeam for maximum pull-in voltage, *Computers & Structures* **83**(15-16): 1320–1329.

Achiche, S., Fan, Z. & Bolognini, F. (2007). Review of automated design and optimization of MEMS, *IEEE International Symposium on Industrial Electronics*, pp. 2150–2155.

ACM (n.d.). Acm taxonomy.
URL: *http://www.computer.org/portal/web/publications/acmtaxonomy*

Agrawal, R. B. & Deb, K. (1994). Simulated binary crossover for continuous search space.

ANSYS Inc. (2009). ANSYS.
URL: *http://www.ansys.com/*

Antonsson, E. K., Cavallaro, J., Mahadevan, R., Maher, M. & Maseeh, F. (1997). Structured design methods for MEMS final report.

Bai, Z. (2002). Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems, *Applied Numerical Mathematics* **43**(1-2): 9–44.

Bao, M. & Wang, W. (1996). Future of microelectromechanical systems (MEMS), *Sensors and Actuators A: Physical* **56**(1-2): 135–141.

Bardeen, J. & Brattain, W. (1948). The transistor, a Semi-Conductor triode, *Physical Review* **74**(2): 230–231.

Bechtold, T., Rudnyi, E. B. & Korvink, J. G. (2003). Automatic generation of compact Electro-Thermal models for semiconductor devices, *Ieice Transactions on Electronics* **86**: 459–465.

Bedair, S. S. & Fedder, G. K. (2005). CMOS MEMS oscillator for gas chemical detection, *Sensors, 2004. Proceedings of IEEE*, Vol. 2, pp. 955–958.

Benkhelifa, E., Farnsworth, M., Tiwari, A., Bandi, G. & Zhu, M. (2010). Design and optimisation of microelectromechanical systems: a review of the state–of–the–art, *International Journal of Design Engineering* **3**(1): 41 – 76.

Brenner, M., Lang, J., Li, J., Qiu, J. & Slocum, A. (2002). Optimum design of a MEMS switch, *2002 International Conference on Modeling and Simulation of Microsystems - MSM 2002*, pp. 214–217.

Calis, M. & Desmulliez, M. (2006). Haptic technologies for MEMS design, *Journal of Physics: Conference Series* **34**: 72–75.

Chaturantabut, S. & Sorensen, D. C. (2010). Nonlinear model reduction via discrete empirical interpolation, *SIAM Journal on Scientific Computing* **32**: 2737.

Chen, C. W. (1996). Complications and implications of linear bacterial chromosomes, *Trends in Genetics* **12**(5): 192–196.

Chen, Y.-p. & Goldberg, D. E. (2002). Introducing start expression genes to the linkage learning genetic algorithm, *Parallel problem solving from Nature, 7 , 351–360.* pp. 351—360.

Chi, T., Hsu, F. & Lin, C. (2007). The effect of GA redundancy on the design of reverse logistics network, *Third International Conference on Natural Computation*, Vol. 5, ICNC, IEEE Computer Society, pp. 504–510.

Chi, T. & Lin, C. (2008). The characteristic of logistics network specific coding in genetic algorithms, *Technical Report 186.*

Clark, J. V., Bindel, D., Kao, W., Zhu, E., Kuo, A., Zhou, N., Nie, J., Demmel, J., Bai, Z., Govindjee, S. et al. (2002). Addressing the needs of complex MEMS design, *Micro Electro Mechanical Systems, 2002. The Fifteenth IEEE International Conference on*, pp. 204–209.

Clark, J. V., Zhou, N. & Pister, K. (1998). *MEMS Simulation Using Sugar v0.5*.

Cobb, C. & Agogino, A. (2006). Case-based reasoning for the design of micro-electro-mechanical systems, *Proceedings of the ASME Design Engineering Technical Conference*, Vol. 2006.

Cobb, C., Zhang, Y. & Agogino, A. (2007a). An integrated MEMS design synthesis architecture using Case-Based reasoning and Multi-Objective genetic algorithms, *Smart structures, devices, and systems III 11-13 December 2006, Adelaire, Australia*, Vol. 6414, SPIE, Bellingham, Wash.

Cobb, C., Zhang, Y. & Agogino, A. (2007b). MEMS design synthesis: Integrating case-based reasoning and multi-objective genetic algorithms, *Proceedings of SPIE - The International Society for Optical Engineering*, Vol. 6414.

Cobb, C., Zhang, Y., Agogino, A. & Mangold, J. (2008a). Case-based reasoning and object-oriented data structures exploit biological analogs to generate virtual evolutionary linkages, *2007 IEEE Congress on Evolutionary Computation, CEC 2007*, pp. 334–341.

Cobb, C., Zhang, Y., Agogino, A. & Mangold, J. (2008b). *Knowledge-based evolutionary linkage in MEMS design synthesis*, Vol. 157.

Crary, S., Juma, O. & Zhang, Y. (1991). Software tools for designers of sensor and actuator CAE systems, *Transducers '91: 1991 International Conference on Solid-State Sensors and Actuators. Digest of Technical Papers*, San Francisco, CA, USA, pp. 498–501.

Crary, S. & Zhang, Y. (1990). CAEMEMS: an integrated computer-aided engineering workbench for micro-electro-mechanical systems, *IEEE Pro-*

ceedings on Micro Electro Mechanical Systems, An Investigation of Micro Structures, Sensors, Actuators, Machines and Robots., Napa Valley, CA, USA, pp. 113–114.

Deep, K. & Thakur, M. (2007). A new crossover operator for real coded genetic algorithms, *Applied Mathematics and Computation* **188**(1): 895–911.

Engesser, M., Franke, A. R., Maute, M., Meisel, D. C. & Korvink, J. G. (2010). A robust and flexible optimization technique for efficient shrinking of MEMS accelerometers, *Microsystem Technologies* **16**(4): 647–654.

ESTECO s.r.l (2009). modeFRONTIER.
URL: *http://www.esteco.com/products.jsp*

Fan, Z., Liu, J., Sorensen, T. & Wang, P. (2009). Improved differential evolution based on stochastic ranking for robust layout synthesis of MEMS components, *IEEE Transactions on Industrial Electronics* **56**(4): 937.

Fan, Z., Wang, J., Achiche, S., Goodman, E. & Rosenberg, R. (2008). Structured synthesis of MEMS using evolutionary approaches, *Applied Soft Computing* **8**(1): 579–589.

Fan, Z., Wang, J., Wen, M., Goodman, E. & Rosenberg, R. (2007). An evolutionary approach for robust layout synthesis of MEMS, *Evolutionary Computation in Dynamic and Uncertain Environments* pp. 519–542.

Fedder, G. K., Iyer, S. & Mukherjee, T. (1997). Automated optimal synthesis of microresonators, *International Conference on Solid-State Sensors and Actuators, Proceedings*, Vol. 2, pp. 1109–1112.

Fedorova, L. . & Fedorov, A. . (2005). Puzzles of the human genome: Why do we need our introns?, *Current Genomics* **6**: 589–595.

Feynman, R. P. (1999). There's plenty of room at the bottom, *Feynman and computation: exploring the limits of computers*, Perseus Books, pp. 63–76.

Fleischer, M. & Fleischer, M. (2003). The measure of pareto optima. applications to multi–objective metaheuristics, *Evolutionary multi-criterion optimization. Second international conference, EMO 2003* **2632**: 519—533.

Fogel, D. B. (1999). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 2 edn, Wiley-IEEE Press.

Forrest, S. & States., U. (1993). *Proceedings of the Fifth International Conference on Genetic Algorithms : University of Illinois at Urbana-Champaign, July 17-21, 1993*, Morgan Kaufmann Publishers, San Mateo Calif.

Freund, R. W. (2004). SPRIM: structure-preserving reduced-order interconnect macromodeling, *iccad*, pp. 80–87.

Gad-el-Hak, M. (2006). *MEMS : introduction and fundamentals*, 2nd ed. edn, CRC/Taylor & Francis, Boca Raton.

Gilleo, K. (2005). MEMS in medicine, *Circuits Assembly* **16**(8): 32–33.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1 edn, Addison-Wesley Professional.

Goldberg, D. E., Lobo, F. G., Deb, K., Harik, G. R. & Wang, L. (1998). Compressed introns in a linkage learning genetic algorithm, *Genetc programming: proceedings of the third annual conference* pp. 551—558.

Google Inc. (2010). Google scholar.
URL: *http://scholar.google.com*

Grayson, A. C., Shawgo, R. S., Johnson, A. M., Flynn, N. T., Li, Y., CIMA, M. J. & Langer, R. (2005). A BioMEMS review: MEMS technology for physiologically integrated devices, *Proceedings of the IEEE* **92**(1): 6–21.

Grefenstette, J. (1986). Optimization of control parameters for genetic algorithms, *IEEE Trans. Syst. Man Cybern.* **16**(1): 122–128.

Gyimesi, M., Avdeev, I. & Ostergaard, D. (2004). Finite-element simulation of micro-electromechanical systems (MEMS) by strongly coupled electromechanical transducers, *Magnetics, IEEE Transactions on* **40**(2): 557–560.

Harik, G. R. (1997). *Learning Gene Linkage to Efficiently Solve Problems of Bounded Difficulty Using Genetic Algorithms*, PhD thesis, The University of Michigan.

Harik, G. R. & Goldberg, D. E. (1997). Learning linkage, *Foundations of Genetic Algorithms 4*, Morgan Kaufmann, pp. 247—262.

Harik, G. R. & Goldberg, D. E. (2000). Linkage learning through probabilistic expression, *Computer Methods in Applied Mechanics and Engineering* **186**(2-4): 295–310.

Haronian, D. (1995). Maximizing microelectromechanical sensor and actuator sensitivity by optimizing geometry, *Sensors and Actuators: A. Physical* **50**(3): 223–236.

Holland, J. (1992). *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence*, 1st MIT press ed. edn, MIT Press.

Hong, Y. S., Lee, J. H. & Kim, S. H. (2000). A laterally driven symmetric micro-resonator for gyroscopic applications, *Journal of micromechanics and microengineering* **10**: 452.

Hornby, G., Kraus, W. & Lohn, J. (2008). *Evolving MEMS resonator designs for fabrication*, Vol. 5216 LNCS of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg.

Hsu, T. (2001). *MEMS and Microsystems: Design and Manufacture*, 1 edn, McGraw-Hill Science/Engineering/Math.

Hung, E. S., Yang, Y. J. & Senturia, S. D. (2002). Low-order models for fast dynamical simulation of MEMS microstructures, *Solid State Sensors*

and Actuators, 1997. TRANSDUCERS'97 Chicago., 1997 International Conference on, Vol. 2, pp. 1101–1104.

Hwang, K., Lee, K., Park, G., Lee, B., Cho, Y. & Lee, S. (2003). Robust design of a vibratory gyroscope with an unbalanced inner torsion gimbal using axiomatic design, *Journal of Micromechanics and Microengineering* **13**(1): 8–17.

IntelliSense Corp (2010). IntelliSuite.
URL: *http://www.intellisensesoftware.com/*

Kamalian, R., Agogino, A. M. & Takagi, H. (2004). The role of constraints and human interaction in evolving MEMS designs: Microresonator case study, *ASME Conference Proceedings* **2004**(46946): 875–883.

Kamalian, R. R. (2004). *Evolutionary Synthesis of MEMS*, PhD thesis, University of California, Berkeley.

Kamalian, R. R., Zhou, N. & Agogino, M. (2002). A comparison of MEMS synthesis techniques, Xiamen, China, pp. 239–242.

Koppelman, G. M. (1989). OYSTER, a three-dimensional structural simulator for microelectromechanical design, *Sensors and Actuators* **20**(1-2): 179–185.

Koza, J. (1992). *Genetic programming : on the programming of computers by means of natural selection*, MIT Press, Cambridge Mass.

L-Edit (n.d.). Tanner research.
URL: *http://www.tanner.com/Labs/mems/*

Langdon, W. B. (2000). Quadratic bloat in genetic programming, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000)*, Morgan Kaufmann, Las Vegas, pp. 451—458.

Lee, K. & Wise, K. (1982). SENSIM: a simulation program for solid-state pressure sensors, *IEEE Transactions on Electron Devices* **29**(1): 34–41.

Levenick, J. R. (1991). Inserting introns improves genetic algorithm success rate: Taking a cue from biology., *Proceedings of the Fourth International Conference on Genetic Algorithms*, Morgan Kaufmann, pp. 123—127.

Li, H. & Antonsson, E. K. (1998). Evolutionary techniques in mems synthesis.

Li, H. & Antonsson, E. K. (1999). Mask-layout synthesis through an evolutionary algorithm, *MSM'99, Modeling and Simulation of Microsystems, Semiconductors, Sensors and Actuators* .

Liu, C. (2006). *Foundations of MEMS*, Pearson/Prentice Hall.

Liu, R., Paden, B. & Turner, K. (2002). MEMS resonators that are robust to process-induced feature width variations, *Microelectromechanical Systems, Journal of* **11**(5): 505–511.

Lohn, J., Kraus, W. & Hornby, G. (2008). Automated design of a MEMS resonator, *2007 IEEE Congress on Evolutionary Computation, CEC 2007*, pp. 3486–3491.

Lu, K. & Kota, S. (2003). Design of compliant mechanisms for morphing structural shapes, *Journal of Intelligent Materials Systems and Structures* **14**(6): 379–391.

Lyshevski, S. (2002). *MEMS and NEMS : systems, devices, and structures*, CRC Press, Boca Raton Fla.

Ma, L. & Antonsson, E. K. (2001). Robust mask-layout synthesis for MEMS, *Technical Proceedings of the 2001 International Conference on Modeling and Simulation of Microsystems*, pp. 128–131.

Maluf, N., Gee, D., Petersen, K. & Kovacs, G. (1995). Medical applications of MEMS, *Proceedings of WESCON'95*, San Francisco, CA, USA, p. 300.

Markus, K. W. (2002). Developing infrastructure to mass-produce MEMS, *Computational Science & Engineering, IEEE* **4**(1): 49–54.

Maute, K. & Frangopol, D. M. (2003). Reliability-based design of MEMS mechanisms by topology optimization, *Computers & Structures* **81**(8-11): 813–824.

Miller, J. & Smith, S. (2006). Redundancy and computational efficiency in cartesian genetic programming, *Evolutionary Computation, IEEE Transactions on* **10**(2).

Mukherjee, T., Fedder, G., Ramaswamy, D. & White, J. (2000). Emerging simulation approaches for micromachined devices, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **19**(12): 1572–1589.

Mukherjee, T., Iyer, S. & Fedder, G. (1998). Optimization-based synthesis of microresonators, *Sensors and Actuators, A: Physical* **70**(1-2): 118–127.

NODAS (2010).
URL: *http://goo.gl/6a0z*

Ongkodjojo, A. & Tay, F. (2002). Global optimization and design for micro-electromechanical systems devices based on simulated annealing, *Journal of Micromechanics and Microengineering* **12**(6): 878–897.

Oxford Dictionaries (2010). "synthesis". oxford dictionaries. april 2010, http://oxforddictionaries.com/view/entry/m_en_gb0838860.
URL: *http://goo.gl/PKJn*

Parkinson, A. & Wilson, M. (1988). Development of hybrid GRG-SQP algorithm for constrained nonlinear programming, *Journal of Mechanics, Transmissions and Automation in Design* **110**: 308.

Parkinson, M. B., Jensen, B. D. & Roach, G. M. (2000). Optimization-based design of a fully-compliant bistable micromechanism, *Proceedings of DETC 00 ASME 2000 Design Engineering Technical Conferences and Computers and Information in Engineering Conference Baltimore*, pp. 1–7.

Puers, B., Peeters, E. & Sansen, W. (1989). CAD tools in mechanical sensor design, *Sensors and Actuators* **17**(3-4): 423–429.

Quarles, T., Pederson, D., Newton, R., Sangiovanni-Vincentelli, A. & Wayne, C. (2010). SPICE.
URL: *http://goo.gl/3rwp*

Raich, A. M. & Ghaboussi, J. (1997). Implicit representation in genetic algorithms using redundancy, *Evol. Comput.* **5**(3): 277–302.

Rebello, K. J. (2005). Applications of MEMS in surgery, *Proceedings of the IEEE* **92**(1): 43–55.

Rechenberg, I. (1994). Evolution strategy, *Computational Intelligence: Imitating Life* pp. 147–159.

Schott, J. R. (1995). *Fault tolerant design using single and multicriteria genetic algorithm optimization*, Thesis, Massachusetts Institute of Technology. Department of Aeronautics and Astronautics.

Schr\öpfer, G., King, D., Kennedy, C. & McNie, M. (2005). Advanced process emulation and circuit simulation for co-design of MEMS and CMOS devices, *Proceedings symposium on design, test, integration and packaging of MEMS/MOEMS (DTIP), Montreux, Switzerland*.

Sedivec, P. J. (2002). *Robust optimization: Design in MEMS*, PhD thesis, Citeseer.

Senturia, S. (2001). *Microsystem design*, Kluwer Academic Publishers, Boston.

Senturia, S., Harris, R., Johnson, B., Kim, S., Nabors, K., Shulman, M. & White, J. (1992). A computer-aided design system for microelectromechanical systems (MEMCAD), *Journal of Microelectromechanical Systems* **1**(1): 3–13.

Silva, M. G. D., Giasolli, R., Cunningham, S. & DeRoo, D. (2002). MEMS design for manufacturability (DFM), *Sensors Expo, Boston (USA)* .

Thierens, D. & Goldberg, D. E. (1993). Mixing in genetic algorithms, *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers Inc., pp. 38–47.

Tischulena, G. (2004). Market analysis for MEMS and microsystems III, 2005-2009, *Technical report*, Nexus Third Report.

van Rossum, G. (2010). Python.
URL: *http://www.python.org*

Vandemeer, J. E., Kranz, M. S. & Fedder, G. (1997). Nodal simulation of suspended MEMS with multiple degrees of freedom, *ASME Winter Annual Conference*.

Verilog-AMS (2010).
URL: *http://www.eda.org/verilog-ams/*

Vudathu, S. P. & Laur, R. (2007). A design methodology for the yield enhancement of MEMS designs with respect to process induced variations, *2007 Proceedings 57th Electronic Components and Technology Conference*, Sparks, NV, USA, pp. 1947–1952.

Weile, D., Michielssen, E. & Goldberg, D. (1996). Genetic algorithm design of pareto optimal broadband microwave absorbers, *IEEE Transactions on Electromagnetic Compatibility* **38**(3): 518–525.

White, J. (2004). CAD challenges in BioMEMS design, *Proceedings of the 41st annual Design Automation Conference*, p. 632.

Wineberg, M. & Oppacher, F. (1994). A representation scheme to perform program induction in a canonical genetic algorithm, *Proceedings of the International Conference on Evolutionary Computation.*, The Third Conference on Parallel Problem Solving from Nature: Parallel Problem Solving from Nature, Springer-Verlag, pp. 292–301.

Wineberg, M. & Oppacher, F. (1996). The benefits of computing with introns, *Proceedings of the First Annual Conference on Genetic Programming*, MIT Press, Stanford, California, pp. 410–415.

Xin, Z., Guangyi, S., Liang, R. & Guizhang, L. (2006). On MEMS design automation, *2007 Chinese Control Conference*, Zhangjiajie, China, pp. 774–778.

Yang, Y. & Yu, C. (2004). Extraction of heat-transfer macromodels for MEMS devices, *Journal of Micromechanics and Microengineering* **14**(4): 587–596.

Ye, W. & Mukherjee, S. (1998). Optimal shape design of three dimensional MEMS with applications to electrostatic comb drives, *American Society of Mechanical Engineers, Applied Mechanics Division, AMD* **228**: 23–30.

Yilmaz, A. S. & Wu, A. S. (2005). Preservation of genetic redundancy in the existence of developmental error and fitness assignment error, *Proceedings of the 2005 conference on Genetic and evolutionary computation*, ACM, Washington DC, USA, pp. 1317–1324.

Zha, X. F. & Du, H. (2003). Manufacturing process and material selection in concurrent collaborative design of MEMS devices, *Journal of Micromechanics and Microengineering* **13**(5): 509–522.

Zhang, Y. (2006). *MEMS Design Synthesis Based on Hybrid Evolutionary Computation*, PhD thesis, University of California, Berkeley.

Zhang, Y., Kamalian, R., Agogino, A. M. & Séquin, C. H. (2005). Hierarchical MEMS synthesis and optimization, *SPIE conference on smart structures and materials*, San Diego California, pp. 7–10.

Zhou, N. (2002). *Simulation and synthesis of MicroElectroMechanical Systems*, PhD thesis, University of California, Berkley.

Zhou, N., Agogino, A. & Pister, K. (2002). Automated design synthesis for Micro-Electro-Mechanical systems (MEMS), *Proceedings of the ASME Design Engineering Technical Conference*, Vol. 2, pp. 267–273.

Zhou, N., Clark, J. V. & Pister, K. S. J. (1998). Nodal analysis for MEMS design using SUGARv0.5, *Santa Clara CA April* pp. 6–8.

Zhou, N., Zhu, B., Agogino, A. M. & Pister, K. S. J. (2001). Evolutionary synthesis of MEMS MicroElectronicMechanical systems design, *Intelligent Engineering System through Artificial Neural Networks*, Vol. 11 of *Proceedings of the Artificial Neural Networks in Engineering (ANNIE2001)*, pp. 197–202.

Zienkiewicz, O. C. & Taylor, R. L. (2006). *The finite element method for solid and structural mechanics*, Elsevier Butterworth-Heinemann.

Zitzler, E. & Thiele, L. (1998). Multiobjective optimization using evolutionary algorithms—A comparative case study, *Parallel Problem Solving from Nature—PPSN V*, pp. 292–301.

Zitzler, E. & Thiele, L. (1999). Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach, *IEEE transactions on Evolutionary Computation* **3**(4): 257.

Appendix A

Automated Component–based MEMS synthesis

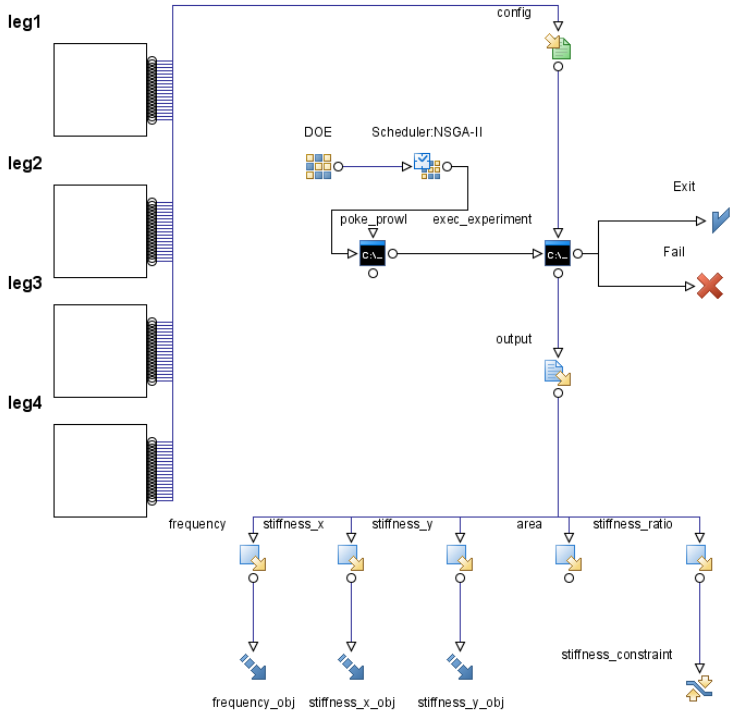
A.1 ModeFrontier workflows

This part of abstract contains the ModeFrontier(ESTECO s.r.l 2009) workflows used in the research described in chapter 4 that amongst other things contains settings of the GA itself – mutation rate, crossover probability, etc.

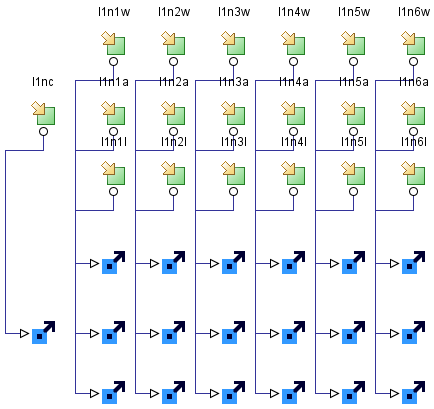
The actual design evaluation and simulation is done by custom Python (van Rossum 2010) software that accessed the component database, constructed the actual MEMS design, evaluated it in SUGAR and returned the SUGARs response estimates to the ModeFrontier(ESTECO s.r.l 2009).

The '`poke_prowl`' batch script does not implement anything usable in terms of GA it rather sends notification meaning that GA is still running. If watchdog program isn't receiving such notification for certain amount of time (that is equal to five minutes), it sends message using 'Prowl' service to the researcher stating that current GA experiment has finished and next successive experiment run can be started.

For the workflows used for the experiments described in chapter 4 please refer to the figures A.1, A.2, A.3, A.4, A.5, A.6 and A.7.



(a) Top–level workflow



(b) Variables for one resonators' leg

Figure A.1: Reference NSGA–II evolution

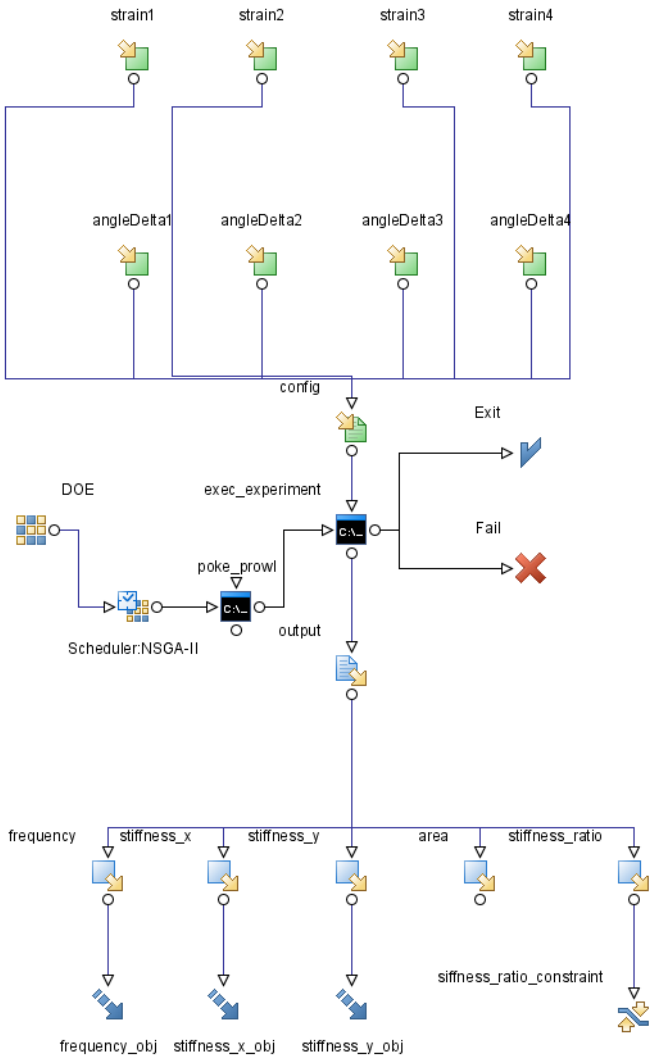


Figure A.2: Changing base angle

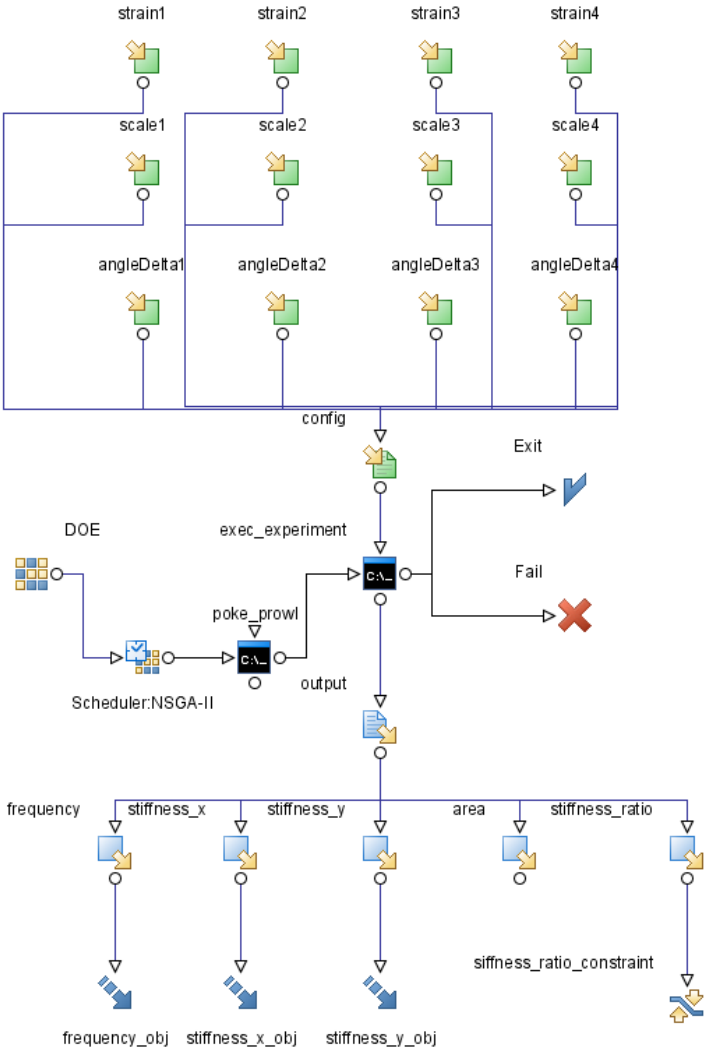


Figure A.3: Changing base angle and scaling of whole leg

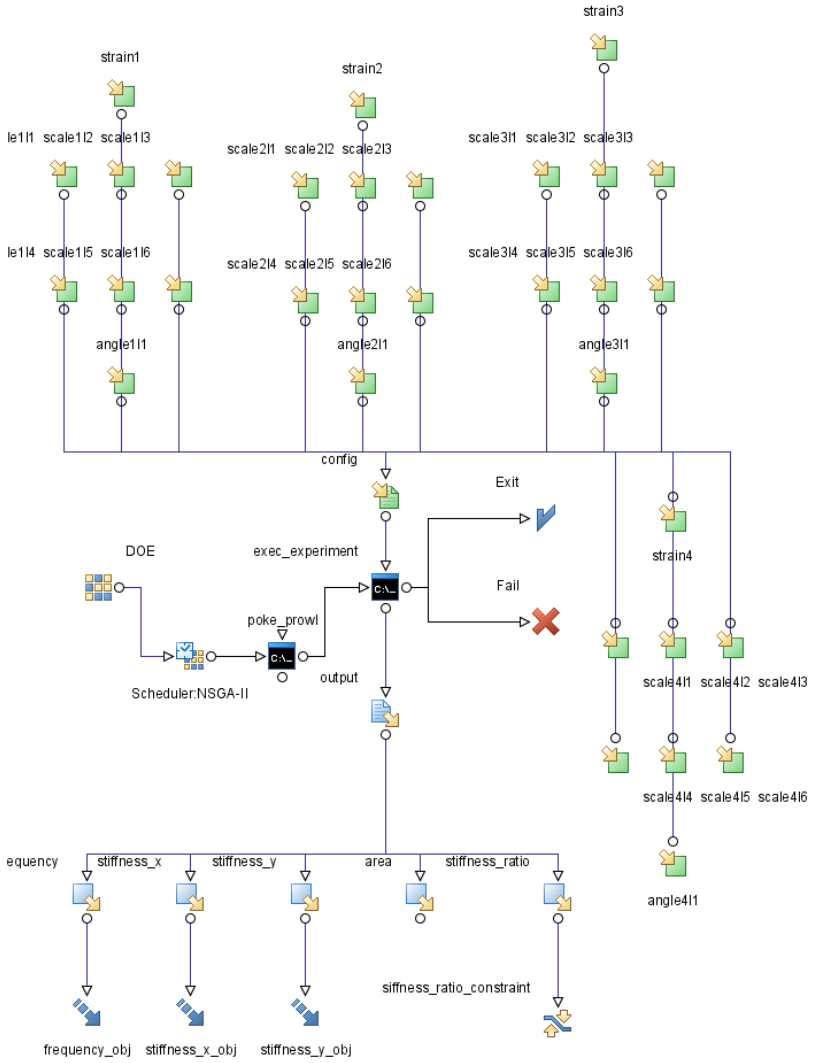


Figure A.4: Per–beam scaling and changing the base angle

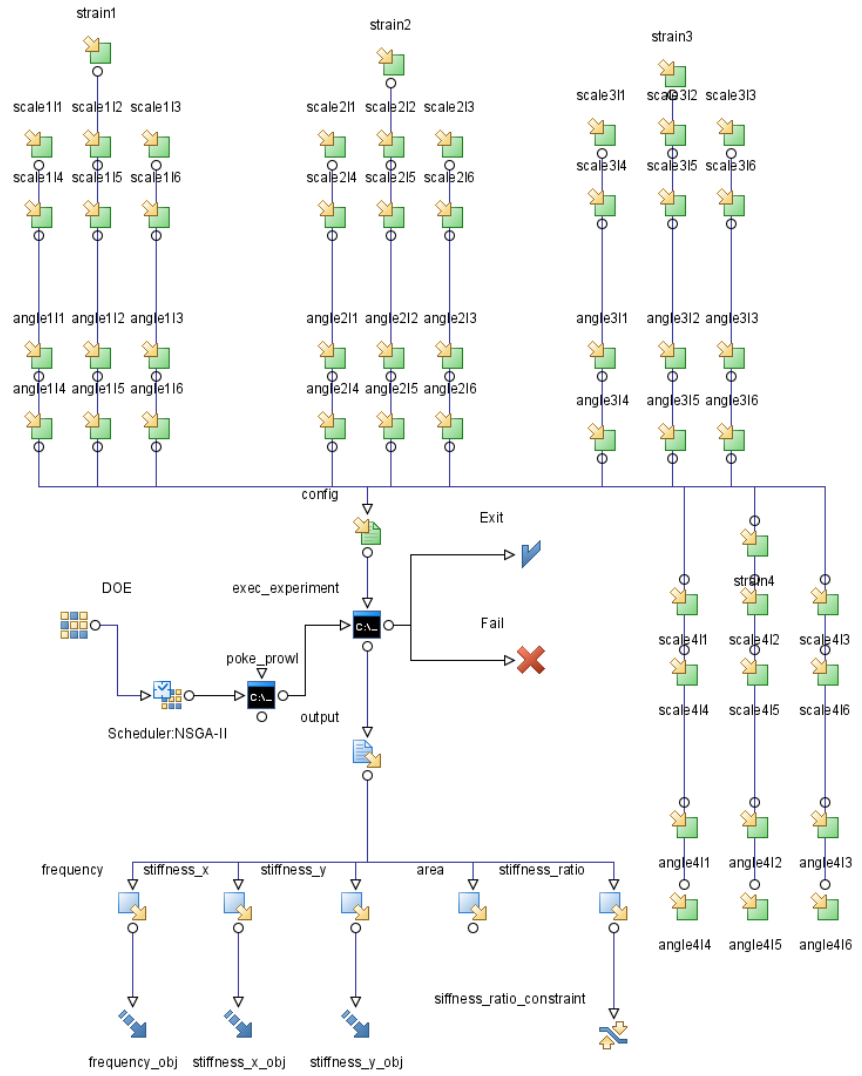


Figure A.5: Per–beam scaling and changing all angles

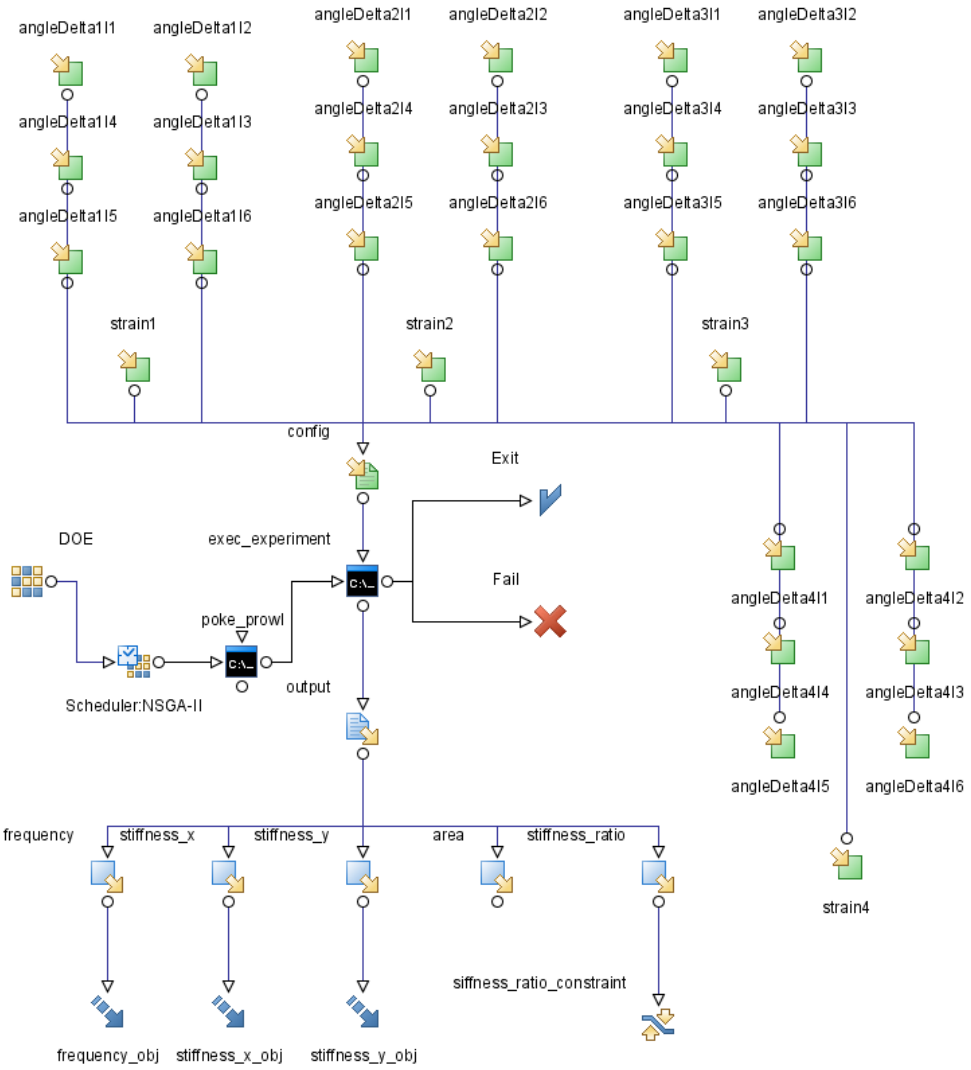


Figure A.6: Changing all angles between nodes

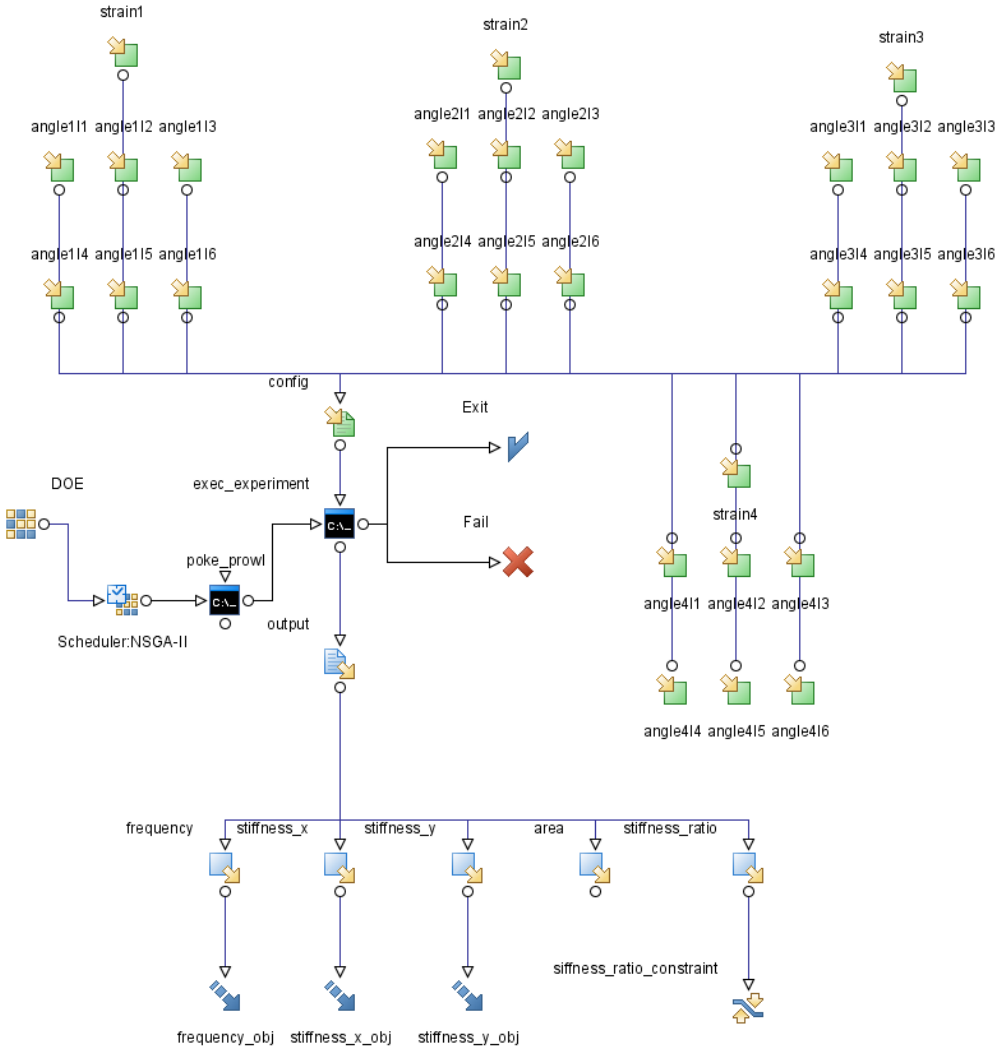


Figure A.7: Replacing all angles between nodes

Appendix B

Error–tolerant design synthesis

This appendix contains results for error–tolerant MEMS layout synthesis experiments that are described in Chapter 5.

Please refer to the tables B.1, B.2, B.3 and B.4 for per–experiment results for each of attempted experiments.

Id	Frequency (Hz)			Stiffness by X ($^N/_m$)			Stiffness by Y ($^N/_m$)		
	Ideal	μ	σ	Ideal	μ	σ	Ideal	μ	σ
0	14827	14583	518.57	1.9008	2.0584	0.5886	0.5625	0.6125	0.1822
1	14897	14648	531.42	1.9019	2.0595	0.5922	0.5617	0.6116	0.1829
2	14903	14649	539.69	1.9028	2.064	0.5955	0.562	0.6131	0.1838
3	14830	14700	489.85	1.9012	2.0631	0.586	0.5607	0.6115	0.1801
4	14839	14563	535.98	1.8982	2.0508	0.5828	0.5616	0.6098	0.18
5	14827	14582	523.66	1.9008	2.0572	0.5889	0.5625	0.6121	0.1823
6	14843	14591	521.1	1.9019	2.0521	0.586	0.5615	0.6092	0.181
7	14865	14636	502.64	1.9005	2.0435	0.5858	0.5635	0.6091	0.1817
8	14859	14631	501.8	1.9003	2.0533	0.5892	0.5635	0.6123	0.1827
9	14817	14554	522.43	1.9005	2.0441	0.5826	0.562	0.6077	0.18
10	14812	14548	532.48	1.8999	2.0644	0.5885	0.5615	0.6137	0.182
11	14877	14600	542.89	1.8928	2.0466	0.5787	0.5603	0.6091	0.1793
12	14858	14663	496.34	1.9016	2.0605	0.5835	0.5622	0.6125	0.1803
13	14908	14611	562.07	1.8976	2.0579	0.5898	0.5628	0.6137	0.1827
14	14853	14613	522.5	1.9016	2.0578	0.5937	0.5622	0.6118	0.1835
15	14827	14539	544.76	1.8986	2.0453	0.5799	0.5605	0.6071	0.179
16	14827	14548	539.37	1.8974	2.046	0.583	0.5604	0.6075	0.18
17	14812	14547	530.74	1.8999	2.0569	0.5873	0.5615	0.6113	0.1816
18	14883	14614	536.87	1.8951	2.055	0.5831	0.5607	0.6113	0.1804
19	14877	14596	547.46	1.8928	2.052	0.5876	0.5603	0.6108	0.182
20	14835	14681	405.65	1.9017	1.9809	0.3853	0.5622	0.5888	0.1183
Avg.	14851	14605	521.35	1.8994	2.0509	0.5771	0.5617	0.6098	0.1784

Table B.1: Non–tolerant designs

Id	Frequency (Hz)			Stiffness by X ($^N/_m$)			Stiffness by Y ($^N/_m$)		
	Ideal	μ	σ	Ideal	μ	σ	Ideal	μ	σ
0	14010	13842	427.92	1.8398	1.865	0.3179	0.5984	0.616	0.1171
1	14010	13852	435.47	1.8398	1.868	0.3156	0.5984	0.618	0.1168
2	14415	14304	348.06	1.9052	2.0067	0.4388	0.5963	0.6156	0.1138
3	14439	14325	345.95	1.9133	2.0141	0.4374	0.5923	0.6106	0.1129
4	13774	13671	312.05	1.9678	2.0383	0.4122	0.56	0.5714	0.1005
5	14537	14425	355.33	1.9722	2.0993	0.5061	0.5952	0.6152	0.115
6	13849	13741	347.39	1.933	1.9893	0.4356	0.5588	0.5707	0.1036
7	14796	14686	358.89	1.974	2.0976	0.4666	0.6	0.6213	0.1148
8	14209	14104	330.68	1.8489	1.8592	0.3623	0.5731	0.5893	0.1127
9	14636	14520	340.84	1.8224	1.8944	0.41	0.5807	0.594	0.1071
10	13495	13361	326.79	1.8998	1.9027	0.4281	0.5613	0.5659	0.1043
11	14289	14183	333.73	1.9584	2.0229	0.4074	0.6052	0.6188	0.1095
12	14739	14623	358.38	2.027	2.1611	0.5216	0.5859	0.6077	0.1162
13	14739	14628	360.91	2.027	2.1643	0.5212	0.5859	0.6106	0.1168
14	14812	14667	420.63	1.9503	2.033	0.4036	0.6105	0.6248	0.1405
15	14456	14322	384.02	1.8621	1.9321	0.3663	0.6062	0.619	0.1414
16	14345	14207	391.84	1.8791	1.9546	0.3635	0.5857	0.6016	0.133
17	14311	14184	385.33	1.8645	1.9336	0.347	0.5826	0.5982	0.1305
18	14577	14438	383.67	1.889	1.9267	0.366	0.5538	0.5669	0.1348
19	13993	13860	362.48	1.8785	1.9388	0.3595	0.5747	0.586	0.1289
20	14456	14318	380.6	1.8621	1.9291	0.364	0.6062	0.6192	0.1399
Avg.	14328	14203	366.24	1.9102	1.9824	0.4072	0.5862	0.602	0.1195

Table B.2: Robust optimisation–inspired error tolerance

Id	Frequency (Hz)			Stiffness by X ($^N/_m$)			Stiffness by Y ($^N/_m$)		
	Ideal	μ	σ	Ideal	μ	σ	Ideal	μ	σ
0	14724	14583	471.74	1.8939	1.9233	0.3506	0.5986	0.605	0.1061
1	14167	14032	476.44	1.9686	2.0262	0.3805	0.5787	0.5908	0.1043
2	14433	14295	461.4	1.8635	1.8966	0.3486	0.5919	0.5992	0.1058
3	14467	14323	458.32	1.9702	1.9869	0.3371	0.5758	0.5786	0.0975
4	14224	14086	447.2	1.9093	1.9436	0.3502	0.5909	0.5979	0.1036
5	14713	14567	483.44	1.8656	1.906	0.3516	0.5904	0.5997	0.1071
6	14323	14188	463.34	1.8988	1.9375	0.344	0.5868	0.5956	0.1019
7	14589	14456	486.57	1.8725	1.9246	0.3496	0.5925	0.6059	0.1071
8	13899	13769	451.23	1.9437	1.98	0.3362	0.5596	0.5676	0.0947
9	14378	14237	458.61	1.8997	1.9363	0.3513	0.5825	0.5907	0.1033
10	14175	14041	449.72	1.859	1.8875	0.3284	0.5727	0.5793	0.0986
11	13969	13846	436.65	1.886	1.9259	0.3345	0.5625	0.5723	0.0978
12	14378	14244	458.11	1.8904	1.9313	0.3541	0.579	0.5886	0.1043
13	14467	14324	463.4	1.9702	1.9903	0.3387	0.5758	0.5797	0.098
14	14253	14087	370.27	1.8638	1.9623	0.4995	0.5898	0.607	0.1271
15	14253	14081	371.7	1.8638	1.9713	0.5279	0.5898	0.6042	0.1282
16	14168	14010	465.34	1.8775	1.9332	0.3755	0.5773	0.586	0.1111
17	14210	14052	444.08	1.8713	1.9197	0.3694	0.5885	0.5951	0.1119
18	14141	13989	464.89	1.8763	1.9325	0.3766	0.5771	0.5867	0.112
19	14168	14011	464.55	1.8775	1.9319	0.3756	0.5773	0.5866	0.1117
20	14597	14441	438.24	1.9185	1.981	0.3674	0.5942	0.6079	0.1164
Avg.	14319	14174	451.68	1.8971	1.9442	0.3689	0.5825	0.5916	0.1071

Table B.3: Response–dependant error tolerance

Id	Frequency (Hz)			Stiffness by X ($^N/_m$)			Stiffness by Y ($^N/_m$)		
	Ideal	μ	σ	Ideal	μ	σ	Ideal	μ	σ
0	11986	11906	231.21	1.4311	1.5127	0.3461	0.635	0.6712	0.1485
1	11196	11127	198.2	1.5487	1.6281	0.3376	0.596	0.6273	0.1267
2	11373	11293	223.63	1.4823	1.5546	0.3434	0.6326	0.6644	0.1408
3	11354	11286	198.3	1.6123	1.7097	0.3811	0.6681	0.7101	0.1575
4	11910	11841	199.58	1.5327	1.6028	0.3239	0.6696	0.6998	0.1338
5	10894	10822	207.15	1.5095	1.5832	0.3293	0.5889	0.6181	0.1242
6	11484	11405	227.99	1.4223	1.508	0.3436	0.6071	0.6434	0.1409
7	11410	11331	217.56	1.4795	1.5581	0.3411	0.6177	0.6493	0.1343
8	10830	10766	192.78	1.8524	1.9413	0.362	0.5709	0.5995	0.1103
9	11424	11352	222.94	1.7081	1.8037	0.3715	0.6626	0.6989	0.1374
10	11511	11441	202.63	1.6096	1.7032	0.3845	0.6889	0.7309	0.1652
11	11529	11444	239.42	1.4181	1.4989	0.3522	0.62	0.655	0.1492
12	10409	10344	169.76	1.8704	1.9051	0.264	0.6443	0.6561	0.0915
13	10993	10917	227.96	1.5619	1.6503	0.3511	0.6062	0.6405	0.1324
14	10481	10418	182.05	1.7627	1.839	0.3441	0.6499	0.6779	0.1227
15	11749	11669	244.81	1.8282	1.9148	0.3865	0.6851	0.7175	0.1372
16	12013	11931	246.76	1.6854	1.7657	0.3495	0.6793	0.7132	0.1393
17	11228	11159	214.79	1.5883	1.6827	0.3472	0.6235	0.6604	0.1318
18	10437	10375	189.66	1.7586	1.8416	0.35	0.6005	0.6292	0.1168
19	11582	11498	234.59	1.5386	1.6204	0.3545	0.628	0.6612	0.1386
20	11978	11898	232.28	1.6671	1.7531	0.3551	0.6753	0.7104	0.1385
Avg.	11323	11249	214.48	1.6128	1.6941	0.3485	0.6357	0.6683	0.1342

Table B.4: Error tolerance in separate objective

Appendix C

Miscellaneous simulation settings

C.1 Material properties

The numerical simulation require definition of properties of material that is used for particular MEMS design. As the material strongly influences the estimated responses, it was decided to prepare a list used settings in this appendix for results to be easily reproduced. Please refer to the Table C.1 for the physical environment definitions and to the listing C.1 for the SUGAR code used to define given environment properties in performed experiments.

Listing C.1: Definition of material 'p1' used for experiments in SUGAR syntax

```
poly = material {
        Poisson = 0.3,
        thermcond = 2.33,
        viscosity = 1.78e-5,
        fluid = 2e-6,
        density = 2300,
        Youngsmodulus = 165e9,
        permittivity = 8.854e-12,
        sheetresistance = 20,
```

Physical properties of material (MUMPS) used for MEMS	
Ambient temperature	$30°C$
Poisson's Ratio	0.3
Air viscosity	$1.78 \times 10^{-5}\ {}^{kg}/_{m \cdot s}$
Gap between the device and the substrate	$2\ \mu m$
Layer thickness	$1.78\ \mu m$
Material density	$2.3 \times 10^{3}\ {}^{kg}/_{m^3}$
Permittivity	$8.854 \times 10^{-12}\ {}^{C^2}/_{\mu N \cdot \mu m^2}$
Sheet resistance	$20\ {}^{\Omega}/_{\square}$
Thermal conductivity	$2.33\ {}^{10^{-6}}/_{C}$
Thermal expansion	$2.3\ {}^{10^{-6}}/_{C}$
Young's modulus	$1.65 \times 10^{11}\ {}^{N}/_{m^2}$

Table C.1: Physical environement properties used in experiments

```
        stress = 0,
11      straingradient = 0,
        thermalexpansion = 2.3e-6,
13      ambienttemperature = 30
}
15
p1 = material {
17      parent = poly,
        h = 1.78e-6,
19      Poisson = 0.3,
        thermcond = 2.33,
21      viscosity = 1.78e-5,
        fluid = 2e-6,
23      density = 2300,
        Youngsmodulus = 165e9,
25      permittivity = 8.854e-12,
        sheetresistance = 20,
27      stress = 0,
        straingradient = 0,
29      thermalexpansion = 2.3e-6,
        ambienttemperature = 30
31 }
```

C.2 I–shaped mass

As SUGAR source code for I–shaped mass that is used for flexure resonator is not easily accessible, it is listed (listing C.2) here for convenience of future researchers. The dimensions $l1, l2, w1, w2$ define the dimensions of given mass as denoted on Figure C.1.

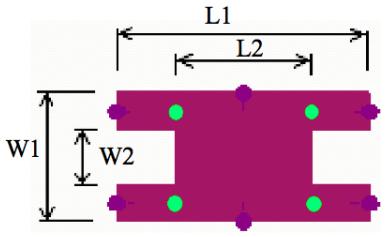


Image from (Zhang 2006, page 49)

Figure C.1: I–shaped mass dimensions

Listing C.2: Definition of I–shaped mass in SUGAR syntax

```
1 -- comb{1,2} are comb connection points
  -- l{1,2} and w{1,2} are mass dimensions as defined in
3 subnet i_shaped_mass (
          leg1, leg2, leg3, leg4,
5         comb1, comb2, material,
          l1, l2, w1, w2, oz
7 )
          local parent = material
9
          _currnodes["leg1"] = leg1
11        _currnodes["leg2"] = leg2
          _currnodes["leg3"] = leg3
13        _currnodes["leg4"] = leg4

15        _currnodes["grill1"] = comb1
          _currnodes["grill2"] = comb2
17
          leg_base_width = (w1 - w2) / 2
```

```
19          -- length is two times bigger because nodes are
                connected in center
            leg_base_length = (l1 - l2)
21
            myOz = oz or 0
23
            mfbeam3d {
25                  _n("grill1"), _n("top_grill_to_mass");
                    material=parent,
27                  l=leg_base_width/2,
                    w=l2, oz=myOz
29          }
            mfbeam3d {
31                  _n("top_grill_to_mass"), _n("leg1");
                    material=parent,
33                  l=leg_base_length,
                    w=leg_base_width, oz=myOz + deg(90)
35          }
            mfbeam3d {
37                  _n("top_grill_to_mass"), _n("leg2");
                    material=parent,
39                  l=leg_base_length,
                    w=leg_base_width, oz=myOz - deg(90)
41          }
            mfbeam3d {
43                  _n("top_grill_to_mass"),
                    _n("bottom_grill_to_mass");
45
                    material=parent,
47                  l=w2 + leg_base_width,
                    w=l2, oz=myOz
49          }
            mfbeam3d {
51                  _n("bottom_grill_to_mass"), _n("leg3");
                    material=parent,
```

```
53                  l=leg_base_length,
                    w=leg_base_width, oz=myOz + deg(90)
55          }
            mfbeam3d {
57                  _n("bottom_grill_to_mass"), _n("leg4");
                    material=parent,
59                  l=leg_base_length,
                    w=leg_base_width, oz=myOz - deg(90)
61          }
            mfbeam3d {
63                  _n("bottom_grill_to_mass"), _n("grill2");
                    material=parent,
65                  l=leg_base_width/2,
                    w=l2, oz=myOz
67          }
    end
```