Analytical modelling and optimization of a thermal convective microfluidic gyroscope

by

Surika Vosloo

Thesis presented at the University of Stellenbosch in partial fulfilment of the requirements for the degree of

Master of Science in Mechanical Engineering

Department of Mechanical and Mechatronical Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Study leader: Prof A.A. Groenwold

March 2010

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: .....

S. Vosloo

Date: .....

Abstract

Analytical modelling and optimization of a thermal convective microfluidic gyroscope

S. Vosloo

Department of Mechanical and Mechatronical Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Thesis: MScEng (Mech)

March 2010

This thesis deals with the mathematical optimization of the detecting chamber of a thermal convective microfluidic gyroscope and the comparison of several different optimization strategies.

An analytical model is developed for the gyroscope and some design considerations are discussed. Sequential approximate optimization strategies are explained and compared to each other by implementing test problems from the literature. The optimization problem is formulated from the analytical model and implemented using the different optimization strategies. Results are presented and compared to find the most effective optimization strategy.

A sequential approximate optimization algorithm is implemented in MATLAB and tested using the gyroscope design problem and common test problems from the literature. Results and iteration history are compared with an existing FORTRAN implementation.

Uittreksel

Analitiese modelering en optimering van n termies-konvektiewe mikrovloeier giroskoop

("Analytical modelling and optimization of a thermal convective microfluidic gyroscope")

S. Vosloo

Departement Meganiese en Megatroniese Ingenieurswese
Universiteit van Stellenbosch
Privaatsak X1, 7602 Matieland, Suid Afrika

Tesis: MScIng (Meg)

Maart 2010

Hierdie tesis handel oor die wiskundige optimering van die deteksiekamer van n termies-konvektiewe mikrovloeier giroskoop en die vergelyking van verskeie optimeringsstrategieë.

'n Analitiese model is opgestel vir die giroskoop en verskeie ontwerpsoorwegings word bespreek. Sekwensiëel benaderde optimeringsstrategieë word bespreek en met mekaar vergelyk, deur dit op toetsprobleme uit die literatuur toe te pas. Die optimeringsprobleem is geformuleer uit die analitiese model en geimplementeer deur gebruik te maak van verskeie optimeringsstrategieë. Resultate word getoon en vergelyk, om die mees effektiewe optimeringsstrategie te vind.

'n Algoritme vir sekwensiëel benaderde optimeringsprobleme is in MATLAB geimplementeer. Die giroskoop probleem, asook probleme uit die literatuur, is gebruik om resultate en iterasie geskiedenis te vergelyk met 'n bestaande FORTRAN implementasie.

Acknowledgements

I would like to express my sincere gratitude to those who have contributed to making this work possible:

- The management of Denel Dynamics (Pty) Ltd for making time and funds available and also for permission to publish the research results.
- Professor Albert A. Groenwold of the University of Stellenbosch as my study leader.

Dedications

Hierdie tesis word opgedra aan my ouers,
Jan en Karien Vosloo,
vir hul ondersteuning, liefde en geduld.

Contents

List of Figures

List of Tables

Nomenclature

Constants:

$g =$	9,81 m/s^2 Gravitational acceleration

Variables:

x	x-coordinate
y	y-coordinate
z	z-coordinate
θ	Rotation Angle
ω	Angular velocity
r	Radius
δ	Deflection
Kn	Knudsen number
Re	Reynolds number
Ma	Mach number
Nu	Nusselt number
Pr	Prandtl number
ν	Kinematic viscosity
μ	Dynamic viscosity

ρ	Density
k	Thermal conductivity
α	Thermal coefficient of resistance
h	Heat transfer coefficient
c_p	Specific heat at constant pressure
c_v	Specific heat at constant viscosity
γ	Specific heat ratio
τ	Time constant
λ_r	Resistivity
Q	Volumetric flow rate
p	Pressure
T	Temperature
U	Velocity
v_m	Mean velocity
L	Length
A	Area
$\mathcal{V}$	Volume
u	Velocity in x direction
v	Velocity in y direction
w	Velocity in z direction
t	Time
V	Voltage
R	Resistance

I	Current
P	Power
C	Capacitance
F	Force
m	Mass
a	Acceleration
σ	Stress
$\mathscr{I}$	Moment of Inertia
a_0	Speed of sound
C_D	Drag coefficient
C_f	Friction coefficient
$\mathscr{S}$	Sensitivity
$\mathscr{L}$	Lagrangian

Vectors and Tensors:

$\vec{v}$	Velocity vector
$\vec{a}$	Acceleration vector
$\vec{g}$	Gravitational acceleration vector
$\vec{\Omega}$	Angular velocity vector

Introduction

Inertial sensors, such as gyroscopes and accelerometers, provide us with the possibility of an automated living and working environment. Due to the growing demand for higher accuracy and lower cost microscale sensors, the importance of research, development and fabrication has increased significantly.

Inertial sensors are intensively used in inertial navigation systems (INS) for space, military or automotive applications. Given a certain initial position and single degree of freedom, an accelerometer can be used to determine the acceleration, velocity and position of a moving object at any given time. Three accelerometers orthogonal to each other, provides for three degrees of freedom. Adding a triaxial gyroscope to our system, enables us to determine the exact position of a moving object with six degrees of freedom, at any given time.

Replacing macroscale sensors with microelectromechanical systems (MEMS) is preferable in most applications, as major size and cost reductions are possible. Sufficient accuracy can often be achieved using MEMS gyroscopes and accelerometers to perform the same duties as their macroscale cousins.

The goal of this thesis is to provide the reader with the basic knowledge to make informed design decisions regarding the design process of a microscale gyroscope and more specifically, a thermal convective microfluidic gyroscope. Important literature is reviewed to give the reader some background information on MEMS, its fabrication processes and available microscale gyroscope concepts. Sequential approximate optimization strategies, for the thermal convective microfluidic gyroscope, are investigated and then compared. The optimization problem is formulated using an analytical model defined specifically for this gyroscope concept. Results are presented and discussed to determine the most effective optimization strategy for our gyroscope problem.

The thermal convective microfluidic gyroscope research field is still largely unexplored and the development of a standardized design process is an important contribution. This thesis presents the reader with an analytical model and optimization strategies, concerning this gyroscope, hoping that we have advanced a step closer to standardization.

Chapter 1

Literature review

1.1 Microelectromechanical systems

Microelectromechanical systems, also known by its acronym *MEMS*, are small mechanical structures that the human eye struggles to see, with some electrical interaction to either actuate it, sense it, or both. This integration of mechanics and electronics on a silicon substrate, through means of microfabrication, is used to create microscale sensors and actuators.

MEMS have become increasingly popular in the last ten years and is expected to become an estimated $15.5 billion industry by the year 2012 [27]. Microsensors ready for batch processing can, for example, achieve much lower unit cost prices than macrosensors and still maintain a sufficient level of accuracy. This is one of the main advantages of microsystems and it is one that is very hard to compete with. Another would be the size of microdevices, as it can easily be integrated with microelectronics and reduce the overall size and cost of any automation or control system.

1.2 Microfabrication

An essential study of current fabrication techniques was done to fully understand the design limitations associated with micro structures. Available materials and their properties also play an important role in designing for fabrication. This chapter will briefly discuss the necessary facts to understand the geometrical, chemical and electrical possibilities and limitations within the fabrication process.

1.2.1 Fabrication materials

The materials used to fabricate microdevices are limited by the fabrication techniques that are currently available. Inorganic materials, such as silicon, silicon dioxide, silicon nitride, aluminium and tungsten

are the most commonly used in microdevices, although certain organic polymers are also popular [28]. Silicon wafers remain the most popular choice for MEMS design, as it is a cheap, easily obtained material for which a number of standard fabrication techniques are already in place [4].

1.2.2 Techniques

Most of the techniques used to manufacture MEMS were adopted from the microelectronic fabrication technologies whilst other techniques are specialized for MEMS fabrication. This chapter only gives a brief overview as to what techniques are currently available and how each of them are used. For a more in depth understanding of the technologies and how to design for the MEMS fabrication process *Fundamentals of Microfabrication* [22] and *Microsystem Technology* [24] will prove most helpful.

Microfabrication techniques can be classified as either *surface micromachining* or *bulk micromachining*. Bulk micromachining can be described as the process used to etch deeply into substrates, whereas surface micromachining removes sacrificial layers from beneath thin-film structures to leave free-standing mechanical structures [28]. One should rather not try and distinguish between surface and bulk micromachining when classifying the individual techniques as most techniques can be of either type.

1.2.2.1 Wafer cleaning

Any wafer subjected to high temperature processes during microfabrication must be cleaned first. According to Senturia [28] the standard set of wafer cleaning steps is called the *RCA cleans*. The RCA cleans include the following steps:

- remove organic coatings in a strong oxidant (usually a 7:3 mixture of concentrated sulfuric acid and hydrogen peroxide)
- remove organic residues (usually in a 5:1:1 mixture of water, hydrogen peroxide and ammonium hydroxide)
- previous step can grow thin oxide layer and a HF etchant must be inserted in the dilute to remove unintentional oxide layers
- remove ionic contaminants (usually in a 6:1:1 mixture of water, hydrochloric acid and hydrogen peroxide)

1.2.2.2 Oxidation

According to Senturia [28] a high quality oxide can be thermally grown on the surface of a silicon wafer; this being one of its many great advantages. Silicon is brought in contact with molecules of oxygen

under very high temperatures (850°C to 1150°C) to form silicon dioxide. The oxygen molecules must diffuse through the already formed oxide layer to reach the silicon surface. The oxidation rate therefore depends on the rate of diffusion and decreases as the oxide layer thickens.

1.2.2.3 Doping

Doping is a process used to modify the electrical properties of a semiconductor by adding impurities to certain areas in the material. A dopant with one valence electron more than the substrate material is called a donor (n-type dopant) and donates an electron to the semiconductor crystal. The mobile charge carriers are called *electrons* and behave like negatively charged species. A dopant with one valence electron less than the substrate material is called an acceptor (p-type dopant) that receives an electron from the semiconductor crystal. These mobile charge carriers are called *holes* and behave like positively charged species [22] . A group-IV semiconductor such as silicon will typically be doped with atoms from group-III (boron) for p-type region and atoms from group-V (phosphorus) for n-type region.

Counter-doping is a process used when an already doped region is doped with an opposite type dopant to change the conductivity of that region. This process is critical for manufacturing of MEMS devices [28].

Doping consists of two important steps, *ion implantation* and *drive-in diffusion*.

Ion implantation

Senturia [28] explains that ion implantation is a process in which energetic dopant atoms are directly shot into the wafer by means of a particle accelerator. The acceleration energy and beam current is used to accurately control the depth and dopant concentration.

Photoresist and oxides can be used as implantation masks to selectively dope surface regions. The projected range in a photoresist implant mask will be $2-3$ times the projected range in silicon [28]. The thickness of the photoresist therefore needs to be at least three times the projected range in silicon.

Electrical conductivity is reduced due to crystal damage caused by the ion beams. Most of the damage can be eliminated by using a high temperature (700°C - 1000°C) annealing process [22].

Drive-in diffusion

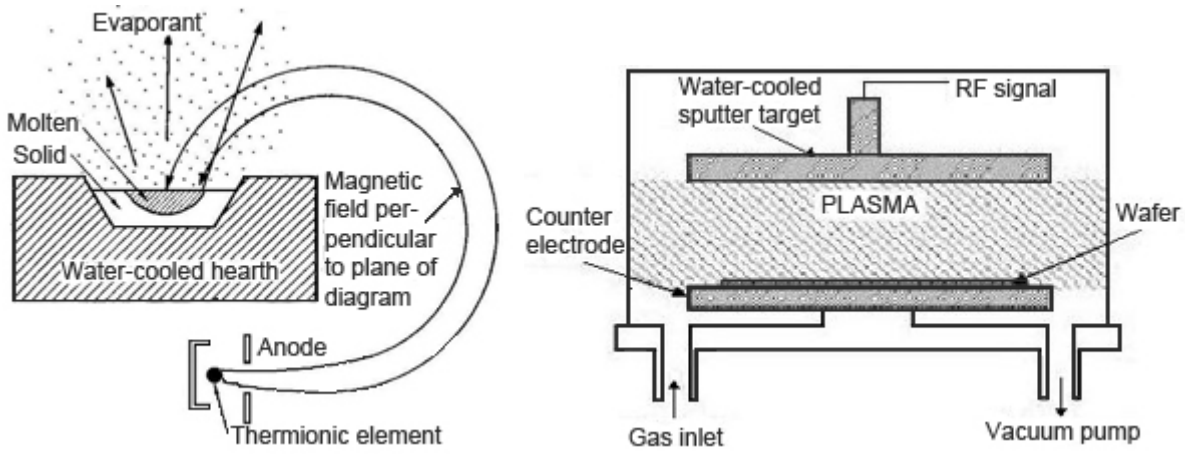
Dopant layers on the surface of the wafer is redistributed using high temperature drive-in anneals. To prevent dopant evaporation at high temperatures, a protective oxide layer is used [28]. Diffusion is the redistribution of dopants from a high concentration to a low concentration.

1.2.2.4 Thin film deposition

The deposition and patterning of a thin film of material is used in most microfabrication processes. A great variety of additive methods can be used for thin film deposition and the most commonly used ones are explained in this section.

Physical vapor deposition

Physical vapor deposition (PVD) applies one of two methods, *evaporation* and *sputtering*. The evaporation process consists of a heat source used to evaporate surface atoms from a source material (usually metals). This process is done under high-vacuum conditions and deposit the evaporated atoms onto the substrate [22]. Several heat sources can be used for evaporation of the source material: an incident electron beam (e-beam) as seen in Figure 1.1a, a resistance heat source, radio frequency (RF) inductance or laser source. The e-beam uses a magnetic field that is targeted on the source material contained in a water-cooled hearth.



(a) Thin film deposition by evaporation using an e-beam evaporation setup

(b) Sputtering process with RF power source

Figure 1.1: Physical vapor deposition

(Figures adapted from [22] and [23])

Sputtering occurs in a low-pressure gas environment and uses a glow discharge to ionize chemically inert atoms. An electric field accelerates these positive ions into the negatively charged target material (the material to be deposited). Atoms from the target material are knocked out (sputtering process) and allowed to reach the substrate material connected to an anode [28]. Figure 1.1b shows the sputtering process using a RF power source to ionize the atoms.

Chemical vapor deposition

Chemical vapor deposition (CVD) uses a heated furnace to which many gases are supplied. A chemical reaction occurs on the surface of the wafers positioned in the furnace to deposit a thin film of material. Deposition may occur on both sides of the wafers as shown in CVD setup in Figure 1.2.

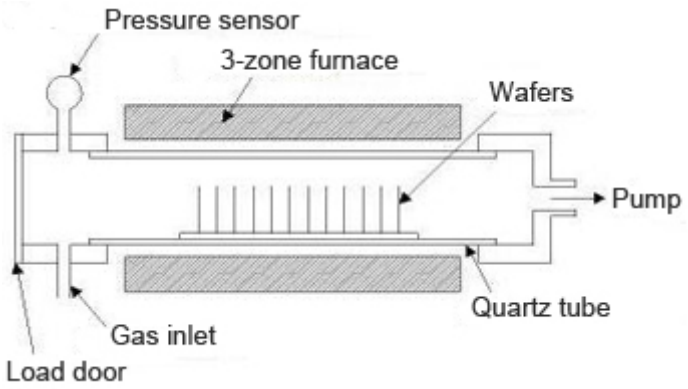


Figure 1.2: Low pressure CVD furnace with vertically positioned wafers

(Figure adapted from Memsnet [23])

Electrodeposition

Electrodeposition (also called electroplating) is an electrochemical process designed to deposit metal ions from a solution onto a substrate. The substrate is immersed in an electrolyte and an electrical potential is applied across the conductive substrate and the counter electrode as shown in Figure 1.3. A redox reaction occurs at the surface of the substrate, resulting in the deposition of a thin layer of atoms. Oxidation occurs at the counter electrode and usually forms small gas bubbles.

The plating uniformity depends on the uniformity of the applied current density [28]. The rates of electroplating differs at the corner regions and for features with different surface areas.

Casting

Casting is a simple process in which the material that needs to be deposited is dissolved in a liquid solvent. The solution can be applied to a substrate by one of two methods, *spinning* or *spraying*.

In spin casting, the solution is applied to a wafer and spun at a high rotational velocity. Figure 1.4 shows the wafer with the applied solution before and after spinning. Some of the solvent is evaporated during spinning. Baking evaporates the remaining solvent and only a thin film of the original material remains [28].

In spray casting the solution is uniformly sprayed onto the wafer. A baking process is used to evaporate the solvent so only a thin uniform layer of material remains.

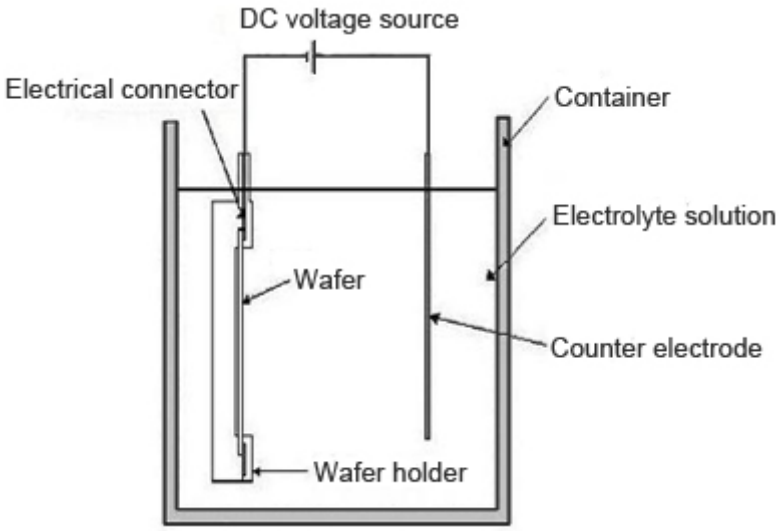


Figure 1.3: Electrodeposition setup

(Figure adapted from Memsnet [23])

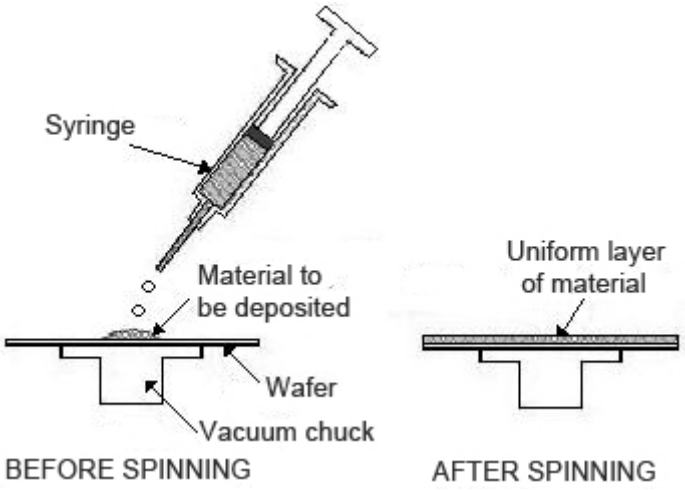


Figure 1.4: Spin casting used for photoresist in lithography

(Figure adapted from Memsnet [23])

Sol-gel deposition

In the sol-gel deposition process a *sol* formation is formed by immersing a suitable chemical precursor in a liquid solution [22]. The sol formation is then transform into gelatinous network or *gel* formation by hydrolysis and condensation, using hydrochloric acid as catalyst. The material, in its gel state, is then applied to a substrate by spinning, spraying or dipping. A sintering process transforms the gelatinous layer into a glass-like layer and then silicon dioxide by densification.

1.2.2.5 Pattern transfer

Pattern transfer consists of two parts: the first is a photo-process whereby a wafer is coated with a photosensitive film and the pattern transferred photographically; the second is either a physical or chemical process whereby material is added (additive process) or removed (subtractive process) to create the pattern [28]. This section describes the different additive and subtractive processes used for pattern transferring.

Optical lithography

Optical lithography is a technique used to transfer a pattern from a mask to a photosensitive material. The photosensitive material is called photoresist and changes its physical properties when exposed to a radiation source [28]. The mask contains clear and opaque regions to selectively expose the resist to this radiation source and change its resistance to a certain chemical solution, called the developer solution. The developer solution will either etch away the exposed photoresist (called positive resist) or the unexposed photoresist (called negative resist) as seen in Figure 1.5.

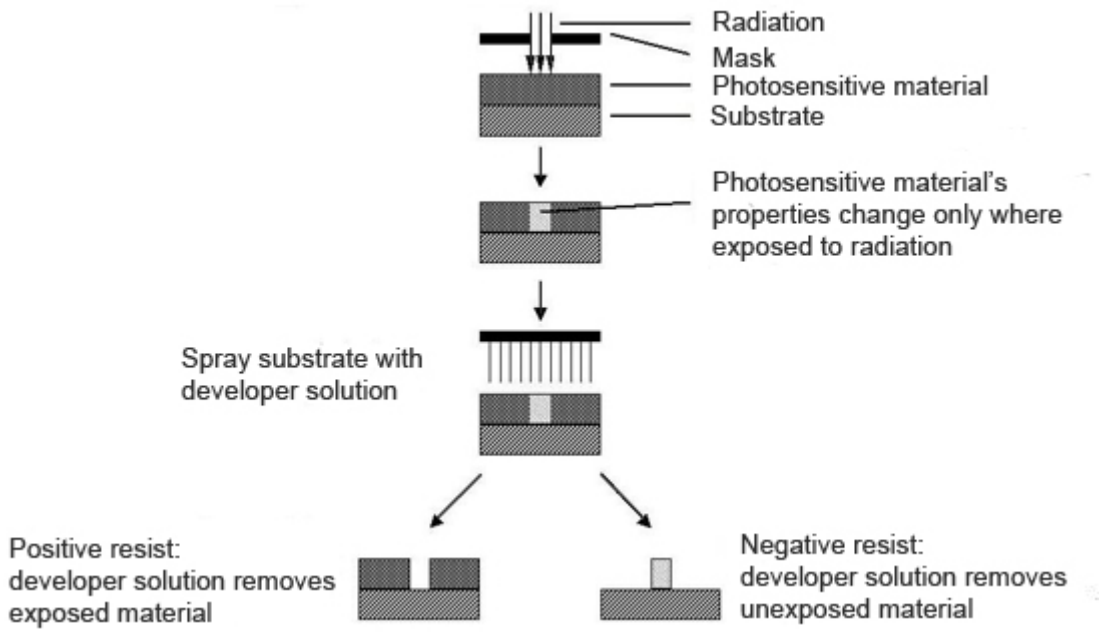


Figure 1.5: Pattern transfer illustrated with positive and negative photoresist

(Figure adapted from Memsnet [23])

The photoresist is not a permanent feature, but only a way to transfer a pattern onto an oxidized substrate as seen in Figure 1.6. The remaining photoresist can be compared to a stencil used to draw a picture. This stencil can be used to deposit material on the part of the substrate that is not covered by the photoresist. This is an additive process called lift-off. Another possibility is to use an etchant to remove

the material that is not covered by the photoresist. This is a subtractive process called etching. Etching and lift-off will be explained in the sections that follow. Once the pattern is transferred, the photoresist is stripped and only the intended pattern remains.

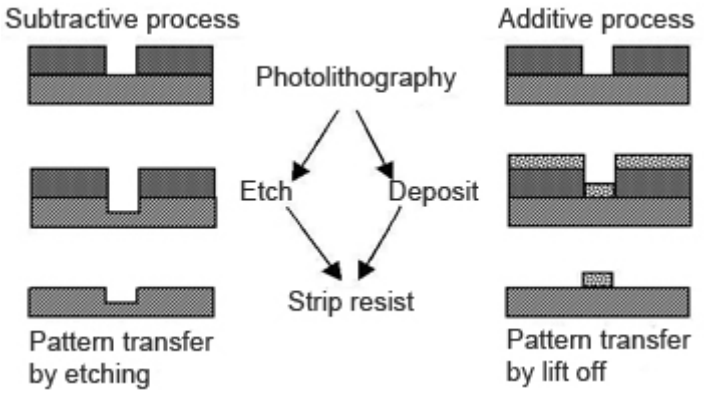


Figure 1.6: Transferring a pattern from the photoresist by etching and lift-off

(Figure adapted from Memsnet [23])

Wet etching

Wet etching is a process where a material is chemically dissolved when it comes in contact with an etchant. A photoresist patterned substrate is typically immersed in a container filled with a suitable etchant. The patterned photoresist serves as a masking layer to ensure selective etching. It is therefore important to use a photoresist (or other masking layer) that does not dissolve when immersed in the etchant, or dissolves much slower than the material to be etched.

The rate of etching and the shape of the resulting etched feature depends on many factors: the type of substrate, the specific chemistry of the etchant, the choice of masking layer, the temperature, and whether or not the solution is well stirred [28].

Isotropic wet etching

As the rate of isotropic wet etching is not orientation-dependant, a typically etched substrate can be seen in Figure 1.7. The isotropic undercut with rounded edges is a characteristic feature.

The masking layer is designed to be smaller than the intended pattern on the substrate. Care has to be taken to ensure accurate calculation of the masking layer dimensions.

Anisotropic wet etching

The rate of anisotropic etching is dependant on the orientation of the substrate. A {100} plane exposed to an etchant, will etch away rapidly while {111} planes will etch away slowly [28]. This occurs due to the

Figure 1.7: Isotropic wet etchings

(Figure adapted from Microsystem Design [28])

fact that atoms in the {111} planes are bounded more tightly than atoms in the {100} planes. As a result of this phenomenon the feature in Figure 1.8 can fabricated, by etching a substrate with a (100) orientation.

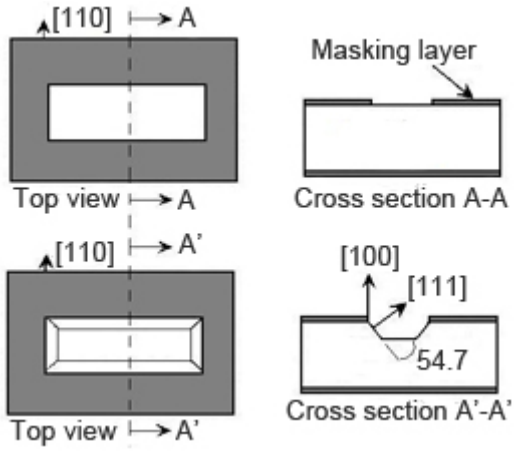


Figure 1.8: Anisotropic wet etching of a silicon wafer

(Figure from Microsystem Design [28])

Dry etching

According to Madou [22], dry etching covers a family of methods by which a solid surface is etched in the gas vapor phase. Physically it is done by ion bombardment and chemically by a chemical reaction through a reactive species at the surface. This is a subtractive process that removes material from a substrate usually covered by a masking layer (oxide or resist). Dry etching should only be considered if a feature with deep side walls or better resolution is needed, as it is a costly and complex process.

Dry-etching techniques can be classified according to the specific setup method as either glow discharge (diode setup) or ion beam (triode setup) [22]. Plasma is generated in the same vacuum chamber where the substrate is located when using the glow discharge method. During ion beam techniques, the plasma is generated in a separate chamber from which ions are extracted and directed towards the substrate by a number of grids [22].

Commonly used dry etching techniques are *plasma etching, reactive ion etching* or *ion beam etching*.

Plasma etching

Plasma Etching (PE) is a dry etching technique that ionizes gas molecules inside a chamber to form neutral chemical species such as chlorine or fluorine atoms [22]. An electrode at the top of a chamber emits a RF excitation which ionizes the gas molecules. The target wafer is contacted with a grounded electrode. Due to ion movement inside the chamber, the plasma will come in contact with unmasked wafer surfaces and react chemically to etch away the material. Etching is highly isotropic.

Reactive ion etching

Reactive ion etching (RIE) differs slightly from PE, as the top electrode is now connected to ground and the electrode on which the wafer is positioned, emits a RF excitation. Moving electrons reach the target wafer and reacts chemically to polarize the substrate surface negatively. A decrease of free electrons in the plasma creates an overall positive charge and this produces an acceleration of ions from the positively charged plasma towards the negatively charged wafer surface [28]. Anisotropic etching occurs due to physical bombardment of ions that are accelerated towards the target surface. As the ions do not strike the sidewalls of the etch, nearly vertical sidewall features can be produced [28].

Another variant of this technique is *deep reactive ion etching* (DRIE) which is used for the fabrication of high-aspect-ratio micro structures. Dry etching stations generate high plasma densities to achieve high etch rates, while operating at low pressures [22].

Ion beam etching

Ion beam etching (IBE) or ion beam milling is a physical etching technique implemented by a triode setup [22]. Usually this technique is very useful for etching of metals and other materials that do not react well to chemical etching. IBE can be explained by means of Figure 1.9 where a Kaufman source (hot filament) is used to enhance ionization.

Madou [22] explains that the plasma source is decoupled from the from the substrate and is placed on a third electrode. The plasma source can be either an RF discharge or a DC source. In Figure 1.9 a DC source is used. Low energy electrodes can be extracted from an auxiliary thermionic cathode (hot filament neutralizer) to neutralize the target substrate. Electrons emitted from the hot filament can be accelerated by a potential difference between a cathode filament and an anode. Ions are extracted from the upper chamber by extraction grids and accelerated towards the target substrate.

Ion bombardment generates heat and with the high energy levels of triode setup, it is very likely to burn polymer masks and etch away any other materials. The substrate should also be tilted so molecules will have the tendency to fall rather than redeposit.

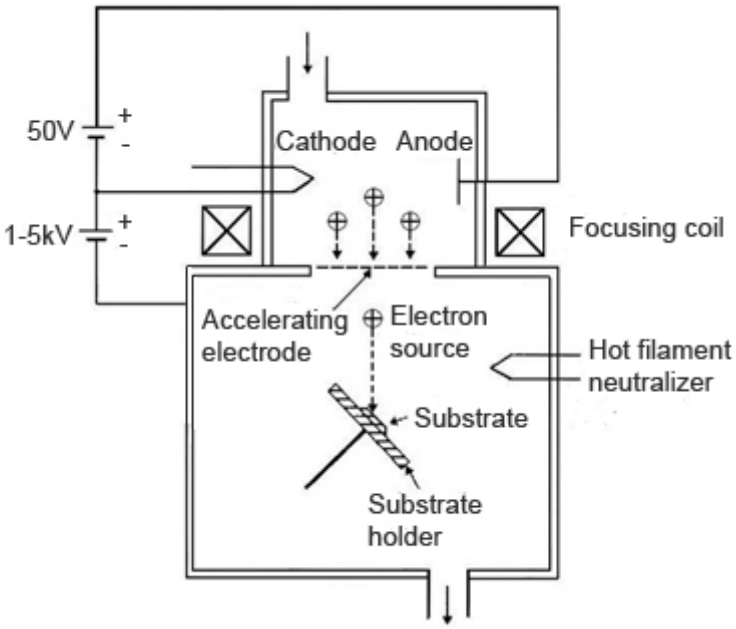


Figure 1.9: The IBE apparatus with triode setup

(Figure adapted from Fundamentals of Microfabrication [22])

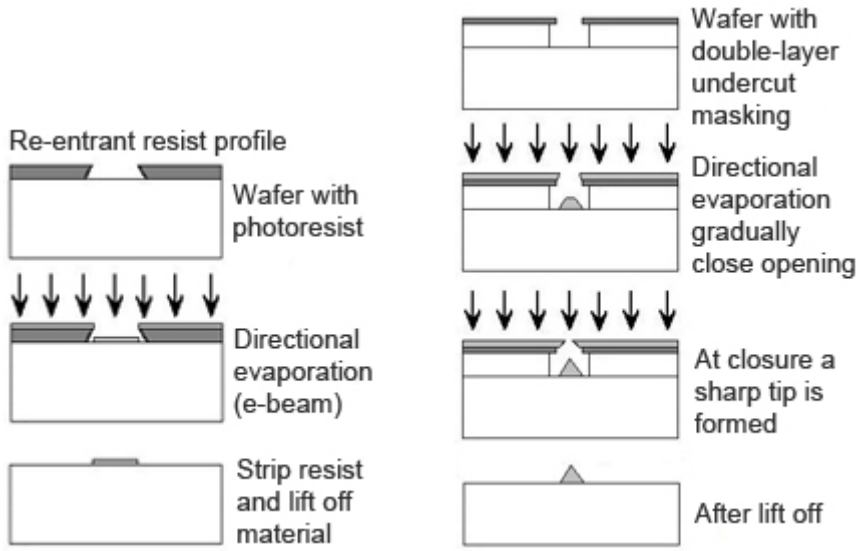
Lift off

Lift off is an additive process used with certain catalytic metals that are difficult to etch with plasmas [28]. Figure 1.10a shows the lift off process using an directional source to evaporate the metal. A photoresist is used to pattern a substrate with the re-entrant resist profile before depositing the evaporated metal. It is important that a discontinuity is formed in the metal deposition, to ensure that the excess metal lifts off when the resist is stripped.

Another common application of the lift off process given by Senturia [28] is used to create the sharp metal tip feature seen in Figure 1.10b. Evaporated metal is deposited onto the edges of a masking layer, which gradually closes the opening. As the opening is closed the deposited feature slopes upwards to form a very sharp tip.

1.2.2.6 Wafer bonding

Two wafers can be joined together using one of three methods used for wafer bonding: direct wafer bonding, anodic bonding and bonding with an intermediate layer [28].



(a) Lift off method for patterning evaporative metals

(b) Modified lift off process to create sharp tips

Figure 1.10: The lift off process

(Figures adapted from [28])

Direct wafer bonding

This process is mainly used for the bonding of two silicon wafers under high temperatures and can be seen in Figure 1.11. Firstly the wafer surfaces are cleaned and hydrated to create the smooth bonding surfaces. Bonding will fail locally if all free particles are not removed from the bonding surfaces. Secondly, the bonding surfaces must be contacted and pressed together to create a hydrogen bond. The wafers are then chemically fused together by placing them in a high temperature furnace [28]. The top wafer can be machined or wet etched to form a thin top layer.

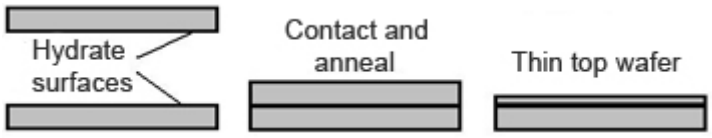


Figure 1.11: The direct wafer bonding of two silicon wafers

(Figure adapted from Microsystem Design [28])

Anodic bonding

Anodic bonding, also called *field-assisted* bonding, is used for the bonding of certain glasses to a conductor and can be explained by means of Figure 1.12 When a conductor such as silicon is placed on glass

and then heated to about 500°C, sodium ions will be repelled from the glass surface if a positive voltage ($300V - 700V$) is applied to the silicon surface [28]. Due to the mobility of the sodium ions in glass, a negative charge will be created at the glass surface. The attraction force between the positively charged silicon surface and the negatively charged glass surface will bring the two wafers in intimate contact. A high temperature anneal will chemically fuse the two wafers together.

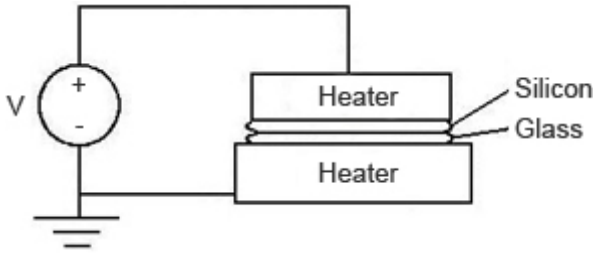


Figure 1.12: Anodic wafer bonding of silicon to glass

(Figure adapted from Microsystem Design [28])

Bonding with an intermediate layer

This process uses an adhesive substance to join two wafers together. When choosing a suitable adhesive, careful consideration must be given to the thermal and cleanliness requirements for microfabrication [28].

1.3 Scaling effects of miniaturization

Scaling down from macro to micro sizes, will introduce certain unexpected but explainable scaling effects. Certain significant properties become insignificant in micro scale and vice versa. To better explain the scaling laws, a scaling variable s will be used, where s^3, for example, corresponds to a scaling of magnitude 10^3.

1.3.1 Surface area, volume and length

Miniaturization of an object changes the ratio of surface area to volume. This scaling law might be explained better by means of Figure 1.13.

A cube with a side of length 10 mm has a surface area of 600 mm^2 and a volume of 1000 mm^3. A smaller cube with a side of length 1 mm has a surface area of 6 mm^2 and a volume of 1 mm^3. The ratio of volume to surface area of the bigger cube is 1.6 and that of the smaller cube is 0.16. The ratio of volume to surface has also decreased by a factor of 10, the same as the factor of miniaturization. Therefore the ratio of volume to surface area has decreased by s^1.

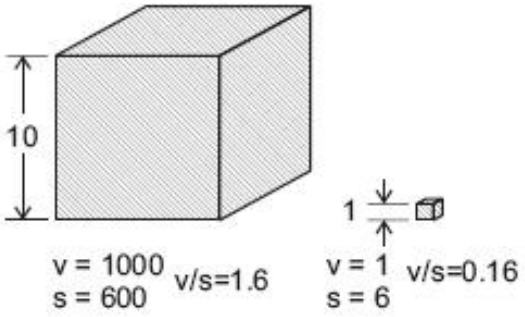


Figure 1.13: The scaling effect of volume, surface and surface/volume ratio

(Figure adapted from Chollet [8])

This decrease in ratio of volume to surface area has a significant effect on certain design criteria of MEMS: friction force (proportional to surface area) will become larger than inertia (proportional to mass, hence volume); heat dissipation (proportional to surface area) will become larger than heat storage (proportional to volume); energy coupling will become more attractive than energy storage [8].

1.3.2 Surface tension

The mass of a liquid scales proportional to volume s^3, whereas surface tension scales proportional to the length s^1. Therefore the weight of the fluid decreases more rapidly than surface tension and it becomes more difficult to flow in micro scale.

1.3.3 Heating and cooling

Miniaturization of an object has a profound effect on the time it takes to heat up or cool down. The heating or cooling time constant is given as

$$\tau = \frac{\rho c_p \mathcal{V}}{h A_s}, \tag{1.3.1}$$

where A_s is surface area, $\mathcal{V}$ is volume, c_p is the specific heat capacity and h is the heat transfer coefficient at the surface [22].

The time constant scales proportional to ratio of volume to surface area also denoted as s^1. Therefore the smaller an object gets, the faster it heats up and cools down.

1.3.4 Strength-to-weight ratio

The strength-to-weight ratio scales as s^{-1} (or l^2/l^3), as strength scales proportional to area l^2 and weight proportional to volume l^3. Therefore it can be assumed that small things are relatively stronger [22]. The strength-to-weight ratio scaling law explains the known fact that an ant can carry almost 10 - 50 times its own weight. Humans can carry approximately one times its own body weight.

1.3.5 Inertia effect

Moment of inertia is one the highest order scaling laws with a scaling of s^5. To prove this statement, moment of inertia can be given as

$$\mathscr{I} = \int r^2 dm, \tag{1.3.2}$$

where r is the radius of a circular disk and m is the mass. Mass scales proportional to volume l^3 and r^2 scales proportional to the area l^2 of the disk. Therefore inertia scales proportional to $l^2 \times l^3 = l^5$.

1.3.6 Microfluidics

Microfluidics introduce a number of important scaling laws. The first is that the flow in capillaries or micro tubes is often laminar. The Reynolds number can be given as

$$Re = \frac{\rho v_m L_c}{\mu}, \tag{1.3.3}$$

where ρ is the fluid density, v_m is the mean velocity, L_c is the characteristic length and μ is the dynamic viscosity of the fluid.

One has to assume a Reynolds number of far less than 100 and often $Re < 1$ due to the dimensions of micro structures. Flow only becomes turbulent at a Reynolds number of about 2000.

Another important scaling law is the domination of viscous forces for Reynolds numbers less than unity, compared to the inertial forces. Take the Navier-Stokes equation

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \overrightarrow{v_g}) = 0, \tag{1.3.4}$$

where ρ is the density of the gas, t is time and $\overrightarrow{v_g}$ is the velocity vector of the gas.

For $Re < 1$ the inertia terms will be dominated by the part of the equation containing viscous forces and can be neglected. The equation now becomes

$$\nabla p = \mu \nabla \overrightarrow{v_g}. \tag{1.3.5}$$

1.4 MEMS and microfluidic gyroscopes

1.4.1 Introduction

A gyroscope is an inertial sensor that measures the angular rotation or velocity of a rotating mass about one or several axes. Curey [9] suggests that there are three main types of gyroscopes: Spinning wheel gyros (macro scale), Sagnac effect gyros (macro and micro scale) and Coriolis vibratory gyros (micro scale). This section will focus mainly on the Coriolis vibratory gyroscope, with the exception of one microfluidic gyroscope.

Most macro- and micromachined gyroscopes use the Coriolis principle for operation. The Coriolis principle may be explained by means of Figure 1.14 where a person rolls a ball from the center of a frictionless rotating disk with a constant velocity v_r. Beeby [4]explains that the person standing in the rotating frame will observe the ball with a curved trajectory. This is due to the Coriolis acceleration that is proportional to the angular velocity of the disk and the radial velocity of the ball.

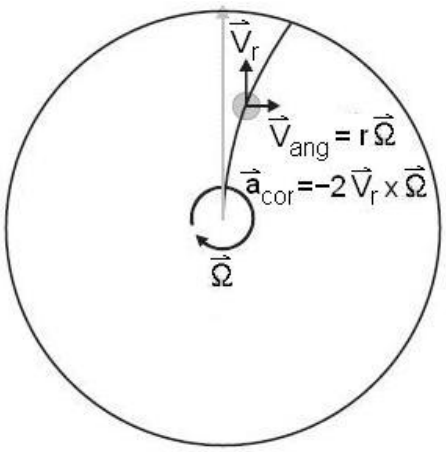


Figure 1.14: A ball rolling from the center of a rotating disk is subjected to a Coriolis acceleration and hence shows a curved trajectory

(Figure adapted from MEMS Mechanical Sensors [4])

The tangential velocity v_{ang} increases radially and therefore the constant Coriolis acceleration is inevitable. The Coriolis acceleration a_c gives rise to the Coriolis force F_c which can be given as

$$F_c = -2m\overrightarrow{\Omega} \times \overrightarrow{v_r}, \tag{1.4.1}$$

where m is the mass of the ball, Ω is the angular velocity of the disk and v_r is the radial velocity of the ball.

Macro scale gyroscopes, using the Coriolis principle as the operating approach, usually consists of some inertial rotor that spins at a high angular velocity. The rotor is supported by three gimbals which gives the system three degrees of freedom. The outer gimbal is fixed to some platform and rotates with it. The two inner gimbals, seen in Figure 1.15, will rotate with respect to the outer gimbal and the inertial rotor will remain in its original position due to the conservation of momentum theory. By measuring the rotation of the gimbals, the rotation of the platform can be calculated.

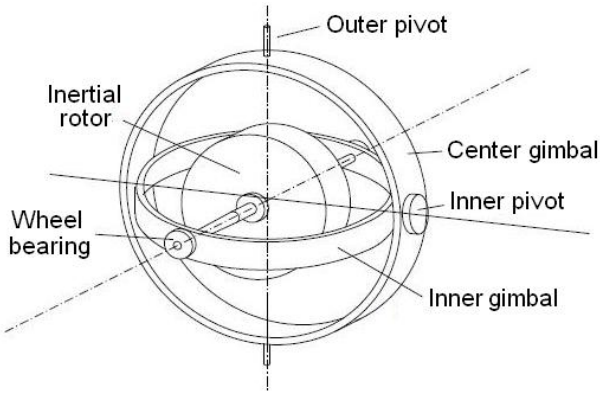


Figure 1.15: Dual-axis macro scale mechanical gyroscope

Beeby [4] explains that no high quality micro scale bearings can be manufactured as the scaling laws are very unfavorable where friction is concerned. Therefore the same approach as in macro scale gyroscopes cannot be used for MEMS gyroscopes. Another innovative approach for MEMS gyroscopes, the MEMS vibratory gyroscope, can be explained by means of Figure 1.16.

Nearly all MEMS gyroscope concepts consists of a vibrating structure that couples energy from a forced primary oscillation mode to a secondary sensing oscillation mode using the Coriolis force [4]. The proof mass in Figure 1.16 is excited along the primary axis or x-axis with a constant frequency and amplitude (usually driven at resonance). An angular velocity about the z-axis couples energy into an oscillation along the secondary axis or the y-axis as a result of the Coriolis force.

1.4.2 Classification

Quite a few MEMS gyroscope concepts have been proposed in the past and they can be classified according to four main criteria: the gyroscope configuration, the modulation principle, the actuation technique and the sensing technique.

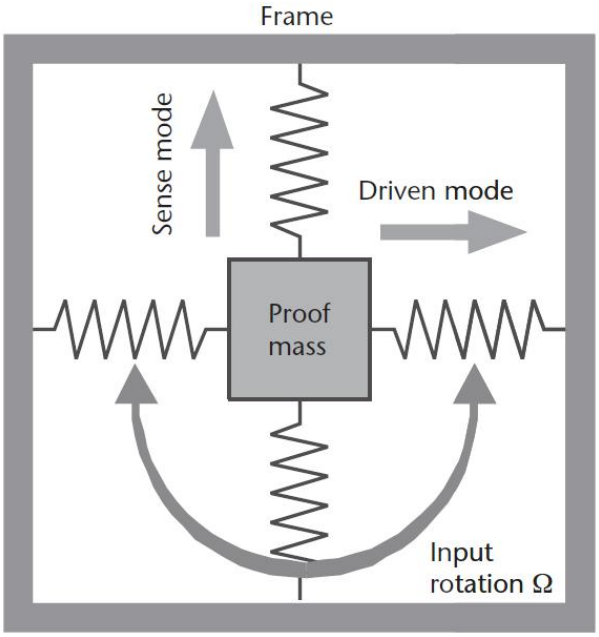


Figure 1.16: MEMS vibratory gyroscope principle

(Figure adapted from MEMS Mechanical Sensors [4])

1.4.2.1 Gyroscope configurations

It is an impossible task to list every known configuration of MEMS vibratory gyroscopes introduced in the literature. Many variations of the same concept exists with only a few modifications of the original configuration. This section gives an overview of the basic configuration concepts and further research opportunities.

Tuning fork vibratory gyros

The tuning fork gyroscope consists of at least one pair of tines and a supporting torsional structure as seen in Figure 1.17. The forced primary oscillation is applied to the first pair of tines. These tines vibrate at an equal constant frequency and amplitude along the x-axis, but in opposing directions. In other words, for the primary mode the tines vibrate in antiphase (180° out of phase) so that no force results on the supporting torsional structure.

Rotation about the z-axis creates two opposing Coriolis forces that couples energy from the primary mode into secondary oscillations along the y-axis. The vibration of the secondary mode is orthogonal to the vibration of the primary mode and its amplitude is proportional to the angular rate of the rotation. This vibration also occurs in antiphase and will be seen in the second pair of tines (or if only one pair exists, it can be seen in the same pair of tines).

One of the main advantages of the tuning fork configuration, is that it is suitable for batch-processing.

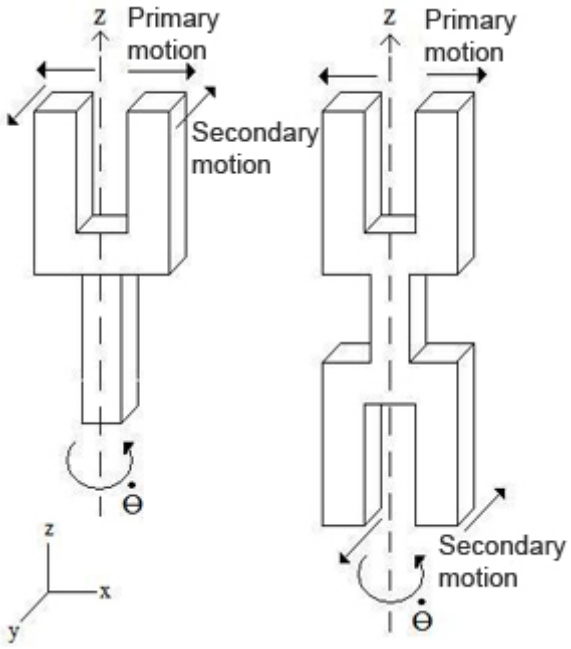


Figure 1.17: Classic tuning fork gyroscope configurations

Although more complex variations do exist, the geometry is fairly simple and easily fabricated from silicon. Another advantage is its symmetrical possibility and stable center of gravity [2].

Vibrating beams

This group of vibratory gyroscopes is a little more diverse than the tuning fork gyroscope. A vibrating beam gyroscope can be a simple resonating elastic beam structure or a beam mass structure.

Figure 1.18 shows the rectangular elastic vibrating beam structures driven by piezoelectric film. Papers were found on triangular beam (Murata's Gyrostar), cylindrical tube [20] and rectangular beam [35] structures.

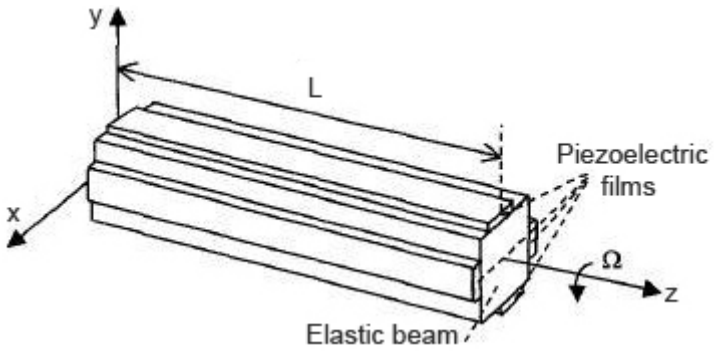


Figure 1.18: Elastic vibrating beam structure driven by piezoelectric film

(Figure adapted from [35])

A piezoelectric film is bonded to the elastic beam structures as shown in Figure 1.18. The beam is excited into primary oscillation by the piezoelectric film. Rotation about the z-axis subjects the elastic structure to the Coriolis effect and a secondary oscillation occurs orthogonal to the vibration of the primary mode. The angular rate of the rotation can be sensed by piezoelectrically detecting the secondary mode.

Yang [35] suggests that greater resonance is possible with the piezoelectric beam gyroscope, which suggests greater sensitivity.

An example of beam mass structures that act as vibrating beam gyroscopes can be seen in Figure 1.19. Once again the first beam pair is excited into primary oscillation by some actuation technique (in this case by electrostatic comb drives). Rotation about the z-axis subjects the mass to the Coriolis effect and couples energy into a secondary vibration orthogonal to the first. The angular rate can be sensed by detecting the secondary mode vibration.

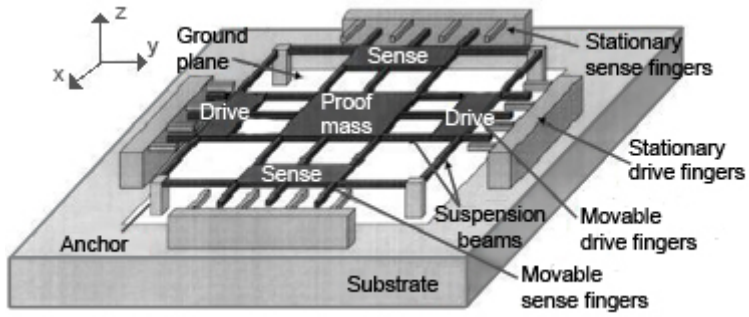


Figure 1.19: Beam mass structure example using comb drive for excitation and detection

(Figure adapted from [1])

An advantage of the beam mass structures is the possibility of symmetry and matched resonant frequencies of primary and secondary modes. This relates to amplified sensitivity (resolution) with the mechanical quality factor.

Vibrating plates

A dual axis vibrating plate gyroscope consists of an inertial rotor suspended by four torsional spring structures as seen in Figure 1.20. The rotor is driven at angular resonance about the z-axis representing the primary mode. Rotation of the platform about the x-axis will result in a rotation of the circular disk about the y-axis due to the Coriolis effect. Similarly, a rotation about the y-axis will result in a rotation about the x-axis. Usually the angular rate is measured by a change in capacitance between the rotor plate and four quarter circle electrodes.

The explained vibrating plate gyroscope can sense angular rate about two axis. Depending on the geometry both single and dual axis gyroscopes are possible. Figure 1.21 shows a single axis vibrating plate from earlier days.

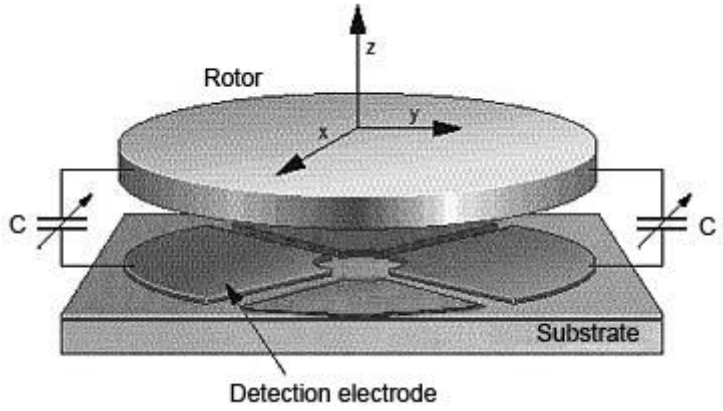


Figure 1.20: A dual axis vibrating plate gyroscope

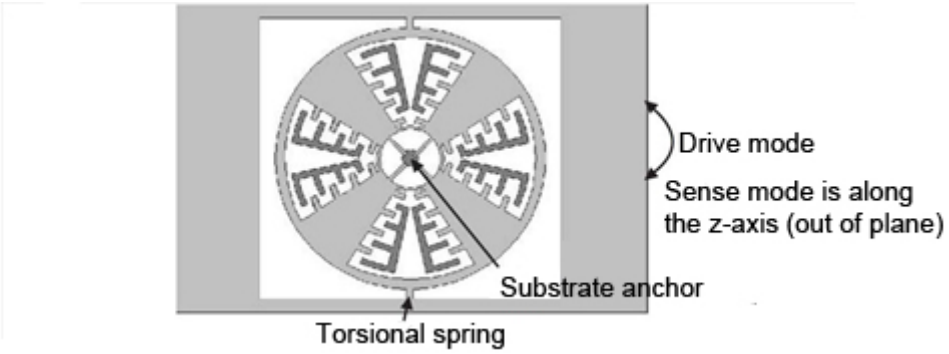


Figure 1.21: A single axis vibrating plate gyroscope

(Figure adapted from MEMS Mechanical Sensors [4])

The vibrating plate gyroscope can sense the angular rate about the two axis orthogonal to the rotor rotation. Vibrating plate gyroscopes can be classified as either angular disk, linear disk or linear plate depending on their geometry [2].

Vibrating ring

The vibrating ring gyroscope consists of a ring anchored in the center by a number curved springs as seen in Figure 1.22. Electrodes are positioned around the periphery of the ring to electrostatically excite the primary mode. The ring is excited to vibrate in-plane and the primary mode vibration has an elliptical shape. Upon rotation about the z-axis, a secondary vibration is excited due to the Coriolis effect. The secondary mode vibration also has an elliptical shape and is offset 45° from the primary mode. The angular rate can be measured by detecting the vibration of the secondary mode capacitively with the positioned electrodes.

Various papers on single axis vibrating ring gyroscopes can be found by scientists such as Najafi [26], Esmaeili [12] and Ayazi [3]. Papers on tri-axial gyroscopes were published by Gallacher [14].

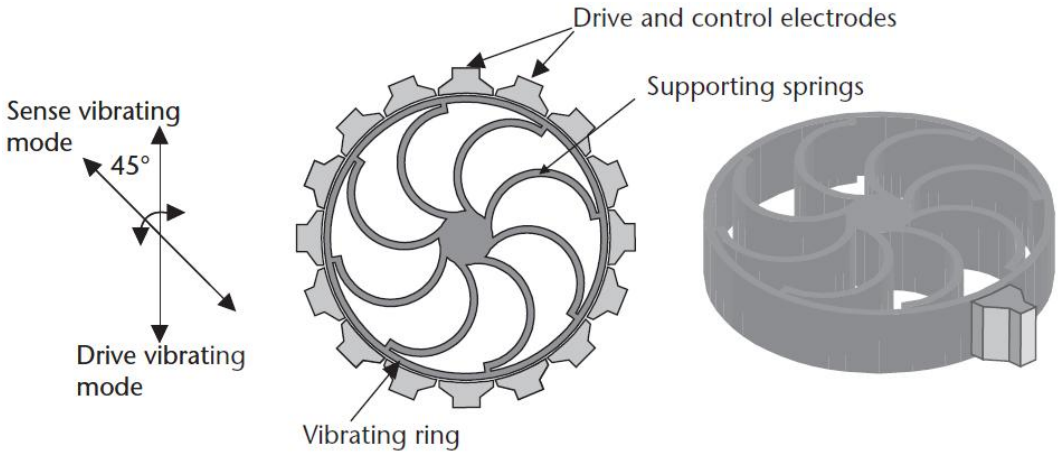


Figure 1.22: A single axis vibrating ring gyroscope

(Figure adapted from MEMS Mechanical Sensors [4])

According to Ayazi [3] the vibrating ring gyroscope holds many advantages. The first is its balanced symmetry that is less sensitive to unwanted vibrations. It is also less sensitive to temperature, since both flexural modes are equally affected by temperature. Electronic tuning is possible, which yields better accuracy.

A disadvantage of this gyroscope concept is its complex and time consuming microfabrication DRIE process. This is however still one of the few MEMS gyroscope concepts that can be developed into a three axial gyroscope.

Thermal convective microfluidic gyroscope

The microfluidic gyroscope consists of a jet pump that periodically circulates a gas through a number of chambers and use the principle of hot-wire anemometry for detection. Zhou and co-workers [36] named the following chambers that can be found in the gyroscope: vibrating chamber, collecting chamber, detecting chamber and tail gas chamber. The thermal convective microfluidic gyroscope can be explained by describing what happens in each of these chambers as seen in Figure 1.23.

The vibrating chamber expands and contracts to periodically circulate the gas through the chambers. As the chamber contracts, pressure builds up and the gas is ejected through the micro jet pump nozzles into the collecting chamber. When the chamber expands again, the pressure in it decreases and gas is drawn from the tail gas chamber into the vibrating chamber. Most of the gas drawn back into the chamber comes from the tail gas chamber and not the nozzle of the micro jet pump, due to the inertia of the moving gas at the nozzles.

The gas in the detecting chamber exhibits a perfect laminar velocity flow profile. According to the hot wire anemometry theory the resistance of the two hot wires are equal (they dissipate heat at the same rate) during laminar flow and zero angular rate [36], as the flow profile is still symmetric.

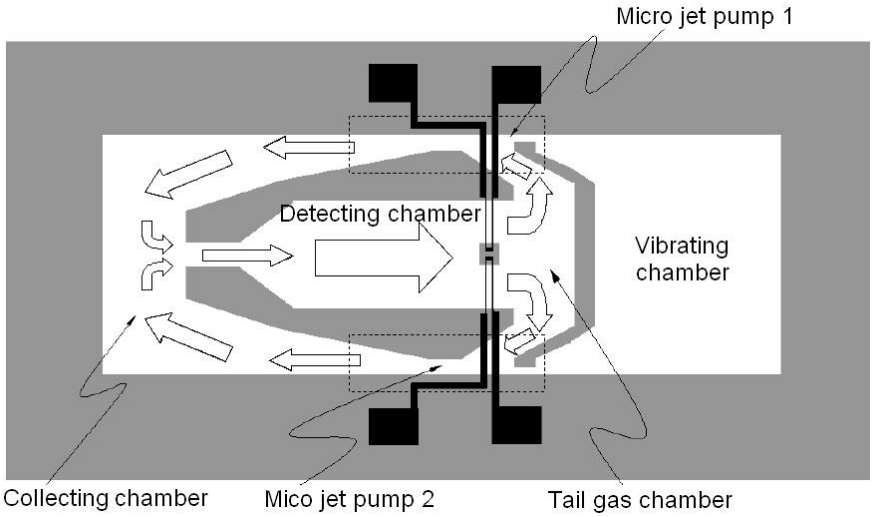


Figure 1.23: A Microfluidic Angular Rate Sensor

(Figure adapted from [36])

If the microfluidic gyroscope is rotated about its sensitive axis, the flow profile changes due to the Coriolis effect, as seen in Figure 1.24. The gas velocity differs at the two hot wires and a change in resistance can be detected using a Wheatstone bridge of resistors. This change in resistance can be used to sense angular rate.

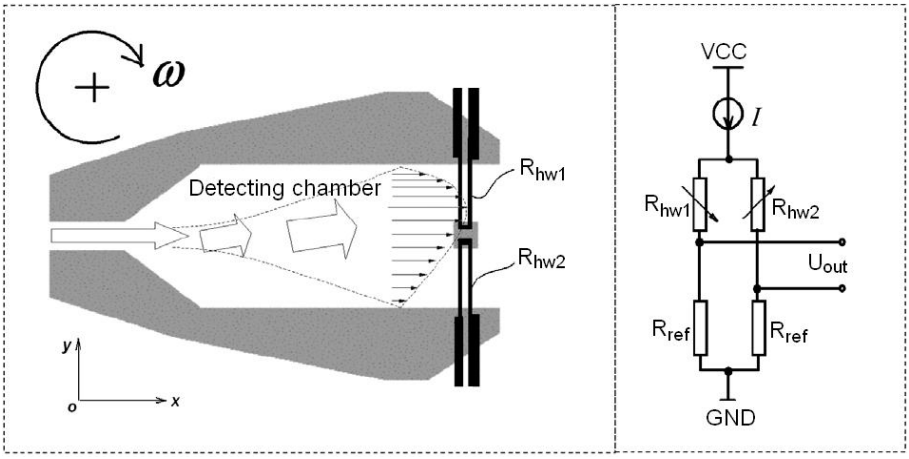


Figure 1.24: Detection theory with hot-wire anemometry

(Figure adapted from [36])

Single axis microfluidic gyroscope models were presented by scientist such as Dau [11] and Zhou [36] to name a few. Dau [10] also developed a dual-axis model of this microfluidics sensor.

The microfluidic gyroscope has a few advantages above the conventional MEMS vibratory gyroscope. The main one being that it has no vibrating mass and therefore can endure more fatigue. It has a longer

life and greater durability.

1.4.2.2 Modulation principles

MEMS vibratory gyroscopes can be either amplitude modulated, phase modulated or direct frequency output (DFOVG) gyroscopes.

Amplitude modulation is the most popular technique and most of the gyroscope configurations introduced in this chapter probably rely on this principle. A primary oscillation is actuated and the Coriolis force couples energy into a secondary oscillation normal to the first. The amplitude of the secondary oscillation will be proportional to the angular velocity for some linear range.

The phase modulation principle is described in a paper by Yang [34], where vibrations are excited in both the x and y-directions and the angular rate is found by detecting the phase difference between the the two vibration signals.

A DFOVG gyroscope proposed by Moussa [25] consists of two drive oscillators that excite two modes of vibration simultaneously, one in and one out of plane. The two electrodes that detect the vibration modes, are connected to the drive oscillators, as a sort of feedback to the oscillator. Due to the Coriolis effect, the resonance frequencies of the two modes of vibration will vary with the applied angular rate. The operating frequencies of the oscillators will vary accordingly, as a result of the feedback from the detection electrode. The difference in operating frequencies is proportional to the angular velocity input for some linear range.

1.4.2.3 Sensing and actuation techniques

A MEMS vibratory gyroscope typically needs a method by which to excite a primary vibration in some beam, plate or membrane and detect a secondary Coriolis induced vibration.

Electrostatic

Electrostatic actuation uses the principle where two plates of opposite charge will attract each other [4]. It is most commonly used (for vibratory gyroscope) in comb-drives, where a number of electrodes are interconnected to form the comb fingers and an inner comb can be actuated by applying a voltage across them. Similarly, a vibration or displacement can be sensed by detection electrodes positioned strategically.

Piezoelectric

Applying a voltage across a piezoelectric material will result in a deformation proportional to that voltage. Similarly, a deformation of the material will result in a proportional voltage. This unique property is favourable, as a vibration can be both excited and sensed using piezoelectric materials.

Thermal

Thermal actuation is possible when bonding two materials, with different thermal coefficients of expansion, also called thermal bimorphs. Applying heat to a thermal bimorph material will result in thermal stresses at the bond surface, which will bend the structure [4].

Electromagnetic

A conductor with an electrical current, placed in a magnetic field, will induce an electromagnetic force perpendicular to both the current and the magnetic field. Permanent magnetic materials, that are compatible with microfabrication, are limited and the magnetic fields are usually generated externally [4]. Both sensing and actuation are possible using a magnetic field.

Piezoresistivity

Piezoresistivity is the effect exhibited by materials to show a change in resistivity due to an applied pressure [4]. This is not really a popular sensing technique due to low manufacturability, as the positioning of resistors require very high accuracy.

Chapter 2

Problem statement

In this thesis, a thermal convective microfluidic gyroscope will be investigated and the detecting chamber will be optimized for sensitivity. This chapter briefly outlines the project objectives and the approach taken to satisfy them.

2.1 The general thermal convective microfluidic gyroscope concept

Consider the thermal convective microfluidic gyroscope proposed by Zhou and co-workers [36]. Figure 2.1 shows the four chambers (vibrating, collecting, detecting and tail gas) described in Section 1.4.2.1.

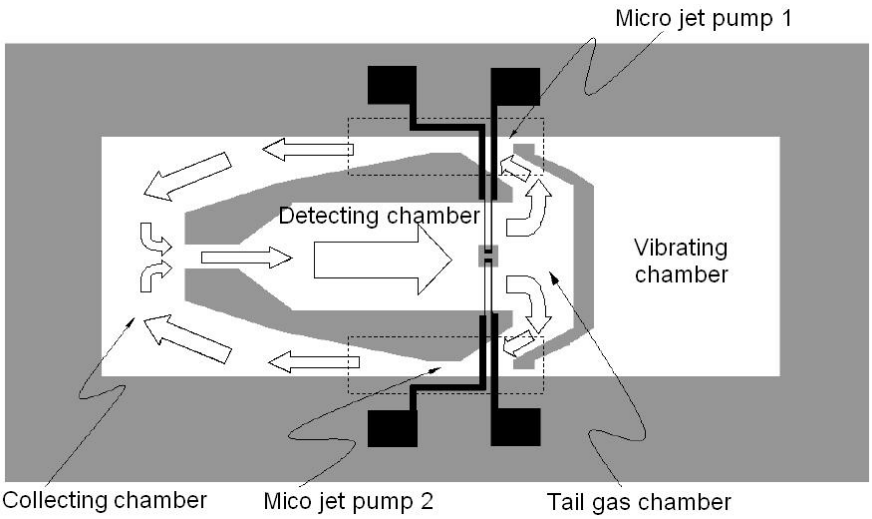


Figure 2.1: A Microfluidic Angular Rate Sensor

(Figure adapted from [36])

A voltage is applied across the terminals of a piezoelectric disc that displaces a membrane periodically. The membrane displacement causes the vibrating chamber to expand and contract and periodically pump the gas through the micro jet pump nozzles, following the collecting chamber into the detecting chamber and finally into the tail gas chamber. This action is repeated to circulate the gas through the chambers.

A perfect laminar flow profile is maintained in the detecting chamber at zero angular velocity and the hot wire resistances are equal. A Coriolis acceleration due to an angular velocity will deflect the flow profile proportional to that angular velocity and the hot wire resistances will change accordingly. A Wheatstone half bridge output voltage will represent the change in hot wire resistances, which can be used to calculate the angular velocity.

The main advantage of this concept is its robustness and long life, due to the fact that it has few moving parts. This is a characteristic that might prove very useful in most applications.

2.2 Existing concepts and shortcomings

Only a few microfluidic gyroscopes have been presented in the literature thus far and Table 2.1 compares some sensor characteristics from them. "—" implies *not determined*. It is clear that this is a relatively new field of study and the standardization of the thermal convective microfluidic gyroscope design process is an important contribution.

Table 2.1: Characteristics from existing thermal convective microfluidic gyroscopes

Prototype	Single/ Dual (axis)	Size [mm]	Resolution [deg/sec]	Linear range [deg/sec]	Sensitivity [mV/deg/sec]	Impact × g	Power [mW]
Dau [11]	Single	14 × 25	0.04	—	0.15	3000	5.5
Zhou [36]	Single	—	—	—	—	—	—
Dau [10]	Dual	14 × 25	0.05	3000	0.107 ; 0.102	—	2.45

Fabrication and testing is essential for development in any field, as it yields valuable experimental results. The standardization of the analytical models and optimization techniques will simplify the design process and encourage fabrication and testing in the future.

2.3 Thesis objectives

This thesis will present the reader with an analytical model for the thermal convective microfluidic gyroscope, in Chapter 3, as well as some important design considerations. Different sequential approximate optimization strategies will be investigated and compared with one another in Chapters 4 and 5. An

existing implementation of the sequential approximate optimization strategy in FORTRAN will be investigated and a similar implementation will be done using MATLAB.

The mathematical optimization problem will be formulated, in Chapter 5, to optimize the detecting chamber of the microfluidic gyroscope for sensitivity. This problem will be solved using both the FORTRAN and MATLAB implementations of the sequential approximate optimization algorithm.

Chapter 3

Analytical model for a thermal convective microfluidic gyroscope

In this chapter a set of design considerations is documented for a thermal convective microfluidic gyroscope. Furthermore, an analytical model is presented for the general gyroscope concept (refer to Section 2.1) and optimization tactics are pointed out.

3.1 Design consideration

A few obvious but important design considerations are:

- A gas must be used for the gyroscope, as a liquid's inertia in micro scale is insignificant when compared to the viscous term in the Navier-Stokes equation.
- The Reynolds number of gasses tends to be higher than those of fluids and laminar flow cannot always be assumed (as in most fluids). The gyroscope should however be designed for laminar flow.
- A gas with increased thermal conductivity will increase the sensitivity of the device, although the density and viscosity of the gas should also be considered.
- The deflection or perturbation of the flow profile will decrease when the velocity across the hot wire increases. Larger deflection means greater sensitivity. Deflection increases with angular rate and the length of the sensitive element from the nozzle.
- Sensitivity, however, increases with the variance of velocity flow between the two hot wires. The variance increases with the gradient of the velocity flow profile which naturally increase with velocity. Careful consideration must therefore be given when choosing the velocity range.

- The position of the two hot wires needs to be carefully selected to ensure sensing the linear part of the velocity flow profile.
- Hot wires must not interfere with the flow profile of the gas, but must also be able to withstand the drag force from the flow.
- Power consumption must be kept at a minimum as these microdevices will most likely be mounted on a circuit board and powered by microelectronic components.
- It is important to manage pressure differences throughout, to maintain a desired flow rate.
- Surface roughness in micro scale may become a significant issue.

3.2 Fluid modelling

There are typically two ways to model a flow field of a fluid: the first is to model the fluid as a collection of molecules, which it really is, and the second is to model the fluid as a continuum (where matter is assumed to be continuous and indefinitely divisible) [13]. The latter is preferred as it is less complicated than the first and will be described in this chapter.

The Knudsen number Kn can be given as a function of the Mach number Ma, the Reynolds number Re and the specific heat ratio (c_p/c_v) γ,

$$Kn = \sqrt{\frac{\pi\gamma}{2}}\frac{Ma}{Re}, \tag{3.2.1}$$

and validates the continuum approach when it is less than 0.1 [13]. The different Knudsen number regimes are summarized in Table 3.1 [13].

Table 3.1: Knudsen number regimes

Euler equations (neglect molecular diffusion)	$Kn \to 0 (Re \to \infty)$
Navier-Stokes equations with no-slip boundary condition	$Kn < 10^{-3}$
Navier-Stokes equations with slip boundary condition	$10^{-3} \leq Kn < 10^{-1}$
Transition regime	$10^{-1} \leq Kn < 10$
Free-molecule flow	$Kn \geq 10$

The Reynolds number Re is the ratio of inertial forces to viscous forces and is given by

$$Re = \frac{v_m L_c}{\nu}, \tag{3.2.2}$$

where v_m is the mean velocity of the fluid, L_c is the characteristic length and ν is the kinematic viscosity of the fluid.

The Mach number Ma is the ratio of inertial forces to elastic forces and can be used as a measure of compressibility. The Mach number is given by

$$Ma = \frac{v_m}{a_0}, \tag{3.2.3}$$

where a_0 is the speed of sound. A compressible fluid, such as air, can generally be treated as an incompressible fluid when $Ma < 0.3$.

3.3 Flow through a rectangular microfluidic channel

In order to simplify the design problem, it will be assumed that flow is always laminar ($Re < 2100$), the fluid is incompressible (density does not change with time and $Ma < 0.3$), the fluid is a Newtonian fluid (μ is constant) and flow is subjected to the no-slip boundary condition (fluid velocity at the channel walls is zero).To validate these assumptions, the gas flow must be designed for $Kn < 10^{-3}$. In the optimization problem (refer to Chapter 5), these conditions are enforced as constraints.

A fluid flows through a microchannel due to a pressure difference across the channel. Usually flow occurs from a high pressure to a low pressure, hence a negative pressure gradient $\frac{\partial p}{\partial z}$. The symmetrical flow profile is depicted in Figure 3.1.

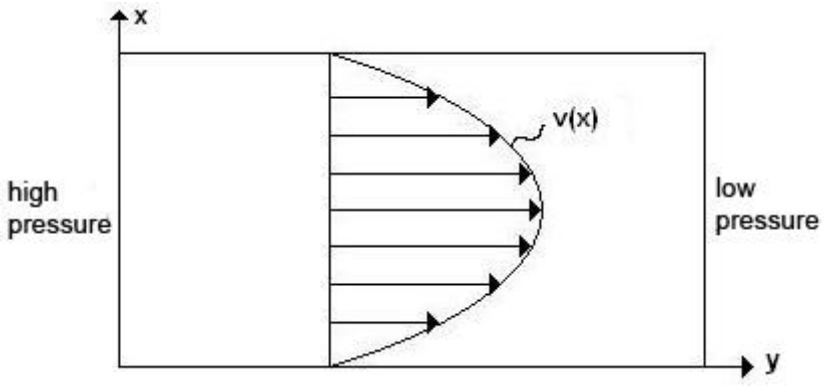


Figure 3.1: Symmetrical flow through a rectangular microchannel

Flow through a microchannel can be described by two governing equations. The first is the conservation of mass equation, also known as the continuity equation

$$\frac{\partial \rho}{\partial t} + \nabla(\rho \overrightarrow{v_g}) = 0, \tag{3.3.1}$$

where ρ is the density of the gas, t is time and $\overrightarrow{v_g}$ is the velocity vector of the gas.

The second equation is the equation for conservation of momentum derived from Newton's second law $F = ma$. The conservation of momentum equation, better known as the Navier-Stokes equation, for an incompressible fluid can be given as

$$\frac{\partial \overrightarrow{v_g}}{\partial t} + (\overrightarrow{v_g}\nabla)\overrightarrow{v_g} = -\frac{1}{\rho}\nabla p + \overrightarrow{g} + \nu\nabla^2\overrightarrow{v_g}, \tag{3.3.2}$$

where $\overrightarrow{g}$ is gravitational acceleration, p is pressure and ν is the kinematic viscosity of the gas.

By assuming incompressible flow, the first term of (3.3.1) becomes zero and the continuity equation can be simplified to

$$\nabla\overrightarrow{v_g} = 0. \tag{3.3.3}$$

The velocity in the y-direction will be a function of only x and z. Therefore by substituting (3.3.3) into (3.3.2) and assuming flow is completely horizontal, the Navier-Stokes equation can be reduced to

$$\frac{1}{\rho}\frac{\partial p}{\partial y} = \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial z^2} - \frac{\partial v}{\partial t}. \tag{3.3.4}$$

3.3.1 Equivalent electrical circuit modelling

Using the $e \rightarrow V$ convention described by Senturia [28], an equivalent circuit model can be created to effectively model Poiseuille flow through a rectangular microchannel.

The effort (across) variable represents the variable responsible for the work done in the system. In an electrical system voltage is the effort variable. In the microfluidic system the effort variable is the pressure difference Δp.

The flow (through) variable represents the rate at which work can be done by the effort variable. Current is the flow variable in an electrical system, but in microfluidics the flow variable is volume flow rate Q.

The equivalent layer resistance of the microchannel can be calculated by

$$R_{layer} = \frac{effort}{flow} = \frac{\Delta p}{Q}. \tag{3.3.5}$$

3.4 Sensitive element

A relationship can be found between the perturbation of the flow profile due to an applied angular acceleration and the difference in resistance measured across two hot-wires. The section explains and

contributes to a model developed by Zhou *et al.* [36].

3.4.1 Perturbation of flow profile due to inertia

The Coriolis acceleration vector $\vec{a}_{\omega_z}$ can be given as

$$\vec{a}_{\omega_z} = -2\vec{\omega}_z \times \vec{v}, \tag{3.4.1}$$

where $\vec{v}$ is the velocity vector and $\vec{\omega}$ is the angular rate vector.

The velocity in the direction of deflection can be found by integration of (3.4.1)

$$\vec{v}_{\omega_z} = \int \vec{a}_{\omega_z} dt = \int -2\vec{\omega}_z \times \vec{v}\, dt. \tag{3.4.2}$$

Solving (3.4.1) at the position of the hot-wires, yield

$$v_{\omega_z} = -2\omega_z v t, \tag{3.4.3}$$

where t is the time it takes one gas particle to travel from the nozzle exit to the sensing element.

The deflection of the velocity flow profile in the direction orthogonal to the direction of flow, can be calculated by the double integration of (3.4.1) and is given by

$$\delta_x = -\omega_z v t^2, \tag{3.4.4}$$

or

$$\delta_x = -\omega_z \frac{{L_{ch}}^2}{v}, \tag{3.4.5}$$

where L_{ch} is the length of the channel that the particle has to travel [11].

3.4.2 Flow velocity sensed by hot-wire anemometry

A hot wire is immersed in the gas and heated by an electrical current. The convective heat transfer of the gas flow across the hot wire can be used as a measure of gas flow velocity.

One of two methods can be used to calculate the steady state output response from the two hot wires, *CCR* (constant current response) or *TCR* (constant temperature response). The CCR method is used in this design and described below.

3.4.2.1 The CCR method

The CCR method is usually the choice for thermal convective gyroscopes, as the circuitry is simple and electronic parts can be minimized. A schematic drawing of the of the CCR circuit is given in Figure 3.2.

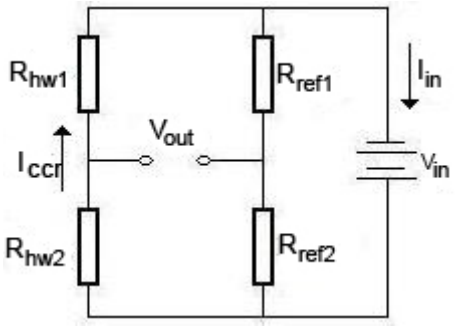


Figure 3.2: Constant current circuit

(Figure adapted from [7])

The amount of heat generated by each hot wire can be calculated using Joule's law (assuming heat transfer is at equilibrium)

$$P = I^2 R_{hw} = hA_{hw}(T_{hw} - T_g), \tag{3.4.6}$$

where I is the input current, R_{hw} is the resistance of the hot wire, h is the heat transfer coefficient, A_{hw} is the surface area of the hot wire and T_{hw} and T_g is the temperatures of the wire and the gas respectively [7].

Assuming natural convection is negligible, the heat transfer coefficient h can be calculated from

$$h = \frac{k}{L_c} Nu, \tag{3.4.7}$$

where k is the thermal conductivity of the fluid, L_c is the characteristic length (the characteristic length for a flat plate can be taken as the width of the plate) and Nu is the dimensionless parameter called the Nusselt number [6].

The Nusselt number can be calculated once the Reynolds and Prandtl numbers are known. The Reynolds number can be calculated from

$$Re = \frac{v_m L_c}{\nu}, \tag{3.4.8}$$

where v_m is the mean fluid velocity and ν is the kinematic viscosity of the fluid [6].

The Prandtl number is calculated from

$$Pr = \frac{c_p \mu}{k}, \tag{3.4.9}$$

where c_p is the specific heat(at constant pressure) and μ is the dynamic viscosity [6].

Once these two dimensionless parameters are known, the Nusselt number can be calculated as

$$Nu = 1.1 C Re^n Pr^{0.31}, \tag{3.4.10}$$

where C and n are empirical constants whose values vary with the Reynolds number [11].

The heat transfer coefficient h varies with the velocity of the gas v and can be written in the form

$$h = a + bv^c, \tag{3.4.11}$$

where a, b and c are coefficient obtained by calibration [36]. The coefficient a accounts for some natural convection. As forced convection will dominate natural convection, we will assume that a is negligible. Substituting (3.4.7) and (3.4.10) into (3.4.11) an approximation for the coefficients a, b and c in (3.4.11) can be found analytically:

$$\begin{aligned} a &= 0 \\ b &= \frac{1.1 k C {L_c}^{n-1} Pr^{0.31}}{\nu^n} \\ c &= n \end{aligned} \tag{3.4.12}$$

The resistance of the hot wires at a reference temperature, T_g, can be given as

$$R_0 = \lambda_r \frac{L_{hw}}{A_c}(1 + \alpha(T_g - 20)), \tag{3.4.13}$$

where λ_r is the resistivity of the hot wire material, L_{hw} and A_c is the length and cross-sectional area of the hot wire and α is the temperature coefficient of resistance (PPM/°C) [5].

The temperature of the two hot wires vary with the velocity gradient across them, due to increased thermal convection with an increased velocity. The change in temperature results in a change in hot wire resistances due to the thermoresistive effect [36]:

$$R_{hw} = R_0(1 + \alpha(T_{hw} - T_g)) \tag{3.4.14}$$

When zero angular rate is applied, the gas flow profile is symmetric about the center of the tube and the hot wire resistances are equal ($R_{hw1} = R_{hw2} = R_{hw}$). Equal hot wire resistances result in a zero voltage output by the Wheatstone half-bridge shown in Figure 3.2. An applied angular rate will perturb the flow profile and resistances will change linearly ($R_{hw1} = R_{hw} + \Delta R$ and $R_{hw2} = R_{hw} - \Delta R$). The voltage output of the Wheatstone half-bridge becomes a function of the change in resistance ΔR between the two hot wires

$$V_{out} = I\Delta R, \tag{3.4.15}$$

where V_{out} is the voltage output and ΔR is the change in resistance.

Substituting (3.4.6) and (3.4.14) we can derive a second equation for the heat transfer coefficient

$$h = \frac{I^2 R_{hw}}{A_{hw}(T_{hw} - T_g)} = \frac{\alpha I^2 R_{hw} R_0}{A_{hw}(R_{hw} - R_0)}. \tag{3.4.16}$$

The hot wire resistance R_{hw} then becomes

$$R_{hw} = \frac{hA_{hw}R_0}{hA_{hw} - \alpha I^2 R_0}. \tag{3.4.17}$$

Substituting (3.4.11) into (3.4.17) gives the hot wire resistance R_{hw} to be

$$R_{hw} = \frac{(a + bv^c)A_{hw}R_0}{(a + bv^c)A_{hw} - \alpha I^2 R_0}. \tag{3.4.18}$$

The change in resistance ΔR can be found by solving the derivative of (3.4.18) and simplifying:

$$\frac{dR_{hw}}{dv} = \frac{-\alpha A_{hw} I^2 {R_0}^2 bcv^{c-1}}{[A_{hw}(a + bv^c) - R_0 I^2 \alpha]^2} \tag{3.4.19}$$

$$\Delta R = \frac{-\alpha A_{hw} I^2 {R_0}^2 bcv^{c-1}}{[A_{hw}(a + bv^c) - R_0 I^2 \alpha]^2} \Delta v \tag{3.4.20}$$

Since the velocity of the gas flow across the hot wire will change proportional to an angular velocity, we can say that

$$\Delta v = \kappa\omega, \tag{3.4.21}$$

where κ is a constant [36]. For the purposes of optimization, discussed in Chapter 5, it is important to write the change in velocity as a function of deflection. It is also evident from (3.4.5) that the deflection, δ, is proportional to angular velocity and can also be given proportional to the change in velocity

$$\Delta v = \kappa\delta, \tag{3.4.22}$$

where κ is still a proportionality constant and must be found numerically.

Substituting (3.4.20), (3.4.5) and (3.4.22) into (3.4.15) gives the output voltage of the Wheatstone half-bridge in terms of angular velocity ω

$$V_{out} = \frac{\alpha A_{hw} I^3 {R_0}^2 bc v^{c-1}}{[A_{hw}(a+bv^c) - R_0 I^2 \alpha]^2} \kappa \left(\frac{\omega {L_{ch}}^2}{\nu_m}\right). \tag{3.4.23}$$

3.4.2.2 Amplifier circuit

The output voltage from the Wheatstone half-bridge needs to be amplified in order to increase the sensitivity of the gyroscope. The single differential operational amplifier (op amp) circuit used to amplify the signal is shown in Figure 3.3.

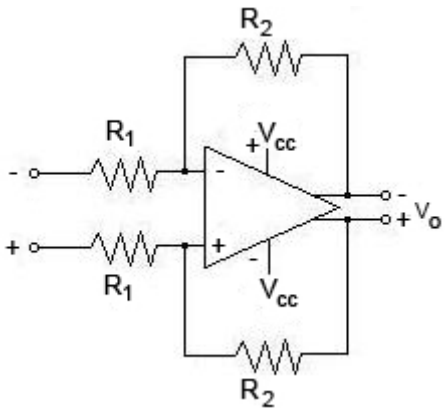


Figure 3.3: Single differential op amp circuit

The amplifier gain for the circuit in Figure 3.3 can be given as

$$Gain = \frac{R_2}{R_1} = \frac{V_o}{V_{out}}, \tag{3.4.24}$$

where V_o is the amplified voltage output and V_{out} is the output voltage from the Wheatstone half-bridge in (3.4.23).

For an ideal op amp, the circuit can be design to give a maximum output, $V_{o,max} = V_{cc}$, at the maximum Wheatstone half-bridge output voltage, $V_{out,max}$, corresponding to the maximum angular velocity ω_{max}.

3.4.2.3 Wheatstone half-bridge power consumption

Power consumption is an important consideration in any design and it will most likely limit the sensitivity of the gyroscope.

Power consumption P_{hw} can be given as a function of the source voltage V_{in} and source current I_{in} in Figure 3.2

$$P_{hw} = V_{in} I_{in}, \tag{3.4.25}$$

where

$$V_{in} = 2IR_{hw} \tag{3.4.26}$$

and

$$I_{in} = \frac{V_{in}}{2R_0} + I. \tag{3.4.27}$$

3.5 Micropump

3.5.1 Piezoelectric actuation

The volumetric displacement of a circular piezoelectric disc is described by Tay [33] as

$$V_{total} = \int_0^b \int_0^{2\pi} w_I r d\theta dr + \int_b^a \int_0^{2\pi} w_{II} r d\theta dr, \tag{3.5.1}$$

where r is the radial distance in polar coordinate and θ is the angle in radians. Parameters w_I and w_{II} describe the deflection of the membrane and the piezoelectric disc respectively. The formulation of parameters w_I and w_{II} is described in Appendix A.

3.6 Structural analysis

It is an impossible task to analyze the entire structure without the help of a FEM solver. The sensitive element was identified as the area where failure is likely to occur first and will be analyzed.

There are two factors contributing to failure at the sensitive elements: stresses due to *drag force* from the fluid flow and insufficient *impact* resistance.

Assuming a thin flat plate in the flow path, the drag force can be calculated as

$$F_D = \frac{1}{2} \rho v^2 A_{obs} C_D, \tag{3.6.1}$$

where A_{obs} is the area of the obstacle in the flow path and C_D is the drag coefficient [6]. For a thin flat plate the drag coefficient, C_D, is equal to the friction coefficient, C_f, as the pressure drag will be zero [6].

Due to gyroscope applications, they need to have a great impact or shock resistance. Impact resistance can be calculated as an acceleration

$$a_{imp} = \frac{\sigma_{max} A_n}{m_{hw}}, \tag{3.6.2}$$

where m_{hw} is the mass of the hot wire, a_{imp} is the impact acceleration the structure can withstand, A_n is the area normal to the force of impact and σ_{max} is the maximum stress the material can withstand.

3.7 Performance analysis

Some important parameters can be analyzed to measure the overall performance of the thermal convective microfluidic gyroscope. A few of these are mentioned in this section.

Gyroscope sensitivity, $\mathscr{S}$ in [mV.s/deg], can be calculated as

$$\mathscr{S} = \frac{V_{out}}{\omega}, \tag{3.7.1}$$

where V_{out} is the Wheatstone bridge output voltage and ω is the angular rate at which the output voltage is measured.

The performances of different sensors are typically compared to each other in the form of performance factors. The performance factor of the microfluidic gyroscope is given as

$$\frac{\mathscr{S}}{V_{in}} = \frac{1}{2}\frac{\alpha \Delta T}{\omega}, \tag{3.7.2}$$

where ΔT is the mean temperature differences of the hot wires due to an angular velocity input ω, V_{in} is the voltage input and α is the temperature coefficient of resistance of the hot wires [10].

The resolution of a gyroscope is the smallest step in angular velocity that can be measured or picked up by the gyroscope. The resolution of a gyroscope typically relies on its sensitivity and the analog equipment used for signal conditioning.

The linear range of a gyroscope is that range of angular velocities for which a proportional relationship can be found between the angular velocity and the voltage output form the Wheatstone half-bridge.

This completes the analytical model of the thermal convective microfluidic gyroscope; we will now proceed to the mathematical optimization of this gyroscope concept.

Chapter 4

Sequential approximate optimization essentials

Sequential approximate optimization (SAO) techniques are algorithmic options for solving non-linear optimization problems by constructing successive approximate analytical functions, called subproblems [15]. These subproblems are constructed at successive approximations $\boldsymbol{x}^{\{k\}}$ to the solution $\boldsymbol{x}^*$ and can be used with any suitable optimization algorithm.

The optimal point of the current subproblem $\mathbf{x}^{\{k*\}}$, becomes the starting point of the next subproblem $\mathbf{x}^{\{k+1\}}$. This process is repeated until the optimal solution of the current subproblem $\mathbf{x}^{\{k*\}}$ becomes equal to the optimal solution of the next subproblem $\mathbf{x}^{\{(k+1)*\}}$, meaning that the solution $\mathbf{x}^{\{k\}}$, where $k = 1,2,3,\cdots$, converges to an optimal solution $\mathbf{x}^*$.

Sampling the real objective and constraint functions during the optimization process will be more accurate than sampling the approximated functions, but this is an expensive and time consuming method. The success and accuracy of SAO depends on the formulation of a suitable approximation to the objective and constraint functions.

4.1 The non-linear optimization problem

The nonlinear inequality-constrained optimization problem is given by

$$\begin{aligned} \min \quad & f_0(\mathbf{x}) \\ \text{subject to} \quad & f_j(\mathbf{x}) \leq 0, \qquad j = 1,2,\ldots,m, \\ & \check{x}_i \leq x_i \leq \hat{x}_i, \qquad i = 1,2,\ldots,n, \end{aligned} \tag{4.1.1}$$

where $f_0(\mathbf{x})$ is the objective function, $f_j(\mathbf{x})$ represents m inequality-constraint functions and $\mathbf{x} = \{x_1, x_2, \cdots, x_n\}^T \in \mathbb{R}^n$ represents n real design variables, while $\check{x}_i$ and $\hat{x}_i$ respectively represent the lower and upper bounds of x_i [16].

4.2 Primal approximate subproblem

For an ordinary nonlinear optimization problem, a suitable approximate continuous subproblem can be generated,

$$\begin{aligned} &\text{minimize } \tilde{f}_0^{\{k\}}(\mathbf{x}) \\ &\text{subject to } \tilde{f}_j^{\{k\}}(\mathbf{x}) \le 0, \qquad j = 1,2,\ldots,m, \qquad (4.2.1) \\ &\qquad\qquad \check{x}_i \le x_i \le \hat{x}_i, \qquad i = 1,2,\ldots,n, \qquad (4.2.2) \end{aligned}$$

where k is the current iteration value and ${f_\alpha}^{\{k\}}(\mathbf{x}), \alpha = 0,1,2,\ldots,m$ is the function value at the current iteration point $\mathbf{x}^{\{k\}}$ [16].

4.3 Approximations

Several approximations were identified and will be introduced in this section.

4.3.1 MMA approximation

Svanberg's well known MMA (method of moving asymptotes) approximation [31] to the objective and constraint functions in (4.1.1) is given as

$$\tilde{f}_j^{\{k\}}(\mathbf{x}) = \sum_{i=1}^{n} \left(\frac{p_{ji}^{\{k\}}}{u_i^{\{k\}} - x_i} - \frac{q_{ji}^{\{k\}}}{x_i - l_i^{\{k\}}} \right) + r_j^{\{k\}}, \qquad j = 0,1,\ldots,m, \qquad (4.3.1)$$

where

$$\begin{aligned} p_{ji}^{\{k\}} &= \left(u_i^{\{k\}} - x_i^{\{k\}}\right)^2 \left\{ \left[\frac{\partial f_j}{\partial x_i}\left(\mathbf{x}^{\{k\}}\right) \right]^+ + \kappa_{ji}^{\{k\}} \right\} \\ q_{ji}^{\{k\}} &= \left(x_i^{\{k\}} - l_i^{\{k\}}\right)^2 \left\{ \left[\frac{\partial f_j}{\partial x_i}\left(\mathbf{x}^{\{k\}}\right) \right]^- + \kappa_{ji}^{\{k\}} \right\} \end{aligned}$$

$$
\begin{aligned}
r_j^{\{k\}} &= f_j(\mathbf{x}^{\{k\}}) - \sum_{i=1}^{n}\left(\frac{p_{ji}^{\{k\}}}{u_i^{\{k\}} - x_i^{\{k\}}} - \frac{q_{ji}^{\{k\}}}{x_i^{\{k\}} - l_i^{\{k\}}}\right) \\
\check{x}_i^{\{k\}} &= \max\{\check{x}_i, 0.9l_i^{\{k\}} + 0.1x_i^{\{k\}}\} \\
\hat{x}_i^{\{k\}} &= \min\{\hat{x}_i, 0.9u_i^{\{k\}} + 0.1x_i^{\{k\}}\} \\
l_i^{\{k\}} &= \begin{cases} x_i^{\{k\}} - 0.5(\hat{x}_i - \check{x}_i) & \text{if } k = 1,2 \\ x_i^{\{k\}} - \gamma_i^{\{k\}}(x_i^{\{k-1\}} - l_i^{\{k-1\}}) & \text{if } k \geq 3 \end{cases} \\
u_i^{\{k\}} &= \begin{cases} x_i^{\{k\}} + 0.5(\hat{x}_i - \check{x}_i) & \text{if } k = 1,2 \\ x_i^{\{k\}} + \gamma_i^{\{k\}}(u_i^{\{k-1\}} - x_i^{\{k-1\}}) & \text{if } k \geq 3 \end{cases} \\
\gamma_{ji}^{\{k\}} &= \begin{cases} 0.7 & \text{if } (x_i^{\{k\}} - x_i^{\{k-1\}})(x_i^{\{k-1\}} - x_i^{\{k-2\}}) < 0 \\ 1.2 & \text{if } (x_i^{\{k\}} - x_i^{\{k-1\}})(x_i^{\{k-1\}} - x_i^{\{k-2\}}) > 0 \\ 1 & \text{if } (x_i^{\{k\}} - x_i^{\{k-1\}})(x_i^{\{k-1\}} - x_i^{\{k-2\}}) = 0 \end{cases} \\
\kappa_{ji}^{\{k\}} &= 10^{-3}\left|\frac{\partial f_j}{\partial x_i}(\mathbf{x}^{\{k\}})\right| + \frac{10^{-6}}{u_i^{\{k\}} - l_i^{\{k\}}}, \qquad i = 1,\ldots,n \text{ and } j = 0,1,\ldots,m.
\end{aligned} \tag{4.3.2}
$$

4.3.2 Incomplete series expansion approximations

Groenwold and Etman [16] derived a number of approximations based on their so-called incomplete series expansions (ISE). These approximation functions will briefly be summarized for direct, reciprocal and exponential intervening variables. It will also be shown how a diagonal quadratic approximation to these approximations can be constructed and provide the user with a wide selection of approximations to the objective and constraint functions.

4.3.2.1 Intervening variables

Direct design variables

The ISE approximation in terms of direct design variables x_i can be given as

$$
\tilde{f}_\alpha(\mathbf{x}) = f_\alpha(\mathbf{x}^{\{k\}}) + \sum_{i=1}^{n}\left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}}(x_i - x_i^{\{k\}}) + \sum_{p=2}^{\bar{p}}\frac{1}{p!}\sum_{i=1}^{n} c_{pi_\alpha}^{\{k\}}\left|x_i - x_i^{\{k\}}\right|^p, \tag{4.3.3}
$$

where $\alpha = 0$ represent the objective function approximation and $\alpha = 1,2,\ldots,m$ represents the inequality constraint function j approximation [16].

A diagonal quadratic approximation can be constructed using (4.3.3) and substituting $\bar{p} = 2$, as higher order approximation usually require immense computational effort and are often unnecessary. This quadratic approximation is given as

$$\tilde{f}_\alpha(\mathbf{x}) = f_\alpha(\mathbf{x}^{\{k\}}) + \sum_{i=1}^{n} \left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}} (x_i - x_i^{\{k\}}) + \frac{1}{2}\sum_{i=1}^{n} c_{2i_\alpha}^{\{k\}} \left(x_i - x_i^{\{k\}}\right)^2, \tag{4.3.4}$$

where $\alpha = 0, 1, 2, \ldots, m$ and $i = 1, 2, \ldots, n$ [16].

Reciprocal intervening variables

To find the approximation in terms of reciprocal intervening variables, (4.3.3) is written in terms of $y_i, i = 1, 2, \ldots, n$. A simple reciprocal term,

$$y_i = \frac{1}{x_i}, i = 1, 2, \ldots, n, \tag{4.3.5}$$

is substituted and the reciprocal approximation becomes

$$\tilde{f}_{\mathrm{R}\alpha} = f_\alpha(\mathbf{x}^{\{k\}}) + \sum_{i=1}^{n} \left(x_i - x_i^{\{k\}}\right) \left(\frac{x_i^{\{k\}}}{x_i}\right) \left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}} + \sum_{p=2}^{\bar{p}} \sum_{i=1}^{n} \frac{c_{pi_\alpha}^{\{k\}}}{p!} \left|\frac{1}{x_i} - \frac{1}{x_i^{\{k\}}}\right|^p, \tag{4.3.6}$$

when written in terms of the original design variables $x_i, i = 1, 2, \ldots, n$ [16].

Higher order reciprocal approximations are hardly ever used, as they would require unnecessary computational effort. A first order linear approximation can be constructed using (4.3.6) and substituting $\bar{p} = 0$. The approximation now becomes

$$\tilde{f}_{\mathrm{R}\alpha} = f_\alpha(\mathbf{x}^{\{k\}}) + \sum_{i=1}^{n} \left(x_i - x_i^{\{k\}}\right) \left(\frac{x_i^{\{k\}}}{x_i}\right) \left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}}. \tag{4.3.7}$$

Exponential intervening variables

To find the approximation in terms of exponential intervening variables, (4.3.3) is written in terms of $y_i, i = 1, 2, \ldots, n$. A simple exponential term,

$$y_i = x_i^{a_i}, i = 1, 2, \ldots, n, \tag{4.3.8}$$

is substituted and the exponential approximation becomes

$$\tilde{f}_{\mathrm{E}\alpha}(\mathbf{x}) = f_\alpha(\boldsymbol{x}^{\{k\}}) + \sum_{i=1}^{n} \left[\left(\frac{x_i}{x_i^{\{k\}}}\right)^{a_{i_\alpha}^{\{k\}}} - 1\right] \left(\frac{x_i^{\{k\}}}{a_{i_\alpha}^{\{k\}}}\right) \left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}} + \sum_{p=2}^{\bar{p}} \sum_{i=1}^{n} \frac{c_{pi_\alpha}^{\{k\}}}{p!} \left|x_i^{a_{i_\alpha}^{\{k\}}} - \left(x_i^{a_{i_\alpha}^{\{k\}}}\right)^{\{k\}}\right|^p, \tag{4.3.9}$$

when written in terms of the original design variables $x_i, i = 1, 2, \ldots, n$ [16].

Again, higher order exponential approximations are hardly ever used, as they would require unnecessary computational effort. A first order linear approximation can be constructed using (4.3.9) and substituting $\bar{p} = 0$. The approximation now becomes

$$\tilde{f}_{\mathrm{E}\alpha}(\mathbf{x}) = f_\alpha(\boldsymbol{x}^{\{k\}}) + \sum_{i=1}^{n} \left[\left(\frac{x_i}{x_i^{\{k\}}} \right)^{a_{i\alpha}^{\{k\}}} - 1 \right] \left(\frac{x_i^{\{k\}}}{a_{i_\alpha}^{\{k\}}} \right) \left(\frac{\partial f_\alpha}{\partial x_i} \right)^{\{k\}}. \tag{4.3.10}$$

4.3.2.2 Diagonal quadratic approximations

The diagonal quadratic approximation from (4.3.4) contains an unknown term $c_{2i_\alpha}^{\{k\}}$ which represents the curvature component of the approximation or the approximate Hessian terms. The approximation is convex if this curvature component $c_{2i_\alpha}^{\{k\}} \geq 0$ for all i [18].

To obtain strictly convex dual subproblems, $c_{2i_\alpha}^{k} = \max(\epsilon_n > 0, c_{2i_\alpha}^{k}) \forall 1$ is enforced for $\alpha = 0$, and $c_{2i_\alpha}^{k} = \max(0, c_{2i_\alpha}^{k}) \forall 1$ is enforced for $\alpha > 0$, with ϵ_n arbitrarily selected as 10^{-5} [18].

Based on the approximation of $c_{2i_\alpha}^{\{k\}}$, several ISE based diagonal quadratic approximations are now possible and we will briefly introduce those implemented in this project.

1. A spherical quadratic approximation based on function values

The spherical quadratic approximation based on function values can be constructed using (4.3.4) and selecting $c_{2i_\alpha}^{\{k\}} \equiv c_{2_\alpha}^{\{k\}} \forall\ i$ [18]. The single unknown term, $c_{2_\alpha}^{\{k\}}$ may be obtained by enforcing the condition

$$f_\alpha(\mathbf{x}^{\{k-1\}}) = \tilde{f}_\alpha(\mathbf{x}^{\{k-1\}}), \tag{4.3.11}$$

which implies that

$$\begin{aligned} f_\alpha(\mathbf{x}^{\{k-1\}}) &= f_\alpha(x^{\{k\}}) + \boldsymbol{\nabla}^T f_\alpha(\boldsymbol{x}^{\{k\}})(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}) + \frac{c_{2_\alpha}^{\{k\}}}{2}(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})^2 \\ c_{2_\alpha}^{\{k\}} &= \frac{2[f_\alpha(\boldsymbol{x}^{\{k-1\}}) - f_\alpha(\boldsymbol{x}^{\{k\}}) - \boldsymbol{\nabla}^T f_\alpha(\boldsymbol{x}^{\{k\}})(\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}})]}{\|\boldsymbol{x}^{\{k-1\}} - \boldsymbol{x}^{\{k\}}\|_2^2}, \end{aligned} \tag{4.3.12}$$

where $\alpha = 0, 1, 2, \ldots, m$ [18].

2. A spherical quadratic approximation based on error norm of the gradients

The spherical quadratic approximation based on the error norm of the gradients can be constructed us-

ing (4.3.4) and selecting $c_{2i_\alpha}^{\{k\}} \equiv c_{2_\alpha}^{\{k\}} \forall\ i$ [18]. The single unknown term, $c_{2_\alpha}^{\{k\}}$ may be obtained by minimizing the least squares error with regard to $c_{2_\alpha}^{\{k\}}$ [19]. The least squares error is given by

$$E_\alpha^{\{k\}} = (\nabla \tilde{f}_\alpha^{\{k\}}(x^{\{k-1\}}) - \nabla f_\alpha(x^{\{k-1\}}))^T (\nabla \tilde{f}_\alpha^{\{k\}}(x^{\{k-1\}}) - \nabla f_\alpha(x^{\{k-1\}})), \tag{4.3.13}$$

where

$$\nabla \tilde{f}_\alpha(x^{\{k-1\}}) = \nabla f_\alpha(\boldsymbol{x}^{\{k\}}) + c_{2_\alpha}^{\{k\}}(x^{\{k-1\}} - x^{\{k\}}). \tag{4.3.14}$$

Substituting (4.3.14) into (4.3.13), the least squares error becomes

$$\begin{aligned} E_\alpha^{\{k\}} \quad = \quad & (\nabla f_\alpha(x^{\{k\}}) + c_{2_\alpha}^{\{k\}}(x^{\{k-1\}} - x^{\{k\}}) - \nabla f_\alpha(x^{\{k-1\}}))^T \\ & (\nabla f_\alpha(x^{\{k\}}) + c_{2_\alpha}^{\{k\}}(x^{\{k-1\}} - x^{\{k\}}) - \nabla f_\alpha(x^{\{k-1\}})). \end{aligned} \tag{4.3.15}$$

Minimizing the least squares error with regard to $c_{2_\alpha}^{\{k\}}$ gives

$$\begin{aligned} \frac{dE_\alpha^{\{k\}}}{dc_\alpha^{\{k\}}} \quad = \quad & (\nabla f_\alpha(x^{\{k\}}) + c_{2_\alpha}^{\{k\}}(x^{\{k-1\}} - x^{\{k\}}) - \nabla f_\alpha(x^{\{k-1\}}))^T (x^{\{k-1\}} - x^{\{k\}}) + \\ & (x^{\{k-1\}} - x^{\{k\}})^T (\nabla f_\alpha(x^{\{k\}}) + c_{2_\alpha}^{\{k\}}(x^{\{k-1\}} - x^{\{k\}}) - \nabla f_\alpha(x^{\{k-1\}})) = 0, \end{aligned} \tag{4.3.16}$$

which leads to

$$c_{2_\alpha}^{\{k\}} = \frac{(x^{\{k-1\}} - x^{\{k\}})^T (\nabla f_\alpha(x^{\{k-1\}}) - \nabla f_\alpha(x^{\{k\}}))}{(x^{\{k-1\}} - x^{\{k\}})^T (x^{\{k-1\}} - x^{\{k\}})}, \tag{4.3.17}$$

where $\alpha = 0, 1, 2, \ldots, m$ [19].

3. A nonspherical approximation based on the components of the gradients

The nonspherical diagonal quadratic approximation based on the components of gradients can be constructed using (4.3.4) and obtaining the n unknown terms, $c_{2i_\alpha}^{\{k\}}$ by means of enforcing the condition

$$\nabla f_\alpha(\mathbf{x}^{\{k-1\}}) = \nabla \tilde{f}_\alpha(\mathbf{x}^{\{k-1\}}), \tag{4.3.18}$$

which implies that

$$\begin{aligned}\nabla f_\alpha(\mathbf{x}^{\{k-1\}}) &= \nabla \tilde{f}_\alpha(\mathbf{x}^{\{k-1\}}) \\ &= \nabla f_\alpha(\mathbf{x}^{\{k\}}) + c_{2i_\alpha}^{\{k\}}(x_i^{\{k-1\}} - x_i^{\{k\}})\end{aligned}$$

$$c_{2i_\alpha}^{\{k\}} = \frac{\nabla f_\alpha(\mathbf{x}^{\{k-1\}}) - \nabla f_\alpha(\mathbf{x}^{\{k\}})}{(x_i^{\{k-1\}} - x_i^{\{k\}})}, \tag{4.3.19}$$

where $i = 1, 2, \ldots, n$ [18].

4. The quadratic Taylor series expansion to the reciprocal approximation

The quadratic Taylor series expansion to the reciprocal approximation can be constructed using (4.3.4) and finding the n unknown terms, $c_{2i_\alpha}^{\{k\}}$.

The second order partial derivatives $c_{2i_\alpha}^{\{k\}}$ for (4.3.7) are obtained as

$$c_{2i_\alpha}^{\{k\}} = \frac{\partial^2 \tilde{f}_{\mathrm{R}_\alpha}}{\partial x_i^2}\left(\mathbf{x}^{\{k\}}\right) = \frac{-2\left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}}}{x_i^{\{k\}}}, \tag{4.3.20}$$

where $i = 1, 2, \ldots, n$ and $\alpha = 0, 1, 2, \ldots, m$ [17].

5. The quadratic Taylor series expansion to the exponential approximation

The quadratic Taylor series expansion to the exponential approximation can be constructed using (4.3.4) and finding the n unknown terms $c_{2i_\alpha}^{\{k\}}$ and $a_{i_\alpha}^{\{k\}}$.

The second order partial derivatives $c_{2i_\alpha}^{\{k\}}$ for (4.3.10) are obtained as

$$c_{2i_\alpha}^{\{k\}} = \frac{\partial^2 \tilde{f}_{\mathrm{E}_\alpha}}{\partial x_i^2}\left(\mathbf{x}^{\{k\}}\right) = \frac{\left(a_{i_\alpha}^{\{k\}} - 1\right)}{\left(x_i^{\{k\}}\right)}\left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}}, \tag{4.3.21}$$

where $i = 1, 2, \ldots, n$ and $\alpha = 0, 1, 2, \ldots, m$ [17].

The unknown exponential terms $a_{i_\alpha}^{\{k\}}$ may be obtained by enforcing

$$\nabla \tilde{f}_{\mathrm{E}_\alpha}(\boldsymbol{x}^{\{k-1\}}) = \nabla f_\alpha(\boldsymbol{x}^{\{k-1\}}), \tag{4.3.22}$$

using the exponential approximation (4.3.10) [17]. This results in

$$a_{i_\alpha}^{\{k\}} = 1 + \ln\left\{\left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k-1\}} / \left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}}\right\} / \ln\left\{\left(x_i^{\{k-1\}}\right) / \left(x_i^{\{k\}}\right)\right\}, i = 1,2,\cdots,n, \tag{4.3.23}$$

where $\alpha = 0,1,2,\ldots,m$ and $\ln(\cdot)$ indicates the natural logarithm [17].

To ensure that (4.3.10) is convex, we enforce $a_i^{\{k\}} \leq 1$ if the derivatives $\partial f/\partial x_i$ are negative [17].

6. The quadratic Taylor series expansion to the MMA approximations of Svanberg

The quadratic Taylor series expansion to the MMA approximation of Svanberg can be constructed by using (4.3.4) and finding the n unknown terms $c_{2i_\alpha}^{\{k\}}$.

The MMA approximation of Svanberg contains reciprocal-like variables

$$y_i = \left(x_i - L_i^{\{k\}}\right)^{-1} \qquad \text{if} \qquad \frac{\partial f_\alpha^{\{k\}}}{\partial x_i} < 0, \tag{4.3.24}$$

or

$$y_i = \left(U_i^{\{k\}} - x_i\right)^{-1} \qquad \text{if} \qquad \frac{\partial f_\alpha^{\{k\}}}{\partial x_i} > 0, \tag{4.3.25}$$

where $L_i^{\{k\}}$ and $U_i^{\{k\}}$ are the lower and upper movable asymptotes respectively [17] and can be calculated by (4.3.2).

For the MMA intervening variables, the second order partial derivatives $c_{2i_\alpha}^{\{k\}}$ are obtained by

$$c_{2i_\alpha}^{\{k\}} = \left(\frac{\partial^2 \tilde{f}_{\mathrm{MMA}_\alpha}}{\partial x_i^2}\right)^{\{k\}} = \frac{-2}{\left(x_i^{\{k\}} - L_i^{\{k\}}\right)}\left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}} > 0, \qquad \text{for} \qquad \partial f_\alpha^{\{k\}}/\partial x_i < 0, \tag{4.3.26}$$

or

$$c_{2i_\alpha}^{\{k\}} = \left(\frac{\partial^2 \tilde{f}_{\mathrm{MMA}_\alpha}}{\partial x_i^2}\right)^{\{k\}} = \frac{2}{\left(U_i^{\{k\}} - x_i^{\{k\}}\right)}\left(\frac{\partial f_\alpha}{\partial x_i}\right)^{\{k\}} > 0, \qquad \text{for} \qquad \partial f_\alpha^{\{k\}}/\partial x_i > 0, \tag{4.3.27}$$

where $i = 1,2,\ldots,n$ and $\alpha = 0,1,2,\ldots,m$ [17].

4.3.3 Comparison between approximations

FORTRAN implementations of the MMA approximation [31] and the ISE approximations [15] to the subproblem in (4.2.1) were constructed by their authors. These FORTRAN algorithms will be implemented, compared and further discussed in Chapter 5. An algorithmic implementation of SAO, using the ISE

approximations, will also be coded in MATLAB and the results will be compared to existing FORTRAN implementations.

4.4 Duality

The primal approximate subproblem in (4.2.1) can be developed into a dual approximate subproblem, using the Lagrangian method to convert an inequality-constrained problem into an unconstrained problem. The Lagrangian can be constructed as

$$\mathscr{L}^{\{k\}}(\mathbf{x},\boldsymbol{\lambda}) = \tilde{f}_0^{\{k\}}(\mathbf{x}) + \sum_{j=1}^{m} \lambda_j \tilde{f}_j^{\{k\}}(\mathbf{x}), \tag{4.4.1}$$

where $\lambda_j, j = 1,2,\ldots,m$ represents the Lagrangian multipliers and gives and indication of the sensitivity of $\mathscr{L}^{\{k\}}(\mathbf{x},\boldsymbol{\lambda})$ to the constraint j [16].

If the primal approximate subproblem (4.2.1) is strictly convex, the global minimizer $\mathbf{x}^{\{k*\}}$ of the approximate subproblem is define by the stationary saddle point $(\mathbf{x}^{\{k*\}}, \boldsymbol{\lambda}^{\{k*\}})$ [16].

The Karush-Kuhn-Tucker (KKT) stationary conditions are defined for problems that are strictly convex and need to be satisfied at the point $\mathbf{x}^*$, which represents the global minimizer of the primal subproblem in (4.2.1). The KKT conditions that need to be satisfied are given as

$$\begin{aligned}
\tilde{f}_j(\mathbf{x}^*) &\leq 0, \quad j = 1,2,\ldots,m,\\
\lambda_j^* \tilde{f}_j(\mathbf{x}^*) &= 0, \quad j = 1,2,\ldots,m,\\
\lambda_j^* &\geq 0, \quad j = 1,2,\ldots,m,\\
\frac{\partial \tilde{f}_0}{\partial x_i}(\mathbf{x}^*) + \sum_{j=1}^{m} \lambda_j^* \frac{\partial \tilde{f}_j}{\partial x_i}(\mathbf{x}^*) &= 0, \quad i = 1,2,\ldots,n,
\end{aligned} \tag{4.4.2}$$

and require $f_\alpha(\mathbf{x}), \alpha = 0,1,2,\ldots,m$ to be at least once continuously differentiable for all α [16].

If the primal subproblem is strictly convex and separable, the saddle point (x^*, λ^*) may be found using the Falk dual [16]

$$\max_{\boldsymbol{\lambda}} \min_{\boldsymbol{x}} \{\mathscr{L}^{\{k\}}(\boldsymbol{x},\boldsymbol{\lambda}) : \check{x}_i \leq x_i \leq \hat{x}_i\} = \max_{\boldsymbol{\lambda}} \gamma(\boldsymbol{\lambda}). \tag{4.4.3}$$

which is defined over a set of bound constraints.

The dual approximate subproblem is formulated by maximizing the Lagrangian over the dual variables (or Lagrangian multipliers)

$$\max_{\boldsymbol{\lambda}} \ \gamma(\boldsymbol{\lambda}) = \tilde{f}_0(\mathbf{x}(\boldsymbol{\lambda})) + \sum_{j=1}^{m} \lambda_j \tilde{f}_j(\mathbf{x}(\boldsymbol{\lambda})) \tag{4.4.4}$$

$$\text{subject to } \lambda_j \geq 0, \qquad j = 1,2,\ldots,m. \tag{4.4.5}$$

and $\tilde{f}_\alpha(\mathbf{x}(\boldsymbol{\lambda})), \alpha = 0,1,2,\ldots,m$ can be represented by any suitable approximation. In using the diagonal quadratic approximations, described in Section 4.3, we can obtain $\gamma(\boldsymbol{\lambda})$ as

$$\begin{aligned}\gamma(\boldsymbol{\lambda}) = f_0(\boldsymbol{x}^{\{k\}}) + \sum_{i=1}^{n} \frac{\partial f_0^{\{k\}}}{\partial x_i}(x_i(\boldsymbol{\lambda}) - x_i^{\{k\}}) + \frac{1}{2}\sum_{i=1}^{n} c_{2i_0}^{\{k\}}(x_i(\boldsymbol{\lambda}) - x_i^{\{k\}})^2 \\ + \sum_{j=1}^{m} \lambda_j \left(f_j(\boldsymbol{x}^{\{k\}}) + \sum_{i=1}^{n} \frac{\partial f_j^{\{k\}}}{\partial x_i}\left(x_i(\boldsymbol{\lambda}) - x_i^{\{k\}}\right) + \frac{1}{2}\sum_{i=1}^{n} c_{2i_j}^{\{k\}}(x_i(\boldsymbol{\lambda}) - x_i^{\{k\}})^2 \right).\end{aligned} \tag{4.4.6}$$

An analytical relationship can be found between the design variables $\mathbf{x}_i$ and the dual variables $\boldsymbol{\lambda}_j$ in order to satisfy the stationary KKT conditions for strictly convex problems

$$\frac{\partial \tilde{f}_0}{\partial x_i}(\mathbf{x}^*) + \sum_{j=1}^{m} \lambda_j^* \frac{\partial \tilde{f}_j}{\partial x_i}(\mathbf{x}^*) = 0 \tag{4.4.7}$$

where $i = 1,2,\ldots,n,$. The design variables $\mathbf{x}_i$ are found in terms of the dual variables $\boldsymbol{\lambda}_j$

$$x_i(\boldsymbol{\lambda}) = \begin{cases} \beta_i(\boldsymbol{\lambda}) & \text{if} \quad \check{x}_i < \beta_i(\boldsymbol{\lambda}) < \hat{x}_i \\ \check{x}_i & \text{if} \quad \beta_i(\boldsymbol{\lambda}) \leq \check{x}_i \\ \hat{x}_i & \text{if} \quad \beta_i(\boldsymbol{\lambda}) \geq \hat{x}_i \end{cases} \tag{4.4.8}$$

and can be used to solve the dual approximate subproblem (4.4.5), where

$$\beta_i(\boldsymbol{\lambda}) = x_i^{\{k\}} - \left(c_{2i_0}^{\{k\}} + \sum_{j=1}^{m} \lambda_j c_{2i_j}^{\{k\}} \right)^{-1} \left(\frac{\partial f^{\{k\}}}{\partial x_i} + \sum_{j=1}^{m} \lambda_j \frac{\partial f_j^{\{k\}}}{\partial x_i} \right), \tag{4.4.9}$$

for $i = 1,2,\ldots,n$ and $\bar{p} = 2$ [16].

An algorithmic optimization solver, such as BFGS or the conjugate gradients method, is used to solve the dual approximate subproblem. These are further discussed in Section 4.7.

4.5 Conservatism and its effect on convergence

Conservatism is a method for enforcing convergence to an optimal solution $\mathbf{x}^*$, if a feasible descent step cannot be made [18]. An iterate k is defined conservative if $\tilde{f}_j^{\{k*\}} \geq f_j^{\{k*\}}$, for $j = 0,1,2,\ldots,m$ [17]. This means that the approximate function values $\tilde{f}_j^{\{k*\}}$ are always larger than, or equal to, the true function values $f_j^{\{k*\}}$, for a conservative iterate.

In the ISE based approximations, a positive curvature component, $c_{2i\alpha}^{\{k\}} \geq 0$ for all i, will ensure strictly convex approximations. If the approximate diagonal curvatures $c_{2i\alpha}^{\{k\}}$ are increased, the conservatism of the approximations will also be increased and vice versa [18]. Larger curvatures or greater conservatism will increase the computational effort to converge to an optimal solution. In using larger curvatures, the optimal solution of the curvature at the current iterate will be closer to the optimal solution at the previous iterate and more iterates will be necessary for convergence.

In the MMA approximation, conservatism is manipulated by adjusting the movable asymptotes and this ensures that the approximations always remain convex. Again, the more the asymptotes are tightened, the greater the computational effort.

In both the FORTRAN implementations of Groenwold and Svanberg an inner SAO loop is introduced to adjust conservatism and thereby force convergence. During the inner loops, Groenwold increases the curvatures [17] and Svanberg tightens the asymptotes of all approximations until the conservatism requirements are satisfied.

Algorithmic implementation to ensure a conservative approximation

Consider the algorithm used by Groenwold and co-workers [18] to ensure that the approximations are conservative.

Taken verbatim from the paper by Groenwold and co-workers [18]:

> Given an initial point $\boldsymbol{x}^{\{0\}}$, a conservative algorithm based on convex separable diagonal quadratic approximations proceeds as follows (using a FORTRAN-like pseudo-language):
>
> 1. **Initialization:** Select positive constants ε_1, ε_2, ε_x, $\hat{k}$, $\chi_1 > 1$, $\chi_2 > 1$. Set $k := 0$, $l := 0$.
> 2. **Simulation and sensitivity analysis:** Compute $f_j(\boldsymbol{x}^{\{0\}})$, $\boldsymbol{\nabla} f_j(\boldsymbol{x}^{\{0\}})$, $j = 0,1,2,\cdots,m$.
> 3. **Construct the approximate curvatures:** Calculate the initial outer curvatures $c_{2i_0}^{\{k\}} > 0$ and $c_{2i_j}^{\{k\}} \geq 0$, $j = 1,2,\cdots,m$.
> 4. **Approximate optimization:** Construct local approximate subproblem $P_D[k]$ at $\boldsymbol{x}^{\{k\}}$. Solve this subproblem to arrive at $(\boldsymbol{x}^{\{k*\}}, \boldsymbol{\lambda}^{\{k*\}})$.
> 5. **Simulation analysis:** Compute $f_j(\boldsymbol{x}^{\{k*\}})$, $j = 0,1,2,\cdots,m$.
> 6. **Test if $\boldsymbol{x}^{\{k*\}}$ is acceptable:**

a) **test if $x^{\{k*\}}$ represents a feasible descent step:** IF $f_0(x^{\{k*\}}) < f_0(x^{\{(k-1)*\}})$ for $k > 0$, AND $\max\{f_j(x^{\{k*\}})\} \leq 0$, $j = 1, 2, \cdots, m$, GOTO Step 8,

b) **test if $x^{\{k*\}}$ represents a conservative step:** IF $\tilde{f}_0(x^{\{k*\}}) \geq (f_0(x^{\{k*\}}) - \varepsilon_1)$, AND $\tilde{f}_j(x^{\{k*\}}) \geq (f_j(x^{\{k*\}}) - \varepsilon_2)$, $j = 1, 2, \cdots, m$, GOTO Step 8.

7. **Initiate an inner loop to effect conservatism:**

a) Set $l := l + 1$.

b) IF $\tilde{f}_0(x^{\{k*\}}) < (f_0(x^{\{k*\}}) - \varepsilon_1)$, set $c_{2i_0}^{\{k\}} := \chi_1 c_{2i_0}^{\{k\}}$.

c) IF $\tilde{f}_j(x^{\{k*\}}) < (f_j(x^{\{k*\}}) - \varepsilon_2)$, set $c_{2i_j}^{\{k\}} := \chi_2 c_{2i_j}^{\{k\}}$, $j = 1, 2, \cdots, m$.

d) GOTO Step 4.

8. **Move to the new iterate:** Set $x^{\{k+1\}} := x^{\{k*\}}$.

9. **Convergence test:** IF $\|x^{\{k+1\}} - x^{\{k\}}\| \leq \varepsilon_x$, OR $k = \hat{k}$, STOP.

10. **Simulation sensitivity analysis:** Compute $\nabla f_j(x^{\{k+1\}})$, $j = 0, 1, 2, \cdots, m$.

11. **Initiate an additional outer loop:** Set $k := k + 1$ and GOTO Step 3.

4.6 Scaling

Any optimization algorithm will struggle to converge to the true optimal solution, due to extreme distortion of the objective function contours as a results of poor scaling [29]. During formulation of the optimization problem, care must be taken to ensure that design variables are more or less of the same order of magnitude. Step length selection and calculation of numerical gradients often introduce difficulties if the design variables are not properly scaled.

It also becomes necessary to scale the constraint functions if they differ by large magnitudes. Normalization is a popular method of scaling the constraint function and proves to be quite effective in primal algorithms. This is not necessarily true for dual algorithms, as the magnitudes of the Lagrangian multipliers are unknown.

4.7 Optimization solvers

Line search descent algorithms for unconstrained minimization of general functions require some initial estimate $\mathbf{x}^0$ to the optimum point $\mathbf{x}^*$. From this initial point the algorithm searches in the direction of descent to determine the next point. The algorithm repeats itself to generate successive solutions, $\mathbf{x}^0, \mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^*$ until a local minima is reached and $\nabla f(\mathbf{x}) = 0$.

The direction of descent at current iterate $x^{\{k\}}$, denoted by $u^{\{k+1\}}$, must be selected to satisfy a negative directional derivative

$$\frac{df(\mathbf{x}^{\{k\}})}{d\lambda}|_{\mathbf{u}^{\{k+1\}}} = \nabla^T f(\mathbf{x}^{\{k\}})\mathbf{u}^{\{k+1\}} < 0, \tag{4.7.1}$$

to ensure descent in the direction $u^{\{k+1\}}$ at the iterate $x^{\{k\}}$. A first order line search method uses the first order gradient vector $\nabla f(\mathbf{x})$ to determine the direction of descent, while second order methods use Newton's methods for solving $\nabla f(\mathbf{x}) = 0$ iteratively [29].

Two popular line search methods used for solving unconstrained minimization problems will be discussed in this section, the first order *conjugate gradient (CG)* method and the second order *Broyden-Fletcher-Goldfarb-Shanno (BFGS)* method.

4.7.1 Conjugate gradient method

The conjugate gradient method will converge exactly in a finite number of iterations, when applied to a positive-definite quadratic function [29]. This method will also perform well near a local minima for other non-quadratic functions, as many show the quadratic form near the local minima [29].

The Fletcher-Reeves conjugate gradient algorithm for general functions [29] is given as follows:

1. Choose initial estimate to the optimum point $\mathbf{x}^0$ and positive tolerances ε_1, ε_2 and ε_3.
2. Compute $\nabla f(\mathbf{x}^0)$ and set $\mathbf{u}^1 = -\nabla f(\mathbf{x}^0)$.
3. For $k = 1, 2, \ldots, n$ do:
   a) Set $\mathbf{x}^{\{k\}} = \mathbf{x}^{\{k-1\}} + \lambda^{\{k\}}\mathbf{u}^{\{k\}}$, where $\lambda^{\{k\}}$ is such that $f(\mathbf{x}^{\{k-1\}} + \lambda^{\{k\}}\mathbf{u}^{\{k\}}) = \min_\lambda f(\mathbf{x}^{\{k-1\}} + \lambda^{\{k\}}\mathbf{u}^{\{k\}})$ (line search).
   b) Compute $\nabla f(\mathbf{x}^{\{k\}})$.
   c) Test if convergence criteria is satisfied:
   if $\| \mathbf{x}^{\{k\}} - \mathbf{x}^{\{k-1\}} \| < \varepsilon_1$, or $\| \nabla f(\mathbf{x}^{\{k\}}) \| < \varepsilon_2$, or $| f(\mathbf{x}^{\{k\}}) - f(\mathbf{x}^{\{k-1\}}) | < \varepsilon_3$, then STOP and $\mathbf{x}^* \cong \mathbf{x}^{\{k\}}$, *else* go to Step 3d.
   d) Compute new descent direction: *if* $1 \leq k \leq n-1$, then $\mathbf{u}^{\{k+1\}} = -\nabla f(\mathbf{x}^{\{k\}}) + \beta^{\{k\}}\mathbf{u}^{\{k\}}$ with $\beta^{\{k\}}$ given by

$$\beta^{\{k\}} = \frac{\| \nabla f(\mathbf{x}^{\{k\}}) \|^2}{\| \nabla f(\mathbf{x}^{\{k-1\}}) \|^2}$$

   else $\beta^{\{k\}} = 0$.
4. Set $\mathbf{x}^0 = \mathbf{x}^n$ and go to Step 3 (restart).

4.7.2 BFGS method

Second order line search methods are not always convergent, but are quadratically convergent when they do converge [29]. These methods require the evaluation of the Hessian matrix $\mathbf{H}(\mathbf{x})$ at each iteration step, which requires great computational effort [29]. The BFGS algorithm (as all Quasi-Newton methods) use approximations to the Hessian matrix $\mathbf{H}(\mathbf{x})$ and these are updated and re-evaluated after each iteration step [29].

The BFGS algorithm for general functions [29] is given as follows:

1. Choose initial estimate to the optimum point $\mathbf{x}^0$ and positive tolerances ε_1, ε_2 and ε_3.
2. Set $\mathbf{G}^0 = \mathbf{I}$, where $\mathbf{G}^0$ is the initial approximation to the Hessian matrix $\mathbf{H}(\mathbf{x})$.
3. Do for iteration $k = 1, 2, \ldots, n$:
   a) Set $\mathbf{x}^{\{k\}} = \mathbf{x}^{\{k-1\}} + \lambda^{\{k\}}\mathbf{u}^{\{k\}}$, where $\mathbf{u}^{\{k\}} = -\mathbf{G}^{\{k-1\}}\nabla f(\mathbf{x}^{\{k-1\}})$ and $\lambda^{\{k\}}$ is such that $f(\mathbf{x}^{\{k-1\}} + \lambda^{\{k\}}\mathbf{u}^{\{k\}}) = \min_\lambda f(\mathbf{x}^{\{k-1\}} + \lambda^{\{k\}}\mathbf{u}^{\{k\}}), \lambda^{\{k\}} \geq 0$ (line search).
   b) Test if convergence criteria is satisfied:
   if $\| \mathbf{x}^{\{k\}} - \mathbf{x}^{\{k-1\}} \| < \varepsilon_1$, or $\| \nabla f(\mathbf{x}^{\{k\}}) \| < \varepsilon_2$, or $| f(\mathbf{x}^{\{k\}}) - f(\mathbf{x}^{\{k-1\}}) | < \varepsilon_3$, then STOP and $\mathbf{x}^* \cong \mathbf{x}^{\{k\}}$, *else* go to Step 3c.
   c) Set $\mathbf{v}^{\{k\}} = \lambda^{\{k\}}\mathbf{u}^{\{k\}}$ and set $\mathbf{y}^{\{k\}} = \nabla f(\mathbf{x}^{\{k\}}) - \nabla f(\mathbf{x}^{\{k-1\}})$.
   d) Compute $\mathbf{G}^{\{k\}}$ used for new descent direction:

$$\mathbf{G}^{\{k\}} = \mathbf{G}^{\{k-1\}} + \left[1 + \frac{\mathbf{y}^{\{k\}T}\mathbf{G}^{\{k-1\}}\mathbf{y}^{\{k\}}}{\mathbf{v}^{\{k\}T}\mathbf{y}^{\{k\}}}\right]\left[\frac{\mathbf{v}^{\{k\}}\mathbf{v}^{\{k\}T}}{\mathbf{v}^{\{k\}T}\mathbf{y}^{\{k\}}}\right] - \left[\frac{\mathbf{v}^{\{k\}}\mathbf{y}^{\{k\}T}\mathbf{G}^{\{k-1\}} + \mathbf{G}^{\{k-1\}}\mathbf{y}^{\{k\}}\mathbf{v}^{\{k\}T}}{\mathbf{v}^{\{k\}T}\mathbf{y}^{\{k\}}}\right].$$

4. Set $\mathbf{x}^0 = \mathbf{x}^n$ and $\mathbf{G}^0 = \mathbf{I}$. Go to Step 3 (restart).

4.8 Algorithmic implementation of SAO in MATLAB

Based on the algorithmic implementation using conservatism in FORTRAN (refer to Section 4.5), the SAO algorithm was implemented in MATLAB. A BFGS solver (*projbfgs.m* implemented by C.T. Kelley [21]) is called to solve for the dual variables.

ISE based approximations, described in Section 4.3.2, were implemented in order to have a thorough comparison of implementations and a variety of optimization techniques to test for.

A flow diagram is given in Figure 4.1 and MATLAB code is documented in Appendix C.

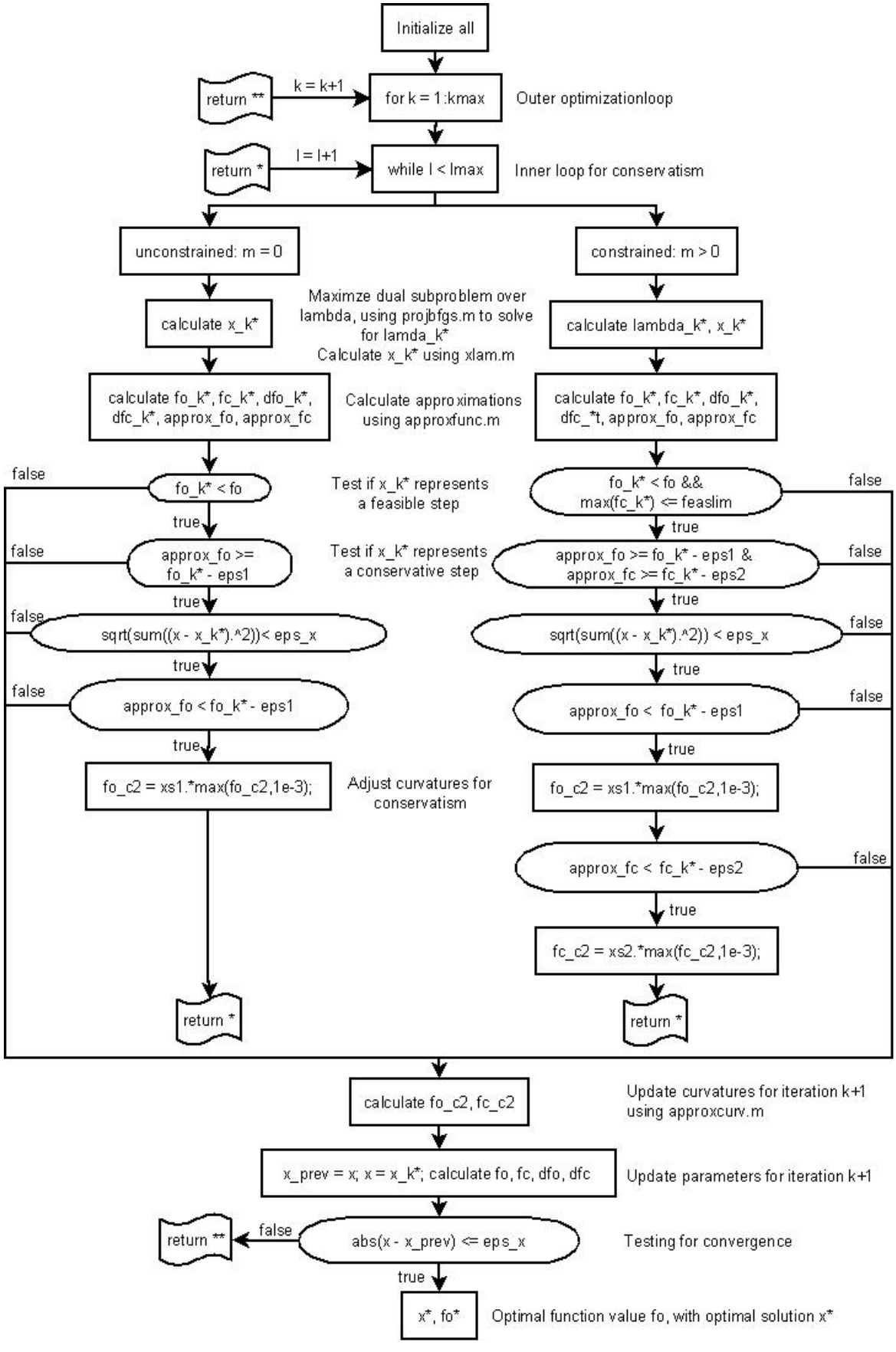


Figure 4.1: Flow diagram of algorithmic implementation of SAO using conservatism in MATLAB

4.9 Test problems

To see how the approximation functions and optimization solvers compare to one another, the FORTRAN implementations of Svanberg [31] and Groenwold [15] were applied to two popular test problems proposed by Svanberg. These test problems will also be solved using the MATLAB implementation of SAO, with the BFGS solver, and a comparison of the results will be an indication of the accuracy of the implementation.

Abbreviations describing each approximation, are listed in Table 4.1 and will be used from here on.

Table 4.1: Approximation abbreviations

Abbreviation	Approximation
SQ1	Spherical quadratic approximation based on function values
SQ2	Spherical quadratic approximation based on error norm of the gradients
NSQ	Nonspherical approximation based on components of the gradients
T2:reciprocal	Quadratic Taylor series expansion to the reciprocal approximation
T2:exponential	Quadratic Taylor series expansion to the exponential approximation
T2:MMA	Quadratic Taylor series expansion to the MMA approximation
MMA	MMA approximation of Svanberg

4.9.1 Svanberg's 5-variate cantilever beam

A standard weight minimization problem was proposed by Svanberg [30], with sizing design variables, subject to a single displacement behavior constraint. This problem may analytically be expressed as

$$\begin{aligned} \min_{\mathbf{x}} f_0(\mathbf{x}) &= c_1 \sum_{i=1}^{5} x_i, \\ \text{subject to } f_1(\mathbf{x}) &= \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - c_2 \leq 0, \\ & 0 < x_i, \quad i = 1,2,\ldots 5, \end{aligned}$$

with $c_1 = 0.0624$ and $c_2 = 1.0$. Results comparing several different approximations, with existing FORTRAN implementations, are presented in Table 4.2 and the convergence histories are depicted in Figure 4.2.

The number of iterations used for this problem, should be almost equal for BFGS and CG algorithms. Small differences occur due to rounding, tolerances and machine precision.

The MATLAB implementation, using a BFGS solver, compares well to the FORTRAN implementation and Table 4.3 compares the function values and iteration history.

Table 4.2: Cantilever beam problem results using several different approximations

Approximation	Solver	$f^{\{*\}}$	Maximum constraint	Outer Iterations	Inner Iterations	KKT residual
SQ1	BFGS	1.3399564	-0.74×10^{-11}	11	19	3.21×10^{-7}
SQ1	CG	1.3399564	-0.37×10^{-12}	12	19	3.92×10^{-7}
SQ2	BFGS	1.3399564	0.259×10^{-11}	12	18	2.44×10^{-7}
SQ2	CG	1.3399564	0.127×10^{-11}	12	18	2.44×10^{-7}
NSQ	BFGS	1.3399564	-0.41×10^{-10}	8	19	3.36×10^{-9}
NSQ	CG	1.3399564	-0.28×10^{-10}	8	19	3.36×10^{-9}
T2:reciprocal	BFGS	1.3399564	-0.68×10^{-10}	9	12	2.24×10^{-7}
T2:reciprocal	CG	1.3399564	0.387×10^{-11}	9	11	5.21×10^{-13}
T2:exponential	BFGS	1.3399564	-0.12×10^{-8}	10	4	2.46×10^{-7}
T2:exponential	CG	1.3399564	-0.94×10^{-16}	10	4	2.46×10^{-7}
T2:MMA	BFGS	1.3399564	0.377×10^{-9}	19	15	7.57×10^{-7}
T2:MMA	CG	1.3399564	0.706×10^{-10}	19	15	7.57×10^{-7}
MMA	Combination	1.3399564	0.769×10^{-11}	18	18	1.7×10^{-7}

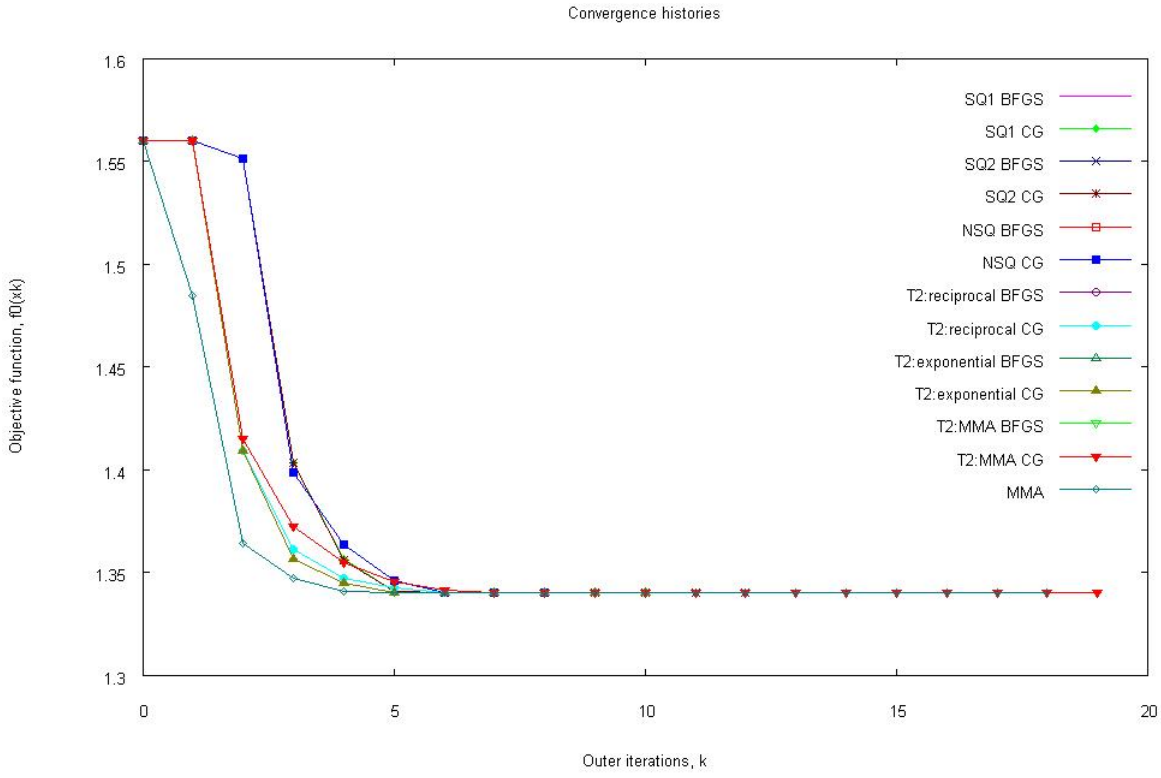


Figure 4.2: Convergence histories of the cantilever beam problem using several different approximations

In theory, using SAO with a BFGS solver should result in iteration sizes equal for both MATLAB and FORTRAN implementations of the same algorithm. Small inaccuracies occur due to the fact that two different compilers are being used to solve the optimization problem and two different implementations of the

BFGS algorithm are used. Outer iterations compare better than inner iterations, which also suggests that inaccuracies occur at a solver level.

Table 4.3: Comparison of FORTRAN and MATLAB implementations applied to the cantilever beam problem

		FORTRAN		MATLAB	
Approximation	$f^{\{*\}}$	Outer Iterations	Inner Iterations	Outer Iterations	Inner Iterations
SQ1	1.3399564	11	19	13	6
SQ2	1.3399564	12	18	12	5
NSQ	1.3399564	8	19	9	6
T2:reciprocal	1.3399564	9	12	9	10
T2:exponential	1.3399564	10	4	10	4
T2:MMA	1.3399564	19	15	19	14

4.9.2 Svanberg's snake problem

Svanberg proposed the *snake problem* [32] for anyone who wants to test a new method of nonlinear optimization. This problem may analytically be expressed as

$$\begin{aligned}
\min_{\mathbf{x}} f_0(\mathbf{x}) &= \sum_{i=1}^{d} (x_i \cos\psi_i + x_{d+i} \sin\psi_i - 0.1 x_{2d+1}) \\
\text{subject to } & \sum_{i=1}^{d} (x_{d+i}^2 + x_i^2) \le d \\
& -2 \le g_i(x) + g_i(x)^7 \le 2, \qquad i = 1,2,\ldots,d \\
& -2 \le h_i(x) + h_i(x)^7 \le 2, \qquad i = 1,2,\ldots,d \\
& -2 \le x_j \le 2, \qquad j = 1,2,\ldots,3d,
\end{aligned} \tag{4.9.1}$$

with

$$\psi_i = \frac{(3i-2d)\pi}{6d}, \quad g_i(x) = \frac{x_i^2 + x_{d+i}^2 - 1}{\delta_s} \quad \text{and} \quad h_i(x) = \frac{x_{2d+i} - 2x_i x_{d+i}}{\delta_s}. \tag{4.9.2}$$

Let d be a given positive integer, selected $d = 10$ and let δ_s be a small positive real number, selected $\delta_s = 0.1$. The initial values to the problem are given as

$$x_i^{\{0\}} = \cos(\psi_i + \frac{\pi}{12}),\ \ x_{d+i}^{\{0\}} = \sin(\psi_i + \frac{\pi}{12}) \ \text{ and } \ x_{2d+i}^{\{0\}} = \sin(2\psi_i + \frac{\pi}{6}), \tag{4.9.3}$$

where $i = 1, 2, \ldots, d$.

Results comparing several different approximations are presented in Table 4.4 and the convergence histories are depicted in Figure 4.3.

Table 4.4: Snake problem results using several different approximations

Approximation	Solver	$f^{\{*\}}$	Maximum constraint	Outer Iterations	Inner Iterations	KKT residual
SQ1	BFGS	-10.02298	0.1588×10^{-5}	51	355	5.57×10^{-6}
SQ1	CG	-10.02298	0.3122×10^{-12}	53	356	1.02×10^{-7}
SQ2	BFGS	-10.02298	0.4248×10^{-5}	42	321	3.71×10^{-6}
SQ2	CG	-10.02298	0.1037×10^{-10}	48	356	4.68×10^{-6}
NSQ	BFGS	-10.02298	0.9827×10^{-6}	395	11378	7.61×10^{-5}
NSQ	CG	-10.02298	0.5192×10^{-10}	260	7234	1.99×10^{-2}
T2:MMA	BFGS	-10.02298	0.1170×10^{-4}	91	1228	5.47×10^{-4}
T2:MMA	CG	-10.02298	0.6240×10^{-11}	109	1143	3.47×10^{-6}
MMA	Combination	-10.02298	0.7286×10^{-6}	76	481	3.91×10^{-7}

This optimization problem is far more difficult to solve than the previous one and the inaccuracies occurring in the number of iterations, become more significant.

The MATLAB implementation, using a BFGS solver, compares well to the FORTRAN implementation and Table 4.5 compares the function values and iteration history. Iteration inaccuracies are becoming more significant as the problem difficulty increases. The optimal function values and design variables remain accurate throughout all test problems.

Both MATLAB and FORTRAN implementation call the BFGS algorithm to solve for the optimal dual variables. The BFGS implementation differs for MATLAB and FORTRAN code and would yield small inconsistencies in tolerances, rounding and logic. Machine precision would differ for MATLAB and FORTRAN compilers and would also have a significant impact on dual variables.

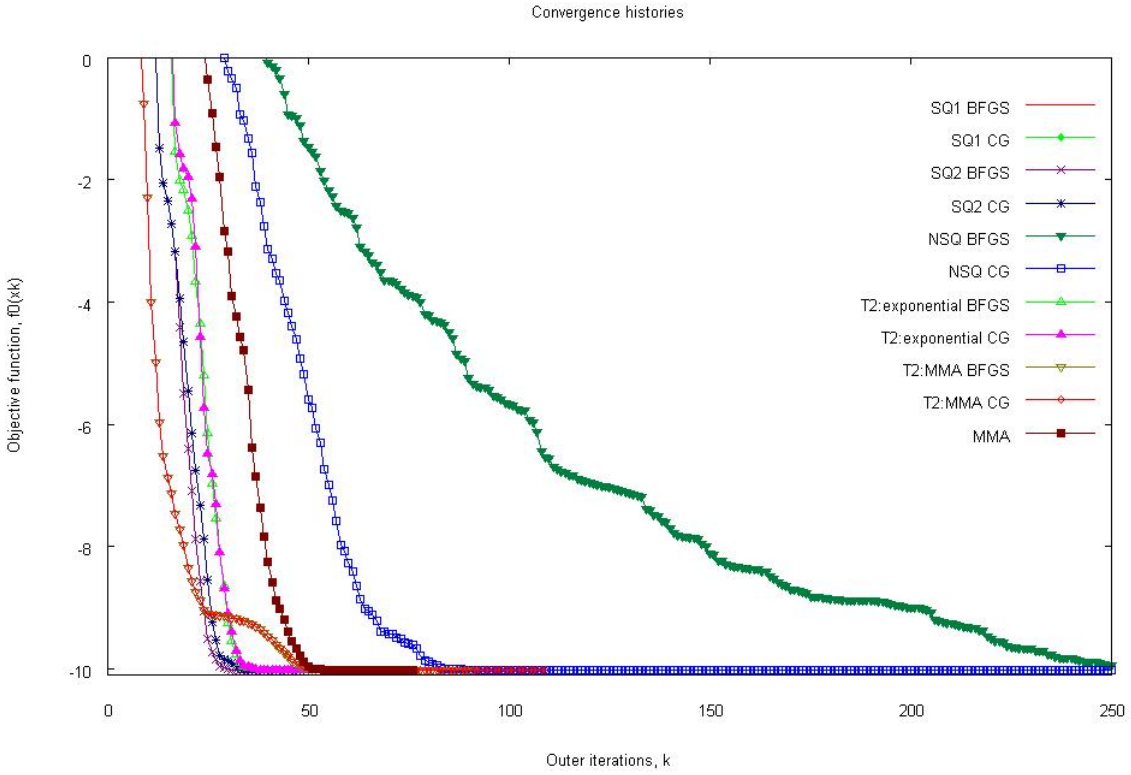


Figure 4.3: Convergence histories of the snake problem using several different approximations

Table 4.5: Comparison of FORTRAN and MATLAB implementations applied to the snake problem

		FORTRAN		MATLAB	
Approximation	$f^{\{*\}}$	Outer Iterations	Inner Iterations	Outer Iterations	Inner Iterations
SQ1	−10.02298	51	355	105	1504
SQ2	−10.02298	42	321	134	1306
NSQ	−10.02298	395	11378	129	1577
T2:MMA	−10.02298	91	1228	89	1176

Chapter 5

Mathematical optimization and design

In this chapter, the analytical model presented in Chapter 3 is used to formulate the optimization problem for the detecting chamber of the microfluidic gyroscope.

Sequential approximate optimization will be used to solve the optimization problem, through means of the MATLAB and FORTRAN implementations of MMA and ISE-based approximations studied in Chapter 4.

5.1 Objective function and variables to be optimized

To optimize the gyroscope for sensitivity, the output voltage V_{out} from Figure 3.2 needs to be maximized. The objective function to be minimized can therefore be taken as

$$f_0 = -V_{out}, \tag{5.1.1}$$

where V_{out} is given in (3.4.23).

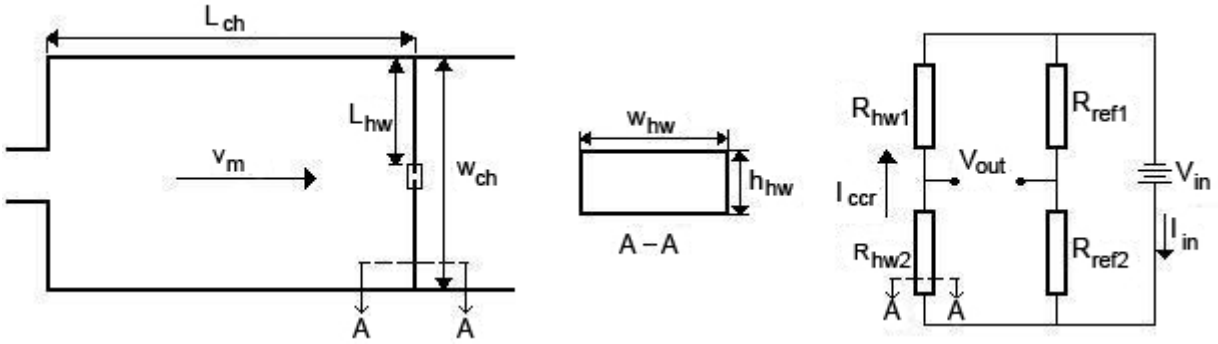


Figure 5.1: Detecting chamber and hot wire circuit, illustrating the design variables

In this problem, eight design variables were identified, $n = 8$, and need to be optimized. These are given in Table 5.1 and illustrated in Figure 5.1.

Table 5.1: Design variables needed for optimization

Variable	Description	Short
$x(1)$	Constant current across hot wires [μA]	I_{ccr}
$x(2)$	Length of the hot wire [mm]	L_{hw}
$x(3)$	Height of the hot wire [mm]	h_{hw}
$x(4)$	Width of the hot wire [mm]	w_{hw}
$x(5)$	Channel width [mm]	w_{ch}
$x(6)$	Mean velocity across hot wires [mm/s]	v_m
$x(7)$	Length from nozzle to hot wire [mm]	L_{ch}
$x(8)$	Channel height [mm]	h_{ch}

5.2 Bounds

Reasonable upper and lower bounds were selected based on the simulations done by Dau and co-workers [11].

The bounds and initial approximation values of the design variables are given in Table 5.2.

Table 5.2: Bounds and initial approximations of design variables

Variable	Lower bound	Upper bound	Initial approximation
$x(1)$	1	100	30
$x(2)$	100×10^{-3}	1000×10^{-3}	400×10^{-3}
$x(3)$	1×10^{-3}	100×10^{-3}	2.5×10^{-3}
$x(4)$	5×10^{-3}	100×10^{-3}	10×10^{-3}
$x(5)$	1	50	25
$x(6)$	0.1×10^{3}	10×10^{3}	3.5×10^{3}
$x(7)$	1	30	7.5
$x(8)$	1	50	25

5.3 Constraint functions

The in-equality constraint functions f_α, considered in this optimization problem, are briefly discussed in this section. Note that $f_\alpha \leq 0$, $\alpha = 1, 2, \ldots, m$ and $m = 10$.

The first three constraint functions, f_1, f_2 and f_3, are necessary for general fluid modelling (refer to Section 3.3). A compressible fluid can generally be treated as an incompressible fluid when $Ma < 0.3$

$$f_1 = Ma - 0.3, \tag{5.3.1}$$

$Kn < 0.01$ validates the continuum approach

$$f_2 = Kn - 10^{-3}, \tag{5.3.2}$$

and $Re_f < 2100$ describes laminar flow

$$f_3 = Re_f - 2100. \tag{5.3.3}$$

Constraint function f_4 is a precautionary measure to ensure a sufficient inertia to viscosity ratio,

$$f_4 = 100 - Re_f \tag{5.3.4}$$

and in this design problem, inertia will be a hundred times larger than viscosity.

Constraint functions f_5 and f_6 describe the structural limitations due to failure (refer to Section 3.6). The gyroscope is designed to withstand an impact acceleration of $15000g$

$$f_5 = 15000(9.81 \times 10^3) - a_{imp}, \tag{5.3.5}$$

and the stress due to the drag force should not exceed the maximum stress of the hot wire material

$$f_6 = Q_D - Q_{max}. \tag{5.3.6}$$

Power consumption of the hot wires is limited to 1 mW by means of constraint function f_7,

$$f_7 = P_{hw} - 10^6 \tag{5.3.7}$$

and the reader is referred to Section 3.4.2.3 for more detail.

Constraint function f_8 limits the hot wire resistances to positive values $R_{hw} > 0$

$$f_8 = \alpha Iccr^2 R_0 - hA_{hw}, \tag{5.3.8}$$

and f_9 describes the *dead zone* between the two hot wires

$$f_9 = 3L_{hw} - w_{ch}. \tag{5.3.9}$$

Constraint function f_{10} limits the deflection of the flow profile to center of the hot wire

$$f_{10} = \delta - 0.5(w_{ch} - L_{hw}). \tag{5.3.10}$$

It might become necessary to adjust this term later on when a better analytical relationship is found between the deflection of the flow profile and the velocity gradient across the hot wire. At this point in time we design for a maximum linear range and assume the deflection produced by half the maximum angular velocity should be in the center of the hot wire.

Scaling of the constraint functions is a challenging task, as we are dealing with a dual algorithm (refer to Section 4.6) and the Hessian is ill-conditioned. By means of trial and error, the normalization of constraint functions $f_\alpha, \alpha = 1,2,\ldots,7$, proved successful. After scaling of the constraint functions, (5.3.10) to (5.3.10) becomes

$$\begin{aligned}
f_1 &= \frac{Ma}{0.3} - 1 \\
f_2 &= \frac{Kn}{10^{-3}} - 1 \\
f_3 &= \frac{Re_f}{2100} - 1 \\
f_4 &= 1 - \frac{Re_f}{100} \\
f_5 &= 1 - \frac{a_{imp}}{15000(9.81 \times 10^3)} \\
f_6 &= \frac{Q_D}{Q_{max}} - 1 \\
f_7 &= \frac{P_{hw}}{10^6} - 1 \\
f_8 &= \alpha Iccr^2 R_0 - hA_{hw} \\
f_9 &= 3L_{hw} - w_{ch} \\
f_{10} &= \delta - 0.5(w_{ch} - L_{hw}),
\end{aligned} \tag{5.3.11}$$

and these are used in the FORTRAN implementations.

5.4 Gradients

In order to calculate the first order derivatives $\frac{df_\alpha}{dx_i}$, we first need to find all functions f_α in terms of the design variables x_i, where $\alpha = 0, 1, 2 \ldots, m$ and $i = 1, 2, \ldots, n$.

The objective function, in terms of the design variables, is given as

$$
\begin{aligned}
f &= -V_{out} \\
&= \frac{\alpha A_{hw} I^3 {R_0}^2 bc v^{c-1}}{\left[A_{hw} b v^c - R_0 I^2 \alpha\right]^2} \kappa\delta \\
&= \frac{\alpha 2 L_{hw} (w_{hw} + h_{hw}) I^3 (\lambda_r L_{hw} (1+5\alpha))^2 \left(1.1 kCLc^{n-1} Pr^{0.31}\right) n v^{n-1}}{(v^n) (h_{hw} w_{hw})^2 \left[2 L_{hw} (w_{hw} + h_{hw}) \left(\left(\frac{1.1 kCLc^{n-1} Pr^{0.31}}{v^n}\right) v^n\right) - \left(\frac{\lambda_r L_{hw} (1+5\alpha)}{h_{hw} w_{hw}}\right) I^2 \alpha\right]^2} \kappa\delta \\
&= \frac{2.2\alpha (w_{hw} + h_{hw}) I^3 {\lambda_r}^2 {L_{hw}}^3 (1+5\alpha)^2 kC {w_{hw}}^{n-1} Pr^{0.31} n v^{n-1}}{v^n \left(h_{hw}^2 w_{hw}\right)^2 \left[\left(\frac{(w_{hw}+h_{hw}) 2.2 L_{hw} kC {w_{hw}}^{n-1} Pr^{0.31} v^n}{v^n}\right) - \left(\frac{\lambda_r L_{hw} (1+5\alpha) I^2 \alpha}{h_{hw} w_{hw}}\right)\right]^2} \kappa\delta \\
&= \frac{2.2\alpha (w_{hw} + h_{hw}) I^3 {\lambda_r}^2 {L_{hw}}^3 (1+5\alpha)^2 kC {w_{hw}}^{n-1} Pr^{0.31} n v^{n-1}}{v^n (h_{hw} w_{hw})^2 \left[\frac{\left((w_{hw}+h_{hw}) 2.2 L_{hw} kC {w_{hw}}^{n-1} Pr^{0.31} v^n (h_{hw} w_{hw}) - \lambda_r L_{hw} (1+5\alpha) I^2 \alpha v^n\right)^2}{(h_{hw} w_{hw})^2 v^{2n}}\right]} \kappa\delta \\
&= \frac{2.2\alpha (w_{hw} + h_{hw}) I^3 {\lambda_r}^2 {L_{hw}}^3 (1+5\alpha)^2 kC {w_{hw}}^{n-1} Pr^{0.31} n v^{n-1} v^n}{\left[2.2(w_{hw} + h_{hw}) w_{hw} h_{hw} L_{hw} kC {w_{hw}}^{n-1} Pr^{0.31} v^n - \lambda_r L_{hw} (1+5\alpha) I^2 \alpha v^n\right]^2} \frac{-{L_{ch}}^2}{v} \kappa\omega \\
&= \frac{-2.2 {x_7}^2 \alpha (x_3 + x4) {x_1}^3 {\lambda_r}^2 x_2 (1+5\alpha)^2 kC {x_4}^{n-1} Pr^{0.31} n {x_6}^{n-2} v^n}{\left[2.2(x_3 + x_4) x_3 kC {x_4}^n Pr^{0.31} {x_6}^n - \lambda_r (1+5\alpha) {x_1}^2 \alpha v^n\right]^2} \kappa\omega. \qquad (5.4.1)
\end{aligned}
$$

The non-zero objective function derivatives are given as

$$
\begin{aligned}
\frac{df_0}{dx_1} &= \frac{-\kappa\omega 6.6 {x_7}^2 \alpha (x_3 + x4) {x_1}^2 {\lambda_r}^2 x_2 (1+5\alpha)^2 kC {x_4}^{n-1} Pr^{0.31} n {x_6}^{n-2} v^n}{\left[2.2(x_3 + x_4) x_3 kC {x_4}^n Pr^{0.31} {x_6}^n - \lambda_r (1+5\alpha) {x_1}^2 \alpha v^n\right]^2} - \\
&\quad \frac{8.8\kappa\omega\alpha^2 {x_7}^2 (x_3 + x4) {x_1}^4 {\lambda_r}^3 x_2 (1+5\alpha)^3 kC {x_4}^{n-1} Pr^{0.31} n {x_6}^{n-2} v^{2n}}{\left[2.2(x_3 + x_4) x_3 kC {x_4}^n Pr^{0.31} {x_6}^n - \lambda_r (1+5\alpha) {x_1}^2 \alpha v^n\right]^3} \\
\frac{df_0}{dx_2} &= \frac{-\kappa\omega \left(2.2 {x_7}^2 \alpha (x_3 + x4) {x_1}^3 {\lambda_r}^2 (1+5\alpha)^2 kC {x_4}^{n-1} Pr^{0.31} n {x_6}^{n-2} v^n\right)}{\left[2.2(x_3 + x_4) x_3 kC {x_4}^n Pr^{0.31} {x_6}^n - \lambda_r (1+5\alpha) {x_1}^2 \alpha v^n\right]^2} \\
\frac{df_0}{dx_3} &= \frac{-\kappa\omega \left(2.2 {x_7}^2 \alpha {x_1}^3 {\lambda_r}^2 x_2 (1+5\alpha)^2 kC {x_4}^{n-1} Pr^{0.31} n {x_6}^{n-2} v^n\right)}{\left[2.2(x_3 + x_4) x_3 kC {x_4}^n Pr^{0.31} {x_6}^n - \lambda_r (1+5\alpha) {x_1}^2 \alpha v^n\right]^2} + \\
&\quad \frac{\kappa\omega (9.68) (2x_3 + x_4) {x_7}^2 \alpha (x_3 + x4) {x_1}^3 {\lambda_r}^2 x_2 (1+5\alpha)^2 k^2 C^2 {x_4}^{2n-1} Pr^{0.62} n {x_6}^{2n-2} v^n}{\left[2.2(x_3 + x_4) x_3 kC {x_4}^n Pr^{0.31} {x_6}^n - \lambda_r (1+5\alpha) {x_1}^2 \alpha v^n\right]^3}
\end{aligned}
$$

$$\begin{aligned}
\frac{df_0}{dx_4} &= \frac{-\kappa\omega\left((x_3+x_4)(n-1)x_4{}^{n-2}+x_4{}^{n-1}\right)2.2x_7{}^2\alpha x_1{}^3\lambda_r{}^2x_2(1+5\alpha)^2kCPr^{0.31}nx_6{}^{n-2}\nu^n}{\left[2.2(x_3+x_4)x_3kCx_4{}^nPr^{0.31}x_6{}^n-\lambda_r(1+5\alpha)x_1{}^2\alpha\nu^n\right]^2}+ \\
&\quad \frac{\kappa\omega(9.68)\left((x_3+x_4)nx_4^{n-1}+x_4^n\right)x_3x_2k^2C^2Pr^{0.62}x_7{}^2\alpha(x_3+x4)x_1{}^3\lambda_r{}^2(1+5\alpha)^2x_4{}^{n-1}nx_6{}^{2n-2}\nu^n}{\left[2.2(x_3+x_4)x_3kCx_4{}^nPr^{0.31}x_6{}^n-\lambda_r(1+5\alpha)x_1{}^2\alpha\nu^n\right]^3} \\
\frac{df_0}{dx_6} &= \frac{-\kappa\omega(2.2)x_7{}^2\alpha(x_3+x4)x_1{}^3\lambda_r{}^2x_2(1+5\alpha)^2kCx_4{}^{n-1}Pr^{0.31}n(n-2)x_6{}^{n-3}\nu^n}{\left[2.2(x_3+x_4)x_3kCx_4{}^nPr^{0.31}x_6{}^n-\lambda_r(1+5\alpha)x_1{}^2\alpha\nu^n\right]^2}+ \\
&\quad \frac{\kappa\omega(9.68)x_3x_7{}^2\alpha(x_3+x4)^2x_1{}^3\lambda_r{}^2(1+5\alpha)^2k^2C^2x_4{}^{2n-1}Pr^{0.62}n^2x_6{}^{2n-3}\nu^n}{\left[2.2(x_3+x_4)x_3kCx_4{}^nPr^{0.31}x_6{}^n-\lambda_r(1+5\alpha)x_1{}^2\alpha\nu^n\right]^3} \\
\frac{df_0}{dx_7} &= \frac{-4.4x_7\alpha(x_3+x4)x_1{}^3\lambda_r{}^2x_2(1+5\alpha)^2kCx_4{}^{n-1}Pr^{0.31}nx_6{}^{n-2}\nu^n}{\left[2.2(x_3+x_4)x_3kCx_4{}^nPr^{0.31}x_6{}^n-\lambda_r(1+5\alpha)x_1{}^2\alpha\nu^n\right]^2}\kappa\omega. \qquad (5.4.2)
\end{aligned}$$

Constraint function f_1 and its non-zero derivatives:

$$\begin{aligned}
f_1 &= \frac{Ma}{0.3}-1 \\
&= \frac{v}{0.3a_o}-1 \\
&= \frac{x_6}{0.3a_o}-1 \qquad (5.4.3)
\end{aligned}$$

$$\frac{df_1}{dx_6} = \frac{1}{0.3a_o} \qquad (5.4.4)$$

Constraint function f_2 and its non-zero derivatives:

$$\begin{aligned}
f_2 &= \frac{Kn}{10^{-3}}-1 \\
&= \sqrt{\frac{\pi\gamma}{2}}\frac{Ma}{(10^{-3})Re}-1 \\
&= \sqrt{\frac{\pi\gamma}{2}}\frac{v\nu}{(10^{-3})a_o\nu D_{ch}}-1 \\
&= \sqrt{\frac{\pi\gamma}{2}}\frac{\nu(h_{ch}+w_{ch})}{(2\times10^{-3})a_ow_{ch}h_{ch}}-1 \\
&= \sqrt{\frac{\pi\gamma}{2}}\frac{\nu(x_5+x_8)}{(2\times10^{-3})a_ox_5x_8}-1 \qquad (5.4.5)
\end{aligned}$$

$$\begin{aligned}
\frac{df_2}{dx_5} &= \frac{\sqrt{\frac{\pi\gamma}{2}}2a_o\nu(x_5-x_8-x_5)}{(4\times10^{-3})a_o{}^2x_8{}^2x_5{}^2} \\
&= \frac{-\sqrt{\frac{\pi\gamma}{2}}\nu}{(2\times10^{-3})a_ox_5{}^2}
\end{aligned}$$

$$\begin{aligned}\frac{df_2}{dx_8} &= \frac{\sqrt{\frac{\pi\gamma}{2}}2a_o\nu(x_8-x_8-x_5)}{(4\times10^{-3})a_o^2{x_8}^2{x_5}^2}\\ &= \frac{-\sqrt{\frac{\pi\gamma}{2}}\nu}{(2\times10^{-3})a_o{x_8}^2}\end{aligned} \tag{5.4.6}$$

Constraint function f_3 and non-zero derivatives:

$$\begin{aligned}f_3 &= \frac{Re_f}{2100}-1\\ &= \frac{\upsilon D_{ch}}{2100\nu}-1\\ &= \frac{2\upsilon w_{ch}h_{ch}}{2100\nu(h_{ch}+w_{ch})}-1\\ &= \frac{2x_5x_6x_8}{2100\nu(x_5+x_8)}-1\end{aligned} \tag{5.4.7}$$

$$\begin{aligned}\frac{df_3}{dx_5} &= \frac{2\nu x_6x_8(x_8+x_5-x_5)}{2100\nu^2(x_5+x_8)^2}\\ &= \frac{2x_6{x_8}^2}{2100\nu(x_5+x_8)^2}\\ \frac{df_3}{dx_6} &= \frac{2x_5x_8}{2100\nu(x_5+x_8)}\\ \frac{df_3}{dx_8} &= \frac{2x_5x_6\nu(x_5+x_8-x_8)}{2100\nu^2(x_5+x_8)^2}\\ &= \frac{2x_5^2x_6}{2100\nu(x_5+x_8)^2}\end{aligned} \tag{5.4.8}$$

Constraint function f_4 and its non-zero derivatives:

$$\begin{aligned}f_4 &= 1-\frac{Re_f}{100}\\ &= 1-\frac{2\upsilon w_{ch}h_{ch}}{100\nu(h_{ch}+w_{ch})}\\ &= 1-\frac{2x_5x_6x_8}{100\nu(x_5+x_8)}\end{aligned} \tag{5.4.9}$$

$$\begin{aligned}\frac{df_4}{dx_5} &= -\frac{2x_6{x_8}^2}{100\nu(x_5+x_8)^2}\\ \frac{df_4}{dx_6} &= -\frac{2x_5x_8}{100\nu(x_5+x_8)}\\ \frac{df_4}{dx_8} &= -\frac{2x_5^2x_6}{100\nu(x_5+x_8)^2}\end{aligned} \tag{5.4.10}$$

Constraint function f_5 and its non-zero derivatives:

$$\begin{aligned} f_5 &= 1 - \frac{a_{imp}}{15000g} \\ &= 1 - \frac{Q_{max} h_{hw} w_{hw}}{15000 g \rho_{hw} L_{hw} w_{hw} h_{hw}} \\ &= 1 - \frac{Q_{max}}{15000 g \rho_{hw} L_{hw}} \\ &= 1 - \frac{Q_{max}}{15000 g \rho_{hw} x_2} \end{aligned} \tag{5.4.11}$$

$$\frac{df_5}{dx_2} = \frac{Q_{max}}{15000 g \rho_{hw} {x_2}^2} \tag{5.4.12}$$

Constraint function f_6 and its non-zero derivatives:

$$\begin{aligned} f_6 &= \frac{Q_D}{Q_{max}} - 1 \\ &= \frac{0.5 \rho_{hw} v^2 C_D}{Q_{max}} - 1 \\ &= \frac{0.5 \rho_{hw} x_6^2 C_D}{Q_{max}} - 1 \end{aligned} \tag{5.4.13}$$

$$\frac{df_6}{dx_6} = \frac{\rho_{hw} x_6 C_D}{Q_{max}} \tag{5.4.14}$$

Constraint function f_7 and its non-zero derivatives:

$$\begin{aligned} f_7 &= \frac{P_{hw}}{10^6} - 1 \\ &= \frac{V_{in} I_{in}}{10^6} - 1 \\ &= \frac{4 I^2 {R_{hw}}^2}{\left(2 \times 10^6\right) R_0} + \frac{2 I^2 R_{hw}}{10^6} - 1 \\ &= \frac{4 I^2 h^2 {A_{hw}}^2 {R_0}^2}{\left(2 \times 10^6\right) R_0 \left(h A_{hw} - \alpha I^2 R_0\right)^2} + \left(\frac{\left(2 \times 10^{-6}\right) I^2 h A_{hw} R_0}{h A_{hw} - \alpha I^2 R_0}\right) - 1 \\ &= \frac{\left(4 \times 10^{-6}\right) I^2 \left(\frac{v^n 1.1 k C {w_{hw}}^{n-1} Pr^{0.31}}{\nu^n}\right)^2 (2 L_{hw} (h_{hw} + w_{hw}))^2 \left(\frac{\lambda_r L_{hw} (1+5\alpha)}{h_{hw} w_{hw}}\right)^2}{2 \left(\frac{\lambda_r L_{hw} (1+5\alpha)}{h_{hw} w_{hw}}\right) \left(\left(\frac{v^n 1.1 k C {w_{hw}}^{n-1} Pr^{0.31}}{\nu^n}\right) (2 L_{hw} (h_{hw} + w_{hw})) - \alpha I^2 \left(\frac{\lambda_r L_{hw} (1+5\alpha)}{h_{hw} w_{hw}}\right)\right)^2} \\ &\quad + \frac{\left(2 \times 10^{-6}\right) I^2 \left(\frac{v^n 1.1 k C {w_{hw}}^{n-1} Pr^{0.31}}{\nu^n}\right) (2 L_{hw} (h_{hw} + w_{hw})) \left(\frac{\lambda_r L_{hw} (1+5\alpha)}{h_{hw} w_{hw}}\right)}{\left(\frac{v^n 1.1 k C {w_{hw}}^{n-1} Pr^{0.31}}{\nu^n}\right) (2 L_{hw} (h_{hw} + w_{hw})) - \alpha I^2 \left(\frac{\lambda_r L_{hw} (1+5\alpha)}{h_{hw} w_{hw}}\right)} - 1 \end{aligned}$$

$$
\begin{aligned}
&= \frac{\left(2\times10^{-6}\right) I^2 v^{2n} 1.1^2 k^2 C^2 {w_{hw}}^{2n-2} Pr^{2(0.31)} 4(h_{hw}+w_{hw})^2 \lambda_r L_{hw}(1+5\alpha) h_{hw} w_{hw}}{\left(v^n 1.1 kC {w_{hw}}^{n-1} Pr^{0.31} 2(h_{hw}+w_{hw}) h_{hw} w_{hw} - v^n \alpha I^2 \lambda_r (1+5\alpha)\right)^2} \\
&\quad + \frac{\left(2\times10^{-6}\right) I^2 v^n 1.1 kC {w_{hw}}^{n-1} Pr^{0.31} (h_{hw}+w_{hw}) \lambda_r L_{hw}(1+5\alpha)}{h_{hw} w_{hw} v^n 2.2 kC {w_{hw}}^{n-1} Pr^{0.31} (h_{hw}+w_{hw}) - v^n \alpha I^2 \lambda_r (1+5\alpha)} - 1 \\
&= \frac{\left(2\times10^{-6}\right) {x_1}^2 {x_6}^{2n} 1.1^2 k^2 C^2 {x_4}^{2n-2} Pr^{2(0.31)} 4(x_3+x_4)^2 \lambda_r x_2 (1+5\alpha) x_3 x_4}{\left({x_6}^n 1.1 kC {x_4}^{n-1} Pr^{0.31} 2(x_3+x_4) x_3 x_4 - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)\right)^2} \\
&\quad + \frac{\left(1.1\times10^{-6}\right) {x_1}^2 {x_6}^n kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) \lambda_r x_2 (1+5\alpha)}{x_3 x_4 {x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)} - 1 \\
&= \frac{\left(9.68\times10^{-6}\right) {x_1}^2 {x_6}^{2n} k^2 C^2 {x_4}^{2n-1} Pr^{2(0.31)} (x_3+x_4)^2 \lambda_r x_2 (1+5\alpha) x_3}{\left({x_6}^n 2.2 kC {x_4}^n Pr^{0.31} (x_3+x_4) x_3 - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)\right)^2} \\
&\quad + \frac{\left(2.2\times10^{-6}\right) {x_1}^2 {x_6}^n kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) \lambda_r x_2 (1+5\alpha)}{x_3 {x_6}^n 2.2 kC {x_4}^n Pr^{0.31} (x_3+x_4) - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)} - 1
\end{aligned} \tag{5.4.15}
$$

$$
\begin{aligned}
\frac{df_7}{dx_1} &= \frac{\left(19.36\times10^{-6}\right) x_1 {x_6}^{2n} k^2 C^2 {x_4}^{2n-1} Pr^{0.62} (x_3+x_4)^2 \lambda_r x_2 (1+5\alpha) x_3}{\left({x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) x_3 x_4 - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)\right)^2} + \\
&\quad \frac{\left(38.72\times10^{-6}\right) v^n \alpha {\lambda_r}^2 (1+5\alpha)^2 {x_1}^3 {x_6}^{2n} k^2 C^2 {x_4}^{2n-1} Pr^{0.62} (x_3+x_4)^2 x_2 x_3}{\left({x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) x_3 x_4 - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)\right)^3} + \\
&\quad \frac{\left(4.4\times10^{-6}\right) x_1 {x_6}^n kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) \lambda_r x_2 (1+5\alpha)}{x_3 x_4 {x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)} + \\
&\quad \frac{\left(4.4\times10^{-6}\right) v^n \alpha {x_1}^3 {x_6}^n kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) {\lambda_r}^2 x_2 (1+5\alpha)^2}{\left(x_3 x_4 {x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)\right)^2}
\end{aligned}
$$

$$
\begin{aligned}
\frac{df_7}{dx_2} &= \frac{\left(9.68\times10^{-6}\right) {x_1}^2 {x_6}^{2n} k^2 C^2 {x_4}^{2n-1} Pr^{0.62} (x_3+x_4)^2 \lambda_r (1+5\alpha) x_3}{\left({x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) x_3 x_4 - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)\right)^2} + \\
&\quad \frac{\left(2.2\times10^{-6}\right) {x_1}^2 {x_6}^n kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) \lambda_r (1+5\alpha)}{x_3 x_4 {x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)}
\end{aligned}
$$

$$
\begin{aligned}
\frac{df_7}{dx_3} &= \frac{\left(9.68\times10^{-6}\right) {x_1}^2 {x_6}^{2n} k^2 C^2 {x_4}^{2n-1} Pr^{0.62} \left[(x_3+x_4)^2 + 2x_3(x_3+x_4)\right] \lambda_r x_2 (1+5\alpha)}{\left({x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) x_3 x_4 - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)\right)^2} - \\
&\quad \frac{\left(42.592\times10^{-6}\right) (2x_3+x_4) {x_1}^2 {x_6}^{3n} k^3 C^3 {x_4}^{3n-1} Pr^{0.93} (x_3+x_4)^2 \lambda_r x_2 (1+5\alpha) x_3}{\left({x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) x_3 x_4 - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)\right)^3} + \\
&\quad \frac{\left(2.2\times10^{-6}\right) {x_1}^2 {x_6}^n kC {x_4}^{n-1} Pr^{0.31} \lambda_r x_2 (1+5\alpha)}{x_3 x_4 {x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)} - \\
&\quad \frac{\left(4.84\times10^{-6}\right) {x_4}^{2n-1} (2x_3+x_4) {x_1}^2 {x_6}^{2n} k^2 C^2 Pr^{0.62} (x_3+x_4) \lambda_r x_2 (1+5\alpha)}{\left(x_3 x_4 {x_6}^n 2.2 kC {x_4}^{n-1} Pr^{0.31} (x_3+x_4) - v^n \alpha {x_1}^2 \lambda_r (1+5\alpha)\right)^2}
\end{aligned}
$$

$$\begin{aligned}
\frac{df_7}{dx_4} &= \frac{\left(9.68\times10^{-6}\right){x_1}^2{x_6}^{2n}k^2C^2Pr^{0.62}\left((2n-1)\left(x_3+x_4\right)^2x_4^{2n-2}+2\left(x_3+x_4\right)x_4^{2n-1}\right)\lambda_r x_2\left(1+5\alpha\right)x_3}{\left({x_6}^n 2.2kC{x_4}^{n-1}Pr^{0.31}\left(x_3+x_4\right)x_3x_4-\nu^n\alpha{x_1}^2\lambda_r\left(1+5\alpha\right)\right)^2} - \\
&\quad \frac{\left(42.592\times10^{-6}\right)\left(nx_4^{n-1}\left(x_3+x_4\right)+x_4^n\right)x_3^2{x_1}^2{x_6}^{3n}k^3C^3{x_4}^{2n-1}Pr^{0.93}\left(x_3+x_4\right)^2\lambda_r x_2\left(1+5\alpha\right)}{\left({x_6}^n 2.2kC{x_4}^{n-1}Pr^{0.31}\left(x_3+x_4\right)x_3x_4-\nu^n\alpha{x_1}^2\lambda_r\left(1+5\alpha\right)\right)^3} + \\
&\quad \frac{\left(2.2\times10^{-6}\right){x_1}^2{x_6}^n kCPr^{0.31}\left(x_4^{n-1}+\left(n-1\right)x_4^{n-2}\left(x_3+x_4\right)\right)\lambda_r x_2\left(1+5\alpha\right)}{x_3x_4{x_6}^n 2.2kC{x_4}^{n-1}Pr^{0.31}\left(x_3+x_4\right)-\nu^n\alpha{x_1}^2\lambda_r\left(1+5\alpha\right)} - \\
&\quad \frac{\left(4.84\times10^{-6}\right)x_3\left(x_4^n+nx_4^{n-1}\left(x_3+x_4\right)\right){x_1}^2{x_6}^{2n}k^2C^2{x_4}^{n-1}Pr^{0.62}\left(x_3+x_4\right)\lambda_r x_2\left(1+5\alpha\right)}{\left(x_3x_4{x_6}^n 2.2kC{x_4}^{n-1}Pr^{0.31}\left(x_3+x_4\right)-\nu^n\alpha{x_1}^2\lambda_r\left(1+5\alpha\right)\right)^2} \\
\frac{df_7}{dx_6} &= \frac{\left(19.36\times10^{-6}\right){x_1}^2 n{x_6}^{2n-1}k^2C^2{x_4}^{2n-1}Pr^{0.62}\left(x_3+x_4\right)^2\lambda_r x_2\left(1+5\alpha\right)x_3}{\left({x_6}^n 2.2kC{x_4}^{n-1}Pr^{0.31}\left(x_3+x_4\right)x_3x_4-\nu^n\alpha{x_1}^2\lambda_r\left(1+5\alpha\right)\right)^2} - \\
&\quad \frac{\left(42.592\times10^{-6}\right){x_6}^{3n-1}n\left(x_3+x_4\right){x_1}^2k^3C^3{x_3}^2{x_4}^{3n-1}Pr^{0.93}\left(x_3+x_4\right)^2\lambda_r x_2\left(1+5\alpha\right)}{\left({x_6}^n 2.2kC{x_4}^{n-1}Pr^{0.31}\left(x_3+x_4\right)x_3x_4-\nu^n\alpha{x_1}^2\lambda_r\left(1+5\alpha\right)\right)^3} + \\
&\quad \frac{\left(2.2\times10^{-6}\right){x_1}^2 n{x_6}^{n-1}kC{x_4}^{n-1}Pr^{0.31}\left(x_3+x_4\right)\lambda_r x_2\left(1+5\alpha\right)}{x_3x_4{x_6}^n 2.2kC{x_4}^{n-1}Pr^{0.31}\left(x_3+x_4\right)-\nu^n\alpha{x_1}^2\lambda_r\left(1+5\alpha\right)} - \\
&\quad \frac{\left(4.84\times10^{-6}\right)x_3 n{x_6}^{2n-1}{x_1}^2k^2C^2{x_4}^{2n-1}Pr^{0.62}\left(x_3+x_4\right)^2\lambda_r x_2\left(1+5\alpha\right)}{\left(x_3x_4{x_6}^n 2.2kC{x_4}^{n-1}Pr^{0.31}\left(x_3+x_4\right)-\nu^n\alpha{x_1}^2\lambda_r\left(1+5\alpha\right)\right)^2}
\end{aligned} \tag{5.4.16}$$

Constraint function f_8 and its non-zero derivatives:

$$\begin{aligned}
f_8 &= \alpha I^2R_o - hA_{hw} \\
&= \frac{\alpha I^2\lambda_r L_{hw}\left(1+5\alpha\right)}{h_{hw}w_{hw}} - \left(\frac{1.1kC{w_{hw}}^{(n-1)}Pr^{0.31}\nu^n}{\nu^n}\right)\left(2L_{hw}\left(h_{hw}+w_{hw}\right)\right) \\
&= \frac{\alpha I^2\lambda_r L_{hw}\left(1+5\alpha\right)}{h_{hw}w_{hw}} - \frac{2.2kC{w_{hw}}^{(n-1)}Pr^{0.31}\nu^nL_{hw}h_{hw}}{\nu^n} - \frac{2.2kC{w_{hw}}^nPr^{0.31}\nu^nL_{hw}}{\nu^n} \\
&= \frac{\alpha{x_1}^2\lambda_r x_2\left(1+5\alpha\right)}{x_3x_4} - \frac{2.2kC{x_4}^{(n-1)}Pr^{0.31}{x_6}^nx_2x_3}{\nu^n} - \frac{2.2kC{x_4}^nPr^{0.31}{x_6}^nx_2}{\nu^n}
\end{aligned} \tag{5.4.17}$$

$$\begin{aligned}
\frac{df_8}{dx_1} &= \frac{2\alpha x_1\lambda_r x_2\left(1+5\alpha\right)}{x_3x_4} \\
\frac{df_8}{dx_2} &= \frac{\alpha{x_1}^2\lambda_r\left(1+5\alpha\right)}{x_3x_4} - \frac{2.2kC{x_4}^{(n-1)}Pr^{0.31}{x_6}^nx_3}{\nu^n} - \frac{2.2kC{x_4}^nPr^{0.31}{x_6}^n}{\nu^n} \\
\frac{df_8}{dx_3} &= \frac{-\alpha{x_1}^2\lambda_r x_2\left(1+5\alpha\right)}{{x_3}^2x_4} - \frac{2.2kC{x_4}^{(n-1)}Pr^{0.31}{x_6}^nx_2}{\nu^n} \\
\frac{df_8}{dx_4} &= \frac{-\alpha{x_1}^2\lambda_r x_2\left(1+5\alpha\right)}{x_3{x_4}^2} - \frac{2.2\left(n-1\right)kC{x_4}^{(n-2)}Pr^{0.31}{x_6}^nx_2x_3}{\nu^n} - \frac{2.2nkC{x_4}^{n-1}Pr^{0.31}{x_6}^nx_2}{\nu^n} \\
\frac{df_8}{dx_6} &= -\frac{2.2kC{x_4}^{(n-1)}Pr^{0.31}n{x_6}^{(n-1)}x_2x_3}{\nu^n} - \frac{2.2kC{x_4}^nPr^{0.31}n{x_6}^{(n-1)}x_2}{\nu^n}
\end{aligned} \tag{5.4.18}$$

Constraint function f_9 and its non-zero derivatives:

$$\begin{aligned} f_9 &= 3L_{hw} - w_{ch} \\ &= 3x_2 - x_5 \end{aligned} \tag{5.4.19}$$

$$\begin{aligned} \frac{df_9}{dx_2} &= 3 \\ \frac{df_9}{dx_5} &= -1 \end{aligned} \tag{5.4.20}$$

Constraint function f_{10} and its non-zero derivatives:

$$\begin{aligned} f_{10} &= \delta - 0.5(w_{ch} - L_{hw}) \\ &= \frac{\omega L_{ch}^2}{\nu} - 0.5(w_{ch} - L_{hw}) \\ &= \frac{\omega x_7^2}{x_6} - 0.5(x_5 - x_2) \end{aligned} \tag{5.4.21}$$

$$\begin{aligned} \frac{df_{10}}{dx_2} &= 0.5 \\ \frac{df_{10}}{dx_5} &= -0.5 \\ \frac{df_{10}}{dx_6} &= -\frac{\omega x_7^2}{{x_6}^2} \\ \frac{df_{10}}{dx_7} &= \frac{2\omega x_7}{x_6} \end{aligned} \tag{5.4.22}$$

The gradients calculated analytically in this section, were compared to gradients calculated with a finite difference method and the maximum error did not exceed 10^{-4}.

5.5 Fluid and material properties

Due to a lack of resources, manufacturing and testing of the gyroscope was not possible and it is important to have some means by which to verify the results. A similar gas and hot wire material, as used in the gyroscope of Dau and co-workers [11], was chosen in order to have some comparative aspects.

Inert Neon gas properties at 1 atmospheric pressure and $T = 25°C$ are given in Table 5.3 below:

Hot wire material and amplifier circuit properties are given in Table 5.4 below:

Table 5.3: Inert Neon gas properties

Property	Description	Value
μ	Dynamic Viscosity [kg/mm.s]	3142×10^{-11}
c_p	Specific Heat (constant pressure) [nJ/kg.C]	1030×10^{9}
c_v	Specific Heat (constant viscosity) [nJ/kg.C]	618×10^{9}
k_f	Thermal conductivity of the fluid [nW/mm.K]	49.1×10^{3}
ρ	Density [kg/mm^3]	0.846×10^{-9}
a_0	Velocity of sound in fluid [mm/s]	435×10^{3}

Table 5.4: Hot wire material and circuit properties

Variable	Description	Value
λ	Resistivity [kΩ.mm]	0.35×10^{-2}
α	Thermal coefficient of resistance [1/°C]	6000×10^{-6}
ρ	Hot wire density [kg/mm^3]	2300×10^{-9}
Q_{max}	Maximum stress for Silicon [kPa]	63.9×10^{6}
C_D	Drag force coefficient	0.9
C	Empirical constant	0.56
n	Empirical constant	0.8

5.6 Input

As previously explained (refer to Section 5.3), the angular velocity is proportional to the output voltage from the Wheatstone half-bridge V_{out} and its only influence on the optimization problem is the linear range that is designed for. Assuming constraint function 10 is a reasonable one, the angular velocity input ω should be equal to half the linear range $\omega_{max}/2$ that is designed for.

In this problem a linear range of 300 deg/s is assumed and the angular velocity input ω will be taken as $\omega = 150$ deg/s.

The proportionality constant κ (refer to Section 3.4.2.1) has no influence on the optimal solution and is assumed to be 1. This constant can only be obtained numerically and influences the objective function value proportionally.

5.7 Results

FORTRAN implementations of the ISE based approximations by Groenwold [16] and the MMA approximation from Svanberg [31] were implemented and compared (see Appendix B for setup). Function values and computational effort, using both BFGS and CG solvers, are compared for the different approximations discussed in Section 4.3. The goal is to find the most suitable solver-approximation combination for this specific gyroscope problem.

Results using a MATLAB implementation, discussed in Chapter 4.8, are compared to the FORTRAN results in order to establish the accuracy of our implementation.

5.7.1 Optimal function values for different approximations and solver algorithms

Results are presented in Table 5.5 and the optimal solution is found to be $f^{\{*\}} = -0.31809192 \times 10^4$. "—" implies *not converged within* 5000 *iterations*.

Table 5.5: Results using several different approximations and solver algorithms

Approximation	Solver	Optimal solution, $f^{\{*\}}$	Maximum constraint	Outer Iter	Inner Iter	KKT residual
SQ 1	BFGS	—	0.1590×10^{-1}	5000	—	—
SQ 1	CG	-0.31809192×10^4	-0.708×10^{-15}	46	209	1.46×10^{-8}
SQ 2	BFGS	—	0.1606×10^{-1}	5000	—	—
SQ 2	CG	-0.31809192×10^4	0.3456×10^{-11}	43	189	3.99×10^{-7}
NSQ	BFGS	—	-0.641×10^{-5}	5000	—	—
NSQ	CG	-0.31809192×10^4	0.9431×10^{-10}	45	172	2.26×10^{-4}
T2:reciprocal	BFGS	-0.31809192×10^4	-0.624×10^{-8}	50	18	3.23×10^{-5}
T2:reciprocal	CG	-0.31809192×10^4	0.1309×10^{-11}	52	7	4.50×10^{-9}
T2:exponential	BFGS	-0.31809192×10^4	0.2572×10^{-7}	68	20	1.92×10^{-4}
T2:exponential	CG	-0.31809192×10^4	-0.164×10^{-12}	71	7	5.17×10^{-5}
T2:MMA	BFGS	-0.31809192×10^4	0.3529×10^{-9}	31	343	2.42×10^{-5}
T2:MMA	CG	-0.31809192×10^4	0.3823×10^{-13}	21	77	4.08×10^{-4}
MMA	Combination	-0.31809192×10^4	0.2840×10^{-11}	17	68	4.15×10^{-6}

The BFGS solver is clearly not suitable for this optimization problem as convergence cannot be accomplished for three of the six quadratic approximations (refer to Figure 5.2 for the convergence curves). This solver algorithm terminated on the maximum number of steps for quadratic approximations 1, 2 and 3, which is an obvious implication that the optimal solution $f^{\{*\}}$ has not been found and those results should be ignored. The algorithm terminated on x-norm for the rest of the approximations, which implies convergence to the optimal solution, $f^{\{*\}} = -0.31809192 \times 10^4$.

The CG solver is quite suitable for this problem as the function value converges to an optimal solution within a reasonable number of iterations for all approximations (refer to Figure 5.2 for the convergence curves). This solver algorithm was terminated on x-norm for all approximations, which implies that $\|\boldsymbol{x}^{\{k+1\}} - \boldsymbol{x}^{\{k\}}\| \leq \varepsilon_x$.

The MATLAB implementation, using a BFGS solver, compares well to the FORTRAN implementation and Table 5.6 compares the function values and iteration history.

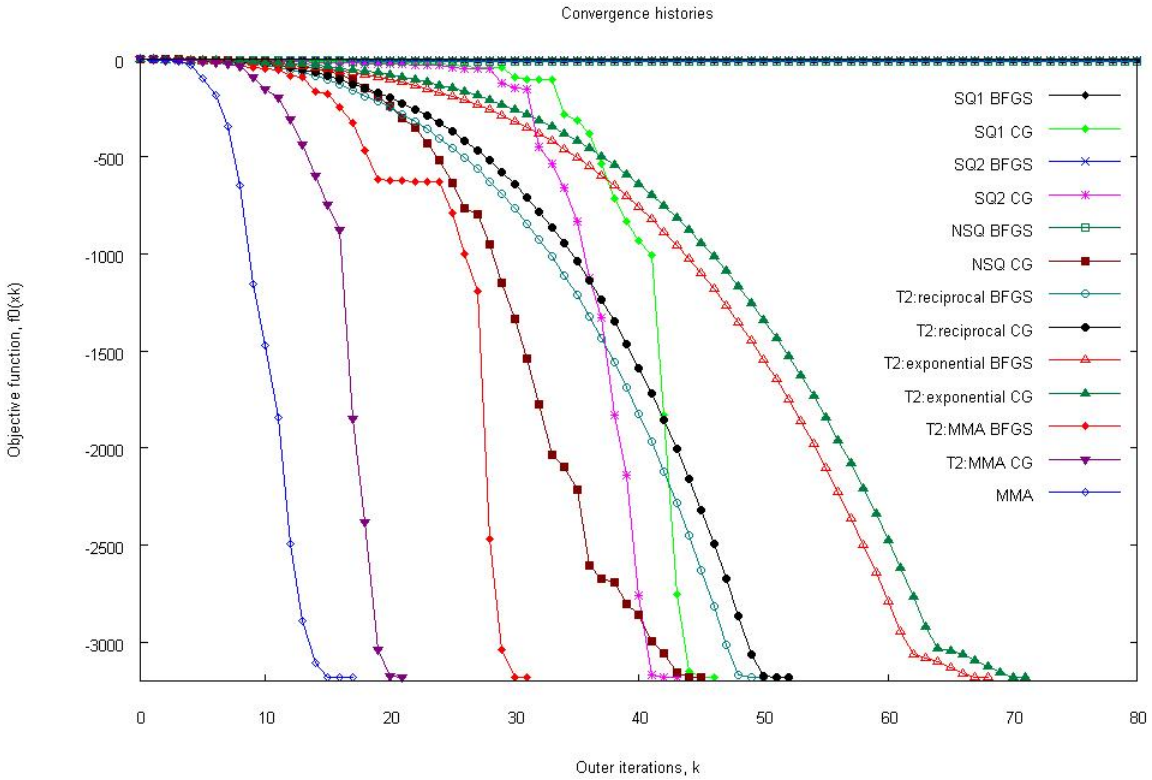


Figure 5.2: Convergence histories for the gyroscope optimization problem using several different approximations

Table 5.6: Comparison of FORTRAN and MATLAB implementations applied to the gyroscope problem

		FORTRAN		MATLAB	
Approximation	$f^{\{*\}}$	Outer Iterations	Inner Iterations	Outer Iterations	Inner Iterations
T2:reciprocal	-0.31809192×10^4	50	18	51	6
T2:exponential	-0.31809192×10^4	68	20	70	8
T2:MMA	-0.31809192×10^4	31	343	383	2661

5.7.2 Design variables and constraint function

All approximation and solver combinations that converged to an optimal solution, $f^{\{*\}} = -0.31809192 \times 10^4$, yield the same optimal design variables. The optimal design variable set $\mathbf{x}^*$ is presented in Table 5.7.

Four of the design variables reached their lower bounds and one reached its upper bound. Adjusting some of these bounds might result in greater sensitivity and it is up to the designer to select reasonable bounds for each gyroscope magnitude. These bounds were selected to compare the results to existing designs.

The influence of the constraint functions are presented in Table 5.8. A non-zero dual variable represents

Table 5.7: Optimal design variables

Variable	Lower bound	Upper bound	Initial approximation	Optimal value
$x(1)$	1	100	30	18.7484
$x(2)$	100×10^{-3}	1000×10^{-3}	400×10^{-3}	100×10^{-3}
$x(3)$	1×10^{-3}	100×10^{-3}	2.5×10^{-3}	1×10^{-3}
$x(4)$	5×10^{-3}	100×10^{-3}	10×10^{-3}	5×10^{-3}
$x(5)$	1	50	25	50
$x(6)$	0.1×10^{3}	10×10^{3}	3.5×10^{3}	0.1×10^{3}
$x(7)$	1	30	7.5	4.07840
$x(8)$	1	50	25	29.5416

Table 5.8: Influence of constraint functions

Constraint	Function value	Dual variable
f_1	-0.999	0
f_2	-0.996	0
f_3	-0.952	0
f_4	-0.819×10^{-5}	0
f_5	-0.186×10^{7}	0
f_6	-0.100×10^{1}	0
f_7	-0.283×10^{-10}	0.3731×10^{4}
f_8	-0.512×10^{2}	0
f_9	-0.497×10^{2}	0
f_{10}	-0.164×10^{-12}	0.1275×10^{3}

an active constraint.

It is clear that two out of the ten constraints are active and the sensitivity of the gyroscope will most likely be influenced by these constraints. As expected, a significant choice needs to be made between the power consumption and the sensitivity of the gyroscope.

5.7.3 Discussion of findings

In reviewing results from Table 5.5 to find the most suitable approximation for this specific optimization problem, we need to consider the deciding factors. Inner iterations are almost as expensive as the outer iterations, as the optimal dual variables need to be solved every time curvature components are adjusted. This implies that the sum of all iterations, inner and outer, would most accurately represent the cost of simulation. Interpreting results using this logic, imply that Groenwold's Taylor series expansion to the reciprocal and exponential approximations [15] proved most efficient and inexpensive.

This logic did not prove as simple when comparing the two solver algorithms. The sum total of iterations for the CG solver were lower with all approximations, but simulation time remained less with the BFGS solver. BFGS solvers use the Hessian matrix, which contains second order derivatives, to find the next

optimal point. This algorithm will prove faster than a first order linear algorithm, such as the CG solver, but requires the subproblem to be continuously differentiable twice. This condition is not true for dual subproblems and therefore the algorithm will not always converge. This was the case for three of the six approximations.

A BFGS solver is preferred for this optimization problem, but convergence cannot always be guaranteed. The CG solver would be a safer choice, although generally more expensive per iteration. It seems the key to low simulation time, is finding the right solver-approximation combination and neither one can be favored over the other.

The MATLAB implementation compared well to the FORTRAN implementation, when considering the number of iterations. Unfortunately, simulation time is inefficiently high when using a MATLAB compiler and should be considered as a last resort.

The resulting objective function value, $f^{\{*\}} = -0.31809192 \times 10^4$, is not the true voltage output from the Wheatstone bridge. The proportionality constant κ (refer to Section 5.6) is assumed to be 1, and it has no influence on the optimal solution. To determine the real voltage output, the proportionality constant should be found by means of numerical simulation.

In reviewing the optimal design variables in Table 5.7, it is clear that the bounds of the channel height and width should be limited relative to other dimensions. More accurate constraints would be possible when incorporating the effect of chamber dimensions on cross-sensitivity, flow profile and pressure differences into the analytical model. Limiting the chamber dimensions by reducing the values of the upper bounds, would not necessarily solve the problem. Optimal values would continue to reach the upper bounds and this only shows that the analytical model needs to be defined more accurately.

The length from the nozzle exit to the hot-wires was found to be $4.1mm$. This value would also be influenced when defining the analytical model for cross-sensitivity, length of fully developed flow and limiting the height and width of the channel.

The design variables might be optimal for the objective and constraint functions in this model, but it is clear that numerical simulation is needed to find the maximum velocity gradient across the hot wires. The width and height of the detecting chamber will have an influence on the velocity gradient across the hot wires and this has not been taken into account.

Chapter 6

Conclusion

6.1 Findings and accomplishments

This thesis presented the reader with an analytical model for a thermal convective microfluidic gyroscope that can be used to formulate an optimization problem and introduced some sequential approximate optimization essentials to apply to the gyroscope.

For the optimization of the detecting chamber of the thermal convective microfluidic gyroscope, several approximations to the objective and constraint functions were compared and two different subproblem solvers were investigated. The CG solver showed an overall better performance and convergence to an optimal solution was always possible. The single best performance, however, was found using the BFGS solver and quadratic Taylor series expansion to the reciprocal and exponential approximations. This showed that the CG solver is a safe choice, but the solver-approximation combination is the key to low simulation time.

The SAO algorithm, using the conservatism principle and a BFGS solver, was implemented in MATLAB and compared to existing implementations. Results compared well for less complex problems, whilst more significant iteration inaccuracies occurred for complex optimization problems. These inaccuracies can be explained by compiler and solver-implementation differences. Using the MATLAB compiler to solve for optimization problems proved time consuming and inefficient. Although a MATLAB implementation of SAO might be sought after, it is not recommended for use with complex problems whenever C-compiler options are available.

6.2 Difficulties

Numerical simulation is a priority in the design process of a thermal convective microfluidic gyroscope. It is unfortunate that very few software packages available are able to simulate the effect of the Coriolis

force due to an angular rotation on a moving fluid. Due to limited resources and access to sufficient software packages, numerical simulation was not a possibility in this thesis.

The impact on our results are obvious. The proportionality constant κ (refer to Section 5.6) was assumed to be 1, and was assumed to have no influence on the optimal solution. This constant can only be determined numerically and it does influence the voltage output from the Wheatstone bridge. Without a true output value, the performance of this gyroscope could not be determined and compared to existing gyroscope characteristics.

Scaling of the constraint functions and design variables were necessary, due to the horrendous curvatures produced for this problem and large order of magnitude differences in the design variables.

6.3 Suggestions and future development

Since analytical models for a thermal convective microfluidic gyroscope were not abundantly found in the literature, our optimization model still needs to be developed further into a usable, standardized model.

A better analytical relationship between the velocity gradient, the chamber width and height, the hot wire positions and the deflection of the flow profile, will be a major contribution to the analytical optimization of this detecting chamber. The only parameter currently limiting the length from the nozzle exit to the hot wires, is constraint function 10 (refer to Section 5.7.2, Table 5.8), which is only an over simplified design assumption and should be defined better. Other factors that would better define the constraints on the distance between the nozzle exit and the hot-wires is the length it would take to reach fully developed flow and cross-sensitivity would increase with this distance.

The fixed structure between the hot wires is currently assumed to have no effect on the fluid flow and no parameter limits its size. The area around the hot wires should be defined more accurately.

Further development to optimize the entire gyroscope, and not just the detecting chamber, will contribute to the standardization of this design process. This gyroscope concept can easily be converted to a dual-axis gyroscope and once a standardized single-axis model is in place, it can be expanded into a dual-axis model.

The hot wire material and fluid properties can be adjusted in order to experiment with several different materials and fluids to find the most suitable combination. The upper and lower bounds can also be experimented with, in order to compare several different order of magnitude gyroscopes with each other.

Numerical simulation and optimization should be integrated with our analytical optimization model, as it is an integral part of the design process.

List of References

[1] S.E. Alper and T. Akin. A symmetric surface micromachined gyroscope with decoupled oscillation modes. *Sensors and Actuators*, 97:347–358, 2002.

[2] M.N. Armenise, C. Ciminelli, F. De Leonardis, R. Diana, V. Passaro, and F. Peluso. Gyroscope technologies for space applications. In *Micro/Nano technologies for space*, 2003.

[3] F. Ayazi and K. Najafi. A HARPSS polysilicon vibrating ring gyroscope. *Journal of Microelectromechanical Systems*, 10(2):169–179, 2001.

[4] S. Beeby, G. Ensell, M. Kraft, and N. White. *MEMS Mechanical Sensors*. Artech House, Inc., Norwood, Massachusetts, 2004.

[5] R.L. Boylestad. *Introductory Circuit Analysis*. Prentice Hall, New Jersey, 2003.

[6] Y. A. Cengel. *Heat Transfer: A Practical Approach*. McGraw-Hill, New York, 2003.

[7] J. Chen and C. Liu. Development and characterization of surface micromachined, out-of-plane hot-wire anemometer. *Journal of Microelectromechanical systems*, 12:979–988, 2003.

[8] F. Chollet and H.B. Liu. A (not so) short introduction to micro electromechanical systems. Technical report, MicroMachines Centre, School of MAE, Nanyang Technological University, Singapore, 2007.

[9] R.K. Curey, M.E. Ash, L.O. Thielman, and C.H. Baker. Proposed IEEE inertial systems terminology and other inertial sensor standards. *Position Location and Navigation Symposium (PLANS2004)*, pages 83–90, 2004.

[10] V.T. Dau, D.V. Dao andT. Shiozawa, H. Kumagai, and S. Sugiyama. Development of a dual-axis thermal convective gas gyroscope. *Journal of Micromechanics and Microengineering*, 16:1301–1306, 2006.

[11] V.T. Dau, D.V. Dao, T. Shiozawa, H. Kumagai, and S. Sugiyama. A single-axis thermal convective gas gyroscope. *Sensors Mator*, 17(8):453–463, 2005.

[12] M. Esmaeili, M. Durali, and N. Jalili. Ring microgyroscope modeling and performance evaluation. *Journal of Vibration and Control*, 12(5):537–553, 2006.

[13] M. Gad-el-Hak. *The MEMS Handbook: MEMS Introduction and Fundamentals*. CRC Press, Boca Raton, Florida, 2006.

[14] B.J. Gallacher, J.A. Neasham, J.S. Burdess, and A.J. Harris. Initial test results from a 3-axis vibrating ring gyroscope. *Journal of Physics*, 34:662–667, 2006.

[15] A.A. Groenwold and L.F.P. Etman. Quick start manual for the SAO-i algorithm. Unpublished material available as an internal University of Stellenbosch report, 2008.

[16] A.A. Groenwold and L.F.P. Etman. Sequential approximate optimization using dual subproblems based on incomplete series expansions. *Struct. Multidisc. Opt.*, 2008. In press. Available online, DOI: 10.1007/s00158-007-0197-0.

[17] A.A. Groenwold, L.F.P. Etman, and D.W. Wood. Approximated approximations for SAO. Unpublished material available as an internal University of Stellenbosch report, 2008.

[18] A.A. Groenwold, L.F.P. Etman, D.W. Wood, and S. Tosserams. A globally convergent algorithm for large-scale simulation-based optimization using conservative convex separable diagonal quadratic approximations. In *Proc. 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Victoria, British Columbia, Canada, September 2008. Accepted, but not presented.

[19] A.A. Groenwold, D.N. Wilke, and S. Kok. Engineering applications of gradient-only optimization to avoid spurious minima. Unpublished material available as an internal University of Pretoria report, 2008.

[20] Y. Kagawa, N. Wakatsuki, T. Tsuchiya, and Y. Terada. A tubular piezoelectric vibratory gyroscope. *IEEE Sensors Journal*, 6(2):325–330, 2006.

[21] C.T. Kelley. Iterative methods for optimization. *Frontiers in Applied Mathematics*, 18:0–180, 1999.

[22] M.J. Madou. *Fundamentals of Microfabrication: The Science of Miniaturization*. CRC Press, Boca Raton, Florida, 2 edition, 2002.

[23] Memsnet. The beginner's guide to MEMS processing. *http://www.memsnet.org/mems/processes/*, Retrieved November, 2007.

[24] W. Menz, J. Mohr, and O. Paul. *Microsystem Technology*. Wiley-VCH, 2001.

[25] H. Moussa and R. Bourquin. Theory of direct frequency output vibrating gyroscopes. *IEEE Sensors Journal*, 6:310–315, 2006.

[26] M.W. Putty and K. Najafi. A micromachined vibrating ring gyroscope. *Digest Solid-State Sensors and Actuators Workshop Hilton Head Island*, pages 213–220, 1994.

[27] Report Buyer. New research outlines status of the mems industry. *http://www.prlog.org/10132641-new-research-outlines-status-of-the-mems-industry.html*, Retrieved November, 2008.

[28] S.D. Senturia. *Microsystem design*. Kluwer Academic Publishers, Dordrecht, Netherlands, 6 edition, 2001.

[29] J.A. Snyman. *Practical Mathematical Optimization*. Springer, New York, 2005.

[30] K. Svanberg. The method of moving asymptotes - a new method for structural optimization. *Int. J. Numer. Meth. Eng.*, 24:359–373, 1987.

[31] K. Svanberg. *Some modeling aspects for the fortran implementation of MMA*, 2004.

[32] K. Svanberg. On a globally convergent version of MMA. In *Proc. Seventh World Congress on Structural and Multidisciplinary Optimization*, 2007.

[33] F.E.H. Tay and W.O. Choong. *Microfluidic and BioMEMS Applications*. Kluwer Academic Publishers, Dordrecht, Netherlands, 2002.

[34] H. Yang, M. Bao, H. Yin, and S. Shen. Two-dimensional excitation operation mode and phase detection scheme for vibratory gyroscopes. *Journal of Micromechanics and microengineering*, 12:193–197, 2002.

[35] J.S. Yang and H.Y. Fang. Analysis of a rotating elastic beam with piezoelectric films as an angular rate sensor. *IEEE Transactions on Ultrasonics, Ferroelectrics and Frequency Control*, 49(6):798–804, 2002.

[36] J. Zhou, G. Yan, Y. Zhu, Z. Xiao, and J. Fan. Design and fabrication of a microfluidic angular rate sensor. In *18th IEEE International Conference on Micro Electro Mechanical Systems (MEMS2005)*, pages 363–366, Miami, USA, 2005.

Appendices

A. Deflection of a membrane by means of piezoelectric actuation

The volumetric displacement of a circular piezoelectric disc is described by Tay [33] as

$$V_{total} = \int_0^b \int_0^{2\pi} w_I r d\theta dr + \int_b^a \int_0^{2\pi} w_{II} r d\theta dr, \tag{.0.1}$$

where r is the radial distance in polar coordinate and θ is the angle in radians. Parameters w_I and w_{II} describe the deflection of the membrane and the piezoelectric disc respectively. They can be given as

$$w_I = L_3 \frac{t_m}{4}(b^2 - r^2)U + \frac{t_m ab}{2}\frac{C_4 L_3}{L_2}(\frac{C_3 L_2}{C_4} - L_1)U \tag{.0.2}$$

and

$$w_{II} = \frac{t_m ab}{2}\frac{C_4 L_3}{L_2}(\frac{C_3 L_2}{C_4} - L_1)U + L_3 \frac{t_m b}{2} F_1 U r - \frac{t_m b}{2a^2}\frac{C_4 L_3}{L_2} F_2 U r^3, \tag{.0.3}$$

where C_p^E is the Young's modulus of the piezoelectric disc at constant electric field, d_{31} is the piezoelectric constant, t_m is the thickness of the membrane, U is the voltage applied across the piezoelectric disc, a is radius of the membrane (distance from the center to supports), b is the radius of the piezoelectric disc and v is Poisson's ratio.

The constants C_3, C_4, L_1, L_2 and L_3 are given as:

$$
\begin{aligned}
C_3 &= \frac{1+v}{2}\frac{b}{a}\ln(\frac{a}{b}) + (\frac{1-v}{4})(\frac{a}{b} - \frac{b}{a}) \\
C_4 &= \frac{1}{2}[(1+v)\frac{b}{a} + (1-v)\frac{a}{b}] \\
L_1 &= \frac{b}{4a}\{[\frac{b^2}{a} + 1]\ln(\frac{a}{b}) + \frac{b^2}{a} - 1\} \\
L_2 &= \frac{b}{4a}[\frac{b^2}{a} - 1 + 2\ln(\frac{a}{b})] \\
L_3 &= \frac{c_p^E d_{31}}{D_e(1+v_e)}
\end{aligned}
\tag{.0.4}
$$

The functions F_1 and F_2 are given by:

$$
\begin{aligned}
F_1 &= \frac{1+v}{2}\frac{b}{r}\ln(\frac{r}{b}) + \frac{1-v}{4}(\frac{r}{b} - \frac{b}{r}) \\
F_2 &= \frac{b}{4r}\{[\frac{b^2}{r} + 1]\ln(\frac{r}{b}) + (\frac{b}{r})^2 - 1\}\langle r-b\rangle^0
\end{aligned}
\tag{.0.5}
$$

B. FORTRAN implementation of SAO algorithm using ISE based approximations

Setup in FORTRAN: Initialize.f

```
      subroutine INITIALIZE (n, ni, x, x_lower, x_upper)
      implicit         none
      include          'CommonStuff.cmn'
      integer          nseg,npres,meshmult,ns,ns1,nd ! problem specific
      integer          i,j,n,ni
      double precision x(nmax),x_lower(nmax),x_upper(nmax)
      double precision pres(250),alphai              ! problem specific
c-----------------------------------------------------------------------c
c    The author of this subroutine is Surika Vosloo: gyro problem
c-----------------------------------------------------------------------c
      approx_f         = 1         ! select approx_f (see the manual)
      approx_c         = 1         ! select approx_c (see the manual)
      force_converge   = 1         ! enforce convergence? 0=No; 1=Yes
      solver_dual      = 1         ! 1=bfgs, 2 = cg-descent
!
      xtol             = 1.d-5     ! specify the x-tolerance
      outermax         = 5000      ! the number of outer loops allowed
      innermax         = 100       ! the number of inner loops allowed
      dml_infinity     = 1.d0      ! the infinity move limit
      biglam           = 1.d8      ! upper bound on the dual variables
      ksubmax          = 100000    ! max no. of subproblem evaluations
      tlimit           = 100000.d0 ! max time allowed per subproblem
      feaslim          = 1.d-8     ! feasibility limit
!
      deltx            = 1.d-8     ! finite difference increment
```

```
      finite_diff      = .false.   ! use finite differences?
!
      atol  = 1.0d-6               ! min curvature of the objective func
      btol  = 1.0d-6               ! min curvatures of constraint funcs
      allow_f = .true.             ! allow escape from conservative loop
      allow_c = .true.             ! allow escape from conservative loop
      random_start     = .false.   ! use random starting point?
!
      n                = 8
      ni               = 10
!
      x(1)             = 30.d0      ! Iccr [uA]
      x(2)             = 400.d-3    ! L_hw [mm]
      x(3)             = 2.5d-3     ! h_hw [mm]
      x(4)             = 10.d-3     ! w_hw [mm]
      x(5)             = 25.d0      ! w_ch [mm]
      x(6)             = 3.5d3      ! vm   [mm/s]
      x(7)             = 7.5d0      ! L_ch [mm]
      x(8)             = 25.d0      ! h_ch [mm]
      x_lower(1)       = 1.d0       ! Iccr
      x_lower(2)       = 100.d-3    ! L_hw
      x_lower(3)       = 1.d-3      ! h_hw
      x_lower(4)       = 5.d-3      ! w_hw
      x_lower(5)       = 1.d0       ! w_ch
      x_lower(6)       = 0.1d3      ! vm
      x_lower(7)       = 1.d0       ! L_ch
      x_lower(8)       = 1.d0       ! h_ch
      x_upper(1)       = 100.d0     ! Iccr
      x_upper(2)       = 1000.d-3   ! L_hw
      x_upper(3)       = 100.d-3    ! h_hw
      x_upper(4)       = 100.d-3    ! w_hw
      x_upper(5)       = 50.d0      ! w_ch
      x_upper(6)       = 10.d3      ! vm
      x_upper(7)       = 30.d0      ! L_ch
      x_upper(8)       = 50.d0      ! h_ch
!
      return
      end subroutine INITIALIZE
```

Setup in FORTRAN: Functions.f

```
      subroutine FUNCTIONS (n,ni,x,f,c)
!----------------------------------------------------------------------!
!  Compute the objective function f and the inequality constraint      !
!  functions c(j), j=1,ni                                              !
!----------------------------------------------------------------------!
      implicit none
      integer          i, j, n, ni, np
      double precision f, x(n), c(ni)
      include 'CommonStuff.cmn'
      double precision kinv, dynu, rho, cp, cv, kf, ao, gamma, mypi
      double precision Ro, alpha, Ahw, Lc, D_ch, L_hw, w_hw, h_hw,resis
      double precision Kn, Ref, Ma, Pr, Rehw, h, b, Cre, nre, prop
      double precision vm, Kappa, wr, dv, w_ch, gap, L_ch, defl, h_ch
      double precision Qmax, m_hw, a_imp, An, Aobs, CD, FD, QD, rho_hw
      double precision Vout, Vin, Iin, Iccr, Phw, Rhw, dR
!
      np=ni ! for old times sake
c----------------------------------------------------------------------c
c    The author of this subroutine is Surika Vosloo: gyro problem
c----------------------------------------------------------------------c
      mypi    = 3.1415926d0
!     --------------------
!     Variables to be optimized
!     --------------------
      Iccr  = x(1)              ! Constant current across hw [uA]
      L_hw  = x(2)              ! Length of hot wire [mm]
      h_hw  = x(3)              ! Height of hot wire [mm]
      w_hw  = x(4)              ! Width of hot wire [mm]
      w_ch  = x(5)              ! Width of chamber [mm]
      vm    = x(6)              ! Mean velocity across hwire [mm/s]
      L_ch  = x(7)              ! Length from nozzle to hw [mm]
      h_ch  = x(8)              ! Height of chamber[mm]
!     -------------------
!     Inert Neon gas properties : Neon @ 1atm pressure, T = 25 degC
!     -------------------
      dynu  = 3142.d-11         ! Dynamic viscocity [kg/mm.s]
      cp    = 1030.d9           ! Specific heat  pressure [nJ/kg.C]
```

```
      cv    = 618.d9            ! Specific heat  viscocity [nJ/kg]
      kf    = 0.0491d6          ! Thermal conductivity [nW/mm.K]
      rho   = 0.846d-9          ! Density [kg/mm^3]
      ao    = 435.d3            ! Speed of sound in the gas [mm/s]
      kinv  = dynu/rho          ! Kinematic viscocity [mm^2/s]
!     --------------------
!     Hot wire properties: Doped Si thermistor
!     --------------------
      Ahw   = (2.d0*L_hw*w_hw)+(2.d0*L_hw*h_hw)
                                !Surface area of Hot wire [m^2]
      D_ch  = (2.d0*w_ch*h_ch)/(h_ch + w_ch)
                                ! Chamber hydraulic diameter p.422
      resis = 0.35d-2           ! Resistivity [kohm.mm]
      alpha = 6000.d-6          ! Thermal coef of R [1/degC]
      Ro    = resis*L_hw*(1.d0 + alpha*5.d0)/(h_hw*w_hw)
                                ! Reference resistance [ohm]
!     ---------------------
!     Rotation modeling
!     ---------------------
      wr    = 150.d0            ! Angular rotation [deg/s] --
      defl  = (wr*L_ch**2)/vm   ! Deflection of flow profile [mm]
      prop  = 1.d0              ! Proportionality constant
      dv    = -prop*defl        ! Delta_v eq. 3.5.17
!     ---------------------
!     Fluid modeling assumptions
!     ---------------------
      gamma = cp/cv             ! Specific heat ratio
      Ref   = (vm*D_ch)/kinv    ! Reynolds no.: gas flow in chamber
      Ma    = vm/ao             ! Mach Number
      Kn    = dsqrt(mypi*gamma/2.d0)*(Ma/Ref)   ! Knudsen Number
!     --------------------
!     CCR
!     --------------------
      Pr    = (cp*dynu)/kf      ! Prandtl number
      Cre   = 0.56d0            ! Empirical constant varying with Re
      nre   = 0.8d0             ! Empirical constant varying with Re
      b     = (1.1d0*kf*Cre*w_hw**(nre-1)*Pr**0.31)/(kinv**nre)
                                ! Empirical value calculated from Nu
      h     = b*vm**nre         ! Heat Transfer Coefficient [W/m^2]
```

```
!     --------------------
!     Structural analysis
!     --------------------
      rho_hw= 2330.d-9           ! Density of hw material [kg/mm^3]
      Qmax  = 63.9d6             ! Maximum stress for Silicon [kPa]
      a_imp = Qmax/(rho_hw*L_hw) ! Impact accelration [m/s^2]
      CD    = 0.9d0              ! Drag force coefficient(p372)
      QD    = 0.5d0*rho*CD*vm**2 ! Drag force [kPa]
!     --------------------
!     Amplifier circuit and Power consumption
!     --------------------
      Rhw   = h*Ahw*Ro/(h*Ahw - alpha*Iccr**2*Ro) ! Hw res@dv=0 [ohm]
      Vin   = Iccr*(2.d0*Rhw)    ! Source Voltage [V]
      dR   = (-alpha*Ahw*Iccr**2*Ro**2*b*nre*vm**(nre-1)*dv)
     &        /((Ahw*b*vm**nre - Ro*Iccr**2*alpha)**2)
                                 ! Resistance diffential[ohm]
      Iin   = Vin/(2.d0*Ro) + Iccr    ! Source current[A]
      Phw   = Vin*Iin            ! Power consumption[W]
!     --------------------
!     Objective function and constraints
!     --------------------
      f     = dR*Iccr
      f     = -f                 ! Maximize Amplified Vout [mV]
      c(1)  = Ma/0.3d0 - 1.d0    ! Ma  < 0.3   for incompressibility
      c(2)  = Kn/1.d-3 - 1.d0    ! Kn  < 10^-3 for Navier-Stokes flow
      c(3)  = Ref/2100.d0  - 1.d0   ! Ref < 2100 for laminar flow
      c(4)  = 1.d0 - Ref/100.d0  ! Ref > 100 for sufficient inertia
      c(5)  = 1.d0 - a_imp/(15000.d0*9.81d3) ! Impact acc > 15000g
      c(6)  = QD/Qmax  - 1.d0    ! Stress due to drag < Qmax
      c(7)  = Phw/1.d6 - 1.d0    ! Power < 1mW
      c(8)  = alpha*Iccr**2*Ro - h*Ahw       ! Ensure Vin > 0
      c(9)  = 3.d0*L_hw - w_ch   ! Gap >0
      c(10) = defl - 0.5d0*(w_ch - L_hw) ! Defl in center of hw
!
      return
      end subroutine FUNCTIONS
```

Setup in FORTRAN: Gradients.f

```
      subroutine GRADIENTS (n,x,ni,gf,gc)
!--------------------------------------------------------------------!
!  Compute the gradients of the objective f, and the inequality      !
!  constraints c, w.r.t. the variables x(i)                          !
!--------------------------------------------------------------------!
      implicit none
      integer          i, j, n, ni, np
      double precision x(n), gf(n), gc(ni,n)
      double precision dfdx(ni*n)
      double precision kinv, dynu, rho, cp, cv, kf, ao, gamma, mypi
      double precision Ro, alpha, Ahw, Lc, D_ch, L_hw, w_hw, h_hw,resis
      double precision Kn, Ref, Ma, Pr, Rehw, h, b, Cre, nre, prop
      double precision vm, Kappa, wr, dv, w_ch, gap, L_ch, defl, h_ch
      double precision Qmax, m_hw, a_imp, An, Aobs, CD, FD, QD, rho_hw
      double precision Vout, Vin, Iin, Iccr, Phw, Rhw, dR
!
      include 'CommonStuff.cmn'
      np=ni ! for old times sake
c--------------------------------------------------------------------c
c    The author of this subroutine is Surika Vosloo: gyro problem
c--------------------------------------------------------------------c
c     Constants imported from Functions
c-------------------------------------c
      mypi  = 3.1415926d0
      dynu  = 3142.d-11         ! Dynamic viscocity [kg/m.s]
      cp    = 1030.d9           ! Specific heat  pressure [J/kg.C]
      cv    = 618.d9            ! Specific heat  viscocity [J/kg]
      kf    = 0.0491d6          ! Thermal conductivity [W/m.K]
      rho   = 0.846d-9          ! Density [kg/m^3]
      ao    = 435.d3            ! Speed of sound in the gas [m/s]
      kinv  = dynu/rho          ! Kinematic viscocity [m^2/s]
      resis = 0.35d-2           ! Resistivity [ohm.m]
      alpha = 6000.d-6          ! Thermal coef of R [1/degC]
      wr    = 150.d0            ! Angular rotation [deg/s]
      Cre   = 0.56d0            ! Empirical constant varying with Re
      nre   = 0.8d0             ! Empirical constant varying with Re
      gamma = cp/cv             ! Specific heat ratio
```

```
      prop  = 1.d0              ! Proportionality constant
      rho_hw= 2330.d-9          ! Density of hw material [kg/m^3]
      Qmax  = 63.9d6            ! Maximum stress for Silicon [Pa]
      CD    = 0.9d0             ! Drag force coefficient
c-------------------------c
c     Gradients
c-------------------------c
      gf(1) =-(33.D0/10.D0*alpha*(2.d0*x(2)*x(3)+2.d0*x(2)*x(4))*x(1)**2
     &         *resis**2*x(2)**2*(1.d0+5.d0*alpha)**2/x(3)**2/x(4)**2*kf
     &         *Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*nre*
     &         x(6)**(nre-1)*prop*wr*x(7)**2/x(6)/(11.D0/10.D0*kf*Cre*
     &         x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(
     &         2.d0*x(2)*x(3)+2.d0*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(
     &         1.d0+5.d0*alpha)/x(3)/x(4))**2+22.D0/5.D0*alpha**2*(2.d0*
     &         x(2)*x(3)+2.d0*x(2)*x(4))*x(1)**4*resis**3*x(2)**3*(1.d0+
     &         5.d0*alpha)**3/x(3)**3/x(4)**3*kf*Cre*x(4)**(nre-1)*(cp*
     &         dynu/kf)**(0.31)/kinv**nre*nre*x(6)**(nre-1)*prop*wr*x(7)
     &         **2/x(6)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**
     &         (0.31)/kinv**nre*x(6)**nre*(2.d0*x(2)*x(3)+2.d0*x(2)*x(4)
     &         )-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**
     &         3)                                              !
      gf(2) =-((11.D0/10.D0*alpha*(2*x(3)+2*x(4))*x(1)**3*resis**2*x(2)
     &         **2*(1.d0+5.d0*alpha)**2/x(3)**2/x(4)**2*kf*Cre*x(4)**(
     &         nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*nre*x(6)**(nre-1)*
     &         prop*wr*x(7)**2/x(6)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(
     &         cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*
     &         x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/
     &         x(3)/x(4))**2)+(11.D0/5.D0*alpha*(2*x(2)*x(3)+2*x(2)*x(4)
     &         )*x(1)**3*resis**2*x(2)*(1.d0+5.d0*alpha)**2/x(3)**2/x(4)
     &         **2*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*
     &         nre*x(6)**(nre-1)*prop*wr*x(7)**2/x(6)/(11.D0/10.D0*kf*
     &         Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**
     &         nre*(2.d0*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)
     &         *(1.d0+5.d0*alpha)/x(3)/x(4))**2-11.D0/5.D0*alpha*(2*x(2)
     &         *x(3)+2.d0*x(2)*x(4))*x(1)**3*resis**2*x(2)**2*(1.d0+5.d0
     &         *alpha)**2/x(3)**2/x(4)**2*kf*Cre*x(4)**(nre-1)*(cp*dynu/
     &         kf)**(0.31)/kinv**nre*nre*x(6)**(nre-1)*prop*wr*x(7)**2/
     &         x(6)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
     &         0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha
```

```
&         *x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**3*(
&         11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
&         kinv**nre*x(6)**nre*(2.d0*x(3)+2.d0*x(4))-alpha*x(1)**2*
&         resis*(1.d0+5.d0*alpha)/x(3)/x(4))))      !
 gf(3) = -((11.D0/5.D0*alpha*x(2)**3*x(1)**3*resis**2*(1.d0+5.d0*
&         alpha)**2/x(3)**2/x(4)**2*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&         kf)**(0.31)/kinv**nre*nre*x(6)**(nre-1)*prop*wr*x(7)**2/
&         x(6)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**
&         (0.31)/kinv**nre*x(6)**nre*(2.d0*x(2)*x(3)+2.d0*x(2)*x(4)
&         )-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**
&         2)+(-11.D0/5.D0*alpha*(2.d0*x(2)*x(3)+2.d0*x(2)*x(4))*
&         x(1)**3*resis**2*x(2)**2*(1.d0+5.d0*alpha)**2/x(3)**3/
&         x(4)**2*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**
&         nre*nre*x(6)**(nre-1)*prop*wr*x(7)**2/x(6)/(11.D0/10.D0*
&         kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)
&         **nre*(2.d0*x(2)*x(3)+2.d0*x(2)*x(4))-alpha*x(1)**2*resis
&         *x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**2-11.D0/5.D0*alpha*(
&         2.d0*x(2)*x(3)+2.d0*x(2)*x(4))*x(1)**3*resis**2*x(2)**2*(
&         1.d0+5.d0*alpha)**2/x(3)**2/x(4)**2*kf*Cre*x(4)**(nre-1)*
&         (cp*dynu/kf)**(0.31)/kinv**nre*nre*x(6)**(nre-1)*prop*wr*
&         x(7)**2/x(6)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&         kf)**(0.31)/kinv**nre*x(6)**nre*(2.d0*x(2)*x(3)+2.d0*x(2)
&         *x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/
&         x(4))**3*(11.D0/5.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
&         0.31)/kinv**nre*x(6)**nre*x(2)+alpha*x(1)**2*resis*x(2)*(
&         1.d0+5.d0*alpha)/x(3)**2/x(4))))          !
 gf(4) = -((11.D0/5.D0*alpha*x(2)**3*x(1)**3*resis**2*(1.d0+5.d0*
&         alpha)**2/x(3)**2/x(4)**2*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&         kf)**(0.31)/kinv**nre*nre*x(6)**(nre-1)*prop*wr*x(7)**2/
&         x(6)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**
&         (0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-
&         alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**2-
&         11.D0/5.D0*alpha*(2.d0*x(2)*x(3)+2.d0*x(2)*x(4))*x(1)**3*
&         resis**2*x(2)**2*(1.d0+5.d0*alpha)**2/x(3)**2/x(4)**3*kf*
&         Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*nre*x(6)
&         **(nre-1)*prop*wr*x(7)**2/x(6)/(11.D0/10.D0*kf*Cre*x(4)**
&         (nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2.d0*
&         x(2)*x(3)+2.d0*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+
&         5.d0*alpha)/x(3)/x(4))**2)+(11.D0/10.D0*alpha*(2.d0*x(2)*
```

```
&         x(3)+2.d0*x(2)*x(4))*x(1)**3*resis**2*x(2)**2*(1.d0+5.d0*
&         alpha)**2/x(3)**2/x(4)**3*kf*Cre*x(4)**(nre-1)*(nre-1)*(
&         cp*dynu/kf)**(0.31)/kinv**nre*nre*x(6)**(nre-1)*prop*wr*
&         x(7)**2/x(6)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&         kf)**(0.31)/kinv**nre*x(6)**nre*(2.d0*x(2)*x(3)+2.d0*x(2)
&         *x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/
&         x(4))**2)+(-11.D0/5.D0*alpha*(2.d0*x(2)*x(3)+2.d0*x(2)*
&         x(4))*x(1)**3*resis**2*x(2)**2*(1.d0+5.d0*alpha)**2/x(3)
&         **2/x(4)**2*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
&         kinv**nre*nre*x(6)**(nre-1)*prop*wr*x(7)**2/x(6)/(11.D0/
&         10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**
&         nre*x(6)**nre*(2.d0*x(2)*x(3)+2.d0*x(2)*x(4))-alpha*x(1)
&         **2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**3*(11.D0/
&         10.D0*kf*Cre*x(4)**(nre-1)*(nre-1)/x(4)*(cp*dynu/kf)**
&         (0.31)/kinv**nre*x(6)**nre*(2.d0*x(2)*x(3)+2.d0*x(2)*
&         x(4))+11.D0/5.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**
&         (0.31)/kinv**nre*x(6)**nre*x(2)+alpha*x(1)**2*resis*x(2)*
&         (1.d0+5.d0*alpha)/x(3)/x(4)**2)))         !
 gf(5) = 0.d0                                       !
 gf(6) = -((11.D0/10.D0*alpha*(2*x(2)*x(3)+2*x(2)*x(4))*x(1)**3*
&         resis**2*x(2)**2*(1.d0+5.d0*alpha)**2/x(3)**2/x(4)**2*kf*
&         Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*nre*x(6)
&         **(nre-1)*(nre-1)/x(6)**2*prop*wr*x(7)**2/(11.D0/10.D0*kf
&         *Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**
&         nre*(2.d0*x(2)*x(3)+2.d0*x(2)*x(4))-alpha*x(1)**2*resis*
&         x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**2)+(-11.D0/10.D0*alpha
&         *(2.d0*x(2)*x(3)+2.d0*x(2)*x(4))*x(1)**3*resis**2*x(2)**2
&         *(1.d0+5.d0*alpha)**2/x(3)**2/x(4)**2*kf*Cre*x(4)**(nre-1
&         )*(cp*dynu/kf)**(0.31)/kinv**nre*nre*x(6)**(nre-1)*prop*
&         wr*x(7)**2/x(6)**2/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*
&         dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2.d0*x(2)*x(3)+2.d0
&         *x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/
&        x(3)/x(4))**2-121.D0/50.D0*alpha*(2.d0*x(2)*x(3)+2.d0*x(2)
&         *x(4))**2*x(1)**3*resis**2*x(2)**2*(1.d0+5.d0*alpha)**2/
&         x(3)**2/x(4)**2*kf**2*Cre**2*(x(4)**(nre-1))**2*(cp*dynu/
&         kf)**(31.D0/50.D0)/(kinv**nre)**2*nre**2*x(6)**(nre-1)*
&         prop*wr*x(7)**2/x(6)**2/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)
&         *(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2.d0*x(2)*x(3)
&         +2.d0*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*
```

```
     &         alpha)/x(3)/x(4))**3*x(6)**nre))          !
      gf(7) =-(11.D0/5.D0*alpha*(2.d0*x(2)*x(3)+2.d0*x(2)*x(4))*x(1)**3*
     &         resis**2*x(2)**2*(1.d0+5.d0*alpha)**2/x(3)**2/x(4)**2*kf*
     &         Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*nre*x(6)
     &         **(nre-1)*prop*wr*x(7)/x(6)/(11.D0/10.D0*kf*Cre*x(4)**(
     &         nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2.d0*
     &         x(2)*x(3)+2.d0*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+
     &         5.d0*alpha)/x(3)/x(4))**2)                !
      gf(8) = 0.d0                                       !
!
      gc(1,1)  = 0.d0                                    !
      gc(1,2)  = 0.d0                                    !
      gc(1,3)  = 0.d0                                    !
      gc(1,4)  = 0.d0                                    !
      gc(1,5)  = 0.d0                                    !
      gc(1,6)  = 1.d0/(ao*0.3d0)                         !
      gc(1,7)  = 0.d0                                    !
      gc(1,8)  = 0.d0                                    !
!
      gc(2,1)  = 0.d0                                    !
      gc(2,2)  = 0.d0                                    !
      gc(2,3)  = 0.d0                                    !
      gc(2,4)  = 0.d0                                    !
      gc(2,5)  = (-dsqrt(0.7074237631354954D16)*dsqrt(cp/cv)/ao/x(5)**2/
     &           x(8)*(x(8)+x(5))*kinv/134217728.d0+dsqrt(
     &           0.7074237631354954D16)*dsqrt(cp/cv)/ao/x(5)/x(8)*kinv/
     &           134217728.d0)/1.d-3                   !
      gc(2,6)  = 0.d0                                    !
      gc(2,7)  = 0.d0                                    !
      gc(2,8)  =(-dsqrt(0.7074237631354954D16)*dsqrt(cp/cv)/ao/x(5)/x(8)
     &           **2*(x(8)+x(5))*kinv/134217728.d0+dsqrt(
     &           0.7074237631354954D16)*dsqrt(cp/cv)/ao/x(5)/x(8)*kinv/
     &           134217728.d0)/1.d-3                   !
!
      gc(3,1)  = 0.d0                                    !
      gc(3,2)  = 0.d0                                    !
      gc(3,3)  = 0.d0                                    !
      gc(3,4)  = 0.d0                                    !
      gc(3,5)  = (2.d0*x(6)*x(8)/(x(8)+x(5))/kinv-2.d0*x(6)*x(5)*x(8)/(
```

```
     &           x(8)+x(5))**2/kinv)/2100.d0             !
      gc(3,6)  = (2.d0*x(5)*x(8)/(x(8)+x(5))/kinv)/2100.d0 !
      gc(3,7)  = 0.d0                                     !
      gc(3,8)  = (2.d0*x(6)*x(5)/(x(8)+x(5))/kinv-2.d0*x(6)*x(5)*x(8)/(
     &           x(8)+x(5))**2/kinv)/2100.d0             !
!
      gc(4,1)  = 0.d0                                     !
      gc(4,2)  = 0.d0                                     !
      gc(4,3)  = 0.d0                                     !
      gc(4,4)  = 0.d0                                     !
      gc(4,5)  =(-2.d0*x(6)*x(8)/(x(8)+x(5))/kinv+2.d0*x(6)*x(5)*x(8)/(
     &           x(8)+x(5))**2/kinv)/100.d0              !
      gc(4,6)  = (-2.d0*x(5)*x(8)/(x(8)+x(5))/kinv)/100.d0
      gc(4,7)  = 0.d0                                     !
      gc(4,8)  =(-2.d0*x(6)*x(5)/(x(8)+x(5))/kinv+2.d0*x(6)*x(5)*x(8)/(
     &           x(8)+x(5))**2/kinv)/100.d0              !
!
      gc(5,1)  = 0.d0                                     !
      gc(5,2)  = (Qmax/rho_hw/x(2)**2)/(1500.d0*9.81d3)   !
      gc(5,3)  = 0.d0                                     !
      gc(5,4)  = 0.d0                                     !
      gc(5,5)  = 0.d0                                     !
      gc(5,6)  = 0.d0                                     !
      gc(5,7)  = 0.d0                                     !
      gc(5,8)  = 0.d0                                     !
!
      gc(6,1)  = 0.d0                                     !
      gc(6,2)  = 0.d0                                     !
      gc(6,3)  = 0.d0                                     !
      gc(6,4)  = 0.d0                                     !
      gc(6,5)  = 0.d0                                     !
      gc(6,6)  = (rho*x(6)*CD)/Qmax                       !
      gc(6,7)  = 0.d0                                     !
      gc(6,8)  = 0.d0                                     !
!
      gc(7,1) = ((11.D0/5.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
     &          kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))*resis*x(2)
     &          *(1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/10.D0*kf*Cre*x(4)**
     &          (nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)
```

```
&          *x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*
&          alpha)/x(3)/x(4))*(11.D0/10.D0*x(1)*kf*Cre*x(4)**(nre-1)
&          *(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+
&          2*x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&          kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4)
&          )-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))+
&          x(1)))+(22.D0/5.D0*x(1)**2*kf*Cre*x(4)**(nre-1)*(cp*dynu
&          /kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*
&          x(4))*resis**2*x(2)**2*(1.d0+5.d0*alpha)**2/x(3)**2/x(4)
&          **2/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**
&          (0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-
&          alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**2
&          *(11.D0/10.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
&          0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))/(
&          11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
&          kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*
&          x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))+x(1))*
&          alpha)+((11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&          kf)**(0.31)/kinv**nre*x(6)**nre)*((2*x(2)*x(3)+2*x(2)*
&          x(4))*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/
&          10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**
&          nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*
&          resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))*(11.D0/10.D0*kf
&          *Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**
&          nre*(2*x(2)*x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**
&          (nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)
&          *x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*
&          alpha)/x(3)/x(4))+11.D0/5.D0*x(1)**2*kf*Cre*x(4)**(nre-1
&          )*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+
&          2*x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&          kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4)
&          )-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))
&          **2*alpha*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)+1))))
&          /1.d6 !
 gc(7,2) = ((11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**
&          (0.31)/kinv**nre*x(6)**nre*(2*x(3)+2*x(4))*resis*x(2)*(
&          1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/10.D0*kf*Cre*x(4)**(
&          nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*
&          x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*
```

```
&          alpha)/x(3)/x(4))*(11.D0/10.D0*x(1)*kf*Cre*x(4)**(nre-1)
&          *(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2
&          *x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&          kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4)
&          )-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))+
&          x(1)))+(11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf
&          )**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))*
&          resis*(1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/10.D0*kf*Cre*
&          x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(
&          2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+
&          5.d0*alpha)/x(3)/x(4))*(11.D0/10.D0*x(1)*kf*Cre*x(4)**(
&          nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*
&          x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*
&          dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)
&          *x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/
&          x(4))+x(1)))+(-11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*
&          dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)
&          *x(4))*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/
&          10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**
&          nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*
&          resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**2*(11.D0/10.D0*
&          x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre
&          *x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre
&          *x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
&          (2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0
&          +5.d0*alpha)/x(3)/x(4))+x(1))*(11.D0/10.D0*kf*Cre*x(4)**
&          (nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(3)
&          +2*x(4))-alpha*x(1)**2*resis*(1.d0+5.d0*alpha)/x(3)/x(4)
&          ))+((11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**
&          (0.31)/kinv**nre*x(6)**nre )*((2*x(2)*x(3)+2*x(2)*x(4))*
&          resis*x(2)*(1.d0+5.d0*alpha))*(1/x(3)/x(4)/(11.D0/10.D0*
&          kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)
&          **nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)
&          *(1.d0+5.d0*alpha)/x(3)/x(4))*(11.D0/10.D0*x(1)*kf*Cre*
&          x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
&          (2*x(3)+2*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*
&          dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)
&          *x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/
&          x(4))-11.D0/10.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)
```

```
     &          **(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))/(
     &          11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
     &          kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)
     &          **2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**2*(11.D0/
     &          10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**
     &          nre*x(6)**nre*(2*x(3)+2*x(4))-alpha*x(1)**2*resis*(1.d0+
     &          5.d0*alpha)/x(3)/x(4))))))/1.d6                !
      gc(7,3) = ((22.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
     &          0.31)/kinv**nre*x(6)**nre*x(2)**2*resis*(1.d0+5.d0*alpha
     &          )/x(3)/x(4)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/
     &          kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4)
     &          )-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))*
     &          (11.D0/10.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
     &          0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))/(
     &          11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
     &          kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)
     &          **2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))+x(1)))+(-
     &          11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31
     &          )/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))*resis*
     &          x(2)*(1.d0+5.d0*alpha)/x(3)**2/x(4)/(11.D0/10.D0*kf*Cre*
     &          x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(
     &          2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+
     &          5.d0*alpha)/x(3)/x(4))*(11.D0/10.D0*x(1)*kf*Cre*x(4)**(
     &          nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*
     &          x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*
     &          dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)
     &          *x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/
     &          x(4))+x(1)))+(-11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*
     &          dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)
     &          *x(4))*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/
     &          10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**
     &          nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*
     &          resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**2*(11.D0/10.D0*
     &          x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre
     &          *x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre
     &          *x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
     &          (2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0
     &          +5.d0*alpha)/x(3)/x(4))+x(1))*(11.D0/5.D0*kf*Cre*x(4)**(
     &          nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*x(2)+
```

```
     &          alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)**2/x(4))
     &          )+((11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
     &          0.31)/kinv**nre*x(6)**nre)*((2*x(2)*x(3)+2*x(2)*x(4))*
     &          resis*x(2)*(1.d0+5.d0*alpha))*(1/x(3)/x(4)/(11.D0/10.D0*
     &          kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)
     &          **nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)
     &          *(1.d0+5.d0*alpha)/x(3)/x(4))*(11.D0/5.D0*x(1)*kf*Cre*
     &          x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
     &          x(2)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
     &          0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-
     &          alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))-
     &          11.D0/10.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
     &          0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))/(
     &          11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
     &          kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)
     &          **2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))**2*(11.D0/
     &          5.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre
     &          *x(6)**nre*x(2)+alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*
     &          alpha)/x(3)**2/x(4))))))/1.d6                    !
      gc(7,4) =((((11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(nre-1)/x(4)**2*
     &          (cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*
     &          x(2)*x(4))*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/(11.D0/
     &          10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**
     &          nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*
     &          resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))*(11.D0/10.D0*
     &          x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre
     &          *x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre
     &          *x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
     &          (2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0
     &          +5.d0*alpha)/x(3)/x(4))+x(1)))+(22.D0/5.D0*x(1)*kf*Cre*
     &          x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
     &          x(2)**2*resis*(1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/10.D0*
     &          kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)
     &          **nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)
     &          *(1.d0+5.d0*alpha)/x(3)/x(4))*(11.D0/10.D0*x(1)*kf*Cre*
     &          x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
     &          (2*x(2)*x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre
     &          -1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*
     &          x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*
```

```
&          alpha)/x(3)/x(4))+x(1))))-11.D0/5.D0*x(1)*kf*Cre*x(4)**(
&          nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*
&          x(3)+2*x(2)*x(4))*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)
&          **2/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
&          0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-
&          alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))*(
&          11.D0/10.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
&          0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))/(
&          11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
&          kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)
&          **2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))+x(1)))+
&          (-11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
&          0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))*
&          resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/10.D0*kf*
&          Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**
&          nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*
&          (1.d0+5.d0*alpha)/x(3)/x(4))**2*(11.D0/10.D0*x(1)*kf*
&          Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**
&          nre*(2*x(2)*x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)
&          **(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*
&          x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+
&          5.d0*alpha)/x(3)/x(4))+x(1))*(11.D0/10.D0*kf*Cre*x(4)**
&          (nre-1)*(nre-1)/x(4)*(cp*dynu/kf)**(0.31)/kinv**nre*
&          x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))+11.D0/5.D0*kf*Cre*
&          x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
&          x(2)+alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/
&          x(4)**2))+((11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*
&          dynu/kf)**(0.31)/kinv**nre*x(6)**nre)*((2*x(2)*x(3)+2*
&          x(2)*x(4))*resis*x(2)*(1.d0+5.d0*alpha))*(1/x(3)/x(4))*
&          (1/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
&          0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-
&          alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)))*
&          ((11.D0/10.D0*x(1)*kf*Cre*x(4)**(nre-1)*(nre-1)/x(4)*(
&          cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*
&          x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&          kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*
&          x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/
&          x(4)))+(11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/
&          kf)**(0.31)/kinv**nre*x(6)**nre*x(2)/(11.D0/10.D0*kf*
```

```
&           Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**
&           nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*
&           (1.d0+5.d0*alpha)/x(3)/x(4))-11.D0/10.D0*x(1)*kf*Cre*
&           x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
&           (2*x(2)*x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(
&           nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)
&           *x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*
&           alpha)/x(3)/x(4))**2*(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*
&           (nre-1)/x(4)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(
&           2*x(2)*x(3)+2*x(2)*x(4))+11.D0/5.D0*kf*Cre*x(4)**(nre-1
&           )*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*x(2)+alpha*
&           x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)**2)))))/
&           1.d6!
 gc(7,5) = 0.d0                                          !
 gc(7,6) = ((11.D0/5.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**
&           (0.31)/kinv**nre*x(6)**nre*nre/x(6)*(2*x(2)*x(3)+2*x(2)*
&           x(4))*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/
&           10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**
&           nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*
&           resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))*(11.D0/10.D0*
&           x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre
&           *x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre
&           *x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*
&           (2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0
&           +5.d0*alpha)/x(3)/x(4))+x(1)))+(-121.D0/50.D0*x(1)*kf**2
&           *Cre**2*(x(4)**(nre-1))**2*(cp*dynu/kf)**(31.D0/50.D0)/(
&           kinv**nre)**2*(x(6)**nre)**2*(2*x(2)*x(3)+2*x(2)*x(4))**
&           2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)/(11.D0/10.D0*kf
&           *Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**
&           nre*(2*x(2)*x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(
&           1.d0+5.d0*alpha)/x(3)/x(4))**2*(11.D0/10.D0*x(1)*kf*Cre*
&           x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(
&           2*x(2)*x(3)+2*x(2)*x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre-
&           1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)
&           +2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)
&           /x(3)/x(4))+x(1))*nre/x(6))+((11.D0/5.D0*x(1)*kf*Cre*
&           x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre)*
&           ((2*x(2)*x(3)+2*x(2)*x(4))*resis*x(2)*(1.d0+5.d0*alpha)/
&           x(3)/x(4)/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)
```

```
     &          **(0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-
     &          alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))*(
     &          11.D0/10.D0*x(1)*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
     &          0.31)/kinv**nre*x(6)**nre*nre/x(6)*(2*x(2)*x(3)+2*x(2)*
     &          x(4))/(11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(
     &          0.31)/kinv**nre*x(6)**nre*(2*x(2)*x(3)+2*x(2)*x(4))-
     &          alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4))-
     &          121.D0/100.D0*x(1)*kf**2*Cre**2*(x(4)**(nre-1))**2*(cp*
     &          dynu/kf)**(31.D0/50.D0)/(kinv**nre)**2*(x(6)**nre)**2*(
     &          2*x(2)*x(3)+2*x(2)*x(4))**2/(11.D0/10.D0*kf*Cre*x(4)**(
     &          nre-1)*(cp*dynu/kf)**(0.31)/kinv**nre*x(6)**nre*(2*x(2)
     &          *x(3)+2*x(2)*x(4))-alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*
     &          alpha)/x(3)/x(4))**2*nre/x(6)))))/1.d6   !
      gc(7,7) = 0.d0                                      !
      gc(7,8) = 0.d0                                      !
!
      gc(8,1) = 2.d0*alpha*x(1)*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4) !
      gc(8,2) = alpha*x(1)**2*resis*(1.d0+5.d0*alpha)/x(3)/x(4)-11.D0/
     &          10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/kinv**
     &          nre*x(6)**nre*(2.d0*x(3)+2.d0*x(4))      !
      gc(8,3) = -alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)**2/x(4)
     &          -11.D0/5.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
     &          kinv**nre*x(6)**nre*x(2)                 !
      gc(8,4) = -alpha*x(1)**2*resis*x(2)*(1.d0+5.d0*alpha)/x(3)/x(4)**2
     &          -11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(nre-1)/x(4)*(cp*dynu/
     &          kf)**(0.31)/kinv**nre*x(6)**nre*(2.d0*x(2)*x(3)+2.d0*
     &          x(2)*x(4))-11.D0/5.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)
     &          **(0.31)/kinv**nre*x(6)**nre*x(2)        !
      gc(8,5) = 0.d0                                      !
      gc(8,6) = -11.D0/10.D0*kf*Cre*x(4)**(nre-1)*(cp*dynu/kf)**(0.31)/
     &          kinv**nre*x(6)**nre*nre/x(6)*(2*x(2)*x(3)+2*x(2)*x(4))!
      gc(8,7) = 0.d0                                      !
      gc(8,8) = 0.d0                                      !
!
      gc(9,1)  = 0.d0                                     !
      gc(9,2)  = 3.d0                                     !
      gc(9,3)  = 0.d0                                     !
      gc(9,4)  = 0.d0                                     !
      gc(9,5)  = -1.d0                                    !
```

```
      gc(9,6)  = 0.d0                                    !
      gc(9,7)  = 0.d0                                    !
      gc(9,8)  = 0.d0                                    !
!
      gc(10,1)  =  0.d0                                  !
      gc(10,2)  =  1.D0/2.D0                             !
      gc(10,3)  =  0.d0                                  !
      gc(10,4)  =  0.d0                                  !
      gc(10,5)  =  -1.D0/2.D0                            !
      gc(10,6)  =  -wr*x(7)**2/x(6)**2                   !
      gc(10,7)  =  2.d0*wr*x(7)/x(6)                     !
      gc(10,8)  =  0.d0                                  !
!
      return
      end subroutine GRADIENTS
```

C. MATLAB implementation of SAO algorithm using ISE based approximations

Setup in MATLAB: SAO.m

```
% Authors: S. Vosloo, A.A. Groenwold
% University of Stellenbosch, South Africa
% Date: 05/05/2009
% Implementing SAO algorithm, using BFGS-solver coded by C.T. Kelley
%-----------------------------------------------------------------------%
close all;
clear all;
clc;
profile on;
%-----------------------------------------------------------------------%
%               USER DEFINED OPTIMIZATION PARAMETERS                    %
%-----------------------------------------------------------------------%

%% INITIALIZATION

%% 1. GROENWOLD TEST PROBLEM INITIALIZATION
%    n = 2;               % Number of design variables
%    m = 2;               % Number of constraint functions
%
%    % Initialize design variables x, with row vectors of size n
%    x_i = [1.5,0.5];     % Initial values
%    x_l = [0.2,0.1];     % Lower bounds
%    x_u = [4.0,1.6];     % Upper bounds

%% 2. GYROSCOPE PROBLEM INITIALIZATION
```

```
   n = 8;               % Number of design variables
   m = 10;              % Number of constraint functions

   % Initialize design variables x, with row vectors of size n
   x_i = [30, 400e-3, 2.5e-3, 10e-3, 25, 3.5e3, 7.5, 25];  % Initial values
   x_l = [1, 100e-3, 1e-3, 5e-3, 1, 0.1e3, 1, 1];          % Lower bounds
   x_u = [100, 1000e-3, 100e-3, 100e-3, 50, 10e3, 30, 50]; % Upper bounds

%% 3. SVANBERG'S SNAKE PROBLEM INITIALIZATION
%
%    nd = 10;
%    n = nd*3;                % Number of design variables
%    m = nd*4+1;              % Number of constraint functions
%
%    % Initialize design variables x, with row vectors of size n
%    x_i = zeros(1,n);
%    x_l = zeros(1,n);
%    x_u = zeros(1,n);
%    for i = 1:nd             % Initial values
%        alphai    = ((3*i - 2*nd)*pi)/(6*nd);
%        x_i(i)      = cos(alphai+pi/12);
%        x_i(i+nd)   = sin(alphai+pi/12);
%        x_i(i+2*nd) = sin(2*alphai+pi/6);
%    end
%    for i = 1:n
%        x_l(i) = -2;          % Lower bounds
%    end
%    for i = 1:n
%        x_u(i) =  2;          % Upper bounds
%    end

%% 4. SVANBERG'S CANTILEVER BEAM PROBLEM INITIALIZATION
%    n = 5;                   % Number of design variables
%    m = 1;                   % Number of constraint functions
%
%    % Initialize design variables x, with row vectors of size n
%    x_i = zeros(n);
%    x_l = zeros(n);
%    x_u = zeros(n);
```

```
%    for i=1:n
%        x_i(i) = 5;           % Initial values
%    end
%    for i = 1:n
%        x_l(i) = 1;           % Lower bounds
%    end
%    for i = 1:n
%        x_u(i) = 10;          % Upper bounds
%    end

%% THE NONLINEAR OPTIMIZATION PROBLEM

%% Identify function which defines non-linear optimization problem
  % opt_func = @fg;      % Function fg.m defines objective and contraint functions
   opt_func = @fg_gyro;
  % opt_func = @fg_snake;
  % opt_func = @fg_cantilever;

%% FILE HANDLING
   file_name = fopen('Gyro_fc1_curv4.txt','w');  %Output file name
   fheading = 'Gyroscope problem: forced convergence - curvature 4';

%% OPTIMIZATION PARAMETERS

%% Maximum number of iterations allowed
   kmax = 1000;         % Maximum number of iterations
   lmax = 1000;         % Maximum iterations for conservative loop
   dualiter = 100000;   % Maximum iterastions for dual subproblem evaluation

%% Define optimization tolerances
   eps_x = 1e-5;        % Convergence tolerance for design variables
   eps1 = 0;            % Appproximation-fo tolerance
   eps2 = 0;            % Appproximation-fc tolerance
   epsno = 1e-6;        % Minimum curvature of objective function
   epsnc = 1e-6;        % Minimum curvature of constraint functions
   dualtol = 1e-9;      % Convergence tolerance for dual variables
   feaslim = 1e-8;      % Printing tolerance for feasibility
   dml_infinity = 0.2;  % Global infinity move limit, 0 < dml < 1
```

```
%% Define upper boudaries of lambda matrix
   lambda_max = 1e8;

%% Define implementation of convergence
   % [force_converge = 0]: Convergence not enforced
   % [force_converge = 1]: Convergence forced by using conservatism
   force_converge = 1;
   xs1 = 2;             % Factor to increase conservatism of fo
   xs2 = 2;             % Factor to increase conservatism of fc

%% Define the approximation curvature components
   % [curv=1]: Spherical quadratic approximation based on function values
   % [curv=2]: Spherical quadratic approximation based on error norm of the gradients
   % [curv=3]: Nonspherical approximation based on the components of the gradients
   % [curv=4]: Quadratic Taylor series expansion to the reciprocal approximation
   % [curv=5]: Quadratic Taylor series expansion to the exponential approximation
   curv = 4;

%-------------------------------------------------------------------%
%                          SAO ALGORITHM                            %
%-------------------------------------------------------------------%

%% INITIALIZATION

k = 0;           % Number of iterations
l = 0;           % Iterations for conservative loop
ltot = 0;        % Total conservative loop iterations

% Initialize design variables x and related vectors
x = x_i;
x_kopt = zeros(1,n);
x_prev = x;
x_u0 = x_u;
x_l0 = x_l;

% Initialize fo, fc, dfo, dfc, fo_c2, fc_c2
[fo,fc,dfo,dfc] = feval(opt_func,n,m,x);
[fo_c2, fc_c2,Ui_prev,Li_prev] = curvinit(n,m,epsno,epsnc,curv,x,x_l,x_u,dfo,dfc);
```

```
% Initialize lambda vectors, lambda_i, lambda_l, lambda_u
lambda_i = zeros(1,m);
lambda_l = zeros(1,m);
lambda_u = ones(1,m);
lambda_u = lambda_max*lambda_u;

fprintf(file_name,'%s \n\n',fheading);
fprintf(file_name,'------------------------------------------------------ \n');
fprintf(file_name,'k   l     fo(x)       x                                 \n');
fprintf(file_name,'------------------------------------------------------ \n');

%% MAIN ALGORITHM
for k = 1:kmax          % start outer loop for k

    % Calculate the move limits
    dml_inf = dml_infinity.*(x_u0 - x_l0);
    x_l = max(x - dml_inf,x_l0);
    x_u = min(x + dml_inf,x_u0);

    % Inner conservatism loop to enforce convergence
    if (force_converge == 1)
        l = -1;
        while (l < lmax)
            l = l+1;

            % For an unconstrained optimization problem, where m = 0
            if (m == 0)

                % Calculate x^{k*}
                x_kopt = x - dfo./fo_c2;
                x_kopt = max(x_l,min(x_kopt,x_u));

                % Compute objective, constraint and approx functions, with x^{k*}
                [fo_kopt,fc_kopt,dfo_kopt,dfc_kopt]= feval(opt_func,n,m,x_kopt);
                [approx_fo,approx_fc] = approxfunc(x_kopt,x,fo,fc,dfo,dfc,fo_c2,..
                ..fc_c2,n,m);

                % Test if x_kopt is acceptable
                % Test if x_kopt represents a feasible descent step
```

```
        if (fo_kopt < fo)
            break;
        end
        % Test if x_kopt represents a conservative step
        if (approx_fo >= (fo_kopt - eps1))
            break;
        end

        % Test if relative step size is too small
        normx = sum((x - x_kopt).^2);
        delta_xnorm = sqrt(normx);
        if (delta_xnorm < eps_x)
            break;
        end

        % Adjust conservatism if x_kopt is not acceptable
        if (approx_fo < (fo_kopt - eps1))
            fo_c2 = xs1.*max(fo_c2,1e-3);
        end

    % For a constrained optimization problem, where m > 0
    else
        % Maximize dual subproblem over lambda, using projbfgs from C.T.
        % Kelley to return lambda^{k*}
        [lambda_kopt,histout,costdata] = projbfgs(lambda_i',@dual,lambda_u',..
        ..lambda_l',dualtol,dualiter,x,x_l,x_u,n,m,fo,fc,dfo,dfc,fo_c2,fc_c2);
        lambda_i = lambda_kopt';

        % Calculate x^{k*} from lambda^{k*}
        x_kopt = xlam(lambda_kopt,x,x_l,x_u,n,m,dfo,dfc,fo_c2,fc_c2);

        % Compute objective, constraint and approx functions, with x^{k*}
        [fo_kopt,fc_kopt,dfo_kopt,dfc_kopt] = feval(opt_func,n,m,x_kopt);
        [approx_fo,approx_fc] = approxfunc(x_kopt,x,fo,fc,dfo,dfc,fo_c2,..
        ..fc_c2,n,m);

        % Test if x_kopt is acceptable:
        % Test if x_kopt represents a feasible descent step
        if ((fo_kopt < fo) && (max(fc_kopt) <= feaslim))
```

```
                break;
            end
            % Test if x_kopt represents a conservative step
            if ((approx_fo >= (fo_kopt - eps1)) & (approx_fc >= (fc_kopt - eps2)))
                break;
            end
            % Test if relative step size is too small
            normx = sum((x-x_kopt).^2);
            delta_xnorm = sqrt(normx);
            if (delta_xnorm < eps_x)
                break;
            end

            % Adjust conservatism if x_kopt is not acceptable
            if (approx_fo < (fo_kopt - eps1))
                %fo_c2 = xs1.*max(fo_c2,1e-3);
                fo_c2 = xs1.*fo_c2;
            end
            for j = 1:m
                if (approx_fc(j) < (fc_kopt(j) - eps2))
                    for i = 1:n
                        %fc_c2(j,i) = xs2*max(fc_c2(j,i),1e-1);
                        fc_c2(j,i) = xs2*fc_c2(j,i);
                    end
                end
            end
        end
    end           % end while loop
end

% No forced convergence
if (force_converge == 0)

    % For an unconstrained optimization problem, where m = 0
    if (m == 0)
        % Calculate x^{k*}
        x_kopt = x - dfo./fo_c2;
        x_kopt = max(x_l,min(x_kopt,x_u));
```

```
        % For a constrained optimization problem, where m > 0
        else
            % Maximize dual subproblem over lambda, using projbfgs from C.T. Kelley
            % to return lambda^{k*}
            [lambda_kopt,histout,costdata] = projbfgs(lambda_i',@dual,lambda_u',..
            ..lambda_l',dualtol,dualiter,x,x_l,x_u,n,m,fo,fc,dfo,dfc,fo_c2,fc_c2);
            lambda_i = lambda_kopt';

            % Calculate x^{k*} from lambda^{k*}
            x_kopt = xlam(lambda_kopt,x,x_l,x_u,n,m,dfo,dfc,fo_c2,fc_c2);
        end
        % Compute objective, constraint and approximated functions, with x^{k*}
        [fo_kopt,fc_kopt,dfo_kopt,dfc_kopt] = feval(opt_func,n,m,x_kopt);
    end

    % Update fo_c2 and fc_c2 for new iteration {k+1}
    [fo_c2, fc_c2,Ui_prev,Li_prev] = approxcurv(n,m,k,epsno,epsnc,curv,x,x_l,x_u,..
     .. fo_kopt,fc_kopt,dfo_kopt,dfc_kopt,x_prev,fo,fc,dfo,dfc,x_kopt,Ui_prev,Li_prev);

    % Update x for new iteration {k+1}
    x_prev = x;
    x = x_kopt;
    [fo,fc,dfo,dfc] = feval(opt_func,n,m,x);

    ltot = ltot + l;    % Total conservative loop iterations

    % Output history to text file
    fprintf(file_name,'%0.0f %3.0f %10.6f',k,l,fo_kopt);
    for ii = 1:n
        fprintf(file_name,'%10.6f',x(ii));
    end
    fprintf(file_name,'\n');

    % Test for convergence
    if (abs(x - x_prev) <= eps_x)
        break;
    end
end              %end outer loop for k
```

```
% Simulation sensitivity analysis
%[fo,fc,dfo,dfc] = feval(opt_func,n,m,x);

%% OUTPUT DATA

%% Output final optimization values to text file
fprintf(file_name,'\n\n');
fprintf(file_name,'          Final optimization parameters\n');
fprintf(file_name,'-------------------------------------------------------- \n');
fprintf(file_name,'Outer loop iterations, k: %0.0f \n',k);
fprintf(file_name,'Inner loop iterations, l: %0.0f \n',ltot);
fprintf(file_name,'Objective function, fo:        %10.6f \n',fo);
fprintf(file_name,'Constraint functions, fc:      ');
for j = 1:m
    fprintf(file_name,'%10.6f',fc(j));
end
fprintf(file_name,'\n');
fprintf(file_name,'Optimal dual variables, lambda:');
for j = 1:m
    fprintf(file_name,'%10.6f',lambda_kopt(j));
end
fprintf(file_name,'\n');
fprintf(file_name,'Optimal design variables, x:   ');
for i = 1:n
    fprintf(file_name,'%10.6f',x(i));
end
fprintf(file_name,'\n');
fprintf(file_name,'-------------------------------------------------------- \n');
fclose(file_name);

profile off;
profile report;
```

Setup in MATLAB: curvinit.m

```
% Authors: S. Vosloo, A.A. Groenwold
% University of Stellenbosch, South Africa
% Date: 05/05/2009
```

```
% Function initializing the approximate curvature components at k = 1
%-----------------------------------------------------------------------%

function [fo_c2, fc_c2,Ui_prev,Li_prev] = curvinit(n,m,epsno,epsnc,curv,x,..
..x_l,x_u,dfo,dfc)

% Initialize approximation curvatures 1,2,3 and 4
fo_c2 = ones(1,n);
fc_c2 = epsnc*ones(m,n);

% Initialize Quadratic Taylor series expansion to reciprocal/exponential approx
if ((curv == 4) || (curv == 5))
    ao = zeros(1,n);
    ac = zeros(m,n);
    for ii = 1:n
        ao(ii) = -1;
        fo_c2(ii) = max(-abs(dfo(ii))*(ao(ii)-1)/x(ii),epsno);
        for jj = 1:m
            ac(jj,ii) = -1;
            fc_c2(jj,ii) = max((-(ac(jj,ii)-1)*abs(dfc(jj,ii))/x(ii)),epsnc);
        end
    end
end

% Quadratic Taylor series expansion to the MMA approximation
if (curv == 6)
    Li = x - (x_u - x_l)./2;
    Ui = x + (x_u - x_l)./2;
    for ii = 1:n
        if (dfo(ii) <= 0)
            fo_c2(ii) = -2*dfo(ii)/(x(ii)-Li(ii));
        else
            fo_c2(ii) =  2*dfo(ii)/(Ui(ii)-x(ii));
        end
    end
    for jj = 1:m
        for ii = 1:n
            if (dfc(jj,ii) <= 0)
                fc_c2(jj,ii) = -2*dfc(jj,ii)/(x(ii)-Li(ii));
```

```
            else
                fc_c2(jj,ii) =  2*dfc(jj,ii)/(Ui(ii)-x(ii));
            end
        end
    end
    fo_c2 = max(fo_c2,epsno);
    fc_c2 = max(fc_c2,epsnc);
    Li_prev = Li;
    Ui_prev = Ui;
else
    Li_prev = 0;
    Ui_prev = 0;
end
```

Setup in MATLAB: approxcurv.m

```
% Authors: S. Vosloo, A.A. Groenwold
% University of Stellenbosch, South Africa
% Date: 05/05/2009
% Function calculating the approximate curvature components
%-------------------------------------------------------------------%

function [fo_c2, fc_c2,Ui_prev,Li_prev] = approxcurv(n,m,k,epsno,epsnc,curv,x,x_l,..
..x_u,fo,fc,dfo,dfc,x_prev,fo_prev,fc_prev,dfo_prev,dfc_prev,x_kopt,Ui_prev,Li_prev)

% Initialization of curve matrices
fo_c2 = zeros(1,n);
fc_c2 = zeros(m,n);

% Tolerances
xtol = 1e-6;
zerotol = 1e-10;
a = 1e-3;
a_lo = -4;
a_hi = -0.1;
mma_hi = 1.2;
mma_lo = 0.7;
```

```
% Spherical quadratic approximation based on function values
if (curv == 1)
    fc_c2temp = zeros(m,1);
    delta_x = zeros(1,n);
    norm = 0;
    nab_fo = 0;
    for ii = 1:n
        delta_x(ii) = x(ii) - x_kopt(ii);
        norm = norm + delta_x(ii)^2;
        nab_fo = nab_fo + dfo(ii)*delta_x(ii);
    end
    norm = max(norm,zerotol);
    fo_c2temp = 2*(fo_prev - fo - nab_fo)/norm;
    for jj = 1:m
        nab_fc = 0;
        for ii = 1:n
            nab_fc = nab_fc + dfc(jj,ii)*delta_x(ii);
        end
        fc_c2temp(jj) = 2*(fc_prev(jj) - fc(jj) - nab_fc)/norm;
    end
    for ii = 1:n
        fo_c2(ii) = max(fo_c2temp,epsno);
        for jj = 1:m
            fc_c2(jj,ii) = max(fc_c2temp(jj),epsnc);
        end
    end
end

% Spherical quadratic approximation based on error norm of the gradients
if (curv == 2)
    fc_c2temp = zeros(m,1);
    delta_x = zeros(1,n);
    norm = 0;
    nab_fo = 0;
    for ii = 1:n
        delta_x(ii) = x(ii)-x_kopt(ii);
        norm = norm + delta_x(ii)^2;
        nab_fo = nab_fo + (dfo_prev(ii) - dfo(ii))*delta_x(ii);
    end
```

```
    norm = max(norm,zerotol);
    fo_c2temp = nab_fo/norm;
    for jj = 1:m
        nab_fc = 0;
        for ii = 1:n
            nab_fc = nab_fc + (dfc_prev(jj,ii) - dfc(jj,ii))*delta_x(ii);
        end
        fc_c2temp(jj) = nab_fc/norm;
    end
    for ii = 1:n
        fo_c2(ii) = max(fo_c2temp,epsno);
        for jj = 1:m
            fc_c2(jj,ii) = max(fc_c2temp(jj),epsnc);
        end
    end
end

% Nonspherical approximation based on the components of the gradients
if (curv == 3)
    delta_x = zeros(1,n);
    for ii = 1:n
        delta_x(ii) = x(ii) - x_kopt(ii);
        if (abs(delta_x(ii)) > xtol)
            fo_c2(ii) = max(((dfo_prev(ii) - dfo(ii))/delta_x(ii)),epsno);
        else
            fo_c2(ii) = a;
        end
        for jj = 1:m
            if (abs(delta_x(ii)) > xtol)
                fc_c2(jj,ii) = max(((dfc_prev(jj,ii) - dfc(jj,ii))/delta_x(ii)),epsnc);
            else
                fc_c2(jj,ii) = epsnc;
            end
        end
    end
end

% Quadratic Taylor series expansion to the reciprocal approximation
if (curv == 4)
```

```
    for ii = 1:n
        fo_c2(ii) = max(abs(dfo(ii))*2/x_kopt(ii),epsno);
        for jj = 1:m
            fc_c2(jj,ii) = max(abs(dfc(jj,ii))*2/x_kopt(ii),epsnc);
        end
    end
end

% Quadratic Taylor series expansion to the exponential approximation
if (curv == 5)
    ao = zeros(1,n);
    ac = zeros(m,n);
    for ii = 1:n
        if ((x(ii)/x_kopt(ii) ~= 1) && (dfo(ii)~= 0) && (dfo_prev(ii)~= 0))
            ao(ii) = 1+(log(dfo_prev(ii)/dfo(ii)))/(log(x(ii)/x_kopt(ii)));
        else
            ao(ii) = -1;
        end
        ao(ii) = max(a_lo,min(a_hi,ao(ii)));
        ao(ii) = min(ao(ii),-1e-4);
        fo_c2(ii) = -abs(dfo(ii))*(ao(ii)-1)/x_kopt(ii);
        fo_c2(ii) = max(fo_c2(ii),epsno);
        for jj = 1:m
            if ((x(ii)/x_kopt(ii) ~= 1) && (dfc(jj,ii)~= 0) && (dfc_prev(jj,ii)~= 0))
                ac(jj,ii) = 1+(log(dfc_prev(jj,ii)/dfc(jj,ii)))/(log(x(ii)/x_kopt(ii)));
            else
                ac(jj,ii) = -1;
            end
            ac(jj,ii) = max(a_lo,min(a_hi,ac(jj,ii)));
            ac(jj,ii) = min(0,ac(jj,ii));
            fc_c2(jj,ii) = -(ac(jj,ii)-1)*abs(dfc(jj,ii))/x_kopt(ii);
            fc_c2(jj,ii) = max(fc_c2(jj,ii),epsnc);
        end
    end
end

% Quadratic Taylor series expansion to the MMA approximation
if (curv == 6)
    gam = zeros(1,n);
```

```
if (k <= 2)
    Li = x_kopt - (x_u - x_l)./2;
    Ui = x_kopt + (x_u - x_l)./2;
    Uio = Ui;
    Lio = Li;
    Uic = Ui;
    Lic = Li;
    Uic1 = Ui;
    Lic1 = Li;
    Li_prev = Li;
    Ui_prev = Ui;
else
   test = (x_kopt - x).*(x - x_prev);
   for ii = 1:n
        if (test(ii) < 0)
            gam(ii) = mma_lo;
        elseif (test(ii) > 0)
            gam(ii) = mma_hi;
        else
            gam(ii) = 1;
        end
   end
   Lio = x_kopt - gam.*(x - Li_prev);
   Uio = x_kopt + gam.*(Ui_prev - x);
   Li_prev = Lio;
   Ui_prev = Uio;
   Lic1 = x_kopt - gam.*(x - Li_prev);
   Uic1 = x_kopt + gam.*(Ui_prev - x);
   Li_prev = Lic1;
   Ui_prev = Uic1;
   Lic = x_kopt - gam.*(x - Li_prev);
   Uic = x_kopt + gam.*(Ui_prev - x);
end
for ii = 1:n
    if (dfo(ii) <= 0)
        fo_c2(ii) = -2*dfo(ii)/(x_kopt(ii) - Lio(ii));
    else
        fo_c2(ii) =  2*dfo(ii)/(Uio(ii) - x_kopt(ii));
    end
```

```
    end
    for ii = 1:n
        if (dfc(1,ii) <= 0)
            fc_c2(1,ii) = -2*dfc(1,ii)/(x_kopt(ii) - Lic1(ii));
        else
            fc_c2(1,ii) =  2*dfc(1,ii)/(Uic1(ii) - x_kopt(ii));
        end
    end
    for jj = 2:m
        for ii = 1:n
            if (dfc(jj,ii) <= 0)
                fc_c2(jj,ii) = -2*dfc(jj,ii)/(x_kopt(ii) - Lic(ii));
            else
                fc_c2(jj,ii) =  2*dfc(jj,ii)/(Uic(ii) - x_kopt(ii));
            end
        end
    end
    fo_c2 = max(fo_c2,epsno);
    fc_c2 = max(fc_c2,epsnc);
else
    Li_prev = 0;
    Ui_prev = 0;
end
```

Setup in MATLAB: approxfunc.m

```
% Authors: S. Vosloo, A.A. Groenwold
% University of Stellenbosch, South Africa
% Date: 05/05/2009
% Function calculating the approximate functions
%-------------------------------------------------------------------%

function [approx_fo, approx_fc] = approxfunc(x1,x2,fo,fc,dfo,dfc,fo_c2,fc_c2,n,m)

delta_x = x1 - x2;
delta_x2 = delta_x.^2;

% % Approximate objective function
```

```
sum0 = dfo.*delta_x;
sum1 = sum(sum0);
sum0 = fo_c2.*delta_x2;
sum2 = sum(sum0);
approx_fo = fo + sum1 + 0.5*sum2;

% % Approximate constraint functions
sum1 = delta_x*dfc';
sum2 = delta_x2*fc_c2';
approx_fc = fc + sum1 + 0.5*sum2;

% % Approximate objective function
% sum1 = 0;
% sum2 = 0;
% for ii = 1:n
%     sum1 = sum1 + dfo(ii)*(x1(ii) - x2(ii));
%     sum2 = sum2 + fo_c2(ii)*(x1(ii) - x2(ii))^2 ;
% end
% approx_fo = fo + sum1 + 0.5*sum2;

% % Approximate constraint functions
for jj = 1:m
    sum1 = 0;
    sum2 = 0;
    for ii = 1:n
        sum1 = sum1 + dfc(jj,ii)*(x1(ii) - x2(ii));
        sum2 = sum2 + fc_c2(jj,ii)*(x1(ii) - x2(ii))^2;
    end
    approx_fc(jj) = fc(jj) + sum1 + (sum2/2);
end
```

Setup in MATLAB: dual.m

```
% Authors: S. Vosloo, A.A. Groenwold
% University of Stellenbosch, South Africa
% Date: 05/05/2009
% Function calculating the dual subproblem
%-------------------------------------------------------------------%
```

```
function [gamma,dgamma] = dual(lambda,x,x_l,x_u,n,m,fo,fc,dfo,dfc,fo_c2,fc_c2)

  % Calculate x ito lambda
  x_lambda = xlam(lambda,x,x_l,x_u,n,m,dfo,dfc,fo_c2,fc_c2);

  % Calculate approximations
  [approx_fo, approx_fc] = approxfunc(x_lambda,x,fo,fc,dfo,dfc,fo_c2,fc_c2,n,m);

  % Find gamma and dgamma ito lambda
  summ = approx_fc.*lambda';
  summ = sum(summ);
  gamma = approx_fo + summ;
  gamma = -gamma;
  dgamma = -approx_fc';

  % Find gamma and dgamma ito lambda
  %   summ = 0;
  %   for jj = 1:m
  %      summ = summ + approx_fc(jj)*lambda(jj);
  %   end
  %   gamma = approx_fo + summ;
  %   gamma = -gamma;
  %   dgamma = -approx_fc';
```

Setup in MATLAB: xlam.m

```
% Authors: S. Vosloo, A.A. Groenwold
% University of Stellenbosch, South Africa
% Date: 05/05/2009
% Function to calculate x in terms of lambda
%-------------------------------------------------------------------%

function x_lam = xlam(lam,x,x_l,x_u,n,m,dfo,dfc,fo_c2,fc_c2)

% Calculate xlam vector
sum1 = lam'*fc_c2;
sum2 = lam'*dfc;
```

```
beta = x - (dfo + sum2)./(fo_c2 + sum1);
x_lam = max(x_l,min(x_u,beta));

% Calculate xlam vector
% for ii = 1:n
%     sum1(ii) = 0;
%     sum2(ii) = 0;
%     for jj = 1:m
%         sum1(ii) = sum1(ii) + lam(jj)*fc_c2(jj,ii);
%         sum2(ii) = sum2(ii) + lam(jj)*dfc(jj,ii);
%     end
%      beta(ii) = x(ii) - ((dfo(ii) + sum2)/( fo_c2(ii) + sum1 ));
%      x_lam(ii) = max(x_l(ii),min(x_u(ii),beta(ii)));
% end
```