Institut National de Recherche en Informatique et en Automatique

Université de Nice-Sophia Antipolis

Multi-objective Optimization Problem, Concurrent Engineering

By

Haris Malik

A REPORT SUBMITTED ON THE STAGE
IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE
IN MATHEMATICAL MODELLING

Research Supervisor's Name:

Mr. Jean-Antoine Désidéri, Director of Research, HDR Head, Project-Team OPALE (INRIA)

Research Supervisor's Signature: _____

1. Prelude to the Stage:

The stage is performed at the French National Institute for Research in Computer Science and Control (Institut National de Recherche en Informatique et en Automatique-INRIA), under the supervision of Mr. Jean-Antoine Désidéri, Director of Research, HDR Head, Project-Team Opale (Optimization and Control, Numerical Algorithms and Integration of Complex Multi-Discipline Systems governed by Partial Differential Equations).

The stage lasted for almost three months starting in May, 2009 until July, 2009. During the tenure at INRIA, the intern mainly worked in the field of numerical approximation of the optimization problem using the freeware software Scilab.

2. Literature Overview

First a brief overview of the literature is presented.

2.1 Historical Use of Optimization in Aerospace

Optimization is the choice of best element from a set of alternatives available to maximize or minimize a real function. The history of optimization dates back to the first known optimization technique of Steepest Descent pioneered by Gauss. With the advent of last century the available techniques are more refined and now find themselves being employed in a multitude of scientific and technological fields.

Aviation, from the beginning is enjoying the benefits of optimization techniques. Early aerospace designs were characterized not only by a great deal of trial and error but also with a considerable amount of analysis. It is also clear the early pioneers had clear goals they wished to meet. Although, these were simply measures of distances covered. As the technologies used in aerospace applications have developed, the goals have become vastly more sophisticated.

2.2 Multiple-objective Optimization

Multiple-Objective optimization problems refer to the optimization problem in simultaneously two or more conflicting/contradicting objectives are tried to be minimized which may be constrained under certain conditions. [1]

Most realistic optimization problems, particularly those in design, require the simultaneous optimization of more than one objective function. Some examples:

- In bridge construction, a good design is characterized by low total mass and high stiffness.
- Aircraft design requires simultaneous optimization of fuel efficiency, payload, and weight.
- In chemical plant design, or in design of a groundwater remediation facility, objectives to be considered include total investment and net operating costs.
- A good sunroof design in a car could aim to minimize the noise the driver hears and maximize the ventilation.
- The traditional portfolio optimization problem attempts to simultaneously minimize the risk and maximize the fiscal return.

Generally there is not a single solution to multi-objective optimization problems that simultaneously optimizes all the criteria to its best. In each case, we look for a trade-off that all the criteria are sufficiently optimized and if further trying to improve on one criterion the other(s) will suffer.

Multi-objective optimization has its root in late nineteenth century welfare economics, in the works of Edgeworth and Pareto. A mathematical description is as follows:

$$\min_{x \in C} F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_{n-1}(x) \\ f_n(x) \end{bmatrix}$$

Where n≥ 2 and

$$C = \{x : h(x) = 0; g(x) \leq 0, a \leq x \leq b\}$$

Denotes the feasible set constrained by equality and inequality constraints and explicit variable bounds. The space in which the objective vector belongs is called the "objective space" and the image of the feasible set under F is called the "attained set".

Multi-objective optimization can be further classified depending upon the type of objectives involved. A brief overview is given here:

- Multi-criterion Optimization

- Multi-point Optimization
- Multi-discipline Optimization

In our problem, we will deal with multi-discipline optimization problem which will consider the objectives to be based on aerodynamics and structures mainly.

2.2.1 Multi-Disciplinary Optimization

Multi-disciplinary optimization (MDO) is a field of engineering that uses optimization methods to solve design problems incorporating a number of disciplines. MDO allows designers to incorporate all relevant disciplines simultaneously. The optimum of the simultaneous problem is superior to the design found by optimizing each discipline sequentially, since it can exploit the interactions between the disciplines. However, including all disciplines simultaneously significantly increases the complexity of the problem.

These techniques have been used in a number of fields, including automobile design, naval architecture, electronics, computers and electricity distribution. But, the greatest number of applications has been in the field of Aerospace engineering. For example, the proposed Boeing blended wing body (BWB) aircraft concept has used MDO extensively in the conceptual and preliminary design stages. These disciplines range from aerodynamics, structural analysis, propulsion, control systems etc.

An important task in multi-objective optimization is to find the Pareto-optimal solutions. The scalar concept of "optimality" does not apply directly in the multi-objective setting. Pareto optimality serves as a useful representation. Their knowledge allows a decision maker to learn more about the trade-offs among the different objectives.

2.3 Pareto Optimality

It is an important concept in economics with broad applications in other fields as engineering, game theory and social sciences. Named after Vilfredo Pareto, an Italian economist, Pareto optimality is a measure of efficiency. An outcome of a game is Pareto optimal if there is no other outcome that makes every player at least as well off and at least one player strictly better off. That is, a Pareto Optimal solution cannot be further improved without hurting at least one player.

A typical definition of Pareto efficiency would be: "A given arrangement is Pareto efficient if there can be no arrangement where change to make one player/person better off will not leave other's position worsened." Mathematically, we can give the following definition:

"A vector of decision variables $x^* \epsilon C$ is Pareto optimal if there does not exist another $x \epsilon C$ such that $f_i(x) \leq f_i(x^*)$ for all i = 1,…, n and $f_j(x) < f_j(x^*)$ for at least one j."

This concept almost always does not give a single solution, but rather a set of solutions called the "Pareto Optimal Set". The vectors x^* corresponding to the solutions included in the Pareto Optimal set are called "non-dominated".

2.3.1 Notion of Dominance/Non-dominance

Let Y ∈ $\mathbb{R}^N$ denotes the vector of design variables. If several minimization problems are to be considered concurrently, a design point Y^1 is said to dominate the design point Y^2, symbolically,

$$Y^1 \succ Y^2$$

iff, for all the criteria to be minimized J = J_A, J_B, …

$$J(Y^1) \leq J(Y^2)$$

and at least one of the inequalities is strict.

Otherwise, the vectors are said to be non-dominated, that is what we need for Pareto Optimal set, i.e.

$$Y^1 \not\succ Y^2, Y^2 \not\succ Y^1$$

2.3.2 Pareto Front

The plot of the objective functions whose non-dominated vectors are in the Pareto optimal set is called the "Pareto Front" [3]. A Pareto front is the set of choices that are Pareto efficient. The Pareto front is particularly useful in engineering; it's a convenient way of considering only Pareto efficient alternatives, rather than considering the full range of every parameter.

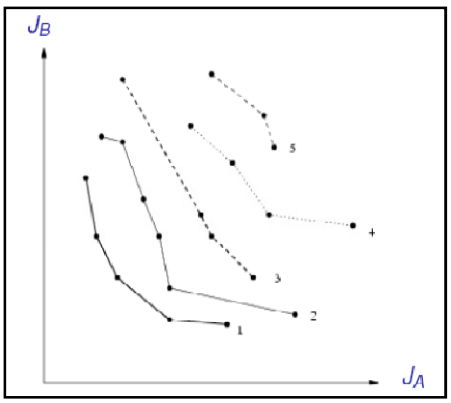

Figure 2-1: Pareto front [3]

2.4 Problem Scenario

We are looking for a multi-objective optimization problem related to aeronautics; here we are concerned with multi-disciplinary optimization with aerodynamic design versus structural design and possibly acoustics, thermal loads etc. We tend to develop a numerical algorithm of a three objective unconstrained minimization problem. The problem under consideration falls into the category of multi-disciplinary optimization problem.

We consider an optimization problem with three objective functions denoted by J_1, J_2 and J_3. Each criterion is considered to be a smooth function of a common design vector $Y \in \mathbb{R}^4$. We aim to find Y^* such that it is the Pareto optimal solution of all the criteria.

2.4.1 Problem Statement

Given a design vector;

$$Y \in \mathbb{R}^N$$

Minimize the given criteria denoted by;

$$J_i(Y) \quad i = 1, 2, \dots, n$$

Where

$$N \geq n$$

Here, we consier N = 4 and n = 3.

The gradients of the criteria are denoted by;

$$u_i = \nabla J_i(Y)$$

Before we continue with the description of our solution and the numerical approach used in order to optimize our given objectives, it is important to briefly discuss some concepts.

Most significantly the number of criterion 'n' considered should be less than the dimension 'N' of the space of design vector. The space of design vector can be a Hilbert Space usually equal to $\mathbb{R}^N$, but it can also be a subspace of L^2.

2.4.2 Pareto Concepts

Here, we will briefly state the important theorems and lemmas and will not go into the details and the proofs of any of them. These concepts are taken from the report by Mr. Jeane-Antoine Désidéri and the reports are listed in the reference section [2]. For proofs and details, these reports maybe consulted. For a further detailed study of the concepts presented here, we refer a text book by K. Miettinen in non-linear multi-objective optimization.

We consider n smooth criteria $J_i(Y)$, where Y is the design vector; $Y \in \mathcal{H}$; $\mathcal{H}$: working space, a Hilbert space equal to $\mathbb{R}^N$, but it can also be a subspace of L^2. The functions or functional are assumed to be of class C^2 in some working open ball of the design space $\mathcal{H}$.

Lemma 1

[2] Let Y^0 be a Pareto optimal point of the smooth criteria $J_i(Y)\ (1 \le i \le n \le N)$, and define the gradient vectors $u_i^0 = \nabla J_i(Y^0)$ in which ∇ denotes the gradient operator. There exists a convex combination of the gradient vectors that is equal to zero: [2]

$$\sum_{i=1}^{n} \alpha_i u_i^0 = 0, \quad \alpha_i \ge 0, \quad \sum_{i=1}^{n} \alpha_i = 1$$

Definition 1: Pareto Stationary

[2] The smooth criteria $J_i(Y)\ (1 \le i \le n \le N)$ are said to be Pareto stationary at the design point Y^0 iff there exists a convex combination of the gradient vectors, $u_i^0 = \nabla J_i(Y^0)$, that is equal to zero:

$$\sum_{i=1}^{n} \alpha_i u_i^0 = 0, \quad \alpha_i \ge 0, \quad \sum_{i=1}^{n} \alpha_i = 1$$

Thus in general, for smooth unconstrained criteria, Pareto stationarity is a necessary condition for Pareto optimality.

Inversely, if the smooth criteria $J_i(Y)\,(1 \le i \le n \le N)$ are not Pareto stationary at a given design point Y^0, descent directions common to all criteria exist. Next, we will devise a method to determine this descent direction.

Lemma 2

[2] Let $\mathcal{H}$ be a Hilbert space of infinite or finite dimension N, and $\{u_i\}\,(1 \le i \le n \le N)$ a family of n vectors in $\mathcal{H}$. Let $\mathcal{V}$ be the set of strict convex combinations of these vectors:

$$\mathcal{U} = \left\{ \omega \,\epsilon\, \mathcal{H} \,\middle/\, \omega = \sum_{i=1}^{n} \alpha_i u_i \,;\, \alpha_i > 0 \,;\, \sum_{i=1}^{n} \alpha_i = 1 \right\}$$

and $\overline{\mathcal{U}}$ its closure (the convex hull of the family). Then, there exists a unique element $\omega\,\epsilon\,\overline{\mathcal{U}}$ of minimum norm, and:

$$\forall\, \overline{u}\,\epsilon\,\overline{\mathcal{U}} : (\overline{u}, \omega) \ge (\omega, \omega) = \|\omega\|^2 := C_\omega$$

Combining Lemma 2 with Definition 1 yields the following:

Theorem 1

Let $\mathcal{H}$ be a Hilbert space of infinite or finite dimension N. Let $J_i(Y)\,(1 \le i \le n \le N)$ be n smooth functions of the vector Y ∈ $\mathcal{H}$, and Y^0 a particular admissible design point, at which the gradient vectors are denoted by $u_i^0 = \nabla J_i(Y^0)$, and [2]

$$\mathcal{U} = \left\{ \omega \,\epsilon\, \mathcal{H} \,\middle/\, \omega = \sum_{i=1}^{n} \alpha_i u_i \,;\, \alpha_i > 0 \,;\, \sum_{i=1}^{n} \alpha_i = 1 \right\}$$

Let ω be the minimal norm element of the convex hull $\overline{\mathcal{U}}$, closure of $\mathcal{U}$. Then:

1. Either ω = 0, and the criteria $J_i(Y)\,(1 \le i \le n \le N)$ are Pareto stationary at Y = Y^0 [2].
2. Or ω ≠ 0 and − ω is a descent direction common to all the criteria; additionally, if ω∈$\mathcal{U}$, the inner product $(\overline{u}, \omega)$ is equal to $\|\omega\|^2$ for all $\overline{u}\,\epsilon\,\overline{\mathcal{U}}$ [2].

As a result to the above described theorem, one is led to identify the vector

$$\omega = \sum_{i=1}^{n} \alpha_i u_i^0$$

by solving the following quadratic form constrained minimization problem in $\mathbb{R}^N$:

$$\min_{\alpha \, \epsilon \, \mathbb{R}^n} \left\| \sum_{i=1}^{n} \alpha_i u_i^0 \right\|^2$$

subject to following constraints:

$$\alpha_i \geq 0 \; (\forall i) \; ; \; \sum_{i=1}^{n} \alpha_i = 1$$

Note that in a finite dimensional setting, and in a functional space setting as well, the above problem can be solved in $\mathbb{R}^n$, so long as the gradients $\{u_i\} \, (1 \leq i \leq n \leq N)$ and their inner products $\{u_{ij}^0 \coloneqq (u_i^0, u_j^0)\}$ are known.

Next we will discuss the possible routines to solve the above given minimization problem. There are a few library routines available with Scilab for optimization namely Optim and Quapro. In order to solve the minimization problem, these routines were tried but neither one of them was able to solve the problem. In the following sections we will discuss the issues that led to the inability of the library routines provided with Scilab to be implemented for our case.

2.4.3 Scilab Routine:Optim

[4], [5] Optim is a utility to solve non-linear optimization problem in Scilab. The simplest call to function optim is given as

[fopt, xopt] = optim(costf,x0)

Where xopt is the value of the design variables vector x that minimizes function costf. The value of the function at x = xopt is given by fopt. An initial guess to the solution, x0, is

provided as argument of the function. The function costf must be defined so that the general call to this function

[f,g,ind] = costf(x,ind)

This function is able to return the value of function f and its gradient g depending upon the value of ind given to it.

The biggest advantage of this routine is that it can be used for any non-linear function. In addition, optim can use a third argument specifying the algorithm used for the solution that can be Quasi-Newton method, Conjugate Gradient method and Non-differentiable method. Function optim allows for constraints to be places on the design variables in the form of a lower bound and an upper bound. These bounds can be represented in the form of arrays where the values each individual component of our design variable can attain must be specified. [6]

In our problem, we are not usually restricted to any kind of algorithm so that option of optim was left alone but we had constraints on our variable in the form of

$$\alpha_i \geq 0 \ (\forall i) \ ; \ \sum_{i=1}^{n} \alpha_i = 1$$

Here, we notice that we only had specific lower bounds but the upper bound is not individual so we cannot simply use an upper bound as the upper bound is in form of a sum of all the components of the design variable.

This limitation of optim was crucial in our minimization problem and we have to look for another library routine.

2.4.4 Scilab Routine: Quapro/Qpsolve (Quadratic Programming)

[4] Quadratic programming is an optimization method applied to the solution of problems in which both the objective function is defined, in general, by an objective function consisting of a quadratic form plus a linear combination of the design variables. The constraint

functions of the problem are still linear functions of the design variables. The problem may be expressed as

$$minimize\ f(x) = {}^{1}/_{2}\, x^T Q x + c^T x$$

Subject to

$$Ax = b$$

$$Gx \leq h$$

And

$$x_L \leq x \leq x_U$$

Where x is the design vector with n variables, Q is an n X n symmetric square matrix, c is a column vector of n constant coefficients, and A, b, G and h can have different dimensions depending upon the number of equality and inequality constraints. x_L and x_U are the vectors representing lower and upper bounds respectively with n variables.

Scilab has a built-in function for solving quadratic programming problems; the function is given by quapro. This function is superseded by another built-in function qpsolve in the recent versions of Scilab. This function can be called using the following syntax:

[x [,iact [,iter [,f]]]]=qpsolve(Q,p,C,b,ci,cs,me)

Q	real positive definite symmetric matrix (dimension n x n).
p	real (column) vector (dimension n)
C	real matrix (dimension (me + md) x n). This matrix may be dense or sparse.
b	RHS column vector (dimension m=(me + md))
ci	column vector of lower-bounds (dimension n).
cs	column vector of upper-bounds. (Same remarks as above).
me	number of equality constraints (i.e. C(1:me,:)*x = b(1:me))
x	optimal solution found.

The condition on Q being definite positive matrix makes it useless for our problem, as we cannot guarantee the positive definitivity of Q since it is a combination of a number of gradients and can be negative in some cases.

There are some libraries written in C++ and in some other languages that can be used with Scilab such as OPT++ [8], but the difficulty and lack of expertise to incorporate libraries

written in other programming languages and the inability to use any of the built-in libraries of Scilab drives us to write our own algorithm in Scilab for the multiple gradient descent direction.

2.4.5 Algorithm

After trying out different library functions from Scilab and not succeeding because of limitations that are already being defined in the previous sections. It was decided that an algorithm will be devised and written in Scilab for the minimizing of our problem which will give us the descent direction common to all the criteria. The minimization problem is the following, given:

$$\omega = \sum_{i=1}^{n} \alpha_i u_i^0$$

Minimize the quadratic form

$$\min_{\alpha \in \mathbb{R}^n} \left\| \sum_{i=1}^{n} \alpha_i u_i^0 \right\|^2$$

subject to following constraints:

$$\alpha_i \geq 0 \ (\forall i) \ ; \ \sum_{i=1}^{n} \alpha_i = 1$$

We start with an initial guess on $\alpha^0 = \left[\frac{1}{n}, \frac{1}{n}, \ldots, \frac{1}{n}\right]$

In our case, we assume n = 3 and N = 4, so that $\alpha^0 = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right]$. We define our criteria J_1, J_2 and J_3 and their respective derivatives as:

$$J_i(Y^0) \ (1 \leq i \leq n \leq N)$$
$$u_i^0 = \nabla J_i(Y^0)$$

where Y^0 is an initial point that can be determined by design of experiment. We will briefly look into this notation of Design of experiment before moving further.

Design of Experiment

Design of experiments, or experimental design, is the design of all information-gathering exercises where variation is present, whether under the full control of the experimenter or not. The purpose of it is to study the effect of some processes or intervention on some objects. Design of experiment is a discipline which has broad applications across all the natural and social sciences.

A methodology for designing experiments was proposed by Ronald A. Fisher, in his innovative book The Design of Experiments (1935). A detailed study of the topic is out of scope of this report.

After selecting one of the vectors from the set resulting from the design of experiments, the emphasis is on finding the minimum norm ω. There can be two conditions:

1. ω = 0 at which we say that it is already a Pareto stationary at Y = Y^0 and we stop [2]. In programming this condition can be satisfied by using an appropriate value of tolerance and

$$\|\omega\| < tol \qquad stop$$

2. The second case can be when ω ≠ 0 and – ω is the descent direction [2]. In this case, if

$$\|\omega\| > tol$$

We will follow the following algorithm.

Once, we find our initial ω ≠ 0, we redefine ω as

$$\omega = \sum_{i=1}^{n-1} \alpha_i u_i^0$$

And define the minimization of

$$q = \min_{\alpha \in \mathbb{R}^{n-1}} \|\omega\|^2 = \min_{\alpha \in \mathbb{R}^{n-1}} \left\| \sum_{i=1}^{n-1} \alpha_i u_i^0 \right\|^2$$

And the constraints are modified in order to account for the nth α, i.e. α_n. The constraints take the following new form:

$$\alpha_i \geq 0 \ (\forall i) \ ; \ \alpha_n = 1 - \sum_{i=1}^{n-1} \alpha_i$$

Next, we define the partial derivatives of our newly defined q with respect to α_i, this can be represented as:

$$\frac{\partial q}{\partial \alpha_i} = 2(\omega, u_i^0 - u_n^0) \qquad \forall \ i = 1,2, \dots n-1$$

In the above equation $u_i^0 - u_n^0$ is used to incorporate the effect of α_n. Once, the evaluation of partial derivatives of q is done, we use the following expression:

$$\alpha_i' = \alpha_i^0 - \rho \left(\frac{\partial q}{\partial \alpha_i}\right)_{\alpha_i^0} \qquad \forall \ i = 1,2, \dots n-1$$

And set $\alpha_i' = 0$ or $\alpha_i' = 1$, depending upon the condition that is if:

$$\frac{\partial q}{\partial \alpha_i} \geq 0$$

$$\Rightarrow \rho_i \leq \alpha_i^0 \Big/ \frac{\partial q}{\partial \alpha_i}$$

And if

$$\frac{\partial q}{\partial \alpha_i} < 0$$

$$\Rightarrow \rho_i \leq 1 - \alpha_i^0 \Big/ \left|\frac{\partial q}{\partial \alpha_i}\right|$$

So, depending upon the value of $\frac{\partial q}{\partial \alpha_i}$, we have the following condition on ρ and we choose the maximum value of ρ as:

$$\rho_{i,max} := \begin{cases} \alpha_i^0 \Big/ \frac{\partial q}{\partial \alpha_i} & \frac{\partial q}{\partial \alpha_i} > 0 \\ 1 - \alpha_i^0 \Big/ \left|\frac{\partial q}{\partial \alpha_i}\right| & \frac{\partial q}{\partial \alpha_i} < 0 \end{cases}$$

Now, for α_n we make use of the condition that we defined earlier

$$\alpha_n = 1 - \sum_{i=1}^{n-1} \alpha_i$$

$$\sum_{i=1}^{n-1} \alpha_i' = \sum_{i=1}^{n-1} \alpha_i^0 - \rho \sum_{i=1}^{n-1} \frac{\partial q}{\partial \alpha_i}$$

$$1 - \alpha_n' = 1 - \alpha_n^0 - \rho Q', \qquad Q' := \sum_{i=1}^{n-1} \frac{\partial q}{\partial \alpha_i}$$

$$\alpha_n' = \alpha_n^0 + \rho Q'$$

Again we have similar conditions on ρ as before;

$$\rho_{n,max} := \begin{cases} {}^{1-\alpha_n^0}/_{Q'} & Q' > 0 \\ {}^{\alpha_n^0}/_{|Q'|} & Q' < 0 \end{cases}$$

Combined, these two gives us a condition on the choice of ρ. We will now consider the minimum of ρ in the set.

$$\rho_{max} = \min_{i=1,\dots,n} \rho_{i,max}$$

We then discretize this ρ_{max} into a number of intervals using the equations given as:

$$\rho_k = \frac{\rho_{max}}{k} \qquad k = 1, \dots, m$$

Where m is some positive integer and its choice depends upon the conditions.

$$\alpha_i' = \alpha_i^0 - \rho_k \left(\frac{\partial q}{\partial \alpha_i}\right)_{\alpha_i^0} \qquad \forall\, i = 1,2, \dots n-1$$

$$\alpha_n' = \alpha_n^0 + \rho_k Q' \qquad \forall\, i = n$$

Now, this new α takes place of α^0 and we again calculate ω for every ρ_k until it reaches ρ_{max} or the value of ω starts to increase. Here, it should be kept in mind that we are looking that none of the criteria should increase, so as soon as one of them increases we stop. At this

point, we again repeat the steps and continue until the convergence in the value of ω is achieved.

At this step, we get out of the loop for calculation of ω and compute an interval using the following equations:

$$\Delta t_{i,max} = \frac{J_i}{u_i . \omega}$$

Then, we take the maximum of these values and discretize the interval [0, t_{max}] into a number of steps.

$$t_{max} = \max_{i=1,..,n} \Delta t_{i,max}$$

Then, reset the initial vector of design variables to

$$Y^0 = Y^0 - t\omega$$

And then calculate

$$j_i(t) = J_i(Y^0 - t\omega) \ (1 \leq i \leq n)$$

The step size should be such that it is the largest strictly positive real number for which all the functions $j_i(t) = J_i(Y^0 - t\omega) \ (1 \leq i \leq n)$ are monotone-decreasing over the interval [0,t_{max}]. We stop when any of the functions start increasing.

At this, we have a new design point $Y^0 = Y^0 - t\omega$ and we again start with the initial steps with this new Y^0 and continue doing until we satisfy the tolerance condition on ω. We continue the procedure until we find the value of ω less than tolerance.

$$\|\omega\| < tol \qquad stop$$

If the above condition is satisfied, that means that we have found one point on our Pareto front and this point is a part of Pareto optimal set.

We continue this procedure for all the points from our set of design of experiments.

Now, we will present the example that we used for our problem and show the results from the run of our code on it.

2.4.6 Example and Solution

We used three criteria J_1, J_2 and J_3 as our examples [8] and have defined them as follows:

$$J_1(y) = 2\left((2+\sqrt{2})y_1^2 + \sqrt{2}y_3^2 + y_4^2\right)$$

$$J_2(y) = 3\left(\frac{5}{3y_1^2} + \frac{3}{2y_2^2} + \frac{2}{y_3^2} + \frac{2}{y_4^2}\right)$$

$$J_3(y) = \frac{1}{y_1^2} + \frac{2}{y_2^2} + \frac{2}{y_3^2} + \frac{1}{4y_4^2}$$

The set of design variables was chosen to be comprised of 11 points and as follows:

$$Y = \left\{\begin{bmatrix}0.9\\0.9\\0.9\\0.9\end{bmatrix}, \begin{bmatrix}1\\1\\1\\1\end{bmatrix}, \begin{bmatrix}1.1\\1.1\\1.1\\1.1\end{bmatrix}, \begin{bmatrix}1.2\\1.2\\1.2\\1.2\end{bmatrix}, \begin{bmatrix}1.3\\1.3\\1.3\\1.3\end{bmatrix}, \begin{bmatrix}1.4\\1.4\\1.4\\1.4\end{bmatrix}, \begin{bmatrix}1.6\\1.6\\1.6\\1.6\end{bmatrix}, \begin{bmatrix}1.7\\1.7\\1.7\\1.7\end{bmatrix}, \begin{bmatrix}1.8\\1.8\\1.8\\1.8\end{bmatrix}, \begin{bmatrix}1.9\\1.9\\1.9\\1.9\end{bmatrix}, \begin{bmatrix}2.0\\2.0\\2.0\\2.0\end{bmatrix}\right\}$$

The results are shown in the Appendix; here we represent the results using the graphs

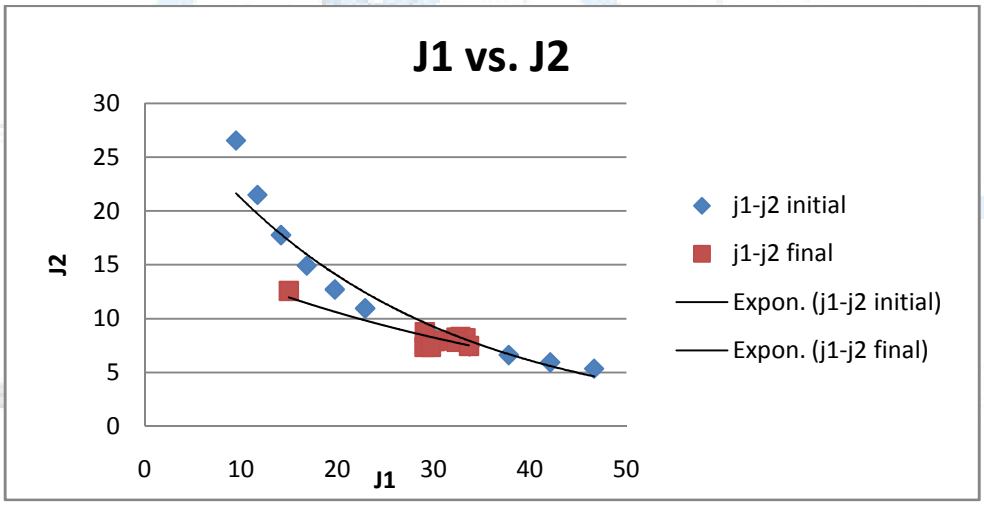

Figure 2-2: Pareto Front for J_1 vs. J_2

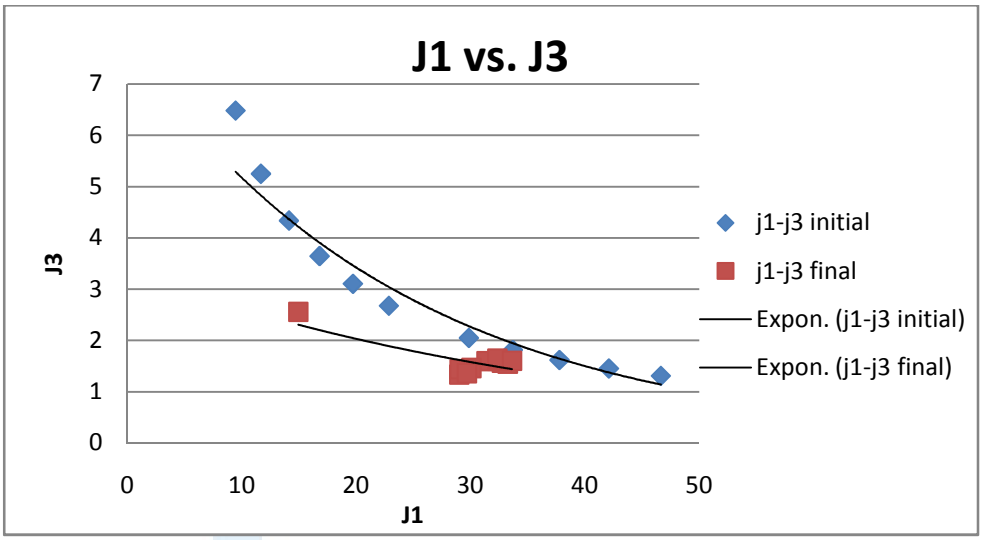

Figure 2-3: Pareto front for J_1 vs. J_3

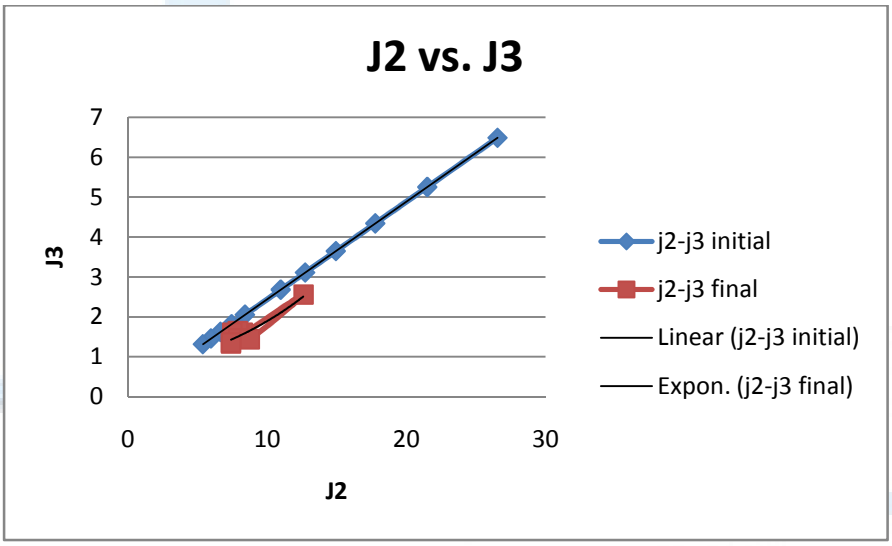

Figure 2-4: Pareto front for J_2 vs. J_3

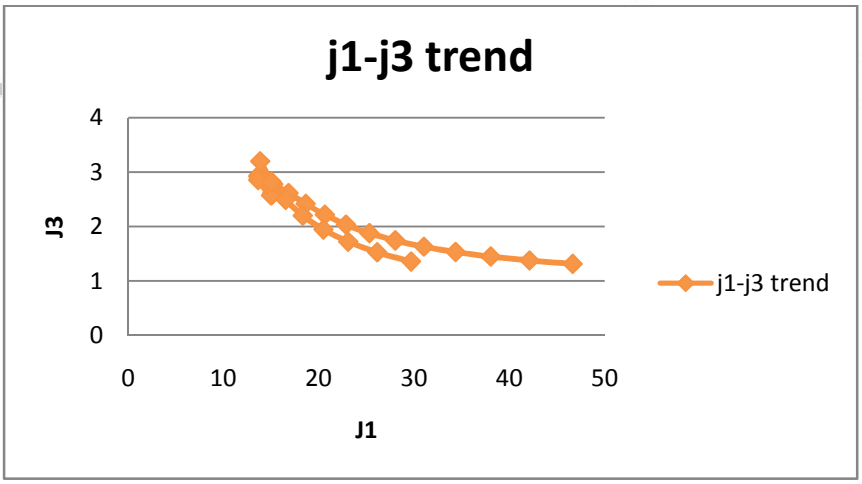

Figure 2-5: Trend for J_2 vs. J_3 as the iterations

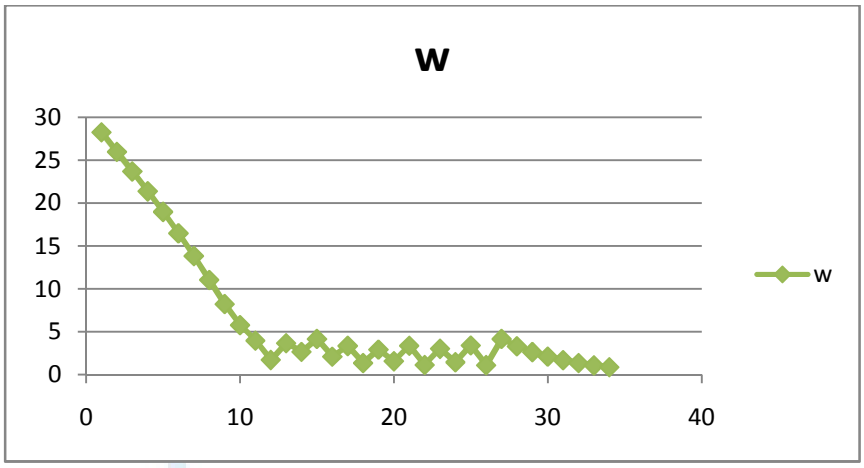

Figure 2-6: Trend of decrease in the value of ω

A.1 Appendix: Results for initial and final iteration

	initial			
Y	j1	j2	j3	w
0.9	9.442052	26.54321	6.481481	8.258974
1	11.65685	21.5	5.25	6.020793
1.1	14.10479	17.7686	4.338843	4.523512
1.2	16.78587	14.93056	3.645833	3.484256
1.3	19.70008	12.72189	3.106509	2.740463
1.4	22.84743	10.96939	2.678571	2.194167
1.6	29.84155	8.398438	2.050781	20.14365
1.7	33.68831	7.439446	1.816609	22.29636
1.8	37.76821	6.635802	1.62037	24.35648
1.9	42.08124	5.955679	1.454294	26.33168
2	46.62742	5.375	1.3125	28.23428

Table 1: Initial Values of the criteria J1, J2 and J3 and the combination of derivatives ω

	final			
Y	j1	j2	j3	w
0.9	32.73655	8.294122	1.562837	0.906045
1	33.25493	8.20128	1.548888	0.894161
1.1	32.3329	8.233263	1.603136	0.945527
1.2	31.38221	7.985843	1.595362	0.944998
1.3	32.34522	7.881773	1.641581	0.993237
1.4	33.62845	7.48323	1.603102	0.96563
1.6	29.00605	7.39153	1.333886	0.918005
1.7	30.07409	7.913961	1.460697	0.88562
1.8	29.05121	8.738566	1.440301	0.905873
1.9	14.92747	12.59104	2.556135	0.813994
2	29.67797	7.391229	1.356703	0.892021

Table 2: Final Values of the criteria J1, J2 and J3 and the combination of derivatives ω

A.2 Appendix: Scilab Code

```
clear
close()
clc

//Design vector
x = 2.0*ones(4,:);//rand(4,:);
x0 = x;

// Definition of Criteria J1, J2 and J3
function y=J1(x)
 y=[2*(2*x(1)^2+sqrt(2)*x(1)^2+sqrt(2)*x(3)^2+x(4)^2)];//[x(1)^2+ 2*x(1)*x(2) + x(2)*x(3) - x(4)^2];
endfunction

function y=J2(x)
 y=[3*(1/x(1)^2+1.5/x(2)^2+2/(3*x(1)^2)+2/x(3)^2+2/x(4)^2)];//[x(2)^2+ 2*x(1)*x(2) +
2*x(2)*x(3) + x(3)^2'];
endfunction

function y=J3(x)
 y=[1*(1/(4*x(4)^2)+2/x(3)^2+(1/x(1)^2+2/x(2)^2))];//[x(3)^2 - x(1)*x(2) + x(2)*x(3) - x(2)^2];
endfunction


//Function defined to calculate partial derivatives
function parq=partialq(a0,u)
 for i = 1 : l
  parq(i)=0;
  for j = 1 : l
   if j==i
    parq(i) = parq(i) + 2*a0(i)*sum((u(:,i)-u(:,3)).*u(:,j));
   else
    parq(i) = parq(i) + 2*a0(j)*sum((u(:,i)-u(:,3)).*u(:,j));
   end
  end
 end
```

```
endfunction

//Function defined to find minimum rho
function rhomin=rho(a0,parq)
 for i = 1:l
 if (parq(i)>0)
   r(i) = a0(i)/parq(i);
 else
   r(i) = (a0(i) - 1)/parq(i);
 end
end

r;

Q = sum(parq);
an = 1 - sum(a0);
if(Q>0)
 r(l+1) = (1 - an)/Q;
else
 r(l+1) = an/Q;
end

rhomin = min(r)
endfunction


//Function to calculate the omega of minimum norm
function [W0,W1,Wmin,anew,a0] = omegamin(ak_new,rhomin,parq)

//Defining Initial J1
for i = 1:l+1
 W1(:,i) = u(:,i)*1;//ak_new(i);
end
W1
//Defining flags which will be used to break from the loop if any conditions are violated
Wmin = W1;
anew = ak_new;
```

```
flag1 = 0;
flag2 = 0;
flag3 = 0;
 while (j <= k)
 //Putting the initial value in old
 ak_old = ak_new;
 j = j+1;
 rhok = j*rhomin/k;
 //calculating new value
 for i = 1 : 2
  ak_new(i) = a0(i) - rhok*parq(i);
  ak_new(l+1) = an + rhok*Q;
  if(ak_new(i)<0)
   'i', i
   //halt()
   flag1 = 1;
   break
  end
 end
 if (sum(ak_new)>1)
  'sum(ak_new)',sum(ak_new);
  ak_new;
  ak_old;
  //halt()
  flag2 = 1;
  break
 end
 if(flag1 == 1)
  ak_new(i) = ak_old(i);
  //halt()
  flag1 = 0;
  //break
 end
 W0 = W1;
 for i = 1:3
  W1(:,i) = u(:,i)*ak_new(i);
  if (norm(W1(:,i))>=norm(W0(:,i)))
```

```
    'i', ' The violating index', i;
    flag3 = 1;
    //halt()
    //break
   end
  end

  if(flag3 == 1)
   'The Minimum W'
   Wmin = W0;
   'The corresponding alpha'
   anew = ak_old';
   a0 = ak_old(1:l);
   break
  else
   'The Minimum W'
   Wmin = W1;
   'The corresponding alpha'
   anew = ak_new';
   a0 = ak_new(1:l);
  end

end
endfunction

function [j1n,j2n,j3n,x1] = criteria(u1,u2,u3,j1,j2,j3,sumW)
endfunction

//initialising everything
nsw = 10
tol = 1.0
it_w = 0

//Opening files to write the results
f1 = file('open','parq.txt','unknown')
f2a = file('open','u1s.txt','unknown')
f2b = file('open','u2s.txt','unknown')
```

```
f2c = file('open','u3s.txt','unknown')
f3 = file('open','js.txt','unknown')
f4 = file('open','ws.txt','unknown')

//Main loop for minimum omega that should be either 0 or less than tolerance
while(nsw>tol & it_w<100)
 it_w = it_w + 1;
//igrad = input("Do you want Gradient? Press 1 for yes, 0 for no.")

//Calculating the value of criteria at some point
j1 = J1(x);
j2 = J2(x);
j3 = J3(x);

//writing the value of criteria to the file
fprintf(f3,'%6.8f %6.8f %6.8f',j1,j2,j3)


//Calculating derivatives at the same point
[u1]=derivative(J1,x);//,H_form='blockmat');
[u2]=derivative(J2,x);//,H_form='blockmat');
[u3]=derivative(J3,x);//,H_form='blockmat');

//Writing all the values of the derivatives to the files
fprintf(f2a,'%6.6f %6.6f %6.6f %6.6f',u1)
fprintf(f2b,'%6.6f %6.6f %6.6f %6.6f',u2)
fprintf(f2c,'%6.6f %6.6f %6.6f %6.6f',u3)
```



```
//It is an option to view the value of criteria and its gradient
//if (igrad==1)
//  'The Criteria functions are'
//  'j1', j1
//  'j2', j2
//  'j3', j3
//  halt()
//  'The Gradients are'
//  'u1' , u1
```

```
//  'u2' , u2
//  'u3' , u3
//  halt()
//end
//
//if(igrad==0)
//'The Criteria functions are'
//  'j1', j1
//  'j2', j2
//  'j3', j3
//  halt()
 //end



//Making a matrix just for convenient use further in the program
u = [u1' u2' u3'];

//The number of criteria concerned will reflect with the number of a's
a = [1/3;1/3;1/3];
a0 = a(1:2);

//Defining cost function both with a =1 to n-1 and also with a = 1 to n
 deff('f=cost(a)','f=(norm((a(1))*u1+(a(2))*u2))^2');



 q = cost(a0);

 l = length(a0);

temp =0
eps = 0.000001
normW = 1
itr=0
//
 while(norm(normW-temp)>eps & itr<1000)
   //'The Partial Derivatives are'
   //'--------------------------'
```

```
//Defining partial derivatives of cost function q wrt all a's from i = 1 to n
//Printing partial derivatives and then finding rho for both a' = 0 or 1 with rho1 for a'=1 and rho2 for 0

parq = partialq(a0,u);

//Taking the sum of partial derivatives of q from 1 to n-1 to calculate alpha n
Q = sum(parq);
an = 1 - sum(a0);

//procedure called to find minimum rho among indvidual maximum
rhomin = rho(a0,parq);
k = 5;
ak_new = a0/k;
ak_old = a0;
j = 0;

//procedure called to find omega in each step
[W0,W1,W,a,a0]=omegamin(ak_new,rhomin,parq);
W;
temp = normW;
normW = norm(W);
itr=itr+1;
//loop defined for evaluating a (new ones) and also calculating
end

flag4 = 0;
sumW = W(:,1)+W(:,2)+W(:,3)
if (sumW == 0)
 'Pareto Optimality'
else
 //finding t max
 t1max = (j1/sum(u1'.*sumW));
 t2max = (j2/sum(u2'.*sumW));
 t3max = (j3/sum(u3'.*sumW));
 j1o = j1;
 j2o = j2;
 j3o = j3;
```

```
t = max(t1max,t2max,t3max);

itr = 0
//discretizing tmax and finding new values of criteria
for itr = 0:10
 h=itr*t/10;
 x1 = x-h*sumW;
 j1n = J1(x1);
 j2n = J2(x1);
 j3n = J3(x1);
 if (j1n>j1o)
  'condition break'
  flag4=1;
  break;
 end
 if (j2n>j2o)
  'condition break'
  flag4=1;
  break;
 end
 if (j3n>j3o)
  'condition break'
  flag4=1;
  break;
 end
 j1o = j1n;
 j2o = j2n;
 j3o = j3n;
end
if (flag4 == 1)
 j1n = j1o;
 j2n = j2o;
 j3n = j3o;
end
if (flag4 == 0)
 j1n;
```

```
    j2n;
    j3n;
   end
  end
  x = x1;
  d1 = j1 - j1n;
  d2 = j2 - j2n;
  d3 = j3 - j3n;
  nsw = norm(sumW)
  fprintf(f4,'%6.8f',nsw)
end


file('close',f1)
file('close',f2a)
file('close',f2b)
file('close',f2c)
file('close',f3)
file('close',f4)
```

References

[1] Andy J. Keane & Prasanth B. Nair: Computational Approaches for Aerospace Design; The Pursuit of Excellence, Published by: John Wiley and Sons, Ltd.

[2] Jean-Antoine Désidéri: Multiple Gradient Descent Algorithm (Research Report No. 6953, Year 2009), INRIA

[3] Jean-Antoine Désidéri: Split of Territories in Concurrent Optimization (Research Report No. 6108, Year 2007), INRIA

[4] Gilberto E. Urroz: Optimization Techniques with Scilab, http://www.infoclearinghouse.com/files/scilab/scilab12.pdf

[5] Michaël Baudin: The Optim Primitive, http://wiki.scilab.org/The_optim_primitive

[6] Joseph Frederic Bonnans: A variant of a Projected Variable Metric Method for Bound Constrained Optimization Problems (Research Report No. 0242, Year 1983), INRIA

[7] Tony Drummond: Libraries for Support of Optimization Problems OPT++ and TAO, Lawrence Berkley National Laboratory, http://acts.nersc.gov/events/SIAMCSE05/docs/OPT++TAO.pdf

[8] Carlos A. Coello Coello: Lecture Notes, Av. Instituto Politecnico Nacional, Mexico, http://www.cs.cinvestav.mx/~emooworkgroup/