Searching for improvement

M.A. Atherton[1] & R.A. Bates[2]
[1]*School of Engineering and Design, Brunel University, UK.*
[2]*Department of Statistics, London School of Economics, UK.*

Abstract

Engineering design can be thought of as a search for the best solutions to engineering problems. To perform an effective search, one must distinguish between competing designs and establish a measure of design quality, or *fitness*. To compare different designs, their features must be adequately described in a well-defined framework, which can mean separating the creative and analytical parts of the design process. By this we mean that a distinction is drawn between coming up with novel design concepts, or architectures, and the process of detailing or refining existing design architecture. In the case of a given design architecture, one can consider the set of all possible designs that could be created by varying its features. If it were possible to measure the fitness of all designs in this set, then one could identify a *fitness landscape* and search for the best possible solution for this design architecture. In this Chapter, the significance of the interactions between design features in defining the metaphorical fitness landscape is described. This highlights that the efficiency of a search algorithm is inextricably linked to the problem structure (and hence the landscape). Two approaches, namely, *Genetic Algorithms* (GA) and *Robust Engineering Design* (RED) are considered in some detail with reference to a case study on improving the design of cardiovascular stents.

1. Introduction

1.1 Search domains

The term *blue print* continues to be used, figuratively at least, long after the original device ceased to be widely used in engineering design. A blue print represented an expectation that the designer's intent would be faithfully reproduced in the finished artefact. It was not necessarily a plan of how to make the object but might indicate why any modifications to the original design had been made. Invariably these revisions of the blue print would be based on actual performance of the object and thus improved designs were often the result of trial-and-error. That is to say, the design process was heuristic.

'Blue prints' for every living thing on earth, uniquely encoded in the form of *deoxyribonucleic acid* (DNA), represent not only component parts but also their interrelationships within the total organism. Under changing circumstance, genetic code is said to adapt in order to survive over successive generations. This idea has been employed in engineering design process, notwithstanding the great differences in the respective timescales, economies and technologies between nature and engineering.

The description of an engineering system, embodied by its design, can be made on two levels: the basic operating principle of the system i.e. its specific *technology* and then, within that technology, particular configurations of design features. Decisions made in the design process at both these levels determine achievement, or otherwise, of successful function (the solution) for a given application (the problem). For example, an innovative concept for a mechanism will not be successful if inappropriate materials and geometry are chosen, and conversely a rather crude mechanism can perform successfully if the detail is right. Parallels can be drawn with nature such as in the design of an eye. At the technology level, design could relate to whether the eye is of a refractive (e.g. human) or reflective (e.g. lobster) configuration. At the feature level, design could relate to the values of lens dimensions and pupil shape that are assigned when the operating principle is refractive. However, there are lower limits on the dimensions of a retinal eye, as at very small scales it cannot function, i.e. there are parametric constraints. This example again highlights that engineering design operates in two broad domains. Designs can be categorized in terms of the technology used and then, within each technology, competing designs can be thought of as a collection of features that are defined by *design parameters*, also called *design factors*.

In engineering at least, the process of designing a solution that utilises new technology is very different to that for deciding design parameter values. The former is usually addressed as a problem of creativity and the latter can be formulated as a mathematical search problem. Figure 1 illustrates this distinction between design in the technology domain and design in the feature domain.

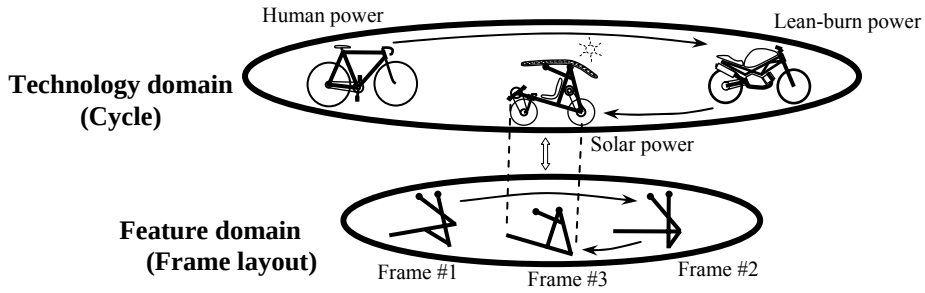

Figure 1: Abstract illustration of search within both technology and feature domains.

In other words, the description of an artefact and the process by which its description is arrived at are inextricably linked. This means that an integrated engineering design process must concurrently create an operating principle and also identify workable design factor values, yet it must employ different methods in the two domains.

Systematic mathematical search tools are practically limited to design in the feature domain due to the difficulty of expressing creativity in a symbolic language. However, there are systematic methods for proposing viable technologies that use knowledge of successful design solutions (e.g. patents) to focus the search on a small number of operating principles for evaluation [1]**.** Essentially, these methods still rely on the creative ability of the designer to make a successful interpretation in the context of the problem. In this Chapter we shall operate in the feature domain and consider it as a mathematical quest for improvement.

Mathematical search is viewed to take place within a *design space* defined by the number of design factors and the set of possible values for them, rather than all possible solutions. This defines the dimension and limits of the design space. For example, a system described by three design factors X_A, X_B and X_C, each having two possible values (e.g. 0 and 1), could be represented as a point in a three-dimensional design space, shown on a unit cube in Figure 2.

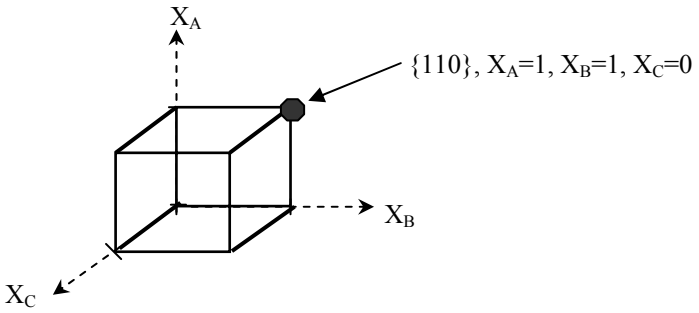

Figure 2: Design space for three design factors with binary levels.

An *efficient* search will rapidly converge on improved solutions regardless of the starting point. An *effective* search will yield significant improvements over existing designs. This implies that a good solution can be found without testing all possible solutions to the problem.

The above example describes a problem with *discrete* design factors. Each factor can take one of two possible values, 0 or 1 and consequently there are 2^3=8 possible solutions. In practice, design factors can be either discrete or continuous (e.g. take any value between 0 and 1). In the latter case, the set of possible solutions ceases to be finite. The distinction between discrete and continuous factors may not appear to be important, but in fact it can have a significant effect on how the design space is searched. This will be discussed in more detail later on in Sections 2 and 3. Design factors such as the length or weight of a component may be continuous, whereas the choice of material for the component may be a discrete factor. However, even in this simple case, the factors can be difficult to classify. The component may only be available in a fixed number of lengths and weights and, conversely, if the material is defined by a factor such as Young's Modulus, it may be possible to consider the material specification as a continuous factor. In all cases it is necessary to define the factors to accurately represent the problem to be solved. It may be the case that it is possible to manufacture customized components, allowing factors to be expressed as continuous, but this is an additional cost over and above the use of standard sizes. This needs to be taken into account when searching the design space for acceptable solutions.

1.2 Why use mathematical models?

Major issues in the design of any engineering system include cost, quality, reliability and demand. In this context, system optimisation can mean many things: minimising cost, improving reliability and so on. Each of these *objectives* will almost certainly be in conflict, and seeking design improvements that simultaneously satisfy them can therefore be a complicated process. Figure 3 outlines an example design scenario.

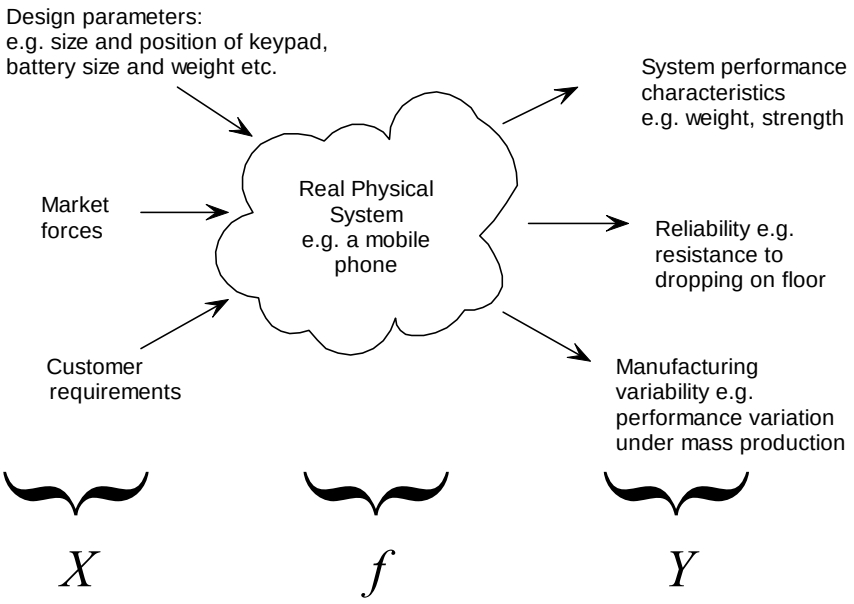

Figure 3: An example design scenario.

Systems that exhibit complex behaviour are often expensive to test during product development. This can immediately rule out a trial-and-error approach. An alternative strategy is to develop mathematical models of the system in order to gain understanding of the relationship between the design and its performance. Mathematically, the system is represented by an equation of the form:

$$Y = f(X) \tag{1}$$

The challenge here is to find *f*, in other words to find out how the inputs to the system (represented by *X*) affect the outputs of the system (represented by *Y*), as indicated in Figure 3. There are two basic approaches to characterising *f*: *physical modelling* and *empirical modelling*. The latter approach is known as a *black box* method as the details of the system are treated as entirely unknown. A third way of characterizing *f*, based on elements of both physical and empirical modelling, is known as *grey box* modelling.

1.2.1 Physical modelling

Complex engineering systems are generally designed from the principles of physics. This leads to mathematical models, often using differential equations, representing the system. An example of this is analogue electronic circuit design where characteristic equations exist for each component of the circuit and these are combined to form banks of differential equations that need to be solved to deduce the *ideal* behaviour of the system. The word ideal is important here as these models are only approximations to the real system, and need refinement if they are to reflect non-ideal behaviour such as manufacturing variation and losses due to electrical resistance, friction or other, possibly unforeseen effects. These physical models are very important as the mathematical theory behind them forms the basis of computer-aided engineering software such as finite element analysis, computational fluid dynamics, and electronic circuit analysis. Software that incorporates this type of analysis is referred to as a simulator, as it can be used to define a physical system and simulate its behaviour on a computer.

1.2.2 Empirical modelling

Physical systems can be tested to gather information about their behaviour. The field of experimental design is concerned with the design of such tests in order to maximise the information gained while minimising the size of the test. The test involves observing the system at a carefully chosen set of design points, each point representing a particular set of design factor values. Referring back to Figure 3, this means observing response features Y, made on the system f while varying the factors X. Once an experiment has been conducted, mathematical models are sought that fit the data gathered. These models are approximations of f, sometimes written as $\hat{f}$, and can be used both to estimate the relationships between factors and responses and to predict the response at untried factor values.

1.3 Building mathematical models

In general, whilst they both employ mathematics, physical and empirical modelling strategies have historically been separate but are becoming more closely linked via grey box modelling. For example a computer model can be used to provide structural information on systems as a starting point for experimentation [2], and estimates of variation in design factors and responses can be used to make physical models more accurate.

In some cases, mathematical models of systems can themselves be complex, and systems are often modelled with powerful computer simulators as described. Complex computer simulations of systems can however be very costly in terms of computation time and in this case black-box modelling can be used to construct simpler empirical models that are faster to evaluate, this is known as the field of computer experiments. Such models are referred to as emulators, meta-models, low-fidelity models or surrogates and their main characteristic is that they trade off accuracy for speed (Figure 4).

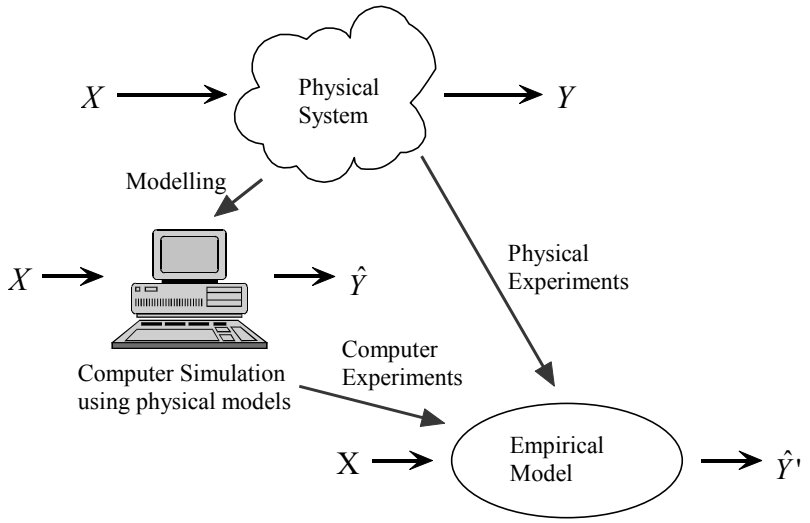

Figure 4: System modelling summary.

There is a close relationship between modelling and optimisation. The availability of mathematical models of complex systems opens the possibility of fully exploring the design space of all feasible combinations of factors to determine the best design, but even for small problems the dimension of the search space can be high and optimisation can still be difficult.

When considering the optimisation of a system or process there are several key decisions to be made about the nature of the search that will ultimately determine the level of success achievable.

(i) **Parameterisation:** The system must be described in terms of design factors, X, (the parameters) so that mathematical methods may be used for experimental design, modelling (both physical and empirical), and optimisation. The nature of each factor needs to be defined, for example whether they are discrete or continuous variables, the operating range etc. This is perhaps the most important aspect of the formulation of a search and optimisation strategy as it determines the set of all possible design solutions.

(ii) **Experimental design:** Any empirical model of the system needs information on how the system responds to changes in design factor values. In neural network terminology this is referred to as the training set. There may be additional constraints on any experiments to be conducted on the system such as non-regular or non-feasible parts of the design space (constraints on certain combinations of factor settings).

(iii) **Modelling strategy:** Any approximate model of the system needs to be accurate.

(iv) **Objectives (or fitness functions):** The objective, or combination of objectives, sometimes referred to as the fitness function, is a statement of the goal of the optimisation process. A typical example would be to maximise strength whilst minimising weight. Even in this relatively simple case one can see that there is a trade-off to be made.

(v) **Numerical optimisation:** Optimisation of the system (or a model of the system) can be either global or local in nature. Local methods seek to improve on previous solutions by changing factor values gradually, while global methods explore the design space more fully by making large changes to factor values. The two strategies can be combined by, for example, performing several competing local searches each at different starting points. Many optimisation algorithms exist, and choosing the right one for a given problem requires knowledge of the complexity of the problem. For example, are the functions to be optimised linear or nonlinear? Similar consideration must be given to any constraints on inputs and outputs of the system, which will also have a functional form.

All the above decisions on how to conduct the search combine to determine the set of possible solutions that will be found. In fact it is the objectives that drive the optimisation process and determine which are the most suitable methods to use. In the simple example of the strength/weight trade-off, it may be desirable to explore sets of possible design solutions that place different emphasis on the two objectives so that a light and weak solution is compared with a heavy and strong one. If this is the case then it is preferable to have models of the system that can be adjusted quickly and efficiently so that the solution space can be explored effectively.

Alternatively, it may be the case that the overall objective is well known and a direct search of the system is appropriate. In this case modelling may be unnecessary, particularly if evaluations of the system are inexpensive.

1.4 Design robustness and variability

An important part of the quality of any design is the ability to cope with unwanted variation or noise. This may be in the form of variation in factor values, variation in manufacturing conditions or variation in the use environment. In the context of design improvement, robustness means the ability of a design to maintain performance in the face of such noise. In order to understand how noise affects a particular design, one needs to first characterize the noise and then see how the design behaves when subjected to it. Unfortunately this can significantly increase the burden of testing, as design factors need to be varied on both a macro-scale, to search for an improved design, and a micro-scale, to identify how small changes in factor values affect performance. More detailed discussion of noise and robustness in design can be found in [3].

In terms of mathematical search, the design improvement problem changes from a deterministic problem, where exact factor values yield exact responses, to a more probabilistic formulation, where factor values are defined by statistical distributions and propagated through the system. Single response values then become response distributions that need to be optimised. Such a response distribution is most simply characterised by taking its mean and variance. Where previously the design improvement goal would be to maximize the response, now the goal might be, for example, maximizing the mean of the response and minimizing the variance of the response, this particular problem formulation is referred to as the *Dual Response* method [4].

The scenario just described, to minimize the variation in performance for a given level of input noise, is known as a *Parameter Design* problem. If one could imagine having control over the amount of noise the system is subject to, for example by specifying more accurate (and more costly) components, an alternative problem formulation could be to find a design that meets given targets of response variation for minimum cost. This is known as a *Tolerance Design* problem. Modern search strategies recognize that Parameter Design and Tolerance Design are linked and that good design solutions can only be reached by considering them simultaneously.

2 Fitness landscapes and interactions

2.1 Feature domains and design performance

The previous section introduced the concept of the feature domain in order to characterise competing engineering design solutions. A design is decomposed first into features and then into design factors that define the design space. Each design can then be thought of as a point in the design space, which represents the full set of possible design solutions associated with the specified engineering problem. In order to compare different designs, specific performance characteristics, or *responses*, must be defined such as weight, strength, power output and so on. These responses, taken together, describe the overall fitness of the design for its intended purpose. If one could imagine knowing the full set of performance characteristics for every design in the design space then this would define the *performance space*, representing the same set of design solutions from the perspective of design performance, rather than design factor values. Figure 5 describes this for the simple case where there are two design factors, $\{x_A, x_B\}$, and two responses, $\{y_A, y_B\}$.

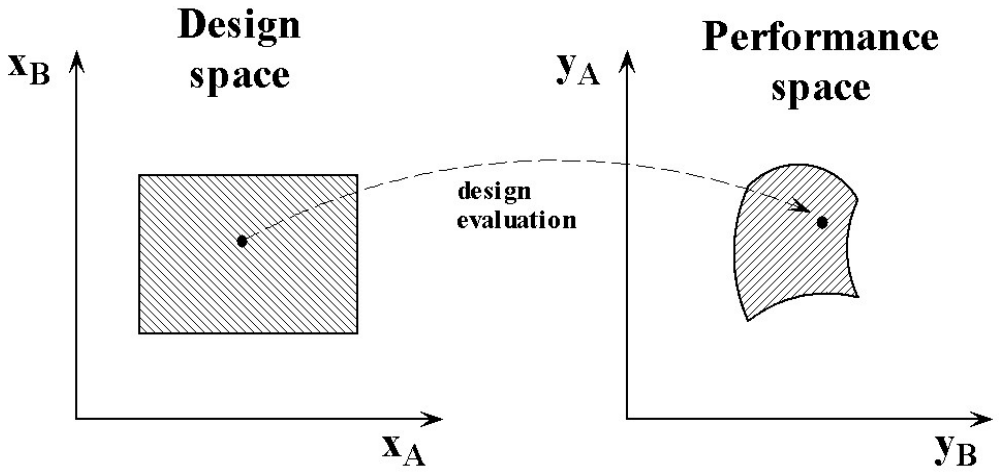

Figure 5: Design and performance spaces.

Of course it is easy to visualize such a case when there are only two design factors and two responses. However the concepts described are still valid for more complicated examples and can serve to describe techniques such as parameter design and tolerance design, mentioned in the previous Section, as well as other design methods such as design centring and yield optimisation that we will not mention further here.

2.2 Fitness for multiple purposes

Beyond three dimensions, the limitations of visualisation mean that diagrams such as that presented in Figure 2 can only provide a partial glimpse of n-dimensional design space. Representing performance requires an additional dimension. Therefore, in practice, attempts to plot design space search are abandoned for abstract mental images and instead simplified visualisations are used to plot performance against a subset of one (or two) design factor(s) or perhaps another performance objective. Indeed most engineering problems have more than one response to satisfy; i.e. they are multiple objective problems.

Satisfying multiple objectives is a challenge faced by organisms too, as we shall see. Invariably, due to interdependencies multiple objectives conflict with each other to the extent that as we increase satisfaction of one objective this typically results in decreasing satisfaction with the other objectives (Figure 6).

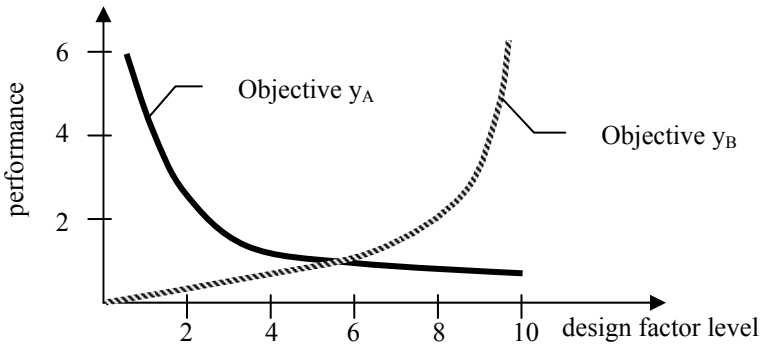

Figure 6: Two performance objectives plotted against a design factor.

Therefore compromise is usually inevitable for design confined to the parameter domain. Ignoring an improved concept design as a means of settling the conflict, trade-off is thus inevitable between multiple objectives and in parameter design two approaches are generally employed:

(a) Selecting one of the main objectives and incorporating the others as *constraints* [5].
(b) Employing a general 'portmanteau' unifying objective or *utility function* [6].

The utility function approach is often preferred for engineering robustness as it enables sensitivity analysis whilst in some cases with the former approach there is no feasible region of design space remaining after constraints are applied. The *desirability function* [7] is one such utility function. It transforms or maps each response into a desirability variable and then combines them geometrically into an overall desirability, *D*, which is effectively a continuous function of the responses. Thus a multivariate problem is expressed as a univariate one.

In biology, the overall performance of an organism is expressed as its *fitness*, in terms of its ability to survive and reproduce [8]. Fitness can be viewed as a utility function measuring survivability or level of adaptation. This level of adaptation can be likened to the elevation of a landscape (Figure 7) in which the peaks are populated by the higher living organisms. Here design factors and responses are combined in a single plot to indicate how adjusting the value of one factor can change the fitness, which is composed of response values.

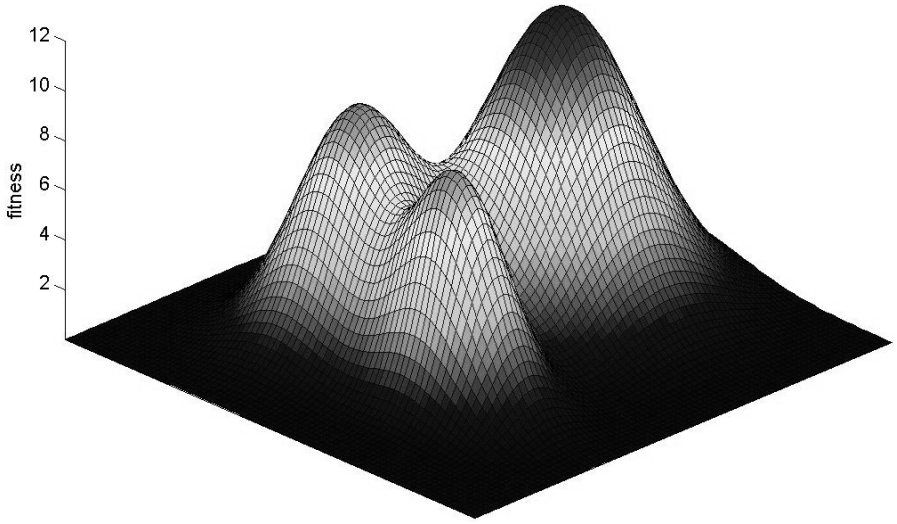

Figure 7: Theoretical fitness landscape.

This fitness landscape is a mathematical concept not a literal terrain but this vivid metaphor can be usefully manipulated. Imagine that the landscape is elastic and at the location for any particular organism the terrain deforms when the living conditions or the fitness of a contingent organism such as a predator, prey or parasite changes. Thus fitness has been termed a 'Red Queen Effect' [9], described as a never-ending race merely to sustain fitness level amidst co-adapting competition. This notion is seen to apply to economic systems. Taking this further we envisage that due to advances of competition the desirability (e.g. its utility function) of a product can diminish whilst its performance remains unchanged.

Quality in human technology has an aspect roughly analogous to biological fitness [10] and stress has been laid on quality loss functions [11] as a powerful measure of utility in engineering problems. The general idea being that the ideal target product performance is one that incurs zero loss-to-society in terms of the cost of, for example, environmental damage, maintenance, injury, inconvenience or some other expense not directly related to the intended function of the product. We now consider how the co-adaptation analogy might combine with the quality loss function in dealing with the multiple objective optimisation of diesel engines.

The primary intent of a diesel-cycle internal combustion engine is to produce useful tractive power. On each cycle of the engine most of the fuel is completely burnt and produces useful energy. The remainder of the fuel is not completely burnt and therefore pollutants, such as particulate ('Smoke') and unburnt hydrocarbon ('HC') emissions, are present in the exhaust gases. In both cases the quality loss function associated with these pollutants is ‘Smaller-the-Better’, as shown in Figure 8. Loss is assumed to be a quadratic function of each pollutant, such that $Loss = ky^2$, where y is, say, the mean output of a pollutant, k is a coefficient that specifies the quadratic curve and $Loss$ is measured in monetary units (British Pounds, £, for instance).

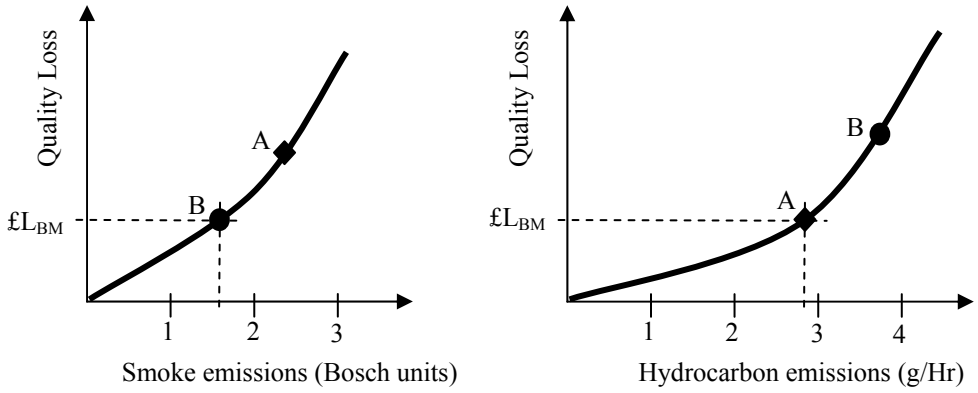

Figure 8: Quality loss functions of HC and smoke for product A and B showing a benchmark loss value, $£L_{BM}$.

Engines from two rival manufacturers, *A* and *B*, are depicted above in relation to each other for the two pollutants (i.e. two objectives): smoke (*S*) and hydrocarbons (*HC*). The performances, y_S and y_{HC}, of each engine can be measured fairly straightforwardly. However, we can't precisely define the quality loss functions for *S* and *HC* because the actual loss incurred for a given emission of pollutant, in terms of damage to health and property, reduced fuel economy and so on, is incalculable.

$$L_{HC} = k_{HC} y_{HC}^2 \tag{2}$$

$$L_S = k_S y_S^2 \tag{3}$$

But as competition contributes to the notion of a fitness landscape for organisms, so can competition help to define the quality loss functions in this case, as follows. Using the best performance (*benchmark, 'BM'*) for each pollutant, an arbitrary loss value, say $£L_{BM}$, can be assigned to both and thus k_{HC} and k_S, the coefficients of the two quadratic functions, determined from rearranging Equation (2) and Equation (3) respectively. Both pollutants are correlated as they are products of not-completely burnt fuel, this enables a portmanteau objective to be calculated for each engine performance, such as the overall 'fitness', total loss, $L = L_{HC} + L_S$. This loss changes if a new benchmark performance is reached or if a difference in the cost weighting between *S* and *HC* emerges. There is a trade-off relationship between the two pollutant emissions and determining quality loss functions by virtue of competitive benchmarking penalises products that stand still [12]. Thus the 'loss landscape' for each pollutant behaves as if it were elastic, changing according to competitive forces.

According to Goodwin [13] more sophisticated descriptions of landscapes tend to move away from the use of such non-generic fitness functions and towards language such as attractors and trajectories, attempting a unification of biology, mathematics and physics through the study of complexity.

2.3 Multi-Criteria Decision Making

Instead of combining individual objectives into a single fitness function, an alternative approach is to keep each performance measure separate. This leads to the idea of a performance space, described in Section 2.1, where it is then possible to show sets of competing design solutions. This is useful in trade-off situations where one objective is in conflict with another. In order to rank

competing solutions, the idea of *Pareto Optimality* [14] can be used, which involves the concept of dominance, where one solution is said to dominate another if one or more of its objective function values are better, and none are worse. This is graphically described in Figure 9 which shows four design solutions, $Y^{(1)},..,Y^{(4)}$ for a problem where the aim is to minimize two design objectives, y_A and y_B and Y represents some combination (or function) of the two objectives, $Y=f(y_A,y_B)$.

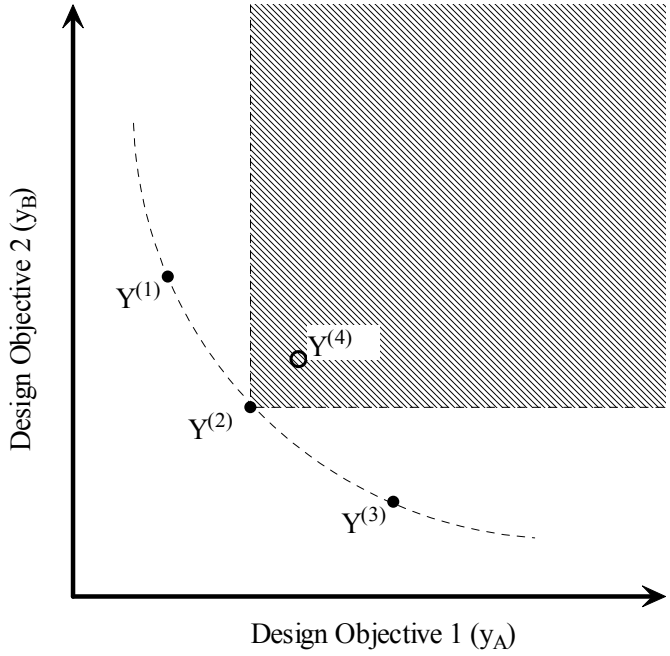

Figure 9: Pareto boundary

The shaded area in Figure 9 represents the region dominated by solution $Y^{(2)}$, therefore solution $Y^{(4)}$ is said to be dominated by $Y^{(2)}$ as its values for both design objectives are worse. The dotted curve shows an estimate of the Pareto Boundary or *Pareto Front*, which represents the set of all non-dominated solutions. In this simple example, the Pareto Front is assumed to be convex, but this may not necessarily be the case. Armed with such information, a designer would be in a good position to decide how to trade-off one objective against another in the search for the design factor values which represent the best design solution.

2.4 Coupling and search

The key to understanding the scope of natural selection theory depends on understanding the structure of the fitness landscape explored by an adapting population. For example, whether it is smooth and single-peaked, rugged and multi-peaked, or just completely random. One must also consider the mechanism by which the population adapts. The fitness landscape of Figure 7 is composed of two design factors and a fitness function, all of which can vary their values on a continuous scale. If a population is described in binary terms, such as a genetic encoding, then the design space becomes discrete and the relationship between one design and its nearest neighbour in design space is not well defined. One could say that the geometry of the search space has been weakened or even destroyed and therefore search strategies need to cope with this.

Genotype spaces are vast. Consider organisms with N different genes each of which has two versions, or alleles, 1 and 0. For a haploid population, such as that of *E-coli*, there are apparently 3000 genes [9]; therefore genotype space is 2^{3000} or 10^{900}. For a diploid population such as in plants that may have 20000 genes then genotype space is $2^{2000}*2^{2000}=10^{12000}$. Therefore let us consider walks across simpler fitness landscapes.

A genotype with three genes, N, each having two alleles, A, has $A^N = 8$ possible genotypes – {000}, {001}, {010}, …{111}. Each genotype is 'next to' those that differ by changing any one of the three genes to the alternative allele value. Figure 10 shows fitness values arbitrarily bestowed on each genotype. The arrows point uphill to fitter neighbours.

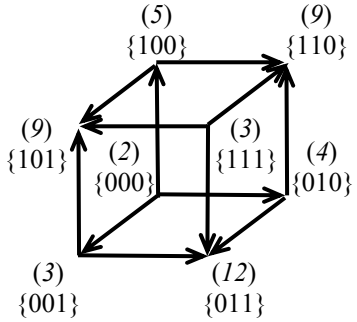

Figure 10: Genotype space (*showing fitness values of each genotype*).

An adaptive walk on this random fitness landscape only moves to a fitter variant from amongst the three immediate neighbours. In some cases these walks end at local peaks, {101} and {110} for example, as shown in Figure 11.

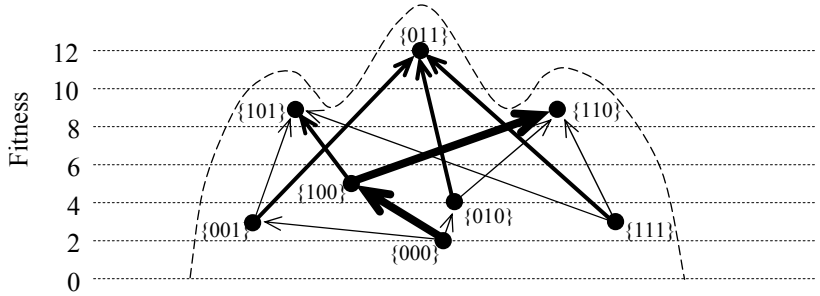

Figure 11: Fitness landscape showing walks of genotype space.

In this simple example there is one global best peak and two local peaks but in a large genotype space the number of local peaks on a random landscape is $2^N/(N+1)$. Hence for $N=100$ there are more than 10^{28} local peaks! Thus adaptive search on random landscapes is difficult because finding the global peak by uphill search becomes almost impossible. Searching the entire design space could feasibly exceed even the most generous estimates of the age of the universe unless more intelligent methods exist. Figure 10 also highlights the lack of geometry of problems posed in this way as the position of each genotype is plotted arbitrarily (in this case to echo the shape of the fitness landscape in Figure 7).

From any initial arbitrary point on a landscape, adaptive walks reach local peaks in a number of steps. The expected length of such walks to local peaks are very short ($ln\ N$) as any initial point is very close to one of the local peaks, which trap the adapting population and prevent further search for

distant higher peaks. Moreover, the higher the fitness, the more difficult it is to find improvement, as each step upward requires twice as many options being searched. However, real landscapes are not random, they are correlated, i.e. nearby points tend to have similar heights.

Gene epistasis or epistatic coupling is where the contribution of one allele of one gene to overall fitness of an organism depends in complex ways on the allele of other genes. Thus a network of epistatic interactions might exist. The *NK* model [9] captures such networks, where *K* reflects the degree to which nodes on the landscape interact. *K=0* represents total independence between nodes. *K=N-1* represents the highest possible value of *K* where all nodes interact with each other. In a more general sense, when $0 \leq K \leq N-1$ then the *K* genes assigned to interact with each gene are chosen at random. In effect *K* alters the ruggedness of the landscape. When *K=0* we have a single smooth-sided peak and as *K* is increased - genes are more interconnected - more conflicting constraints exist and so the landscape becomes more rugged with more local peaks (Figure 12).

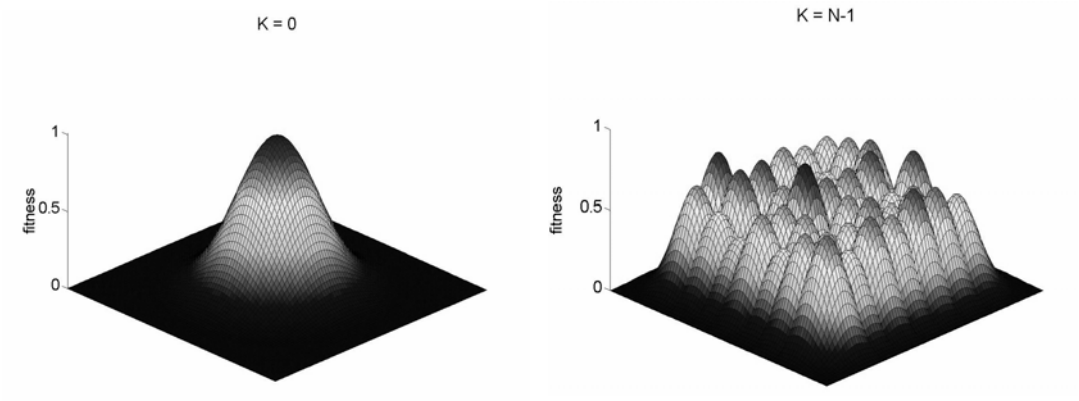

Figure 12: Effect of epistasis on the ruggedness of a fitness landscape.

Many rugged peaks occur because the best states of the shared epistatic inputs for one gene will be different than for its partner and thus in conflict - there is no way to satisfy both as much as if there was no cross-coupling between their epistatic inputs. In other words, as *K* increases there are so many constraints in conflict that there are a large number of compromises rather than a single best solution. As landscapes become more rugged, adaptation finds it more difficult to make the crossing. *K* is like increasing the compression of a compressed computer programme. With *K=0* changing any gene can only change the genotype fitness by at most *1/N*. Therefore the side of the peak is smooth and from any random starting point the number of directions uphill reduces by only one with each step. This dwindling of options is in sharp contrast to random landscapes where the number of uphill options reduces by half at each step. Gradualism works only on such a smooth single-peaked landscape. Thus as *K* increases the number of peaks increases, ruggedness increases, peak heights drop and locality of search increases. More interestingly, at moderate degrees of ruggedness, the highest peaks can be selected from the greatest number of critical positions; i.e. high peaks have the largest surrounding slopes [9].

3. Some methods for design improvement

Here we describe and compare in some detail two methods for searching the design space for improved designs. The first method, Robust Engineering Design, is built on the traditional field of Design of Experiments and has both a classical and a more modern approach. The second method defines the search problem in more biological terms and uses Genetic Algorithms to search for improvement.

3.1 Robust Engineering Design

Robust Engineering Design (RED) seeks to make engineering products robust to variation in both manufacture and use. A key aspect of RED is to understand the significance of each design factor on system performance through a highly structured search (Figure 12). Exposing the design to representative noise conditions and subsequently observing its behaviour are fundamental to the method. The design space can be searched directly, using physical prototypes or indirectly using a representative model such as a simulation model. Some parallels can be drawn with the search in *Genetic Algorithms* (GA) (see later section) but in general, for RED, very careful selection and arrangement of design factor values is required.

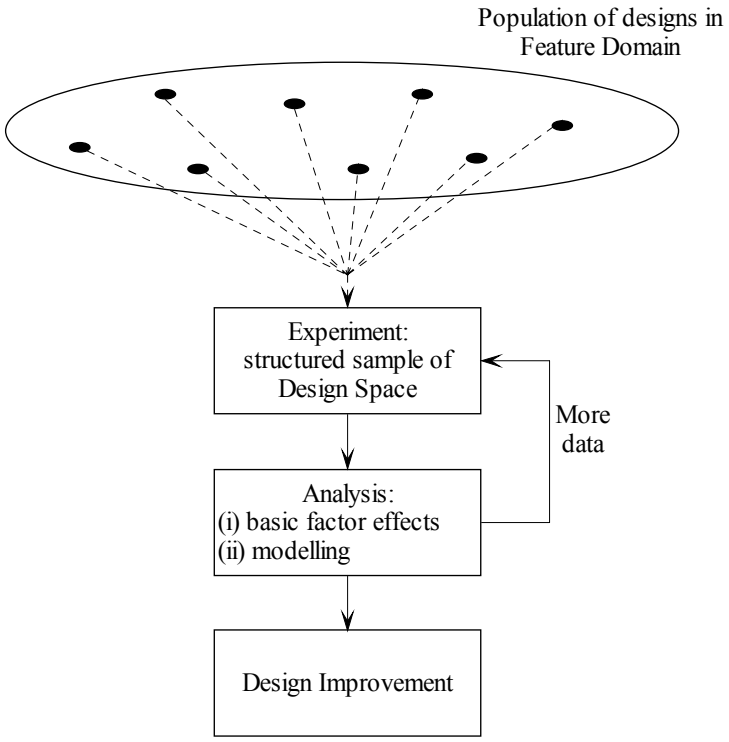

Figure 13: General RED procedure.

Figure 13 shows three main stages in the RED methodology: experimentation, analysis and design improvement, or optimisation. Experimentation involves choosing the type and size of an *experimental design plan* that will be used to evaluate different designs. Depending on the type of experiment chosen, the analysis stage interprets the results and provides information on the

relationship between the design factors and the responses. This information is carried forward to the optimisation stage, where improved designs are sought. Choosing and executing an experiment appears, on the face of it, to be the first step in applying RED methods. However, this can only be done once the method of design improvement has been decided. The first step is in fact to determine the design objectives. This will define how each design solution is judged and will point to the type of analysis method and therefore the type of experiment required.

There is no single method for performing RED; rather there are many different methods that can be used in the three stages described. One important distinction between different methods is whether a model is built to describe the relationship between factors and responses as part of the analysis stage (see Figure 4, Section 1.3). This is sometimes referred to as *Model-based RED*. Another important distinction for experimentation is whether, for model-based RED, the structure of the analysis model needs to be specified beforehand (model-based experimental design), or whether the data collected is used to determine the model structure (model-free experimental design).

3.1.1 'Classic' Robust Engineering Design

In relation to Equation 1, $y = f(x)$, the 'classic' approach to RED assumes that a simple *additive* relationship exists between the design factors (x) and some transformation, η, of the response (y). That is, *classic RED* is 'model first' in that usually a first or second order polynomial is budgeted to search the design space. An additive relationship is represented in Equation 5 for three design factors.

$$\eta = g_1(x_A) + g_2(x_B) + g_3(x_C) \tag{5}$$

It is important to note here that, in general, additivity with respect to η does not imply additivity with respect to y [15]. 'Additivity' is so central to classic RED, that the avoidance of interactions or cross terms (e.g. $x_A x_C$) between the chosen design factors is a dominant issue because they can render the assumed model unreliable.

(i) Orthogonal Arrays

An *Orthogonal Array* (OA) is a predetermined matrix commonly used for coding the design factor levels to be used in a set of classic RED experiments (Table 1). It is the experiment plan.

Table 1: Simple Orthogonal Array (L_4)

	design factor *A*	design factor *B*	design factor *C*	high noise data	low noise data	data transformation (unspecified)
Experiment α	0	0	0	$y_{11}\ y_{12}\ y_{13}$	$y_{14}\ y_{15}\ y_{16}$	η_α
Experiment β	0	1	1	$y_{21}\ y_{22}\ y_{23}$	$y_{24}\ y_{25}\ y_{26}$	η_β
Experiment γ	1	0	1	$y_{31}\ y_{32}\ y_{33}$	$y_{34}\ y_{35}\ y_{36}$	η_γ
Experiment δ	1	1	0	$y_{41}\ y_{42}\ y_{43}$	$y_{44}\ y_{45}\ y_{46}$	η_δ

Each column of an OA represents the values a particular design factor will take. The allocation of levels in each column is balanced with the other columns such that between any two columns each factor level is paired an equal number of times with the levels of the other columns and vice versa. The effect of this *orthogonality* is to search design space efficiently and also enable the average value of η for each design factor level to be compared. Data is collected for each experiment under discrete conditions of noise. Figure 14 illustrates the nature of the search and also highlights how each design factor is tested evenly against changes in the levels of other design factors.

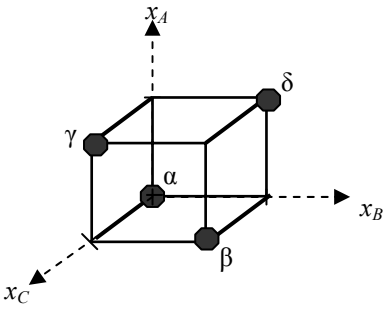

Figure 14: Balanced search of 3D-design space by Orthogonal Array.

From Table 1 it can be seen that, in terms of η, the average effect of design factor A at level 0 is calculated, according to the first column, as the mean η of the first two experiments (Equation 6).

$$\bar{\eta}_{A0} = \tfrac{1}{2}(\eta_\alpha + \eta_\beta) \tag{6}$$

And so on for all factor levels yielding six *mean design factor effects*, which are all the permutations of the two-value combination means from η_α, η_β, η_γ, η_δ (illustrated in Figure 15). For comparison with Figure 5, Experiment δ could be described as A1 B1 C0 or Design {110}; and therefore η_δ could be expressed as η_{A1B1C0} or more simply η_{110}.

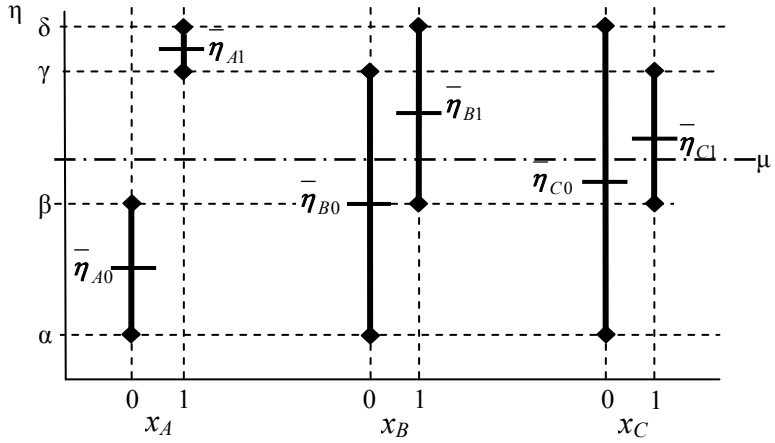

Figure 15: Mean design factor effects.

Consider predicting the value of η_{jkl} for an untried configuration $x_{Aj,}$ x_{Bk} *and* x_{Cl}, where j, k and l signify the levels of each design factor. From Equation 5 each of the three terms, e.g. $g_1(x_A)$, can be viewed as a contribution to η , and from Figure 15 this can be developed into Equation 7:

$$\eta_{jkl} = (\bar{\eta}_{Aj} - \mu + \frac{\mu}{3}) + (\bar{\eta}_{Bk} - \mu + \frac{\mu}{3}) + (\bar{\eta}_{Kl} - \mu + \frac{\mu}{3}) = \bar{\eta}_{Aj} + \bar{\eta}_{Bk} + \bar{\eta}_{Cl} - 2\mu \tag{7}$$

This is of more direct use with the Orthogonal Array for prediction than the more familiar general form of Equation 5.

There is an underlying assumption inherent to the OA/additive prediction model combination expressed in the above equations, i.e. the effects associated with all of the OA columns account for the system performance within acceptable confidence limits. In other words, any significant design

factors or interactions not handled by the columns that vary will corrupt the predictive power of the classic RED method. Therefore design factors are carefully selected, grouped and allocated to an OA in accordance with the additive model being used. In effect, these design factor assignments centre on the issue of interactions.

(ii) Interactions

Dealing with interactions in classic RED has two schools of thought. One school (e.g. [11]) advocates saturating the OA columns with design factors or combinations of factors. These assignments are judged to be independent of each other. The other school (e.g. [16]) allows some columns to be unassigned, in effect allocating these *degrees of freedom* to tracking the effects of potential interactions.

Modifying the general form of the simple additive model (Equation 5) to include an interaction term:

$$\eta = g_1(x_A) + g_2(x_B) + g_3(x_C) + g_4(\frac{x_A}{x_C}) \tag{8}$$

This now means that with an incremental change in x_A, *say* Δx_A, the contribution to η, say $\Delta\eta$ is also dependent on x_C and the coefficients g_1, g_3, and g_4. Indeed, the net effect of Δx_A on $\Delta\eta$ might be in the opposite direction to that without the interaction (Equation 5). In such cases this is termed negative or *antisynergistic* interaction, and if not included in the experiment plan, renders predictions unreliable for yielding improvement. Where interactions boost the effect of the design factors involved this is termed positive or *synergistic* interaction. The term *superadditivity* has also been used to describe the effects of design factors boosted by interactions.

Interactions may, to some extent at least, be an artefact of the scale, units or metric, or distribution of the original data. In such cases the interaction is considered to be *transformable* and a *data transformation*, expressed as η above, is considered to offer the potential to improve additivity [11, 17, 18]. Thus we seek a suitable transformation (Equation 9).

$$\eta = h(y) \tag{9}$$

(iii) Transformations

In classic RED it is desirable, when relevant, to differentiate between factor levels that most influence mean effects (*location effects*) and factor levels that minimise variability (*dispersion effects*). Therefore the transformations used often seek to reflect both the mean response and the variability in the response and are sometimes termed *Noise Performance Measures* (NPM).

In statistical terms data transformations attempt to enhance three statistical properties of the data [16, 18, 19]:

(a) Independence between mean and variance of each experimental trial.
(b) Simplicity of the mathematical model.
(c) Normality of error distribution.

Non-linear transformations such as $\eta = log(y)$ dominate those used, but have little effect unless the ratio y_{max}/y_{min} of all the data is greater than two or three.

The Signal-to-Noise Ratio (SNR) is a transformation that has been widely used in classic RED although it does not escape statistical criticism [15, 19]. But it does help to simplify the analysis and roughly demonstrates the statistical properties above. Moreover, it is linked to quality loss functions such as Equation 2.

For a set of quality characteristic readings, $y_1, y_2, \ldots, y_n$, the average quality loss, Q, is:

$$Q = (\frac{1}{n})\{L(y_1) + L(y_2) + \ldots + L(y_n)\} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{10}$$

For a 'Nominal-is-Best' (NB) problem, where M is the target value and the mean is μ, it can be shown that when n is large, Q approaches:

$$Q = k\{(\mu\text{-}M)^2 + \sigma^2\} \tag{11}$$

That is, the quality loss has two components:

(i) $k(\mu\text{-}M)^2$ an *accuracy quality loss* proportional to the deviation of the mean from the target.

(ii) $k\sigma^2$ a *precision quality loss* proportional to the mean squared deviation about the mean.

If Q is adjusted to bring the mean (μ) on target (M) then this first component will disappear and the second will be modified by the adjustment. This represents a two-stage optimisation philosophy **[9]**, which is also addressed later in this Chapter in model-based RED. The adjustment is to increase each reading by M/μ, which adjusts Q to the Quality Loss after adjustment, Q_a:

$$Q_a = k(\left[\frac{M}{\mu}\right]\sigma)^2 = kM^2(\frac{\sigma^2}{\mu^2}) \tag{12}$$

Attention need only be focused on (μ^2/σ^2), since for a given quality characteristic, k and M are constants. This is the Signal-to-Noise Ratio, and as σ^2 is the effect of noise factors and μ^2 is the desirable part of the data, then it can be viewed conceptually as the ratio of $\frac{\text{power of signal}}{\text{power of noise}}$.

Therefore, minimising Q_a, the quality loss after adjustment (or sensitivity to noise), is equivalent to maximising the inverse measure of variability proportional to mean, (μ^2/σ^2). It also converts what is in effect a constrained optimisation problem into an unconstrained one as there is only one metric to optimise rather than two, however this conversion does not allow for a thorough search of solution space, as described in Section 2.2. In view of Table 2, a $\log_{10}$ transformation could improve the additivity of the main effects, although generally this is sometimes applied thoughtlessly and is of questionable validity when it is. Thus, the SNR_{NB} based on Equation 12 is expressed in decibels as:

$$\eta = SNR_{NB}(dB) = 10\log_{10}(\frac{\mu^2}{\sigma^2}) \tag{13}$$

3.1.2 'Model-based' Robust Engineering Design

The goals of model-based RED are the same as for classic RED. The difference addressed here is that experimentation is used to build and validate an empirical model of the system that will then be used for engineering design. In the previous section there was some discussion about interactions and their effect on designing experiments. In this section, interactions are considered more generally as part of the experimental design and modelling problem.

We have already discussed the motivation for modelling when direct evaluation of the target system is not possible or feasible given constraints on time and resources. In the case of robust design the motivation for modelling is even stronger as we shall see.

(i) Definitions of robustness

The subject of robustness in an engineering sense can cover a wide range of concepts such as the ability of a system to cope with unexpected inputs or failure of sub-systems or components. The Santa Fe Institute, a research organisation committed to understanding complexity, has a working list of definitions [20]. We repeat here one definition of robustness which can be embodied by the following constrained optimisation problem:

1. Attain a target level of performance, subject to:
2. Minimising variation around that target.

This is related to the Signal-to-Noise Ratio of the previous section, but by redefining the problem in this way we retain generality over the problem and can look to the many algorithms available in numerical optimisation to help solve this type of constrained optimisation problem, including genetic algorithms, global random search algorithms and local optimisation methods such as steepest descent. Of course these algorithms require many evaluations of the system at different settings in their search for optimal solutions, which is why emulators that are fast and accurate statistical models of systems are important in this field.

(ii) Building accurate emulators

By definition, the target system is complex and expensive to evaluate. Complexity in this case means that the relationships between the design factors themselves may be non-linear and in turn their relationship to the systems response(s) may also be non-linear.

In the process of designing an experiment, an early decision to be made is whether to specify the emulator model ahead of performing the experiment or not. If this is possible then it is natural to ask the following question,

"Given a particular emulator, and a fixed number of trials, what is the best experimental design?"

By 'best experimental design' we mean a plan that will extract the maximum amount of information for a given cost, in this case the number of trials in the experiment. This leads to the field of *Optimal Design*, where certain characteristics of experimental design and model are optimised in order to maximise the efficiency of the experiment. For example, given the following polynomial emulator:

$$y = \phi_0 + \phi_1 x_A + \phi_2 x_B + \phi_3 x_A x_B + \phi_4 x_A^2$$

We could then ask the question:

"What is the best 7-point design to identify this emulator?"

We can start with a set of 7 points placed randomly with values in the range [-1, +1] and optimise them with respect to the chosen desirable characteristic to find the best experiment design. Figure 16 shows the results using *D-Optimal* design theory, where the determinant of the information matrix is maximised, see [21] for more detail on optimal design.

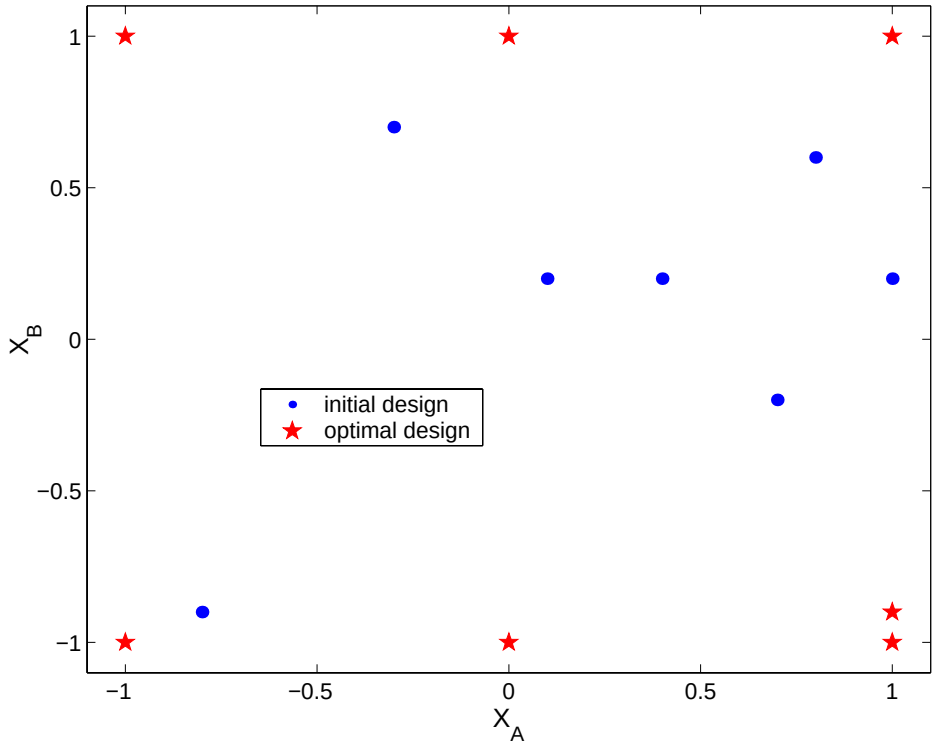

Figure 16: An example D-Optimal experiment design.

As we have discussed, it is generally the case that there is some knowledge of the system, but often not enough to confidently rule out possible interactions between factors. Indeed, it is often the case that even if there is some knowledge of interactions, these assumptions should be tested via experimentation. So, given that it may not be possible, or even desirable, to specify a particular model in advance of experimentation, the question to be asked becomes the following:
"What is the best experiment design for a fixed number of trials, given no prior assumptions on the model?"

In this case, the best experimental design is one that fills the design space in the most efficient way. Two standard space-filling designs are *Latin Hypercube Sampling* (LHS) designs [22], and *Lattice* designs [23]. These experimental designs seek to distribute observations evenly throughout the entire design region. The rationale is that we do not know anything about the behaviour of the system in the design region, so the best we can do is sample this space as evenly as possible. Other strategies, such as sequential design methods, may also be useful as information gathered by an initial experiment can be used to direct subsequent observations.

Using space-filling designs leads to the use of alternative emulator types, often referred to as spatial models, which seek to characterise the response surface in terms of the distance between observations.

These models do not require any assumptions on the relationships between factors to be made prior to experimentation, and are generally more adaptive than polynomial models. Figure 17 shows an example LHS design with 7 points and two factors.

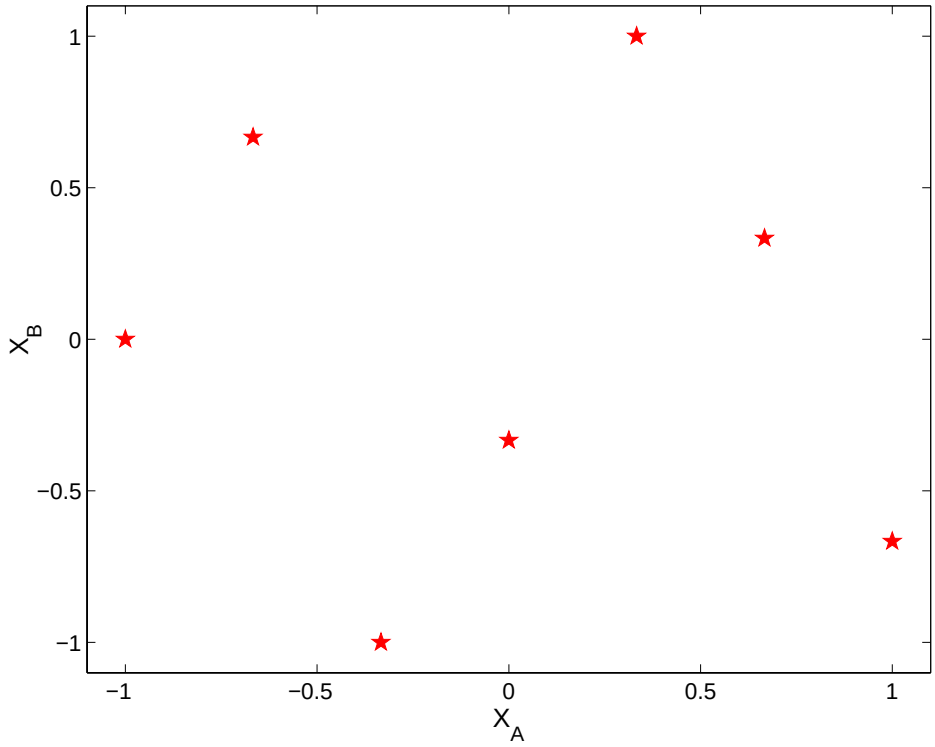

Figure 17: An example LHS design.

From this brief discussion one can see that there is a strong relationship between experimental design and modelling.

(iii) Emulator validation

Once constructed, the emulator models need to be validated to assess their accuracy. If it is not possible to conduct further trials, then statistical methods such as generalised cross-validation (GCV) can be used to estimate the accuracy of the emulators [24]. Otherwise additional experiments can be conducted at previously untried settings and the results compared with the equivalent emulator estimates to estimate prediction accuracy.

(iv) Using emulators for Robust Engineering Design

After conducting experiments and performing the emulator building and validation process, the emulators can be used for Robust Engineering Design. They can be evaluated directly at any point within the design region. In addition sensitivity analysis on the emulators themselves can be used to provide estimates of variability. The main advantage is that this can be achieved quickly, with an evaluation taking seconds, or even less, to perform. This means that designers are more inclined to perform what-if analysis, and a systematic search of the design space (for example using a global optimiser) will be more likely to find a globally optimal solution to the design problem.

3.2 Genetic Algorithms

Genetic Algorithms (GA) are founded on the theory of 'survival of the fittest' combined with the information exchange processes of natural genetics. This information exchange, which is structured yet pseudo-random, forms the basis of the search method. GA relies upon the assumption that in nature, complex non-linear relationships between design factors have to be processed efficiently. Therefore the system under investigation is considered to be a black box in which there are only two aspects of interest, namely the coding of the design configuration and its performance or 'fitness'. The GA procedure is illustrated in Figure 18.

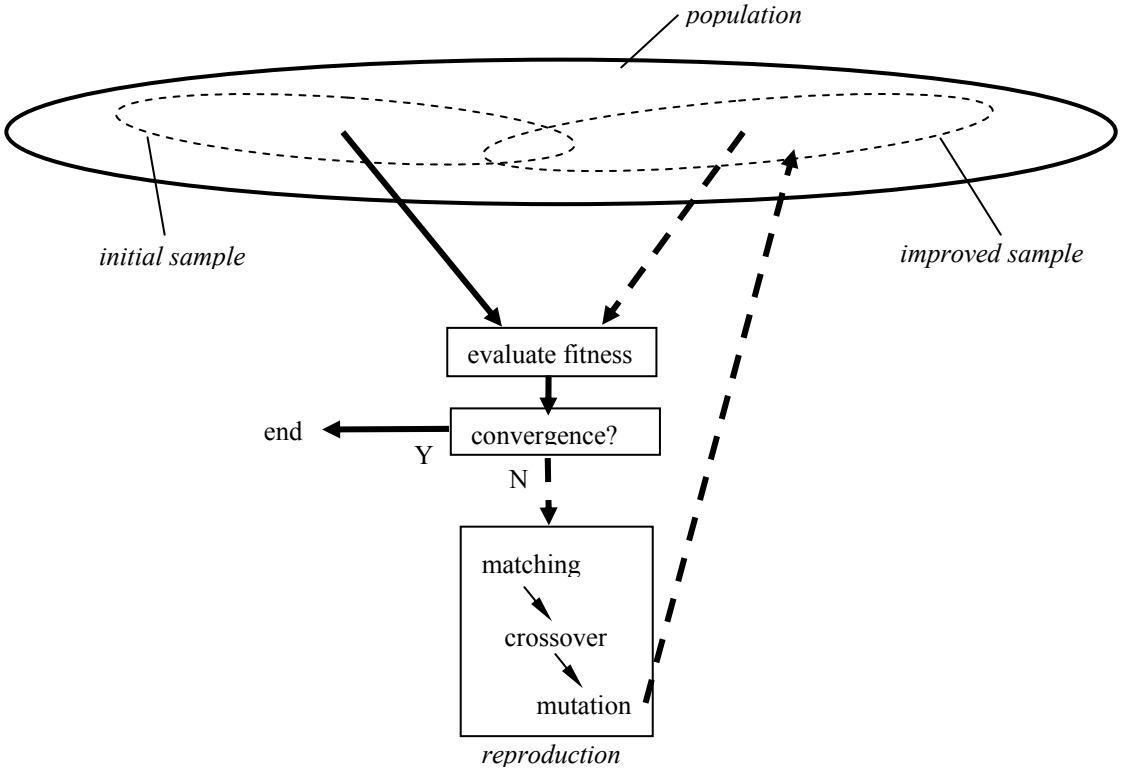

Figure 18: General GA procedure.

The starting point is an initial random sample population but too small a sample size risks the GA converging at a local optimum. Fixing of operator values in GAs is difficult as it depends upon problem type, population size, coding and other issues. Thus, wide ranges of values are quoted in the literature [23, 24].

For brevity, let us consider a simple example. An initial sample in a simple design experiment comprising four two-level design factors could be coded as shown in Table 2.

Table 2: Initial random GA coding.

	design factor *A*	design factor *B*	design factor *C*	design factor *D*	fitness
Exp 1	0	1	1	0	31
Exp 2	1	0	1	0	76
Exp 3	1	1	1	1	48
Exp 4	1	1	0	0	104

Reproduction progresses typically in terms of giving the design configuration ('string') with a higher fitness a greater role in spawning a subsequent generation until fitness values converge at a maximum value.

3.2.1 Matching

One method of matching is to allocate a higher probability of contribution to a dominant string based on its percentage of the total fitness for the generation ('sample' in Figure 18), as shown in Table 3.

Table 3: Initial random GA coding with matching probability values.

	design factor *A*	design factor *B*	design factor *C*	design factor *D*	fitness	% of total
Exp 1	0	1	1	0	31	12.0
Exp 2	1	0	1	0	76	29.3
Exp 3	1	1	1	1	48	18.5
Exp 4	1	1	0	0	104	40.2
					259	100.0

Strings selected for reproduction are entered into a mating pool. In Table 3, experiments 2 and 4 would have a relatively high probability of forming a mating pair based on their superior fitness.

3.2.2 Crossover

Crossover tends to pass on desirable traits. A position along the string is chosen as a crossover point, say in-between *B* and *C* in Table 5. Code either side of this crossover point is then swapped between the mating pair, as indicated in Table 4 below.

Table 4: Crossover of the GA coding.

First generation (parents)

A B C D

Exp. 2 = 1 0 | 1 0

Exp. 4 = 1 1 | 0 0

Second generation (offspring)

A B C D

Exp. 2' = 1 0 | 0 0

Exp. 4' = 1 1 | 1 0

3.2.3 Mutation

This plays a secondary but important role in producing a ‘random walk’ through design space by virtue of an occasional alteration of the value of a design factor.

For example if the first offspring in the second generation above underwent a random mutation of design factor *A* then perhaps Exp. 2' = {0000}. The incidence of mutations is generally limited to the order of between one per thousand and one hundred per thousand crossover transfers.

In general, further generations would be evaluated until the improvement in fitness converged to the desired level. As the generations unfold it enables the identification of successful combinations of design factors to be identified. These schema or building blocks can then be fixed, which focuses subsequent searches of design space.

3.2.4 Schemata and Epistasis

Comparing the code for Exp. 2 and Exp. 4 in Table 3 reveals that two alleles are common to both, namely, *A*1 and *D*0. This 'coadapted' set of alleles can be an indication of significant *epistasis* (interaction) between the two design factors.

A *schema* is a template incorporating a metasymbol, ‘*’, to represent all the strings that contain the epistasis in question, i.e. {1**0} for this case. Furthermore, *building blocks* are particularly fit, short schemata and play an important role in the Genetic Algorithm. The matching operator tends to be biased towards building blocks that possess higher fitness values thus ensuring their representation. Crossover and mutation have the ability to promote new building blocks but this tends to diminish with the crossover of similar strings. Building blocks tend to increase exponentially as a proportion of the sample population as the search continues - a fact apparently unique to GA and called *implicit parallelism*. Tracking the development of the best schema provides an estimate of the rate of the convergence of the GA.

Thus coding of interactions, i.e. building blocks, is critical to the performance of the GA. For example, simply placing the crossover between interacting alleles will destroy a schema.

3.2.5 Diploidy and Dominance.

A *Diploid* code is based on the double-stranded chromosome of DNA as opposed to the single strand of haploid organisms, which tend to be relatively uncomplicated lifeforms. The additional strand provides a mechanism for remembering useful alleles and allele combinations.

Effectively the redundant memory of diploidy permits multiple solutions to the same problem to be carried along with only one particular solution expressed. This helps the diploid population to adapt more quickly, particularly to changes in environment over time, compared with haploid coding.

Dominance identifies which allele takes precedence (is expressed) in genotype-phenotype mapping. This mapping should be allowed to develop.

A three-alphabet or triallelic scheme, -1, 0, 1 combines allele information and dominance mapping at a single position (Table 5). Here 0 dominates -1 and 1 dominates 0.

Table 5: Crossover of the GA coding with dominance.

First generation (parents *including reserved diploid code*)

A B C D
Exp 2 = 1 0 | 1 0
1 -1|-1 1
Exp 4 = 1 1 | 0 0
-1 0 |-1 0

Second generation (offspring *including reserved diploid code*)

A B C D
Exp 2' = 1 0 | 0 0
1 -1|-1 1
Exp 4' = 1 1 | 1 1
-1 0 |-1 0

Comparing Table 5 with Table 4, the resultant code for offspring Exp. 4' is {1111} instead of {1110} due to the reserved allele *D1* dominating D0. In addition, the reserved status operator shields such alleles from harmful selection in a currently hostile environment. A famous example is the peppered moth where the original white camouflage for lichen covered tree trunks was held in abeyance whilst a black form dominated in areas where trees had been darkened by the industrial revolution.

Mutation places a 'load' on the adaptive plan through its random movements away from the optimal configuration. Therefore it is desirable to keep mutation rate as low as possible, consistent with mutation's role of supplying missing alleles and without affecting the efficiency of the adaptive plan. Under dominance a given minimal rate of occurrence of alleles can be maintained with a mutation rate that is the square of the rate required without dominance. In other words, the robustness of search is enhanced by dominance.

3.3 Comparing model-based RED and GA for the design of cardiovascular stents

3.3.1 Background

It is common for human arteries to become blocked (a stenosis) by disease that can severely restrict blood flow to vital organs. Mechanical cage-like devices, known as cardiovascular stents, are often inserted to dilate these blockages and restore the blood flow. Unfortunately, without the intervention of drugs there is a significant risk that a stented artery will become re-blocked (a restenosis). Numerous investigations have identified the flow pattern over the stent to be a key factor and as a consequence elaborate stent patterns have been designed for less disruptive effects on the blood flow. The two successful stent patterns in Figure 19 can be seen to differ quite markedly, which raises the question *"Are there better untried stent pattern designs?"*

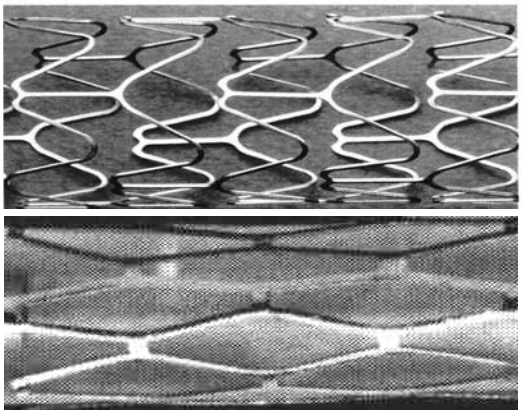

Figure 19: (upper) Guidant/ACS Multilink™ stent and (lower) Palmaz Schatz PS153™ stent [27].

Experimenting with new stent designs in live patients is not only a sensitive subject but it is also very difficult to gather flow measurements. In vitro experiments are more workable but are also time-consuming and costly. Therefore computer simulations are an attractive option in order to test a large number of stent patterns.

A reasonable first approximation is to model say a 3mm-diameter artery as an idealised cylinder, however the ratio of overall size to important stent detail, typically 30, severely limits mesh discretisation in the Computational Fluid Dynamics (CFD) model (Figure 20).

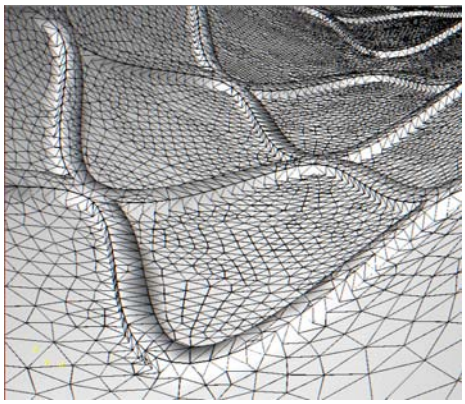

Figure 20: CFD mesh discretisation for full 3D-stent model of PS153™ [28].

We can simplify this model in two ways in order to improve this meshing. Firstly, assuming that the stent pattern is repeating, the model can focus on a single segment of the pattern (Figure 21).

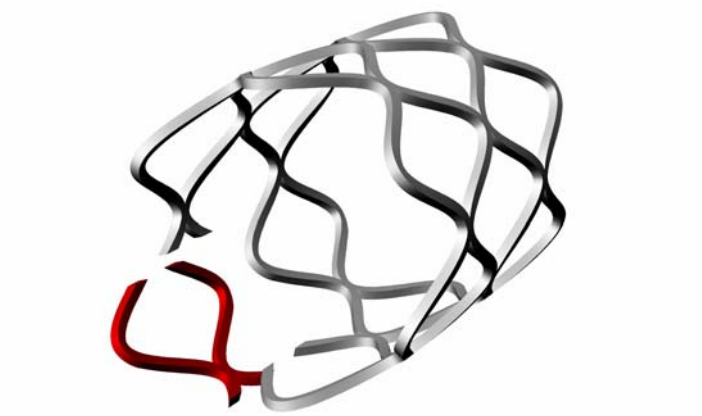

Figure 21: Partial model of PS153™ stent cut from a full stent [28].

Secondly, as the stent diameter is much larger than the thickness of material it is made from, then we can construct a flat model of the partial stent (Figure 22).

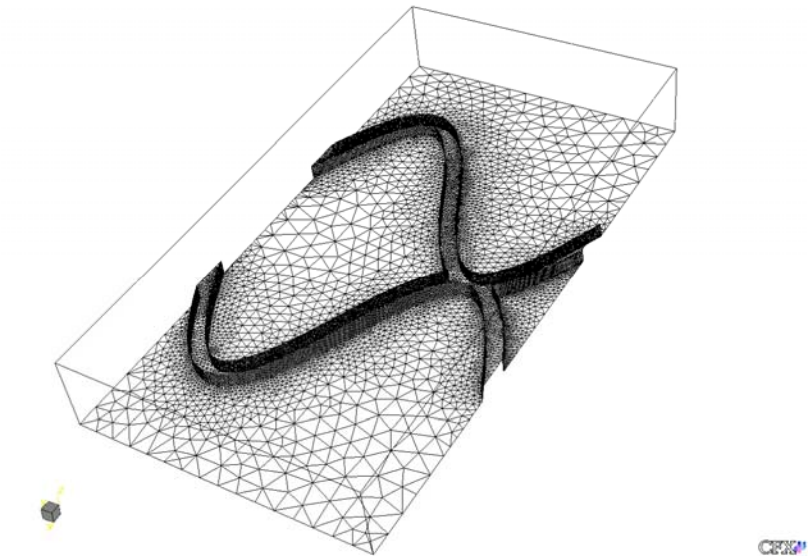

Figure 22: CFD mesh discretisation for partial model of stent.

Comparing Figures 20 and 22, the mesh discretisation and hence the fluid flow detail can be observed to be much finer in the partial model for similar computer memory allocation.

3.3.2 Parameterisation for Computer Experiments

Stents employ a variety of patterns, some elaborate, and the inference is that there are thousands of potential designs. In order to systematically explore the range of possibilities using computer models we must 'parameterise' the pattern; i.e. identify a number of key features or design factors that sufficiently capture the scope of stent design. Continuing our simple approach we can describe the generic repeating stent pattern using five design factors, namely:

(i) *Strut Thickness*: the thickness of the material from which the stent is cut and having a range of 0.08mm to 0.10mm.

(ii) *Strut Section Ratio*: expressed as the ratio of width to thickness, ranging from 1:1 to 1.5:1.

(iii) *Pattern Skew*: see Figure 23, defined by the relative position of the peak within one pitch (distance 1.0). Thus a value of 0.5 defines a symmetrical curve and 0.9 produces distinct asymmetry.

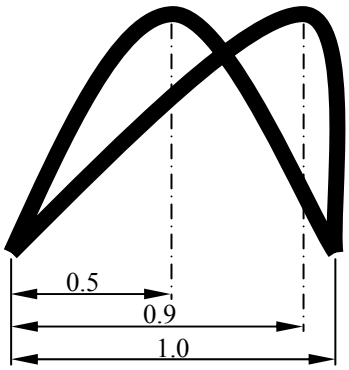

Figure 23: Range of Pattern Skew.

(iv) *Repeating Pattern*: specifies whether a longitudinally adjacent stent segment is merely a copy or a mirror image of the existing segment, i.e. two levels.

(v) *Shape Order*: defines the degree of curvature of a segment. Two levels were used.

Repeating Pattern and Shape Order for a symmetrical pattern are illustrated in Figure 24. The pattern on the left has a sharper '1st order' shape curve and is mirrored, whilst that on the right is a smoother '2nd order' shape curve copied longitudinally. Note that a copied pattern requires a link for structural integrity.

Figure 23: Effect of Repeating Pattern and Shape Order [28].

Noise factors were also considered in the model. Firstly, the degree of *strut embedding* in the artery wall, which has the effect of reducing the strut thickness in the CFD model. Secondly, the *flow inlet angle* to the partial stent model characterises the different flow conditions a stent design will experience depending upon patient and location.

Flow velocities or wall shear stresses in a 3D-flow field need to be summarised succinctly in order to quantitatively assess the performance of each stent design. We devised a scalar quantity that averaged wall shear stress over the whole surface, termed Dissipated Power [28] that was inspired by an observation that the diameter of arteries as they branch into smaller arteries do so according to minimisation of energy losses rather than a conservation of total area. Thus a minimum value for Dissipated Power was sought.

3.3.3 GA

Table 8 summarises the alleles used in the GA 'chromosome' for encoding stent designs. The two alleles used for both Strut Section Ratio and Strut Thickness have the capacity to represent four values but only three are required. Therefore incorporating a dummy level renders the fourth value in the allele sequence equal to the third. With this encoding the number of unique stent design combinations possible is 288.

Table 8: Chromosome encoding of stent design.

Strut Section Ratio		*Strut Thickness*		*Pattern Skew*			*Copy*	*Order*
0	1	1	0	0	1	1	1	0

With such a short chromosome length and a high simulation cost, the GA parameter settings used in order to avoid extreme local convergence were a population size of 10, crossover probability of 0.75 and mutation probability of 0.02. In our search 11 generations passed before convergence (Figure 24), involving 20 mutations and 40 crossovers. A total of 27 unique designs were tested under 4 noise conditions (a total of 108 CFD simulations), covering approximately 10% of the design space available.

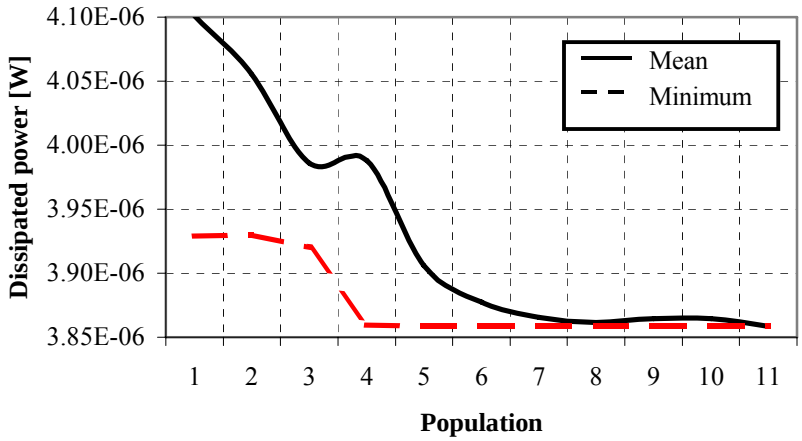

Figure 24: GA convergence [29].

The stent design solution at convergence is defined as shown in Table 9.

Table 9: 'Optimum' design resulting from GA.

Parameter	Value
Strut Section Ratio	1:1
Strut Thickness	0.08
Pattern Skew	0.5
Repeating Pattern	Mirror
Shape Order	1st
Dissipated Power (W^2)	92.25×10^{-6}

3.3.4 model-based RED

Following on from Section 3.1.2, a model-based RED approach using an emulator. This requires the design and noise factors to be continuous parameters and so the two discrete design factors, Shape Order (1st order) and Repeating Pattern (mirror), were fixed at the values already confirmed to be the best in initial studies. Thus only 12 trials were necessary (Table 10) in order to predict the response for any set of values for the three design factors and two noise factors.

Table 10: Experimental plan [30].

	continuous factor setting				
run no.	skew	thickness (mm)	width ratio	embedding (%)	inlet angle (deg)
1	0.65	0.087	1	36.36	60
2	0.57	0.089	3.55	7.27	0
3	0.79	0.093	1.73	21.82	10.91
4	0.72	0.095	2.45	50.91	54.55
5	0.54	0.1	2.09	43.64	32.73
6	0.9	0.098	3.91	58.18	27.27
7	0.68	0.091	4.64	14.55	43.64
8	0.86	0.096	2.82	65.45	21.82
9	0.5	0.08	1.36	0	16.36
10	0.83	0.082	4.27	72.73	5.45
11	0.76	0.084	5	29.09	38.18
12	0.61	0.086	3.18	80	49.09

The emulator is combined with a global optimiser in order to determine the values for the three design factors that yield the lowest value for the sum of squares of the Dissipated Power at the four noise factor settings. Assuming the same treatment for all four discrete factor settings this equates to a maximum of 12 x 4 = 48 CFD simulations.

The results plotted in Figure 25 show the main effects of the factors on the response and it can be seen that the inlet angle noise factor and the strut thickness design factor both have non-linear effects, which is of interest in identifying design solutions that are robust to noise.

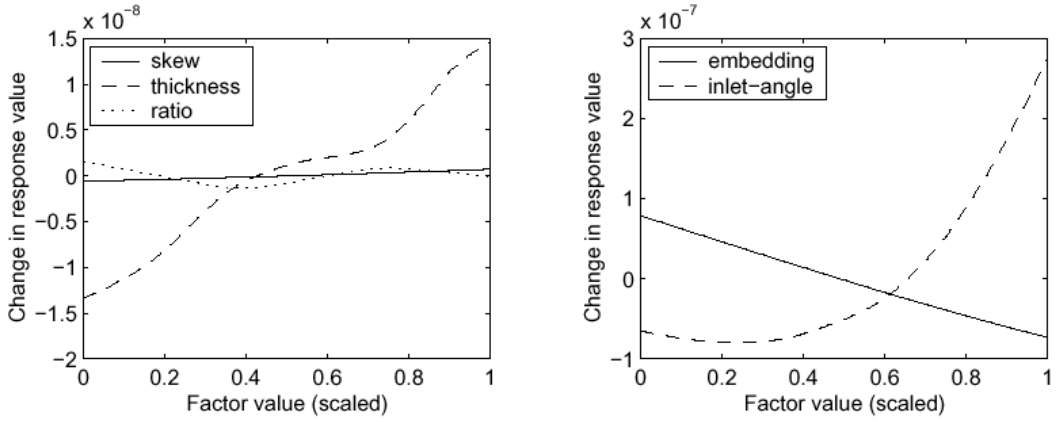

Figure 25: Emulator main effects plot (response values in W^2) [30].

The optimum configuration (Table 11) is very similar to that found by the GA but has a slightly better performance.

Table 11: 'Optimum' design resulting from RED.

Parameter	Value
Strut Section Ratio	1:1.5
Strut Thickness	0.08
Pattern Skew	0.518
Repeating Pattern	Mirror
Shape Order	1st
Dissipated Power (W^2)	91.07 x 10^{-6}

3.3.5 Discussion

GA and RED treat noise and robustness differently. The use of two levels for noise (high and low) for the GA immediately assumes that noise has a linear effect on the design response, whereas the model-based RED shows that inlet angle has a non-linear effect.

The GA treats continuous design factors as discrete, which restricts the search for an improved design and does not enable an understanding of how the design factors affect the response. However, RED treats the continuous factors as continuous and searches a larger space of designs as a result – but discrete factors must be considered separately. In addition RED provides insight into the design problem through analysis, and this may aid the designer in understanding the design problem and help in finding improved design solutions.

Both the GA and RED searches found improved designs, the RED design giving slightly better results (Figure 26).

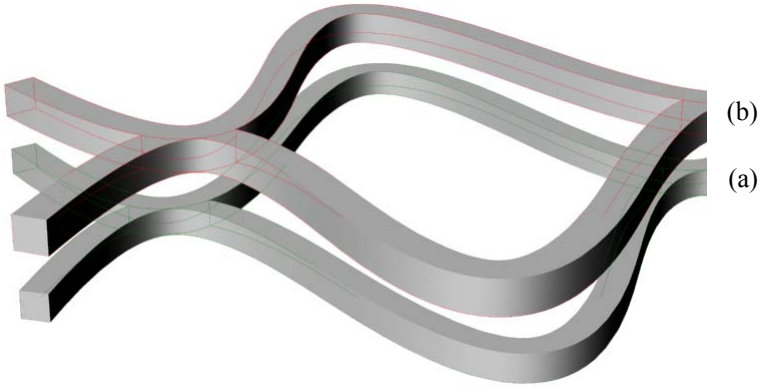

Figure 26: = Comparison of the two 'optimum' designs. CFD performance (Dissipated Power):
(a) GA = 92.25 x 10^{-6} W^2
(b) RED = 91.08 x 10^{-6} W^2

It is also interesting to note that the GA search required 27 x 4 = 108 CFD simulations, whereas the RED search required a maximum of 12 x 4 = 48 CFD simulations. In this medical engineering example, the stent pattern has to accommodate variations in artery geometry between patients. A more dynamic solution, if it were possible, would adopt whatever shape necessary in order to minimise disruption to the flow for each patient.

A large design space is produced by the few design factors considered, however neither GA nor RED can search technology options, rather they are parameter searches for improvement of an existing working principle, in other words adaptation. The GA search converged effectively within a few generations although choosing appropriate values for mutation and crossover was an additional uncertainty in configuring the search.

These studies both highlight some of the challenges involved in automating the redesign process and the importance of incorporating noise into the design process more generally. If sources of noise are not taken into account, then an improved design will not necessarily be robust to them and may fail as a result. The amount of time and resources available to the designer strongly influences the search method. In this regard, the RED method is more efficient as it required less design evaluations than the GA approach, and also achieved a marginally better result. The method of coding a design and the choice of performance measure are critical to the success of any strategy for design improvement. The use of dissipated power as a measure inspired by nature and solely used for improving the design seems to work well.

4. Summary

The powers and limitations of the theory of natural selection are not fully understood 140 years after Darwin's thesis. How stripes and spots appear is not explained by natural selection [13], it merely suggests that once there the pattern will stay if it offers an advantage. The popular image of natural selection, tirelessly sifting for useful variations among random mutations as the primary source of order, has in extreme cases led to a belief in gene survival as the principal driver above that of the host species. Goodwin [13, 31, 32] insists that the gene's eye view cannot be complete as some aspects of an organism's form persist in spite of natural selection, not because of it. In the early 20th century, Thompson [33, 34] raised the point that form was not selected, it was inevitable, an argument not inconsistent with Darwin's. Neither of these statements are 'Lamarckian heresy', i.e. the theory that evolution is a response to the environment. However, whilst Thompson was unable to persuade most of his peers of the importance of form and pattern formation, they have begun to remerge in the past two decades as an identifiable field of study.

The explosion in computer power has helped theoretical ideas about patterning that are difficult to test experimentally and the study of complex natural systems has begun to benefit engineering design. Kauffman [9] highlights two limitations to neo-Darwinian theory without self-organisation: Firstly, some systems change their behaviour massively with minor changes to detail. Secondly, accumulation of minor improvements does not always hold. For example, a maximally compressed computer programme has no redundancy and therefore it is very fragile to change. Hence starting with a long program becomes progressively more difficult to compress with an evolutionary search because as redundancy is squeezed out there are fewer and fewer clues as to where to search next. Not only that but a minimal program cannot be found by searching every possible configuration, as it could take aeons. Thus redundancy appears to be an essential element in assembling complex systems by adaptive search, and the processes of adaptation and product development are seen to be deeply similar. Therefore design factors, objective function(s) and search methods are intimately linked on the fitness landscape topology, and competition effectively renders the landscape elastic. Adaptive walks progressively worsen on the more rugged landscapes that result from strong interactions between design factors. This helps to explain why complex adaptive systems appear never to reach an endpoint.

What does this mean for the process of designing complex engineering systems?
Mechanisms of robustness are very different between nature and conventional engineering; this is an issue of complexity. Indeed, non-equilibrium may be more prevalent in engineering systems than we realise. Therefore more consideration should be given to the use of data transformations in design experiments to reveal hidden pattern. For example, phase space plots in determining the viable values for design factors or including a term for the rate of entropy production for dissipative systems. Not only is it impractical to search the entire design space but the true best design remains an unknown. We can conclude that improvement is often a sufficient description and a realistic goal in optimisation, and that is why the two words are used interchangeably in engineering.

Which search methods should be employed in engineering design?

It is tempting to see the relevance of our favourite theorems in a complex problem but the *No Free Lunch Theorems* (NFL) [35] show that the *average* performance of *any* pair of algorithms across *all* possible problems is *identical*. This means that if the structure of a problem is not incorporated into a search algorithm then there are no formal assurances that it will be more effective than even a random trial-and-error approach. Generally calculus-based, enumerative and random methods are ruled out because they are too demanding of knowledge and time. We have been unable to consider the full range of search methods based on natural phenomena such as *Simulated Annealing*, *Tabu Search* and *Ant Colony Search* [36]. Elements of these may be incorporated into improved search methods and, notwithstanding the NFL Theorems, a general theory of optimisation based on nature may yet emerge.

Engineering design is not limited to searching parameter values for improvement. In engineering design, improved global search, limited to the concept design level, has been made by classifying patented inventions so that an appropriate working principle can be matched to a given problem [1]. We have considered Genetic Algorithms (GA) and there is a beauty in their global performance through local action. The major shortcoming of GA is their complete dependence upon the 'detectors' (performance measures) to determine the coding, which risks a search too inefficient for expensive engineering experiments. In Robust Engineering Design (RED) we have seen that optimisation is about finding the underlying system function through physical and empirical modelling. In other words, information gathering is a more overt aspect of RED than GA. Modelling and optimisation can therefore be closely related in engineering design, which accords with the NFL Theorems. Apparent conflict between additivity, interactions, orthogonal search and fitness landscapes are tackled differently by RED and GA methods. The issue of interactions needs to be addressed carefully. In classic RED the use of linear models is dominated by additivity concerns, which restricts this approach to smaller regions of design space than the model-based RED approach.

There are several criteria for engineering design algorithms that emerge from the above consideration of search, namely:

(a) Design factors often need to be coded as discrete values rather than remain as continuous variables in order to configure a design space.
(b) In practice, by optimisation we mean improvement by virtue of selecting the best solution in the search space we have defined rather than the very best from all possible solutions.
(c) A useful method for engineering improvement is a trade-off between the more general global search methods and the specialised local search algorithms.
(d) It is important to efficiently search a large number of possible solutions without getting stuck at local optima.
(e) Probabilistic rules dominate the decision process, which are enhanced when populations rather than individuals form the basis of each search step.
(f) Directed search approaches are favoured from amongst the many optimisation methods and algorithms available to engineering, such as Genetic Algorithms and Robust Engineering Design.

Finally, one must consider the overall resources available and the complexity of the system under investigation when embarking upon a search for an optimal design solution. If there are unlimited resources for experimentation and the system is highly complex (and therefore difficult to model), then a simple random search may prove effective. If only a few observations of system performance are possible, then a more considered approach, involving a carefully designed experiment is likely to be the most appropriate path to follow.

References

[1] Altshuller, G. *The Innovation Algorithm: TRIZ, Systematic Innovation and Technical Creativity*, Technical Innovation Center, 1999.
[2] Atherton, M.A. & Bates, R.A. Bond graph analysis in Robust Engineering Design. *Qual. Reliab. Engng. Intl.*, **16**, pp. 325-335, 2000.
[3] Atherton, M.A. & Bates, R.A., Robustness and complexity, *Design in Nature Vol. 1: Nature & Design*, WIT Press, pp63-84, 2004.
[4] Vining, G.G. & Myers, R.H. Combining Taguchi and Response Surface Philosophies: A Dual Response Approach. *Journal of Quality Technology,* **22**, pp. 38-44, 1990.
[5] Dieter,G. *Engineering Design*, McGraw-Hill, 1983.
[6] Thurston, D.L., Carnahan, J.V. & Liu, T. Optimisation of design utility. *Trans of AMSE J. of Mech. Desig*n, **116**, pp. 801-808, 1997.

[7] Derringer, G. & Suich, R. Simultaneous optimisation of several response variables. *J. of Quality Technology*, **12(4)**, pp. 214-219, 1980.
[8] Bak, P. *How Nature Works: the science of self-organised criticality*, Oxford University Press: Oxford, pp. 118-121, 1997.
[9] Kauffman, S. *At Home in the Universe: the search for laws of self-organisation and complexity*, Oxford University Press: Oxford, pp. 216-217, 1995.
[10] Vogel, S. *Cat's Paws and Catapults: mechanical worlds of nature and people*, Penguin Science: London, pp. 246-247, 1998.
[11] Taguchi, G. *Introduction to Quality Engineering*, Asian Productivity Press and UNIPUB, p121, 1986.
[12] Atherton, M.A. & Wynn, H.P. Multiple quality objectives in robust engineering design. *Proc. of the 1st Int. Conf. on Quality and its Applications*, Newcastle-upon-Tyne, pp. 537-544, 1991.
[13] Goodwin, B. *How The Leopard Changed Its Spots*, Phoenix: London, pp. 92-96, 1997.
[14] Pareto, V. *Manual of political economy*; translated by Ann S. Schwier and Alfred N. Page. Macmillan, London, 1972.
[15] Parks, J.M. On stochastic optimization: Taguchi Methods™ demystified; its limitations and fallacy clarified. *Probabilistic Engineering Mechanics*, **16(1)**, pp. 87-101, 2001.
[16] Grove, D.M. & Davis, T.P. *Engineering Quality & Experimental Design*, Longman Scientific & Technical, Harlow, 1992.
[17] León, R.V., Shoemaker, A.C. & Kacker, R.N. Performance measures independent of adjustment, *Technometrics*, **29(3)**, pp. 253-265, 1987.
[18] Box, G.E.P. Signal-to-noise ratios, performance criteria and transformations. *Technometrics*, **30(1)**, pp. 19-27, 1988.
[19] Box, G.E.P., Hunter, G.H. & Hunter, J.S. *Statistics for Experimenters*, John Wiley & Sons, 1978.
[20] Santa Fe Institute: http://discuss.santafe.edu/robustness/stories/storyReader$9, accessed 23 November 2004.
[21] Pukelsheim, F., *Optimal Design of Experiments*, John Wiley & Sons, 1993.
[22] McKay, M.D., Conover, W.J. & Beckman, R.J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code, *Technometrics*, **21**, pp. 239-245, 1979.
[23] Fang, K.-T. & Wang, Y. *Number-Theoretic Methods in Statistics*, Chapman & Hall, 1994.
[24] Jobson, J.D., *Applied Multivariate Data Analysis Volume II: Categorical and multivariate methods,* Springer-Verlag, 1992.
[25] Holland, J.H. *Adaptation in Natural and Artificial Systems: an introductory analysis with applications to biology, control, and artificial intelligence*, MIT Press, Massachusetts, 1992.
[26] Goldberg, D.E. *Genetic Algorithms: in search, optimisation, and machine learning*, Addison-Wesley Longman, Massachusetts, 1989.
[27] Serruys, P.W. & Kutryk, M.J.B. *Handbook of Coronary Stents (2nd Edition)*, Martin Dunitz, 1998.
[28] Atherton, M.A., Tesch, K. & Collins, M.W. Effects of stents under asymmetric inflow conditions. *Biorheology*, **39(3-4)**, pp. 501-506, 2002.
[29] Tesch, K., Atherton, M.A. & Collins, M.W. Genetic Algorithm search for stent design improvements. *Adaptive Computing in Design and Manufacture V*, Springer-Verlag, pp. 99-107, 2002.
[30] Atherton, M.A., & Bates, R.A. Robust optimization of cardiovascular stents: a comparison of methods. *Engineering Optimization*, **36(2)**, pp207-217, 2004.
[31] Goodwin, B. Development as a Robust Natural Process. *Thinking about Biology*, eds. W.D. Stein & F.J. Varela, SFI Studies in the Sciences of Complexity, Lect. Note Vol. III, Addison-Wesley, pp. 123-148, 1993.

[32] Goodwin, B.C., Kauffman, S. & Murray, J.D. Is morphogenesis an intrinsically robust process? *J. Theor. Biol*., **163**, pp. 135-144, 1993.
[33] Thompson, D.W. *On Growth and Form (abridged)*, Cambridge University Press, Cambridge, 1997.
[34] Chaplain, M.A.J., Singh, G.D. & McLachlan, J.C. (eds). *On Growth and Form: spatio-temporal pattern formation in biology*, John Wiley & Sons, Chichester, 1999.
[35] Wolpert, D.H. & Macready, W.G. No Free Lunch Theorems for Optimization, *IEEE Transactions on Evolutionary Computation*, **1(1)**, pp. 67-81, 1997.
[36] Song, Y.-H. & Irving, M.R. Optimisation techniques for electrical power systems: Part 2 Heuristic optimisation methods, *IEE Power Engineering Journal*, June 2001, pp. 151-160, 2001.