

Computer Assisted Optimisation of Cable Design

Hugh G. Sasse, Mohammed M. Al-Asadi*, Alistair P. Duffy*;
Kenneth G. Hodge# & Arthur J. Willis#*

*Department of Electronic and Electrical
Engineering, De Montfort University, Leicester,
England LE1 9BH

#Brand-Rex Ltd, Viewfield Industrial Estate,
Glenrothes, Fife, Scotland KY6 2RS

Principle Contact: email hgs@dmu.ac.uk

Abstract

The design of communication cable systems requires that the best solution be found for several competing parameters. Often, a single ideal solution is not obtainable and a trade-off must be found. A means of visualising the solution space can assist in this search, as can computationally efficient search methods. This paper presents two such techniques: evolutionary computing (concentrating specifically on genetic algorithms); and "colour mapping" for visualising the space by mapping independent parameters into a colour space. These two techniques are complementary. Genetic algorithms are computationally intensive, simple to implement and can cope with a large number of variables. They have shown their utility in other fields, but seem to be under utilised in cable design. Colour maps are novel, computationally simple and require careful selection of the mapping functions. They are suitable for a limited number of independent variables but provide an output that is particularly intuitive.

1 Introduction

A Genetic Algorithm is a means of solving a broad class of optimisation problems. For example finding sets of input parameters to an equation or numerical model that give rise to a desired set of outputs. This is an approach based on responding to the statement: "We will know the answers are correct when we see them, but we are not sure what mix of input variables will give us these answers". The Genetic Algorithm is loosely based on Darwin's Theory of Evolution, and specifically how this evolution takes place in the cells of living things. The algorithms themselves have been well researched and applied to other areas of the engineering field[1,2], and they have features which make them attractive to the cable design community. This paper aims to explain the functioning of the genetic algorithm and suggests its role in cable design and manufacture.

There are a number of ways of solving optimisation problems: problems in which the best value(s) must be found for some variable(s). In many cases there may be a system of n equations with m unknown variables, such that $m > n$, therefore a unique solution may not exist; the purpose of the Genetic Algorithm is then to find the one or more families of variables that satisfy the equations. In this way this optimisation is a process of searching. Examples of other methods available for the solution of such problems include linear programming, hill climbing, simulated annealing, and purely random searches[3]. All of these methods including genetic algorithms have their own benefits and disadvantages. In order to present a balanced picture, the next section will review some of these techniques before concentrating on the Genetic Algorithm.

2. Optimisation Techniques

This section outlines some of the techniques used in optimising solutions to equations, or optimising numerical models all these cases there is some measure of quality of the proposed solution, and this may be considered to be the solution's fitness: fitness for the purpose to which the equation or model is put in fact.

2.1 Random Searches

Random searches are simple to implement, but they are an inefficient way of finding a solution to a given problem. Basically, this search method involves taking a series of guesses at the solution, until a satisfactory solution is found. Clearly there is no intelligence used in the search; results from one attempt provide no "seed" information for the next attempt at a solution. However, random perturbation can be helpful in other searching strategies, and random stimulation is even advocated in the discipline of lateral thinking, to provoke another train of thought that may yield a solution to the problem[4]. Randomness is an important part of other searching methods such as Genetic Algorithms where it is used in the mutation of genes and often in the selection of breeders, and, of course, in the creation of the initial population. An example would be finding values of inductance (L) and capacitance (C) in a tuned circuit, to give a desired resonant frequency. A random search would simply take a guess at L and C independently until two values were found to give the correct answer.

2.2 Linear Programming

Linear programming is basically a graphical method for use in problems where the equations that specify the problem are known and are linear for example in simple mixing problems. The equations are expressed as lines bounding a 2-dimensional area and some point in this area must be chosen to maximise some property such as profit. So the best solution lies on the boundary, usually at a vertex of an enclosing polygon.[5,6,7,8].

For this class of problem it can be very effective, and the concept has been extended to nonlinear programming, but there are many problems whose nature is very hard to express mathematically where the relationships governing performance are unknown, and then an appropriate heuristic approach is needed.

2.3 Hill Climbing

Hill climbing introduces the idea of a solution surface, which covers a solution space. If there are only two dimensions to the solution space this is like a hilly landscape where the "height" of a point on the surface is a measure of how good the solution is, i.e.

its fitness. The computer can only examine one point at a time, and cannot "see" the whole space. So the hill-climbing algorithm is to move to another, better, solution from the current one by only going in a direction that takes you "higher". Higher is fitter. This moving higher would involve testing points in the immediate neighbourhood of the current position to determine the best direction of travel. In practice the number of dimensions of the solution space is determined by the number of unknowns in the problem.

This technique does not need the problem to be specified purely as equations, and so can be used where the nature of the problem is not clear, but a solution's quality can be measured. For simple cases where the surface is convex such as the tuned circuit example in section 2.1, this method is adequate, but if there is more than one hill and valley in the landscape, then the climber may only reach the top of a small hill, and not find the mountain across the valley, even if the valley is shallow. If the tuned circuit is modelled so as to include third resonances, it is possible that one of these may be found, completely missing the fundamental resonance actually required.

To overcome the problem of finding a local maximum, or minimum (the climber would just keep going downhill to find a minimum), random stimulation can be used. In this case you tolerate moving away from a good solution in the hope that you will find a better solution by re-starting the search elsewhere. This leads naturally to the idea of simulated annealing.

2.4 Simulated annealing

When metal is heated and allowed to cool, the crystal structure of the metal changes. As the metal cools, the vibrating atoms in it tend to reach a configuration with minimal energy as they continually realign themselves with their neighbours. Simulated annealing regards the solution space as being at a given temperature, and a point is allowed to move about in this space (vibrate) due to this temperature. A movement towards a better solution is always accepted, but a move away from a good solution is only accepted probabilistically. The probability of accepting such a move increases with temperature.[8,9] In this analogy, energy is related to the inverse of fitness, so that minimal energy means maximal fitness. As the system "cools" (the vibration decreases) so the points move towards the minimal "energy" states, which represent the best quality solutions. This technique has been successfully applied in many areas, such as coding theory [10] and PCB design [11,12].

3 The Genetic Algorithm

3.1 Operation

The genetic algorithm is different from the search strategies discussed above because it is not based on a model of motion in a solution space, but it is based on a biological model. Competing individuals express their own solution to the problem within their "genetic makeup", and they are allowed to breed to create new offspring, which will hopefully benefit from the mix of solutions (genes) from their parents. Less fit individuals may have less chance of breeding, and their genetic lines will die off relatively quickly.

The makeup of an individual is a (proposed) solution to the problem. This is expressed in a chromosome, the genes of which hold the parameters subject to variation which may yield a

solution. This chromosome is represented in the computer as a string of bits. What this string represents depends on the problem being tackled: binary strings can represent anything in a computer so it may be representing several integers or floating point numbers, or complex numbers, parts of a parse tree (where the solution is a program) or even a series of characters to be fed to an application. The manipulation of this chromosome as a genetic entity takes place entirely at a bitwise level, only the testing of proposed solutions extracts the parameters from it and applies them to the problem.

Individuals have associated with their chromosome a fitness. How this fitness is determined is problem specific. It is usual for the fitness to relate to the breeding chances of the individual. This reflects the cases in nature where only the fittest dominant animals, best adapted for their environment, get to select mates.

The process of breeding is where two individuals copy their chromosomes and align the copies. They then exchange parts of these chromosomes with each other, so that the two resulting child chromosomes have components from each parent. This is called crossover. Therefore one child may have the first part of its chromosome before the crossover point from one parent, and the rest from the other parent. The second child is the other way around. The location of the crossover point, is selected at random, and in some genetic algorithms there may be more than one crossover point.

If the string of bits is now regarded as a collection of variables, then some variables will be received intact from one parent (those before the crossover point), and some will be received intact from the other parent. Therefore most of the variables will be received intact from the parents. The crossover point may land in the middle of a variable, so that will get some bits from one parent and some from the other, and thus will be a new value (if the variable is not an identical value in both parents). Hence, the location of the child's position in the solution space jumps relative to that of the parents. This overcomes the local maximum problem of hill climbing by avoiding the adherence to a local solution only, and is in some ways similar to a random search, but with *a priori* information from previous searches (the parents).

In biological systems, occasionally a mutation will occur; sometimes this can be beneficial for the species, particularly when the species is under evolutionary stress. In genetic algorithms mutation is achieved by altering the value of a bit (or bits) at random, with a very low probability. This mutation is done when the new individual is created. Mutation thus performs a jump to a random position in the solution space, reaping the benefits of random stimulation whilst holding on to the directed search of the genetic algorithm. It is important to realise that mutation is not the main agent of change in genetic algorithms or in nature, but that crossover is. That is why mutation rates are kept reasonably low. Allowing the mutation rate to vary when more change is necessary can be useful particularly when the diversity of the population is small, but change in the environment provides an evolutionary stressor.

Computationally, and practically, the population can only be finite, and there must be some means of imposing this limit. Commonly the least fit individuals are replaced by the children, so that even if the breeding parents are selected at random the later generations should, overall, be fitter than those before. Clearly the population must be large enough to ensure genetic diversity is maintained, whilst being small enough to permit timely evolution.

The fitness may be determined on a one-by-one basis, each

chromosome expressing a solution to the problem. Alternatively, if, for example, the problem being tackled is in the area of games, such as a chess playing program, then the individuals may measure their fitness against the whole of the current population. This is sometimes known as tournament selection [1]. As stated before, the fitness may be used to select which individuals breed. There are various schools of thought on this, one is that only the fittest should breed [13,14], and one is that a random selection biased towards the fittest should be used. Alternatively, an entirely random selection procedure could be used and the weakest weeded out by replacing them with the children of the breeders. If a strong "selectionist" strategy is used, so that only the fittest and no others are chosen as breeders, then the population can be saturated with one type of individual. This will usually result in a less than satisfactory solution, and the lack of genetic diversity will result in stagnation. The fitness of the population will not improve. This can be worked around by making the mutation rate a function of the diversity of the population, so that if the population is "saturated" then the mutation rate increases. If a less selectionist strategy is used convergence will take longer.

3.2 Factors affecting Performance of the GA

Variables in the genetic algorithm which can be modified to tune its performance are:

- The population size, where a small population may saturate with one chromosome type, or take many generations to find a solution. A large one may take too long for each generation.
- The chromosome implementation: how the problem space is expressed in the implementation. Here, a poor design of chromosome may lead to mutations taking individuals outside the solution space. Also, if the chromosome doesn't cover the solution space properly, the answer will be less than adequate; for example if the chromosome only represents a real when a complex number is needed then a G.A. could not find the square root of -1.
- The number of breeders per generation, where too few breeders may mean the convergence to a good solution takes too long. Having too many may mean that good solutions in a generation may only get one "breeding season" before they are overwritten.
- The mutation rate, if too high will result in damage to the offspring, too low and the diversity will not be maintained.
- How the mutation is implemented
- How the breeders are selected (The fittest, Monte Carlo, etc). Too much opportunity given to only the best individuals will result in the population saturating with the current best, so the GA reaches only a local maximum, and does not pick up valuable traits from the others in the population. Too weak a selection criterion will result in slow convergence.
- How the fitness is determined (tournament, etc). Tournament selection can be slow, but not re-adjusting a chromosome's relative fitness could lead to a weak population.

All of these factors have no clear solution and, therefore, experience and some trialing is beneficial. It may indeed be possible to optimise the performance using another genetic algorithm to find values for these parameters.

3.3 Colour maps

To obtain an intuitive grasp of the solution space, some means of representing it is necessary. If there is only one variable then the quality of the solutions may be represented by a graph. If there are two dimensions there a surface plot may be adequate. Often there will be more variables that can be considered. For example, the diameter of conductors in a cable, and also their separation will affect the impedance of the cable Z , the attenuation α , and also the return loss rl . Other parameters may also be affected, such as the manufacturing cost, the minimum bend diameter, and the weight.

If one considers the first three electrical characteristics above (Z , α , rl), it is clear that they will all be altered simultaneously as the conductor dimensions and spacing change. To represent three quantities on a surface plot is difficult, but since the human eye can respond to three different colour stimuli at once it seems intuitive to assign primary colours to each of these properties.

There is a large body of work on colour theory[16,17], and there are a number of computationally intense methods of representing colour. For the purposes of interpreting the results it was clear that an additive form of colour was needed, so that components that together contributed to the fitness could be seen to be adding up to give that fitness. Whilst the HSV colour system is more intuitive, the colour system supported by our printing language and by the visual display, is the RGB (Red, Green and Blue) colour space. This is computationally simpler though it is device dependent, but as can be seen from the results, this is not a significant problem.

It can be seen that the way these colours mix give identifiable regions within the solution space, which are immediately apprehended by the eye, since segmenting space into coloured regions is a natural operation for the eye. It is immediately obvious where the best performance (denoted by the whiter regions) lies, and the distribution of the primary colours gives an intuitive feel to how the cable's performance is affected by the variation of the diameter and separation parameters.

4 Results.

This section illustrates the implementation of a relatively simple GA to illustrate its application in a problem in the field of cable design.

4.1 Shielded Transmission Line

The first example chosen was to determine suitable dimensions for a shielded lossless transmission line so that it would have a given high frequency impedance.

The cable has 2 inner conductors, of radius d , the centres of which are separated by a distance s . The internal diameter of the outer screen is D and the dielectric thickness is τ . The impedance is given by [15]:

$$Z_0 = \frac{\eta}{\pi} \left(\ln \left(2p \left(\frac{1-q^2}{1+q^2} \right) \right) - \frac{(1+4p^2)}{16p^4} \right) (1-q^2)$$

where

$$p = \frac{S}{d}, \quad q = \frac{S}{D}, \quad \frac{\eta}{\pi} = 120 \Omega$$

The GA predicted results for $Z_0 = 100\Omega$ for a cable with air dielectric. Figure 1 illustrates results for varying the number of

breeders for various population sizes, and shows the cumulative number of successful convergences for the given population size. An ideal situation would be where there was successful convergence for every number of breeders (a gradient of 1 on the graph). It can be seen that this situation is approached as the population size increases. It is also clear that smaller populations provide fewer convergent solutions in general, due to the limited potential of the smaller gene pool.

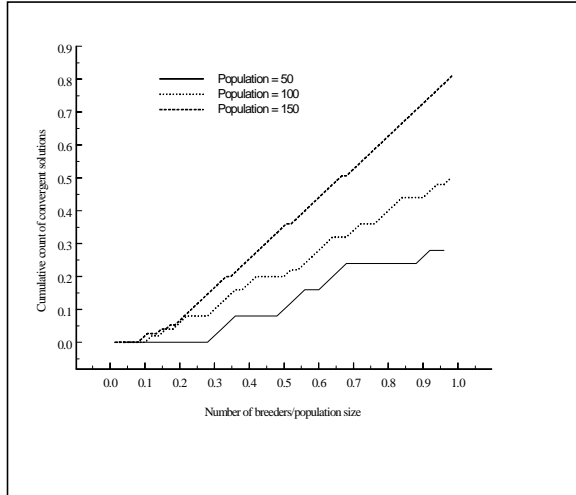


Figure 1 Proportional cumulative count of convergent solutions as a function of the number of breeder/population size for three values of population size.

Figure 2 shows p versus q for all solutions (including the non-convergent solutions, most of which were within a few milliohms of 100Ω and Figure 3 shows p versus q for only the convergent solutions. The resulting curve is clear, as are the non-convergent solutions.

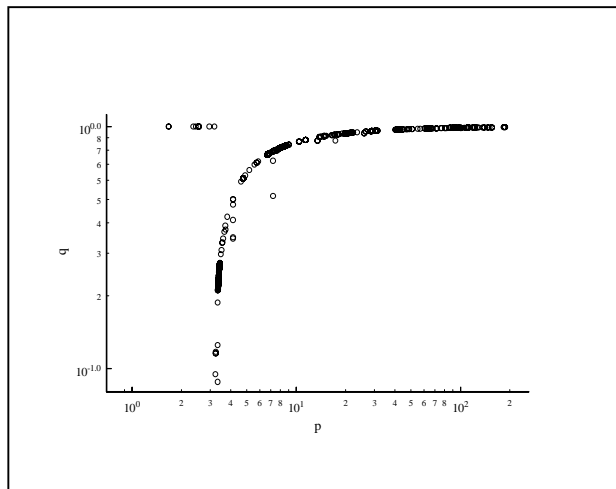


Figure 2. Solutions for a shielded cable (including non-convergent solutions) for 100Ω operation showing the design variables 'p' and 'q'.

A genetic algorithm was developed in C++ with a number of assumptions built into it which, while actually weakening it, provided further evidence to the robustness of the method, because the technique works even in sub-optimal circumstances. In particular the random number generator used in the GA was not initialised to a random value upon startup. This allowed repeatable trials, but meant that there was some inherent bias in the tests. Also the mutation function would only alter the value of one bit in the chromosome at any time, not each bit with equal probability (in which case more than one could be mutated at one time). Despite this the algorithm with mutation converged successfully in many cases. This demonstrates that although the impedance equation was one equation in two unknowns (p , q , to give the desired Z_0) a solution was still found.

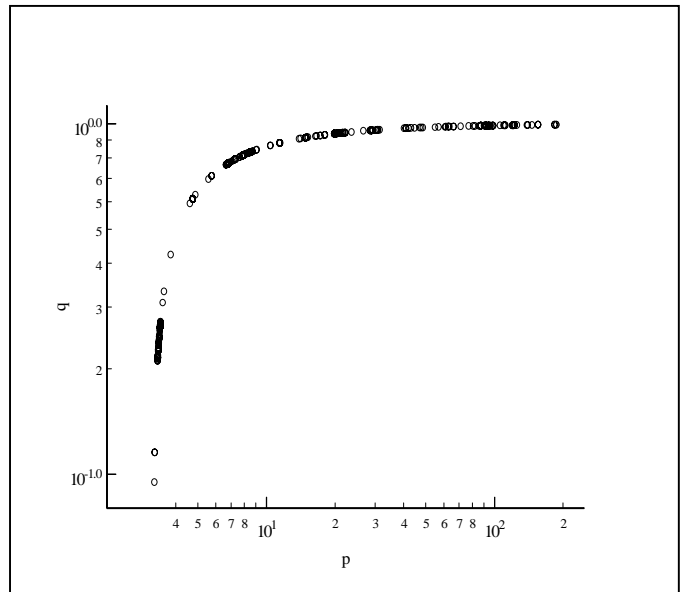


Figure 3. Reproduction of figure 2 (including only convergent solutions).

The problems of how to design the chromosome are well illustrated here. If $q > 1$ then an attempt will be made to take natural logarithm of a negative number. if $p < 1$ then the two inner conductors will touch. Therefore it was necessary to ensure that the binary representations in the chromosome could not produce these values of p and q . This was achieved by a linear mapping from p to p' such that $p' = (m p + c)$, choosing m and c so that p' is in the correct domain; similarly for q .

Tests were run for this equation with different population sizes and number of breeders per generation. The results broadly support the view that more breeders speeds up convergence, that large populations converge more quickly than small populations, and that even when convergence took more than the expected time, the fitness of the fittest variable was still fair (less than one milliohm out in most cases).

The fitness functions for impedance, return loss and attenuation are displayed using a colour map in Figure 4.

5 Other Possible Applications

Genetic algorithms are robust, and they do not need an exact

description of how to solve the problem. Provided that a solution can be recognised when it is found, they lend themselves to areas where analysis is difficult.

- Simultaneous Equations with N equations in $N + k$ unknowns. The equation for impedance in the shielded cable example above was for $N = 1, k = 2$.
- In numerical modelling if the GA is to evolve a model being simulated, then the fitness of the simulation could be tested at intervals. If this fitness is less than satisfactory, then the simulation could cease, otherwise it could continue for a more accurate result.

6 Discussion and Conclusion

This paper has introduced Genetic Algorithms in the context of other techniques to solve problems where a "good" solution may be identified but the input variables cannot be clearly identified. Many of the aspects which make GAs difficult to implement successfully have been introduced. A specific set of results were presented which identified the design ratios for the familiar problem of a shielded transmission line. Additional results for varying population sizes and numbers of breeding pairs in each generation showed how the effectiveness of the GA generally increases as these two increase, however there is a clear observation about the marginal gains of constantly increasing these.

Providing they are well thought out, and properly implemented, Genetic Algorithms are an additional powerful weapon in the armoury of the cable designer.

A technique for giving an overview of the problem space has also been demonstrated, using colour maps. This has been shown to be an intuitive means of grasping the interaction of a number of facets of the problem.

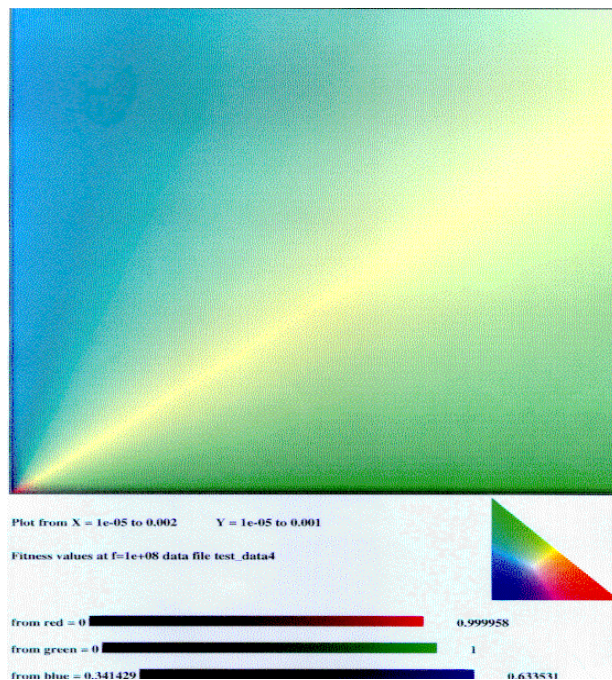


Figure 4 Colour map for fitness functions, red = Z_0 , green = RL , blue = α . Fitness is proportional to colour intensity where. X axis is τ and Y-axis is S .

7 References

- [1] Heitkoetter, Joerg and Beasley, David, eds. (1998) "The Hitch-Hiker's Guide to Evolutionary Computation: A List of Frequently Asked Questions (FAQ)", USENET: comp.ai.genetic. Available via anonymous FTP from rtfm.mit.edu:/pub/usenet/news.answers/ai-faq/genetic/
- [2] Clive Davidson, "Creatures from Primordial Silicon" New Scientist, 15 November 1997, pp 30-34
- [3] Peter Coveney and Roger Highfield "Frontiers of Complexity" Faber and Faber (England), Ballantine Books (USA) 1995
- [4] Edward de Bono, "Po -- Beyond Yes and No", Harmondsworth Penguin, 1973
- [5] Gass, S. L. "Linear Programming" 4th Edition McGraw Hill 1975
- [6] Danzig, G.B. "Linear Programming and Extensions" Princeton University Press 1963
- [7] Ralston, Anthony (Ed) "Encyclopedia of Science and Engineering" Van Norstrand Reinhold 1983
- [8] Casti, John L. "Five Golden Rules: Great Theories of 20th Century Mathematics" J. Wiley and Sons 1996
- [9] Gershenfeld, Neil "The nature of Mathematical Modelling" Cambridge University Press 1999
- [10] M. C. Forman, A. Aggoun and M. McCormick; "Simulated Annealing for Optimisation and Characterisation of Quantisation Parameters in Integral 3D Image Compression" Second IMA Conference on Image Processing: Mathematical Methods, Algorithms and Applications (forthcoming publication). 1998
- [11] Kirkpatrick, S. Gerlatt Jr, C.D., Vecchi, M.P. "Optimization by Simulated Annealing" Science 220 671-680 1983
- [12] Kirkpatrick, S. "Simulated Annealing: Quantitative Studies" 1984
- [13] John H Holland, "Genetic Algorithms" Scientific American, July 1992, pp. 44-50
- [14] Rick L. Riolo, "The Amateur Scientist: Survival of the Fittest Bits" Scientific American, July 1992, pp 89-91
- [15] S. Ramo, J. R. Whinney and T. Van Duzer "Fields and Waves in Communication Electronics", Third Edition, J. Wiley and Sons, New York, 1994.
- [16] J.D. Foley, A. van Dam, "Fundamentals of Interactive Computer Graphics" Addison-Wesley, 1982
- [17] Poynton, Charles "Frequently-Asked Questions about Color" <http://www.inforamp.net/%7Epoynnton/ColorFAQ.html>

Authors Biographies



Hugh Sasse graduated from The University of York in 1985, and since then has been working in the Machine Vision Group and later the Applied Electromagnetics Group at De Montfort University, Leicester. He embarked on part-time study for MPhil/PhD in 1999. His main research interests are computer modelling of novel antenna structures

and other communication channel components, with an emphasis on optimisation of such structures.

Mohammed M. Al-Asadi, Alistair P. Duffy, Kenneth G. Hodge and Arthur J. Willis. For photos and biographies see "Return Loss Prediction For Cascaded Systems", paper 17-04 in these proceedings.