THE FULLY AUTOMATED OPTIMIZATION OF A MICROWAVE HPA USING SIGNAL INJECTION AND DIFFERENTIAL EVOLUTION

Alex Scarbro BEng. (Hons) - Electronic Design Manager - Linwave Technology Ltd.

Abstract

This paper will discuss how the manual optimization of a high power amplifier was replaced with a fully automated ATE. The rationale for selecting Differential Evolution will be demonstrated and suitable objective (cost) functions explained. The development of a highly re-usable LabVIEW based global optimizer will be shown and its practical applications in "real-world" problems demonstrated.

Introduction

When operating a multi-octave high power amplifier at or close to saturation it becomes highly non-linear. This results in a significant increase in the level of harmonic products. In some extreme cases, these harmonics can begin to "steal" power from the fundamental carrier, since they can fall in-band.

A well understood approach to counter this effect is to terminate and ultimately cancel these harmonics in some way. It should be noted that this technique of cancellation is only really suitable for systems where the fundamental power is fixed. If the fundamental power varies then the amplitude of the harmonics generated in the amplifier will also vary. This results in the harmonics becoming vastly more difficult to effectively cancel.

When an amplifier is only operating over a relatively narrow range of frequencies, passive components can be connected to its output to introduce the correct canceling termination. However, when operating over a wider range of frequencies these terminations must be adjusted with respect to frequency to achieve the greatest recovery of fundamental power.

This active adjustment could be performed through a number of different approaches:

1. Varying the termination for each harmonic, electronically or mechanically, at the amplifiers output.
2. Artificially generating low power harmonic products that are injected into the amplifiers input.
3. Artificially generating high power harmonic products that are injected into the amplifiers output.

This adjustment is a simple process for a human to perform at a single frequency. In fact, we humans are rather good at exactly this kind of "tweaking" of controls for optimization purposes. However, in a production environment, over many frequencies and perhaps temperatures, this becomes a very time consuming exercise that would be best achieved through an Automated Test Equipment (ATE).

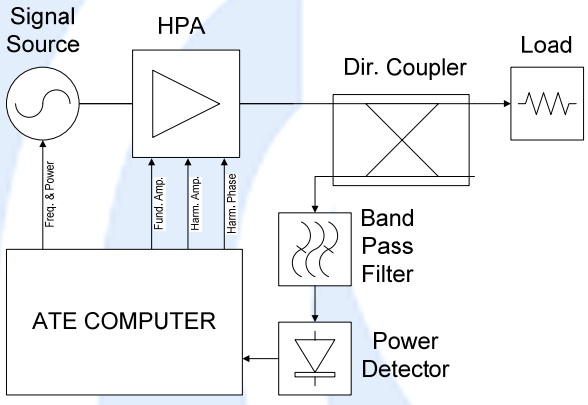


Figure 1. Test system configuration

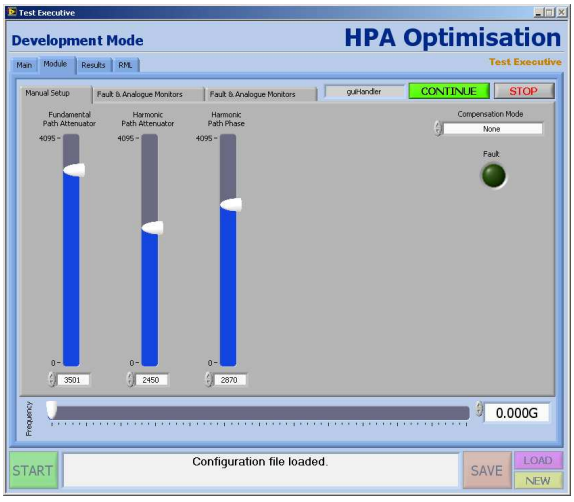


Figure 2. Manual amplifier optimization

Test System Configuration
Figure 1 shows our test system configuration. A microwave signal source, under computer control, generates a fundamental tone that drives the amplifier. The amplifier has the ability to control the amplitude of the incoming fundamental tone and the amplitude/phase of an injected 2nd harmonic tone by electronic means. The output of the amplifier is connected to a high power terminating load via a directional coupler. Between the coupled port and a power head is attached a band-pass filter. This ensures we only measure the power in the fundamental tone and not that of any spurious or harmonic products. The power meter reading is transferred to the computer via a high speed GPIB interface.

Manual Optimization
Figure 2 shows a manual interface for optimizing the fundamental and harmonic characteristics. This module was developed for a customer as part of an early generation ATE. This simple ATE allowed our customer to manually adjust the parameters and quantify the increase in power achievable across the amplifiers operating band.

Automatic Optimization
Our brief was to now develop an ATE that would autonomously search a multi-dimensional problem space. It must achieve maximum power in the fundamental tone at each frequency. It should do so as quickly as possible, whilst maintaining its reliability and repeatability. If the ATE were to incorrectly determine the optimal harmonic phase, for example, we could see partial or even total summation of the harmonics rather than cancellation. This is far from ideal and could result in a significant reduction of power in the fundamental tone.

During our previous attendance of the April 2009 ARMMS conference, we had the opportunity to review the work of Q. Lu et al. on the application of a Genetic Algorithm to a patch antenna optimization problem [1]. In this instance the patch dimensions (a, b) and the feed point position (X_p, Y_p) were optimized simultaneously to achieve the best possible return loss over a range of frequencies. Parallels between this and our own multi-dimensional problem led us to consider using a global optimizer as this might be a more suitable approach than attempting to develop our own custom algorithm.

Global Optimizer Overview
A global optimizer is a search heuristic that seeks out the best solution of an objective function over the whole search space. This is in contrast to a local optimizer which is used to find and solve the nearest solution.

For the optimizer to be truly generic, it should be completely independent of the problem and only view our system as a black box with no additional "knowledge". In numerical optimization, this black box is referred to as an objective function. The objective function would typically contain a mathematical expression that requires solving and would also derive a cost from the difference between the wanted result and the calculated result.

Objective function $\mathbf{F(x) = F(x_0, x_1, x_2)}$

Where: $\mathbf{x_0, x_1, x_2}$ are our parameters to be optimized.
F(x) is the cost of applying those parameters to our system.

In our problem, the objective function will have three inputs (fundamental amplitude, harmonic amplitude and harmonic phase). When evaluated, the objective function will take the three input parameters and program the amplitude & phase adjusters. It will then calculate the "cost" of the parameters based on the output power measured from the amplifier. The higher the output power, the lower the cost.

Table 1. Our problem in optimization terms.

Parameter Quantization	All our parameters are essentially discrete due to the use of D/A converters to drive the amplitude and phase adjusters: • 12bit fundamental attenuator (0 to 4096 equals >30dB range) • 12bit harmonic attenuator (0 to 4096 equals >30dB range) • 12bit harmonic phase shifter (0 to 4096 equals >360 ° range)
Parameter Dependence	We have a parameter dependant objective function. Not all the parameters can be optimized independently.
Dimensionality	We have three variables, therefore this is a three dimensional problem.
Modality	There may be one or more optimal parameter set. Therefore, our problem is multi-modal.

Time dependency	The optimum will be stationary with respect to time, after the HPA has stabilized in temperature & output power.
Noise	The measured output power will not be entirely noise free due to quantization in the D/A converters, measurement quantization in the power meter and variation in the power head's detected voltage.
Constraints	The objective function has no constraints. The parameters, however, do have hard limits.
Differentiability	Our objective function is essentially a black box whereby given the three parameter values and a fixed RF input level we calculate a cost. The systems behavior is therefore not easily computable and certainly not differentiable.

The three algorithms we considered were a Genetic Algorithm (**GA**), Particle Swarm Optimization (**PSO**) and Differential Evolution (**DE**). None of these selected algorithms require our problem to be differentiable.

Genetic Algorithm Overview

GA's are derived from our own evolutionary processes. In a typical GA, a population of binary strings (called chromosomes), are used to encode possible solutions to an optimization problem. Table 2 shows a possible encoding of our candidate solutions.

The evolution usually begins with a population of randomly generated chromosomes. In each generation, the fitness of every chromosome in the population is evaluated. A number of chromosomes are randomly selected from the current population (based on their fitness), recombined and possibly randomly mutated, to form a new population. This happens in much the same way as our own chromosomes are spliced and mutated during conception. The new population is then used in the next evolution of the algorithm. It should be noted that the mutation processes are not annealed. This means although classic GA may find the optimum solution, the population as a whole will never converge on to a single solution.

Table 2. The complete chromosome string with a total of 36 bits encoding the candidate solution.

Fundamental Amplitude (Bits 0 – 11)	Harmonic Amplitude Bits (12 – 23)	Harmonic Phase (Bits 24-35)

Particle Swarm Optimization

PSO was inspired by the movement and interaction of swarming animals, developed by Kennedy and Eberhart in 1995 it was targeted at neural network training. PSO works by allowing particles to intelligently move through a search space. Each particle has a position (x_i), a velocity (v_i) and its own "best value found so far" (p_i) parameter. The globally "best value found so far" parameter (p_g) is also stored. Each particle has its velocity updated with the following rule, once per iteration:

Velocity Update Function $\mathbf{v_i = \omega v_i + \varphi_p r_p (p_i - x_i) + \varphi_g r_g (p_g - x_i)}$

Where: $\boldsymbol{\omega}$ is the inertial weight, encouraging the particle to maintain its present vector.
$\boldsymbol{\varphi_p}$ is the weighting factor that controls steering of the particle towards p_i.
$\mathbf{r_p}$ is a uniformly random number between the limits of 0 and 1.
$\boldsymbol{\varphi_g}$ is the weighting factor that controls steering of the particle towards p_g.
$\mathbf{r_g}$ is a uniformly random number between the limits of 0 and 1.

Adjusting the three weighting factors ($\boldsymbol{\omega}$,$\boldsymbol{\varphi_p}$ and $\boldsymbol{\varphi_g}$) allows PSO to be tuned to a particular problem. It should be noted that PSO is purely iterative and not an evolutionary strategy since each member of the population lives for the entire duration of the optimization process.

Figure 3 illustrates PSO operating on a multi-modal problem. As the optimization executes, those particles with the best value found so far, attract the other particles by way of p_g. After a number of iterations, the particles will swarm closer and closer together. It should be noted that convergence on the global optimum, as opposed to a local optimum, is not guaranteed due to the annealing nature of PSO (i.e. it can become trapped in local minima).

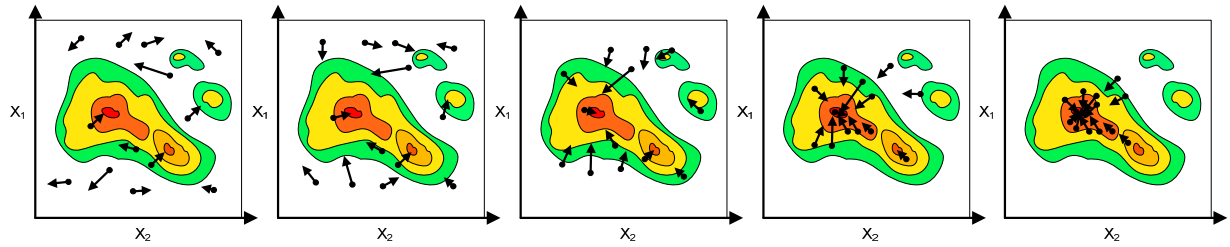


Figure 3. A sequence showing PSO converging.

Differential Evolution Overview

DE was first proposed by Price and Storn in a technical report in 1995 as a solution to the Chebyshev polynomial fitting problem. DE is exceptionally easy to implement, in any programming language and lends itself to parallel processing (if the objective function can also be evaluated in parallel).

The stages of evolving the population in DE are shown in Figure 4. As with GA and PSO, we begin with a randomly distributed population ($P_{x,g}$) of points covering the whole of the search space.

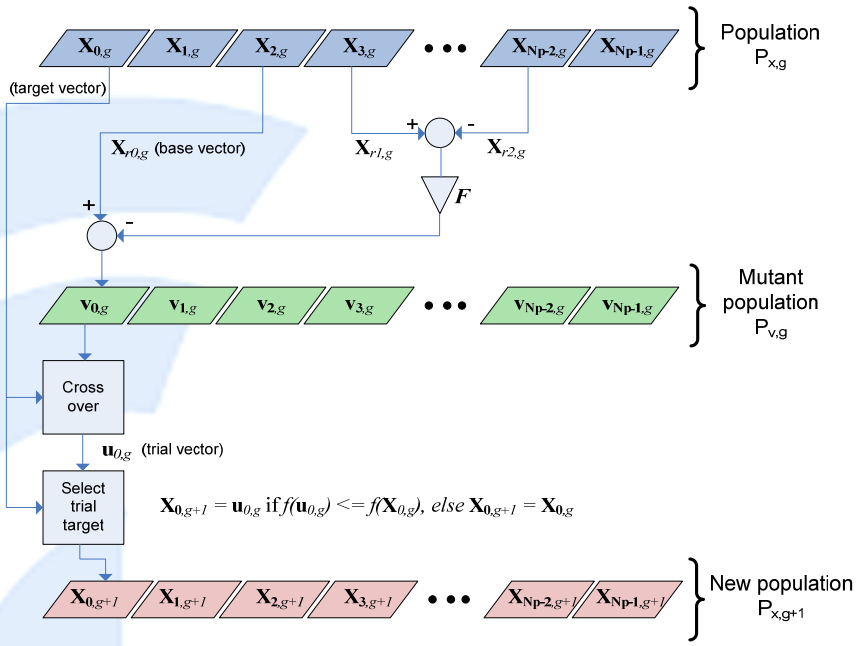


Figure 4. A flow chart of DE in operation

We then generate three random indices (r_0, $r_1$and r_2). These are used to index our population ($P_{x,g}$) and select three members ($X_{r0,g}$, $X_{r1,g}$ and $X_{r2,g}$). A difference vector is then calculated between $X_{r1,g}$ and $X_{r2,g}$ (See Figure 6). That difference vector is scaled down by the factor *F* and transposed on to member $X_{r0,g}$ creating a new member ($V_{0,g}$) in the mutant population ($P_{v,g}$). To increase DE's robustness, crossover is implemented based on the crossover constant (Cr).

Figure 5 illustrates the crossover process. For every dimension of candidate $v_{0,g}$ a uniformly random number is generated and compared to Cr. If the generated number is less than or equal to Cr, then the parameter of trial vector $u_{0,g}$ is inherited from the candidate $v_{0,g}$, otherwise the parameter is inherited from $X_{0,g}$. Finally, if the trial vector $u_{0,g}$ has a lower objective function value than $X_{0,g}$, then it replaces $X_{0,g}$ in the next generation of the population ($P_{x,g+1}$).

This is all that is required to implement DE.

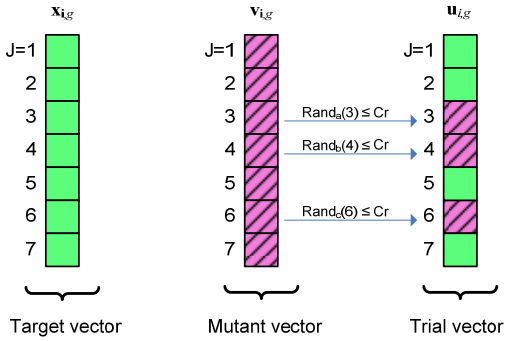


Figure 5. An illustration of the crossover process for a seven dimensional vector (D=7).

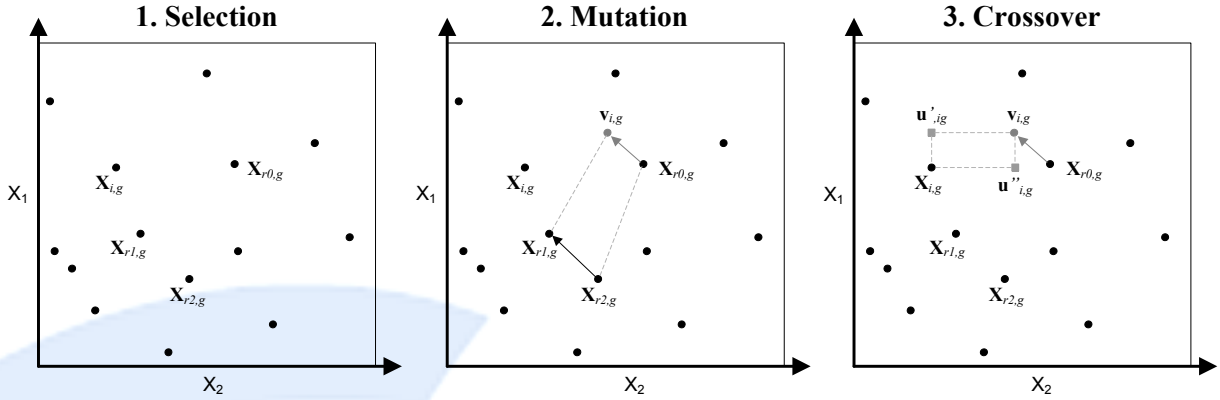


Figure 6. The three main steps of DE with crossover.

Termination

There are a number of conditions on which we can choose to terminate an optimization process. These may include the following common mechanisms:

1. A maximum number of generations (or iterations) have been reached.
2. A satisfactory fitness level has been reached.
3. A maximum amount of time has elapsed.
4. A statistic of the population has reached a satisfactory level. (E.g. the population standard deviation).
5. User intervention.

Global Optimizer Selection

Due to commercial pressures it was decided that rather than having to implement and test each algorithm with our system, we should establish whether the performance of these three algorithms (GA, PSO and DE) had already been compared. This led us to the excellent video lecture of Thiemo Krink (University of Aarhus, Denmark) [2]. In the video Thiemo provides us with a basic overview of the three algorithms and also demonstrates their relative performance using the two popular benchmarking functions shown in Figure 7 and 8. We have summarized our findings from Thiemo's video in Table 3.

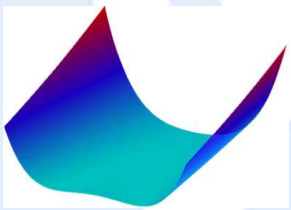


Figure 7. Rosenbrock function.

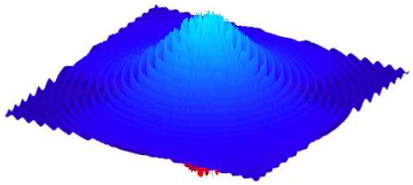


Figure 8. Schaffer F6 function.

Table 3. Global optimizer summary.

Type	Summary
GA	• Inherently noisy - it never truly converges due to continued mutation. This can lead to unreliability in some unique problems such as the Schaffer F6 test function [3]. • Can only encode binary (fixed-point) solutions without a more complicated crossover process. • Requires lots of parameter tuning for best performance.
PSO	• May converge on a sub-optimal solution. • Can handle floating-point candidate solutions. • Requires lots of parameter tuning for best performance.
DE	• Very easy to implement. • Very good efficiency. • Can handle floating-point candidate solutions. • Very reliable with very little parameter tuning.

To quote Thiemo "Differential Evolution is an incredible algorithm for numerical optimization". Based on the conclusions drawn in Table 4, we determined that DE should be our algorithm of choice.

Existing DE Implementations

Having undertaken extensive research to source expert knowledge of DE, we discovered "Differential Evolution – A Practical Approach to Global Optimization" (K Price, R. Storn and J. Lampinen) [4]. This book is a comprehensive guide to implementing DE and applying it to real world optimization problems. It comes highly recommended. Storn also maintains a website on which DE source code can be downloaded in a dozen different programming languages, including LabVIEW [5]. This virtual instrument package (VI in LabVIEW terminology) was created by Franz Josef Ahlers [6] and has proven to be an excellent starting point.

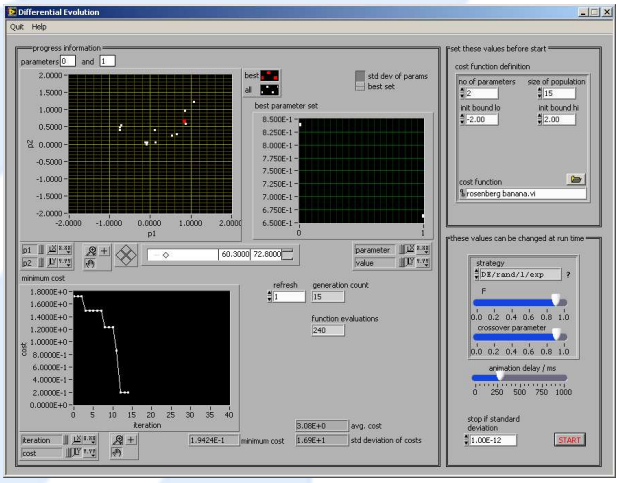


Figure 9. Franz's LabView DE program.

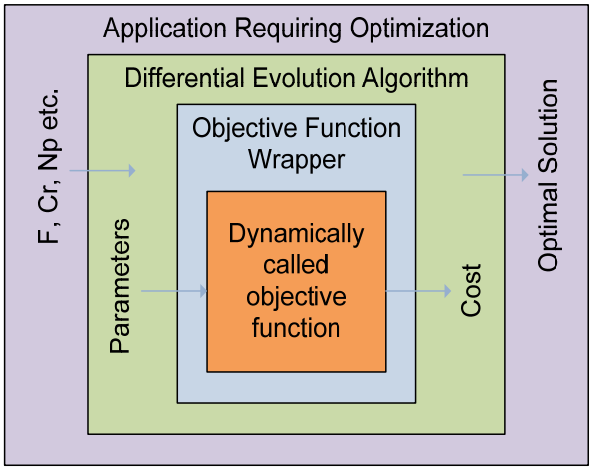


Figure 10. Franz's DE program structure

Franz's program allowed a number of purely numerical test objective functions to be optimized. These functions were implemented as dynamically callable "cost functions". Therefore the same optimization algorithm could be used to optimize different objective functions elsewhere in our ATE, without having to be modified.

The plots in Figure 9 show the population (with the best vector in red) and the cost versus generation number. There are also controls for the strategy, objective function selection and the optimization constants (*F* and Cr).

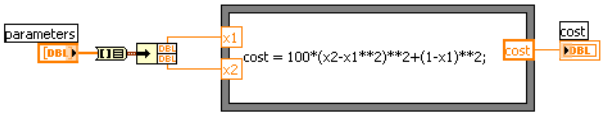


Figure 11. Franz's Rosenbrock test function.

Note the objective test function, in Figure 11, has only one input (an array of double precision-floating point controls containing the parameter values) and a single output (a double-precision floating point indicator containing the cost of those parameters). This is fine for purely numerical problems, but makes interacting with

systems in the outside world difficult. This means we would have to modify the algorithm each time we wish to communicate with a difference DUT or piece of test equipment.

Modeling

To gain some confidence in DE, with our particular optimization problem, it was decided we should create a model of our systems expected characteristics. This was achieved with a simple equation in LabVIEW and is plotted in Figure 7. The whiter the regions on the surface indicate higher fundamental output power.

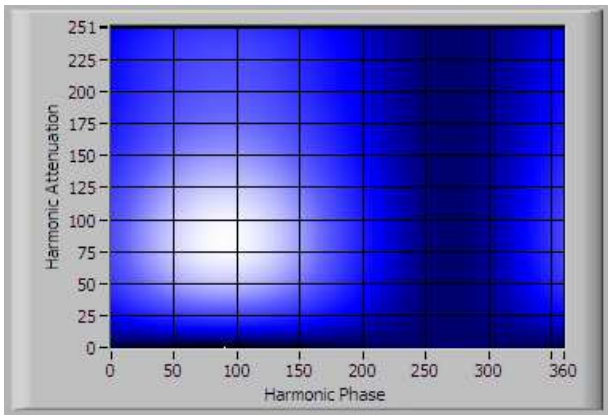


Figure 12. A model of our search space.

We expected the fundamental output power to peak at a particular harmonic phase (due to cancellation), then 180 degrees later we expect that power to be at a minimum (due to summation). We also expected that, at the optimal phase, the fundamental power would increase as we increase the harmonic level up until a point where we started to produce a higher level harmonic than that naturally generated by the amplifier. This modeled search space was integrated in to an objective function with the following relationship:

Objective function $\mathbf{F(x) = F(x_0, x_1) = 1 / L(x_0, x_1)}$

Where: $\mathbf{x_0, x_1}$ are the harmonic amplitude and harmonic phase parameters.
$\mathbf{L(x_0, x_1)}$ is the fundamental power achieved by applying those parameters to our system.

The objective function was evaluated with Franz's DE program with great success. We observed reliable convergence within 20 generations on the optimal solution, albeit without any quantization or measurement noise present in our system model.

ATE Implementation

Taking Franz's DE implementation as a starting point, a number of key changes were made to the optimizer:

1. A generic data array was flowed throughout the algorithm and into the objective function. This allowed information key to executing each unique objective function (such as GPIB connections, settling times, power meter offsets etc) to be different from one optimization task to another without having to modify the outer structure of the algorithm in any way.
2. Individual upper and lower limits were added for each parameter.
3. The mutation process was modified such that if a mutated candidate falls outside the parameters bounds, then a new set of r_0, r_1 and r_2 indices are calculated and the mutation was repeated.
4. The ability to initialize each vector based on a seeded distribution was added.

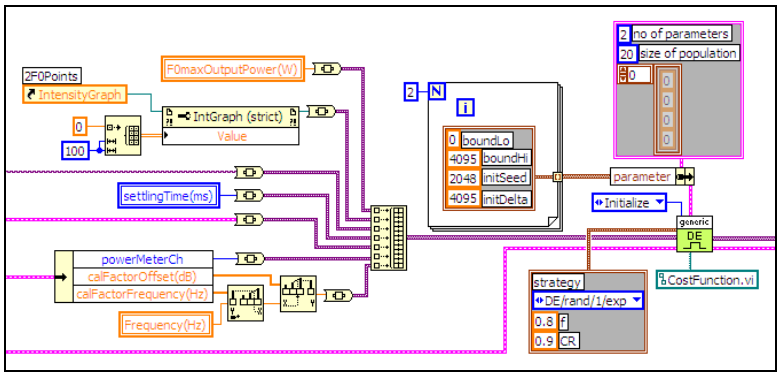


Figure 13. The initialization of our modified DE algorithm in LabVIEW.

Initializing the DE algorithm is a very simple process as seen in Figure 13. We set the parameter bounds, population size, mutation factor etc. We also "pack" the generic data types as discussed earlier. The DE algorithm can now be called with the "Evolve" option as many times as is required before reaching our terminating condition.

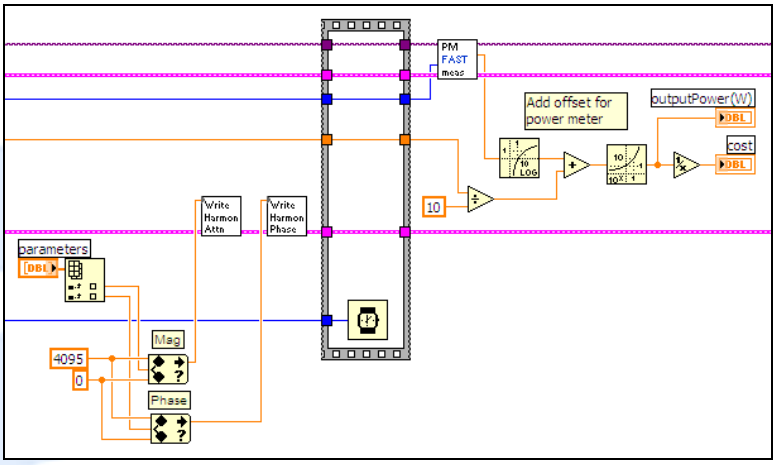


Figure 14. Our objective function in LabVIEW.

The objective function in Figure 14 shows the parameters being indexed and then passed to the harmonic amplitude and phase adjuster modules. After a delay, determined by the settling time, the power meter is instructed to perform a measurement. We operate the power meter, an Agilent E4418 with E-Series sensor, in 200 readings/second mode. This requires some careful configuration to achieve the best performance. Finally the returned measurement is calibrated by an offset factor and the corresponding cost calculated.

Real World Verification

It was decided that the fundamental and harmonic optimization should be split in to two phases (see Figure 15 and 16). This allowed us to firstly establish what peak fundamental output power could be achieved without the harmonic cancellation enabled. Then the harmonic cancellation would then be enabled, optimized and the peak fundamental output power measured again for comparison.

The search space shown in Figure 16 is colour coded with point at the red end of the spectrum indicating higher power when compared to points at the blue end of the spectrum. The points coloured white indicate the absolute maximum power observed so far in the optimization process. Note the similarity of the search space when compared to our original model (when rotated through 90°).

The higher density of points around the global maximum indicates the algorithm is converging on an optimal solution.

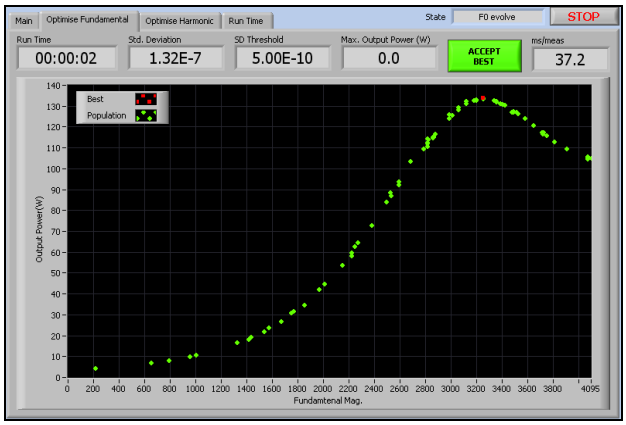


Figure 15. Our fundamental optimization.

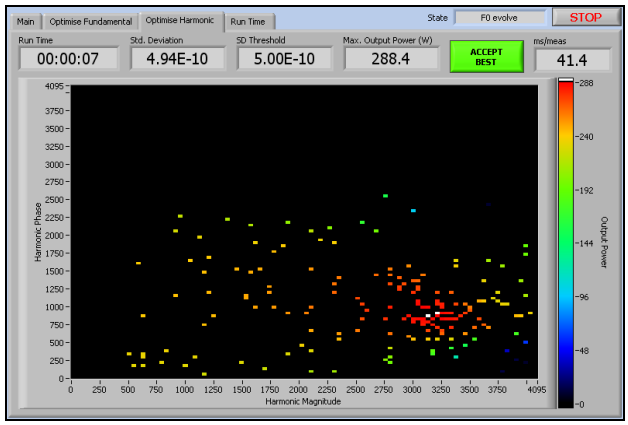
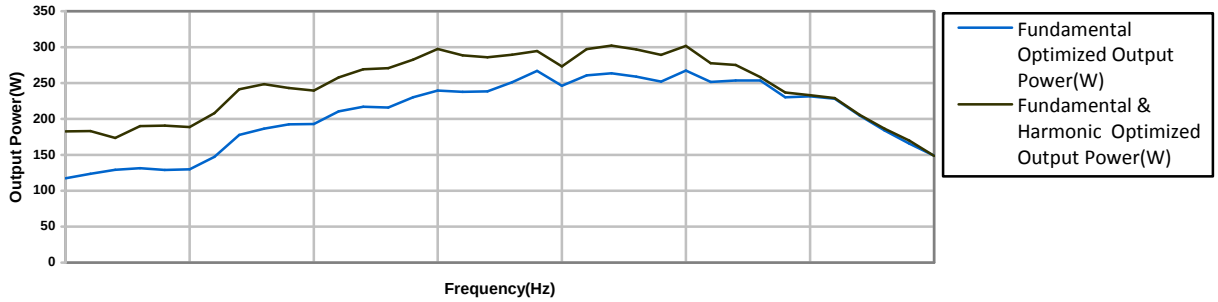


Figure 16. Our harmonic optimization.

The difference in output power, shown in Figure 17 is quite substantial with this particular HPA.

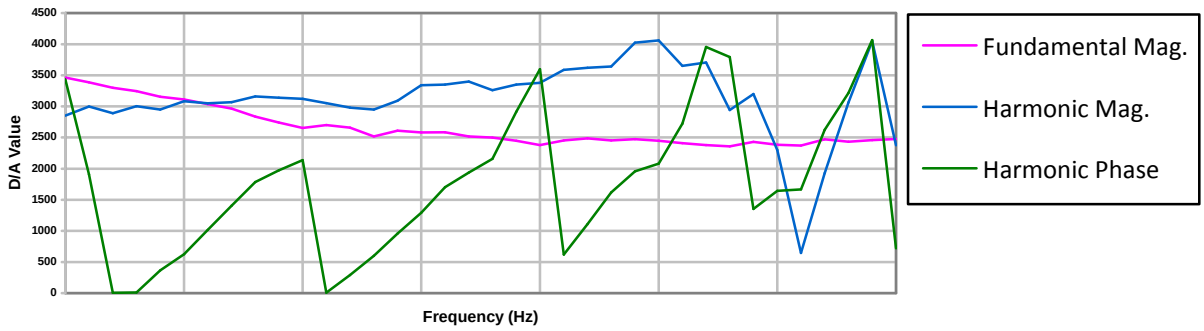
Figure 17. Optimized Output Power vs Frequency



The optimal values shown in Figure 18 are very promising. The smooth fundamental amplitude trace tracks the large signal gain characteristic we would expect for this type of HPA. The harmonic phase wraps cleanly too. If we could correlate the D/A values back to a real phase change, we suspect we would see another smooth characteristic versus frequency. Finally both the harmonic amplitude and phase seem to follow their respective trends right up to the point where the harmonic cancellation offers no real power increase. Beyond this frequency, the harmonic cancellation should be disabled in the final application.

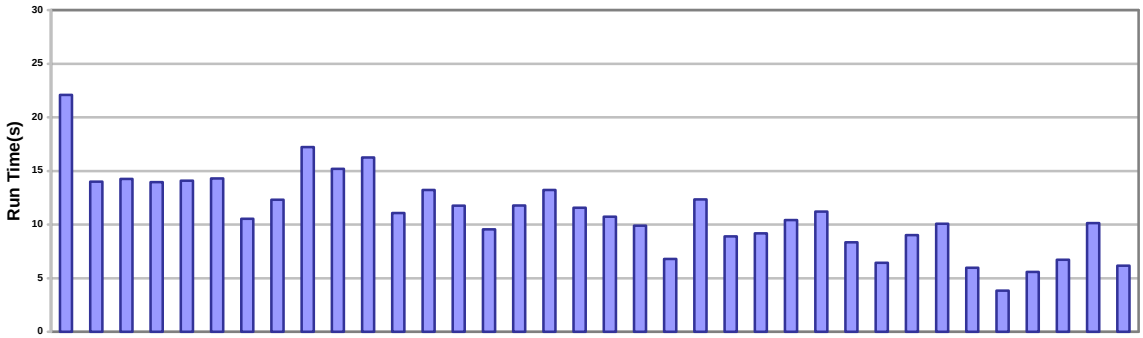
Taking in to consideration that fact that the ATE we have developed is completely automated, having it generate such smooth optimization characteristics is an excellent result when compared to the results generated by a human undertaking the same exercise. We have not only de-skilled the optimization process, we have also improved the repeatability and reliability when compared to the original manual method.

Figure 18. Optimized Fundamental & Harmonic Values vs Frequency



Finally the combined execution time for optimizing both the fundamental and harmonic parameters is shown in Figure 19. Given the steps involved in manually optimizing each parameter the automated run times are a welcome improvement (Particularly where there is parameter inter-dependence in the system).

Figure 19. Run Time vs Frequency



Conclusions

Our brief of creating a fully automated ATE for the purpose of amplifier optimization has been fulfilled.

The Differential Evolution global optimization algorithm has been developed in to a highly re-useable LabVIEW module. We have now used the same module in two other optimization tasks elsewhere in the ATE. In both cases, we needed only to develop a problem specific objective function.

Test results have been demonstrated showing favorable optimisation performance of a multi-octave HPA.

References

[1] Optimum Design of a Probe Fed Dual Frequency Patch Antenna Using Genetic Algorithm,
Q. Lu, E. Korolkiewicz, S. Danaher, Z. Ghassemlooy and A. Sambell.
http://soe.unn.ac.uk/ocr/papers/2009/QL-Oxford%20conf-09.pdf

[2] Differential Evolution and PSO in Partitional Clustering,
Thiemo Krink at the Josef Stefan Institute in Slovenia.
http://videolectures.net/solomon_krink_depso/

[3] Schaffer F6 Test Function.
http://zhanggw.wordpress.com/category/algorithm/geneticalgorithm/

[4] Differential Evolution: A Practical Approach to Global Optimisation,
K Price, R. Storn and J. Lampinen.
http://www.springer.com/computer/foundations/book/978-3-540-20950-8

[5] Differential Evolution (DE) for Continuous Function Optimization (an algorithm by K. Price and R. Storn).
http://www.icsi.berkeley.edu/~storn/code.html

[6] LabVIEW DE developed by Franz Josef Ahlers.
http://www.icsi.berkeley.edu/~storn/de_labview.zip