Proceedings of DETC'03
ASME 2003 Design Engineering Technical Conferences and
Computers and Information in Engineering Conference
Chicago, Illinois, USA, September 2-6, 2003

DETC2003/DAC-48803

Optimizing Optimization for Design Optimization

Petter Krus
Department of Mechanical Engineering
Linköping University
58183 Linköping, Sweden
petkr@ikp.liu.se

Johan Andersson
Department of Mechanical Engineering
Linköping University
58183 Linköping, Sweden
johan@ikp.liu.se

ABSTRACT

Design optimization is becoming and increasingly important tool for design. In order to have an impact on the product development process it must permeate all levels of the design in such a way that a holistic view is maintained through all stages of the design. One important area is in the case of optimization based on simulation, which generally requires non-gradient methods and as a consequence direct-search methods is a natural choice. The idea in this paper is to use the design optimization approach in the optimization algorithm itself in order to produce an efficient and robust optimization algorithm.

The result is a single performance index to measure the effectiveness of an optimization algorithm, and the COMPLEX-RF optimization algorithm, with optimized parameters.

INTRODUCTION

The rapid development in simulation methods and the general increase in hardware performance imply that design methods based on different kinds of numerical optimization for system design, are becoming increasingly important. Numerical optimization methods require that the object function is evaluated (using simulation) a large number of times, but they are very attractive since they can optimize complete non-linear systems and do not rely on grossly simplified models as more analytical methods do. Work in this area has shown that optimization can be used both for parameter optimization and for component sizing, see Krus, Jansson and Palmberg 93. Over the year a number of more or less advanced schemes for design optimization has evolved. There is a relative rich literature in design omtimization, see for example G.V. Reklaitis, A. Ravindran, K.M. Ragsdell ,1983, Papalambros, Douglass, Wilde 1988, or C Onwubiko 2000. In many cases however, a quick answer is preferred instead of a highly efficient method that takes more effort to set up. There are also situations where there is very little information regarding the nature of the object function, gradients can not be obtained explicitly and constraints are implicit. This is true when evaluation of the object function relays on simulation of dynamic systems. In these situations direct search methods are very attractive, and therefore the focus in this paper is on a specific direct search method, the COMPLEX-RF, which is a promising candidate for direct search optimization, and on optimization of such methods

SIMULATION AND OPTIMIZATION

Optimization based on simulation puts very high demands on the numerical efficiency and robustness of the simulation. Since a high number of simulations need to be done, typically ranging from a few hundred to tens of thousands, low simulation times are of course very important.

Another thing is that in simulation based optimization, parameters can vary substantially especially in the initial stages of the optimization. Some of these parameter sets can result in highly dynamic systems that would drive down the time step size of a variable time step method. This could result in very long simulation times, which would be wasted on solutions that usually are far from the optimum anyway. Therefore, it can be concluded that simulation based optimization benefits strongly from the deterministic simulation times obtained using fixed time steps. The time step may, however, be different in different subsystems instead.

Put together, these are strong case for the distributed modeling approach using bi-directional delay lines, se Auslander 1968 and Krus, Jansson, Palmberg and Weddfelt 1990, which is highly efficient, robust and normally is used with a fixed time step.

These requirements have resulted in the following strategy that is adopted for the development of the HOPSAN simulation package developed at Linköping University.

- Modeling on a detailed equation level using a symbolic math package to generate implementation
- The distributed modeling approach using bi-directional delay lines for partitioning of systems
- Different time step for different parts of the model

OPTIMIZATION

If a system model in the form of a simulation model is defined, it is possible to use optimization based on simulation. Using this method, the system is simulated using different sets of system parameters x_{sp}. From each system evaluation a set of system characteristics, y_{sc} are obtained and using these, an objective function, *f,* is formulated.

In general the simulation is used to obtain the performance characteristics of the system.

$$Y_s = F(X_{sp}) \tag{1}$$

The object function is a function of the system characteristics.

$$f_{obj} = G(Y_{sc}) \tag{2}$$

there may also be a violation flag, that indicates if implicit constraints are violated, that also is a function of system characteristics.

$$c_{viol} = C(Y_{sc}) \tag{3}$$

Another way to deal with constraints is to use a penalty function that is included in the objective function instead.

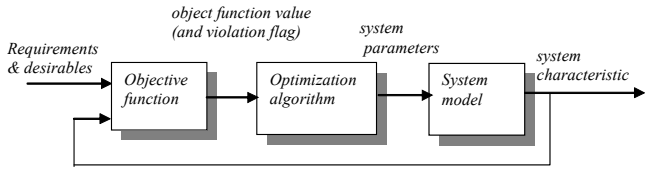

Figure 1. Optimization based on simulation.

In the general case there exist many explicit relations between parameters in the system. In fact in manual design, great efforts are made to obtain explicit design relations and there are many cases where system parameters are coupled and cannot be chosen independently from each other. It is therefore appropriate to define a layer of explicit design relations where relatively few independent optimization variables are expanded to the full set of system parameters.

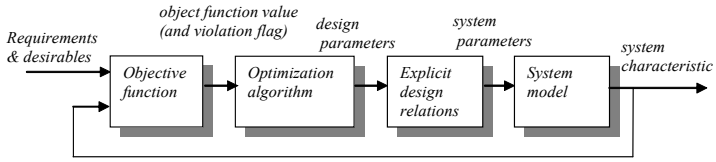

Figure 2. Optimization based on simulation with layer of explicit design relations.

The explicit design relations can be written as :

$$X_{sp} = R(X_{dp}) \tag{4}$$

Where X_{dp} is the vector of design parameters. The whole optimization problem can then be written as:

$$\text{Maximize } f_{obj} = G(F(R(X_{dp}))) \tag{5}$$

THE COMPLEX ALGORITHM

There are basically two families of optimization methods used in engineering. The gradient methods are widely used and are suitable for problems where the gradient of the object function can be calculated explicitly at each point. This is the case in many structure optimization applications. The other group is the non-gradient methods. These methods do not rely explicitly on gradient information in each point. These methods are, therefore, of more general use since gradient information is not generally available, especially if parts of the object function are evaluated using simulation of non-linear systems. The modified version of the original Complex method (by Box 1965) has been found to be one of the simplest and most easy to use methods, and has been used for system optimization of hydraulic systems See Krus et al 1993. The implementation shown here has also been described in Krus and Gunnarsson 93. This implementation of the Complex method is also used in some major Swedish companies.

The method can be used to maximize the function.

$$F(x_1, x_2, ..., x_N) \tag{6}$$

subjected to the constraints

$$g_i < x_i < h_i \tag{7}$$

where i = 1,2,....,M. The implicit variables $x_{N+1},...,x_M$ are dependent functions of $x_1,....x_N$. For design, $x_1,...,x_N$ are the design parameters X_{dp} and the dependent functions $x_{N+1},...,x_M$ are a subset of the vector of system characteristics Y_c. The constraints g_i and h_i are either constants or functions of $x_1,....x_N$. In the implementation used here, an initial complex of *m* points is generated. The variables at each point are generated using random numbers.

$$x_{ij} = g_i + r_{ij}(h_i - g_j) \tag{8}$$

Here j is an index that indicates a point in the Complex and i an index that indicates a variable. r_{ij} is a random number in the interval [0,1]. If the implicit constraints are not fulfilled, a new point is generated until the implicit constraints are fulfilled. The number of points m in the complex must be, such that $m >= N + 1$, where N is the number of independent variables

The object function is evaluated at each point. The point having the lowest value is replaced by a point reflected in the centroid of the remaining points by a factor α.

$$x_{ij}(new) = x_{ic} + \alpha(x_{ic} - x_{ij}(old)) \tag{9}$$

The centroid is calculated as

$$x_{ic} = \frac{1}{m-1}\left(\sum_{j=1}^{m} x_{ij} - x_{ij}(old)\right) \tag{10}$$

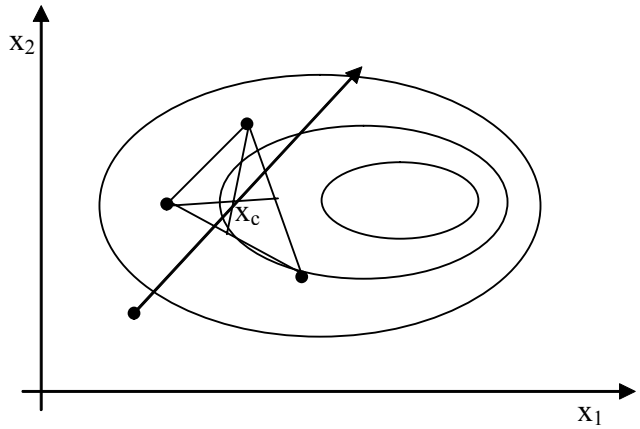

Figure 3. The Complex method. Reflection of the worst point through the centroid of the remaining points.

Box (1965) recommends α= 1.3. If a point repeats as the lowest value on consecutive trials, it is moved one half the distance towards the centroid of the remaining points. In this case:

$$x_{ij}(new') = x_{ic} + (x_{ic} - x_{ij}(new))/2 \tag{11}$$

The Complex-RF optimization method used here is a modified version of the Complex method by Box (1965). It is modified by introducing some randomization in the search. This avoids premature collapse of the method.

$$x_{ij}(new) = x_{ic} + \alpha(x_{ic} - x_{ij}(old)) + r \tag{12}$$

Here r is random noise in an interval, which is a fraction of the mean distribution of the parameter sets in the complex. Another modification involves what happens if a point repeats as the lowest value on consecutive trials. Instead of just moved halfway towards the centroid it is also mirrored in the centroid. This handles constraints or sharp edges in the object function better since it avoids a premature collapse on the edge.

$$x_{ij}(new') = x_{ic} - (x_{ic} - x_{ij}(new))/2 + r \tag{13}$$

Convergence rate

The maximum contraction speed of the Complex method in each object function evaluation can be estimated to (Krus and Gunnarsson 1993):

$$\frac{\Delta x_{k+1}}{\Delta x_k} = \left(\frac{\alpha}{2}\right)^{\frac{1}{2m}} \tag{14}$$

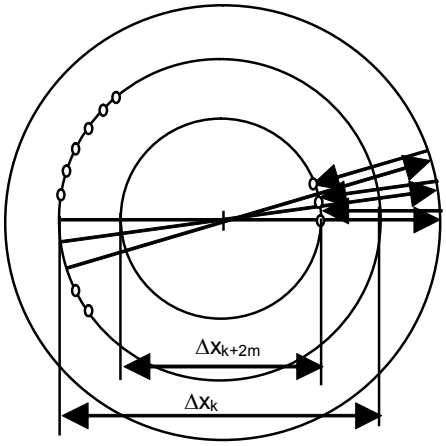

Figure 4. Contraction of the complex after *2m* steps. The optimum is in the middle.

The number of parameters in the complex m is function of the number of independent variables $m=\kappa n$ where typically $1.5<\kappa<2$.

here Δx(k) is the spread of the Complex parameter set at a particular evaluation no k. α is the reflection factor in the Complex method and this has normally the value 1.3, and n is the number of optimization variables (The number of parameter sets in the Complex method is set to typically two times the number of optimization parameters). Expressed as a function of the original spread Δx_0 the following expression is obtained.

$$\varepsilon = \frac{\Delta x_{k+1}}{\Delta x_0} = \left(\frac{\alpha}{2}\right)^{\frac{k}{2\kappa n}} \tag{15}$$

This means that the number of calculations needed to reduce the spread down to ε of the original spread can be estimated by:

$$k = -4.64271\, n \log(\varepsilon) \tag{16}$$

From this simple relationship follows that the number of evaluations to reduce a complex a certain amount is linearly dependent on the number of points in the Complex. In reality the objective function can be much more complicated than assumed here, but this estimate gives a lower bound to

the number of evaluations necessary and a fair description of the behavior near optimum.

An interesting aspect is to study the amount of information gained in each evaluation. In general the amount of information (in bits) to represent a value can be expressed as

$$I = Log_2 \frac{x}{\Delta x} = -S \tag{17}$$

where Δx is the uncertainty of the variable and x its nominal value. S is the entropy, and information I represents negentropy. Therefore the change in entropy, in each iteration, can be written as:

$$\Delta S = nLog_2 \left(\frac{\alpha}{2} \right)^{\frac{1}{2\kappa n}} \tag{18}$$

the multiplication with n comes from the fact that all n variables gain information. This can be simplified to:

$$\Delta S = Log_2 \left(\frac{\alpha}{2} \right)^{\frac{1}{2\kappa}} \tag{19}$$

with a=1.3 and k =2 yields

$$\Delta S = Log_2 \left(\frac{1.3}{2} \right)^{\frac{1}{4}} = -0.155 \tag{20}$$

This means that the system is gaining 0.155 bits of information at each evaluation (which may seem like a very small value). Note that this is independent of the number of optimization variables. However, for more optimization variables it takes longer time to converge since more information is needed. This represents an upper theoretical limit for the amount of information gained in each function evaluation. In reality even a benign object function gives a convergence rate several times lower than this.

It is also possible to include a forgetting factor, which ensures that the Complex is made up predominantly with recent parameter sets. This is necessary if the objective function varies over time. In that case old objective function values become increasingly unreliable and should be replaced by new ones. This is particularly true if the optimization is to be used to optimize parameters in a real process. In this case there may be drift in the parameters of the physical system. Introducing a forgetting factor has also been found to improve the success rate in other situations as well.

If the objective function is stationary but noisy, (that is there are local variations in the objective function between points close to each other in parameter space) this is indistinguishable from time dependent noise since the probability of having a parameter set return to exactly the same position (and thus the same objective function value) is very small.

Near the optimum the objective function is assumed to be (minimum problem):

$$f = f_{opt} - C(x - x_{opt})^2 \tag{21}$$

this can also be written as:

$$f_e = f_{opt} - f = C(x - x_{opt})^2 \tag{22}$$

Consider the objective function with the lowest value.

$$f_{e,\min} = f_{opt} - f_{\min} = C(x_{opt} - x_{\min})^2 \tag{23}$$

Assuming that

$$x_{opt} - x_{\min} = \Delta x \tag{24}$$

which is the spread of the parameter sets of the complex. This yields:

$$f_{e,\min}(k+1) = f_{e,\min}(k) \left(\frac{\alpha}{2} \right)^{\frac{1}{m}} \tag{25}$$

Subtracting a factor γ from the exponent yields

$$f_{e,\min,f}(k+1) = f_{e,\min}(k) \left(\frac{\alpha}{2} \right)^{\frac{1-\gamma}{m}} \tag{26}$$

This equation is convergent as long as γ<1. This can be rewritten as:

$$f_{e,\min,f}(k+1) = f_{e,\min}(k) \left(\frac{\alpha}{2} \right)^{\frac{1}{m}} - f_{e,\min}(k) \left(\frac{\alpha}{2} \right)^{\frac{1}{m}} \left(1 - \left(\frac{\alpha}{2} \right)^{\frac{-\gamma}{m}} \right) \tag{27}$$

This can also be written as

$$f_{e,\min}(k+1) = f_{e,\min,0}(k+1) - f_{e,\min,0}(k+1) \left(1 - \left(\frac{\alpha}{2} \right)^{\frac{-\gamma}{m}} \right) \tag{28}$$

or

$$f_{e,\min,f}(k+1) = f_{e,\min}(k+1) - f_{0e} \tag{29}$$

where

$$f_{0e} = \left(f_{\min}(k+1) - f_{opt}(k+1) \right) \left(1 - \left(\frac{\alpha}{2} \right)^{\frac{-\gamma}{m}} \right) \tag{30}$$

which can be approximated as

$$f_{0e} \approx \left(f_{\min}(k+1) - f_{\max}(k+1) \right) \left(1 - \left(\frac{\alpha}{2} \right)^{\frac{-\gamma}{m}} \right) \tag{31}$$

After each object value evaluation all stored object function evaluations in the complex are diminished with this value f_{0e}. Using this approach, an objective function value is gradually diminished, until it is replaced. The value of g can then be chosen to yields the desired properties. The proper selection will be discussed later.

Performance evaluation

In order to determine what parameter settings that are the best for the optimization algorithm some suitable test functions are needed that can test the ability of the optimization method to find the right optimum in a reasonable amount of function evaluations. One suitable test function is the so called "Rosenbrock's banana", see equation (36). This is a function that exhibit highly different gradients in different directions. This means that the search algorithm just finds the ridge but fails to detect the gradient along the ridge and consequently to proceed to the optimum.

$$f(x_1, x_2) = 100\left(x_1^2 - x_2\right)^2 + \left(1 - x_1\right)^2 \tag{32}$$

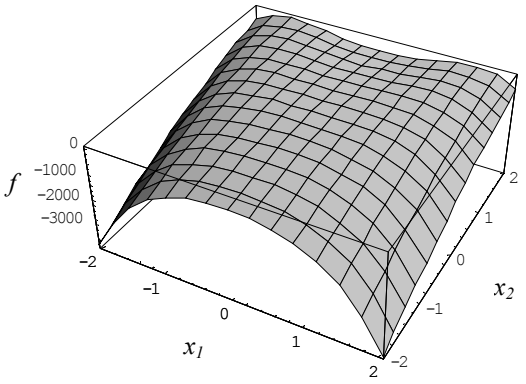

Figure 5. The test function Rosenbrock's banana.

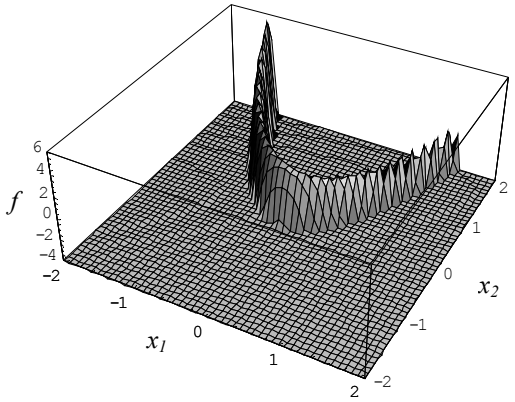

Figure 6. A close-up on the Rosenbrock's banana function between -4 and 6.

In order to investigate the effect on the performance of the algorithm depending on the reflection distance α, the system was optimized 100 times for each value of α. The success rate of the optimization could then be estimated by dividing the number of optimization runs that came within 1% of the true maximum value with the total number of optimization runs. The result is almost a step function at α=1, see Figure 7. In Figure 8 the required number of function evaluations is plotted as a function of the reflection factor α.

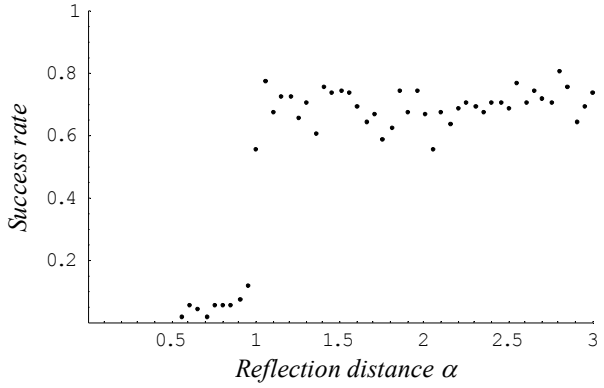

Figure 7. Success rate as a function of the reflection factor α.

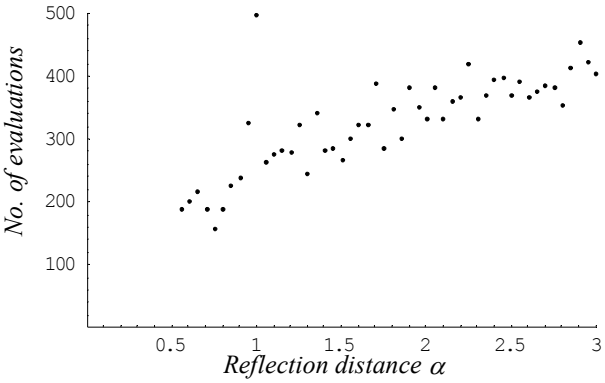

Figure 8. The number of function evaluations need for convergence as a function of α.

The conclusion from this test is that the value for α=1.3 which is recommended in the literature by Box 1965 seems to be a good choice although the behavior is not very sensitive to this value.

With the introduction of the randomization factor, *r,* and the forgetting factor, γ, it is also necessary to study the behavior for variations in these. Using α=1.3 the carpet plots in Figure 9 and Figure 10 are obtained, by conducting 300 optimizations (each taking anything from 100-1000 object function evaluations, se Figure 11) for each of the points in the carpet plot, (which total in a substantial number of object function evaluations). The hit rate shows a rather flat optimum starting already at small values of both *r* and γ, while the mean number of function evaluations shows a minimum for *r* values around 0.1. It does not seem to be much affected by the forgetting factor.

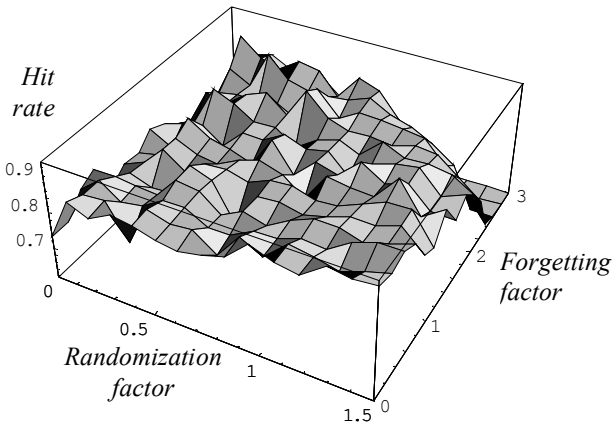

Figure 9.The probability of finding the true optimum (hit rate), as a function of randomization factor and forgetting factor.

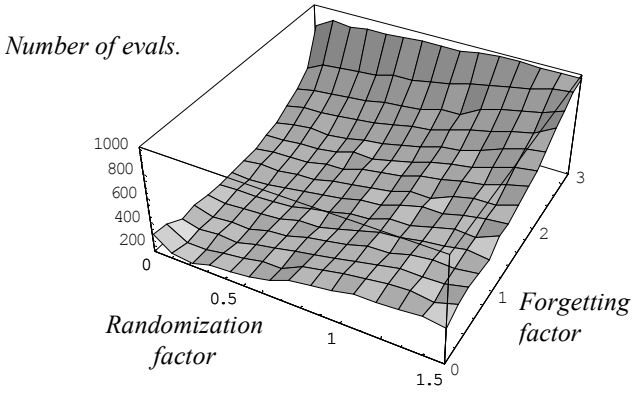

Figure 10. The mean number of object function evaluations needed to converge, as a function of randomization factor and forgetting factor.

For this function it seems rather obvious that a reasonable choice of the parameters would be: $r=0.15$ and $\gamma=0.15$.

Optimization of the optimization algorithm

However, when dealing with optimization it would be satisfying if these two functions could be combined into one object function. The object function should be a performance index of the method. The objective of any optimization algorithm is to gather information about the optimum solution. This can be expressed as reducing the uncertainty a certain amount and thus increasing the amount of information as in eq. (17). The cost is simply the amount of evaluations, k, needed to converge at the optimum. In this case the mean value, k_m, would be taken. The object function could be:

$$f'_{obj,opt} = \frac{1}{k_m} \tag{33}$$

In order to take the hit rate into consideration, this can be compared to a target probability for finding the right optimum. Let the hit rate for the optimization be *Popt* and the target hit rate be *P*. The risk of not finding the optimum is then $(1-P_{opt})$. However, if the optimization is repeated m times the risk of not finding the optimum in any of these is reduced and by solving the equation

$$(1 - P) = (1 - P_{opt})^m \tag{34}$$

The number of times needed to reduce the risk to $1-P$ can be calculated as:

$$m = \frac{Log_2(1-P_t)}{Log_2(1-P_{opt})} \tag{35}$$

Therefore the object function could be modified to

$$f'_{obj,opt} = \frac{1}{mk_m} = \frac{Log_2(1-P_{opt})}{Log_2(1-P_t)k_m} \tag{36}$$

since P_t is a fixed parameter and can be regarded as requirements. The object function could be reduced to:

$$f_{obj,opt} = \frac{Log_2(1-P_{opt})}{k_m} \tag{37}$$

Since the Log_2 is used (other bases could also be used) this represents one over the number of times it is required to evaluate the objective function in order to get a 50% probability of reaching optimum.

For some objective functions it is so simple to find the optimum that the probability of finding the right optimum is: $P=1$. Consequently m=1 in the previously defined meta-object function. For this case it is better to derive the information in the system from the uncertainty expressed in the entropy S defined in eq. 41. The uncertainty is a combination of the uncertainty in finding the correct optimum and the uncertainty in tolerance introduced by the stop criteria.

$$S = nLog_2((1-P_{opt}) + \frac{\Delta x}{x}) \tag{38}$$

The optimization criteria would then be to maximize the amount of information gained in average of each function evaluation. This can be written as:

$$f'_{obj,opt} = \frac{I}{k_m} = -\frac{n}{k_m} Log_2((1-P_{opt}) + \frac{\Delta x}{x}) \tag{39}$$

For the case where

$$(1-P_{opt}) >> \frac{\Delta x}{x} \tag{40}$$

Eq. (43) transforms into something equivalent to Eq. (40). If on the other hand the probability of finding the right optimum reached certainty. The equation approaches:

$$f'_{obj,opt} = \frac{I}{k_m} = \frac{n}{k_m} Log_2 \frac{\Delta x}{x} \tag{41}$$

Therefore eq. (42) is the function chosen as the meta object function for optimizing the optimizer.

When plotting this function for the same optimization runs as before it can be seen that there is ridge at a randomization factor r=0.15, see Figure 11. Furthermore, it can be seen that the optimal value for the forgetting factor lies around 0.2. However, the performance is not as sensitive to the value of the forgetting factor as it is to the value of the randomization factor. It is also interesting to compare the performance index (bits/function evaluation) with the theoretical upper level of 0.15. Here it is less than a tenth of that even for the best choice of parameters.

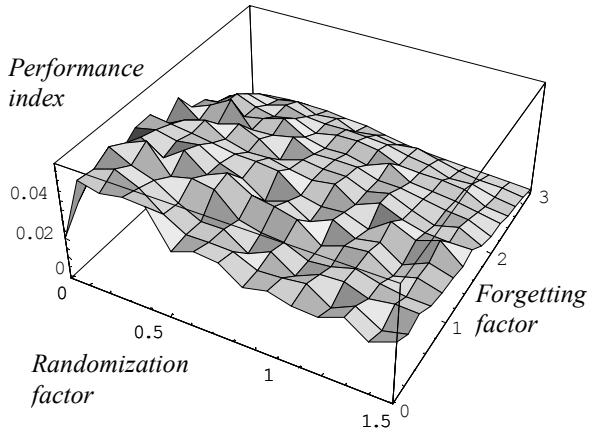

Figure 11. The performance metric of the method, as a function of the randomization factor and the forgetting factor for Rosenbrock´s banana.

In order to further study the influence of the randomization and forgetting factor, another test function is applied, see equation (42). This function is the step formed function shown Figure 12. This test will show how efficient the method is to handle integer problems.

$$f(x_1, x_2) = \left\lfloor \sin(x_1) \cdot \sin(x_2) \cdot x_1 \cdot n + 0.3 \right\rfloor / n \tag{42}$$

where n=*4* and $\lfloor a \rfloor$ denotes the integer part of a.

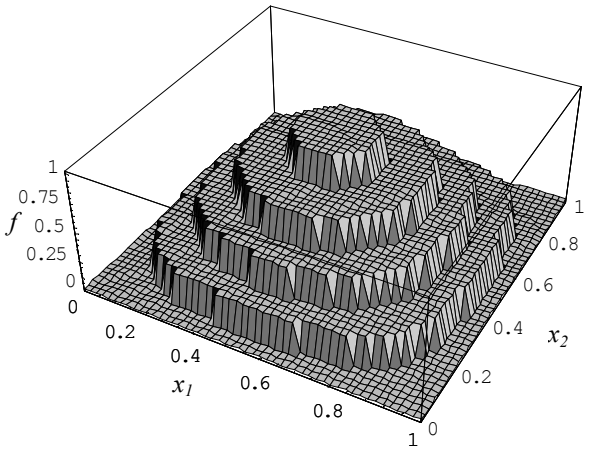

Figure 12. Test function that represents an integer problem.

This test function has been evaluated by conducting 200 optimizations each for different setting of the randomization and forgetting factors. The result is presented in Figure 13.

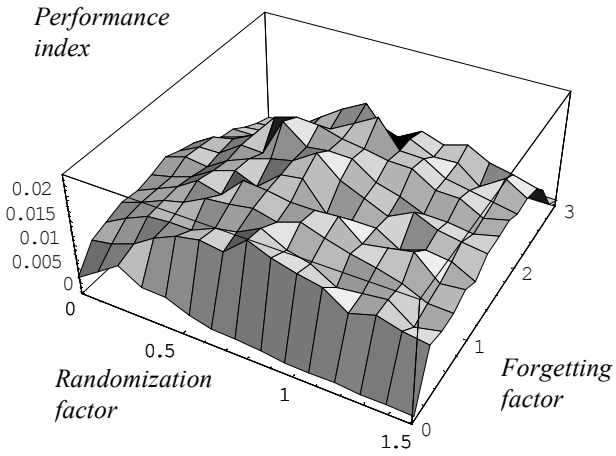

Figure 13. The performance index of the method, as a function of randomization factor and forgetting factor for the quantizesed hill function

As can be seen from Figure 13, optimal performance is obtained for a randomization factor around 0.2 and a forgetting factor as high as 1.0. Since the forgetting factor gradually reduces the value of old solutions the points in the complex will not have the same values even if they all are on the same plateau. Thus the complex will keep on moving and is therefore capable of jumping up to the next step. The forgetting factor could thus bee seen as an introduction of a virtual gradient of the objective function.

Finally a very simple test function was used. This is simply a continuous version of equation (42), which is simply a hill.

$$f(x_1, x_2) = \sin(x_1)\sin(x_2) \tag{43}$$

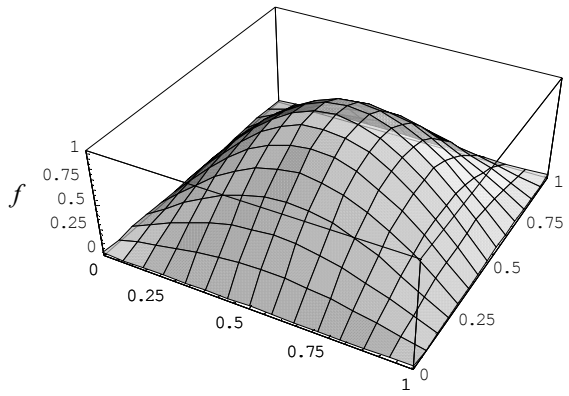

Figure 14. The continuous hill function

The performance index of this function is shown in figure 15. Evidently, for this simple function there is no use of having either the forgetting factor or the randomization factor, although it is tolerant to variations in these parameters. Interestingly the performance index of 0.2 is actually larger than the theoretically derived value of 0.155 (eq. 20). This can probably be explained by the fact that the parameters are evenly distributed within the parameter space, while the theoretical derivation assumed they all very located along the rim.

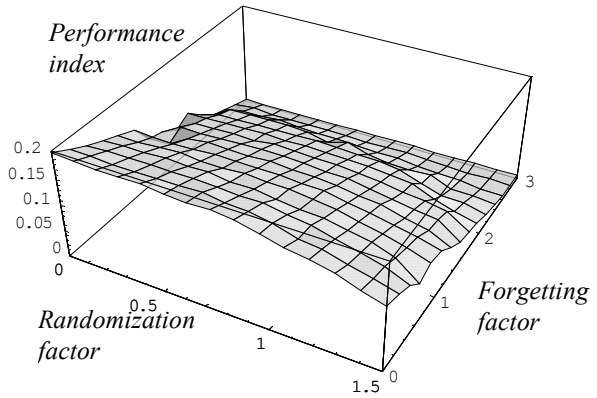

Figure 15. The performance index of the method as a function of randomization factor and the forgetting factor for the continuos hill function.

OPTIMIZATION OF THE COMPLEX METHOD

The next step is to use the COMPLEX-method for optimizing the optimization parameters in the COMPLEX-method. The chosen object function is the sum of the performance indices for the two test functions. Using optimization it is possible to optimize both the reflection factor α, the randomization factor r and the forgetting factor γ. Using eq. (35) as the test object function, an optimization of the optimization yielded the following values.

α =1.55
r =0.49
γ =0.32

With these values a preformance index of 0.082 was achieved for the Rosenbrock´s banana, which should be compared with the value for the original version of the algorithm was 0.025. The sensitivity in γ is very low for this test object function, since different runs gave values from 0.3 to 0.9. α and r are, however, more sensitive and converge within a few percent each time. It is interesting to note that the optimum value of α is much larger than the recommended 1.3, when used together with the other parameters. This optimization can of course be done for any of the above test functions or from a combination of them. As a first test, however, the Rosenbrock's banana seems to be a relatively representative function.

DESIGN EXAMPLE

A design problem studied is to develop a hydraulic actuation system that controls a fighter aircraft. The system model, depicted in Figure 16, is a simulation model consisting of a six degree of freedom model of a fighter aircraft and a model of its hydraulic actuation system. There is also a flight control unit. This could either represent an actual flight control unit or just a system needed to represent a pilot to fly the aircraft through the simulation. Even if the focus of this optimisation is not to design a flight control system, it needs to be included in the optimisation since different controllers may be needed for different actuation system parameters. There is also a simple engine model to represent the two engines in the aircraft. There is a more comprehensive description of this in Krus and Andersson 2003.

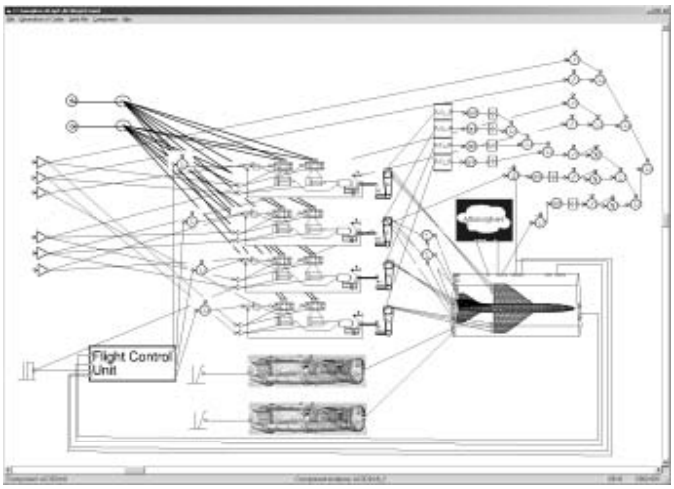

Figure 16. The Hopsan simulation model of the aircraft actuation system.

The explicit design relations

In this example there are many explicit design relations that can be used to reduce the number of optimisation variables x. the most obvious ones are symmetry relations. The symmetry requirement imposed means that there is a left-right symmetry in the control system, which means that many of the design variables are transformed into two system variables. Another useful mechanism for parameter reduction that also falls into this category is the use scaling. A component such as a servo valve has many design parameters but the driving requirements for a servo valve are usually only flow capacity and bandwidth (speed). This means that it can be assumed that most real valves can be described by only two performance parameters and in this case only one is used which is size. The pistons are also only described by one parameter, which is the piston area.

The objective function

The main objective is to produce an actuation system that can turn the aircraft as fast as it is possible while having as low weight as possible. That means that the components should have as small size as possible. In addition the pressure variations in the actuators are something that should be kept down in order to promote stable systems. In this example there are no constraints, except in the explicit design relations. The objective function can be written as

$$f_{obj} = -\left(\frac{Ie_{\varphi}}{Ie_{\varphi 0}} + \frac{Ie_{\theta}}{Ie_{\theta 0}} + \frac{Ip}{Ip_0} + \frac{m_s}{m_{0s}} \right) \tag{44}$$

Here Ie_{φ} is the integrated error in yaw angle, Ie_{θ} is the integrated error in tip. Ip is the sum of integrated pressure variations in all the actuators (high pass filtered to remove the DC component). Finally, m_s is the total weight of the actuator system. The coefficients in the denominator with the indexes 0 are used to normalise the different objectives and are obtained from one initial design. The optimisation algorithm is set up for finding maximum, hence the negative sign in front of the expression.

The gradients can not be obtained explicitly since a full non-linear simulation model is used. Therefore, a non-gradient method such as the COMPLEX-method seems a natural choice. The outcome of the optimization is shown in Figure 17 to Figure 19. Figure 17 shows the orientation angles for the optimized turn, whereas Figure 18 shows pressures in the hydraulic actuators during the turn. In Figure 19 the adjustment of the optimization parameters are shown as a function of the number of simulation runs.

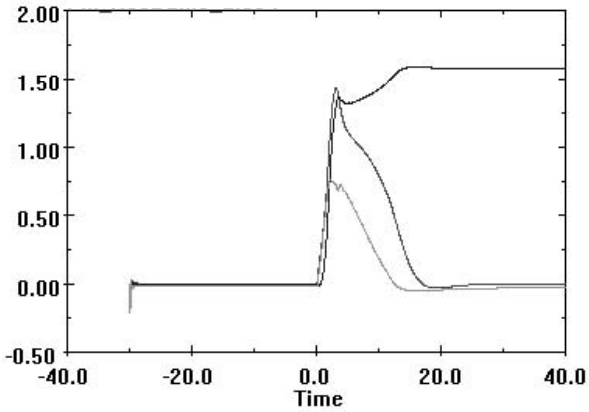

Figure 17. Orientation angles for the optimized turn.

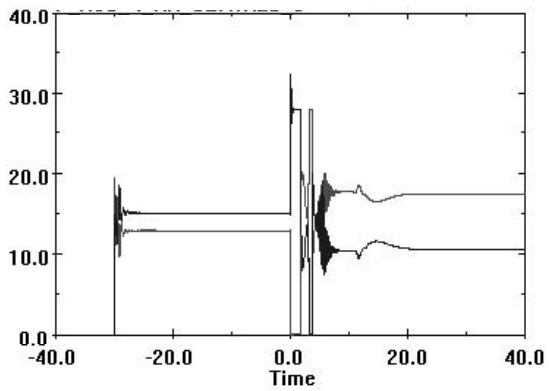

Figure 18. Pressure variation in the hydraulic actuators during manoeuvring.

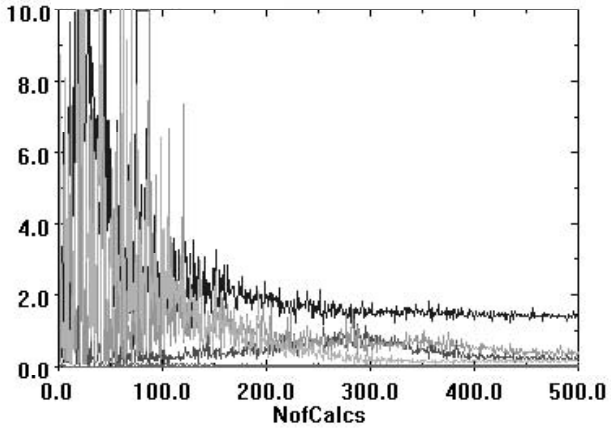

Figure 19. Convergence of the optimization parameters.

CONCLUSIONS

Optimization techniques are at the core of computational engineering design and optimization can be used on full-scale simulation models for system optimization. In many design problems the underlying models are of moderate computational intensity, and simple direct search methods with predictable performance are attractive to use.

In order to evaluate an optimization method a performance index is needed. In this paper, a simple expression, based on information theory, has been derived that considers the balance between computational cost, the probability of finding the optimal solution, and the tolerance of the found optimum. This index can be used both to tune parameters within the optimization algorithm, and to compare different optimization methods.

Here, the performance index has then been used for optimization of the parameters in the COMPLEX-RF algorithm for the optimization of a well known test function. The behaviour of the COMPLEX-RF method on other test functions where also investigated and give an indication that the optimized parameters seems reasonable although not strictly optimal for other object functions as well. Finally a more realistic complex system, based on a simulation model, was optimized using the algorithm with good result. It can therefore be concluded that the presented COMPLEX-RF method is a strong candidate for direct-search optimization.

REFERENCES

G.V. Reklaitis, A. Ravindran, K.M. Ragsdell. 'Engineering Optimization', John Wiley & Sons Inc. ISBN 0-471-05579-4. 1983.

P.Y. Papalambros, D.J. Wilde, 'Principles of Optimal Design', Cambridge University Press, ISBN 0-521-42362-7, 1988.

C. Onwubiko, 'Introduction to Engineering Design Optimization', Prentice Hall, ISBN 0-201-47673-8, 2000.

D.M. Auslander, 'Distributed System Simulation with Bilateral Delay-Line Models' *Journal of Basic Engineering*, Trans. ASME p195-p200, June 1968.

P Krus, K Weddfelt, J-O Palmberg, 'Fast Pipeline Models for Simulation of Hydraulic Systems'. Presented at ASME annual winter meet , Atlanta 1991. Also in *ASME Journal of Dynamic Systems, Measurement and Control*. March 1994.

M. J. Box. 'A new method of constrained optimization and a comparison with other methods'. *Computer Journal, 8:42--52,* 1965.

P Krus, A Jansson, J-O Palmberg, 'Optimization Using Simulation for Aircraft Hydraulic System Design', Proceedings of IMECH International Conference on Aircraft Hydraulics and Systems, London, UK, 1993

P Krus, S Gunnarsson, 'Numerical Optimization for Self Tuning Electrohydraulic Control Systems'. Procedings of JHPS International Symposium on Fluid Power, Tokyo, Japan, 1993.

P Krus, J Andersson, 'Simulation Based Optimisation for Aircraft Systems', SAE Aerospace Congress & Exhibition, Montreal, Canada, 2003.