Optim Eng (2015) 16:27–48
DOI 10.1007/s11081-014-9247-9

Optimization of the Complex-RFM optimization algorithm

Johan A. Persson · Johan Ölvander

Received: 5 April 2013 / Accepted: 4 January 2014 / Published online: 15 February 2014

Abstract This paper presents and compares different modifications made to the Complex-RF optimization algorithm with the aim of improving its performance for computationally expensive models with few variables. The modifications reduce the required number of objective function evaluations by creating and using surrogate models (SMs) of the objective function iteratively during the optimization process. The chosen SM type is a second order response surface. The performance of the modified algorithm is demonstrated for both analytical and engineering problems and compared with the performance of a number of existing algorithms. A meta-optimization of the algorithm is also performed to optimize its performance for arbitrary problems. To emphasize the fact that the modified algorithm uses meta-models it is denoted Complex-RFM.

Keywords Optimization · Surrogate models · Meta-optimization

1 Introduction

The desire in industry to perform optimizations of their systems is driving the need for efficient optimization algorithms. This is especially true in respect of computationally demanding models where one model call might take hours to perform. Consequently, the required computational time of an optimization may be significant since an optimization needs numerous model calls to converge. A popular solution to this problem is to involve computationally effective surrogate models (SMs), also known

J. A. Persson (✉) · J. Ölvander
Division of Machine Design, Linköping University, Linköping, Sweden
e-mail: johan.persson@liu.se

J. Ölvander
e-mail: johan.olvander@liu.se

as metamodels in earlier literature, in the optimization process (Duvigneau and Praveen 2007; Kitayama et al. 2011).

SMs are numerical approximations of how the output of a system or model varies when the parameters of the system or model are varied. If the SM is considered to be an accurate reanimation of the original model, it is beneficial to perform analyses on the SM instead of the original model for engineering applications, since the wall-clock time of one model call may be reduced to parts of seconds (Johnson et al. 2008).

There are two main approaches to use SMs to reduce the wall-clock time of an optimization process (Duvigneau and Praveen 2007). The first and probably the most popular method is to create a global SM, which has a desired accuracy over the whole design space, of the original model before the optimization is started (Park and Dang 2010). Then the optimization is performed on the SM and consequently the received optimum is the optimum of the SM. The other approach is to create and fit SMs iteratively during the optimization process.

The benefit when a global SM is fitted before the optimization is started is that the created SM may be used afterwards for other analyses of the system it reanimates. However, since the location of the optimum is often unknown before an optimization has been performed, the SM needs to be accurate over the whole domain (Duvigneau and Praveen 2007). It would be unfortunate to have a low accuracy in the parts of the design space where the optimum is located. The samples used to fit the SM therefore need to be spread over the design space, meaning that samples may be drawn in places that are not reached by the optimization algorithm.

One advantage when SMs are fitted iteratively during the optimization process is that the only model calls of the original model that are made are those that are needed for the progress of the optimization. Consequently, no unnecessary samples are drawn. The vicinity of the found optimum should also be reanimated accurately by the last SM since the samples that are used to fit the SMs are drawn closer and closer to the optimum. A drawback is that the first SMs will be coarser since the initial samples are few. This means that the SM may be a bad reanimation of the original model, which might make the optimization converge to a bad solution.

The purpose of the work is to modify an existing optimization algorithm to improve its performance for moderately computationally expensive models where an optimization takes around a day to perform. The aim of the proposed method is to make the algorithm more effective by using SMs iteratively during the optimization. The parameters of the algorithm are also optimized to optimize the performance of the algorithm for arbitrary problems.

Many different SMs exist and several attempts have been made to compare their performances for both analytical functions and engineering problems (Jin et al. 2001, 2003; Shan and Wang 2010; Tarkian et al. 2011). However, it is difficult to draw any general conclusion about which SM performs best (Wang and Shan 2007). Since the SM is updated iteratively during the optimization process, it is undesirable to wait for an optimization of the SM parameters each time the SM is updated. The SM type that is chosen should therefore be one where the SM parameters can be determined quickly. Polynomial response surfaces (PRSs) can be updated fast using least square estimations, which may be performed by simple matrix operations, and they also give accurate local estimations. For this reason, PRSs are chosen as SMs.

The chosen optimization algorithm, which is modified to use SMs iteratively during the design process, is the Complex-RF algorithm. The performance of Complex-RF has been demonstrated for engineering problems in several papers; see for example works by Andersson (2001), Johansson et al. (2002) and Krus and Ölvander (2012).

To emphasise that the modified algorithm creates and calls models during the optimization, it is denoted Complex-RFM, where the M stands for metamodels.

1.1 Paper outline

Section 2 describes PRSs and the proposed optimization algorithm is presented in Sect. 3. Section 4 briefly describes meta-optimization. Section 5 demonstrates the performance of Complex-RFM for a few analytical functions and an engineering example and is followed by the conclusions in Sect. 6.

2 Polynomial response surfaces

PRSs are SMs where the value of a new point is estimated from a mathematical function of a polynomial form. This can be seen in Eq. 1.

$$\hat{y} = \beta_0 + \sum_{i=1}^{N} \beta_i x_i + \sum_{i=1}^{N} \sum_{j=1}^{N} \beta_{ij} x_i x_j. \tag{1}$$

This can also be written in matrix form, as shown in Eq. 2

$$\hat{y} = \mathbf{X_v} \beta, \tag{2}$$

where

$$\mathbf{X_v} = [1\ x_1\ x_2 \cdots x_N\ x_1x_1\ x_1x_2 \cdots\ x_Nx_N],$$
$$\beta = [\beta_0\ \beta_1\ \beta_2\ \cdots \beta_N\ \beta_{11}\ \beta_{12} \cdots\ \beta_{NN}^T].$$

x_i stands for the value of the ith variable of the design point which is evaluated. The coefficients β_i need to be determined in order to use this SM. This can be done effectively by performing a least square estimation. It is possible to perform a least square estimation by creating a matrix system as shown in Eq. 3. This matrix system consists of one row for each of the m samples, s_i, that are used to fit the response surface. Each row is a realization of Eq. 1 for one sample.

$$\mathbf{X}\beta = \mathbf{y}, \tag{3}$$

where

$$\mathbf{X} = \begin{bmatrix} \mathbf{X_v}(s_1) \\ \mathbf{X_v}(s_2) \\ \vdots \\ \mathbf{X_v}(s_m) \end{bmatrix} \mathbf{y} = y(s_1)\ y(s_2) \cdots y(s_m)^T.$$

This equation system is solved in a least square sense by performing the operations seen in Eq. 4. A least square estimation finds a solution which minimizes the errors squared for Eq. 3.

$$\hat{\beta} = \mathbf{X}^t\mathbf{X}^{-1}\mathbf{X}^t\mathbf{y}. \tag{4}$$

It is necessary to use as at least as many samples as there are coefficients to fit a PRS, but it is recommended to use an oversampling of 50 % (Redhe et al. 2002). This is a drawback when PRSs are used for problems with many variables, since the combination terms scale with the number of variables squared. It is desirable to use as high a degree of polynomial as possible to capture the behaviour of the problem as well as possible. But a high degree of the polynomial means that more samples are needed to fit the PRS. A second degree polynomial is probably sufficient to give accurate estimations around the optimum and is therefore a good trade-off between accuracy and efficiency.

3 Optimization algorithms

This section presents the Complex-RFM algorithm, but begins by describing the Complex-RF algorithm that it is based on. The problems that are solved by these algorithms to demonstrate the performance of Complex-RFM are also solved by two additional algorithms—a sequential approximate optimization (SAO) based on a genetic algorithm (GA) and sequential quadratic programming (SQP).

3.1 Complex-RF

The Complex algorithm was proposed by Box (1965) and is a modified version of the Nelder–Mead Simplex algorithm (1965). A flowchart of the algorithm is shown in Fig. 1.

Complex-RF is a non-gradient based algorithm which starts by placing k points randomly in the design space (Krus and Ölvander 2003). These k points span the Complex. The objective function values of the k points are evaluated by calling the original model and the worst point is identified. This point is reflected through the centroid of the other points. The objective function value at the new location is evaluated by calling the original model and comparing it with the objective function values of the other points. If the new point is still worse than the other points, it is moved along the imaginary line halfway towards the centroid of the other points until it is no longer worst. This is shown at the bottom of Fig. 1.

The algorithm progresses by moving one point at a time until convergence or the maximum number of objective function evaluations is reached. Since the starting points are spread randomly in the design space, the Complex algorithm is suitable for problems which are sensitive to the choice of starting point. Additionally, it is applicable to a wide range of problems since it does not rely on a linear or differentiable objective function.

The R in Complex-RF stands for randomization factor and is used when a point is moved (Krus and Ölvander 2003). Instead of restricting the point to move along the

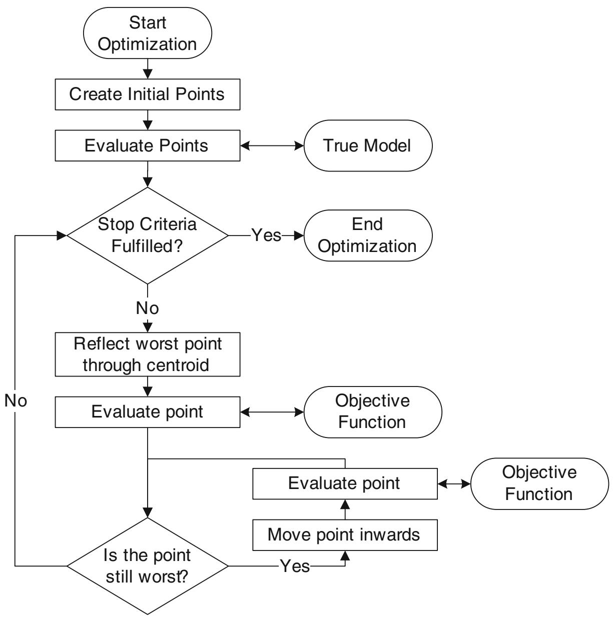

Fig. 1 A schematic flowchart for the Complex-RF optimization algorithm

imaginary line, it is placed in a random location close to the line. A higher randomization factor means that the point can be placed further away from the line. The *F* in Complex-RF denotes forgetting factor and is used to ensure that newer points are preferred in the Complex (Krus and Ölvander 2003). This is useful if the objective function varies with time or has local plateaus where the algorithm might get stuck.

3.2 Complex-RFM

The proposed modification of the Complex algorithm uses PRSs as SMs throughout the optimization and its flowchart is shown in Fig. 2. The algorithm performs the same operations as Complex-RF until enough samples have been drawn. This number corresponds to an oversampling of 50 % for the second order response surface. When the function "Original Model / SM" in Fig. 2 makes that model call, it fits a PRS to the objective function values of the model calls. During the remainder of the optimization, a PRS is called whenever the algorithm wants to determine the objective function value. A pseudo code for the Original Model/SM function is found in Annex 1.

This could be interpreted such that the original model will never be called after the first PRS is created. However, it is desirable that the SM should be called whenever its result can be trusted and the original model should otherwise be called. But how should this be determined?

For this algorithm, it is handled by the function "SM-Check", whose pseudo code can be found in Annex 2. This function is executed at the end of each iteration.

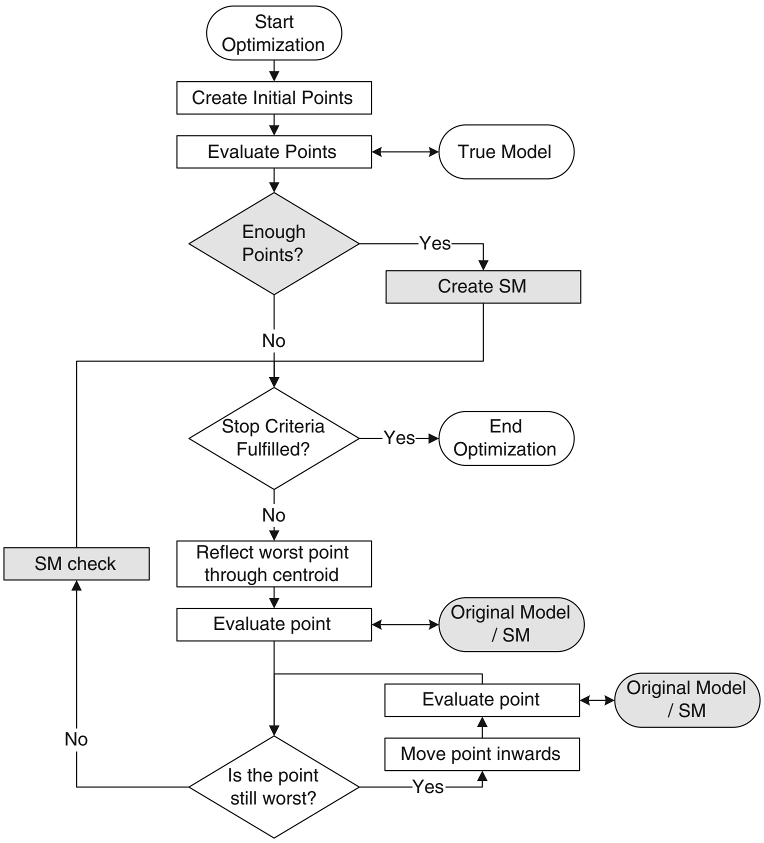

Fig. 2 A schematic flowchart for the Complex-RFM optimization algorithm

If an SM was used to estimate the objective function value of the latest point, the characteristics of the used PRS are investigated. If the accuracy of the PRS is better than a certain threshold, ε, the quality of the SM is considered good enough and the function "SM-check" is omitted for a number of iterations. Both the threshold and the number of omitted iterations are parameters which may be optimized in order to maximize the performance of the algorithm as described in Sect. 4.

It is possible to estimate the accuracy of the SM by comparing its estimation of a point with the value calculated from the original model. But this information may be redundant since it is tempting to incorporate the value of the new point into a new SM. Since the new point is placed in a region of the design space where the optimization algorithm is currently operating, a new SM which takes the point into account is preferred. Consequently, a new SM is used by the optimization algorithm as soon as a call of the original model is made and therefore the estimated accuracy will be the accuracy of the previous model.

One approach is to compare the two latest SMs and analyse how much they differ. If they agree well, it is probable that the SM is accurate in the area of the design space which the optimization algorithm currently operates in. The determination of how well the two latest SMs agree may be performed in several ways, but here the focus is to compare the coefficients, β_i, of the two latest response surfaces. For this algorithm, two response surfaces are deemed to be equal if the maximum percentage difference between the values of their coefficients is below a certain threshold. To increase the accuracy of the SM in the area in which the optimization works, the SMs are fitted by using only the latest samples, corresponding to an oversampling of 50 %.

3.3 SQP: fmincon

SQP is an iterative optimization method which uses gradient based methods to converge (Bonnans et al. 2006; Nocedal and Wright 2006). The SQP implementation that is used is the function fmincon from the software package MATLAB. Unlike the Complex algorithm, a starting point must be supplied by the user and fmincon will return the same suggested optimum every time an optimization of a problem is performed with the same starting point. The need for a starting point and the gradient based solving procedure make fmincon sensitive to the choice of starting point for multimodal problems.

Different algorithms are used by fmincon depending on the settings used when the functions are called. Here, fmincon is used with its standard settings and no gradients are supplied when the objective functions are called. Instead, fmincon uses finite differences to estimate the gradients.

3.4 SAO based on a GA

A GA is an optimization algorithm which reanimates nature's way of evolution (Goldberg and Holland 1988; Holland 1975). It starts by creating an initial population with parameter values selected randomly in the design space.

The objective function value of each individual in the population is then calculated, and the further evolution of the population is inspired by the Darwinian principle of survival of the fittest. Hence, the best individuals are chosen for reproduction, using genetic operators such as cross over and mutation.

It is possible to control the behaviour of the algorithm by changing the number of individuals in the population and the number of generations, as well as by modifying other factors such as the mutation and crossover operators. For example, if a lot of mutation is allowed, each individual may mutate and get dramatically changed parameter values, which encourage the search for the global optimum.

A drawback with GAs is that they usually require numerous objective function evaluations to converge since the fitness value of all individuals of all generations need to be estimated (Kitayama et al. 2011). This can however be remedied since it is possible to estimate the value of each individual in a generation in parallel. If many computers or cores are available it is therefore possible to reduce the wall clock time to the time it takes to simulate one individual multiplied by the number of generations needed.

An SAO begins by drawing some initial samples of the problem and fits an SM to the samples (Forrester et al. 2008; Kitayama et al. 2011). An optimization is then performed on the SM to find its optimal value. The original problem is simulated at the found optimum and this sample is used to update the SM. Another optimization is performed on the updated SM and once again the value of the original model at the found optimum is calculated by simulating the original model. The SM is then updated once again and this process continues until the same optimum has been found several times in a row or the maximum number of simulations of the original model is reached.

The SAO that is used to solve the following problems uses Kriging as SM. Kriging originates from geostatistics and is widely used as SM by the engineering community. The interested reader is encouraged to read the work by Isaaks and Srivastava (1989) for a good introduction to Kriging.

4 Meta-optimization

In order to maximize the performance of an optimization algorithm, its parameters can be tuned, so called meta-optimization (Grefenstette 1986; Krus and Ölvander 2012; Smit and Eiben 2009). An illustration of how to perform an optimization in order to optimize the parameters of the optimization algorithm is shown in Fig. 3.

To enable optimization of the parameters, an objective function must be chosen. The most interesting characteristics of an optimization algorithm for a computationally expensive model are the probability of finding the optimum, P, and the required number of model calls, n_{rec} (Krus and Ölvander 2012). The objective function should therefore be a combination of both properties. If the hit-rate is P, then the probability of not finding the optimum is $1 - P$. The probability of not finding the optimum for q optimizations could thus be expressed as $(1 - P)^q$ and consequently the probability of finding the optimum is calculated according to Eq. 5.

$$P_{found} = 1 - (1 - P)^q. \tag{5}$$

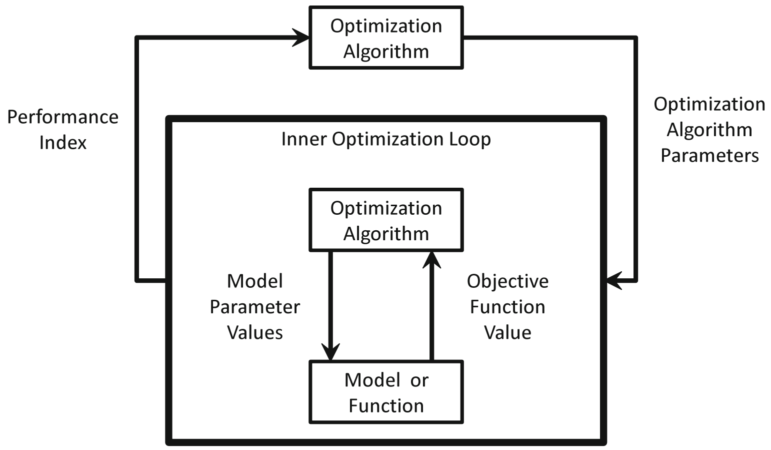

Fig. 3 A schematic flowchart for optimization of the parameters of an optimization algorithm

However, different optimization algorithms require different numbers of objective function calls to converge and consequently the exponent q in Eq. 5 should be replaced with the allowed number of model calls divided by the required number of model calls, n_{calls}. If the allowed number of model calls is set high, the probability of finding the optimum will be close to one for all algorithms. Here, the number is set to 100, as shown in Eq. 6, meaning that Eq. 6 can be interpreted as the probability of finding the optimum by performing 100 model calls. Naturally, it might be impossible to perform optimizations corresponding to exactly 100 model calls, but the purpose of the performance index is to enable comparison of the performance of different optimization algorithms and the difficulty of different problems.

$$Perf\,Index = 1 - (1 - P)^{100/n_{calls}}. \tag{6}$$

It can be noted that the performance index equals one for all algorithms where the probability of finding the global optimum is 100 %. Fortunately, this is no major issue since if two algorithms have the same hit-rate, the one which requires less model calls is the superior.

5 Performance demonstration and meta-optimization of Complex-RFM

The performance of Complex-RFM is demonstrated for two analytical problems and three engineering examples. To get an idea of whether the performance is good or bad, the same problems are solved with the other algorithms that are presented in Sect. 3. 1,000 optimizations are performed for each algorithm and problem in order to estimate the corresponding hit-rates, mean number of model calls and performance indexes. The starting point for each optimization is randomly selected to be within the lower and upper bounds of the parameter values for the different problems. For the SAO, the number of individuals in the population is set to 40, the generation gap is set to 95 % (meaning that the two best individuals in each generation are kept) and the optimization is stopped when the same optimum has been found five consecutive times. Since meta-optimization of Complex-RFM is in focus, its parameters are optimized to maximize its performance for each problem. This process can be seen in Fig. 3. The Complex-RF algorithm with standard values for R and F is used as an optimizer to optimize the parameters of Complex-RFM for the studied problems. The four parameters are the forgetting and randomization factors, γ and R_{fak}, as well as the following two parameters:

ε is the threshold which decides if two SMs are equal or not.
ς is the number of iterations "SM-check" is omitted when an SM is deemed accurate.

The comparison will be biased towards Complex-RFM since its parameters are optimized whereas the parameters of the other algorithms are not. To make it more fair, standard parameters for Complex-RFM that can be used to solve arbitrary problems are suggested in Sect. 5.7 and all problems are solved with these parameters as well. This means that all algorithms optimize the problems with standard parameters and the comparison should be more fair.

The criterion for whether an optimization was successful or not is important for the performance index and in the following problems it is assumed to be a solution that is among the best 0.1 % of all possible solutions. This is a reasonable assumption of a successful optimization from an engineering point of view and is for example used by Reisenthel and Lesieutre (2011). This is estimated by drawing 10 million random samples in the design space for the analytical problems and 100,000 samples for the engineering problems.

5.1 Analytical test function: the Peaks function

Peaks is an analytical function which can be viewed by writing peaks in the MATLAB command window and its equation is shown in Eq. 7. It consists of a flat area surrounding several peaks, which builds a global minimum in [0.231 −1.626] and a local minimum in [−1.348 0.205].

$$\begin{aligned} g_1(x) = {} & 3(1-x_1)^2 exp\left\{-x_1^2-(x_2+1)^2\right\} \\ & -10\frac{x_1}{5}-x_1^3-x_2^5 \quad exp\left\{-x_1^2-x_2^2\right\}-\frac{1}{3}exp\left\{-(x_1+1)^2-x_2^2\right\}, \end{aligned} \tag{7}$$

subject to

$$-3 \le x_i \le 3, \quad i = 1, 2.$$

For the comparison of the hit-rates of the different optimization algorithms, optimizations that result in a value of less than −6.4 are considered successful. The resulting hit-rates and mean numbers of required calls of Peaks can be seen in Table 1 and the optimal parameters for the Complex-RFM algorithm in Table 7. The average optimum function value for all 1,000 optimizations for the same optimization algorithm is shown in the row "Mean Objective Value" and the standard deviations of these function values are shown in the row below, "Standard Deviation". There are two columns for Complex-RFM. The one named "optimal" has been optimized to be as efficient as possible for this problem, whereas the one named "standard" uses the suggested parameter values that are found in Table 7.

A Welch's t test (1947) of the mean values and standard deviations of the optimums found by Complex-RF and Complex-RFM reveals that the optimums belong to different distributions ($t = 3.69$, df $= 1933, p \le 0.01$). Even though the hit-rate of the Complex-RFM is lower than the hit-rate for Complex-RF, the decrease in evaluations of peaks

Table 1 Comparison of the performance of a few optimization algorithms for Peaks

Entities	Complex-RF	Complex-RFM Optimized	Complex-RFM Standard	fmincon	SAO
Hit-rate (%)	67.4	56.7	60.1	31.8	92.8
Function evaluations	116	39	51	27	47
Mean Optimum Value	−5.21	−4.58	−4.86	−2.70	−6.35
Standard Deviation	2.05	2.50	2.21	2.87	0.80
Performance index	0.620	0.883	0.835	0.758	0.996

results in a better performance index for Complex-RFM. Also note that fmincon gets a high performance index even though the hit-rate is a rather mediocre 30% since the number of function calls is low. The GA-based SAO clearly outperforms the other algorithms since it has a higher hit-rate and requires fewer function calls.

5.2 Analytical test function: Hartmann6

To investigate the impact of increasing number of variables, the Harmann6 function is chosen. It has six variables and is consequently difficult to plot, but its equation is shown in Eq. 8.

$$f(x) = -\sum_{i=1}^{4} c_i e^{-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2}. \tag{8}$$

The numerous coefficients of this function can be found in Annex 3. The function has a global minimum of -3.32 in $x_{opt} = [0.2\ 0.15\ 0.475\ 0.275\ 0.31\ 0.655]$ and an optimization is deemed successful if the value at the suggested optimum is less than -2.7. The resulting performance of the different optimization algorithms can be found in Table 2, whereas the optimal parameters for Complex-RFM can be found in Table 7.

All optimization algorithms display high hit-rates for this problem. This means that fmincon would be the recommended algorithm since it needs the fewest function evaluations to converge. It should also be noted that Complex-RFM is much more effective than Complex-RF. Complex-RFM needs many function evaluations to converge, but most are performed on the SM. Since Hartmann6 is an analytical function it is redundant, but if it had been a computationally expensive model it would have been essential.

5.3 Engine Timing Model with closed-loop control

An engineering optimization problem is to optimize the control system of the Engine Timing Model that is part of the demo package of MATLAB Simulink. It can be found in the Simulink toolbox under the name “sldemo_ enginewc”. The control system consists of two parameters—a proportional and an integral part. Since the purpose of the control system is to adapt the output signal, y, as accurately as possible to the reference signal, y_{ref}, the objective function is chosen to be the error squared between the two signals for each time-step, as shown in Eq. 9.

Table 2 Comparison of the performance of a few optimization algorithms for the Hartmann6 function

Entities	Complex-RF	Complex-RFM Optimized	Complex-RFM Standard	fmincon	SAO
Hit-rate (%)	100	100	99.9	100	98.0
Function evaluations	500	268	235	129	60
Mean Optimum Value	−3.29	−3.29	−3.28	−3.28	−3.18
Standard Deviation	0.063	0.060	0.116	0.056	0.0203
Performance index	1	1	0.947	1	0.999

$$y = \sum_{t=1}^{T} (y_{ref}(t) - y(t))^2. \tag{9}$$

The values of the two parameters are constrained to a range between 0.001 and 1 and the criterion for a successful optimization is that the design has an objective function value of less than 3.5e7.

The results can be seen in Table 3 and it is notable that the performance index of Complex-RFM is more than double the indices of the other optimization algorithms. This means that there is at least a doubled chance to find the optimum with Complex-RFM when the allowed computational budget is 100 simulations. The gradient-based fmincon struggles with this problem and has the worst performance index.

5.4 Anti-Windup PID Control demonstration with feedforward control

Another engineering optimization problem that is part of the MATLAB Simulink demo package is a demonstration of Anti-Windup for a PID controller. The model can be found in the Simulink toolbox under the name "sldemo_ antiwindupfeedforward" and the controller consists of four parameters. The parameters are the proportional, integral and derivative parts as well as the filter coefficient. Those parameters are allowed to be between [0.001 0.001 −5 0.001] and [15 15 5 5]. This problem also uses Eq. 9 as objective function and a value below 2,000 is considered successful.

The results can be seen in Table 4. The GA-based SAO performs best thanks to the fact that it requires few function evaluations to converge and it is closely followed by Complex-RFM.

Table 3 Comparison of the performance of a few optimization algorithms for the Engine Timing Model

Entities	Complex-RF	Complex-RFM Optimized	Complex-RFM Standard	fmincon	SAO
Hit-rate (%)	39.7	42.8	37.9	17.6	13.4
Function evaluations	158	54	67	92	50
Mean Optimum Value	3.67e7	3.64e7	3.73e7	4.48e7	3.77e7
Standard Deviation	4.45e6	3.52e6	5.27e6	3.13e7	2.24e6
Performance index	0.274	0.644	0.509	0.190	0.250

Table 4 Comparison of the performance of a few optimization algorithms for the Anti-Windup PID Control model

Entities	Complex-RF	Complex-RFM Optimized	Complex-RFM Standard	fmincon	SAO
Hit-rate (%)	85.5	61.5	53.5	31.8	24.5
Function evaluations	497	120	160	160	49
Mean Optimum Value	1,580	1,792	1,890	2,390	2,120
Standard Deviation	344	398	405	915	337
Performance index	0.322	0.549	0.380	0.213	0.436

5.5 Electrical motorcycle

The last engineering application is the design of a gearbox with two gears for an electric motorcycle. The model is implemented in MATLAB Simulink and a screenshot is shown in Fig. 4. The model consists of a battery, an electric motor, a gearbox and the motorcycle itself. A more detailed description can be found in Annex 4.

For the design problem, the target is to design an electric motorcycle with as high acceleration as possible and the speed after 5 s is used as an acceleration measure. The design variables are the gear ratios of the first and second gears and the velocity when the gear is changed. The design variables are considered to lie between [1 1 1] and [20 20 75].

The criterion for a successful optimization for this comparison is a suggested design which leads to a speed after 5 s of more than 72 km/h. The performance of the optimization algorithms is shown in Table 5. fmincon has major difficulties to converge to the global optimum. The SAO and Complex-RFM perform approximately similar and outperforms the other algorithms with a large margin.

5.6 Standard parameters

Complex-RFM has a set of parameters that need to be determined. The optimal values of the parameters of Complex-RFM for each of the different problems are

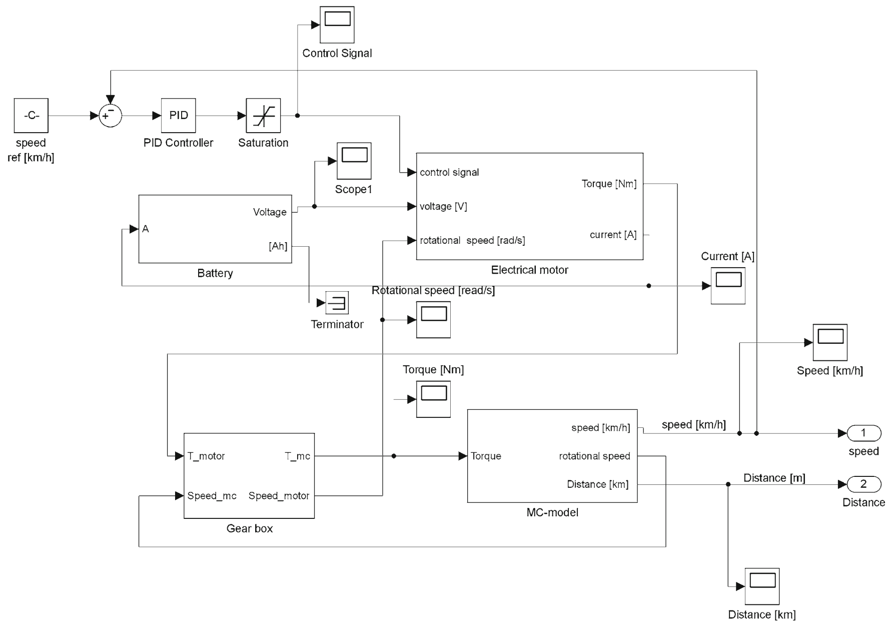

Fig. 4 A screenshot of the electric motorcycle model from MATLAB Simulink

Table 5 Comparison of the performance of a few optimization algorithms for the electrical motorcycle

Entities	Complex-RF	Complex-RFM Optimized	Complex-RFM Standard	fmincon	SAO
Hit-rate (%)	89.0	88.6	89.3	27.5	55.7
Function evaluations	353	126	146	131	50
Mean Optimum Value	−73.9	−73.4	−73.6	−66.4	−71.9
Standard Deviation	2.76	4.61	3.04	10.02	2.36
Performance index	0.465	0.822	0.784	0.218	0.804

shown in Table 7. An optimization which takes the performance indices of all of the presented functions into account is performed to get a recommendation of optimal parameters for an arbitrary problem. This objective function can be seen in Eq. 10. The numbers that are used to normalize each performance index consist of the optimal performance index for each problem, which can be seen in the "Complex-RFM optimized" column of Table 6.

$$minf(x) = -\left(\left(\frac{pi_{Peaks}}{0.883}\right)^2 + \left(\frac{pi_{Hart6}}{1}\right)^2 + \left(\frac{pi_{Engine}}{0.644}\right)^2 + \left(\frac{pi_{PID}}{0.549}\right)^2 + \left(\frac{pi_{ELMC}}{0.822}\right)^2\right), \tag{10}$$

where

$$x = [R_{fak}\gamma\varepsilon\varsigma].$$

The performance of Complex-RFM when the suggested standard parameters are used is shown in Table 6. Performance is slightly worse than when optimal parameters for each problem are used, which can be expected. Complex-RFM nevertheless performs well. The optimal parameters obtained when Eq. 10 is solved are noted suggested standard parameters and are shown in the bottom of Table 7.

5.7 Summary of the comparison

The resulting performance indexes for the different algorithms and problems are summarized in Table 6. Here it can be seen that fmincon performs well for the analytical problems but struggles to solve the engineering problems. The SAO performs reasonably well for all problems. The most interesting comparison is between Complex-RFM and Complex-RF since the impact of the modifications can be seen there. Complex-RFM outperforms Complex-RF for all problems, which indicates that the modifications have improved the algorithm. Both are well suited to solve the engineering problems according to Table 6. Complex-RFM generally has the highest performance indices of the compared algorithms. This means that it has the highest chance of finding the optimum if the budget for the optimization is fixed.

It is important to mention that the comparison is biased towards Complex-RFM since its parameters are optimized to maximize its performance, whereas the other algorithms are used with their standard settings. It is probably possible to improve the

Table 6 Performance indices for the optimization algorithms and problems

Problems	Complex-RF	Complex-RFM Optimized	Complex-RFM Standard	fmincon	SAO
Peaks	0.620	0.883	0.835	0.758	0.996
Hartmann6	1	1	0.947	1	0.999
Engine Timing	0.274	0.644	0.509	0.190	0.250
PID demonstration	0.322	0.549	0.380	0.213	0.436
EL-MC	0.465	0.822	0.784	0.218	0.804

Table 7 Optimal parameters for Complex-RFM for each problem

Problems	R_{fak}	γ	ε	ς
Peaks	1	0.05	25	0.01
Hartmann6	0.2	0.05	2	0.002
Engine Timing	0.86	0	1	0.005
PID demonstration	0.79	0.04	3	0.001
Electrical MC	0.28	0.06	2	0.008
Suggested standard	0.50	0.20	4	0.004

performance of the SAO by tuning its parameters but here it is only used to give more reference performance indices for the comparison. Even though fmincon and the SAO are used with their standard settings it is interesting to see their performances to get a better understanding of the capacity of the other algorithms. The Complex-RFM with standard parameters performs well, which indicates that it is suitable for these kinds of problems.

It should also be pointed out that the criterion for when an optimization is successful or not affects the hit-rate and therefore also the value of the performance index. Other definitions of successful optimizations may therefore lead to different conclusions regarding the performance of the algorithms. The measure of a successful optimization being a solution that is better than 99.9 % of all possible solutions is meaningful from an engineering perspective. However, from a strictly mathematical perspective one might argue that other convergence criteria might be more accurate.

6 Conclusions

A new modified version of the Complex-RF optimization algorithm, Complex-RFM, is proposed. The intended use of Complex-RFM is for low-dimensional, moderately expensive models, with optimization wall-clock times of around a day. This is done by creating and using SMs iteratively during the optimization process. The calculations performed between each model call are kept low to make the algorithm fast and easy to use and modify.

A meta-optimization is performed to get a recommendation for standard parameters of Complex-RFM that can be used when an arbitrary problem is solved. Complex-RFM is a good all-round algorithm for optimization of problems with few variables according to the performed experiments. The experiments show that for a fixed computational budget, Complex-RFM has a greater chance than Complex-RF of finding a good optimum for Simulink system models with few parameters. It also outperforms the gradient-based fmincon and performs similarly to a GA-based SAO.

A suggestion for future work is to evaluate the performance of Complex-RFM for other problems since all tests up to this point have been done in MATLAB and Simulink.

Annex 1: pseudo code for the Original Model/SM activity of Complex-RFM

```
Have enough samples been drawn to create a PRS SM?
    Yes
        Estimate the objective function value by calling the PRS
    No
        Estimate the objective function value by calling the original model
```

Annex 2: pseudo code for the SM check activity of Complex-RFM

```
Was the original model called for estimating the objective function?
    Yes
    No
        Is the jump counter = jump max?
        Yes
            Estimate the objective function value by calling the original model
            Create a new PRS using the latest model calls
            Is the percentage difference between the two latest PRSs smaller than $\epsilon$?
            Yes
                jump max= $\varsigma$
            No
                jump max = 0
                jump counter = 0
        No
            jump counter = jump counter +1
```

Annex 3: coefficients of the Hartmann6 function

See Table 8.

Table 8 Parameters for Hartmann6

a						
i\j	1	2	3	4	5	6
1	10	3	17	3.5	1.7	8
2	0.05	10	17	0.1	8	14
3	3	3.5	1.7	10	17	8
4	17	8	0.05	10	0.1	14
c						
i	1	2	3	4		
	1	1.2	3	3.2		
p						
i\j	1	2	3	4	5	6
1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	0.2348	0.1451	0.3522	0.2883	0.3047	0.665
4	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

The equation can be seen in Eq. 8

Annex 4: a description of the electric motorcycle model

The electric motorcycle model consists of four major parts—a battery, an electric motor, a gearbox and the motorcycle itself. The parameters of the model can be seen in Table 9.

Table 9 Parameters for the electric motorcycle model

Motorcycle parameters	
$m = 200$	(kg)
$A = 0.75$	
$C_d = 0.75$	Drag coefficient (−)
$\rho = 1.2$	Air density (kg/m^3)
$\mu = 0.1$	Friction coefficient (−)
$r = 0.3$	Wheel radius (m)
Electric motor parameters	
$L = 0.01$	Inductance (H)
$R = 0.05$	Resistance (Ohm)
$k_e = 0.18$	Motor constant (Nm/A)
$k_m = k_e$	
$I_{max} = 300$	Maximum current (A)
Battery parameters	
$U_{max} = 72$	Maximum voltage (V)
$Ah_{max} = 60$	
$K_{bat} = 0.18$	Battery coefficient to reduce the voltage
Gearbox parameters	
$u_1 = 6$	Gear ratio first gear
$u_2 = 4$	Gear ratio second gear
$v_{shift} = 60$	Speed when shifting from u_1 to u_2
$speed_{ref} = 150$	Target speed

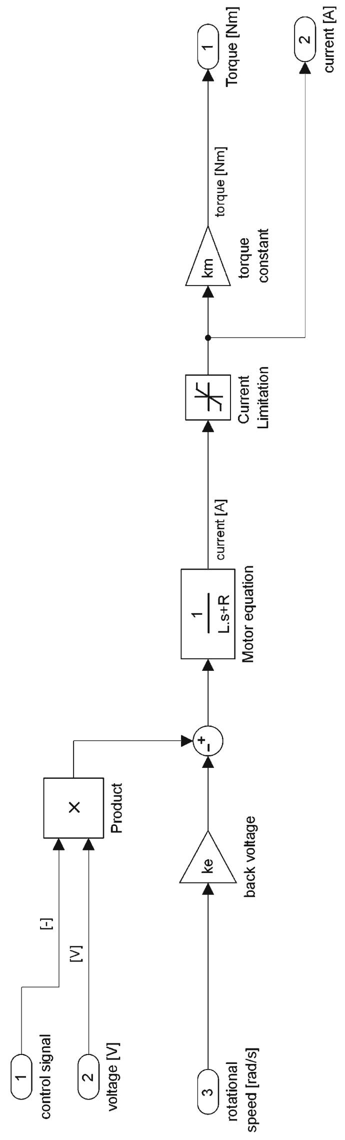

Fig. 5 A screenshot of the electric motor submodel from MATLAB Simulink

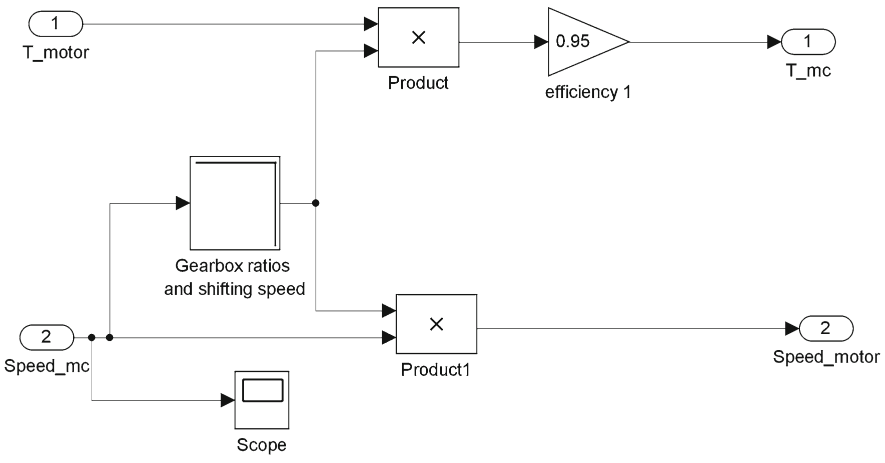

Fig. 6 A screenshot of the gearbox submodel from MATLAB Simulink

The battery model

The voltage of the battery is modeled as a linearly decreasing function of the energy it has delivered, see Eq. 11.

$$V(t) = U_{max} - K_{bat} \int_{0}^{t} A dt. \tag{11}$$

However, the battery can only deliver a certain amount of energy. When the maximum number of Ah is reached, the battery is considered empty and the voltage is set to zero.

The electric motor

The equations used to model the electric motor are shown in Eq. 12.

$$L\frac{di}{dt} = C_{in}U - k_E\omega - R \cdot i, \tag{12}$$
$$T_{out} = k_m i.$$

A picture of the motor model is shown in Fig.5. The control signal to the motor (C_{in} in Eq. 12) enters the model at the top and is used to reduce the voltage to the motor. The control signal is between -1 and 1 where 1 means full throttle.

The gearbox

The gearbox has two gears and it is modeled using three parameters, u_1, u_2 and v_{shift}. The gear ratio is changed from u_1 to u_2 at the rotational speed v_{shift}. In the gearbox, the output torque is amplified by a factor u_1 (or u_2) and the torque is then reduced by

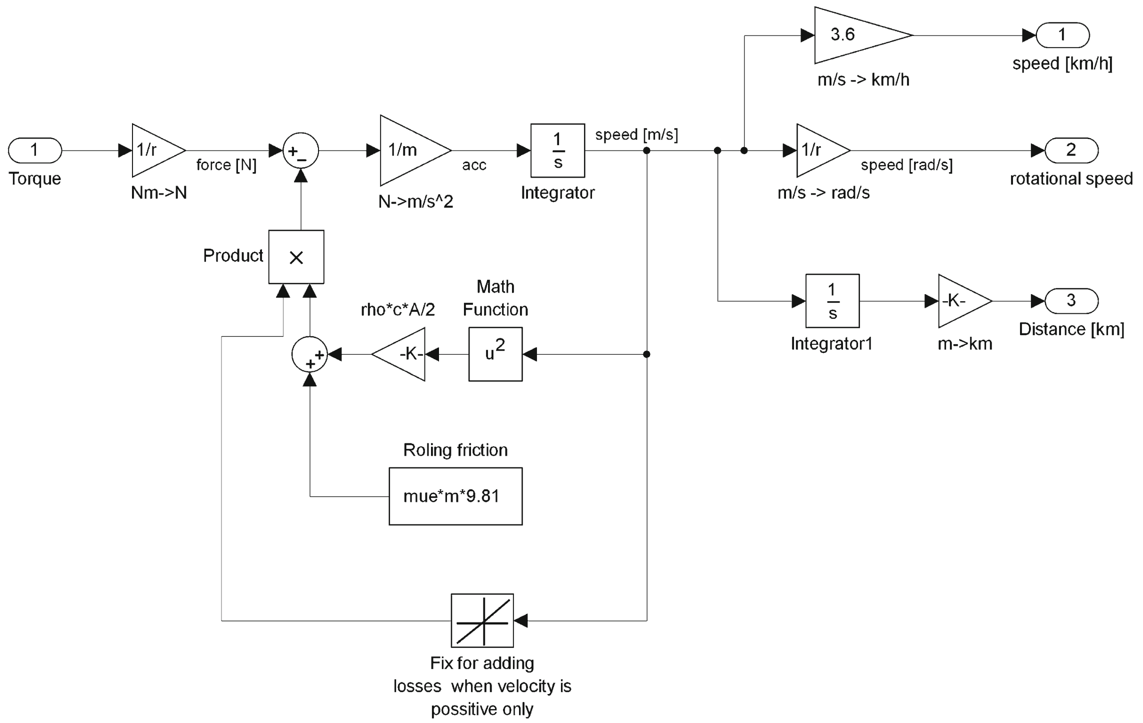

Fig. 7 A screenshot of the motorcycle submodel from MATLAB Simulink

a factor of 0.95 which models all mechanical losses of the transmission. The model of the gearbox is shown in Fig. 6.

The motorcycle model

The motorcycle itself is modeled using Newton's second law, where the input torque is divided by the wheel radius to give the force driving the motorcycle forward. From this force a rolling resistance and the air drag are subtracted, see Eq. 13.

$$\frac{T_{in}(t)}{r} - C_d A \rho \frac{v(t)^2}{2} - \mu \cdot mg = m \cdot a(t). \qquad (13)$$

The acceleration $a(t)$ is then integrated to obtain the velocity, which in turn is integrated in order to calculate the distance travelled by the motorcycle, see Fig. 7.

References

Andersson J (2001) Multiobjective optimization in engineering design applications to fluid power systems. PhD Thesis, Linköping University, Linköping

Bonnans JF, Gilbert JC, Lemaréchal C, Sagastizábal CA (2006) Numerical optimization: theoretical and practical aspects. Springer, New York

Box M (1965) A new method of constrained optimization and a comparison with other methods. Comput J 8(1):42–52

Duvigneau R, Praveen C (2007) Meta-modeling for robust design and multi-level optimization. In: 42e Colloque dAérodynamique Appliquée, Couplages et Optimisation Multidisciplinaires, INRIA Sophia Antipolis, 19–21 March

Forrester A, Sóbester A, Keane A (2008) Engineering design via surrogate modelling: a practical guide. Wiley, Chichester

Goldberg DE, Holland JH (1988) Genetic algorithms and machine learning. Mach Learn 3(2):95–99

Grefenstette JJ (1986) Optimization of control parameters for genetic algorithms. IEEE Trans Syst Man Cybern 16(1):122–128

Holland JH (1975) Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Ann Arbor

Isaaks EH, Srivastava RM (1989) Applied geostatistics. Oxford University Press, New York

Jin R, Chen W, Simpson TW (2001) Comparative studies of metamodelling techniques under multiple modelling criteria. Struct Multidiscip Optim 23(1):1–13

Jin R, Du X, Chen W (2003) The use of metamodeling techniques for optimization under uncertainty. Struct Multidiscip Optim 25(2):99–116

Johansson A, Andersson J, Palmberg JO (2002) Optimal design of the cross-angle for pulsation reduction in variable displacement pumps. In: Power transmission and motion control: PTMC 2002, pp 319–333

Johnson RT, Montgomery DC, Jones B, Fowler JW (2008) Comparing designs for computer simulation experiments. In: Proceedings of the 40th conference on winter simulation, winter simulation conference, pp 463–470

Kitayama S, Arakawa M, Yamazaki K (2011) Sequential approximate optimization using radial basis function network for engineering optimization. Optim Eng 12(4):535–557

Krus P, Ölvander J (2003) Optimizing optimization for design optimization. In: Design engineering technical conferences and computers and information in engineering conference, 2003. ASME Press, New York

Krus P, Ölvander J (2012) Performance index and meta-optimization of a direct search optimization method. Eng Optim 1(ahead-of-print):1–19

Nelder JA, Mead R (1965) A simplex method for function minimization. Comput J 7(4):308–313

Nocedal J, Wright S (2006) Numerical optimization, series in operations research and financial engineering. Springer, New York

Park HS, Dang XP (2010) Structural optimization based on CAD–CAE integration and metamodeling techniques. Comput-Aided Des 42(10):889–902

Redhe M, Forsberg J, Jansson T, Marklund PO, Nilsson L (2002) Using the response surface methodology and the d-optimality criterion in crashworthiness related problems. Struct Multidiscip Optim 24(3):185–194

Reisenthel PH, Lesieutre DJ (2011) A numerical experiment on allocating resources between design of experiment samples and surrogate-based optimization infills. In: 13th AIAA non-deterministic approaches conference

Shan S, Wang GG (2010) Metamodeling for high dimensional simulation-based design problems. J Mech Des 132:051009

Smit SK, Eiben AE (2009) Comparing parameter tuning methods for evolutionary algorithms. In: IEEE congress on evolutionary computation, 2009. CEC'09. IEEE, Piscataway, pp 399–406

Tarkian M, Persson J, Ölvander J, Feng X (2011) Multidisciplinary design optimization of modular industrial robots. In: Proceedings of the ASME 2011 international design engineering technical conferences and computers and information in engineering conference, IDETC/CIE 2011, Washington, DC, USA, 28–31 August 2011, pp 867–876

Wang GG, Shan S (2007) Review of metamodeling techniques in support of engineering design optimization. J Mech Des 129:370

Welch BL (1947) The generalization of 'student's' problem when several different population variances are involved. Biometrika 34(1/2):28–35