RESEARCH ARTICLE OPEN ACCESS

An Adaptive Approach for Single Objective Optimization

Ram Krishna Rathore *, Kaushal Sharma **, Amit Sarda***
*Assistant Professor in the Department of Mechanical Engineering at CCET, Bhilai (C.G), India
** Engineer- Inspection, RITES Ltd. India,
*** Associate Professor in the Department of Mechanical Engineering at CCET, Bhilai,

ABSTRACT
The use of evolutionary computation in the solution of optimization problems of non-linear type is not new. Many such problems having single or multiple objectives are now routinely solved using different evolutionary methodologies. Through this project, I am delighted to share some recent advances in the area of evolutionary computing. Some critical issues, such as design of an efficient evolutionary algorithm, an efficient constraint handling procedure, scalability issue of algorithms are dealt in this project. The discussion of the topics and subsequent engineering and numerical case studies presented in this project should be useful to non-linear single objective problems alike. Nonlinear Programming by Quadratic Lagrangian (NLPQL) techniques are extensively used for solving realistic optimization problems, particularly in structural mechanics. The common arrangement of NLPQL techniques is briefly discussed and it is shown how these techniques can be tailored for distributed computing. Still, NLPQL techniques are responsive topic to errors in parameters and gradient evaluations. Typically they take more time to compute the converged solution with more number of simulation calls. In case of noisy function values, a radical enhancement of the performance can be gained through Adaptive Nonlinear Programming by Quadratic Lagrangian (A-NLPQL) compared to the version with conventional NLPQL. Numerical results are presented for a set of six standard test examples.
Keywords - *NLPQL, Single Objective Optimization, nonlinear programming, distributed computing*

I. INTRODUCTION

Several types of optimization techniques are existed to solve diverse problems. Even though, for designers to employ optimization at their place of work they require to comprehend the hypothesis, the theory and the procedures for these techniques. This is due to realistic problems might necessitate altering algorithmic parameters and constant scaling and adjusting the available techniques to fulfills the definite application. Especially, the user might have to practice various optimization techniques to locate one that can be effectively applied. The definitive objective of every such choice is either to minimize the attempt required or maximize the required advantage. Because also of these objectives in any physical circumstances can be uttered as a function of definite design variables, optimization might also be distinct as the method of finding the circumstances that provide the maximum or minimum value of a function.

It is remarkable to remind that the key growths in the field of arithmetic techniques of unrestrained optimization have been prepared in the United Kingdom just in the l960s. The improvement of the simplex technique by Dantzig in 1947 for linear programming problems and the annunciation of the principle of optimality in 1957 by Bellman for dynamic programming problems lined the direction for progress of the techniques of constrained optimization Work by Kuhn and Tucker in 1951 on the essential and adequacy circumstances for the optimal solution of programming problems laid the foundations for a great deal of afterwards research in nonlinear programming. The offerings of Zoutendijk and Rosen to nonlinear programming in the early 1960s have been very important. Even though no particular method has been establish to be communally suitable for nonlinear programming examples, effort of Carroll and Fiacco and McCormick authorized lots of intricate problems to be solved by means of the well-known methods of unconstrained optimization Geometric programming was created in the l960s by Duffin, Zener, and Peterson. Gomoiy did revolutionary work in integer programming, one of the greater stimulating and rapidly growing areas of optimization. The reason for this is that usually real-world applications fall under this class of problems. Dantzig and Charnes and Cooper created stochastic programming methods and resolved problems by assuming design parameters to be autonomous and usually distributed. The need to optimize more than single objective or goal while fulfilling the physical boundaries led to the growth of multi-disciplinary programming techniques. Goal programming is a famous method for solving precise types of single objective optimization problems. The goal programming was initially projected for linear

problems by Charnes and Cooper in 1961. The basics of game hypothesis were laid by von Neumann in 1928 and ever since then the method has been useful to answer numerous mathematical economics and military problems. Simply in the last few years has game theory been useful to solve engineering design problems. Genetic algorithms, Simulated annealing, and neural network methods signify a new class of mathematical programming methods that have come into fame during the last decade. [35]

A significant solution with relevance to the Non linear programming based optimization methods to engineering fields has been the elevated computational cost because of the huge amount of simulation calls necessary for these techniques[1]. A general approach to decrease the computational attempt for such optimization techniques when integrated with simulation models is to use metamodeling techniques. Researchers have been quite active in developing models and methods that improve the efficiency of the NLPQLs in terms of the number of simulation calls. Some of these approaches are based on fitness approximations in which neural network [3–5], response surface [6], Kriging [7], and radial basis function [8] methods are used for metamodeling. Others use fitness inheritance approaches [9,10] in which the fitness of an offspring is inherited from its parents. A comprehensive review of fitness approximation and metamodeling approaches can be found in Ref. [16] and Refs. [17–19] respectively. The fitness approximation methods are of two types: off-line (non-adaptive) and on-line (adaptive). In off-line techniques, metamodels are created independently and earlier at the beginning of an optimization algorithm [4,6–8,20,21]. The deficiency of the offline techniques is that it is complex to attain both an excellent reliability metamodel above the complete design space and at the same time keep a small number of simulation calls [18,20]. The on-line techniques utilize a group of metamodeling with the simulation model at the optimization process while adaptively enhancing the metamodel [2,3,5,11–14]. Most of the on-line techniques created until now are focused on single-objective optimization.

The study on how to implant metamodeling inside Non Linear Programming by Quadratic Lagrangian (NLPQL) remains sparse. In on-line techniques, the primary phases of the NLPQL, coarse design points are formed with metamodels are created. These metamodels are then steadily enhanced as further simulation data become accessible. Some of this type of techniques employs regression metamodeling, which is well-known to necessitate a huge number of simulation calls. Another uncertain problem in the present adaptive techniques is how to impartially choose when to switch to the metamodel in its place of using the simulation in the optimization. Typically, the toggling among the definite simulation model and the consequent metamodel is intuitively decided. Furthermore, the reliability of the metamodel may vary extensively in the optimization procedure and this can cause fluctuation.

I employ a goal measure to decide whether a simulation model or its Kriging metamodel substitution have to be used to assess design points. The projected decisive factor is created on the basis of the metamodels expected error, which can be simply attained as a consequence from Kriging and Latin Hypercube Sampling. In the anticipated technique, the Kriging metamodels for objective and restraint functions are constructed and adaptively enhanced inside a NLPQL by means of Latin Hypercube Sampling technique (as A-NLPQL or Adaptive-Non Linear Programming by Quadratic Lagrangian). The technique is universal and needs no extra simulation call previous to the beginning of the optimization process to construct the Kriging metamodels. These present results demonstrate that the projected technique decides the problem frequently reported in the literature, that is, the metamodel possibly of small reliability and that it may create false optima.

II. PROBLEMS IDENTIFICATION

Now days there are so many techniques, which are working on optimization technique; they take lots of iterations and population creation. The common issues appear with the working of conventional or simple single-objective algorithms are as follows:

1. Simulation calls counts are more in case of conventional optimization techniques.
2. It requires a control of design of experiment for the initial population creation, which has to deal with the efficiency of the various algorithm and design points generation.
3. Time required for converging the solutions and simulations are more.
4. A Response Surface Optimization system draws its information from its own Response Surface component, and so is dependent on the quality of the response surface.

On the basis of above listed problem the algorithms are evaluated and compared to show the usability.

III. OBJECTIVE

Although kriging's and NLPQLs have been extensively used in engineering design optimization, the significant confront still faced by designers in using these methods is their high computational cost due to the population-based nature of these methods. In particular, a number of techniques incorporating metamodeling with NLPQL based methods have

been reported in the literature [Hailong You, 2009]. A metamodel means a simplified approximation of the original simulation model.

The objective of Research Thrust is to develop an approach to measure the uncertainty in the prediction of responses from the metamodels so that the risk of generating false optima can be reduced. The goal is to develop a NLPQL that can converge to the Pareto front using significantly fewer number of simulation calls compared to a conventional NLPQL.

IV. ASSUMPTIONS

The In generating the robust optimization approach, we make the following assumptions:

- The range of parameter uncertainty is known as an interval (or several discrete intervals) *a priori*. Interval uncertainty is not required to be continuous.
- An acceptable variation range for each objective function in the optimization,
- Simulations used in optimization problems are considered as "black boxes" that will provide the identical responses (outputs) when the same inputs are supplied.
- Design variables and/or parameters in optimization problems can be continuous-discrete.

V. ADAPTIVE NON-LINEAR PROGRAMMING BY QUADRATIC LAGRANGIAN (A-NLPQL) APPROACH

Adaptive–Non linear Programming by Quadratic lagrangian (A-NLPQL) is a mathematical optimization method that combines a Latin Hypercube Sampling (LHS) Design of Experiments, a Kriging response surface, and the NLPQL optimization algorithm. It is a gradient-based algorithm based on a response surface which provides a refined, global, optimized result. Adaptive-NLPQL Single-Objective optimization supports a single objective, multiple constraints, and is limited to continuous parameters. It is available only for Direct Optimization systems.

Like the NLPQL method, this method solves constrained nonlinear programming problems of the form:

Minimize: $F = f(\{x\})$

Subject to: $g_k(\{x\}) \le 0 \qquad k = 1, \ldots, K$

$h_l(\{x\}) \le 0 \qquad l = 1, \ldots, L$

where $\{x_L\} \le \{x\} \le \{x_U\}$

The purpose is to refine and reduce the domain intelligently and automatically to provide the global maxima.

A-NLPQL Steps

1. **LHS Sampling**: Latin Hypercube Sampling (LHS) is used for the Kriging construction. When a new LHS is generated after a domain reduction, all the existing design points between the new bounds are kept. In the two-dimensional example below, only three new design points are evaluated because three old ones are kept.

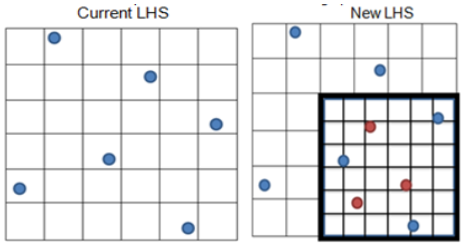

Fig. 1 LHS sampling

2. **Kriging Generation:** A response surface is created for each output, based on the current LHS and consequently on the current domain bounds.
3. **NLPQL Algorithm:** NLPQL is run on the current Kriging response surface to find potential candidates. A few NLPQL processes are run at the same time, beginning with different starting points, and consequently, giving different candidates.
4. **Candidate Point Validation:** All the obtained candidates are either validated or not, based on the Kriging error predictor. The candidate point is checked to see if further refinement of the Kriging surface will change the selection of this point. A candidate is considered as acceptable if there aren't any points, according to this error prediction, that call it into question. If the quality of the candidate is called into question, the domain bounds are reduced; otherwise, the candidate is calculated as a verification point.

- **Refinement Point Creation** (If the selection will *not* be changed): When a new verification point is calculated, it is inserted in the current Kriging as a refinement point and the NLPQL process is restarted.
- **Domain Reduction** (If the selection *will* be changed): When candidates are validated, new domain bounds must be calculated. If all of the candidates are in the same zone, the bounds are reduced, centered on the candidates. Otherwise, the bounds are reduced as an inclusive box of all candidates. At each domain reduction, a new LHS is generated (conserving design points between the new bounds) and a new Kriging is generated based on this new LHS.

5. **Convergence and Stop Criteria:** The optimization is considered to be converged when the candidates found are stable. However, there are three stop criteria that can stop the algorithm: the maximum number of

evaluations, the maximum number of domain reductions, and the percentage of input ranging.

The workflow of the A-NLPQL optimization technique is given in figure 2.

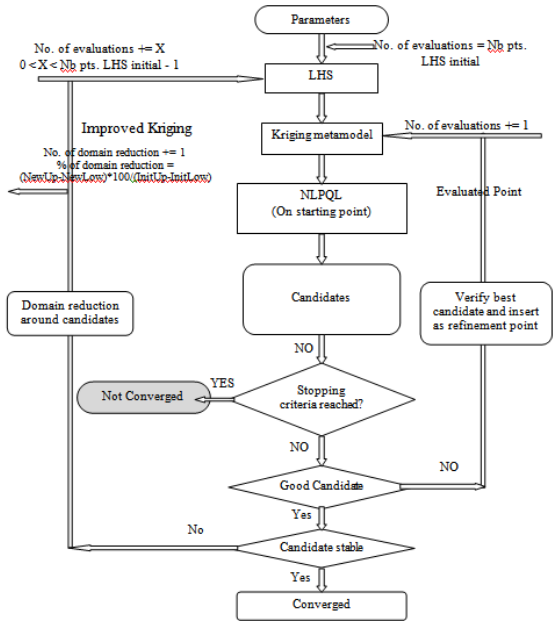

Fig. 2 Flowchart of Adaptive NLPQL approach.

VI. ASSESSMENT THROUGH EXAMPLES

Adaptive-NLPQL is a hybrid optimization method which combines an online LHS (DOE) and Kriging response surface with the NLPQL algorithm in a flexible Optimization system. It uses the same general approach as NLPQL, but extends it by using the Kriging error predictor to reduce the number of evaluations needed to local the global optimum.

To illustrate how A-NLPQL optimization works, we will use six different problems to examine different functions and apply both the NLPQL and A-NLPQL optimization methods to the problem. Then, we will review the results and examine why Adaptive NLPQL optimization method is better suited to finding the converged solution for the given problem.

In this section, we use six numerical examples with different degrees of difficulty to illustrate the applicability of the proposed A-NLPQL, compared to the NLPQL. All of these six examples are optimizations problems with constraint functions. As a typical example of my results, we use the examples, to present a detailed comparison of the NLPQL, and A-NLPQL, for that the results for the five numerical examples used in literature and other one engineering example also. In order to compare the conventional NLPQL and A-NLPQL, the same initial population of design points is used for all experiments for each example. The same settings are used for all examples.

Number of LHS Initial Samples	110
Number of Screening Samples	1300
Number of Starting Points	110
Maximum Number of Evaluations	300
Maximum Number of Domain Reductions	10
Percentage of Domain Reductions	0.1
Maximum Number of Candidates	3

Problem 1: The problem is a non-convex analytic function with two input parameters taken from, Jui-Yu Wu, 2012[24]. The definition of the problem is as follows:

Minimize $f(x_1, x_2)$

Where $-3.0 \le x_1, x_2 \le 3$

And $f(x_1, x_2) = 3(1-x_1)^2 e^{[-x_1^2-(x_2+1)^2]} - 10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right) e^{[-x_1^2-x_2^2]} - \frac{1}{3} e^{[-(x_1+1)^2-x_2^2]}$

This analytic function has three local maxima, one local minima, and one universal minimum point at (**0.2282**;**-1.6256**), with a corresponding objective function value of **-6.5511**.

Optimization Status	
Converged	Yes
Number of Iterations	10
Number of Evaluations	55
Number of Failures	0
Size of Generated Sample Set	10
Number of Candidates	3

Vs

Optimization Status	
Converged	Yes
Number of Evaluations	55
Number of Domain Reductions	5
Number of Failures	0
Size of Generated Sample Set	55
Number of Candidates	3

Fig. 3 optimization status of NLPQL (Left) and A-NLPQL (Right)

Name	P1 - wb_x1	P2 - wb_x2	P3 - wb_y	
			Parameter Value	Variation from Reference
Starting Point	-0.1256	-0.8258	1.7549	0.00 %
Candidate Point 1	0.22832	-1.6255	-6.5511	-473.31 %
Candidate Point 2	0.14448	-1.5914	-6.4541	-467.78 %
Candidate Point 3	0.41207	-1.746	-5.9999	-441.90 %

Vs

Name	P1 - wb_x1	P2 - wb_x2	P3 - wb_y	
			Parameter Value	Variation from Reference
Candidate Point 1	0.22881	-1.6252	-6.5511	0.00 %
Candidate Point 2	0.25152	-2.1496	-3.8246	41.62 %
Candidate Point 3	1.2414	-1.2918	-0.74605	88.61 %

Fig. 4 NLPQL (left) and A-NLPQL's (right)

Candidate points of converged solution

In the figure 4 the candidate points of converged solution are given in NLPQL and A-NLPQL candidate 1 has been selected which is $f(x)$ -6.5511 for both case the results are same. The Pareto frontiers from the NLPQL and A-NLPQL, respectively, are non-convex as shown in Figure 5.

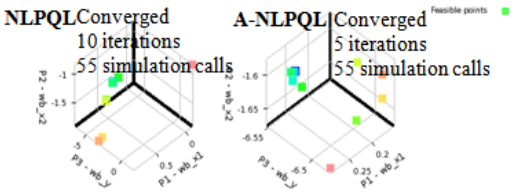

Fig. 5 Pareto solutions for NLPQL and A-NLPQL

Figure 5 shows a typical set of Pareto optimal solutions as obtained from one of the 19 iterations of the NLPQL and A-NLPQL. The results from A-NLPQL are in good agreement with the NLPQL. Figure 5 shows the *NumSimCall* (number of simulation calls) for 55 iterations. The results show that for problem 1, the *NumSimCall* has been reduced by 5 iterations using the proposed A-NLPQL compared to the NLPQL; so the optimization process was much faster.

Times taken to get a converged solution of the problem by both the methods, is the other criteria. In this section both the methods are run with the same settings and time and number of simulation calls are compared for the same. Time taken for complete the converged solution for problem 1 through NLPQL is 21min and through A-NLPQL is 15min.

Problem 2: (Pressure Vessel Design Problem) Definition

In this section, the engineering problem from Jui-Yu Wu, 2012[24] has been taken to further test the performance of the proposed A-NLPQL in solving problems in a discontinuous search space.

This problem involves four decision variables, four inequality constraints, and eight boundary conditions. This problem attempts to minimize the total cost ($f(\mathbf{x})$), including cost of materials welding and forming. A cylindrical vessel is capped at both ends by hemispherical heads. Four design variables exist: thickness of the shell $x1$, thickness of the head $x2$, inner radius $x3$, and length of the cylindrical section of the vessel, excluding the head $x4$. The definition of the problem is as follows:

Minimize

$$f(x)=0.6224x_1x_3x_4+1.7781x_2x_3^2+3.1661x_1^2x_4+19.84x_1^2x_3$$

Subjected to $g_1(x) = -x_1+0.0193x_3 \leq 0$

$g_2(x) = -x_2+0.00954x_3 \leq 0$

$g_3(x) = -\pi x_2^2x_4-(4/3)\pi x_3^3+1296000 \leq 0$

$g_4(x) = x_4-240 \leq 0$

Where $0<x_1, x_2<100$ **and** $10<x_3, x_4<200$

Optimization Status		Optimization Status	
Converged	Yes	Converged	Yes
Number of Iterations	26	Number of Evaluations	52
Number of Evaluations	221	Number of Domain Reductions	2
Number of Failures	0	Number of Failures	0
Size of Generated Sample Set	26	Size of Generated Sample Set	52
Number of Candidates	3	Number of Candidates	3

Fig. 6 Prob. 2 optimization status of NLPQL (Left) and A-NLPQL (Right)

Fig. 7 NLPQL (left) and A-NLPQL's (right) Candidate points of converged solution of Prob. 2

The best known solution is $(x) = (0.193, 45.343, 10, 200)$, where $f(x) = 8333.7$ by A-NLPQL. The Pareto frontiers from NLPQL and A-NLPQL, respectively, are non-convex as shown in Figure 8.

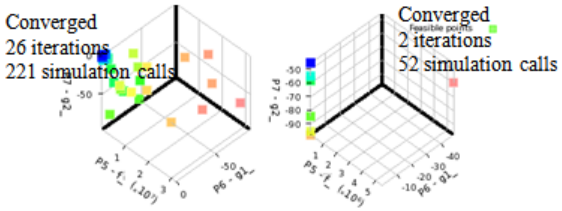

Fig. 8 Pareto solutions for NLPQL (Left) and A-NLPQL (Right)

Figure 8 shows a typical set of Pareto optimal solutions as obtained from one of the 19 iterations of the NLPQL and A-NLPQL. The results from A-NLPQL are good as compared with the NLPQL. Figure 8 shows the *NumSimCall* (number of simulation calls) for 30 iterations. The results show that for problem 2, the *NumSimCall* has been reduced by 169 simulation calls and 24 iterations using the proposed A-NLPQL compared to the NLPQL; so the optimization process was much faster. Time taken for complete the converged solution for problem 2 through NLPQL is 25min and through A-NLPQL is 10min.

Problem 3: Definition

This standard problem is taken from Yong Wang et. al, 2007[23]. The problem formulation has been given as:

Maximize $f(x) = x_1^2+(x_2-1)^2$

subject to $g(x) = x_2-x_1^2 \leq 0$

where $-1 \leq x_1, x_2 \leq 1$

The optimum solution is $(x) = (\pm 1/\sqrt{2}, 1/2)$, where $f(x) = 0.75$.

Optimization Status	
Converged	Yes
Number of Iterations	5
Number of Evaluations	24
Number of Failures	0
Size of Generated Sample Set	5
Number of Candidates	2

VS

Optimization Status	
Converged	Yes
Number of Evaluations	12
Number of Domain Reductions	1
Number of Failures	0
Size of Generated Sample Set	12
Number of Candidates	2

Fig. 9 Prob.3 optimization status of NLPQL (Left) and A-NLPQL (Right)

N...	P1 - x1_	P2 - x2_	P3 - f_ Parameter Value	P3 - f_ Variation from Reference	P4 - g1_ Parameter Value	P4 - g1_ Variation from Reference
Starting Point	1	1	1	0.00 %	0	0.00 %
Candidate Point 1	0.70711	0.5	0.75	-25.00 %	-4.2099E-09	
Candidate Point 2	1	1	1	0.00 %	0	

N...	P1 - x1_	P2 - x2_	P3 - f_ Parameter Value	P3 - f_ Variation from Reference	P4 - g1_ Parameter Value	P4 - g1_ Variation from Reference
Candidate Point 1	-0.70711	0.5	0.75	0.00 %	0	100.00 %
Candidate Point 2	0.70711	0.5	0.75	0.00 %	-5.5511E-16	0.00 %

Fig. 10 NLPQL (left) and A-NLPQL's (right) Candidate points of converged solution of Prob. 3

The Pareto frontiers from NLPQL and A-NLPQL, respectively, are non-convex as shown in Figure 11.

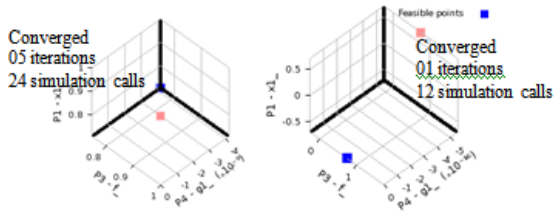

Fig. 11 Pareto solutions for NLPQL (Left) and A-NLPQL (Right) of prob.3

Figure 11 shows a typical set of Pareto optimal solutions as obtained from the NLPQL and A-NLPQL. The results from A-NLPQL are good as compared with the NLPQL, that is $f(x) = 0.75$. Figure 11 shows the *NumSimCall* (number of simulation calls) for 30 iterations. The results show that for problem 3, the *NumSimCall* has been reduced by 12 simulation calls and 04 iterations using the proposed A-NLPQL compared to the NLPQL; so the optimization process was much faster. Time taken for complete the converged solution for problem 3 through NLPQL is 10min and through A-NLPQL is 5min.

Problem 4: Definition

This standard problem is taken from Yong Wang et. al, 2007[23]. The problem formulation has been given as:

Minimize $f(x) = (x_1-10)^3+(x_2-20)^3$

Subject to $g_1(x) = -(x_1-5)^2-(x_2-5)^2+100 \leq 0$

$g_2(x) = (x_1-6)^2-(x_2-5)^2-82.81 \leq 0$

Where $13 \leq x_1 \leq 100$ **and** $0 \leq x_2 \leq 100$

For the above given problem the optimization status are as shown below:

Optimization Status	
Converged	Yes
Number of Iterations	9
Number of Evaluations	38
Number of Failures	0
Size of Generated Sample Set	9
Number of Candidates	3

VS

Optimization Status	
Converged	Yes
Number of Evaluations	25
Number of Domain Reductions	3
Number of Failures	0
Size of Generated Sample Set	25
Number of Candidates	3

Fig. 12 Prob.4 optimization status of NLPQL (Left) and A-NLPQL (Right)

N...	P1 - x1_	P2 - x2_	P3 - f_ Parameter Value	P3 - f_ Variation from Reference	P4 - g1_ Parameter Value	P4 - g1_ Variation from Reference	P5 - g2_ Parameter Value	P5 - g2_ Variation from Reference
Starting Point	20	70	1.26E+05	0.00 %	-3900	0.00 %	-4111.8	0.00 %
Candidate Point 1	13	20.361	27.047	-99.98 %	-71.972	98.15 %	-269.78	93.44 %
Candidate Point 2	22.542	26.435	2239.2	-98.22 %	-51.759	98.67 %	-268.65	93.47 %
Candidate Point 3	13	38.385	6240.9	-95.05 %	-950.53	75.63 %	-1148.3	72.07 %

N...	P1 - x1_	P2 - x2_	P3 - f_ Parameter Value	P3 - f_ Variation from Reference	P4 - g1_ Parameter Value	P4 - g1_ Variation from Reference	P5 - g2_ Parameter Value	P5 - g2_ Variation from Reference
Candidate Point 1	13	17.806	16.442	-99.77 %	-7.481E-09	-21,93,366 .67 %	-197.81	13.40 %
Candidate Point 2	18.753	28.087	1199.5	-83.43 %	-243.86	-71,50,06,9 .00 %	-453.18	-98.40 %
Candidate Point 3	28.302	30.357	7241.2	0.00 %	-3.4106E-13	0.00 %	-228.41	0.00 %

Fig. 13 NLPQL (left) and A-NLPQL (right) Candidate points of converged solution of Prob.4

The Pareto frontiers from NLPQL and A-NLPQL, respectively, are non-convex as shown in Figure 14.

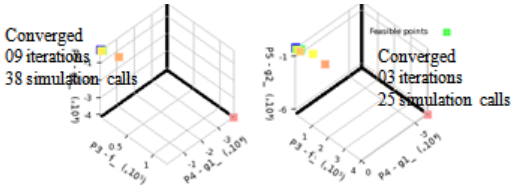

Fig. 14 Pareto solutions for NLPQL (Left) and A-NLPQL (Right) of Prob.4

Figure 14 shows the optimum solution calculated with NLPQL is $x = (13, 20.361)$, where $f(x) = 27.047$ and with A-NLPQL $x = (13, 17.806)$, where $f(x) = 16.442$. Both constraints are active. Figure 14 shows a typical set of Pareto optimal solutions as obtained from the NLPQL and A-NLPQL. Figure 14 shows the *NumSimCall* (number of simulation calls) for 30 iterations. The results show that, for problem 4, the *NumSimCall* has been reduced by 13 simulation calls and 06 iterations using the proposed A-NLPQL compared to the NLPQL; so the optimization process was much faster. Time taken for complete the converged solution for problem 4 through NLPQL is 12min and through A-NLPQL is 7min.

Problem 5: Definition

This problem is taken from Hira and Gupta, 2011[41]. The problem formulation has been given as:

Maximize $f(x) = 2x_1+3x_2$

Subjected to $g_1(x) = x_1+x_2 \leq 30$

$g_2(x) = x_1-x_2$ □ **0**

where $0 \le x_1 \le 20$ **and** $3 \le x_2 \le 12$

For the above problem the optimization status are as shown below:

Optimization Status		Optimization Status	
Converged	Yes	Converged	Yes
Number of Iterations	4	Number of Evaluations	15
Number of Evaluations	17	Number of Domain Reductions	1
Number of Failures	0	Number of Failures	0
Size of Generated Sample Set	4	Size of Generated Sample Set	15
Number of Candidates	2	Number of Candidates	3

Vs

Fig. 15 Prob.5 optimization status of NLPQL (Left) and A-NLPQL (Right)

N...	P1 - x1_	P2 - x2_	P3 - f_ Parameter Value	P3 - f_ Variation from Reference	P4 - g1_ Parameter Value	P4 - g1_ Variation from Reference	P5 - g2_ Parameter Value	P5 - g2_ Variation from Reference
Starting Point	12	5.255	7.9	0.00 %	3.3	0.00 %	0.7	0.00 %
Candidate Point 1	18	12	72	811.39 %	30	809.09 %	6	757.14 %
Candidate Point 2	12	5.255	7.9	0.00 %	3.3	0.00 %	0.7	0.00 %

N...	P1 - x1_	P2 - x2_	P3 - f_ Parameter Value	P3 - f_ Variation from Reference	P4 - g1_ Parameter Value	P4 - g1_ Variation from Reference	P5 - g2_ Parameter Value	P5 - g2_ Variation from Reference
Candidate Point 1	18	12	72	17.47 %	30	12.75 %	6	-42.64 %
Candidate Point 2	19.624	10.25	69.996	14.20 %	29.873	12.27 %	9.3739	-10.39 %
Candidate Point 3	18.534	8.0741	61.291	0.00 %	26.608	0.00 %	10.46	0.00 %

Fig. 16 NLPQL (left) and A-NLPQL (right) Candidate points of converged solution of Prob.5

The Pareto frontiers from NLPQL and A-NLPQL, respectively, are non-convex as shown in Figure 17.

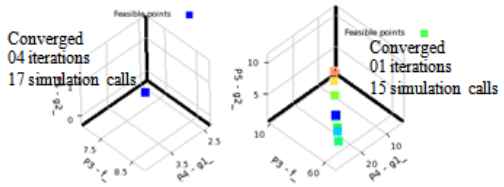

Fig. 17 Prob.5 Pareto solutions for NLPQL (Left) and A-NLPQL (Right)

Figure 16 shows the optimum solution calculated with NLPQL and A-NLPQL, which is (equal) x = (18, 12), where $f(x) = 72$. Both constraints are active. Figure 17 shows a typical set of Pareto optimal solutions as obtained from the NLPQL and A-NLPQL. Figure 17 shows the *NumSimCall* (number of simulation calls) for 30 iterations. The results show that, for problem 5, the *NumSimCall* has been reduced by 02 simulation calls and 03 iterations using the proposed A-NLPQL compared to the NLPQL; so the optimization process was much faster. Time taken for complete the converged solution for problem 5 through NLPQL is 03min and through A-NLPQL is 02min.

Problem 6: Definition (TP 4)

This example is a relative standard example taken from Jui-Yu Wu, 2012 [24]. TP 4 involves 13 decision variables, nine inequality constraints, and 26 boundary conditions, as follows:

Minimize $f(x) = 5\sum_{n=1}^{4} x_n - 5\sum_{n=1}^{4} x_n^2 - \sum_{n=5}^{13} x_n$

Subject to $g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0,$

$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0,$

$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0,$

$g_4(x) = -8x_1 + x_{10} \le 0,$

$g_5(x) = -8x_2 + x_{11} \le 0,$

$g_6(x) = -8x_3 + x_{12} \le 0,$

$g_7(x) = -2x_4 - x_5 + x_{10} \le 0,$

$g_8(x) = -2x_6 - x_7 + x_{11} \le 0,$

$g_9(x) = -2x_8 - x_9 + x_{12} \le 0$

Where $0 \le x_n \le 1, n = 1, 2, \ldots, 9,$

$0 \le x_n \le 100, n = 10, 11, 12,$

$0 \le x_{13} \le 1$

For the above given problem the optimization status are as shown below:

Optimization Status		Optimization Status	
Converged	Yes	Converged	Yes
Number of Iterations	13	Number of Evaluations	221
Number of Evaluations	292	Number of Domain Reductions	1
Number of Failures	0	Number of Failures	0
Size of Generated Sample Set	13	Size of Generated Sample Set	221
Number of Candidates	3	Number of Candidates	3

Fig. 18 Porb.6 optimization status of NLPQL (Left) and A-NLPQL (Right)

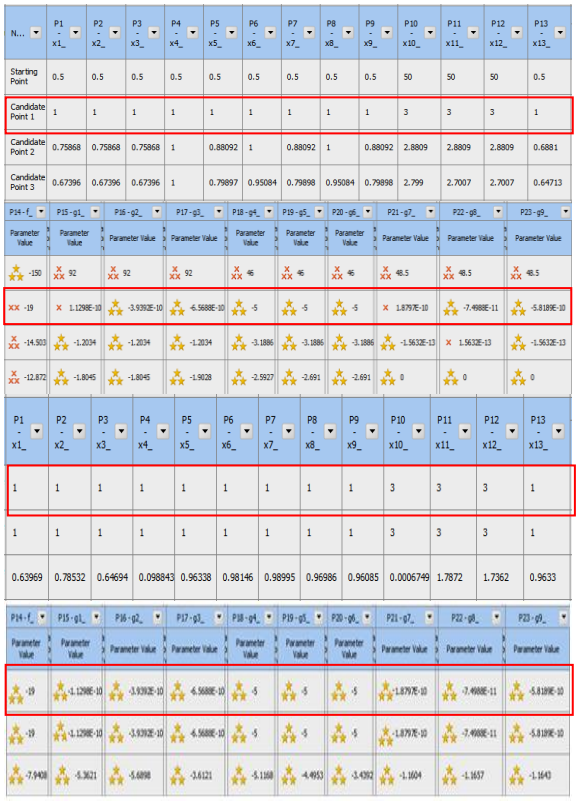

Fig. 19 NLPQL (Upper) and A-NLPQL (Lower) Candidates of converged solution of Prob.6

The Pareto frontiers from NLPQL and A-NLPQL, respectively, are non-convex as shown in Figure 20.

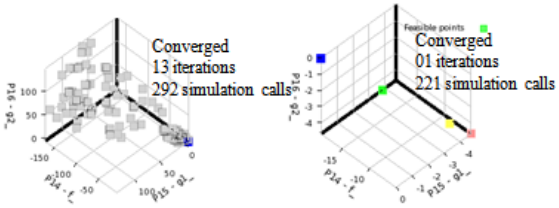

Fig. 20 Prob.6 Pareto solutions for NLPQL (Left) and A-NLPQL (Right)

Figure 19 shows the optimum solution calculated with NLPQL, where NLPQL is not able to satisfy $g_1(x)$ and $g_7(x)$ constraints with $f(x) = -19$ and A-NLPQL $x = (1,1,1,1,1,1,1,1,3,3,3,1)$, where $f(x) = -19$ with all constraints activated. Both constraints are active. Figure 20 shows a typical set of Pareto optimal solutions as obtained from the NLPQL and A-NLPQL. Figure 20 shows the *NumSimCall* (number of simulation calls) for 30 iterations. The results show that, for problem 6, the *NumSimCall* has been reduced by 71 simulation calls and 12 iterations using the proposed A-NLPQL compared to the NLPQL; so the optimization process was much faster. Time taken for complete the converged solution for problem 6 through NLPQL is 37min and through A-NLPQL is 13min.

5.1 Assessment of NLPQL and A-NLPQL

The obtained results for these six test examples show that the number of simulation calls (*NumSimCall*) used in the A-NLPQL is significantly fewer than the NLPQL, while the obtained Pareto solution for NLPQL method is comparable. Furthermore, as shown in Table 1, the A-NLPQL has smaller STD of the *NumSimCall* (based on 20 iteration runs) than the NLPQL, which indicates that compared to the NLPQL and the A-NLPQL has a more stable performance on the reduction of the *NumSimCall*.

Table 1 Statistics for the *NumSimCall*

S. No	Example	NLPQL		A-NLPQL	
		Mean	STD	Mean	STD
1	All Six examples	46.5	107.75	38.5	72.45

Based on the data in Table 2, the reduction of the *NumSimCall* for each example is calculated based on the mean and STD value. This calculation performing for the A-NLPQL over the NLPQL is also shown in Table 2.

Table 2 Reduction in the *NumSimCall*, *NumIterations* & Time required

S. No	Test Example	Reduction in *NumSimCall* $1-(\frac{A-NLPQL}{NLPQL})$	Reduction in *NumIterations* $1-(\frac{A-NLPQL}{NLPQL})$	Time (Min)		% Reduction in time
				NLPQL	*A-NLPQL*	
1	Problem 1	00%	50%	21	15	29%
2	Problem 2	76%	96%	25	10	60%
3	Problem 3	50%	80%	10	05	50%
4	Problem 4	34%	67%	12	07	42%
5	Problem 5	12%	75%	03	02	33%
6	Problem 6	24%	92%	37	13	65%

As shown in Table 2, on the average, the proposed A-NLPQL can save about 41% in the *NumSimCall*, 81% in the *NumIterations* and 52% time over the NLPQL. It is observed that the A-NLPQL outperforms the NLPQL and is more stable than the NLPQL, in terms of the number of simulation calls, for these six examples.

VII. CONCLUSION

We discussed about the various individual optimization and metamodeling techniques, by means of which an improved LHS and kriging assisted NLPQL technique was developed to enhance the computational efficiency of a single-objective optimization. In this project we proposed an enhanced NLPQL, called Adaptive Non Linear Programming Lagrangian (A-NLPQL) in which the online LHS and kriging-based metamodel is inter connected within a NLPQL.

NLPQL can add accuracy to the response surface-based approach, but is highly dependent on the quality of the starting point. A-NLPQL is an adaptive method that combines a DOE (LHS), an internal response surface (kriging), domain reduction and error prediction. It provides both accuracy and speed without needing prior results to initialize the optimization, and allows you to balance your available time and resources with your desired level of accuracy. While a Response Surface Optimization or the NLPQL algorithm may be sufficient for exploring problems that are convex or smooth, the A-NLPQL algorithm is a better optimization choice when you are not already very familiar with your problem.

The results show that, on the average. A-NLPQL outperforms both a conventional NLPQL and our recently developed A-NLPQL and has higher stability in terms of the number of simulation calls used in the optimization.

REFERENCES

[1] Schittkowski, K., "NLPQL: A Fortran subroutine for solving constrained nonlinear programming problems," *Annals of Operations Research*, Vol. 5, pp. 485-500, 1985.

[2] Farina, M., 2001, "A Minimal Cost Hybrid Strategy for Pareto Optimal Front Approximation," Evolutionary Optimization, an international journal on the internet, 3(1), pp. 41–52 _available online at www.jeo.org_.

[3] Wyss, G. D., and Jorgensen, K. H., 1998. "A User s Guide to LHS: Sandia's Latin Hypercube Sampling Software," Sandia National Laboratories Technical Report SAND98-0210, Albuquerque, NM.

[4] Hailong You, Xi"an, Maofeng Yang, Dan Wang, Jia, Xinzhang, "Kriging Model combined with latin hypercube sampling for surrogate modeling of analog integrated circuit performance", Quality of Electronic Design, 2009. ISQED 2009. Quality Electronic Design, Page(s): 554 - 558

[5] Papadrakakis, M., Lagaros, N., and Tsompanakis, Y., 1999, "Optimization of Large-Scale 3D Trusses Using Evolution Strategies and Neural Networks," Int. J. Space Struct., **14**(3), pp. 211–223.

[6] Hong, Y.-S., Lee, H., and Tahk, M.-J., 2003, "Acceleration of the Convergence Speed of Evolutionary Algorithms Using Multilayer Neural Networks," Eng. Optimiz., **35**(1), pp. 91–102.

[7] Mansour Keramat and Richard Kielbasa, "Latin Hypercube Sampling Monte Carlo Estimation of Average Quality Index for Integrated Circuits," Analog Integrated Circuits And Signal Processing, vol. 14, no. 1/2, pp. 131-142, 1997.

[8] Jones, D., Schonlau, M., and W. Welch, 1998. "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, Vol. 13, pp. 455-492.

[9] Hart, W. E., Giunta, A. A., Salinger, A. G., and van Bloemen Waanders, B. G., 2001. "An Overview of the Adaptive Pattern Search Algorithm and its Application to Engineering Optimization Problems," abstract in *Proceedings of the McMaster Optimization Conference: Theory and Applications*, McMaster University, Hamilton, Ontario, Canada

[10] Chen, J.-H., Goldberg, D. E., Ho, S.-Y., and Sastry, K., 2002, "Fitness Inheritance in Single-Objective Optimization," *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, July 9–13, Morgan Kaufmann, New York.

[11] Smith, R., Dike, B., and Stegmann, S., 1995, "Fitness Inheritance in Genetic Algorithms," *Proceedings of the ACM Symposiums on Applied Computing*, ACM, Nashville, TN, February 26–28, pp. 345–350.

[12] Jin, Y., Olhofer, M., and Sendhoff, B., 2001, "Managing Approximate Models in Evolutionary Aerodynamic Design Optimization," *Proceedings of IEEE Congress on Evolutionary Computation*, Vol. 1, pp. 592–599.

[13] Jin, Y., Olhofer, M., and Sendhoff, B., 2002, "A Framework for Evolutionary Optimization With Approximate Fitness Functions," IEEE Trans. Evol. Comput., **6**(5), pp. 48–494.

[14] Nain, P. K. S., and Deb, K., 2003, "Computationally Effective Search and Optimization Procedure Using Coarse to Fine Approximations," *Proceedings of the Congress on Evolutionary Computation (CEC-2003)*, Canberra, Australia, pp. 2081–2088.

[15] Nair, P. B., and Keane, A. J., 1998, "Combining Approximation Concepts With Genetic Algorithm-based Structural Optimization Procedures," *AIAA/ASME/ ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference and Exhibit, 39th, and AIAA/ASME/AHS Adaptive Structures Forum*, Long Beach, CA, Apr. 20–23, Collection of Technical Papers, Pt. 2, A98-25092 06-39, AIAA-1998-1912.

[16] Oduguwa, V., and Roy, R., 2002, "Single-Objective Optimization of Rolling Rod Product Design Using Meta-Modeling Approach," *Proceedings of the Genetic and Evolutionary Computation Conference*, New York, July 9–13, Morgan Kaufmann, New York, pp. 1164–1171.

[17] Jin, Y., 2005, "A Comprehensive Survey of Fitness Approximation in Evolutionary Computation," Soft Comput., **9**(1), pp. 3–12.

[18] Wang, G. G., and Shan, S., 2007, "Review of Metamodeling Techniques in Support of Engineering Design Optimization," ASME J. Mech. Des., **129**(4), pp. 370–380.

[19] Simpson, T. W., Peplinski, J., Koch, P. N., and Allen, J. K., 2001, "Metamodels for Computer-based Engineering Design: Survey and Recommendations," Eng. Comput., **17**(2), pp. 129–150.

[20] Simpson, T. W., Booker, A. J., Ghosh, D., Giunta, A. A., Koch, P. N., and Yang, R.-J., 2004, "Approximation Methods in Multidisciplinary Analysis and Optimization: A Panel Discussion," Struct. Multidiscip. Optim., **27**, pp. 302–313.

[21] Wilson, B., Cappelleri, D., Frecker, M., and Simpson, T. W., 2001, "Efficient Pareto Frontier Exploration Using Surrogate Approximations," Optim. Eng., **2**, pp. 31–50.

[22] Koch, P. N., Wujek, B. A., Golovidov, O., and Simpson, T. W., 2002, "Facilitating Probabilistic Multidisciplinary Design Optimization Using Kriging Approximation Models," *Proceedings of the Ninth AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Atlanta, GA, Sept. 4–6, AIAA 2002-5415.

[23] YongWang, Hui LIU, Zixing Cai and Yuren Zhou, "An orthogonal design based constrained evolutionary optimization algorithm", Taylor & Francis, Engineering Optimization. Vol. 39, No. 6, September 2007, 715–736

[24] Jui-Yu Wu, "Solving Constrained Global Optimization Problems by Using Hybrid Evolutionary Computing and Artificial Life Approaches" *Hindawi Publishing Corporation, Mathematical Problems in Engineering*, Volume 2012, Article ID 841410, 36 pages, doi:10.1155/2012/841410

[25] Kyoungwoo Park, Park-Kyoun Oh, Hyo-Jae Lim, The application of the CFD and Kriging method to an optimization of heat sink, *International Journal of Heat and Mass Transfer* 49 (2006) 3439–3447,

[26] R. Noorossana, Sam Davanloo Tajbakhsh and A. Saghaei, "An artificial neural network approach to Single-response optimization", *The International Journal of Advanced Manufacturing Technology*, Volume 40, Numbers 11-12, 1227-1238, DOI: 10.1007/s00170-008-1423-7 (2008)

[27] M. Oudjenea, L. Ben-Ayed, A. Delam´ezi`erb, J.-L. Batoz, "Shape optimization of clinching tools using the response surface methodology with Moving Least-Square approximation", *journal of materials processing technology* 209 (2009) 289–296

[28] Sangeeta Yadav , K. K. Pathak, Rajesh Shrivastava, Shape Optimization of Cantilever Beams Using Neural Network, *Applied Mathematical Sciences*, Vol. 4, 2010, no. 32, 1563 – 1572

[29] M. Ramu, V. Prabhu Raja, P. R. Thyla, M. Gunaseelan, "Design Optimization of Complex Structures Using Metamodels", *Jordan Journal of Mechanical and Industrial Engineering*, Vol. 4, Number 5, November 2010, ISSN 1995-6665, p.p. 653 - 664

[30] Muromaki, T.; Hanahara, K.; Nishimura, T.; Tada, Y.; Kuroda, S.; Fukui, T., "Single-Objective Shape Design of Crane-Hook Taking Account of Practical Requirement", *Institute of Materials, London England*, 2010, ISBN No- 1861250045, AIP Conf. Proc. / Volume 1233 / Issue, pp. 632-637

[31] Rashmi Uddanwadiker, Stress Analysis of Crane Hook and Validation by Photo-Elasticity, *Scientific research*, vol. 3, p.p.935-941, 2011

[32] Ossi Heinonen & Sami Pajunen, "Optimal design of stiffened plate using metamodeling techniques". *Rakenteiden Mekaniikka (Journal of Structural Mechanics)* (2011), Vol. 44, No 3, 2011, pp. 218-230

[33] Daryoush Safarzadeh, Daryoush Safarzadeh, Shamsuddin Sulaiman, Faieza Abdul Aziz, Desa Bin Ahmad and Gholam Hossein Majzoobi,"An investigation into the hook dynamics and effect of hook parameters on the sway angles in hydraulic cranes", *Scientific Research and Essays*, Vol. 6(6), pp. 1303-1316, 18 March, 2011

[34] Wilson, B., Cappelleri, D., Frecker, M., and Simpson, T. W., 2001, "Efficient Pareto Frontier Exploration Using Surrogate Approximations," Optim. Eng., 2, pp. 31–50.

[35] Shapour Azar, Brian J. Reynolds, Sanjay Narayanan, "comparison of two single objective optimization techniques with and within Genetic algorithms," Proceedings of the 1999 ASME Design Engineering Technical Conferences September 12-15, 1999, Las Vegas, Nevada, DETC99/DAC-8584

[36] Zhangjun Huang, Mingxu Ma and Chengen Wang, "An Archived Differential Evolution Algorithm for Constrained Global Optimization", International Conference on Smart Manufacturing Application, April. 9-11, 2008 in KINTEX, Gyeonggi-do, Korea

[37] Kirsch, U., 1981, Optimal Structural Design, McGraw-Hill Co., New York.

[38] Myers RH, Montgomery DC. Response surface methodology. New York: John Wiley & Sons Inc.; 1995.

[39] J. W. Dally and W. F. Riley, "Experimental Stress analysis," Springer Publisher, New York, 1993.

[40] P.K. Gupta and D.S. Hira, "Problems in Operations Research: Principles and Solutions", S. Chand & company LTD; 2011, ISBN:81-219-0281-9.

[41] Belegundu, Ashok D and Chandrupatla, Tirupathi R. *Optimization Concepts and Applications in Engineering.* s. l.: Pearson Education, 2005.

[42] H. A. Rothbart, "Mechanical Design Handbook: Measurement, Analysis, and Control of Dynamic Systems," McGraw-Hill, Columbus, 2006.

[43] J. Sacks, S. B. Schiller, W. J. Welch, "Design for computer experiments". Technometrics, 1989, Vol. 31, No. 1, pp. 41-47.