NASA/TM—2015–218733

Extension of an Object-Oriented Optimization Tool: User's Reference Manual

Chan-gi Pak and Samson S. Truong
Armstrong Flight Research Center, Edwards, California

March 2015

NASA/TM—2015–218733

Extension of an Object-Oriented Optimization Tool: User's Reference Manual

Chan-gi Pak and Samson S. Truong
Armstrong Flight Research Center, Edwards, California

National Aeronautics and
Space Administration

Armstrong Flight Research Center
Edwards, CA 93523-0273

March 2015

Abstract

The National Aeronautics and Space Administration Armstrong Flight Research Center has developed a cost-effective and flexible object-oriented optimization (O^3) tool that leverages existing tools and practices and allows easy integration and adoption of new state-of-the-art software. This object-oriented framework can integrate the analysis codes for multiple disciplines, as opposed to relying on one code to perform analysis for all disciplines. Optimization can thus take place within each discipline module, or in a loop between the O^3 tool and the discipline modules, or both. Six different sample mathematical problems are presented to demonstrate the performance of the O^3 tool. Instructions for preparing input data for the O^3 tool are detailed in this user's manual.

Nomenclature

ADS	automated design synthesis
AIC	aerodynamic influence coefficient
ATW	Aerostructures Test Wing
BBBC	Big-Bang-Big-Crunch
CG	center of gravity
D & DT	damage and damage tolerance
DOT	design optimization tools
FE	finite element
GA	genetic algorithm
LSCT	Low-Boom Supersonic Civil Transport
MAC	modal assurance criteria
MDAO	multidisciplinary design, analysis, and optimization
NASA	National Aeronautics and Space Administration
O^3	object-oriented optimization
V-f	velocity versus frequency
V-g	velocity versus damping
$\mathbf{X}_{CG}$	design variable vector at CG location
$\mathbf{X}_{GO}$	global optimum solution
$\mathbf{X}_i$	design variable vector
$\mathbf{XL}_i$	lower bound of design variable vector $\mathbf{X}_i$
$\mathbf{XU}_i$	upper bound of design variable vector $\mathbf{X}_i$
α	parameter limiting the size of the design space
β	parameter controlling the influence of $\mathbf{X}_{GO}$
ϑ	standard normal random number

Introduction

Supporting the Aeronautics Research Mission Directorate guidelines, the National Aeronautics and Space Administration (NASA) Armstrong Flight Research Center (AFRC) has developed a FORTRAN-based object-oriented optimization (O^3) tool (ref. 1). Over the past several years, an object-oriented Multi-disciplinary Design, Analysis, and Optimization (MDAO) tool, as shown in figure 1, has been developed and tested at NASA AFRC using the O^3 tool (ref. 2).

The O^3 tool provides a computational environment in which the optimizer can effectively receive objective and constraint function values from various disciplines through interface variables. The basic flow of the O^3 tool is shown in figure 2. An input deck is prepared prior to running the O^3 tool. The gray circle on the left in figure 2 represents how the O^3 tool processes incoming information in its optimization procedure. The red and green modules on the right in the figure each represent a different discipline of the MDAO tool as specified in figure 1. For simplicity, the sample problems provided in this user's manual are single-discipline runs.

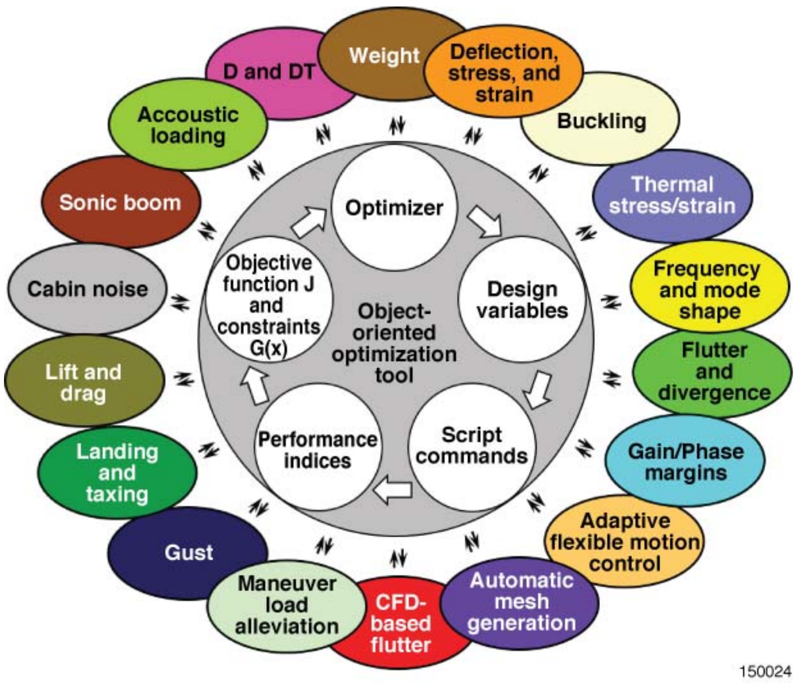


Figure 1. The object-oriented multidisciplinary design, analysis, and optimization tool.

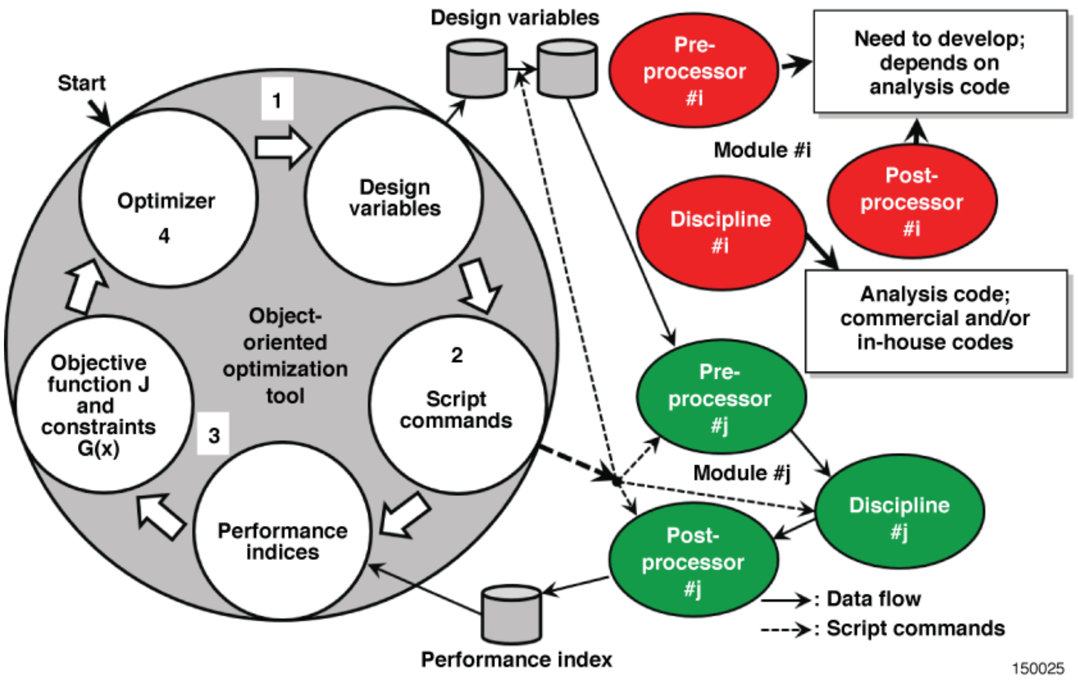


Figure 2. The object-oriented optimization tool flowchart.

The O^3 tool works in this way:

1. When starting the O^3 tool, the design variables are printed and saved in an external ASCII file that will be used to communicate with the analysis modules. The ASCII file cannot be used by two programs at the same time; the user must copy the ASCII file to a different file by way of of a user-defined script command.
2. The next step is submitting a pre-process job using a script command. A pre-processor code reads in the copy of the design variables generated by the O^3 tool and creates input data for an analysis code. Then, a script command submits an analysis job. The analysis code can be a commercial or an in-house code. Output files created by the analysis code are then post-processed. Using a post-processing code, required performance indices are computed.
3. The O^3 tool reads in performance indices from each discipline module, and computes the objective function and constraint functions.
4. An extreme value is searched using the objective and constraint function values, and the next design variable values will be computed. If the convergence criteria are satisfied, then terminate optimization; otherwise go back to step 1.

This computer program has been used for the development of a structural dynamic finite element (FE) model tuning tool, unsteady aerodynamic model tuning tools, and the MDAO tool. The structural dynamic FE model tuning tool, as shown in figure 3, has been used for FE model validation and updates for the following problems: the Quiet Spike^{TM} noseboom (refs. 3 and 4); the Aerostructures Test Wing 1 (ATW1) (ref. 5); the X-37 (The Boeing Company, Chicago, Illinois) drogue chute test fixture (ref. 6); the ATW2 (refs. 7, 8, and 9); and the X-56A (Lockheed Martin, Bethesda, Maryland) (ref. 10).

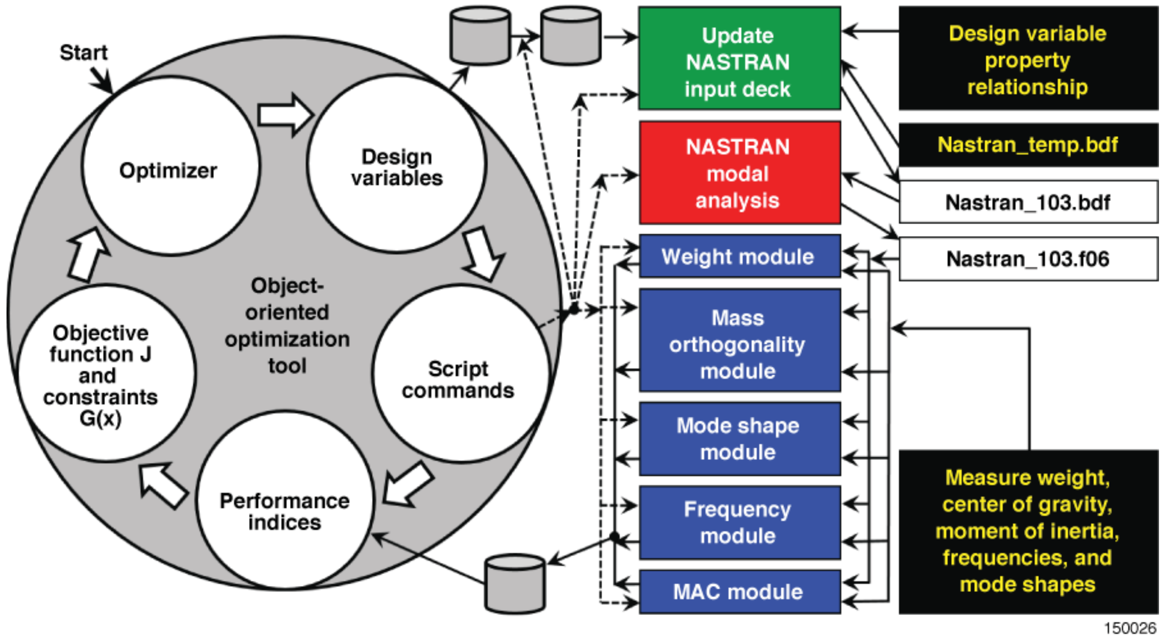

Start
Optimizer
Design variables
Object-oriented optimization tool
Script commands
Objective function J and constraints G(x)
Performance indices
Update NASTRAN input deck
NASTRAN modal analysis
Weight module
Mass orthogonality module
Mode shape module
Frequency module
MAC module
Design variable property relationship
Nastran_temp.bdf
Nastran_103.bdf
Nastran_103.f06
Measure weight, center of gravity, moment of inertia, frequencies, and mode shapes
150026

Figure 3. The structural dynamic finite element model tuning tool using the object-oriented optimization tool.

A computer code for unsteady aerodynamic model tuning based on the direct method has been developed using the O^3 tool together with the preprocessor, ZAERO (ZONA Technology Inc., Scottsdale, Arizona), and postprocessor codes, as shown in figure 4. Unsteady aerodynamic model tuning requires wind-tunnel or flight-flutter test data; this technique has been applied to validate an unsteady aerodynamic model of the ATW2 with respect to its flight-test data (ref. 11). An unsteady aerodynamic model tuning tool based on an indirect method also has been developed using the O^3 tool, as shown in figure 5; however, the new tool is not yet fully tested.

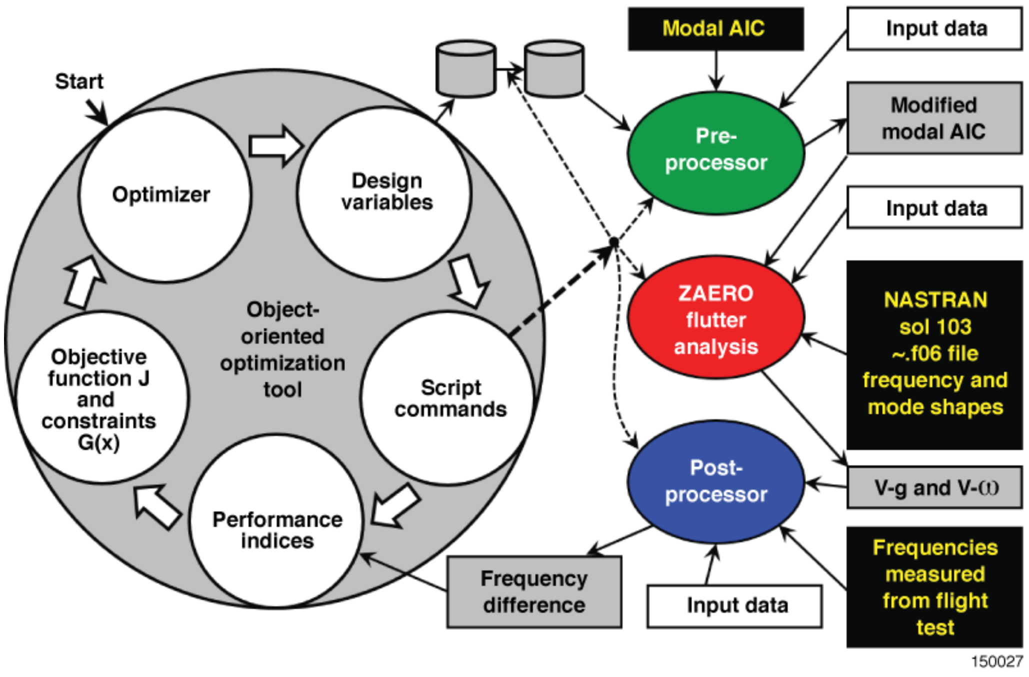
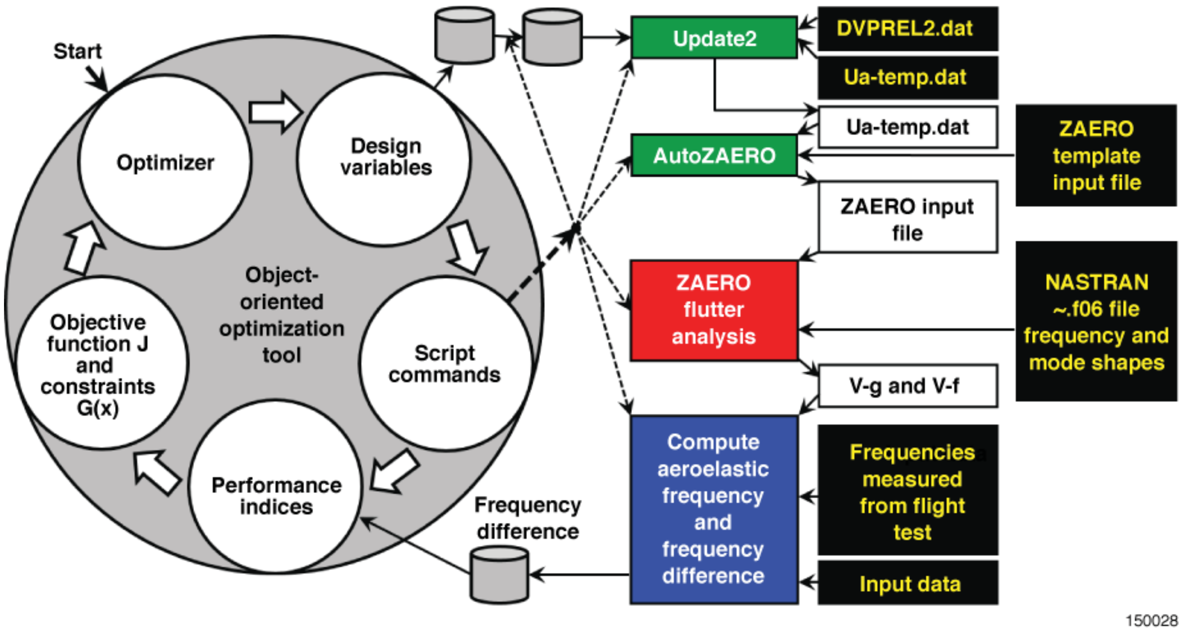


Figure 4. Unsteady aerodynamic model tuning based on the direct method using the object-oriented optimization tool.



Start
Optimizer
Design variables
Object-oriented optimization tool
Script commands
Objective function J and constraints G(x)
Performance indices
Update2
DVPREL2.dat
Ua-temp.dat
Ua-temp.dat
AutoZAERO
ZAERO template input file
ZAERO input file
ZAERO flutter analysis
NASTRAN ~.f06 file frequency and mode shapes
V-g and V-f
Compute aeroelastic frequency and frequency difference
Frequencies measured from flight test
Input data
Frequency difference
150028

Figure 5. Unsteady aerodynamic model tuning based on an indirect method using the object-oriented optimization tool.

The O^3 tool has been used to evaluate several real-world optimization problems including flutter characteristic improvement for the Ikhana MQ-9 Predator B (General Atomics Aeronautical Systems, Inc., San Diego, California) aircraft with a fire pod (ref. 2) and aeroelastic tailoring and flutter mass balancing studies to improve the performance of the X-56A aircraft (ref. 12). The same tool has been used for multidisciplinary design optimization studies of the N+2 Low-Boom Supersonic Civil Transport (LSCT) aircraft and a hybrid wing-body (HWB) aircraft, as shown in figures 6(a) and 6(b), respectively, in order to minimize weight and analyze aeroelastic effects.

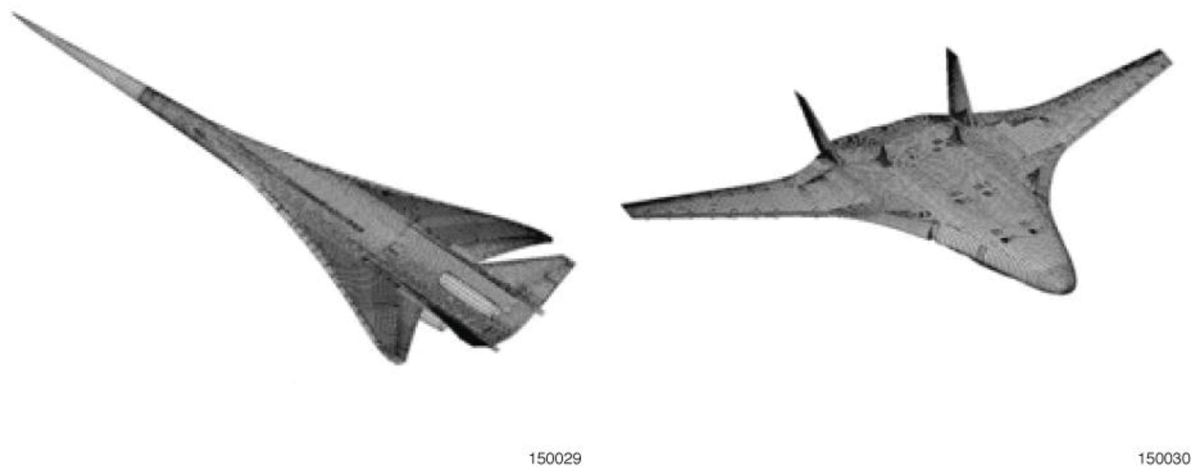


Figure 6. Sample multidisciplinary design optimization problems: (a) N+2 Low-Boom Supersonic Civil Transport aircraft; and (b) hybrid wing body aircraft.

The O^3 tool includes various optimizer programs, such as the gradient-based automated design synthesis (ADS) and design optimization tools (DOT) and global optimizers such as the genetic algorithm (GA) and the Big-Bang-Big-Crunch (BBBC) algorithm. The ADS and the BBBC are the newest additions to the modified O^3 tool. Further testing of the ADS and the BBBC algorithm are needed in order to validate its performance in the O^3 tool before utilizing it with appropriate applications. These algorithms are discussed in the “Optimizer Background” section below.

The primary objective of this user’s manual is to document changes made after the first version of the O^3 tool (ref. 1) was released. The quick reference manual is summarized in the appendix. The second objective is to test the ADS and the BBBC. The results obtained from these algorithms will be compared in the sample problems below to compare their performance.

Optimizer Background

In the O^3 tool, the user chooses an optimization methodology and defines objective and constraint functions from performance indices. The user also provides starting and side constraints for continuous as well as discrete design variables and external file names for performance indices which communicate between the O^3 tool and each analysis module. The performance indices can be total weight, safety factors, frequencies, lift, drag, noise levels, flutter speeds, gain and phase margins, et cetera.

Each discipline module consists of three sub-modules as shown in figure 2, that is, pre-processor, analyzer, and post-processor modules. The pre-processor module is used to create and update input files based on the design variable values provided by the O^3 tool before executing the analyzer module. The analyzer module can be a commercial or an in-house code for a specific discipline. Multi-fidelity analyzer modules can be incorporated with the current O^3 tool environment. The script command executes the analyzer module automatically. Users can use a script file to execute a series of analyses in sequential order. The post-processor module is used to post-process the output file computed from the analyzer module, and to compute the performance indices automatically.

Four optimizer codes are included in the O3 tool. The codes are divided into two categories: gradient-based optimizers and global optimizers. The gradient-based optimizer codes are ADS (ref. 13) and DOT (ref. 14); the global optimizer codes are GA (ref. 15) and BBBC (refs. 16, 17, and 18).

A. Automated Design Synthesis (ADS)

Automated design synthesis, which is the predecessor of DOT, is a public-domain numerical optimization program with several built-in optimizer algorithms that can be used to solve various optimization problems. Along with the inputs of the objective and constraint functions, ADS needs three pieces of information in order to acquire a solution: the strategy, the optimizer, and the one-dimensional search.

B. Design Optimization Tools (DOT)

The DOT is a commercial optimization code that can be used to solve a wide variety of nonlinear optimization problems. When the optimizer requires the values of the objective and constraint functions corresponding to a proposed design, it returns control to the O^3 tool. The O^3 tool calls the optimizer again to obtain the next design point; this iterative process continues until the optimizer returns a parameter to indicate that the optimum objective function is reached.

C. The Genetic Algorithm (GA)

The GA does not require gradient calculations and can be started with random seeds, eliminating some user input and allowing for solutions that may not be readily apparent even to experienced designers. In the case of multiple local minima problems, genetic algorithms are able to find the global optimum results, while gradient-based algorithms may converge to the local optimum value.

D. The Big-Bang-Big-Crunch (BBBC) Algorithm

The BBBC algorithm is a global optimization method that relies on one of the theories of the evolution of the universe, namely, the big-bang-big-crunch theory. The algorithm generates random design variables in the Big-Bang phase. These design variables are randomly selected over the entire design space except for one design variable vector. Current design configuration is also selected as the initial random design variables to guarantee the final design improvement.

The first step is the selection of the number of population (N) random design variable vectors $\mathbf{X}_i$ (i = 2, 3 … N) using a uniform random number generator such that equation (1):

$$\mathbf{XL}_i \leq \mathbf{X}_i \leq \mathbf{XU}_i \tag{1}$$

where, vectors $\mathbf{XL}_i$ and $\mathbf{XU}_i$ are the lower and upper bounds of design variable vectors $\mathbf{X}_i$, respectively. The current design configuration is saved in the design variable vector $\mathbf{X}_1$.

The second step shrinks those design variable vectors to a single representative design point via a center of gravity (CG) in the Big-Crunch phase. The CG is the weighted average of the candidate minimum solution with respect to the inverse of the objective function J_i (ref. 16) such that equation (2):

$$\mathbf{X}_{CG} = \frac{\sum_{i=1}^{N} \frac{\mathbf{X}_i}{J_i}}{\sum_{i=1}^{N} \frac{1}{J_i}} \tag{2}$$

where $\mathbf{X}_{CG}$ is the design variable vector at the CG location.

The third step is the computation of the new candidate design variables for the next Big-Bang step. These new candidate design variable vectors are normally distributed around the CG location, $\mathbf{X}_{CG}$, using a standard normal random number generator program as shown in equation (3):

$$\mathbf{X}_i^K = \mathbf{X}_{CG} + \frac{\vartheta\alpha(\mathbf{XU}_i - \mathbf{XL}_i)}{K+1} \tag{3}$$

where ϑ is the standard normal random number, α is the parameter limiting the size of the design space or search domain, and K is the number of current Big-Bang iterations (ref. 16). In order to improve computational efficiency, the modified form of equation (3), introduced by Camp (ref. 17), is used in this study and is given in equation (4):

$$\begin{aligned}\mathbf{X}_i^K = \beta\mathbf{X}_{CG} + (1-\beta)\{\gamma\mathbf{X}_{GO} + (1-\gamma)\mathbf{X}_i^{K-1}\} \\ + \frac{\vartheta\alpha(\mathbf{XU}_i - \mathbf{XL}_i)}{K+1}\end{aligned} \tag{4}$$

where β is the parameter controlling the influence of the global optimum solution $\mathbf{X}_{GO}$. Values of α, β, and γ can range from 0 to 1, but in the O^3 tool, the default value of α is 1, while the default value for β and γ is 0.7. The user when using BBBC decides whether to use the default values or to provide user-defined values, depending on the problem.

After a number of sequential Big-Bang and Big-Crunch processes, during which the distribution of the randomness within the design space during the Big-Bang step becomes smaller and smaller about the CG location computed during the Big-Crunch step, the algorithm converges to a solution. Studies have shown that this algorithm is capable of quick convergence even in long, narrow, parabolic-shaped flat valleys, or in the existence of several local minima.

Applications

Detailed instructions for preparing input data cards, DESVAR, DOPTPRM and INDEX, for executing the O^3 tool are explained in the appendix. Free sequence of these input data cards are used in the O^3 tool. The following information is provided through the use of each input command:

1. DESVAR cards for each design variable (appendix, section A.1)
    a. Continuous versus discrete design variable
    b. Starting value
    c. Lower and upper limit of design variable
    d. Name of table for a discrete design variable.

2. DOPTPRM card (appendix, section A.2)
    a. Optimization methodology
    b. Control variables for optimizer routines ADS, DOT, GA, and BBBC.

3. INDEX cards for each performance index (appendix, section A.3)
    a. Objective function versus constraint function
    b. Scaling factor in case of objective function
    c. Small allowable value in case of equality as well as inequality constraints
    d. Is gradient supplied by the user?
    e. Name of script file for the performance index (interface variable)
    f. Name of output file where the performance index is saved
    g. Name of script file for the gradient of the performance index (when supplied by user)
    h. Name of output file where the gradient of the performance index is saved (when supplied by user).

Six different mathematical optimization test problems (one maximization problem and five minimization problems) are presented herein as examples of the application and use of the O^3 tool with each of the four optimizers ADS, DOT, GA, and BBBC.

Depending on the type of optimizer selected in the O^3 tool, the objective function results may have an opposite sign to their value. The gradient optimizers ADS and DOT are set to minimization, while the global optimizers, GA and BBBC, are set by default to maximization. Thus, if the user desires a maximization search when using the gradient optimizers or a minimization search when using the global optimizers, the optimizers will multiply the objective function solution by -1. If a gradient optimizer (such as ADS or DOT) is selected and maximization is used, the result of the objective function will have an opposite sign to that of the exact solution, for example, if ADS or DOT outputs a negative result for the objective function, the correct result is actually positive and vice versa. Similarly, if a global optimizer (such as GA or BBBC) is selected and minimization is used, the sign of the objective function result for this case is the opposite of what is outputted, for example, if GA or BBBC outputs a negative objective function value, the value is actually positive and vice versa. For convenience, all of the objective function outputs, *f(x,y)*, for the sample mathematical optimization test problems in this user's manual have already incorporated these appropriate sign changes for consistency when providing these results in the corresponding tables below.

A. Sample 1: The Maximization Problem

For the objective function in equation (5), use maximization to find the global maximum:

$$f(x,y) = \cos^2(rr \cdot 9\pi)\, e^{\left(\frac{-rr^2}{0.15}\right)} \qquad (5)$$

where *rr* is defined in equation (6) as

$$rr = \sqrt{(0.5-x)^2 + (0.5-y)^2} \qquad (6)$$

The surface plot for this objective function is shown in figure 7.

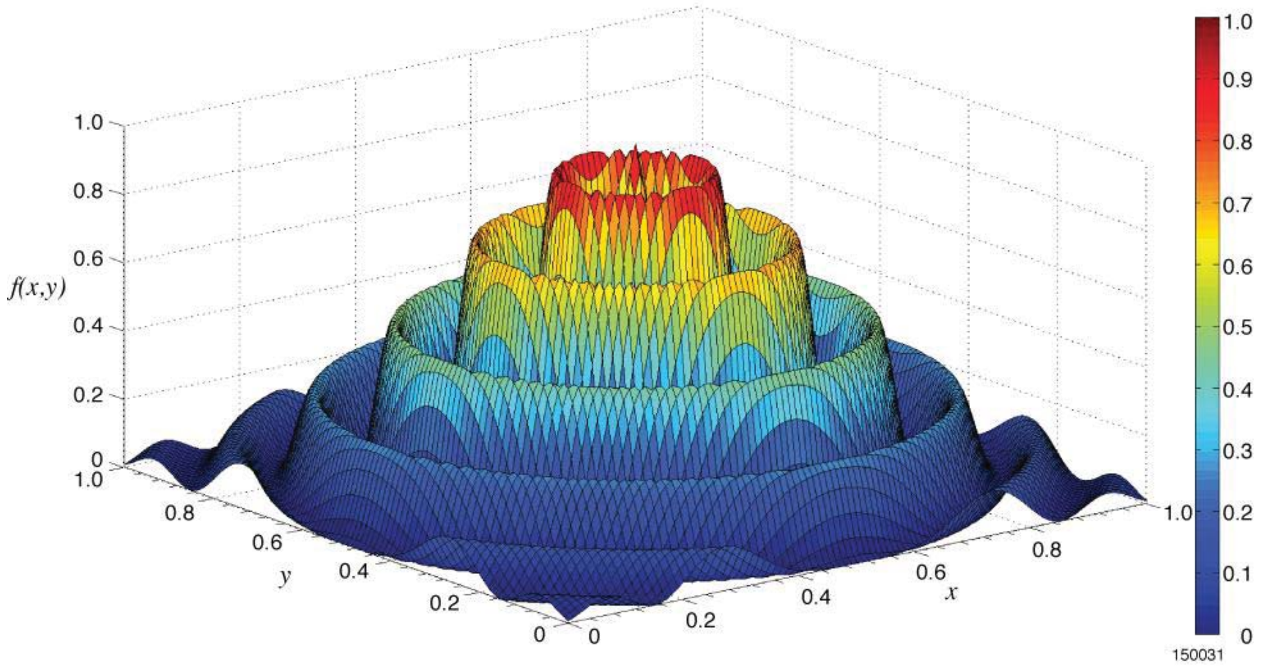


Figure 7. The surface plot for the sample maximization problem.

Using the O^3 tool, ADS, DOT, GA, and BBBC are herein each used to determine the two design variables, that is, the values of *x* and *y*, which provide the global maximum for the above objective function.

1) *Solution using the Automated Design Synthesis (ADS)*

Unlike global optimization searches, such as GA and BBBC, ADS is an optimization methodology that uses the gradient-based search technique. Depending on the initial design variable selected, the correct output design variables that describe the global minimum or maximum may or may not result. Rather, ADS tends to find local minima or maxima and output their corresponding design variables based on that particular result.

The input data card for the ADS simulation with a starting value of 0.0 for both design variables is shown below:

DOPTPRM	IOPT2		3	ICAS	1	NRWK	1000		
+	NRIWK		500	IGRAD	0	DELOBJ	0.00001		
+	IOPT		3	IONED	1				
DESVAR	1	0	0.0		0.0	1.0			1.0
DESVAR	2	0	0.0		0.0	1.0			1.0
INDEX	1		1	0	1.0	0			
+	Sample Problem 1: Objective Function								
+	f								
+	f.dat								

In order to select the ADS optimization methodology, the IOPT2 parameter, which tells the O^3 tool which optimizer to use, is set to 3, and because this is a maximization problem, ICAS, the flag parameter for a minimization (ICAS = 0) or maximization (ICAS = 1), is set to 1. Detailed information discussing IOPT2 and ICAS can be found in section A.2 of the appendix. The default values of the dimensioned work array sizes NRWK and NRIWK are 300 and 1000, respectively. For larger, more complex problems, these two numbers should be increased by the user. Gradient calculation control is represented by IGRAD; if the user wishes the optimizer to calculate the gradients, IGRAD is set to 0; if IGRAD is set to 1, the user must provide the gradients. The maximum relative change between two consecutive iterations desired to indicate convergence is represented by DELOBJ. The default value for DELOBJ is 0.001, but for this example, it is set at a more strict value of 0.00001. Unique to ADS, IOPT, and IONED are the ADS optimizer and one-dimensional search algorithm to be used; these two parameters are further detailed and explained in section A.2 of the appendix.

For this simulation, starting values of 0.0, 0.45, 0.48, and 1.0 were used for both design variables, with lower and upper bound limits between 0.0 and 1.0. For this example and subsequent examples in this section, both design variables (DESVAR) had a scaling factor of 1.0, as shown at the far right of the input card. Under the INDEX card, “f” represents the script or batch file, f.bat, which executes the executable file of the objective function shown in equation (1), while “f.dat” represents the external output file for that particular performance index. The O^3 tool outputs an external file, design_variables, which cannot be shared with any other executable and as a result, a copy of the external file, design_var, is created in the script file, f.bat. This process will apply to any application and will be the case for all sample cases in this user’s manual. The appendix contains detailed definitions and descriptions on how these input cards are compiled.

The results for this ADS simulation are shown in table 1. The exact solution to this mathematical problem is also shown in the table as the “Exact Global.” Out of the four sample runs using ADS for this problem, only one run with starting values of $x = 0.48$ and $y = 0.48$ converged close to the exact solution. Since ADS is a gradient-based optimizer, it has a tendency to search for local minima or maxima. Only if the starting design variable value is extremely close to the exact solution will the optimum solution of the optimizer converge to the desired value. This outcome is not ideal, because in most cases the exact solution isn’t known ahead of time. A gradient-based optimizer will likely yield an undesirable solution to this kind of problem.

Table 1. Results for the maximization problem.

Maximum	Initial *x*	Initial *y*	Optimum *x*	Optimum *y*	Objective function *f(x,y)*	Number of function calls
Exact Global	-	-	0.5	0.5	1.0	-
ADS	0.00	0.00	0.03245	0.03245	0.05294	55
ADS	0.45	0.45	0.42212	0.42212	0.92162	39
ADS	0.48	0.48	0.50004	0.50004	1.00000	35
ADS	1.00	1.00	0.96753	0.96753	0.05294	55
DOT	0.00	0.00	0.03246	0.03246	0.05294	23
DOT	0.45	0.45	0.42206	0.42206	0.92162	23
DOT	0.48	0.48	0.50000	0.50000	1.00000	23
DOT	1.00	1.00	0.96747	0.96747	0.05294	21
GA	0.00	0.00	0.49986	0.49999	0.99998	4,000
GA	0.45	0.45	0.49894	0.49958	0.99894	2,800
GA	0.48	0.48	0.50112	0.50109	0.99802	4,000
GA	1.00	1.00	0.49994	0.50013	0.99998	2,000
BBBC	0.00	0.00	0.49945	0.50039	0.99963	1,800
BBBC	0.45	0.45	0.49936	0.50037	0.99956	1,800
BBBC	0.48	0.48	0.49930	0.50045	0.99944	1,800
BBBC	1.00	1.00	0.49953	0.50047	0.99964	1,800

2) *Solution using the Design Optimization Tools (DOT)*

Like ADS, the DOT optimization methodology uses the gradient-based search technique. Depending on the optimization problem and the initial design variable selected, the output design variable may or may not be the global minimum or maximum, and an incorrect solution may be provided instead. Unlike ADS, however, DOT has a faster convergence, and for most problems will provide a more accurate solution than ADS.

The input data card for the DOT simulation with a starting value of 0 for both design variables is shown below:

DOPTPRM	IOPT2		1	ICAS	1	MAXDOT	1		
+	NRWK		1000	NRIWK	500	NGMAX	2		
+	IGRAD		0	DELOBJ	0.00001				
DESVAR	1	0	0.0		0.0	1.0			1.0
DESVAR	2	0	0.0		0.0	1.0			1.0
INDEX	1		1	0	1.0	0			
+	Sample Problem 1: Objective Function								
+	f								
+	f.dat								

The IOPT2 parameter is changed to 1 in order to select the DOT optimization methodology. The number of maximum DOT optimizations desired is represented by MAXDOT; for this problem, MAXDOT is set to its default value of 1. In addition, NGMAX represents the number of retained constraints for gradient calculations; NGMAX is further explained in detail in section A.2 of the appendix. The same starting design variable values are used in this run in order to compare performance with ADS, GA, and BBBC.

The results for this DOT simulation are shown in table 1. Notice that the DOT gradient optimizer has a fast convergence, in slightly more than 20 optimization iterations, as opposed to the convergence rate of ADS, GA, or BBBC. Moreover, DOT gives nearly identical results to those produced by ADS. For the four different DOT cases with different starting design variable *x* and y values, however, the resulting optimum results were all different. This result demonstrates the weakness of the gradient-based optimizer, as it tends to find local maxima or minima as opposed to global maxima or minima. Out of the four DOT sample runs, three did not converge to the exact value but rather to a local maximum near the starting design variable input. It is still possible to obtain optimum design variable results around the global optimum, but the starting design variable value must be close to the solution. In this case, a starting design variable value of (0.48, 0.48) for *x* and *y*, respectively, did converge to the exact solution. In most situations, the solution won't be known, and using the DOT optimizer will likely yield inaccurate global optimum results.

3) *Solution using the Genetic Algorithm (GA)*

The input data card for the GA simulation with a starting value of 0.0 for both design variables is shown below:

DOPTPRM	IOPT2		2	ICAS	1	IPOP	200		
+	IGEN		20	EPSOBJ	0.000001				
DESVAR	1	0	0.0		0.0	1.0			1.0
DESVAR	2	0	0.0		0.0	1.0			1.0
INDEX	1		1	0	1.0	0			
+	Sample Problem 1: Objective Function								
+	f								
+	f.dat								

The IOPT2 parameter is changed to 2 in order to select the GA optimization methodology. In addition, IPOP and IGEN values dictate how many desired optimizations to be performed. Furthermore, a convergence accuracy criterion was specified to be 10^{-6} for the EPSOBJ value. Note that the EPSOBJ parameter is only used for GA and BBBC. The same starting design variable values are used again in this run in order to compare its performance with that of the DOT.

Results for this GA simulation are shown in table 1. For this simulation, 4,000 optimization iterations (= IPOP × IGEN) were requested in each run. If the solution converges for five consecutive generations with identical results, the optimization will terminate early. If the user wishes to change the number of consecutive generations with which the solution must converge, the user has the option to change the default value of 5 in the NCONV parameter, which is only available for the global optimizers GA and BBBC. Notice that for each run, the optimum *x* and *y* values converged almost to the exact solution of (0.5, 0.5) and the resulting objective function was near 1.0. Even with a starting design variable value at the lower- and upper-boundary search limits, GA was able to find the global optimum. This result demonstrates one of the benefits of using a global optimizer, as it tries to find the global minimum or maximum, unlike the gradient-based optimizers ADS and DOT. If the EPSOBJ value is more restrictive, or lower, while increasing the number of optimization iterations, a more accurate result for the design variables can be obtained.

4) *Solution using the Big-Bang-Big-Crunch Algorithm (BBBC)*

The input data card for the BBBC with a starting value of 0.0 for both design variables is shown below:

DOPTPRM	IOPT2		6	ICAS	1	NPOP	200		
+	NBANG		20	BETA	0.70	GAMMA	0.70		
+	EPSOBJ		0.000001						
DESVAR	1	0	0.0		0.0	1.0			1.0
DESVAR	2	0	0.0		0.0	1.0			1.0
INDEX	1		1	0	1.0	0			
+	Sample Problem 1: Objective Function								
+	f								
+	f.dat								

To use the BBBC optimization methodology, the IOPT2 parameter is changed to 6. Similar to the GA optimizer, NPOP and NBANG will help dictate the number of optimization iterations (= NPOP × NBANG) desired. In this case, 4,000 iterations were requested, but the solution converged in only 1,800 optimization iterations for all of the BBBC runs, as shown in table 1. In addition, values of BETA (β) and GAMMA (γ) were both set to 0.70, which is their default value, since it is best for the O^3 tool to determine the global maximum due to the complexity of the surface plot for this optimization function, as shown in figure 7. The two scaling parameters, BETA and GAMMA, are defined in section A.2 of the appendix. Using the same initial starting values of (0.0, 0.0), (0.45, 0.45), (0.48, 0.48) and (1.0, 1.0) for the two design variables, as in the previous cases, the BBBC optimizer resulted in better results at the extremities of the design variable search limit than did ADS and DOT. In addition, ADS as well as DOT perform better when the starting initial *x* and *y* values are closer to the exact solution when compared to BBBC. When comparing to GA, BBBC converged in a shorter amount of time due to its convergence to the approximate solution in fewer iterations. The GA more closely approximated the exact solution at the extreme ends of the search range at (0.00, 0.00) and (1.00, 1.00), while BBBC performed better when the initial starting design variables were closer to the exact solution at (0.45, 0.45) and (0.48, 0.48).

B. Sample 2: Beale's Function

For the objective function in equation (7), find the global minimum.

$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2 \tag{7}$$

The surface plot for Beale's function is shown in figure 8.

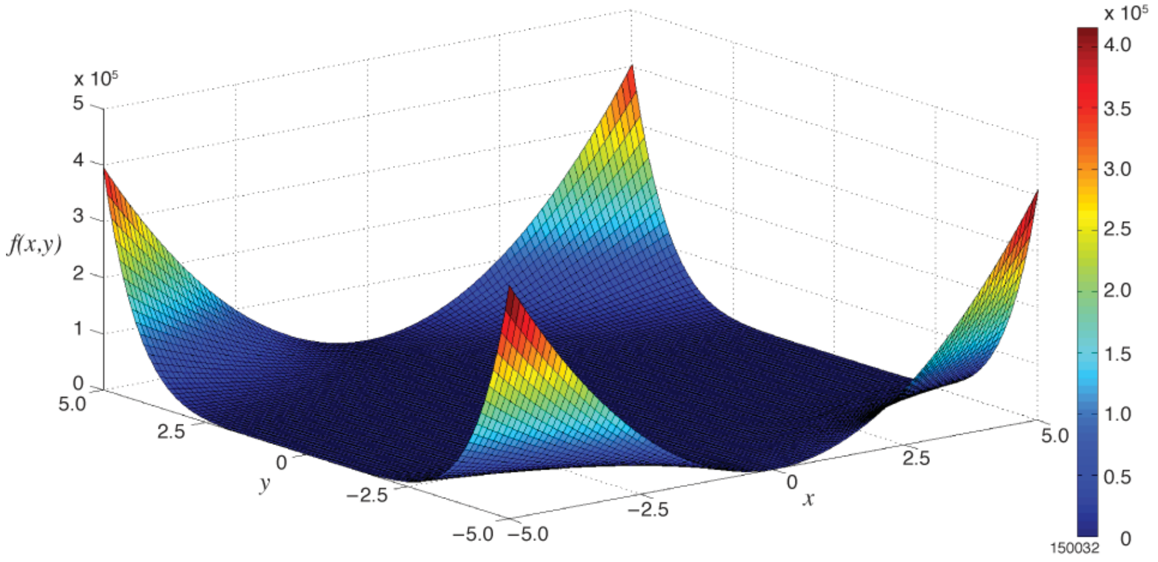


Figure 8. The surface plot for Beale's function.

Input cards for the ADS, DOT, GA, and BBBC simulations are similar to the previous example. The goal of this problem is to find the global minimum, that is, ICAS = 0. Upper- and lower-boundary search limits for the two design variables were between -5.0 and +5.0. The results of the optimization runs for Beale's function are shown in table 2. Note that the global minimum solution to Beale's function is such that *f(3.0,0.5) = 0.0*, as shown in the first row of table 2.

Table 2. Results for Beale's function.

Minimum	Initial *x*	Initial *y*	Optimum *x*	Optimum *y*	Objective function *f(x,y)*	Number of function calls
Exact Global	-	-	3.0	0.5	0.0	-
ADS	-2.50	-1.00	-2.27815	1.32433	1.01374	54
ADS	0.00	0.00	2.82512	0.44602	0.00696	123
ADS	2.95	0.45	2.94872	0.48701	4.46E-04	52
ADS	5.00	5.00	3.46857	0.51393	0.26497	47
DOT	-2.50	-1.00	2.90498	0.46978	0.00226	34
DOT	0.00	0.00	2.96851	0.49172	1.67E-04	60
DOT	2.95	0.45	2.99598	0.49890	2.83E-06	28
DOT	5.00	5.00	3.29427	0.56352	0.01022	28
GA	-2.50	-1.00	2.95971	0.47277	0.00646	4,000
GA	0.00	0.00	3.00594	0.48972	0.00315	3,200

GA	2.95	0.45	3.00279	0.49989	1.61E-05	4,000
GA	5.00	5.00	3.04503	0.51440	6.01E-04	4,000
BBBC	-2.50	-1.00	3.00199	0.49984	1.03E-05	2,400
BBBC	0.00	0.00	3.00199	0.49985	1.03E-05	2,400
BBBC	2.95	0.45	2.98123	0.49517	5.80E-05	2,200
BBBC	5.00	5.00	3.00321	0.50127	6.87E-06	4,200

For this example, four different pairs of starting values were used in order to compare the performances of ADS, DOT, GA, and BBBC. Notice that when using ADS and DOT, the closer the initial design variable values were to the solution, the more accurate the answer was, as shown when x = 2.95 and y = 0.45. When the initial starting design variable values were further from the exact solution, solutions to ADS and DOT tended to deviate away from the correct answer. In this example, DOT was able to closely estimate the solution in all four runs as the local minimum in this case was also the global minimum. Between ADS and DOT, DOT performed better in all four sample runs in approximating the exact solution to Beale's function and with a faster convergence. Using GA resulted in more accurate solutions, regardless of the initial starting design variable value. With 4,000 optimization iterations desired, solution convergence was slower than that of the gradient-based optimizers, but as in the previous example, the resulting solutions were much better when compared to the exact solution. For the BBBC runs, values of β and γ in the input cards were set to 0.05 and 1.0, respectively. In this example, the surface plot of the optimization function, shown in figure 8, was known in advance. Unlike the previous maximization function, an idea of the global optimum can be inferred from figure 8. Thus, more emphasis was placed on the $\mathbf{X}_{GO}$ parameter in equation (4) by setting β to a low value. The first two BBBC runs and the fourth run resulted in better design variable and objective function values than did ADS, GA, or DOT. The exception is the third run, with initial starting design variable values of (2.95, 0.45), in which GA produced a more accurate result. Overall, the global optimizers GA and BBBC performed better in estimating the exact solution to this function.

C. Sample 3: Booth's Function

For the objective function in equation (8), find the global minimum.

$$f(x,y) = (x+2y-7)^2 + (2x+y-5)^2 \tag{8}$$

The surface plot for Booth's function is shown in figure 9.

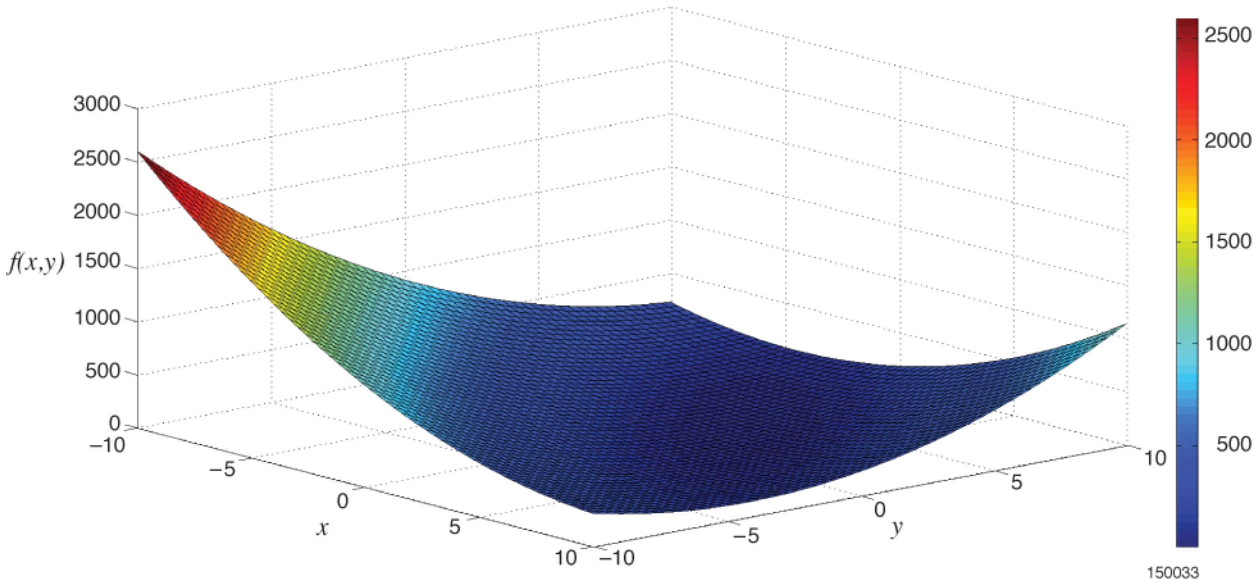


Figure 9. The surface plot for Booth's function.

Input cards for this minimization problem are similar to those used in the previous example. Upper- and lower-boundary search limits for the two design variables were between -10.0 and +10.0. The results of the optimization runs for Booth's function are shown in table 3. Note that the global minimum solution to Booth's function is such that *f(1.0, 3.0) = 0.0,* as shown in the first row of table 3.

Table 3. Results for Booth's function.

Minimum	Initial *x*	Initial *y*	Optimum *x*	Optimum *y*	Objective function *f(x,y)*	Number of function calls
Exact Global	-	-	1.0	3.0	0.0	-
ADS	-10.0	-10.0	0.99924	3.00164	6.34E-06	62
ADS	0.00	0.00	1.00423	2.99581	3.55E-05	78
ADS	0.95	2.95	1.00899	2.99057	1.71E-04	55
ADS	10.0	10.0	0.99427	2.99625	4.06E-04	66
DOT	-10.0	-10.0	1.00051	2.99958	4.75E-07	32
DOT	0.00	0.00	1.00019	2.99984	6.37E-08	34
DOT	0.95	2.95	1.00004	2.99992	1.50E-08	28
DOT	10.0	10.0	1.00004	2.99996	3.77E-09	26
GA	-10.0	-10.0	0.96960	3.06004	0.00804	3,200
GA	0.00	0.00	0.99008	2.99584	9.09E-04	3,800
GA	0.95	2.95	1.00764	2.99144	1.35E-04	3,600
GA	10.0	10.0	0.98886	3.00894	2.23E-04	3,200
BBBC	-10.0	-10.0	0.98097	3.01507	6.52E-04	3,200
BBBC	0.00	0.00	0.99136	3.00791	1.39E-04	2,200
BBBC	0.95	2.95	1.00840	2.99376	1.28E-04	3,600
BBBC	10.0	10.0	0.99441	3.02014	0.00128	2,400

As before, four different pairs of starting design variable values were selected in order to compare the performances of the various algorithms within the O^3 tool. Due to the relative absence of local minima and maxima for Booth's function, ADS and DOT did not encounter the problem experienced in the maximization example earlier as shown in the results in table 1. Rather, both of the gradient-based optimizers performed well in estimating the global minimum solution within and at the extreme ends of the design variable search limits. Likewise, GA was able to converge nearly to the exact global minimum solution, regardless of the initial design variable values within the search limits; however, it required more iterations in order to converge to the desired solution and the resulting objective function was not as small when compared to that of ADS or DOT. For BBBC, values of β and γ in the input cards were set to 0.05 and 1.0, respectively. The BBBC results were similar to those of GA, but DOT performed better than BBBC for this function. Overall, these four optimizers were able to pinpoint the exact global minimum solution of (1.0, 3.0) for Booth's function.

D. Sample 4: The Easom Function

For the objective function in equation (9), find the global minimum:

$$f(x, y) = -\cos(x)\cos(y)\, e^{-[(x-\pi)^2 + (y-\pi)^2]} \tag{9}$$

The surface plot of the Easom function is shown in figure 10.

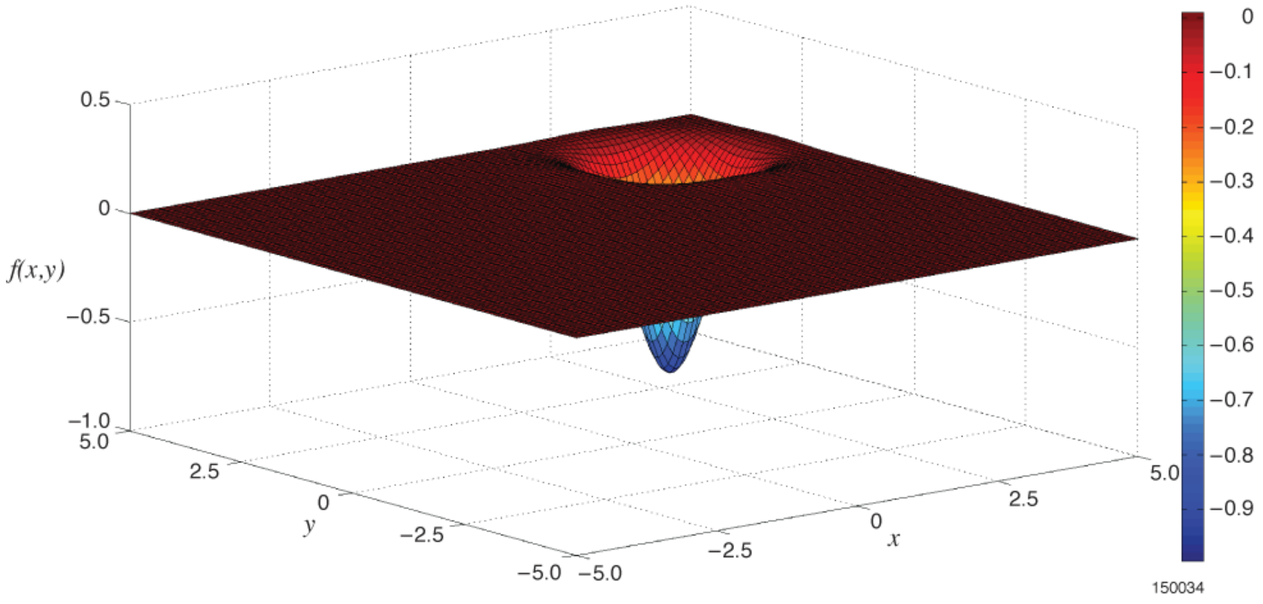


Figure 10. The surface plot for the Easom function.

The lower- and upper-boundary search limits for the two design variables for this function were set between -5.0 and +5. The results of the optimization runs based on four different algorithms for the Easom function are shown in table 4. Note that the global minimum solution for the Easom function is such that *f(π, π) = -1.0*. For the BBBC runs, β and γ were both set to their default value of 0.7.

Table 4. Results for the Easom function.

Minimum	Initial *x*	Initial *y*	Optimum *x*	Optimum *y*	Objective function *f(x,y)*	Number of function calls
Exact Global	-	-	3.15149	3.14159	-1.0	-
ADS	-5.00	-5.00	-5.00000	-5.00000	-2.14E-59	2
ADS	0.00	0.00	3.14035	3.14035	-1.00000	55
ADS	3.00	3.00	3.14155	3.14155	-1.00000	55
ADS	5.00	5.00	5.00000	5.00000	-8.05E-05	2
DOT	-5.00	-5.00	-5.00000	-5.00000	-2.14E-59	3
DOT	0.00	0.00	0.00000	0.00000	-2.68E-09	3
DOT	3.00	3.00	3.14159	3.14159	-1.00000	19
DOT	5.00	5.00	3.14121	3.14121	-1.00000	21
GA	-5.00	-5.00	3.14151	3.14002	-1.00000	4,000
GA	0.00	0.00	3.13764	3.14325	-0.99997	4,000
GA	3.00	3.00	3.16530	3.13987	-0.99915	4,000
GA	5.00	5.00	3.15290	3.14323	-0.99980	4,000
BBBC	-5.00	-5.00	3.13116	3.14720	-0.99979	3,000
BBBC	0.00	0.00	3.13116	3.14720	-0.99979	3,000
BBBC	3.00	3.00	3.13504	3.14164	-0.99994	4,200
BBBC	5.00	5.00	3.13116	3.14721	-0.99979	3,000

Similar to the previous sample maximization problem, the Easom function again demonstrated the fundamental weakness of ADS and of DOT as gradient optimizers. As shown in Table 4, ADS did not perform well at the lower and upper extremities of the design variable search limit, while DOT did not perform well for its first two runs when the initial starting design variable values were far away from the exact solution. Notice that in these cases the optimization runs quit after a couple of iterations since gradient values are already zero at the starting points. The ADS performed well at starting design variable points (0.0, 0.0) and (3.0, 3.0), while DOT performed well at (3.0, 3.0) and at (5.0, 5.0) when it came to acquiring the approximation to the exact solution to the Easom function. On the other hand, the global optimizers, GA and BBBC, performed well in approximating the exact Easom function solution of (π, π), regardless of the starting design variable values. For the Easom function, using the global optimizers would guarantee a close approximation of the exact solution, as opposed to using the gradient-based optimizers, particularly at the lower and upper ends of the design variable search limit.

E. Sample 5: The Goldstein-Price Function

For the objective function in equation (10), find the global minimum.

$$\begin{gathered} f(x,y) = [1 + (x+y+1)^2(19 - 14x + 3x^2 - 14y + 6xy + 3y^2)] \cdot \\ [30 + (2x-3y)^2(18 - 32x + 12x^2 + 48y - 36xy + 27y^2)] \end{gathered} \tag{10}$$

The surface plot for the Goldstein-Price function is shown in figure 11.

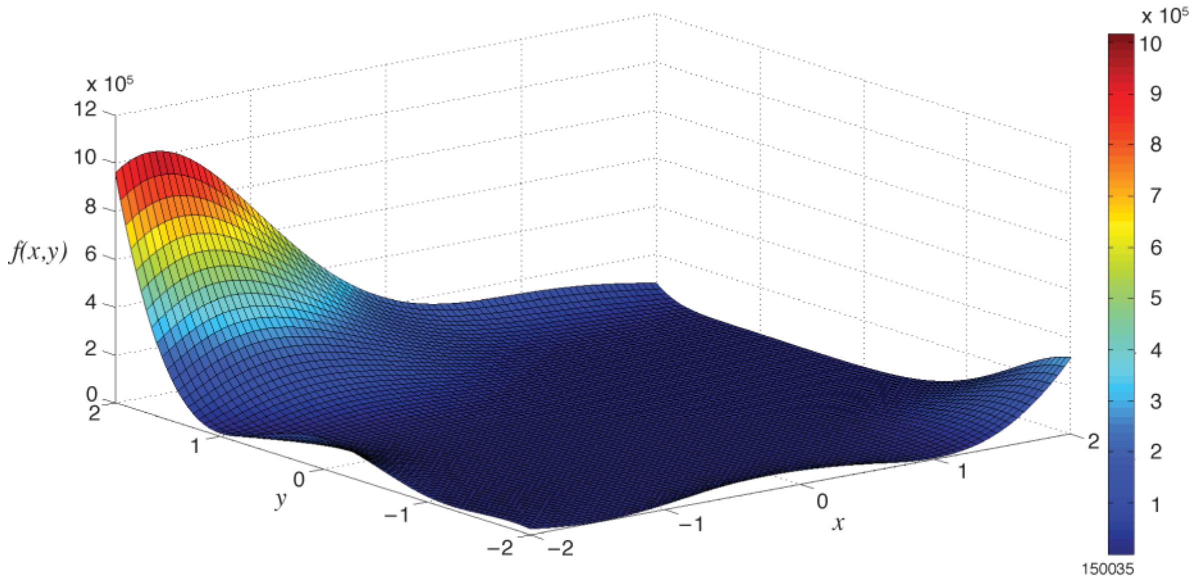


Figure 11. The surface plot for the Goldstein-Price function.

For the Goldstein-Price function, lower- and upper-boundary search limits for the two design variables were set between -2.0 and +2.0. The results of the optimization runs using the four different algorithms for the Goldstein-Price function are shown in table 5. Note that the global minimum solution for the Goldstein-Price function is such that *f(0, -1) = 3.0*. For the BBBC runs, a value of 0.05 and 0.98 were assumed for β and γ, respectively.

Table 5. Results for the Goldstein-Price function.

Minimum	Initial *x*	Initial *y*	Optimum *x*	Optimum *y*	Objective function *f(x,y)*	Number of function calls
Exact Global	-	-	0	-1	3.0	-
ADS	-2.00	-2.00	-1.50403	-2.00000	1,977.60	29
ADS	0.50	-0.50	-0.09667	-1.03285	5.45030	48
ADS	0.05	-1.05	-0.00324	-1.00032	3.00247	51
ADS	2.00	2.00	2.00000	0.32335	92.3673	41
DOT	-2.00	-2.00	-0.00454	-1.00377	3.00767	48
DOT	0.50	-0.50	-2.10E-04	-1.00033	3.00004	39
DOT	0.05	-1.05	9.43E-06	-0.99999	3.00000	31
DOT	2.00	2.00	-0.00091	-0.99927	3.00058	46
GA	-2.00	-2.00	-0.00732	-0.99887	3.01577	4,000
GA	0.50	-0.50	-4.60E-04	-1.00088	3.00030	3,800
GA	0.05	-1.05	-6.12E-04	-1.00145	3.00081	2,800
GA	2.00	2.00	-0.00970	-1.02212	3.19572	3,800
BBBC	-2.00	-2.00	-0.00132	-1.00161	3.00110	2,400
BBBC	0.50	-0.50	-0.00110	-1.00111	3.00057	2,400
BBBC	0.05	-1.05	2.26E-04	-1.00013	3.00003	3,600
BBBC	2.00	2.00	4.45E-05	-1.00061	3.00017	2,400

Among the four optimizers, ADS performed the worst in approximating the exact solution to the Goldstein-Price function. Three of the runs for ADS produced design variable and objective function results that were not near the exact solution. Only the third run with initial design variables (0.05, -1.05) did well, as its location is nearest to the exact solution. At the lower and upper limits of the search domain, DOT performed fairly well and better than GA, but BBBC best approximated the exact solution to the Goldstein-Price function at the extremities of the search domain. When the initial starting design variables were set closer to the exact solution at (0.50, -0.50) and (0.05, -1.05), however, DOT performed best among all four of the optimizers. When comparing BBBC to GA, BBBC tended to give more accurate results in fewer iterations, the exception being the second run, in which the GA solution was better. Overall, using DOT, GA, or BBBC resulted in solutions that were on target with regard to the exact solution to the Goldstein-Price function.

F. Sample 6: The Three-Hump Camel Function

For the objective function in equation (11), find the global minimum.

$$f(x,y) = 2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2 \tag{11}$$

The generalized surface plot overview for the three-hump camel function is shown in figure 12(a), while figure 12(b) shows a zoomed-in detailed view of this function, revealing that the “flat valley” in figure 12(a) is not what it appears to be.

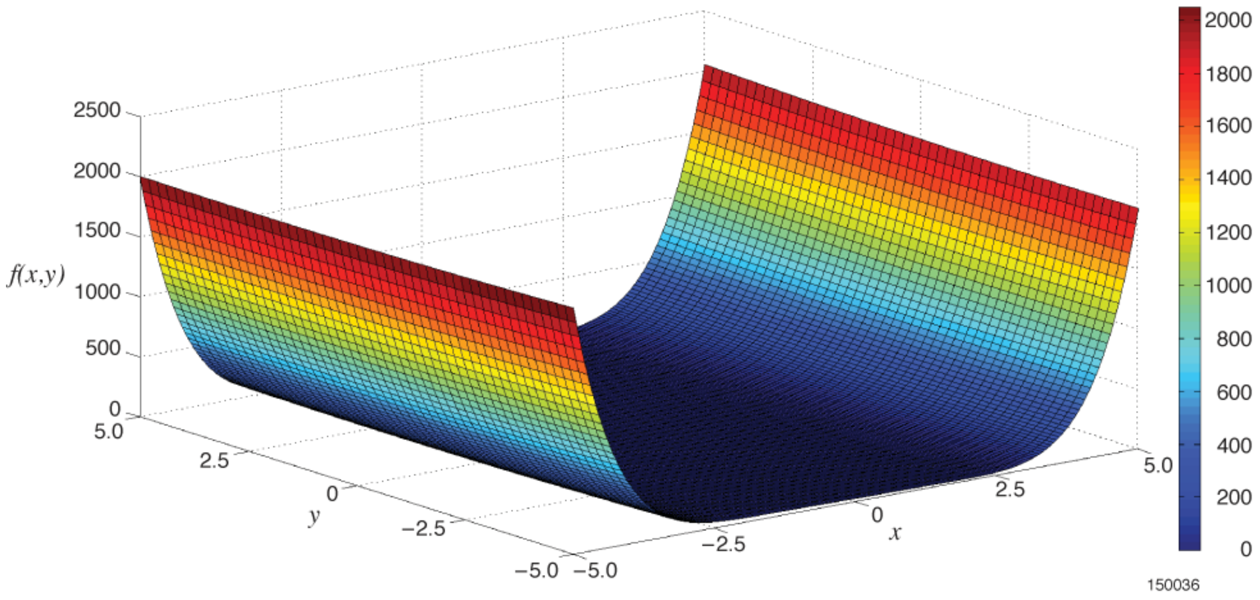


Figure 12(a). Surface plot for the three-hump camel function.

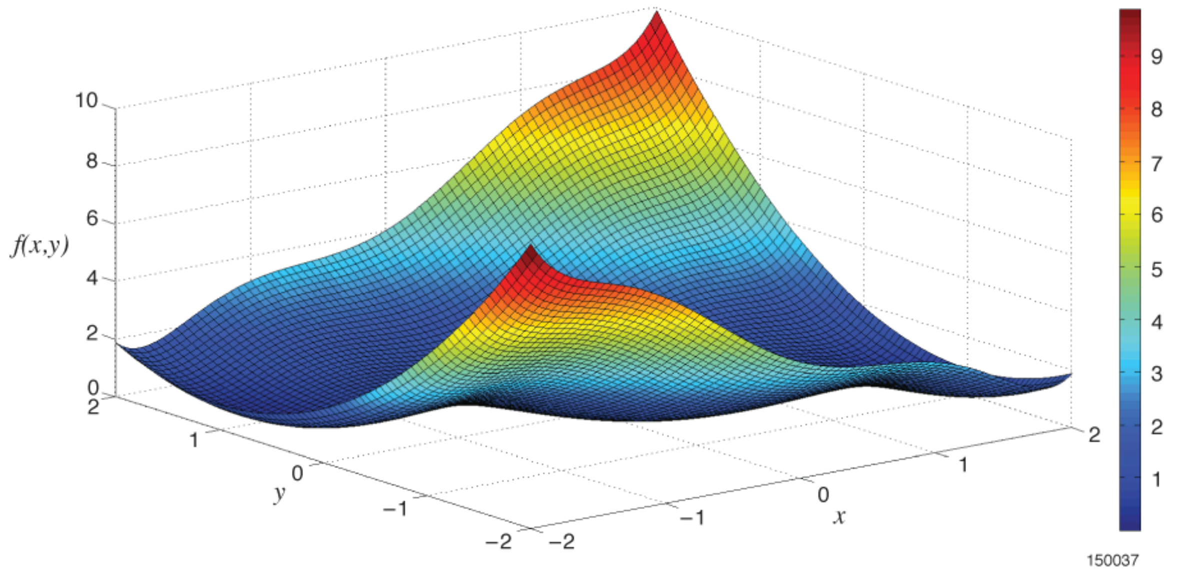


Figure 12(b). Surface plot for the three-hump camel function.

Figure 12. Surface plots for the three-hump camel function.

Lower- and upper-boundary search limits for the two design variables were between -5.0 and +5.0. The results of the optimization runs for the three-hump camel function are shown in table 6. The global minimum solution of the three-hump camel function is such that *f(0, 0) = 0.0*. It should be noted for this example that the BBBC input cards had β set to 0.07, with γ set to 1.0.

Table 6. Results for the three-hump camel function.

Minimum	Initial *x*	Initial *y*	Optimum *x*	Optimum *y*	Objective function *f(x,y)*	Number of function calls
Exact Global	-	-	0.0	0.0	0.0	-
ADS	-5.0	-5.0	1.93982	-5.00000	16.83932	46
ADS	-2.0	-2.0	1.74523	-0.86054	0.29882	76
ADS	2.0	2.0	-1.75223	0.87840	0.29877	69
ADS	5.0	5.0	-1.94982	5.00000	16.83636	47
DOT	-5.0	-5.0	0.00357	-0.00748	5.47E-05	34
DOT	-2.0	-2.0	1.10E-05	-2.47E-05	5.82E-10	36
DOT	2.0	2.0	3.04E-06	-8.70E-06	6.77E-11	37
DOT	5.0	5.0	-0.00375	0.00782	5.99E-05	34
GA	-5.0	-5.0	0.01168	-0.00789	2.43E-04	4,000
GA	-2.0	-2.0	0.00659	-0.00999	1.21E-04	4,000
GA	2.0	2.0	0.00196	-0.00629	3.49E-05	3,800
GA	5.0	5.0	0.01168	-0.00789	2.43E-04	4,000
BBBC	-5.0	-5.0	0.00359	-0.00355	2.56E-05	3,600
BBBC	-2.0	-2.0	-5.84E-04	0.00234	4.80E-06	2,800
BBBC	2.0	2.0	-5.03E-04	0.00242	5.14E-06	2,800
BBBC	5.0	5.0	0.00149	0.00321	1.96E-05	2,200

Among the four optimizers, BBBC best approximated the exact solution when the initial *x* and *y* design variables were set at the lower and upper search limits of -5.0 and 5.0, respectively, while DOT performed best for the middle two runs at initial *x* and *y* design variables of -2.0 and 2.0 due to the proximity to the exact solution. In addition, GA was able to approximate the exact solution at all ranges of the design variable search domain, although with slightly less accuracy and in more iterations than BBBC. The ADS, however, performed worst among the four optimizers. For all four sample runs in the search domain, ADS was unable to obtain a close approximation of not only the optimum *x* and *y* design variable values, but also the target objective function value, as shown in table 6. For the three-hump camel function, with the exception of ADS, using any one of the three remainder algorithms would yield an idea of what is the correct global minimum solution.

Conclusion

The object-oriented optimization (O^3) tool has been modified and demonstrated in this report. The O^3 tool now includes the gradient-based Automated Design Synthesis (ADS) optimizer as well as the Big-Bang-Big-Crunch (BBBC) global optimizer. The feasibilities of the O^3 tool leveraging with other executable codes have been demonstrated by using simple mathematical equations that readers can easily understand. Six different optimization functions have been presented, each with unique surface plot features, and testing of the O^3 tool capabilities has been performed in order to locate the exact global maximum or minimum. From the results tables for each optimization problem, it can be inferred that the gradient-based optimizers, ADS and Design Optimization Tools, are much faster in their calculations due to the lower number of function calls, but gradient optimizers tend to converge on local minima or maxima, as opposed to converging on the desired global maxima or minima. Global optimizers such as GA and BBBC, on the other hand, work very well for locating the exact global maximum or minimum for each of the sample problems. Despite requiring more function calls or iterations than the gradient-based optimizers, GA and BBBC were able to produce objective function results close to the exact global value, even at the extreme ends of the inputted search domain. The results in this report show the flexibility of the O^3 tool for optimization problems ranging from mathematical functions to aerospace applications.

Appendix

The object-oriented optimization (O^3) tool input data cards are presented and explained in this appendix.

A.1 DESVAR

DESVAR: Defines a design variable for design optimization.

Format:

1	2	3	4	5	6	7	8
DESVAR	ID	IOPT	XSTART	XL	XU	TABLE	FFFFF

Example:

1	2	3	4	5	6	7	8
DESVAR	2	0	3.5+3	1.0-5	1.0+4		1.0

1	2	3	4	5	6	7	8
DESVAR	101	1	2.0	0.0	5.0	ddv-01.dat	1.0

Field:

DESVAR (A10)
ID (I5) Unique design variable identification number. (Integer>0).

IOPT (I5) = 0: Continuous design variable
= 1: Discrete design variable

XSTART (F20.5) Initial starting value. (Real, XL$\leq$ XSTART$\leq$ XU)

XL (F10.5) Lower bound of design variable. (Real, default = 1.e+20)

XU (F10.5) Upper bound of design variable. (Real, default = 1.e+20)

TABLE (A20) Name of table for a discrete design variable (remark 1)

FFFFF (F10.5) Scaling factor

Remark:

1. The following data should be prepared for each discrete design variable table:

cdv(L), cdv(U), fix (prepare one line for each domain; 3 free format)
cdv(L): lower bound of continuous value
cdv(U): upper bound of continuous value
fix: fixed value within this domain

ex 1) if $2.5 \leq x < 3.5$: $x = 3$ and $3.5 \leq x < 4.5$: $x = 4$ then

cdv(L)=2.5	cdv(U)=3.5	fix=3.0
cdv(L)=3.5	cdv(U)=4.5	fix=4.0

ex 2) if $2.0 \le x < 3.0$: $x = 2$ and $3.0 \le x < 4.0$: $x = 3$ then

cdv(L)=2.0	cdv(U)=3.0	fix=2.0
cdv(L)=3.0	cdv(U)=4.0	fix=3.0

A.2 DOPTPRM

DOPTPRM: Override default values of parameters used in design.

Format:

1	2	3	4	5	6	7
DOPTPRM	PAR1	VAL1	PAR2	VAL2	PAR3	VAL3
+	PAR4	VAL4	-etc.-			

Example:

1	2	3	4	5	6	7
DOPTPRM	IOPT2	2	ICAS	0	MAXDOT	1
+	NRWK	1000	NRIWK	500	NGMAX	2
+	IGRAD	0				

Field:

DOPTPRM	(A10)	
PARi	(A10)	Name of the design optimization parameter. Allowable names are given in Tables A.1, A.2, A.3 and A.4. (character).
VALi	(I10 or F10.5)	Value of the parameter. (real or integer, see Tables A.1, A.2, A.3, A.4, A.8, A.9, A.10, and A.11).

Remark:

Only one DOPTPRM entry is allowed in the Bulk Data Section.

Table A.1. PARi names and descriptions for general input.

Name	Description, Type, and Default Value
ICAS	Flag for minimization or maximization (default = 0) = 0: minimization = 1: maximization
IOPT2	Optimization methodology in O^3 tool (default = 0)

	Continuous Design Variables (CDV) Discrete Design Variables (DDV) = 0: Sensitivity analysis for ranking design variables = 1: DOT (CDV) = 2: GA (CDV or DDV) = 3: ADS (CDV) = 6: BBBC (CDV or DDV)
RESTART	Flag for restarting optimization (default = 0) = 0: don’t use restart option = 1: restart optimization

Table A.2. PARi names and descriptions for sensitivity analysis.

Name	Description, Type, and Default Value
INPICK	Sorting of design variables will be based on this performance index number.
DELTA	Design variable changes for sensitivity analysis

Table A.3. PARi names and descriptions for automated design synthesis (integers) (ref. 13).

Name	Description, Type, and Default Value
ICNDIR	Restart parameter for conjugate direction and variable metric methods. Unconstrained minimization is restarted with a steepest descent direction every ICNDIR iterations. (default = NDV+1)
IGRAD	Gradient calculation control. Specifies whether the gradients are calculated by ADS or the user. (default = 0) = 0: by ADS = 1: by user as indicated by the value of INFO
INFO	Information parameter. On first call to ADS, INFO = 0 or -2. = 0: user does not wish to override internal parameters = -2: user wishes to override and change internal parameters When control returns ADS input to the calling program, INFO = 0, 1, or 2. = 0: optimization is complete = 1: user must evaluate objective and constraint functions again, and call ADS again. = 2: user must evaluate the gradient of the objective and the NGT constraints identified by the vector IC, and call ADS again. Note that all gradient information is calculated by finite difference within ADS.
IONED	One-dimensional search algorithm to be used. = 1: Find the minimum of an unconstrained function using the Golden Section method.

	= 2: Find the minimum of an unconstrained function using the Golden Section method followed by polynomial interpolation.
	= 3: Find the minimum of an unconstrained function by first finding bounds and then using polynomial interpolation.
	= 4: Find the minimum of an unconstrained function by polynomial interpolation/extrapolation without first having to find bounds on the solution.
	= 5: Find the minimum of a constrained function using the Golden Section method.
	= 6: Find the minimum of a constrained function using the Golden Section method followed by polynomial interpolation.
	= 7: Find the minimum of a constrained function by first finding bounds and then using polynomial interpolation.
	= 8: Find the minimum of a constrained function by polynomial interpolation/extrapolation without first having to find bounds on the solution.
IOPT	Optimizer to be used. (default = 0) = 0: None. Go directly to one-dimensional search. (This option should only be used for program development.) = 1: Fletcher-Reeves algorithm for unconstrained minimization = 2: Davidon-Fletcher-Powell (DFP) variable metric method for unconstrained minimization = 3: Broydon-Fletcher-Goldfarb-Shanno (BFGS) variable metric method for unconstrained minimization = 4: Method of Feasible Directions (MFD) for constrained minimization = 5: Modified Method of Feasible Directions for constrained minimization
KPRINT	A four-digit print control. KPRINT = IJKL, where I, J, K, and L have the following definitions: I – ADS system print control. 0: No print. 1: Print initial and final information. 2: Same as 1 plus parameter values and storage needs. 3: Same as 2 plus scaling information calculated by ADS. J – Strategy print control. 0: No print. 1: Print initial and final optimization information. 2: Same as 1 plus OBJ and X at each iteration. 3: Same as 2 plus G at each iteration. 4: Same as 3 plus intermediate information. 5: Same as 4 plus gradients of constraints. K – Optimizer print control. 0: No print. 1: Print initial and final optimization information. 2: Same as 1 plus OBJ and X at each iteration. 3: Same as 2 plus constraints at each iteration. 4: Same as 3 plus intermediate optimization and one-dimensional search information. 5: Same as 4 plus gradients of constraints.

L – One-Dimensional search print control, (debug only).
0: No print.
1: One-dimensional search debug information.
2: More of the same.

Example: KPRINT = 3120 represents that I = 3, J = 1, K = 2, and L = 0.

ISCAL	Scaling parameter. If ISCAL = 0, no scaling is done. If ISCAL = 1, the design variables, objective and constraints are scaled automatically. (default = 1)
ISTRAT	Optimization strategy to be used. (default = 0) = 0: None. Go directly to optimizer. = 1: Sequential unconstrained minimization using the exterior penalty function method = 2: Sequential unconstrained minimization using the linear extended interior penalty function method = 3: Sequential unconstrained minimization using the quadratic extended interior penalty function method = 4: Sequential unconstrained minimization using the cubic extended interior penalty function method = 5: Augmented Lagrange Multiplier Method = 6: Sequential Linear Programming = 7: Method of Centers (Method of Inscribed Hyperspheres) = 8: Sequential Quadratic Programming
ITMAX	Maximum number of iterations allowed at the optimizer level. (default = 40)
ITRMOP	The number of consecutive iterations for which the absolute or relative convergence criteria must be met to indicate convergence at the optimizer level. (default = 3)
ITRMST	The number of consecutive iterations for which the absolute or relative convergence criteria must be met to indicate convergence at the strategy level. (default = 2)
JONED	The one-dimensional search parameter (IONED) to be used in the Sequential Quadratic Programming method at the strategy level. (default = IONED)
JSMAX	Maximum number of iterations allowed at the strategy level. (default = 20)
NRIWK	Dimensioned size of work array IWK. A good estimate is 300 for a small problem. Increase the size of NRIWK as the problem grows larger. If NRIWK is too small, an error message will be printed and the optimization will be terminated. (default = 300)
NRWK	Dimensioned size of work array WK. NRWK should be set quite large, starting at about 1000 for a small problem. If NRWK has been given too small a value, an error message will be printed and the optimization will be terminated. (default = 1000)

Table A.4. PARi names and descriptions for automated design synthesis (real numbers) (ref. 13).

Name	Description, Type, and Default Value
ALAMDZ	Initial estimate of the Lagrange multipliers in the Augmented Lagrange Multiplier Method. (default = 0)
BETAMC	Additional steepest descent fraction in the method of centers. After moving to the center of the hypersphere, a steepest descent move is made equal to BETAMC times the radius of the hypersphere. (default = 0)
CT	Constraint tolerance in the Method of Feasible Directions or the Modified Method of Feasible Directions. A constraint is active if its numerical value is more positive than CT. (default = -0.03)
CTL	Same as CT, but for linear constraints. (default = -0.005)
CTLMIN	Same as CTMIN, but for linear constraints. (default = 0.001)
CTMIN	Minimum constraint tolerance for nonlinear constraints. If a constraint is more positive than CTMIN, it is considered to be violated. (default = 0.01)
DABALP	Absolute convergence criteria for the one-dimensional search when using the Golden Section method. (default = 0.0001)
DABOBJ	Maximum absolute change in the objective between two consecutive iterations to indicate convergence in optimization. (default = ABS(F0)/10,000, where F0 is the objective function value for the initial design.)
DABOBM	Absolute convergence criterion for the optimization sub-problem when using sequential minimization techniques. (default = ABS(F0)/1,000)
DABSTR	Same as DABOBJ, but used at the strategy level. (default = ABS(F0)/10,000)
DELALP	Relative convergence criteria for the one-dimensional search when using the Golden Section method. (default = 0.005)
DELOBJ	Maximum relative change in the objective between two consecutive iterations to indicate convergence in optimization. (default = 0.001)
DELOBM	Relative convergence criterion for the optimization sub-problem when using sequential minimization techniques. (default = 0.01)
DELSTR	Same as DELOBJ, but used at the strategy level. (default = 0.001)
DLOBJ1	Relative change in the objective function attempted on the first optimization iteration. Used to estimate initial move in the one-dimensional search. Updated as the optimization progresses. (default = 0.1)
DLOBJ2	Absolute change in the objective function attempted on the first optimization iteration. Used to estimate the initial move in the one-dimensional search. Updated as the optimization progresses. (default = 1000)

DX1	Maximum relative change in a design variable attempted on the first optimization iteration. Used to estimate the initial move in the one-dimensional search. Updated as the optimization progresses. (default = 0.01)
DX2	Maximum absolute change in a design variable attempted on the first optimization iteration. Used to estimate the initial move in the one-dimensional search. Updated as the optimization progresses. (default = 0.2)
EPSPEN	Initial transition point for extended penalty function methods. Updated as the optimization progresses. (default = -0.05)
EXTRAP	Maximum multiplier on the one-dimensional search parameter, ALPHA in the one-dimensional search using polynomial interpolation/extrapolation. (default = 5.0)
FDCH	Relative finite difference step when calculating gradients. (default = 0.01)
FDCHM	Minimum absolute value of the finite difference step when calculating gradients. This prevents too small a step when X(I) is near zero. (default = 0.001)
GMULTZ	Initial penalty parameter in Sequential Quadratic programming. (default = 10)
PSAIZ	Move fraction to avoid constraint violations in Sequential Quadratic Programming. (default = 0.95)
RMULT	Penalty function multiplier for the exterior penalty function method. Must be greater than 1.0. (default = 5)
RMVLMZ	Initial relative move limit. Used to set the move limits in sequential linear programming, method of inscribed hyperspheres and sequential quadratic programming as a fraction of the value of X(I), I = 1, NDV. (default = 0.2)
RP	Initial penalty parameter for the exterior penalty function method or the Augmented Lagrange Multiplier method. (default = 10)
RPMAX	Maximum value of RP for the exterior penalty function method or the Augmented Lagrange Multiplier method. (default = 1.0E+10)
RPMULT	Multiplier on RP for consecutive iterations. (default = 5.0)
RPPMIN	Minimum value of RPPRIM to indicate convergence. (default = 1.0E-10)
RPPRIM	Initial penalty parameter for extended interior penalty function methods. (default = 100)
SCFO	The user-supplied value of the scale factor for the objective function if the default or calculated value is to be overridden. (default = 1.0)
SCLMIN	Minimum numerical value of any scale factor allowed. (default = 0.001)

STOL	Tolerance on the components of the calculated search direction to indicate that the Kuhn-Tucker conditions are satisfied. (default = 0.001)
THETAZ	Nominal value of the push-off factor in the Method of Feasible Directions. (default = 0.1)
XMULT	Multiplier on the move parameter, ALPHA, in the one-dimensional search to find bounds on the solution. (default = 2.618034)
ZRO	Numerical estimate of zero on the computer. Usually the default value is adequate. If a computer with a short word length is used, ZRO = 1.0E-04 may be preferred. (default = 0.00001)

Table A.5. Automated design synthesis program selection options (ref. 13).

STRATEGY	OPTIMIZER (IOPT)				
(ISTRAT)	1	2	3	4	5
0	✓	✓	✓	✓	✓
1	✓	✓	✓	✗	✗
2	✓	✓	✓	✗	✗
3	✓	✓	✓	✗	✗
4	✓	✓	✓	✗	✗
5	✓	✓	✓	✗	✗
6	✗	✗	✗	✓	✓
7	✗	✗	✗	✓	✓
8	✗	✗	✗	✓	✓
ONE-D SEARCH (IONED)					
1	✓	✓	✓	✗	✗
2	✓	✓	✓	✗	✗
3	✓	✓	✓	✗	✗
4	✓	✓	✓	✗	✗
5	✗	✗	✗	✓	✓
6	✗	✗	✗	✓	✓
7	✗	✗	✗	✓	✓
8	✗	✗	✗	✓	✓

Table A.6. Automated design synthesis real parameters stored in array WK.

Parameter	Location	Default	Modules Where Used		
			ISTRAT	IOPT	IONED
ALAMDZ	1	0.0	5	-	-
BETAMC	2	0.0	7	-	-
CT[1]	3	-0.03	-	4,5	-
CTL	4	-0.005	-	4,5	-
CTLMIN	5	0.001	-	4,5	-
CTMIN	6	0.01	-	4,5	-
DABALP[2]	7	0.0001	-	ALL	-
DABOBJ	8	ABS(F0)/10000	ALL	-	-

DABOBM	9	ABS(F0)/1000	ALL	-	-
DABSTR	10	ABS(F0)/10000	ALL	-	-
DELALP[3]	11	0.005	-	-	1, 2, 5, 6
DELOBJ	12	0.001	-	ALL	-
DELOBM	13	0.01	ALL	-	-
DELSTR	14	0.001	ALL	-	-
DLOBJ1	15	0.1	-	ALL	-
DLOBJ2	16	1000.0	-	ALL	-
DX1	17	0.01	-	ALL	-
DX2	18	0.2	-	ALL	-
EPSPEN	19	-0.05	2, 3, 4	-	-
EXTRAP	20	5.0	-	-	ALL
FDCH	21	0.01	-	ALL	-
FDCHM	22	0.001	-	ALL	-
GMULTZ	23	10.0	8	-	-
PSAIZ	24	0.95	8	-	-
RMULT	25	5.0	1, 5	-	-
RMVLMZ	26	0.2	6, 7, 8	-	-
RP	27	10.0	1, 5	-	-
RPMAX	28	1.0E+10	1, 5	-	-
RPMULT	29	0.2	1, 5	-	-
RPPMIN	30	1.0E-10	2, 3, 4	-	-
RPPRIM	31	100.0	2, 3, 4	-	-
SCFO	32	1.0	ALL	ALL	ALL
SCLMIN	33	0.001	ALL	ALL	ALL
STOL	34	0.001	-	4, 5	-
THETAZ	35	0.1	-	4, 5	-
XMULT	36	2.618034	-	-	1, 2, 3, 5, 6, 7
ZRO	37	0.00001	ALL	ALL	ALL

[1]If IOPT = 4, CT = -0.1
[2]If IONED = 3 or 8, DABALP = 0.001
[3]If IONED = 3 or 8, DELALP = 0.05

Table A.7. Automated design synthesis integer parameters stored in array IWK.

Parameter	Location	Default	Modules Where Used		
			ISTRAT	IOPT	IONED
ICNDIR	1	NDV+1	-	ALL	-
ISCAL	2	1	ALL	ALL	ALL
ITMAX	3	40	-	ALL	-
ITRMOP	4	3	-	1, 2, 3	-
ITRMST	5	2	ALL	-	-
JONED	6	IONED	8	-	-
JSMAX	7	20	ALL	-	-

Table A.8. PARi names and descriptions for design optimization tools (integers).

Name	Description, Type, and Default Value
IGMAX	=0: only gradients of active and violated constraints are calculated. >0: up to NGMAX gradients are calculated, including active, violated, and near active constraints. (default = 0)
IGRAD	Similar to the definition of IGRAD for ADS = -1 or 0: by DOT = 1: by user
IPRINT	Control parameter for printing. (default = 3) = 0 no output = 1 internal parameters, initial information and results. = 2 same plus objective function and X-vector at each iteration = 3 same plus G-vector and critical constraint numbers. = 4 same plus gradients. = 5 same plus search direction. = 6 same plus set IPRNT1 = 1 and IPRNT2 = 1 = 7 same except set IPRNT2 = 2
IPRNT1	= 1: print scaling factors for the X vector. (default = 0)
IPRNT2	= 1: print miscellaneous search information. = 2: turn on print during one-dimensional search process. This is for debugging only. (default = 0)
ISCAL	Similar to the definition of ISCAL for ADS Design variables are rescaled every ISCAL iteration. Set ISCAL = -1 to turn off scaling. (default = number of design variable)
ITMAX	Similar to the definition of ITMAX for ADS. (default = 100).
ITRMOP	Similar to the definition of ITRMOP for ADS. (Integer; default = 2).
ITRMST	Similar to the definition of ITRMST for ADS. (Integer > 0; default = 2).
JTMAX	Maximum number of iterations allowed for the Sequential Linear Programming Method. This is the number of linearized sub-problems solved. (Integer $\geq$ 0; default = 50).
JPRINT	Ref. 14
JWRITE	Ref. 14
MAXDOT	Ref. 14
MAXINT	Ref. 14

METHOD	Ref. 14
NGMAX	Ref. 14
NRIWK	Similar to the definition of NRIWK for ADS.
NRWK	Similar to the definition of NRWK for ADS.

Table A.9. PARi names and descriptions for design optimization tool (real numbers).

Name	Description, Type, and Default Value
CT	Similar to the definition of CT for ADS.
CTMIN	Similar to the definition of CTMIN for ADS. (default = 0.003)
DABOBJ	Similar to the definition of DABOBJ for ADS. (default = MAX[0.0001*ABS(F0),1.e-20])
DABSTR	Similar to the definition of DABSTR for ADS. (default = 0.003)
DELOBJ	Similar to the definition of DELOBJ for ADS. (default = 0.001)
DELSTR	Ref. 14 (default = 0.001)
DOBJ1	Similar to the definition of DLOBJ1for ADS. (default = 0.1)
DOBJ2	Similar to the definition of DLOBJ2for ADS. (default = 0.2*ABS(F0))
DX1	Similar to the definition of DX1 for ADS. (default = 0.01)
DX2	Similar to the definition of DX2 for ADS. (default = 0.2*ABS[x(l)])
FDCH	Similar to the definition of FDCH for ADS. (default = 0.001)
FDCHM	Similar to the definition of FDCHM for ADS (default = 0.0001)
RMVLMZ	Ref. 14. (default = 0.4)

Table A.10. PARi names and descriptions for the genetic algorithm.

Name	Description, Type, and Default Value
EPSOBJ	Epsilon value for convergence criteria (default = 0.0001)
FDIF	Relative fitness differential; range from 0 (none) to 1 (maximum). (default = 1)
ICON	Print optimum results flag; 0 or 1 (default = 1) 0 = off 1 = on

IDIG	Number of significant digits (i.e., number of genes) retained in chromosomal encoding (default = 6) (Note: This number is limited by the machine floating point precision. Most 32-bit floating point representations have only 6 full digits of precision. To achieve greater precision, this routine could be converted to double precision, but note that this would also require a double precision random number generator, which likely would not have more than 9 digits of precision if it used 4-byte integers internally.)
IELITE	Elitism flag; 0 or 1 (default = 0) 0 = off 1 = on (Applies only to reproduction plans 1 and 2; see IREP for more info.)
IGEN	Number of generations over which solution is to evolve. (default = 500)
IMUT	Mutation mode; 1/2/3/4/5/6 (default = 2) 1 = one-point mutation, fixed rate 2 = one-point, adjustable rate based on fitness 3 = one-point, adjustable rate based on distance 4 = one-point+creep, fixed rate 5 = one-point+creep, adjustable rate based on fitness 6 = one-point+creep, adjustable rate based on distance
IPOP	Number of individuals in a population (default = 100)
IREP	Reproduction plan; 1/2/3 (default = 1) 1 = Full generational replacement 2 = Steady-state-replace-random 3 = Steady-state-replace-worst
IVRB	Printed output; 0/1/2 (default = 0) 0 = None 1 = Minimal 2 = Verbose
NCONV	Convergence criteria; number of the same global optimum (default = 5)
PCROSS	Crossover probability; must be ≤ 1.0 (default = 0.85) If crossover takes place, either one or two splicing points are used, with equal probabilities
PMUT	Initial mutation rate; should be small (default = 0.005) (Note: The mutation rate is the probability that any one gene locus will mutate in any one generation.)
PMUTMN	Minimum mutation rate; must be ≥ 0.0 (default = 0.0005)
PMUTMX	Maximum mutation rate; must be ≤ 1.0 (default = 0.25)

R	Positive multiplier for inequality constraints. (default = 10)

Table A.11. PARi names and descriptions for the Big-Bang-Big-Crunch algorithm.

Name	Description, Type, and Default Value
ALPHA	Parameter limiting the size of the search domain. (default = 1; range: 0 - 1)
BETA	Parameter controlling the influence of the best optimum on the location of new candidate optimum. (default=0.7; range: 0 - 1)
EPSOBJ	Epsilon value for convergence criteria. (default = 0.0001)
GAMMA	Parameter controlling the influence of the global best optimum and local best optimum based on the best optimum. (default=0.7; range: 0 - 1) Xnew(ides, ipop) = beta*xcg(ides) + (1 – beta)*(gamma*xglb(ides) + (1 – gamma)*xold(ides, ifit(1))) + random*alpha*(xdesu(ides) – xdesl(ides))/(ibang + 1)
NBANG	Number of Big-Bangs-Big-Crunches. (default = 20)
NCONV	Convergence criteria number of the same global optimum. (default = 5)
NPOP	Number of individuals in a population (default = 50)

A.3 INDEX

INDEX: Prepare INDEX cards for each performance index. Object and constraint functions will be defined from performance indices.

Format:

1	2	3	4	5	6
INDEX	ID	INTOBJ	INTCON	FACOBJ	INTGRA
+	TASK				
+	SCRIPT				
+	OUTPUT				
+	SCRIPT_GRAD (needed when INTGRA=1)				
+	OUTPUT_GRAD (needed when INTGRA=1)				

Example:

INDEX	1	1	0	1.0	0
+	Total Weight: based on analytical equation				
+	f				
+	f.dat				
INDEX	2	0	1	0.0	0
+	First inequality constraint: based on analytical equation				
+	g1				

+	g1.dat				
INDEX	3	0	1	0.0	0
+	Second inequality constraint: based on analytical equation				
+	g2				
+	g2.dat				

INDEX	1	1	0	1.0	1
+	Total Weight: based on analytical equation				
+	f				
+	f.dat				
+	fdot				
+	fdot.dat				
INDEX	2	0	1	0.0	1
+	First inequality constraint: based on analytical equation				
+	g1				
+	g1.dat				
+	g1dot				
+	g1dot.dat				
INDEX	3	0	1	0.0	1
+	Second inequality constraint: based on analytical equation				
+	g2				
+	g2.dat				
+	g2dot				
+	g2dot.dat				

Field:

ID	(I10)	Unique integer variable identification number. (integer>0)
INTOBJ	(I10)	Part of objective function ? Yes then 1, 2, or 3; No then 0 1: linear obj(i) 2: quadratic obj(i)**2 3: absolute |obj(i)| ex) obj= a1*obj(1) + a2*obj(2)**2 + a3*|obj(3)| + ...
INTCON	(I10)	Part of constraints? Yes then 1 or 2; No then 0 1: inequality constraint 2: equality constraint
FACOBJ	(F10.5)	Scaling factor for objective function (real, default=1.0) a1, a2, ... (scaling factors) ex) obj= a1*obj(1) + a2*obj(2) + ... or epsilon for constraints g(i) <= facobj(i) for inequality constraints Lagrange multiplier for equality constraints
INTGRA	(I10)	User supplied gradients ? Yes then 1 ; No then 0

TASK	(A70)	Task description
SCRIPT	(A70)	Name of script file for this performance index
OUTPUT	(A70)	Name for output file where the performance indices are saved. write(unit,*) performance index format(real; double precision; free format)
SCRIPT_GRAD	(A70)	Name of script file for analytical gradient computations
OUTPUT_GRAD		(A70) Name for output file where gradient of performance index with respect to design variables are saved. write(unit,*) ndv format(integer; double precision; free format) write(unit,*) (dx(i),i=1,ndv) format(real; double precision; free format) where, ndv=number of design variable dx(i)=gradients

References

1. Pak, Chan-gi, *Preliminary Development of an Object-Oriented Optimization Tool*, NASA/TM-2011-216419, 2011.

2. Pak, Chan-gi, and Wesley Li, "Multidisciplinary Design, Analysis, and Optimization Tool Development Using a Genetic Algorithm," ICAS 2008-9.5.1, September 2008.

3. Spivey, Natalie D., Claudia Y. Herrera, Roger Truax, Chan-gi Pak, and Donald Freund, "Quiet Spike^{TM} Build-up Ground Vibration Testing Approach," AIAA-2007-1775, April 2007.

4. Herrera, Claudia Y., and Chan-gi Pak, "Build-up Approach to Updating the Mock Quiet Spike^{TM} Beam Model," AIAA-2007-1776, April 2007.

5. Lung, Shun-fat, and Chan-gi Pak, *Structural Model Tuning Capability in an Object-Oriented Multidisciplinary Design, Analysis, and Optimization Tool*, NASA/TM-2008-214640, 2008.

6. Pak, Chan-gi, "Finite Element Model Tuning Using Measured Mass Properties and Ground Vibration Test Data," *J. Vib. Acoust.*, 131(1), doi:10.1115/1.2981092, January 2009.

7. Lung, Shun-fat, and Chan-gi Pak, *Updating the Finite Element Model of the Aerostructures Test Wing with Ground Vibration Test Data*, NASA/TM-2009-214646, 2009.

8. Pak, Chan-gi, and Shun-fat Lung, "Reduced Uncertainties in the Flutter Analysis of the Aerostructures Test Wing," ICAS 2010-9.6.1, September 2010.

9. Pak, Chan-gi, and Shun-fat Lung, "Flutter Analysis of Aerostructures Test Wing with Test Validated Structural Dynamic Model," *Journal of Aircraft*, Vol. 48, No. 4, 2011, pp. 1263-1272.

10. Pak, Chan-gi, and Samson Truong, "Creating a Test-Validated Structural Dynamic Finite-Element Model of the X-56A Aircraft Structure," *Journal of Aircraft (AIAA Early Edition)*, doi:10.2514/1.C033043, 2014.

11. Pak, Chan-gi, "Unsteady Aerodynamic Model Tuning for Precise Flutter Prediction," *Journal of Aircraft*, Vol. 48, No. 6, 2011, pp. 2178-2184.

12. Li, Wesley, and Chan-gi Pak, "Aeroelastic Optimization Study Based on X-56A Model," (slide presentation), presented at the AIAA Atmospheric Flight Mechanics Conference, June 2014.

13. Vanderplaats, Garret N., "ADS - A FORTRAN Program for Automated Design Synthesis," NASA Contractor Report 177985, 1985.

14. *DOT Design Optimization Tools User's Manual Version 5.0*, Vanderplaats Research & Development, Inc., Colorado Springs, Colorado, 1999.

15. Charbonneau, Paul, and Barry Knapp, *A User's Guide to PIKAIA 1.0.*, NCAR/TN-418+IA, 1995.

16. Erol, Osman K., and Ibrahim Eksin, "A New Optimization Method: Big Bang-Big Crunch," *Advances in Engineering Software*, Vol. 37, No. 2, 2006, pp. 106–111, doi:10.1016/j.advengsoft.2005.04.005.

17. Camp, C., “Design of Space Trusses Using Big Bang-Big Crunch Optimization,” J. Struct. Eng., 133(7), 2007, pp. 999-1008, doi:10.1061/(ASCE)0733-9445(2007)133:7(999).

18. Kaveh, A., and S. Talatahari, “Size Optimization of Space Trusses Using Big Bang-Big Crunch Algorithm,” *Computers and Structures*, Vol. 87, Nos. 17-18, 2009, pp. 1129–1140, doi:10.1016/j.compstruc.2009.04.011.