NASA/TM—2011–216419

Preliminary Development of an Object-Oriented Optimization Tool

Chan-gi Pak

Dryden Flight Research Center, Edwards, California

January 2011

NASA/TM—2011–216419

Preliminary Development of an Object-Oriented Optimization Tool

Chan-gi Pak

Dryden Flight Research Center, Edwards, California

National Aeronautics and
Space Administration

Dryden Flight Research Center
Edwards, CA 93523-0273

January 2011

Available from:

NASA Center for AeroSpace Information
7115 Standard Drive
Hanover, MD 21076-1320
443-757-5802

Abstract

The National Aeronautics and Space Administration Dryden Flight Research Center has developed a FORTRAN-based 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. The 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 central executive module and the discipline modules, or both. Six sample optimization problems are presented. The first four sample problems are based on simple mathematical equations; the fifth and sixth problems consider a three-bar truss, which is a classical example in structural synthesis. Instructions for preparing input data for the O^3 tool are presented.

Nomenclature

AIC	aerodynamic influence coefficient
BFGS	Broyden–Fletcher–Goldfarb–Shanno
CDV	continuous design variables
CEM	central executive module
CG	center of gravity
DC	gradient-based algorithm with continuous design variable
DDV	discrete design variables
DFRC	Dryden Flight Research Center
DOT	design optimization tool
G	constraint functions
GA	genetic algorithm
GC	genetic algorithm with continuous design variables
GD	genetic algorithm with discrete design variables
i	#i: name of discipline module i
j	#j: name of discipline module j
J	performance index
k	#k: name of discipline module k
NASA	National Aeronautics and Space Administration
O^3	object-oriented optimization tool
P_1	applied external load
P_2	applied external load
x	design variable
X_1	design variable #1
X_2	design variable #2
X_3	design variable #3
$\{X\}$	vector of design variables

Introduction

New methodologies, technologies, and design concepts facilitate the design of advanced aircraft with improved performance as well as reduced operating costs and weight. The aerospace industry has historically focused on developing aircraft that have multifunctional mission capabilities and expanded flight envelopes, while at the same time attempting to reduce manufacturing costs as well as the weight of the airframe. More recently, environmental concerns impose further design constraints: aircraft designers should now design for reductions in cabin and engine exhaust noise, sonic booms, and NOx emissions, as well as improved fuel efficiency. These complicated requirements and constraints demand multidisciplinary consideration for successful advanced aircraft design.

Supporting the Aeronautics Research Mission Directorate (ARMD) guidelines, the National Aeronautics and Space Administration (NASA) Dryden Flight Research Center (DFRC) has developed an object-oriented optimization (O^3) tool. The tool leverages existing tools and practices and allows easy integration and adoption of new state-of-the-art software. A computer code for finite element (FE) model tuning (refs. 1, 2) has been developed using the O^3 tool together with MSC/NASTRAN (MSC.Software Corporation, Santa Ana, California), a computer software program. The primary objective of this model tuning code is to obtain a ground-vibration-test-validated structural dynamics FE model that can provide a reliable flutter analysis to define the flutter placard speed to which an aircraft can be flown prior to flight flutter testing (ref. 3).

Optimization has made its way into many mainstream applications. For example, MSC/NASTRAN has developed solution sequence 200 for design optimization (ref. 4), and MATLAB (The MathWorks, Natick, Massachusetts) has developed an optimization toolbox (ref. 5). Other applications, such as ZAERO (ZONA Technology Inc., Scottsdale, Arizona) aeroelastic panel code (ref. 6) and CFL3D Navier-Stokes solver (ref. 7) do not include a built-in optimizer.

Most commercially available multidisciplinary design, analysis, and optimization (MDAO) tools have been developed to perform within limited disciplines with a single-fidelity modeling capability. These tools are typically developed as a single large software application that performs analysis for all disciplines but has little or no capability to integrate multi-fidelity and multidisciplinary components that have already been developed as stand-alone analysis codes. Although a multitude of tools have been developed and are well-adapted to interdisciplinary aircraft design and analysis, they have not been developed to work together.

The primary and long-term objective of the development of the O^3 tool is to generate a “central executive” capable of using disparate software packages in a cross-platform network environment so as to quickly perform optimization and design tasks in a cohesive and streamlined manner. 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 executive and the discipline modules, or both. Figure 1 shows a typical set of discipline modules and their relation to the central executive.

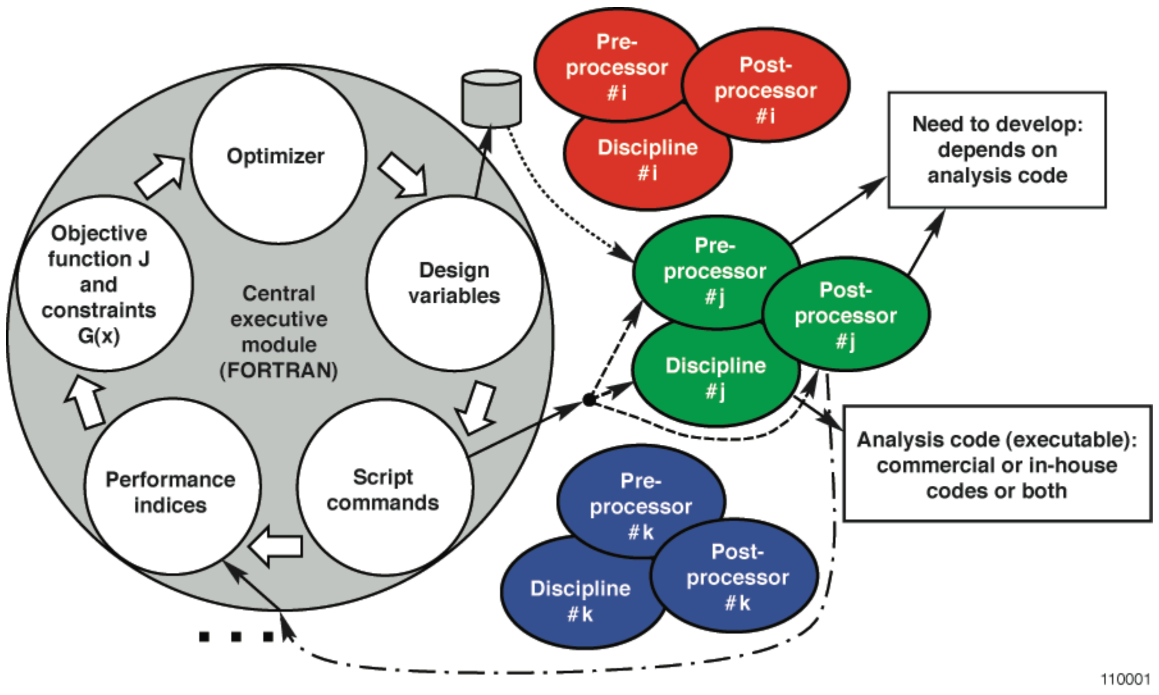

Figure 1. The object-oriented optimization tool.

Background

At the heart of the O^3 tool is the central executive module (CEM), shown in figure 1. The CEM was written in FORTRAN. The script commands for each performance index were submitted through the use of the FORTRAN "call system" command, as shown in appendix A. In this CEM, the user will choose an optimization methodology and define the objective and constraint functions from performance indices. The user will also provide starting and side constraints for continuous as well as discrete design variables and external file names for performance indices that communicate between the CEM and each analysis module. The performance indices can be total weight, safety factors, frequencies, lift, drag, noise level, flutter speed, gain and phase margin, et cetera.

Two optimizer software types are included in the O^3 tool: design optimization tools (DOTs) (ref. 8) based on a gradient-based algorithm; and the genetic algorithm (GA) (ref. 9). 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 user's program. The user's program calls the optimizer again to obtain the next design point; this process is repeated until the optimizer returns a parameter to indicate that the optimum objective function is reached.

The GA does not require gradient calculations and can be started with random seeds, eliminating some of the need for user input and allowing for solutions that may not be readily apparent even to experienced designers (ref. 10). In the case of multiple local minima problems, GAs are able to find the global optimum results, while gradient-based algorithms may converge to the local optimum value.

Different types of optimization methodology are available by using two different optimizer and continuous as well as discrete design variables. Optimizers include:

- GA with continuous design variables (GC) or discrete design variables (GD)
- a gradient-based algorithm, that is, DOT, with a continuous design variable (DC)
- GC before DOT to perform the global optimization with DOT (GC + DC)

- GD after DOT to convert continuous design variables to discrete design variables (GC + DC + GD and DC + GD).

Additional optimizer software can be added to this module in the future if needed.

Each discipline module consists of three sub-modules: the 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 CEM 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 within the current CEM environment. The script command will execute 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 automatically compute the performance indices.

Applications

Detailed instructions for preparing the O^3 data input cards DESVAR, DOPTPRM, and INDEX, are explained in appendix B. Free sequence of these input data cards is used in the O^3 tool. The information in the following lists will be provided by each input command.

DESVAR cards for each design variable:

- continuous versus discrete design variable
- starting value
- lower and upper limit of design variable
- name of table for a discrete design variable.

DOPTPRM card:

- optimization methodology
- control variables for optimizer routines GA and DOT.

INDEX cards for each performance index:

- objective function versus constraint function
- scaling factor in case of objective function
- small allowable value in case of equality as well as inequality constraints
- user-supplied gradient or not
- name of script file for the performance index (interface variable)
- name of output file where the performance index is saved
- name of script file for the gradient of the performance index (when user-supplied)
- name of output file where the gradient of the performance index is saved (when user-supplied).

Six sample problems are now presented to demonstrate the code. The first four sample problems are based on simple mathematical equations and show the basic concept used in the current O^3 tool. The fifth and sixth problems involve a three-bar truss, which is a classical example in structural synthesis.

Sample Problem 1: Mathematical Equation Without User-Supplied Gradients

Consider the following optimization problem statements, equations (1) and (2):

$$\text{Minimize: } f(x) = x^2 - 2x + 3 \tag{1}$$

$$\text{Subject to: } g(x) = -x^2 + 3x - 1 \le 0 \tag{2}$$

From the inequality constraint given in equation (2), the feasible domain for the design variable x will be $x \leq \frac{3-\sqrt{5}}{2}$ or $x \geq \frac{3+\sqrt{5}}{2}$. Therefore, the global minimum of the objective function f is at $x = \frac{3-\sqrt{5}}{2}$ as shown in figure 2, and the corresponding f value is

$$f\left(\frac{3-\sqrt{5}}{2}\right) = \left(\frac{3-\sqrt{5}}{2}\right)^2 - 2\left(\frac{3-\sqrt{5}}{2}\right) + 3 = \frac{9+5-6\sqrt{5}}{4} - 3 + \sqrt{5} + 3 = \frac{14}{4} - \left(\frac{3}{2} - 1\right)\sqrt{5}$$
$$= \frac{7}{2} - \frac{\sqrt{5}}{2} = \frac{7-\sqrt{5}}{2} = 2.3820$$

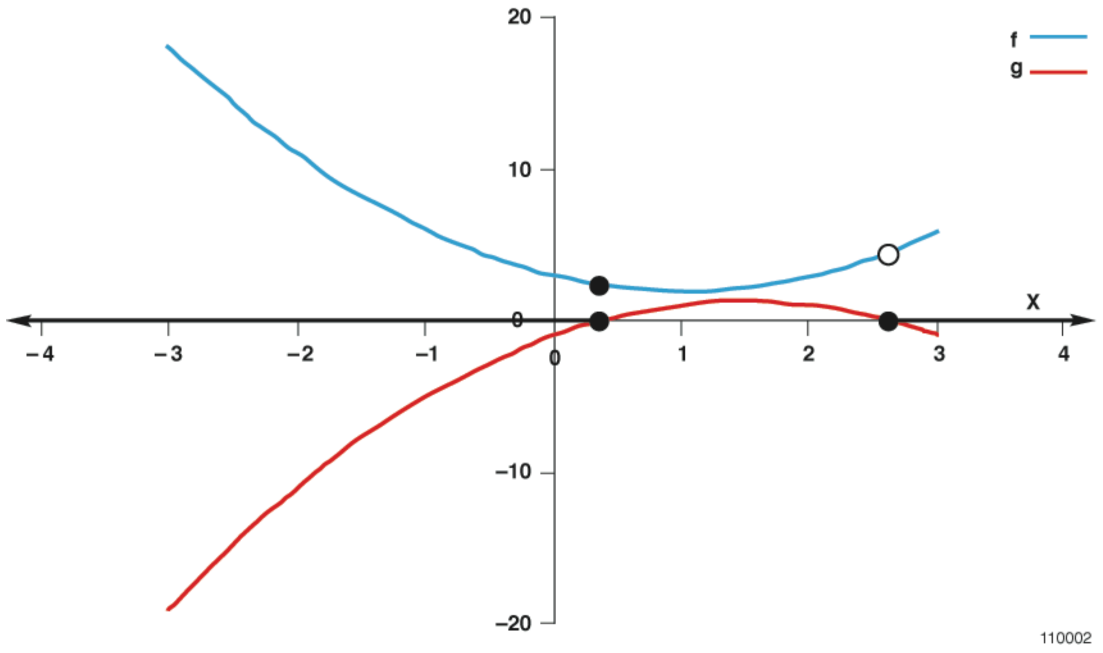

Figure 2. The f and g curves for sample problem 1.

In this problem, the local minimum of the objective function f is at $x = \frac{3+\sqrt{5}}{2}$, the open circle in figure 2. The corresponding f value is

$$f\left(\frac{3+\sqrt{5}}{2}\right) = \left(\frac{3+\sqrt{5}}{2}\right)^2 - 2\left(\frac{3+\sqrt{5}}{2}\right) + 3 = \frac{9+5+6\sqrt{5}}{4} - 3 + \sqrt{5} + 3 = \frac{14}{4} + \left(\frac{3}{2} - 1\right)\sqrt{5}$$
$$= \frac{7}{2} + \frac{\sqrt{5}}{2} = \frac{7+\sqrt{5}}{2} = 4.6180$$

These analytical results are summarized in table 1.

Table 1. Results from sample problem 1.

Minimum	Initial x	Optimum x	Objective function f	Constraint function g	Number of function calls
Exact global	N/A	$\frac{3-\sqrt{5}}{2}=0.38197$	$\frac{7-\sqrt{5}}{2}=2.3820$	0	N/A
Exact local	N/A	$\frac{3+\sqrt{5}}{2}=2.6180$	$\frac{7+\sqrt{5}}{2}=4.6180$	0	N/A
DOT	0.1	0.38197	2.3820	6.2173E-15	8
DOT	4.0	2.6180	4.6180	-1.15463E-14	8
GA	0.1	0.38192	2.3820		1000
GA	4.0	0.38192	2.3820		1000

For the computer simulation with the O^3 tool, the gradient-based search, DOT, with a starting value of 0.1, is selected; these results are also given in table 1. Input data cards for this simulation are as follows:

DOPTPRM	IOPT2	2	ICAS	0	MAXDOT	1
+	NRWK	1000	NRIWK	500	NGMAX	2
+	IGRAD	0	CT	-0.00001	CTMIN	0.00001
DESVAR	1	0	0.1	0.0	2.0	
INDEX	1	1	0	1.0	0	
+	Sample problem 1: Object function					
+	f					
+	f.dat					
INDEX	2	0	1	0.0	0	
+	Sample problem 1: Constraint function					
+	g					
+	g.dat					

The script files f.bat and g.bat, FORTRAN source codes for executable files obj.exe and const.exe, and external files for performance indices f.dat and g.dat are given in appendix C, section C.1. An external file, design_variables, represented in figure 3, for the O^3 tool cannot be shared with the other executable codes obj.exe and const.exe, therefore, a copy of the external file, design_var, is created in the script file f.bat given in appendix C, section C.1.1. In the first script file, f.bat, the obj command in the second line will execute computations of the function f found in equation (1). The corresponding FORTRAN program is provided in appendix C, section C.1.3. The performance index f is saved in the external file f.dat, as shown in this FORTRAN program. In the second script file, g.bat, the const command will execute computations of the function g found in equation (2). The FORTRAN source code is given in appendix C, section C.1.4. In this case, the performance index g will be saved in the external file g.dat. Based on these two performance indices, f and g, the objective function and the constraint function will be computed as shown in appendix A.

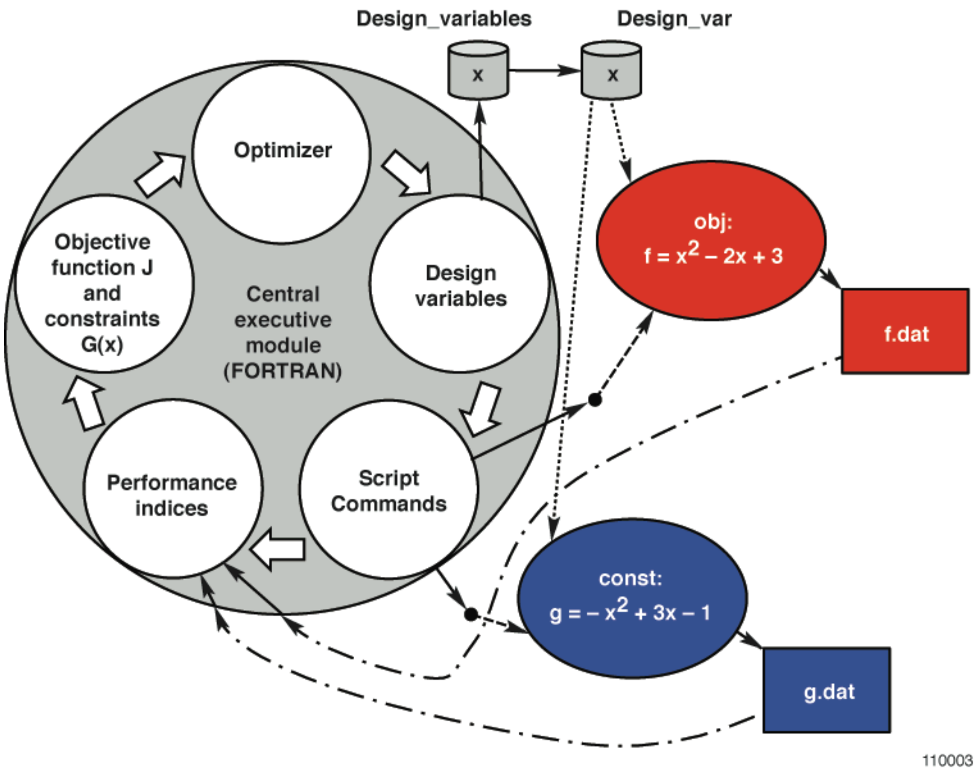

Figure 3. The problem structure of sample problem 1.

The results from this simulation are shown in table 1. Note from this table that DOT optimizer converges to the exact global minimum value. The starting x value of 0.1 converges to the optimum value of 0.38197 within eight optimization iterations.

Another DOT simulation with a starting x value of 4.0 is also performed; the corresponding DESVAR card for this simulation is as follows:

DESVAR	1	0	4.0	-1.0	4.0	

All of the other input data cards, DOPTPRM, and INDEX, are the same as those used above. Input data for the second simulation are as follows:

DOPTPRM	IOPT2	2	ICAS	0	MAXDOT	1
+	NRWK	1000	NRIWK	500	NGMAX	2
+	IGRAD	0	CT	-0.00001	CTMIN	0.00001
DESVAR	1	0	4.0	-1.0	4.0	
INDEX	1	1	0	1.0	0	
+	Sample problem 1: Object function					
+	f					
+	f.dat					
INDEX	2	0	1	0.0	0	
+	Sample problem 1: Constraint function					
+	g					
+	g.dat					

In this second DOT simulation, the local minimum value of 2.618 is obtained as shown in table 1; this is the limitation of the gradient-based optimizer, which cannot overcome the difficulties with discontinuous design space.

The GA optimizer is selected for the third numerical simulation; the input data cards, with starting x value of 0.1, are given as:

DOPTPRM	IOPT2	1	IPOP	100	IGEN	10
DESVAR	1	0	0.1	0.0	2.0	
INDEX	1	1	0	1.0	0	
+	Sample problem 1: Object function					
+	f					
+	f.dat					
INDEX	2	0	1	0.0	0	
+	Sample problem 1: Constraint function					
+	g					
+	g.dat					

Two different starting values of x, 0.1 and 4.0, are used in this simulation, and these results are also given in table 1. In the case of the GA optimizer, it always converges to the global minimum value; this is the major benefit of using the global optimizer, such as the GA optimizer. The optimization iteration of 1000 (=IPOP x IGEN) is, however, somewhat large compared to the gradient-based optimizers.

Sample Problem 2: Mathematical Equation With User-Supplied Gradients

Consider the same optimization problem statements given in sample problem 1, that is, equations (1) and (2):

$$\text{Minimize: } f(x) = x^2 - 2x + 3 \tag{1}$$

$$\text{Subject to: } g(x) = -x^2 + 3x - 1 \le 0 \tag{2}$$

with the following user-supplied “analytical” gradients:

$$\frac{df(x)}{dx} = 2x - 2 \tag{3}$$

$$\frac{dg(x)}{dx} = -2x + 3 \tag{4}$$

Input data cards for the DOT simulation are as follows:

DOPTPRM	IOPT2	2	ICAS	0	MAXDOT	1
+	NRWK	1000	NRIWK	500	NGMAX	2
+	IGRAD	1	CT	-0.00001	CTMIN	0.00001
DESVAR	1	0	0.1	0.0	2.0	

INDEX	1	1	0	1.0	1	
+	Sample problem 2: Object function					
+	f					
+	f.dat					
+	fdot					
+	fdot.dat					
INDEX	2	0	1	0.0	1	
+	Sample problem 2: Constraint function					
+	g					
+	g.dat					
+	gdot					
+	gdot.dat					

The basic structure of this optimization problem is shown in figure 4. The script files f.bat, fdot.bat, g.bat, and gdot.bat; and the corresponding FORTRAN programs obj.f, obj_grad.f, const.f, and const_grad.f are given in appendix C. The results from this simulation are summarized in table 2.

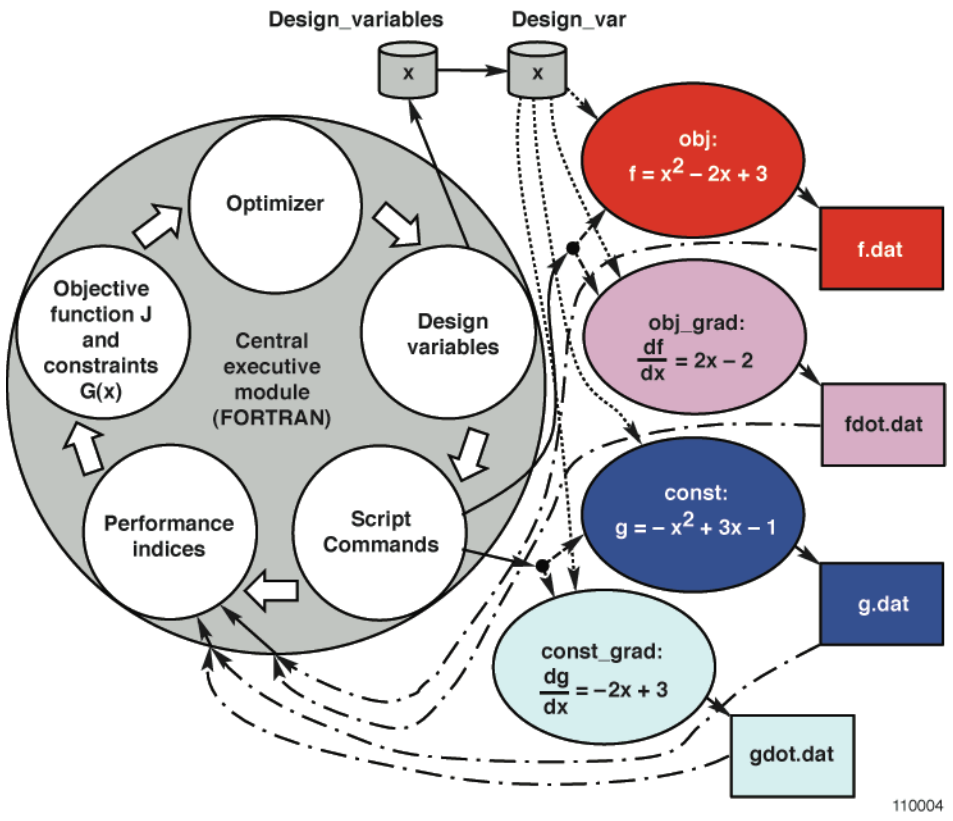
Design_variables
Design_var
x
x
Optimizer
Objective function J and constraints G(x)
Central executive module (FORTRAN)
Design variables
Performance indices
Script Commands
obj: $f = x^2 - 2x + 3$
f.dat
obj_grad: $\frac{df}{dx} = 2x - 2$
fdot.dat
const: $g = -x^2 + 3x - 1$
g.dat
const_grad: $\frac{dg}{dx} = -2x + 3$
gdot.dat
110004

Figure 4. The problem structure of sample problem 2.

Table 2. Results from sample problem 2.

	Initial x	Optimum x	Objective function f	Constraint function g	Number of function calls	Number of gradient calls
Exact global	N/A	$\frac{3-\sqrt{5}}{2}=0.38197$	$\frac{7-\sqrt{5}}{2}=2.3820$	0	N/A	N/A
DOT	0.1	0.38197	2.3820	6.2173E-15	6	2

The number of function calls are decreased from 8 to 6 with the user-supplied gradients approach as shown in tables 1 and 2.

Sample Problem 3: Equality Constraint Without User-Supplied Gradients

Consider the following optimization problem statements with an equality constraint. Three design variables are used in this sample problem.

$$\text{Minimize:} \quad f(X_1, X_2, X_3) = (X_1 + X_2)^2 + (X_2 + X_3)^2 \tag{5}$$

$$\text{Subject to:} \quad h(X_1, X_2, X_3) = X_1 + 2X_2 + 3X_3 - 1 = 0 \tag{6}$$

The problem structure of this sample problem is given in figure 5. In this problem, performance indices are f and h as shown in figure 5.

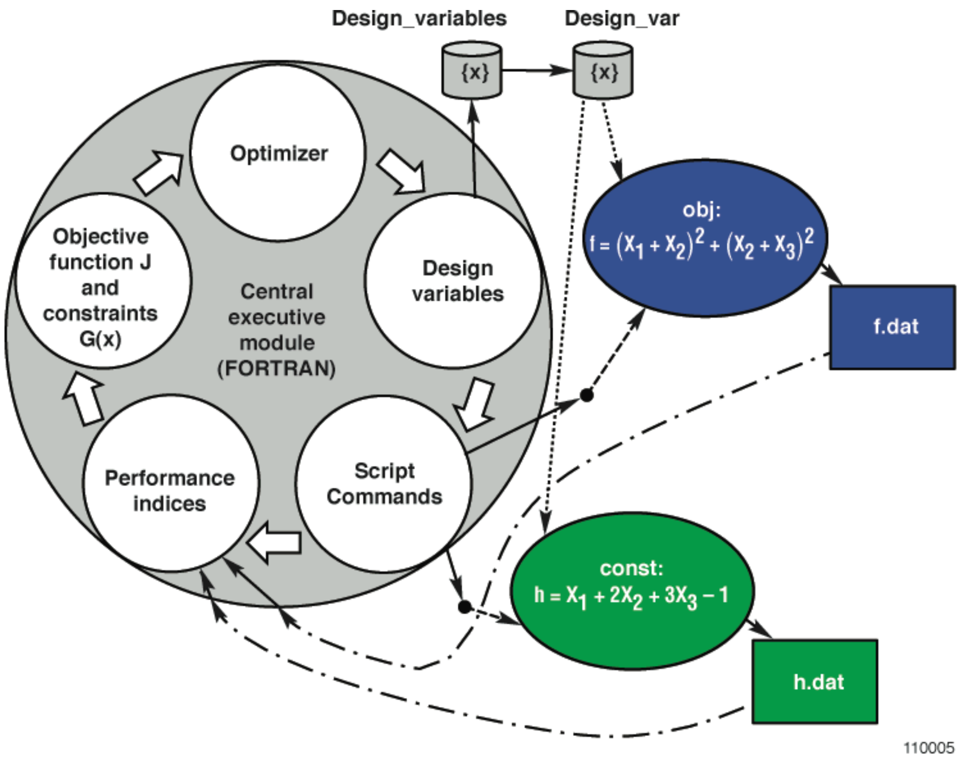
Design_variables
Design_var
{x}
{x}
Optimizer
Objective function J and constraints G(x)
Central executive module (FORTRAN)
Design variables
Performance indices
Script Commands
obj:
$f = (X_1 + X_2)^2 + (X_2 + X_3)^2$
f.dat
const:
$h = X_1 + 2X_2 + 3X_3 - 1$
h.dat
110005

Figure 5. The problem structure of sample problem 3.

The exact solution can be obtained as follows: Rewrite the equality constraint, equation (6), as shown by equation (7):

$$X_1 = 1 - 2X_2 - 3X_3 \tag{7}$$

Substituting equation (7) into equation (5) gives equation (8):

$$f\left(X_2, X_3\right) = \left(1 - X_2 - 3X_3\right)^2 + \left(X_2 + X_3\right)^2 \tag{8}$$

From equation (8), the minimum f is at equations (9) and (10):

$$\frac{\partial f\left(X_2, X_3\right)}{\partial X_2} = -2\left(1 - X_2 - 3X_3\right) + 2\left(X_2 + X_3\right) = 4X_2 + 8X_3 - 2 = 0 \tag{9}$$

$$\frac{\partial f\left(X_2, X_3\right)}{\partial X_3} = -6\left(1 - X_2 - 3X_3\right) + 2\left(X_2 + X_3\right) = 8X_2 + 20X_3 - 6 = 0 \tag{10}$$

From equations (9) and (10), $X_2 = -0.5$ and $X_3 = 0.5$. Therefore, from equations (7) and (5), $X_1 = 0.5$ and $f = 0$.

The exact solution as well as the DOT simulation results in ref. 8 are summarized in table 3.

Table 3. Results from sample problem 3.

Variable	Exact	Results in reference 8 using two inequality constraints	DOT in this study using Lagrange multiplier
Design variable X_1	0.5	.475	.501
Design variable X_2	-0.5	-.469	-.500
Design variable X_3	0.5	.488	.500
Objective function f	0	3.8e-4	1.4e-06
Constraint function g_1	N/A	8.4e-5	N/A
Number of function calls	N/A	43	43

In the O^3 tool, an equality constraint h in equation (6) is added to the objective function using the Lagrange multiplier λ as shown in equation (11):

$$\bar{f}\left(X_1, X_2, X_3, \lambda\right) = f\left(X_1, X_2, X_3\right) + \lambda h\left(X_1, X_2, X_3\right) \tag{11}$$

Input data cards for sample problem 3 with an equality constraint are as follows:

DOPTPRM	IOPT2	2	ICAS	0	MAXDOT	1
+	NRWK	1000	NRIWK	500	NGMAX	2
+	IGRAD	0				
DESVAR	1	0	-4.0	-10.0	10.0	

DESVAR	2	0	1.0	-10.0	10.0	
DESVAR	3	0	2.0	-10.0	10.0	
INDEX	1	1	0	1.0	0	
+	Sample problem 3: Object function					
+	f					
+	f.dat					
INDEX	2	0	2	2.0	0	
+	Sample problem 3: Equality constraint					
+	h					
+	h.dat					

The Lagrange multiplier of 2.0 is used in this sample problem. The script files f.bat, h.bat, and other required programs, obj.f and const.f, are given in appendix C, section C.3. The results from this simulation are summarized in table 3. Note from this table that the DOT optimizer converges to the exact solution.

The major difference between the DOT simulation in this study and that in ref. 8 is the method used to handle the equality constraints. The Lagrange multiplier is used in this study, however, two inequality constraints,

$$g_1 = -\left(X_1 + 2X_2 + 3X_3 - 1\right) \le \varepsilon$$

$$g_2 = X_1 + 2X_2 + 3X_3 - 1 \le \varepsilon$$

where, ε = a small number, are used to solve the problem with the equality constraint in reference 8.

Instead of solving equation (6), the following equation

$$-\varepsilon \le X_1 + 2X_2 + 3X_3 - 1 \le \varepsilon$$

is used in reference 8. It can be concluded from the results presented in table 3 that the results obtained from using the Lagrange multiplier are much more accurate than those obtained by using two inequality constraints. Accuracy and convergence of the optimization using two inequality constraints are strong function of the small number ε. When ε is a large number, good convergence and bad accuracy can result. On the other hand, when a very small number is selected for the ε value, good accuracy and bad convergence can result, because the feasible domain for the design variables will be very narrow.

Sample Problem 4: Equality Constraint With User-Supplied Gradients

Recall the optimization problem statements in sample problem 3, that is, equations (5) and (6). The problem structure of this sample problem with user-supplied gradients is presented in figure 6.

$$\text{Minimize:} \quad f\left(X_1, X_2, X_3\right) = \left(X_1 + X_2\right)^2 + \left(X_2 + X_3\right)^2 \tag{5}$$

$$\text{Subject to:} \quad h\left(X_1, X_2, X_3\right) = X_1 + 2X_2 + 3X_3 - 1 = 0 \tag{6}$$

with the following user-supplied gradients:

$$\frac{df(X_1,X_2,X_3)}{dX_1}=2(X_1+X_2)$$

$$\frac{df(X_1,X_2,X_3)}{dX_2}=2(X_1+X_2)+2(X_2+X_3)$$

$$\frac{df(X_1,X_2,X_3)}{dX_3}=2(X_2+X_3)$$

$$\frac{dh(X_1,X_2,X_3)}{dX_1}=1$$

$$\frac{dh(X_1,X_2,X_3)}{dX_2}=2$$

$$\frac{dh(X_1,X_2,X_3)}{dX_3}=3$$

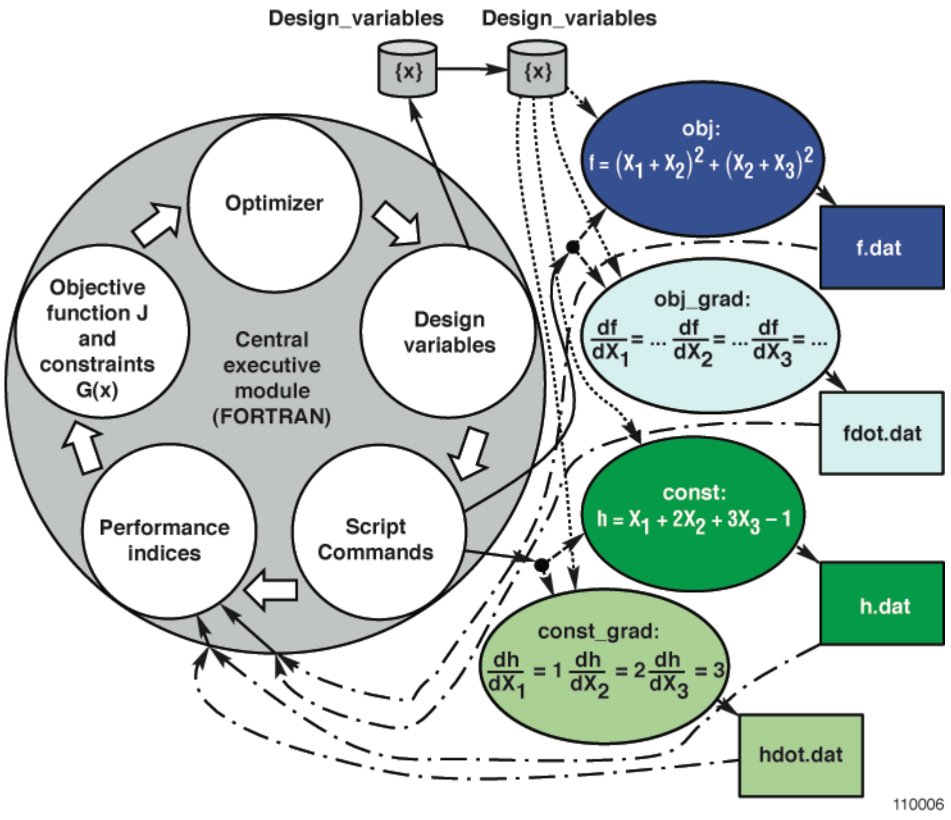

Figure 6. The problem structure of sample problem 4.

Input data cards for sample problem 4 are as follows:

DOPTPRM	IOPT2	2	ICAS	0	MAXDOT	1
+	NRWK	1000	NRIWK	500	NGMAX	2
+	IGRAD	1				
DESVAR	1	0	-4.0	-10.0	10.0	
DESVAR	2	0	1.0	-10.0	10.0	
DESVAR	3	0	2.0	-10.0	10.0	
INDEX	1	1	0	1.0	1	
+	Sample problem 4: Object function					
+	f					
+	f.dat					
+	fdot					
+	fdot.dat					
INDEX	2	0	2	2.0	1	
+	Sample problem 4: Equality constraint					
+	h					
+	h.dat					
+	hdot					
+	hdot.dat					

The script files f.bat, h.bat, fdot.bat, hdot.bat, and FORTRAN source codes, obj.f, obj_grad.f, const.f, and const_grad.f, are given in appendix C, section C.4. The results from this simulation are summarized in table 4.

Table 4. Results from sample problem 4.

Variable	Exact	DOT in this study
Design variable X_1	0.5	.500
Design variable X_2	-0.5	-.500
Design variable X_3	0.5	.500
Objective function f	0	1.6e-27
Number of function calls	N/A	16
Number of gradient calls	N/A	5

Note from this table that DOT optimizer converges to the exact solution. The total number of function calls, 43, in table 3, is reduced to 16, as shown in table 4, with improved accuracy. Better convergence and accuracy are the major advantages of using the user-supplied gradients.

Sample Problem 5: Three-Bar Truss Without User-Supplied Gradients

The optimization of a three-bar truss problem, as shown in figure 7, is now discussed. In this problem, the objective is to minimize the total volume of the structure.

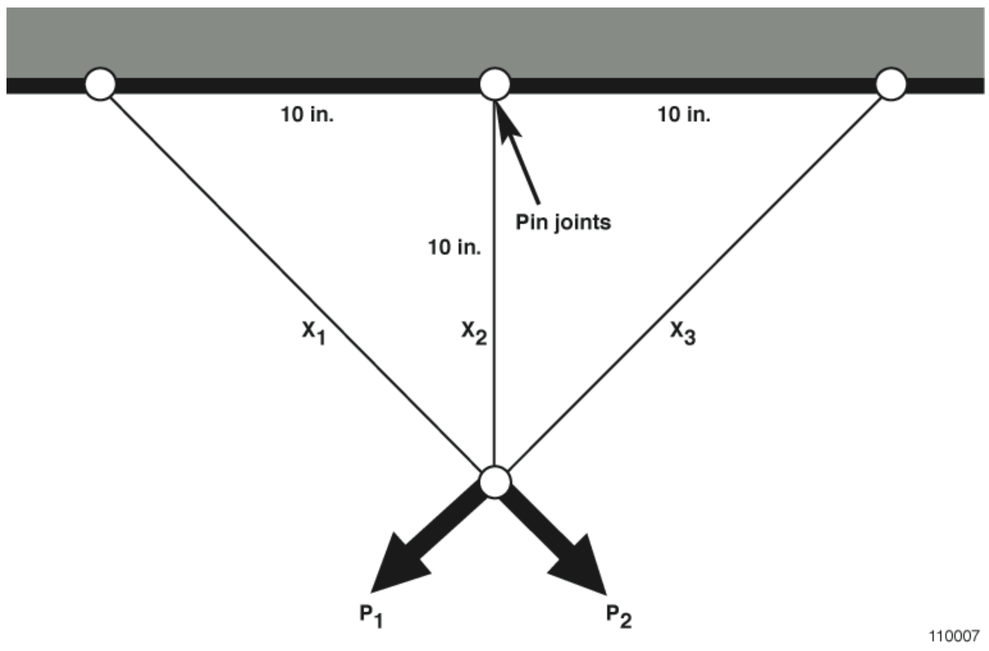

Figure 7. Three-bar truss load conditions.

In figure 7, the design variables X_1 and X_2 correspond to the cross-sectional areas of bar 1 and bar 2, respectively. The area of bar 3 is "linked" to be the same as bar 1 for symmetry. The constraints are tensile and compressive stress constraints in bar 1 and bar 2 under loading P_1=20000. The loadings P_1 and P_2 are applied separately. The optimization problem statement (ref. 8), in the standard form for optimization, is given as:

$$\text{Minimize: } f\left(X_1, X_2\right) = 2\sqrt{2}X_1 + X_2$$

$$\text{Subject to: } g_1\left(X_1, X_2\right) = \frac{2X_1 + \sqrt{2}X_2}{2X_1\left(X_1 + \sqrt{2}X_2\right)} - 1 < 0 \text{ and } g_2\left(X_1, X_2\right) = \frac{1}{X_1 + \sqrt{2}X_2} - 1 < 0 \text{ where}$$

$$0.01 \le X_i \le 2 \quad \mathrm{i} = 1, 2$$

The problem structure of this sample problem is presented in figure 8. Three performance indices are used. The first performance index, f, is for the objective function, and the second and third performance indices, g_1 and g_2, are for the inequality constraints.

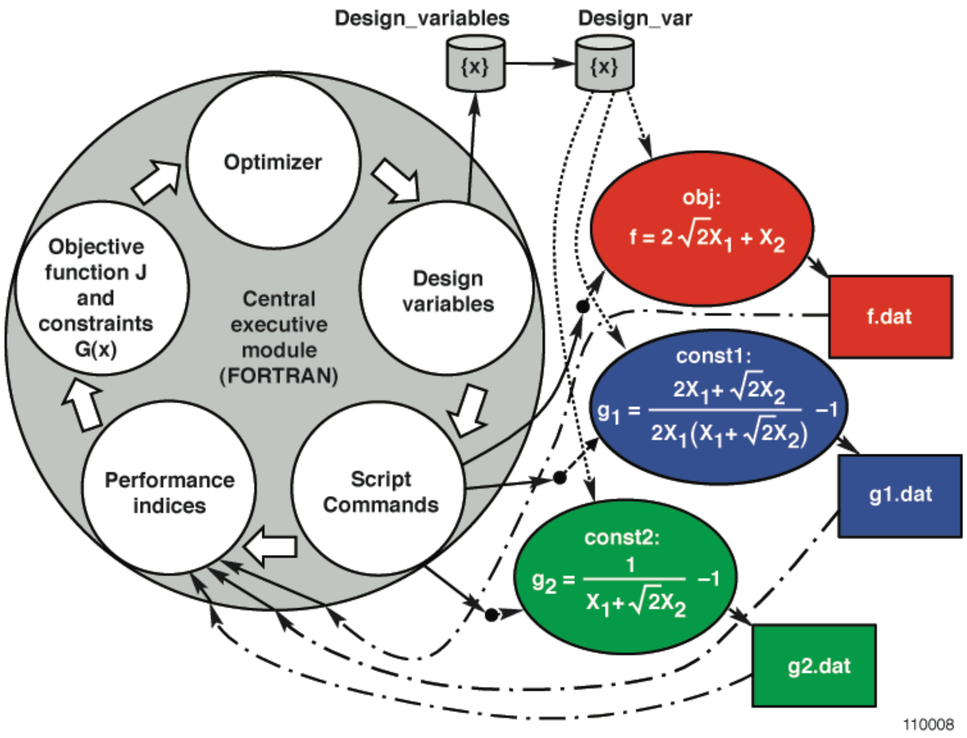
Design_variables
Design_var
{x}
{x}
Optimizer
Objective function J and constraints G(x)
Central executive module (FORTRAN)
Design variables
Performance indices
Script Commands
obj:
$f = 2\sqrt{2}X_1 + X_2$
f.dat
const1:
$g_1 = \frac{2X_1 + \sqrt{2}X_2}{2X_1(X_1 + \sqrt{2}X_2)} - 1$
g1.dat
const2:
$g_2 = \frac{1}{X_1 + \sqrt{2}X_2} - 1$
g2.dat
110008

Figure 8. The problem structure of sample problem 5.

The input data cards for the three-bar truss problem are as follows:

DOPTPRM	IOPT2	2	ICAS	0	MAXDOT	1
+	NRWK	1000	NRIWK	500	NGMAX	2
+	IGRAD	0				
DESVAR	1	0	1.0	0.01	2.0	
DESVAR	2	0	1.0	0.01	2.0	
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					

The script files f.bat, g1.bat, and g2.bat, and FORTRAN source codes for executable files obj.exe, con1.exe, and con2.exe, are given in appendix C, section C.5. The results from this computer simulation are provided in table 5.

Table 5. Results from sample problem 5.

Variable	DOT in this study	DOT in reference 8
Design variable X_1	.799	.799
Design variable X_2	.372	.372
Objective function f	2.633	2.633
Constraint function g_1	2.8e-03	2.8e-3
Constraint function g_2	-2.5e-01	-6.2e-1
Number of function calls	29	29

Note from this table that the design variables computed from DOT optimizer in this study converge to those given in reference 8. Also note that the constraint value g_2 at the optimum design is significantly different; this is mainly because the constraint function g_2 is quite "flat" near the optimum design.

Sample Problem 6: Three-Bar Truss With User-Supplied Gradients

Recall the optimization problem statements in sample problem 5,

$$\text{Minimize: } f\left(X_1, X_2\right) = 2\sqrt{2}X_1 + X_2$$

$$\text{Subject to: } g_1\left(X_1, X_2\right) = \frac{2X_1 + \sqrt{2X_2}}{2X_1\left(X_1 + \sqrt{2X_2}\right)} - 1 < 0 \text{ and } g_2\left(X_1, X_2\right) = \frac{1}{X_1 + \sqrt{2}X_2} - 1 < 0 \text{ where}$$

$$0.01 \le X_i \le 2 \quad \mathrm{i} = 1,2$$

with the following user-supplied gradients:

$$\frac{df\left(X_1, X_2\right)}{dX_1} = 2\sqrt{2}$$

$$\frac{df\left(X_1, X_2\right)}{dX_2} = 1$$

$$\frac{dg_1\left(X_1, X_2\right)}{dX_1} = \frac{2\left(2X_1^2 + 2\sqrt{2}X_1X_2\right) - \left(2X_1 + \sqrt{2}X_2\right)\left(4X_1 + 2\sqrt{2}X_2\right)}{\left(2X_1^2 + 2\sqrt{2}X_1X_2\right)^2} = \frac{-X_1^2 - \sqrt{2}X_1X_2 + X_2^2}{\left(X_1^2 + \sqrt{2}X_1X_2\right)^2}$$

$$\frac{dg_1\left(X_1, X_2\right)}{dX_2} = \frac{\sqrt{2}\left(2X_1^2 + 2\sqrt{2}X_1X_2\right) - \left(2X_1 + \sqrt{2}X_2\right)2\sqrt{2}X_1}{\left(2X_1^2 + 2\sqrt{2X_1X_2}\right)^2} = \frac{-1}{\sqrt{2}\left(X_1 + \sqrt{2}X_2\right)^2}$$

$$\frac{dg_2\left(X_1,X_2\right)}{dX_1}=\frac{-1}{\left(X_1+\sqrt{2}X_2\right)^2}$$

$$\frac{dg_2\left(X_1,X_2\right)}{dX_2}=\frac{-\sqrt{2}}{\left(X_1+\sqrt{2}X_2\right)^2}$$

The input data cards are as follows:

DOPTPRM	IOPT2	2	ICAS	0	MAXDOT	1
+	NRWK	1000	NRIWK	500	NGMAX	2
+	IGRAD	1				
DESVAR	1	0	1.0	0.1	2.0	
DESVAR	2	0	1.0	0.1	2.0	
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					

The script files f.bat, fdot.bat, g1.bat, g1dot.bat, g2.bat, g2dot.bat, and other required programs are given in appendix C, section C.6. The results from this simulation are provided in table 6. The total number of function calls, 29, in table 5, is reduced to 23, as shown in table 6.

Two more sample applications of the O^3 tool are shown in figures 9 and 10. Future work should focus on developing an unsteady aerodynamic model tuning tool and an object-oriented MDAO tool.

Table 6. Results from sample problem 6.

Variable	DOT in this study
Design variable X_1	.799
Design variable X_2	.371
Objective function f	2.633
Constraint function g_1	2.9e-03
Constraint function g_2	-2.5e-01
Number of function calls	23
Number of gradient calls	3

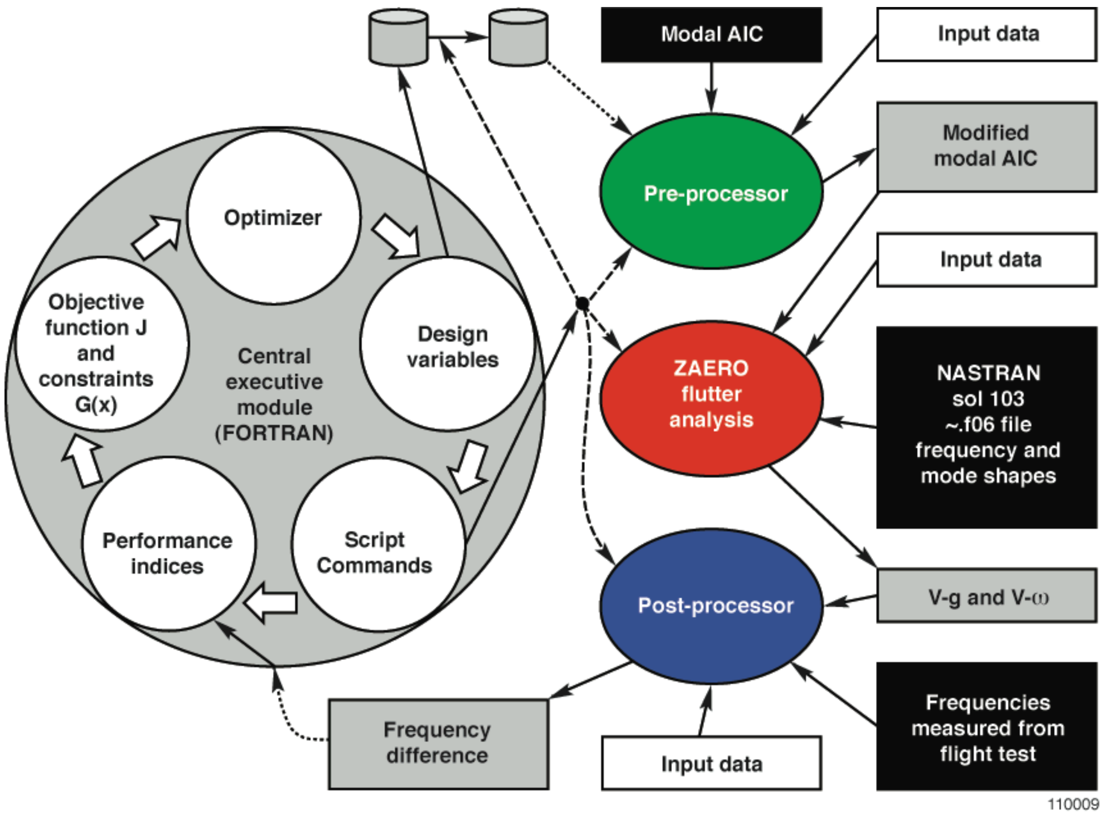

Figure 9. The problem structure of unsteady aerodynamic model tuning.

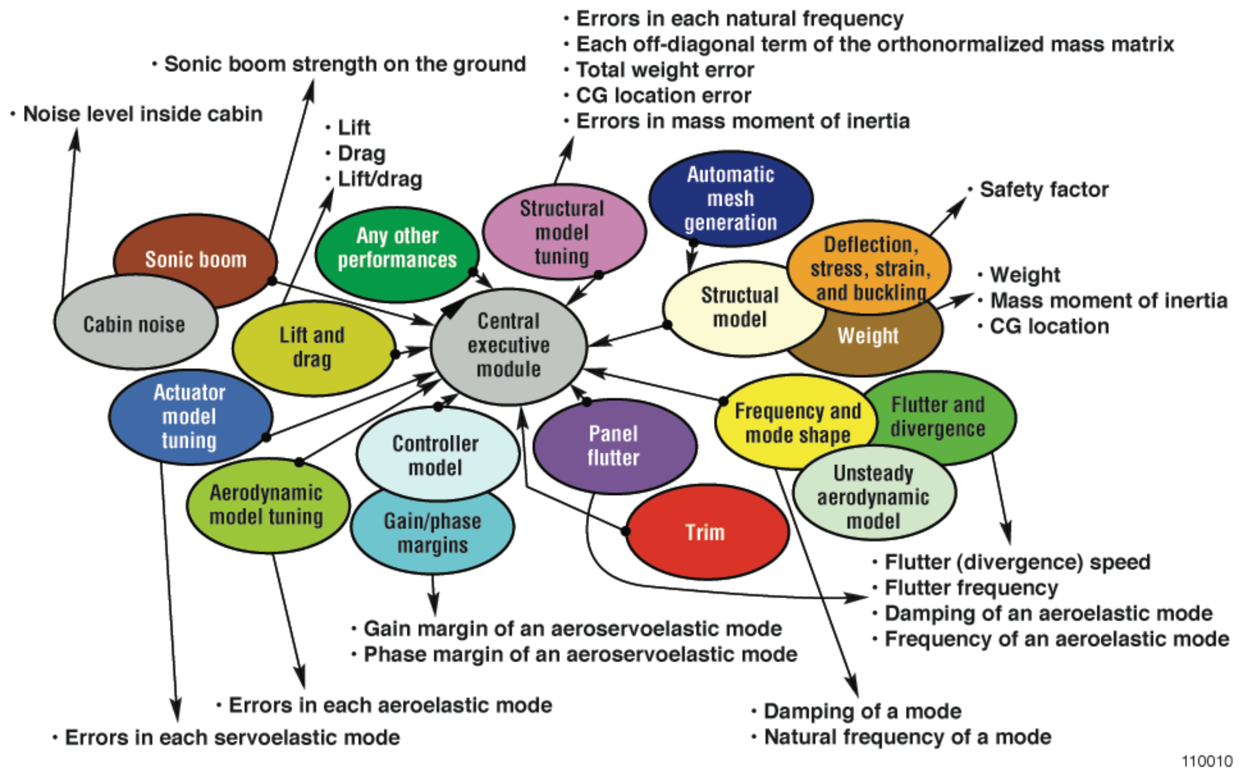
• Errors in each natural frequency
• Each off-diagonal term of the orthonormalized mass matrix
• Total weight error
• CG location error
• Errors in mass moment of inertia
• Sonic boom strength on the ground
• Noise level inside cabin
• Lift
• Drag
• Lift/drag
• Safety factor
Automatic mesh generation
Structural model tuning
Any other performances
Sonic boom
Deflection, stress, strain, and buckling
• Weight
• Mass moment of inertia
• CG location
Cabin noise
Lift and drag
Central executive module
Structual model
Weight
Actuator model tuning
Controller model
Panel flutter
Frequency and mode shape
Flutter and divergence
Unsteady aerodynamic model
Aerodynamic model tuning
Gain/phase margins
Trim
• Flutter (divergence) speed
• Flutter frequency
• Damping of an aeroelastic mode
• Frequency of an aeroelastic mode
• Gain margin of an aeroservoelastic mode
• Phase margin of an aeroservoelastic mode
• Errors in each aeroelastic mode
• Errors in each servoelastic mode
• Damping of a mode
• Natural frequency of a mode
110010

Figure 10. Sample performance indices for each discipline.

Conclusion

The NASA Dryden Flight Research Center FORTRAN-based object-oriented optimization (O^3) tool is developed and demonstrated. The feasibilities of O^3 tool leveraging with other executable codes are shown by way of simple mathematical equations that also enable understanding of the basic concept of the O^3 tool. The results demonstrate the flexibility of the O^3 tool for optimization problems and indicate the ease of implementation of the tool for engineering problems such as structural model tuning; unsteady aerodynamic model tuning; multidisciplinary design, analysis, and optimization; and other optimization problems using commercial codes, in-house executable codes, or both. Sample performance indices for the development of a multidisciplinary design, analysis, and optimization tool are presented.

Appendix A

This appendix presents instructions for preparing the input data for the object-oriented optimization (O^3) tool. Further discussion is given in the "Background" section in the body of the report.

```
subroutine objfun(obj,g,ndv,nintv,intobj,intcon,facobj,eps,icas,script_name,output_name,fintv)
implicit real*8(a-h,o-z)
dimension intobj(*),intcon(*),facobj(*)
character*70 script_name(*),output_name(*)
dimension g(*),fintv(*)
obj=0.0
ii=0
do i=1,nintv                          : nintv (number of performance indices)
    call system(script_name(i))       : Script commands are executed
c
c     objective function
c
    if(intobj(i).ge.1.and.intobj(i).le.3) then
        open(99,file=output_name(i))  : open external file for each performance index
        read(99,*) objtmp             : read each performance index
        close(99)
        fintv(i)=objtmp               : save performance index for post-processing
        if(intobj(i).eq.1) obj=obj+objtmp*facobj(i)   : facobj(i) = weighting factors
        if(intobj(i).eq.2) obj=obj+objtmp**2*facobj(i)
        if(intobj(i).eq.3) obj=obj+dabs(objtmp)*facobj(i)
    endif
c
c     inequality constraints
c
    if(intcon(i).eq.1) then
        open(99,file=output_name(i))  : open external file for each performance index
        read(99,*) const              : read each performance index
        close(99)
        fintv(i)=const                : save performance index for post-processing
        ii=ii+1
        g(ii)=const-facobj(i)         : facobj(i) = small epsilon values for inequality constraints
    endif
c
c     equality constraints; use Lagrange multiplier
c
    if(intcon(i).eq.2) then
        open(99,file=output_name(i))  : open external file for each performance index
        read(99,*) const              : read each performance index
        close(99)
        fintv(i)=const                : save performance index for post-processing
        obj=obj+facobj(i)*const**2    : facobj(i) = Lagrange multipliers
    endif
enddo
if(icas.eq.1) obj=-obj                : change sign for switching between min and max problems
return
end
```

Appendix B

This appendix presents and explains the input data cards used for the object-oriented optimization (O^3) tool. The appendix expands on the discussion found in the "Applications" section in the body of the report.

B.1. DESVAR

DESVAR: Defines a design variable for design optimization.

Format:

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

Example:

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

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

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≤XSTART≤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)

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 $2.5 \le x < 3.5 : x = 3$ and $3.5 \le x < 4.5 : x = 4$ 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

B.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 B.1, B.2, B.3, and B.4 (character).
VALi	(I10 or F10.5)	Value of the parameter (real or integer, see tables B.1, B.2, B.3, and B.4).

Remark:

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

Table B.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 Central Executive Module (default = 0) Continuous design variables (CDV) Discrete design variables (DDV) = 0: Exit = 1: GA(CDV or DDV) = 2: DOT(CDV) = 3: GA(CDV) + DOT(CDV) = 4: GA(CDV) + DOT(CDV) + GA(DDV) = 5: DOT(CDV) + GA(DDV)

Table B.2. PARi names and descriptions for the genetic algorithm.

Name	Description, type, and default value
IGEN	Number of generations (default = 2)
IPOP	Number of populations (default = 3)

Table B.3. PARi names and descriptions for design optimization tools (integers).

Name	Description, type, and default value
IGMAX	If IGMAX=0, only gradients of active and violated constraints are calculated. If IGMAX>0, up to NGMAX gradients are calculated, including active, violated, and near active constraints (default = 0).
IGRAD	Specifies whether the gradients are calculated (default = 0) = -1 or 0: by DOT = 1: by user
IPRINT	Print control parameter (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	If IPRNT1=1, print scaling factors for the X vector (default = 0)

IPRNT2	If IPRNT2=1, print miscellaneous search information. If IPRNT2=2, turn on print during one-dimensional search process. This is for debugging only (default = 0).
ISCAL	Design variables are rescaled every ISCAL iteration Set ISCAL = -1 to turn off scaling (default = number of design variable).
ITMAX	Maximum number of iterations allowed at optimizer level during each design cycle (default = 100)
ITRMOP	Number of consecutive iterations for which convergence criteria must be satisfied to indicate convergence at the optimizer level (integer; default = 2)
ITRMST	Number of consecutive iterations for which convergence criteria must be met at the optimizer level to indicate convergence in the sequential linear programming method (integer > 0; default = 2)
JPRINT	Sequential linear programming and sequential quadratic programming subproblem print. If JPRINT>0, IPRINT is turned on during approximate subproblem. This is for debugging only (default = 0).
JTMAX	Maximum number of iterations allowed at the optimizer level for the sequential linear programming method. This is the number of linearized subproblems solved (integer $\geq$ 0; default = 50).
JWRITE	File number to which to write iteration history information. This is useful for using post-processing programs to plot the iteration process. This is only used if JWRITE>0 (default = 0).
MAXDOT	Maximum number of DOT optimizations (default = 1)
MAXINT	Maximum integer number that can be defined (default = 2000000000)
METHOD	Optimization method: (integer 0, 1, 2, or 3; default = 1) 0 or 1: Modified method of feasible directions (default) 2: Sequential linear programming 3: Sequential quadratic programming If the problem is unconstrained (NCON=0), the BFGS algorithm will be used if METHOD=0 or 1; the Fletcher-Reeves algorithm will be used if METHOD=2. NCON = number of constraints (automatically counted)
NGMAX	Number of retained constraints used for METHOD=2 or 3. Also, the maximum number of constraints retained for gradient calculations when METHOD=1 (default = NCON, but not more than 2 * NDV) NDV = number of design variables (automatically counted)

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 B.4. PARi names and descriptions for design optimization tools (real numbers).

Name	Description, type, and default value
CT	A constraint is active if its numerical value is more positive than CT. CT is a small negative number (default = -0.03).
CTMIN	A constraint is violated if its numerical value is more positive than CTMIN (default = 0.003).
DABOBJ	Maximum absolute change in the objective between ITRMOP consecutive iterations to indicate convergence in optimization (default = MAX[0.0001*ABS(F0),1.e-20])
DABSTR	Maximum absolute change in the objective between ITRMST consecutive iterations of sequential linear programming and sequential quadratic programming methods to indicate convergence to the optimum (default = 0.003)
DELOBJ	Maximum relative change in the objective between ITRMOP consecutive iterations to indicate convergence in optimization (default = 0.001)
DELSTR	Maximum relative change in the objective between ITRMST consecutive iterations of sequential linear programming method to indicate convergence to the optimum (default = 0.001)
DOBJ1	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).
DOBJ2	Absolute change in the objective function attempted on the first optimization iteration [default = 0.2*ABS(F0)]
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*ABS[x(l)])
FDCH	Relative finite difference step when calculating gradients (default = 0.001)
FDCHM	Minimum absolute value of the finite difference step when calculating gradients. This prevents too small a step when X(l) is near zero (default = 0.0001).
RMVLMZ	Maximum relative change in design variables during the first approximate subproblem in the sequential linear programming method. That is, each design variable is initially allowed to change by +- 40%. This move limit is reduced as the optimization progresses (default = 0.4).

B.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

Appendix C

This appendix contains script files and analysis computer programs used for the six sample problems discussed in the body of the report. A detailed discussion is provided in the "Applications" section above.

C.1. Sample Problem 1

C.1.1. f.bat

```
copy design_variables design_var
obj
```

C.1.2. g.bat

```
const
```

C.1.3. obj.f

```
      implicit real*8(a-h,o-z)
      open(1,file='design_var')
      open(2,file='f.dat')
      read(1,*) dum,x
      y=x**2-2.*x+3.
      write(2,*) y
      stop
      end
```

C.1.4. const.f

```
      implicit real*8(a-h,o-z)
      open(1,file='design_var')
      open(2,file='g.dat')
      read(1,*) dum,x
      y=-x**2+3.*x-1.
      write(2,*) y
      stop
      end
```

C.2. Sample Problem 2

C.2.1. f.bat

```
copy design_variables design_var
obj
```

C.2.2. fdot.bat

```
obj_grad
```

C.2.3. g.bat

```
const
```

C.2.4. gdot.bat

```
const_grad
```

C.2.5. obj.f

```
      implicit real*8(a-h,o-z)
      open(1,file='design_var')
      open(2,file='f.dat')
      read(1,*) dum,x
      y=x**2-2.*x+3.
      write(2,*) y
      stop
      end
```

C.2.6. obj_grad.f

```
      implicit real*8(a-h,o-z)
      open(1,file='design_var')
      open(2,file='fdot.dat')
      read(1,*) dum,x
      y=2.*x-2.
      n=1
      write(2,*) n
      write(2,*) y
      stop
      end
```

C.2.7. const.f

```
      implicit real*8(a-h,o-z)
      open(1,file='design_var')
      open(2,file='g.dat')
      read(1,*) dum,x
      y=-x**2+3.*x-1.
      write(2,*) y
      stop
      end
```

C.2.8. const_grad.f

```
      implicit real*8(a-h,o-z)
      open(1,file='design_var')
      open(2,file='gdot.dat')
      read(1,*) dum,x
      y=-2.*x+3.
      n=1
      write(2,*) n
      write(2,*) y
      stop
      end
```

C.3. Sample Problem 3

C.3.1. f.bat

```
copy design_variables design_var
obj
```

C.3.2. h.bat

```
const
```

C.3.3. obj.f

```
      implicit real*8(a-h,o-z)
      open(1,file='design_var')
      open(2,file='f.dat')
      read(1,*) dum,x1
      read(1,*) dum,x2
      read(1,*) dum,x3
      y=(x1+x2)**2+(x2+x3)**2
      write(2,*) y
      stop
      end
```

C.3.4. const.f

```
      implicit real*8(a-h,o-z)
      open(1,file='design_var')
      open(2,file='h.dat')
      read(1,*) dum,x1
      read(1,*) dum,x2
      read(1,*) dum,x3
      h=x1+2*x2+3*x3-1.
      write(2,*) h
      stop
      end
```

C.4. Sample Problem 4

C.4.1. f.bat

```
copy design_variables design_var
obj
```

C.4.2. fdot.bat

```
obj_grad
```

C.4.3. h.bat

```
const
```

C.4.4. hdot.bat

```
con_grad
```

C.4.5. obj.f

```
implicit real*8(a-h,o-z)
open(1,file='design_var')
open(2,file='f.dat')
read(1,*) dum,x1
read(1,*) dum,x2
read(1,*) dum,x3
y=(x1+x2)**2+(x2+x3)**2
write(2,*) y
stop
end
```

C.4.6. obj_grad.f

```
implicit real*8(a-h,o-z)
dimension ydot(3)
open(1,file='design_var')
open(2,file='fdot.dat')
read(1,*) dum,x1
read(1,*) dum,x2
read(1,*) dum,x3
n=3
ydot(1)=2.*(x1+x2)
ydot(2)=2.*(x1+x2)+2.*(x2+x3)
ydot(3)=2.*(x2+x3)
write(2,*) n
write(2,*) (ydot(i),i=1,n)
stop
end
```

C.4.7. const.f

```
implicit real*8(a-h,o-z)
open(1,file='design_var')
open(2,file='h.dat')
read(1,*) dum,x1
read(1,*) dum,x2
read(1,*) dum,x3
h=x1+2*x2+3*x3-1.
write(2,*) h
stop
end
```

C.4.8. con_grad.f

```
      implicit real*8(a-h,o-z)
      dimension ydot(3)
      open(1,file='design_var')
      open(2,file='hdot.dat')
      read(1,*) dum,x1
      read(1,*) dum,x2
      read(1,*) dum,x3
      n=3
      ydot(1)=1.
      ydot(2)=2.
      ydot(3)=3.
      write(2,*) n
      write(2,*) (ydot(i),i=1,n)
      stop
      end
```

C.5. Sample Problem 5

C.5.1. f.bat

```
copy design_variables design_var
obj
```

C.5.2. g1.bat

```
con1
```

C.5.3. g2.bat

```
con2
```

C.5.4. obj.f

```
      implicit real*8(a-h,o-z)
      open(1,file='design_var')
      open(2,file='f.dat')
      read(1,*) dum,x1
      read(1,*) dum,x2
      y=2.*sqrt(2.)*x1+x2
      write(2,*) y
      stop
      end
```

C.5.5. con1.f

```
implicit real*8(a-h,o-z)
open(1,file='design_var')
open(2,file='g1.dat')
read(1,*) dum,x1
read(1,*) dum,x2
g1=(2.*x1+sqrt(2.)*x2)/(2.*x1*(x1+sqrt(2.)*x2))-1.
write(2,*) g1
stop
end
```

C.5.6. con2.f

```
implicit real*8(a-h,o-z)
open(1,file='design_var')
open(3,file='g2.dat')
read(1,*) dum,x1
read(1,*) dum,x2
g2=1./(x1+sqrt(2.)*x2)-1.
write(3,*) g2
stop
end
```

C.6. Sample Problem 6

C.6.1. f.bat

```
copy design_variables design_var
obj
```

C.6.2. fdot.bat

```
obj_grad
```

C.6.3. g1.bat

```
constraints
```

C.6.4. g1dot.bat

```
con1_grad
```

C.6.5. g2.bat

(empty)

C.6.6. g2dot.bat

```
con2_grad
```

C.6.7. obj.f

```
implicit real*8(a-h,o-z)
open(1,file='design_var')
open(2,file='f.dat')
read(1,*) dum,x1
read(1,*) dum,x2
y=2.*sqrt(2.)*x1+x2
write(2,*) y
stop
end
```

C.6.8. obj_grad.f

```
implicit real*8(a-h,o-z)
dimension ydot(2)
open(1,file='design_var')
open(2,file='fdot.dat')
read(1,*) dum,x1
read(1,*) dum,x2
n=2
ydot(1)=2.*sqrt(2.)
ydot(2)=1.
write(2,*) n
write(2,*) (ydot(i),i=1,n)
stop
end
```

C.6.9. constraints.f

```
implicit real*8(a-h,o-z)
open(1,file='design_var')
open(2,file='g1.dat')
open(3,file='g2.dat')
read(1,*) dum,x1
read(1,*) dum,x2
g1=(2.*x1+sqrt(2.)*x2)/(2.*x1*(x1+sqrt(2.)*x2))-1.
g2=1./(x1+sqrt(2.)*x2)-1.
write(2,*) g1
write(3,*) g2
stop
end
```

C.6.10. con1_grad.f

```
implicit real*8(a-h,o-z)
dimension ydot(2)
open(1,file='design_var')
open(2,file='g1dot.dat')
read(1,*) dum,x1
read(1,*) dum,x2
n=2
d1=(x1+sqrt(2.)*x2)**2
ydot(1)=-(2.*x1*x1+2.*sqrt(2.)*x1*x2+2.*x2*x2)/(2.*x1*x1*d1)
ydot(2)=-1./(sqrt(2.)*d1)
write(2,*) n
write(2,*) (ydot(i),i=1,n)
stop
end
```

C.6.11. con2_grad.f

```
implicit real*8(a-h,o-z)
dimension ydot(2)
open(1,file='design_var')
open(2,file='g2dot.dat')
read(1,*) dum,x1
read(1,*) dum,x2
n=2
d1=(x1+sqrt(2.)*x2)**2
ydot(1)=-0.5/d1
ydot(2)=-sqrt(2.)/d1
write(2,*) n
write(2,*) (ydot(i),i=1,n)
stop
end
```

References

1. 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.

2. Lung, Shun-fat, and Chan-gi Pak, "Updating the Finite Element Model of the Aerostructures Test Wing Using Ground Vibration Test Data," AIAA-2009-2528.

3. Pak, Chan-gi, and Shun-fat Lung, *Reduced Uncertainties in the Flutter Analysis of the Aerostructures Test Wing*, NASA/TM-2011-216421, 2011.

4. *MSC.NASTRAN 2005 quick reference guide*, MSC.Software Corporation, 2004.

5. *Matlab 7 getting started guide*, The MathWorks, Inc., 2008.

6. *ZAERO version 8.0 user's manual*, ZONA Technology, Inc., 2007.

7. Krist, Sherrie L., Robert T. Biedron, and Christopher L. Rumsey, "CFL3D User's Manual (Version 5.0)," NASA-TM-1998-208444, 1998.

8. *DOT design optimization tools user's manual version 5.0*, Vanderplaats Research & Development, Inc., 2001.

9. Charbonneau, P., and B. Knapp, *A user's guide to PIKAIA 1.0*, National Center for Atmospheric Research, 1995.

10. Goldberg, D., *Genetic algorithms in search, optimization, and machine learning*, Addison-Wesley Professional, 1989.

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)* 01-01-2011	**2. REPORT TYPE** Technical Memorandum	**3. DATES COVERED** *(From - To)*
4. TITLE AND SUBTITLE Preliminary Development of an Object-Oriented Optimization Tool		**5a. CONTRACT NUMBER**
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) Pak, Chan-gi		**5d. PROJECT NUMBER**
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) NASA Dryden Flight Research Center P.O. Box 273 Edwards, CA 93523-0273		**8. PERFORMING ORGANIZATION REPORT NUMBER** H-3079
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) National Aeronautics and Space Administration Washington, DC 20546-0001		**10. SPONSORING/MONITOR'S ACRONYM(S)** NASA
		11. SPONSORING/MONITORING REPORT NUMBER NASA/TM-2011-216419

12. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified - Unlimited
Subject Category 59 Availability: NASA CASI (443) 757-5802 Distribution: Standard

13. SUPPLEMENTARY NOTES
Pak, NASA Dryden Flight Research Center

14. ABSTRACT
The National Aeronautics and Space Administration Dryden Flight Research Center has developed a FORTRAN-based object-oriented optimization (O3) tool that leverages existing tools and practices and allows easy integration and adoption of new state-of-the-art software. The 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 central executive module and the discipline modules, or both. Six sample optimization problems are presented. The first four sample problems are based on simple mathematical equations; the fifth and sixth problems consider a three-bar truss, which is a classical example in structural synthesis. Instructions for preparing input data for the O3 tool are presented.

15. SUBJECT TERMS
Central executive module; Genetic algorithm,; Gradient-based optimizer; Multidisciplinary design, analysis, and optimization (MDAO); Quick reference guide

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19b. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			STI Help Desk at e-mail: help@sti.nasa.gov
					19b. TELEPHONE NUMBER ***(Include area code)***
U	U	U	UU	43	(443) 747-5802