Preliminary Development of an Object-Oriented Optimization Tool

Chan-gi Pak

Dryden Flight Research Center, Edwards, California

January 2011
Since its founding, NASA has been dedicated to the advancement of aeronautics and space science. The NASA scientific and technical information (STI) program plays a key part in helping NASA maintain this important role.

The NASA STI program operates under the auspices of the Agency Chief Information Officer. It collects, organizes, provides for archiving, and disseminates NASA’s STI. The NASA STI program provides access to the NASA Aeronautics and Space Database and its public interface, the NASA Technical Report Server, thus providing one of the largest collections of aeronautical and space science STI in the world. Results are published in both non-NASA channels and by NASA in the NASA STI Report Series, which includes the following report types:

- **TECHNICAL PUBLICATION.** Reports of completed research or a major significant phase of research that present the results of NASA Programs and include extensive data or theoretical analysis. Includes compilations of significant scientific and technical data and information deemed to be of continuing reference value. NASA counterpart of peer-reviewed formal professional papers but has less stringent limitations on manuscript length and extent of graphic presentations.

- **TECHNICAL MEMORANDUM.** Scientific and technical findings that are preliminary or of specialized interest, e.g., quick release reports, working papers, and bibliographies that contain minimal annotation. Does not contain extensive analysis.

- **CONTRACTOR REPORT.** Scientific and technical findings by NASA-sponsored contractors and grantees.

- **CONFERENCE PUBLICATION.** Collected papers from scientific and technical conferences, symposia, seminars, or other meetings sponsored or co-sponsored by NASA.

- **SPECIAL PUBLICATION.** Scientific, technical, or historical information from NASA programs, projects, and missions, often concerned with subjects having substantial public interest.

- **TECHNICAL TRANSLATION.** English-language translations of foreign scientific and technical material pertinent to NASA’s mission.

Specialized services also include organizing and publishing research results, distributing specialized research announcements and feeds, providing help desk and personal search support, and enabling data exchange services.

For more information about the NASA STI program, see the following:

- Access the NASA STI program home page at [http://www.sti.nasa.gov](http://www.sti.nasa.gov)

- E-mail your question via the Internet to help@sti.nasa.gov

- Fax your question to the NASA STI Help Desk at 443-757-5803

- Phone the NASA STI Help Desk at 443-757-5802

- Write to: NASA STI Help Desk NASA Center for AeroSpace Information 7115 Standard Drive Hanover, MD 21076-1320
Preliminary Development of an Object-Oriented Optimization Tool

Chan-gi Pak

Dryden Flight Research Center, Edwards, California

January 2011
NOTICE
Use of trade names or names of manufacturers in this document does not constitute an official endorsement of such products or manufacturers, either expressed or implied, by the National Aeronautics and Space Administration.

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

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

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\\textsuperscript{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\\textsuperscript{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\\textsuperscript{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 O³ 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 O³ 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 O³ data input cards DESVAR, DOPTPRM, and INDEX, are explained in appendix B. Free sequence of these input data cards is used in the O³ 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 O³ 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):

Minimize: \( f(x) = x^2 - 2x + 3 \) \hspace{2cm} (1)

Subject to: \( g(x) = -x^2 + 3x - 1 \leq 0 \) \hspace{2cm} (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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Initial x</th>
<th>Optimum x</th>
<th>Objective function f</th>
<th>Constraint function g</th>
<th>Number of function calls</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exact global</td>
<td>N/A</td>
<td>(\frac{3-\sqrt{5}}{2} = 0.38197)</td>
<td>(\frac{7-\sqrt{5}}{2} = 2.3820)</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>Exact local</td>
<td>N/A</td>
<td>(\frac{3+\sqrt{5}}{2} = 2.6180)</td>
<td>(\frac{7+\sqrt{5}}{2} = 4.6180)</td>
<td>0</td>
<td>N/A</td>
</tr>
<tr>
<td>DOT 0.1</td>
<td>0.38197</td>
<td>2.3820</td>
<td>6.2173E-15</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>DOT 4.0</td>
<td>2.6180</td>
<td>4.6180</td>
<td>-1.15463E-14</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>GA 0.1</td>
<td>0.38192</td>
<td>2.3820</td>
<td>1000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GA 4.0</td>
<td>0.38192</td>
<td>2.3820</td>
<td>1000</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For the computer simulation with the O³ 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:

<table>
<thead>
<tr>
<th>DOPTPRM</th>
<th>IOPT2</th>
<th>ICAS</th>
<th>MAXDOT</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>NRWK</td>
<td>1000</td>
<td>NRIWK</td>
<td>500</td>
</tr>
<tr>
<td>+</td>
<td>IGRAD</td>
<td>0</td>
<td>CT</td>
<td>-0.00001</td>
</tr>
<tr>
<td>DESVAR</td>
<td>1</td>
<td>0</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>INDEX</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.0</td>
</tr>
<tr>
<td>+ Sample problem 1: Object function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ f</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ f.dat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
</tr>
<tr>
<td>+ Sample problem 1: Constraint function</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ g</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+ g.dat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 O³ 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:

<table>
<thead>
<tr>
<th>DESVAR</th>
<th>1</th>
<th>0</th>
<th>4.0</th>
<th>-1.0</th>
<th>4.0</th>
</tr>
</thead>
</table>

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:

<table>
<thead>
<tr>
<th>DOPTPRM</th>
<th>IOPT2</th>
<th>2</th>
<th>ICAS</th>
<th>0</th>
<th>MAXDOT</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>NRWK</td>
<td>1000</td>
<td>NRIWK</td>
<td>500</td>
<td>NGMAX</td>
<td>2</td>
</tr>
<tr>
<td>+</td>
<td>IGRAD</td>
<td>0</td>
<td>CT</td>
<td>-0.00001</td>
<td>CTMIN</td>
<td>0.00001</td>
</tr>
<tr>
<td>DESVAR</td>
<td>1</td>
<td>0</td>
<td>4.0</td>
<td>-1.0</td>
<td>4.0</td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>Sample problem 1: Object function</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>f</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>f.dat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>Sample problem 1: Constraint function</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>g</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>g.dat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Sample problem 1: Object function

Sample problem 1: Constraint function
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:

<table>
<thead>
<tr>
<th>DOPTPRM</th>
<th>IOPT2</th>
<th>IPOP</th>
<th>IGEN</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESVAR</td>
<td>1</td>
<td>0.1</td>
<td>0.0</td>
</tr>
<tr>
<td>INDEX</td>
<td>1</td>
<td>0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

+ Sample problem 1: Object function

+ $f$

+ $f$.dat

+ 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):

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

Subject to: $g(x) = -x^2 + 3x - 1 \leq 0$  \hspace{1cm} (2)

with the following user-supplied “analytical” gradients:

$$\frac{df(x)}{dx} = 2x - 2$$  \hspace{1cm} (3)

$$\frac{dg(x)}{dx} = -2x + 3$$  \hspace{1cm} (4)

Input data cards for the DOT simulation are as follows:

<table>
<thead>
<tr>
<th>DOPTPRM</th>
<th>IOPT2</th>
<th>ICAS</th>
<th>MAXDOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ NRWK</td>
<td>1000</td>
<td>500</td>
<td>NGMAX 2</td>
</tr>
<tr>
<td>+ IGRAD</td>
<td>1</td>
<td>-0.00001</td>
<td>CTMIN 0.00001</td>
</tr>
<tr>
<td>DESVAR</td>
<td>1</td>
<td>0.1</td>
<td>0.0</td>
</tr>
</tbody>
</table>

+ 2.0
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.

![Diagram of the problem structure of sample problem 2.](image)

Figure 4. The problem structure of sample problem 2.
Table 2. Results from sample problem 2.

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

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.

Minimize: \( f(X_1, X_2, X_3) = (X_1 + X_2)^2 + (X_2 + X_3)^2 \)  \( (5) \)

Subject to: \( h(X_1, X_2, X_3) = X_1 + 2X_2 + 3X_3 - 1 = 0 \)  \( (6) \)

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.

![Figure 5. The problem structure of sample problem 3.](image)
The exact solution can be obtained as follows: Rewrite the equality constraint, equation (6), as shown by equation (7):

\[ X_1 = 1 - 2X_2 - 3X_3 \]  \hspace{1cm} (7)

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

\[ f(X_2, X_3) = (1 - X_2 - 3X_3)^2 + (X_2 + X_3)^2 \]  \hspace{1cm} (8)

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

\[ \frac{\partial f(X_2, X_3)}{\partial X_2} = -2(1 - X_2 - 3X_3) + 2(X_2 + X_3) = 4X_2 + 8X_3 - 2 = 0 \]  \hspace{1cm} (9)

\[ \frac{\partial f(X_2, X_3)}{\partial X_3} = -6(1 - X_2 - 3X_3) + 2(X_2 + X_3) = 8X_2 + 20X_3 - 6 = 0 \]  \hspace{1cm} (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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Exact</th>
<th>Results in reference 8 using two inequality constraints</th>
<th>DOT in this study using Lagrange multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design variable ( X_1 )</td>
<td>0.5</td>
<td>0.475</td>
<td>.501</td>
</tr>
<tr>
<td>Design variable ( X_2 )</td>
<td>-0.5</td>
<td>-0.469</td>
<td>-.500</td>
</tr>
<tr>
<td>Design variable ( X_3 )</td>
<td>0.5</td>
<td>0.488</td>
<td>.500</td>
</tr>
<tr>
<td>Objective function ( f )</td>
<td>0</td>
<td>3.8e-4</td>
<td>1.4e-06</td>
</tr>
<tr>
<td>Constraint function ( g_1 )</td>
<td>N/A</td>
<td>8.4e-5</td>
<td>N/A</td>
</tr>
<tr>
<td>Number of function calls</td>
<td>N/A</td>
<td>43</td>
<td>43</td>
</tr>
</tbody>
</table>

In the O3 tool, an equality constraint \( h \) in equation (6) is added to the objective function using the Lagrange multiplier \( \lambda \) as shown in equation (11):

\[ f(X_1, X_2, X_3, \lambda) = f(X_1, X_2, X_3) + \lambda h(X_1, X_2, X_3) \]  \hspace{1cm} (11)

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

<table>
<thead>
<tr>
<th>DOPTPRM</th>
<th>IOPT2</th>
<th>ICAS</th>
<th>ICAS</th>
<th>MAXDOT</th>
<th>NGMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>NRWK</td>
<td>1000</td>
<td>NRIWK</td>
<td>500</td>
<td>2</td>
</tr>
<tr>
<td>+</td>
<td>IGRAD</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESVAR</td>
<td>1</td>
<td>0</td>
<td>-4.0</td>
<td>-10.0</td>
<td>10.0</td>
</tr>
</tbody>
</table>
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,

\[ g_1 = -(X_1 + 2X_2 + 3X_3 - 1) \leq \varepsilon \]
\[ g_2 = X_1 + 2X_2 + 3X_3 - 1 \leq \varepsilon \]

where, \( \varepsilon \) = a small number, are used to solve the problem with the equality constraint in reference 8.

Instead of solving equation (6), the following equation

\[-\varepsilon \leq X_1 + 2X_2 + 3X_3 - 1 \leq \varepsilon \]

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 \( \varepsilon \). When \( \varepsilon \) is a large number, good convergence and bad accuracy can result. On the other hand, when a very small number is selected for the \( \varepsilon \) 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.

Minimize:  
\[ f(X_1, X_2, X_3) = (X_1 + X_2)^2 + (X_2 + X_3)^2 \]  
(5)

Subject to:  
\[ h(X_1, X_2, X_3) = X_1 + 2X_2 + 3X_3 - 1 = 0 \]  
(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:

<table>
<thead>
<tr>
<th>DOPTPRM</th>
<th>IOPT2</th>
<th>ICAS</th>
<th>MAXDOT</th>
<th>NRWK</th>
<th>NRIWK</th>
<th>NGMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESVAR</td>
<td>1</td>
<td>0</td>
<td>-4.0</td>
<td>-10.0</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>DESVAR</td>
<td>2</td>
<td>0</td>
<td>1.0</td>
<td>-10.0</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>DESVAR</td>
<td>3</td>
<td>0</td>
<td>2.0</td>
<td>-10.0</td>
<td>10.0</td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.0</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

+ 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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Exact</th>
<th>DOT in this study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design variable X₁</td>
<td>0.5</td>
<td>.500</td>
</tr>
<tr>
<td>Design variable X₂</td>
<td>-0.5</td>
<td>-.500</td>
</tr>
<tr>
<td>Design variable X₃</td>
<td>0.5</td>
<td>.500</td>
</tr>
<tr>
<td>Objective function f</td>
<td>0</td>
<td>1.6e-27</td>
</tr>
<tr>
<td>Number of function calls</td>
<td>N/A</td>
<td>16</td>
</tr>
<tr>
<td>Number of gradient calls</td>
<td>N/A</td>
<td>5</td>
</tr>
</tbody>
</table>

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.
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:

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

$$0.01 \leq X_i \leq 2 \quad i = 1, 2$$

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.
Figure 8. The problem structure of sample problem 5.

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

<table>
<thead>
<tr>
<th>DOPTPRM</th>
<th>IOPT2</th>
<th>ICAS</th>
<th>MAXDOT</th>
<th>NRWK</th>
<th>NRIWK</th>
<th>NGMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td>0</td>
<td></td>
<td>1000</td>
<td>500</td>
<td>1</td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1.0</td>
<td>0.01</td>
<td>0</td>
<td>2.0</td>
</tr>
<tr>
<td>DESVAR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESVAR</td>
<td>2</td>
<td>0</td>
<td>1.0</td>
<td>0.01</td>
<td>0</td>
<td>2.0</td>
</tr>
<tr>
<td>INDEX</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
| +       | 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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>DOT in this study</th>
<th>DOT in reference 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design variable X_1</td>
<td>.799</td>
<td>.799</td>
</tr>
<tr>
<td>Design variable X_2</td>
<td>.372</td>
<td>.372</td>
</tr>
<tr>
<td>Objective function f</td>
<td>2.633</td>
<td>2.633</td>
</tr>
<tr>
<td>Constraint function g_1</td>
<td>2.8e-03</td>
<td>2.8e-3</td>
</tr>
<tr>
<td>Constraint function g_2</td>
<td>-2.5e-01</td>
<td>-6.2e-1</td>
</tr>
<tr>
<td>Number of function calls</td>
<td>29</td>
<td>29</td>
</tr>
</tbody>
</table>

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,

Minimize: \( f(X_1, X_2) = 2\sqrt{2}X_1 + X_2 \)

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

\[
0.01 \leq X_i \leq 2 \quad i = 1, 2
\]

with the following user-supplied gradients:

\[
\frac{df(X_1, X_2)}{dX_1} = 2\sqrt{2}
\]

\[
\frac{df(X_1, X_2)}{dX_2} = 1
\]

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

\[
\frac{dg_1(X_1, X_2)}{dX_2} = \frac{\sqrt{2}\left(2X_1^2 + 2\sqrt{2}X_1X_2\right) - \left(2X_1 + \sqrt{2}X_2\right)2\sqrt{2}X_1}{\left(2X_1^2 + 2\sqrt{2}X_1X_2\right)^2} = \frac{1}{\sqrt{2}\left(X_1 + \sqrt{2}X_2\right)^2}
\]
\[
\frac{dg_2(X_1, X_2)}{dX_1} = \frac{-1}{(X_1 + \sqrt{2}X_2)^2}
\]

\[
\frac{dg_2(X_1, X_2)}{dX_2} = \frac{-\sqrt{2}}{(X_1 + \sqrt{2}X_2)^2}
\]

The input data cards are as follows:

<table>
<thead>
<tr>
<th>DOPTPRM</th>
<th>IOPT2</th>
<th>ICAS</th>
<th>MAXDOT</th>
<th>NRWK</th>
<th>NRIWK</th>
<th>NGMAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DESVAR</td>
<td>1</td>
<td>0</td>
<td>1.0</td>
<td>0.1</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>DESVAR</td>
<td>2</td>
<td>0</td>
<td>1.0</td>
<td>0.1</td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1.0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>+</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 O3 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.

<table>
<thead>
<tr>
<th>Variable</th>
<th>DOT in this study</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design variable $X_1$</td>
<td>.799</td>
</tr>
<tr>
<td>Design variable $X_2$</td>
<td>.371</td>
</tr>
<tr>
<td>Objective function $f$</td>
<td>2.633</td>
</tr>
<tr>
<td>Constraint function $g_1$</td>
<td>2.9e-03</td>
</tr>
<tr>
<td>Constraint function $g_2$</td>
<td>-2.5e-01</td>
</tr>
<tr>
<td>Number of function calls</td>
<td>23</td>
</tr>
<tr>
<td>Number of gradient calls</td>
<td>3</td>
</tr>
</tbody>
</table>

Figure 9. The problem structure of unsteady aerodynamic model tuning.
The NASA Dryden Flight Research Center FORTRAN-based object-oriented optimization (O³) tool is developed and demonstrated. The feasibilities of O³ tool leveraging with other executable codes are shown by way of simple mathematical equations that also enable understanding of the basic concept of the O³ tool. The results demonstrate the flexibility of the O³ 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 (O3) tool. Further discussion is given in the “Background” section in the body of the report.

**subroutine** objfun(obj,g,nadv,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
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
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
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
endo
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 (O³) 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:

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>DESVAR</td>
<td>ID</td>
<td>IOPT</td>
<td>XSTART</td>
<td>XL</td>
<td>XU</td>
<td>TABLE</td>
</tr>
</tbody>
</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 \(2.5 \leq x < 3.5: x = 3\) and \(3.5 \leq x < 4.5: x = 4\) then
\[
\begin{align*}
\text{cdv}(L) &= 2.5 & \text{cdv}(U) &= 3.5 & \text{fix} &= 3.0 \\
\text{cdv}(L) &= 3.5 & \text{cdv}(U) &= 4.5 & \text{fix} &= 4.0
\end{align*}
\]

ex 2) if \(2.0 \leq x < 3.0: x = 2\) and \(3.0 \leq x < 4.0: x = 3\) then
\[
\begin{align*}
\text{cdv}(L) &= 2.0 & \text{cdv}(U) &= 3.0 & \text{fix} &= 2.0 \\
\text{cdv}(L) &= 3.0 & \text{cdv}(U) &= 4.0 & \text{fix} &= 3.0
\end{align*}
\]

B.2. DOPTPRM

DOPTPRM: Override default values of parameters used in design.

**Format:**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOPTPRM</td>
<td>PAR1</td>
<td>VAL1</td>
<td>PAR2</td>
<td>VAL2</td>
<td>PAR3</td>
<td>VAL3</td>
</tr>
<tr>
<td>+</td>
<td>PAR4</td>
<td>VAL4</td>
<td>-etc.-</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example:**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOPTPRM</td>
<td>IOPT2</td>
<td>2</td>
<td>ICAS</td>
<td>0</td>
<td>MAXDOT</td>
<td>1</td>
</tr>
<tr>
<td>+</td>
<td>NRWK</td>
<td>1000</td>
<td>NRIWK</td>
<td>500</td>
<td>NGMAX</td>
<td>2</td>
</tr>
<tr>
<td>+</td>
<td>IGRAD</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description, type, and default value</th>
</tr>
</thead>
</table>
| 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description, type, and default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGEN</td>
<td>Number of generations (default = 2)</td>
</tr>
<tr>
<td>IPOP</td>
<td>Number of populations (default = 3)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description, type, and default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IGMAX</td>
<td>If IGMAX=0, only gradients of active and violated constraints are calculated. If IGMAX&gt;0, up to NGMAX gradients are calculated, including active, violated, and near active constraints (default = 0).</td>
</tr>
</tbody>
</table>
| 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 |
<p>| IPRNT1 | If IPRNT1=1, print scaling factors for the X vector (default = 0) |</p>
<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPRNT2</td>
<td>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).</td>
</tr>
<tr>
<td>ISCAL</td>
<td>Design variables are rescaled every ISCAL iteration. Set ISCAL = -1 to turn off scaling (default = number of design variable).</td>
</tr>
<tr>
<td>ITMAX</td>
<td>Maximum number of iterations allowed at optimizer level during each design cycle (default = 100).</td>
</tr>
<tr>
<td>ITRMOP</td>
<td>Number of consecutive iterations for which convergence criteria must be satisfied to indicate convergence at the optimizer level (integer; default = 2).</td>
</tr>
<tr>
<td>ITRMST</td>
<td>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 &gt; 0; default = 2).</td>
</tr>
<tr>
<td>JPRINT</td>
<td>Sequential linear programming and sequential quadratic programming subproblem print. If JPRINT&gt;0, IPRINT is turned on during approximate subproblem. This is for debugging only (default = 0).</td>
</tr>
<tr>
<td>JTMAX</td>
<td>Maximum number of iterations allowed at the optimizer level for the sequential linear programming method. This is the number of linearized subproblems solved (integer ≥ 0; default = 50).</td>
</tr>
<tr>
<td>JWRITE</td>
<td>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&gt;0 (default = 0).</td>
</tr>
<tr>
<td>MAXDOT</td>
<td>Maximum number of DOT optimizations (default = 1)</td>
</tr>
<tr>
<td>MAXINT</td>
<td>Maximum integer number that can be defined (default = 2000000000)</td>
</tr>
<tr>
<td>METHOD</td>
<td>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).</td>
</tr>
<tr>
<td>NGMAX</td>
<td>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).</td>
</tr>
</tbody>
</table>
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).

<table>
<thead>
<tr>
<th>Name</th>
<th>Description, type, and default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CT</td>
<td>A constraint is active if its numerical value is more positive than CT. CT is a small negative number (default = -0.03).</td>
</tr>
<tr>
<td>CTMIN</td>
<td>A constraint is violated if its numerical value is more positive than CTMIN (default = 0.003).</td>
</tr>
<tr>
<td>DABOBJ</td>
<td>Maximum absolute change in the objective between ITRMOP consecutive iterations to indicate convergence in optimization (default = MAX[0.0001*ABS(F0),1.e-20])</td>
</tr>
<tr>
<td>DABSTR</td>
<td>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)</td>
</tr>
<tr>
<td>DELOBJ</td>
<td>Maximum relative change in the objective between ITRMOP consecutive iterations to indicate convergence in optimization (default = 0.001)</td>
</tr>
<tr>
<td>DELSTR</td>
<td>Maximum relative change in the objective between ITRMST consecutive iterations of sequential linear programming method to indicate convergence to the optimum (default = 0.001)</td>
</tr>
<tr>
<td>DOBJ1</td>
<td>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).</td>
</tr>
<tr>
<td>DOBJ2</td>
<td>Absolute change in the objective function attempted on the first optimization iteration [default = 0.2*ABS(F0)]</td>
</tr>
<tr>
<td>DX1</td>
<td>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).</td>
</tr>
</tbody>
</table>
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)])

Relative finite difference step when calculating gradients (default = 0.001)

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).

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:**

<table>
<thead>
<tr>
<th>INDEX</th>
<th>ID</th>
<th>INTOBJ</th>
<th>INTCON</th>
<th>FACOBJ</th>
<th>INTGRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>TASK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>SCRIPT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>OUTPUT</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>SCRIPT GRAD (needed when INTGRA=1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+</td>
<td>OUTPUT GRAD (needed when INTGRA=1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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
<table>
<thead>
<tr>
<th>INDEX</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1.0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total weight: Based on analytical equation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>f</td>
<td>f.dat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>fdot</td>
<td>fdot.dat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>First inequality constraint: Based on analytical equation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g1.dat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g1dot</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g1dot.dat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>INDEX</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0.0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Second inequality constraint: Based on analytical equation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g2.dat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g2dot</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>g2dot.dat</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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`

```fortran
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`

```fortran
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`

```fortran
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`

```fortran
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
derend

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
derend

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

```fortran
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

```bash
copy design_variables design_var
obj
```

C.5.2. g1.bat

```bash
con1
```

C.5.3. g2.bat

```bash
con2
```

C.5.4. obj.f

```fortran
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**

```fortran
implicit real*8(a-h,o-z)
open(1,file='design_var')
open(2,file='g1.dat')
read(1,*) dum,x1
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**

```fortran
implicit real*8(a-h,o-z)
open(1,file='design_var')
open(3,file='g2.dat')
read(1,*) dum,x1
dum,x2

g2=1./(x1+sqrt(2.)*x2)-1.
write(3,*) g2
stop
end
```

C.6. **Sample Problem 6**

C.6.1. **f.bat**

```bash
copy design_variables design_var
```

C.6.2. **fdot.bat**

```bash
obj_grad
```

C.6.3. **g1.bat**

```bash
constraints
```

C.6.4. **g1dot.bat**

```bash
con1_grad
```

C.6.5. **g2.bat**

```bash
(EMPTY)
```

C.6.6. **g2dot.bat**

```bash
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*

```fortran
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*

```fortran
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


Preliminary Development of an Object-Oriented Optimization Tool

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.

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