AIAA 79-0723R

Adaptable Structural Synthesis Using Advanced Analysis and Optimization Coupled by a Computer Operating System

J. Sobieszczanski-Sobieski* and R. B. Bhatt†
NASA Langley Research Center, Hampton, Va.

A finite element program is linked with a general purpose optimization program in a "programing system" that includes user supplied codes that contain problem dependent formulations of the design variables, objective function, and constraints. The result is a system adaptable to a wide spectrum of structural optimization problems. In a sample of numerical examples, the design variables are the cross-sectional dimensions and the parameters of overall shape geometry; constraints are applied to stresses, displacements, buckling, and vibration characteristics, and structural mass is the objective function. Thin-walled, built-up structures and frameworks are included in the sample. Details of the system organization and characteristics of the component programs are given.

Nomenclature

$F(\ )$	= objective function (merit function)
$f(\ )$	= function of the design variables v, used in lieu of $F(\ )$ and $g(\ )$
$g_j(\ )$	= constraint function of design variables
i	= design variable subscript
j	= constraint function subscript
K	= stiffness matrix
L	= load vector
m	= number of constraints
n	= number of design variables
u	= displacements of nodes in a finite element model
v	= vector of design variables, v_i
w	= mass

Introduction

ACCEPTANCE of structural optimization methods in design practice has not kept pace with recent significant progress in theoretical developments. It appears that the prevailing cause of the acceptance lag lies in the disappointment a practicing engineer is likely to experience when attempting to use a typical currently available optimization program in a particular application. Often it is found that the program is not efficient enough for a production application, difficult to use without knowledge of the inner workings, and not sufficiently general to fit the problem at hand.

These difficulties stem from two sources. First, most available structural optimization systems have been aimed at development and demonstration of methodology. Consequently, they combine either sophisticated analysis with only rudimentary resizing capability,[1] or advanced optimization techniques with analysis capability limited to the extent necessary to test the optimization part of the package.[2] Second, each design situation tends to be unique and may require a particular formulation of the design variables, objective function, and constraints; and it is practically impossible in development of optimization codes to anticipate and accommodate all such potentially possible formulations. The result usually is software that accepts only the variable, objective function, and constraint formulations consistent with the developer's preconceived notions.

One approach to remedy the poor efficiency and difficulty of use is to exploit the full potential of a production structural analysis code coupled with a quality optimization algorithm in an integrated system that appears to a user as a "black box." A notable example of such an approach is given in Ref. 3.

This report deals with an alternative approach of a "programing system" (a term introduced in Ref. 4) which consists of an analysis program, a general purpose optimization program, and two problem dependent interface programs that make it adaptable to any particular application. It is explained how this approach achieves a generality unattainable in the usual black box approach, while retaining efficiency and ease of use. Once the programing system is adapted to a particular problem, it can then be operated in effect as a black box in production applications.

A description is given of the system's components, organization (based on use of a standard computer operating system as a connecting network), and execution options. This description is given in general terms to emphasize that the approach is independent of the particular components used to build the system and that it would apply also to nonstructural problems. The description also includes information on implementation of a specific programing system for structural synthesis (PROSSS). This system consists of a finite element program SPAR,[5,6] and a general purpose optimization program CONMIN[7] and is based on the CDC-NOS computer operating system.[8] The system's performance is demonstrated by numerical examples involving a variety of variables governing cross-sectional sizing and overall geometry, and constraints that involve statics, dynamics and instability.

Components and Organization of the System

This section describes the computer programs and the connecting network that constitute a programing system. The function of the system is to find a vector of design variables v_i that minimize $F(v)$ while satisfying constraint equations. A design variable is a quantity whose change alters behavior of the object of optimization, e.g., cross-sectional area of a structural member. Constraints are numerically formulated conditions that the object of optimization must satisfy to be acceptable, e.g., stress to be less than an allowable. In mathematical notation

$$F(v) \rightarrow \min \tag{1a}$$

subject to

$$g_j(v) \leq 0 \qquad 1 \leq j \leq m \tag{1b}$$

Presented as Paper 79-0723 at the AIAA/ASME/ASCE/AHS 20th Structures, Structural Dynamics and Materials Conference, St. Louis, Mo., April 4-6, 1979; submitted May 2, 1979; revision received March 31, 1980. This paper is declared a work of the U.S. Government and therefore is in the public domain.

*Senior Research Scientist. Associate Fellow AIAA.

†Visiting Scientist, George Washington University Center.

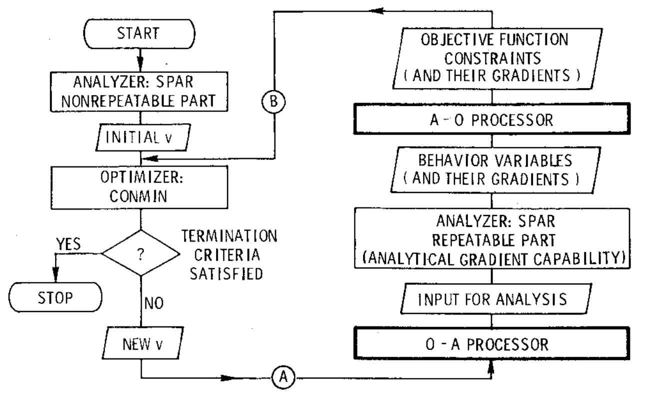

START
ANALYZER: SPAR NONREPEATABLE PART
INITIAL v
OPTIMIZER: CONMIN
TERMINATION CRITERIA SATISFIED
?
YES
NO
STOP
NEW v
A
B
OBJECTIVE FUNCTION CONSTRAINTS (AND THEIR GRADIENTS)
A - O PROCESSOR
BEHAVIOR VARIABLES (AND THEIR GRADIENTS)
ANALYZER: SPAR REPEATABLE PART (ANALYTICAL GRADIENT CAPABILITY)
INPUT FOR ANALYSIS
O - A PROCESSOR

Fig. 1 Basic flow of an optimization procedure.

The basic flow of the procedure to solve Eqs. (1) is shown in Fig. 1. The principal components of the system are, in general terms: optimizer, analyzer, processors interfacing optimizer to analyzer (O-A) and vice versa (A-O), and the connecting framework. The flow chart in Fig. 1, and other flow charts shown subsequently, are general and do not depend on the particular programs used as the principal components.

Analyzer

The function of the analyzer is to compute values of the behavior variables which characterize the physical object's response to the input quantities.

Overall Characteristics

In PROSSS, the analyzer is the finite element program SPAR documented in Ref. 5. SPAR was selected for the analyzer's function because of its computer efficiency, modularity, and data base capability. Input quantities consist of structural cross-sectional dimensions, material properties, element connectivity data, nodal point coordinates, and loads. Output quantities consist of displacements, internal forces, stresses, eigenvalues, and eigenmodes for vibration and buckling, etc. Another output quantity is the structural mass that is commonly used as the objective function, a measure of goodness. The library of finite elements in SPAR is adequate for analysis of skeletal and thin-walled structures. SPAR is a collection of individual programs (processors) that communicate with each other through a data base. The data base consists of one or more files which contain data sets output from the different processors. Each data set has a specific identifying name with which any processor can access it for input. Subroutines documented in Ref. 6 are available to store and retrieve the SPAR data sets by name from the SPAR data base. These subroutines can be executed by Fortran CALL statements and hence can be used to make the SPAR data storage accessible to non-SPAR Fortran programs.

SPAR executes on a processor-by-processor basis; each processor execution is commanded by a separate explicit command. A string of such commands interlaced with the input numerical data is written by the user for the problem at hand, and is called a runstream. The data base facilitates an efficient and selective data transfer from SPAR to other programs, and the individual processor control allows limiting the number of processors executed repetitively in an optimization loop to a minimum.

Computation of Gradients

Most of the efficient mathematical optimization algorithms require not only the objective function and the constraint values but also their gradients, all evaluated for a given set of input values of the design variables v_i. Therefore the gradients are indicated in parentheses in Fig. 1 as an optional output of the analyzer. The gradients can be computed by a finite difference technique or by an analytical technique. An example of analytical gradient is differentiation of the matrix load-deflection equation $K \cdot u = L$ with respect to a design variable v_i. The result is a matrix equation

$$K\frac{\partial u}{\partial v_i} = \frac{\partial K}{\partial v_i} \cdot u + \frac{\partial L}{\partial v_i}$$

from which $\partial u / \partial v_i$ can be obtained at a relatively small computational cost by reusing the previously decomposed stiffness matrix K, as shown in Refs. 9 and 10. In SPAR, the analytical gradient computation is implemented by means of runstreams established specifically for this purpose. These runstreams are dependent only on the types of finite elements used in the mathematical model and therefore are a permanent part of PROSSS.

Optimizer

The function of the optimizer is to calculate a new vector of design variables v on the basis of the values of the objective function and the constraints, and, optionally, their gradients returned by the analyzer in response to a previously defined vector v.

In PROSSS, the optimizer is the program CONMIN[9] which is based on the mathematical nonlinear programing technique of feasible-usable directions. In this report CONMIN is viewed as a black box and attention is focused on the type of data it requires from the rest of the system, and on its execution options, since these features influence organization of the programing system.

The following execution modes are available in CONMIN: 1) execution that requires current values of the objective function and constraints; 2) execution that requires current values of the objective function, constraints and their gradients; and 3) execution accelerated because of linearity of either the objective function and/or constraints.

Program CONMIN consists of a subroutine CONMIN containing the optimization algorithm whose execution options and termination criteria are user controlled by a set of input parameters. The subroutine is called by a main program whose construction depends upon the computational size of the problem. For problems that are computationally small, the main program may contain a block of code to calculate the objective function, constraints, and their gradients, as shown in Fig. 2a. For larger problems, it may be convenient to separate that block of code into a subroutine as indicated in Fig. 2b. Finally, Fig. 2c shows an organization for problems so large computationally that a stand-alone program is required for computation of the objective function and constraints. In this case, the main program calls the subroutine CONMIN, saves intermediate data, and stops, ready to be restarted after execution of the analysis program has been completed. The versions of the main program used in PROSSS are those shown in Figs. 2a and c.

Interface Processors

The optimizer provides input information to the repeatable part of the analyzer through an optimizer to analyzer (O-A) processor and the analyzer supplies the information to the optimizer through the analyzer to optimizer (A-O) processor. The O-A and A-O processors are user supplied and problem dependent. Capability of adding these two programs is the basis for the system's generality.

Optimizer to Analyzer Processor

The function of the O-A processor is to convert the design variables to a set of input parameters written in a format required by the analyzer. In the case of structural optimization, these parameters are structural member sizes and nodal point coordinate data that are actual physical design variables and that are seldom in a one-to-one direct

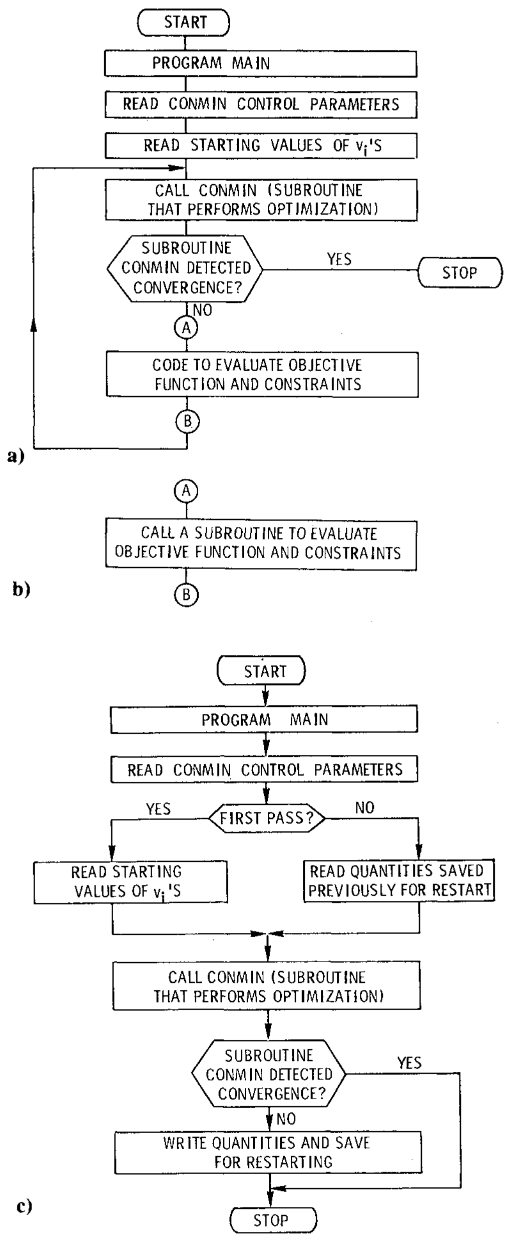

START
PROGRAM MAIN
READ CONMIN CONTROL PARAMETERS
READ STARTING VALUES OF v_i'S
CALL CONMIN (SUBROUTINE THAT PERFORMS OPTIMIZATION)
SUBROUTINE CONMIN DETECTED CONVERGENCE?
YES
STOP
NO
A
CODE TO EVALUATE OBJECTIVE FUNCTION AND CONSTRAINTS
B
a)
CALL A SUBROUTINE TO EVALUATE OBJECTIVE FUNCTION AND CONSTRAINTS
b)
FIRST PASS?
READ STARTING VALUES OF v_i'S
READ QUANTITIES SAVED PREVIOUSLY FOR RESTART
WRITE QUANTITIES AND SAVE FOR RESTARTING
c)

Fig. 2 Organization of the user supplied MAIN program for execution of optimization subroutine CONMIN: a) analysis embedded in main program, b) analysis contained in a subroutine, and c) analysis in another separate program.

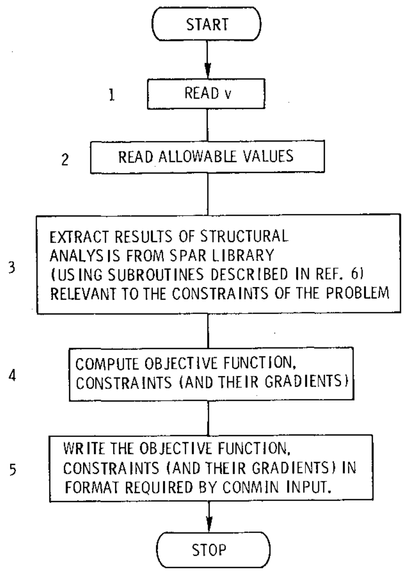

START
1 READ v
2 READ ALLOWABLE VALUES
3 EXTRACT RESULTS OF STRUCTURAL ANALYSIS FROM SPAR LIBRARY (USING SUBROUTINES DESCRIBED IN REF. 6) RELEVANT TO THE CONSTRAINTS OF THE PROBLEM
4 COMPUTE OBJECTIVE FUNCTION, CONSTRAINTS (AND THEIR GRADIENTS)
5 WRITE THE OBJECTIVE FUNCTION, CONSTRAINTS (AND THEIR GRADIENTS) IN FORMAT REQUIRED BY CONMIN INPUT.
STOP

Fig. 3 A-O processor flow chart.

equivalence relationship to the design variables output by the optimizer. Thus, in a typical application, the conversions within the A-O processor are not limited to format changes only but also include such commonly used techniques as variable linking, scaling, change from direct to reciprocal variables, etc. (e.g., Ref. 11). In PROSSS, the O-A processor reads an output vector v from CONMIN, computes the structural parameters, and embeds them in a runstream written for the SPAR execution.

Analyzer to Optimizer Processor

The function of the A-O processor is to compute the objective function, the constraints, and their gradients, if required, and to provide them in the format required by the optimizer. To do so the A-O processor extracts the pertinent behavior variables, defined as quantities characterizing response of the object of optimization to external stimuli such as stresses, displacements, natural vibration frequencies and mode vectors, buckling loads, etc., from the SPAR data base and combines them with the allowable values to form the constraint equations. Frequently the allowable values are functions of v (instead of being constants) as, for example, in the case of local buckling constraints (e.g., Ref. 12). Computation of such variable allowable values can be included among the A-O processor's functions. The A-O processor may also be equipped with a logic to limit the set of constraints to those whose probability of remaining or becoming active is high, as proposed in Ref. 11. Organization of the A-O processor, illustrated by the flow chart in Fig. 3, is problem independent, but the processor contains a section of code (box 4) which does depend on the problem at hand and must be tailored to it. In addition, the parameters in the call statements to the subroutines (box 3) (see Ref. 6) that access the SPAR data base depend on the kind of data sets that need to be extracted as required by the particular constraints and objective function.

Connecting Network

A connecting network (executive software) is required to carry out a computational process such as shown in Fig. 1. It is also required to enable the user to monitor progress of the optimization process, and to stop and restart without loss of information generated before the interruption.

In PROSSS, the CDC-NOS (network operating system) documented in Ref. 8 serves as the connecting network using the approach described in Ref. 13. The CDC-NOS furnishes the user with a repertory of commands [job control language, (JCL)] for executing programs in sequences, including if-test branching and transferring to a labeled statement, and for manipulation of permanent and temporary files. These capabilities are common in most current operating systems, consequently such systems as IBM's CMS or UNIVAC's Exec 8 could function as a connecting network equally as well as CDC-NOS.

Execution Options

A variety of execution flows can be set up using the components described previously. Organization of each flow option depends on how the optimizer is used, and on whether gradients are required as input to the optimizer and, if so, whether these gradients are generated analytically or by finite differences. The flow options discussed in this report are the five shown in Table 1.

Basic Flow Options

The two optimization procedures defining rows of Table 1 are: nonlinear mathematical programing (NLP) and piecewise

Table 1 Optimization flow options

Method	No. gradients supplied to optimizer	Gradients supplied to optimizer Finite difference	Analytical
NLP	1.1	1.2	1.3
PLA	Not applicable	2.2	2.3

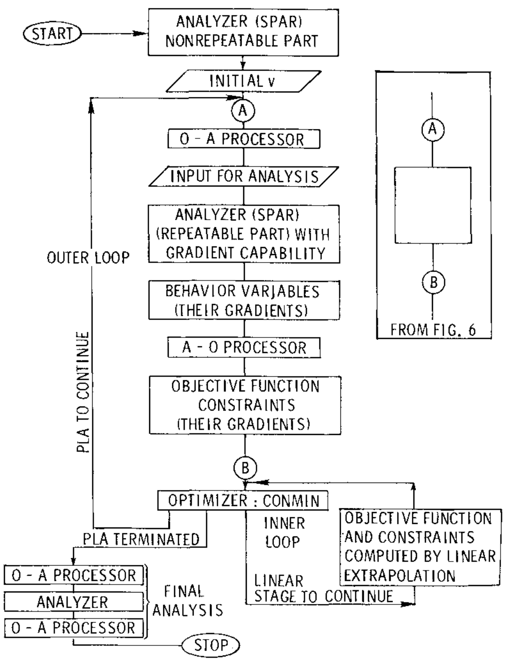

START
ANALYZER (SPAR) NONREPEATABLE PART
INITIAL v
A
0 - A PROCESSOR
INPUT FOR ANALYSIS
ANALYZER (SPAR) (REPEATABLE PART) WITH GRADIENT CAPABILITY
OUTER LOOP
BEHAVIOR VARIABLES (THEIR GRADIENTS)
A - 0 PROCESSOR
PLA TO CONTINUE
OBJECTIVE FUNCTION CONSTRAINTS (THEIR GRADIENTS)
B
OPTIMIZER : CONMIN
INNER LOOP
OBJECTIVE FUNCTION AND CONSTRAINTS COMPUTED BY LINEAR EXTRAPOLATION
LINEAR STAGE TO CONTINUE
PLA TERMINATED
0 - A PROCESSOR
ANALYZER
0 - A PROCESSOR
FINAL ANALYSIS
STOP
FROM FIG. 6

Fig. 4 Flow chart for option 2.3 (Table 1).

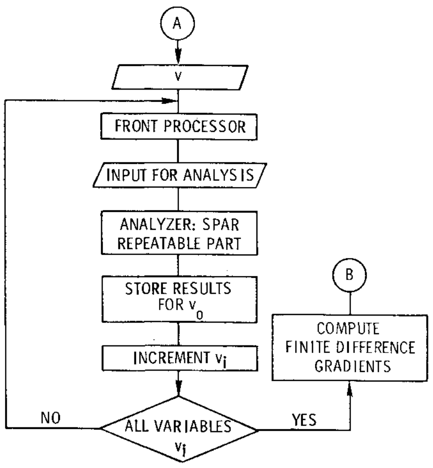


Fig. 5 Flow chart for finite-difference gradient computation to be substituted for the portion of the flow chart between connectors A and B in Fig. 1 to form option 1.2.

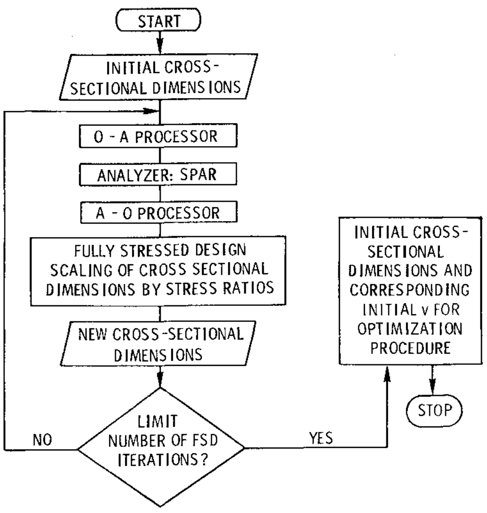

START
INITIAL CROSS-SECTIONAL DIMENSIONS
0 - A PROCESSOR
ANALYZER: SPAR
A - 0 PROCESSOR
FULLY STRESSED DESIGN SCALING OF CROSS SECTIONAL DIMENSIONS BY STRESS RATIOS
NEW CROSS-SECTIONAL DIMENSIONS
NO
LIMIT NUMBER OF FSD ITERATIONS?
YES
INITIAL CROSS-SECTIONAL DIMENSIONS AND CORRESPONDING INITIAL v FOR OPTIMIZATION PROCEDURE
STOP

Fig. 6 Flow chart for generation of the initial fully stressed design (FSD).

linear approximations (PLA). Under the conventional NLP approach, the objective function and constraints are treated as nonlinear functions of the design variables. In the PLA procedure, which has been successfully used in a number of applications,[11,14,15] the nonlinear optimization progresses as a sequence of linear optimization subproblems (stages). A linear approximation based on the Taylor series expansion, $f = f_0 + \nabla f_0^T \Delta v$, is used to compute the objective function and the constraint functions within each subproblem (stage). Side constraints on v control the linearization error.

Efficiency of PLA stems from replacing the full analysis of the physical problem with approximate analysis by the linear extrapolation, which in structural applications requires a computer time at least an order of magnitude smaller than the full analysis time. Additional time savings result at each stage from the optimizer's capability to execute faster when the problem is defined as linear (mode 3 in section "Optimizer"). The number of consecutive linear stages required for overall convergence depends on the degree of the problem nonlinearity.

The computing capabilities defining columns of Table 1 are: computation of the behavior variables without gradients, inclusion of gradients computed by finite differences, and inclusion of gradients computed analytically.

Each of the five resulting options defined by intersections of Table 1 rows and columns requires its own organization of the procedure flow as illustrated by the flow charts of Figs. 1, 4 and 5. Option 1.1 is shown in Fig. 1. When executed with the analytical gradients indicated in parenthesis, it becomes option 1.3. Option 2.3 is depicted in Fig. 4. Insertion of a finite-difference gradient computation shown in Fig. 5 into the flow charts in Figs. 1 and 4 defines options 1.2 and 2.2, respectively. In the PLA options (2.2 and 2.3), the optimizer's main program is organized as shown in Fig. 2a to take advantage of the simplicity of the linear approximation. The main program also contains the termination criteria for PLA (outer loop in Fig. 4). The organization shown in Fig. 2c is used in the NLP options (1.1, 1.2, and 1.3).

For all options, SPAR has been separated into nonrepeatable and repeatable parts to minimize the amount of numerical work done in the optimization loop. This separation is achieved by writing a separate runstream for each of the two parts. For example, in option 1.1 for invariant overall geometry, the nonrepeatable part generates nodal coordinates, material properties, constraint data, and defines the loads, and the repeatable part generates solutions of the load-deflection equations.

Auxiliary Option for Fully Stressed Design

If strength constraints are present in the problem, then convergence of all the foregoing optimization procedures can be improved by using a limited number (say 3-5) of fully stressed design (FSD) iterations to generate initial cross-sectional dimensions of the structural members. Allowable stresses used in the FSD procedure can include material allowables (e.g., yield stress), and local buckling stresses that are functions of the cross-sectional dimensions as described in Ref. 16.

The FSD procedure is executed in the programing system using analyzer in a loop shown in the flow chart in Fig. 6. This flow chart may be inserted just ahead of the initial variable vector boxes in Figs. 1 and 4.

Application Procedure and Its Generality

PROSSS exists in two basic forms: skeleton form and specialized form. The skeleton form consists of the following: 1) the problem independent component programs such as SPAR, CONMIN, and the programs controlling execution of the loops in Figs. 5 (finite difference gradients) and 6 (FSD procedure); 2) the SPAR runstream files for analytical gradients; 3) the procedure files; and 4) the sets of JCL statements for each option.

To be used in a specific application, the skeleton form has to be turned into a specialized form. Problem dependent interface processors and the input data must be created and stored as files. The input data include also the SPAR runstreams. In addition, standard names in the JCL statement file corresponding to the option chosen must be replaced with names selected for the problem dependent files. Thus the specialized form of PROSSS ready to be executed consists of skeleton form permanent files, an additional set of files containing the problem dependent program and data, and a corresponding set of JCL statements.

Once the specialized form has been set up for a particular application, it can be protected from unauthorized alterations by using "software locks" (passwords) on all its files except the input data files. Several such specialized forms can be created from the common skeleton form for a variety of applications. Each such "frozen" specialized form can be used as a black box for a given class of problems that differ only by input data.

In industrial applications, preparation of the specialized forms would fall naturally into the domain of the staff specialist, while their application would be the task of the production oriented engineers. In research applications, the system's modularity permits its major components SPAR and CONMIN to be replaced with other equivalent programs, and execution flows different than those described in the foregoing also can be constructed. Thus, PROSSS can be used as a test bed for development of new optimization procedures. Although PROSSS has been developed for structural applications, it could be adapted to synthesis problems in other engineering disciplines by an appropriate replacement of the analyzer (e.g., replacing structural analysis by a computational aerodynamic program).

Application Examples

Examples presented in this section, when taken collectively, illustrate the variety of design variable formulations and types of constraints that can be handled by the system described. The design variables include cross-sectional dimensions with linking and in a reciprocal form (see Ref. 11), and coordinates of the nodal points. The constraints include limits on stresses, displacements, natural frequencies, and buckling loads. All options of the optimization procedure discussed in the preceding sections were exercised in the course of generating the examples.

Example 1: Rod-Panel Structure

The finite element model of a shear panel, stiffened in both directions with rods that have axial stiffness only is shown in Fig. 7. Initially, three design variables were chosen i.e., v_1 = cross-sectional area of transverse stringers, v_2 = cross-sectional area of longitudinal stringers, and v_3 = thickness of the panels. In this case and in all other examples, the cross-sectional dimension variables were handled by the optimizer in their reciprocal form to improve convergence.[11] The structure was optimized, as a nonlinear optimization problem, with no gradients required from the analyzer (option 1.1). The set of constraints consisted of stresses corresponding to two loading cases shown in Fig. 7. The starting values and the final optimized values of the design variables are given in Table 2. The history of iterations in the optimization process is shown graphically in Fig. 8. The objective function converges smoothly and there is no significant variation after three iterations as can be seen in the figure.

The rod-panel structure was next optimized considering ten design variables as shown in Fig. 9. A simple modification of only the O-A processor code implemented this change. The problem was treated by both NLP (options 1.1 and 1.2) and PLA (option 2.2). In this and all subsequent applications of the PLA, the objective function was allowed to change by 30% in one linear stage, and there were no move limits on the design variables. Only three linear stages were needed for satisfactory convergence using the PLA option. The difference in the final objective function values obtained by the two procedures was only 1.8%. The history of iterations of PLA (option 2.2) is shown in Fig. 9. Discontinuities of the objective functions shown in Fig. 9 at the beginning of each new stage reflects the results of new analyses performed at the outset of each stage. Figure 9 illustrates also, in addition to the linear stages, the iterations performed by the optimizer within a linear stage. This detail is shown only for the first linear stage. Increase of the number of design variables from 3 to 10 resulted in 21% lighter structure.

Subsequently, the set of constraints was augmented by inclusion of displacement limits and a limit on the first natural frequency. This change was accomplished by a modification to only the A-O processor code. Compared to the previous case, the objective function is greater, as expected, because of the additional active constraints. The

Table 2 Initial and final values of design variables for Example 1 (Fig. 8)

	Initial	Final
v_1	10.00 cm^2	8.46 cm^2
v_2	10.00 cm^2	4.30 cm^2
v_3	0.50 cm	0.17 cm

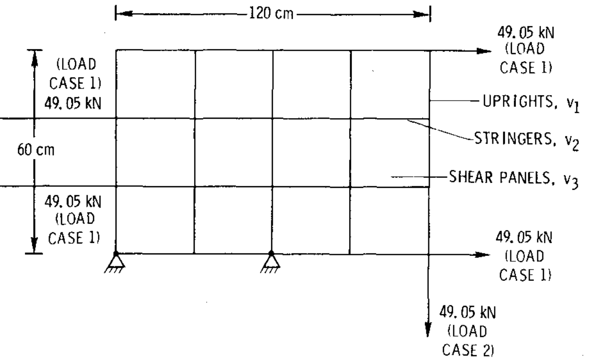


Fig. 7 Example 1, stiffened panel structure.

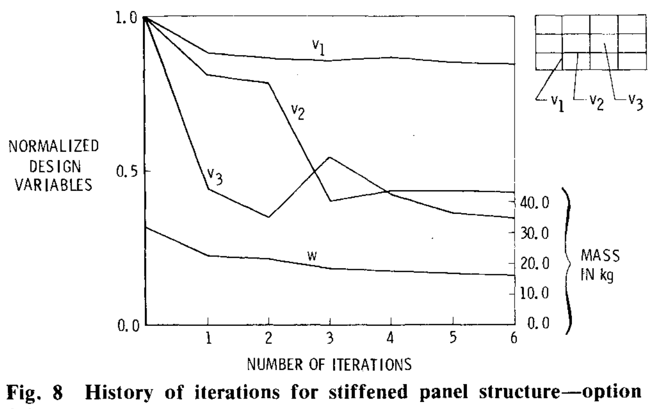


Fig. 8 History of iterations for stiffened panel structure—option 1.1.

Table 3 Initial and final values of design variables for example 1 (Fig. 8) for larger number of design variables

	Stress constraints only		Constraints on stress, displacement, and frequency	
	Initial	Final	Initial	Final
v_1	6.51 cm^2	4.17 cm^2	6.51 cm^2	7.09 cm^2
v_2	6.52 cm^2	4.19 cm^2	6.52 cm^2	4.91 cm^2
v_3	8.11 cm^2	8.06 cm^2	8.11 cm^2	23.16 cm^2
v_4	5.61 cm^2	4.16 cm^2	5.61 cm^2	6.15 cm^2
v_5	5.63 cm^2	4.15 cm^2	5.63 cm^2	4.42 cm^2
v_6	5.62 cm^2	4.19 cm^2	5.62 cm^2	4.74 cm^2
v_7	5.66 cm^2	4.17 cm^2	5.66 cm^2	9.35 cm^2
v_8	1.91 cm	1.69 cm	1.91 cm	10.00 cm
v_9	1.86 cm	1.70 cm	1.86 cm	10.00 cm
v_{10}	0.18 cm	0.14 cm	0.18 cm	0.24 cm

Table 4 Initial and final values of design variables for example 2, variant 1 (Fig. 11)

Design variable	Initial	Final
v_1[a]	0.749	0.397
v_2	0.400	0.100
v_3	0.625	0.376
v_4	0.435	0.115
v_5	0.667	0.256
v_6	0.500	0.122
v_7	0.125 cm	0.100 cm
v_8	0.125 cm	0.100 cm
v_9	0.125 cm	0.100 cm
v_{10}	0.125 cm	0.100 cm

[a] v_1-v_6 have been normalized with respect to initial cross-sectional area, 108.0 cm^2.

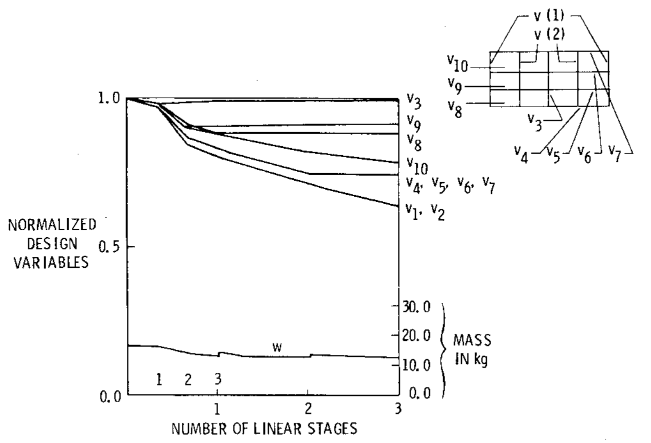
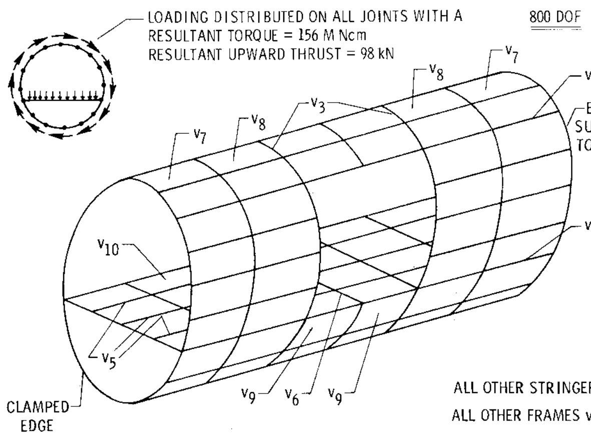


Fig. 9 Design variables and a history of iterations for stiffened panel structure—option 2.2.



LOADING DISTRIBUTED ON ALL JOINTS WITH A RESULTANT TORQUE = 156 M Ncm
RESULTANT UPWARD THRUST = 98 kN
800 DOF
v_8
v_7
v_1
EDGE SUBJECT TO LOAD
v_3
v_8
v_7
v_1
v_{10}
v_5
v_9
v_6
v_9
CLAMPED EDGE
ALL OTHER STRINGERS v_2
ALL OTHER FRAMES v_4

Fig. 10 Example 2, stiffened cylindrical shell, variant 1.

starting values of the design variables and the final optimum values are shown in Table 3 which indicates increase of the cross-sectional areas and thicknesses caused by the displacement and frequency constraints.

The effect of starting the optimization from a set of design variables corresponding to the result of three FSD iterations was tested for the rod-panel structure. The result was reduction in the number of optimizer iterations in option 1.1 from 14 when starting from minimum gages to 4, without affecting the final result.

Example 2: Stiffened Cylindrical Shell

Several variants of a circular cylindrical shell reinforced by frames and longerons were studied. Finite element model of the computationally largest variant (referred to as variant 1) is shown in Fig. 10. This variant is built up of membrane panels to represent skin, and of beam elements (axial, bending, and torsional stiffnesses) simulating transverse frames and longerons. Each frame and longeron may be regarded as a lumped representation[12] of several real frames and longerons. One end of the shell is clamped around the circumference, the other end is loaded by concentrated loads simulating distributed forces equivalent to a transverse force and torque. This variant has a large cut out and a floor, and represents a simplified model of a transport aircraft fuselage segment.

This structure was expected to constitute a demanding test case for the following two reasons. First, the model contains 798 degrees of freedom, so it is a computationally large problem as far as optimization by mathematical programing is concerned. Second, the overall bending state of stress in a shell of this type depends on the in-plane stiffness of the frames; therefore, the design variables that govern the member cross-sectional dimensions become strongly coupled[12] and the optimization process is more difficult to converge.

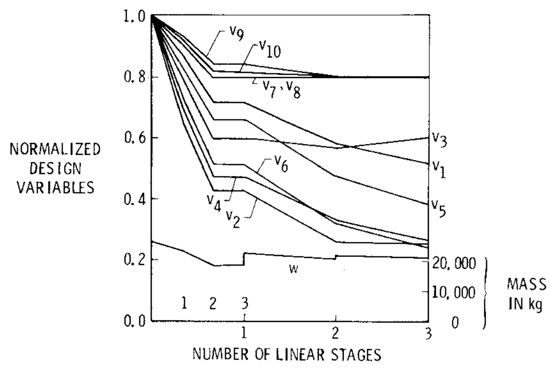


Fig. 11 History of iterations for example 2, variant 1 using option 2.2.

Variant 1 was optimized by PLA using finite-difference gradients (option 2.2) and the ten design variables shown in Fig. 10. As indicated in the figure, many structural parameters are linked to a single design variable. Variables v_1-v_6 govern the cross-sectional areas of the beam elements which have a channel cross section whose proportions remain constant as its area changes. Thus, the cross-sectional area becomes a single variable that governs all the beam's stiffness parameters. Variables v_7-v_{10} govern the membrane panel thicknesses. The history of iterations of optimization with stress constraints on beam elements and equivalent stress (Huber-von Mises stress) constraints on the panel elements is shown in Fig. 11. Convergence is quite good considering the problem size and use of the piecewise linear approximations. Table 4 shows the starting values of design variables and the final optimum values. As expected, the elements flanking the cut out have grown in the optimization process, as illustrated in Fig. 12.

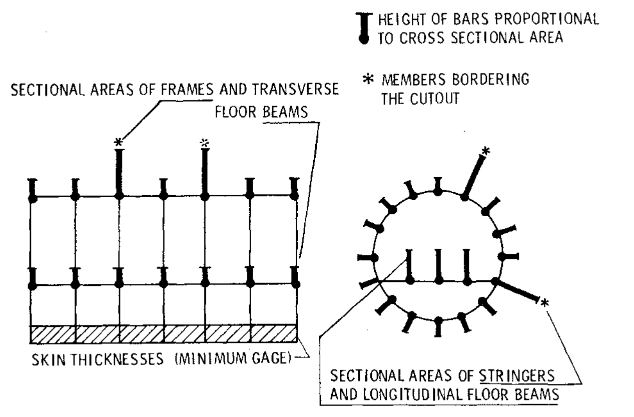

HEIGHT OF BARS PROPORTIONAL TO CROSS SECTIONAL AREA
* MEMBERS BORDERING THE CUTOUT
SECTIONAL AREAS OF FRAMES AND TRANSVERSE FLOOR BEAMS
SKIN THICKNESSES (MINIMUM GAGE)
SECTIONAL AREAS OF STRINGERS AND LONGITUDINAL FLOOR BEAMS

Fig. 12 Relative member sizes obtained by optimization for example 2, variant 1.

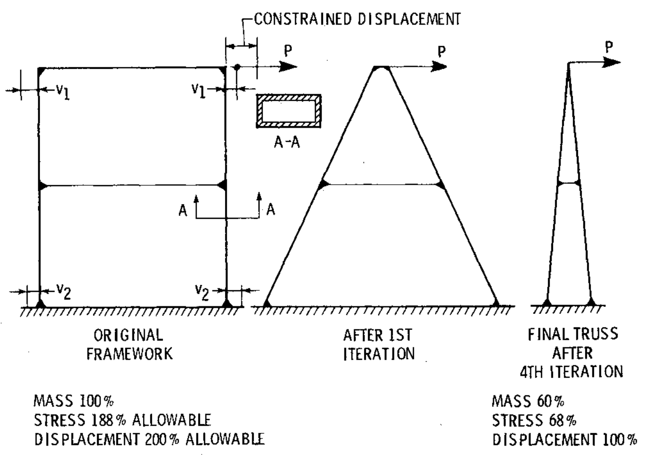


Fig. 13 Initial and optimized positions of transverse frames in example 2, variant 3.

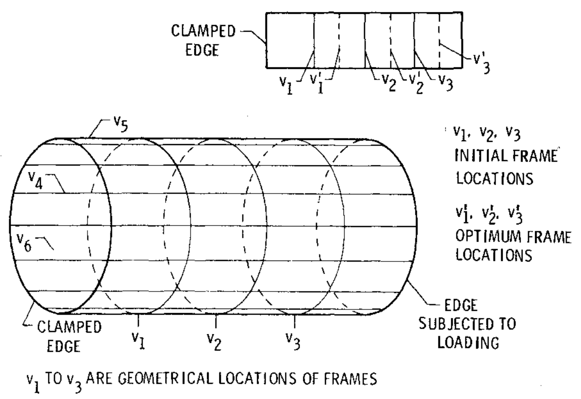

CLAMPED EDGE
v1 v'1 v2 v'2 v3 v'3
v5
v4
v6
v1, v2, v3 INITIAL FRAME LOCATIONS
v'1, v'2, v'3 OPTIMUM FRAME LOCATIONS
CLAMPED EDGE
v1 v2 v3
EDGE SUBJECTED TO LOADING
v1 TO v3 ARE GEOMETRICAL LOCATIONS OF FRAMES

Fig. 14 Transformation of a framework (example 3) to a truss by optimization with geometrical variables.

To further demonstrate the adaptability of the procedure, a simplified variant 2 of the shell structure was formed by eliminating the floor and two end bays and substituting rods for longerons. Initially, the structure was optimized with stress constraints only. Subsequently, the resultant structure was optimized with an additional overall shell buckling constraint which required 21% increase of the buckling load over and above the buckling load computed for the structure optimized with stress constraints only. Both optimizations were carried out by option 1.1. A comparison of these two results showed that the structural mass increased by 9.6% because of the additional buckling constraint. Additional optimization of this variant (with two loading cases) was carried out with stress constraints only using analytical gradients (option 2.3). Use of analytical gradients was found to reduce the execution time to approximately one sixth of that required for option 1.1. Three design variables, one for frames, one for longerons, and one for the skin were used in this case.

Locations of the node points in the finite element model were considered as design variables in a further simplified variant 3 of the shell structure. This variant has the cut out eliminated, is subject to only one loading case (transverse force), but has the longerons restored to the beam form. Previously defined cross-sectional variables were retained and the three geometrical variables governed locations of the three intermediate frames. Optimization using option 1.1 with stress and overall shell buckling constraints resulted in an expected translation of the frames toward the loaded and unsupported end where the additional support is most needed. The initial and optimized portions of frames are shown in Fig. 13.

Example 3: Portal Framework

A framework shown in Fig. 14 has been optimized with geometrical variables only, to demonstrate the optimizer's ability to transform structural shape. The variables defined in Fig. 14 are intended to allow the frame to transform itself into a truss. The constraints were imposed on stress and the horizontal displacement indicated in Fig. 14. Optimization, carried out by means of option 1.1 has indeed produced the expected transformation of shape to an almost triangular truss. A side constraint on length of the top horizontal member, necessary to preserve that member's nonzero length to avoid a matrix singularity, has kept the top of the frame from shrinking to a point.

Summary of Applications

Summarizing the application examples, the following observations are noted. Transforming the system from one optimization option to another was simple to accomplish by changing the sequence in which components of the system were called for execution. Adaptation from one variable and constraint combination to another was carried out by changes in the O-A and A-O processor codes. These adaptations as well as changes from one structure to another did not require any changes to the connecting network nor to the analyzer and optimizer.

It was a routine matter to monitor the status of the optimization process by means of displaying the intermediate data files. Stopping and restarting were facilitated by storing intermediate data.

Among the options tested, option 2.3, the PLA with analytical gradients, turned out to be by far the most cost effective, especially for larger problems such as variant 2 in example 2. Computer time required for that application was 40 s of CPU time on a CDC-Cyber 175 for option 2.3 compared to 250 s of CPU time consumed by the least efficient method, option 1.1.

Conclusions

A computer programing system is described which combines a sophisticated optimization program, a sophisticated structural analysis program, and user supplied and problem dependent interface programs, for solution of problems in structural optimization. Standard utility capabilities existing in modern computer operating systems are used to integrate these programs. This approach results in flexibility of the optimization procedure organization and versatility of the formulation of constraints and design variables. Variability of structural layout and overall shape geometry are included, and optimization applications range from meeting static strength and stiffness requirements, through accounting for buckling failure, to satisfying vibration limitations. Features of the programing system are illustrated by numerical examples.

Five options are described for organizing the optimization procedures. The options comprise various combinations of nonlinear mathematical programing and piecewise linear approximations with analytical and finite difference gradient techniques. Because of the system's inherent modularity, other software components could be substituted for the particular ones used herein to achieve a similar capability.

The system can be used in the following two basic ways. First, it may be used as a research test bed for development of optimization techniques and analysis oriented towards optimization applications. In this role, the system offers flexibility of execution and sequencing including restart and monitoring capabilities. Second, it may be used as an application tool that can be adapted by a specialist to a very wide scope of different types of problems and then used as a black box by a production oriented engineer.

Acknowledgments

The capability used herein to generate analytical derivatives of structural analysis results with respect to structural parameters has been implemented in SPAR by Gary L. Giles and James L. Rogers of NASA Langley Research Center. This contribution is gratefully acknowledged.

References

[1]"The NASTRAN Theoretical Manual (Level 16.0)," NASA SP-221 (03), March 1976.

[2]Schmit, L.A., Jr. and Miura, H., "A New Structural Analysis/Synthesis Capability-ACCESS 1," *AIAA Journal,* Vol. 14, May 1976, pp. 661-671.

[3]Haftka, R.T. and Prasad, B., "Programs for Analysis and Resizing of Complex Structures," *Trends in Computerized Structural Analysis and Synthesis,*" Pergamon Press, N.Y., 1978, pp. 323-330.

[4]Schrem, E., "From Program Systems to Programing Systems for Finite Element Analysis," presented at U.S.-Germany Symposium: Formulations and Computational Methods in Finite Element Analysis, Massachusetts Institute of Technology, Boston, Mass., Aug. 1976.

[5]Whetstone, W.D., "SPAR Structural Analysis System Reference Manual, System Level II," Vol. I, NASA CR-145098-1, Feb. 1977.

[6]Giles, G.L. and Haftka, R.T., "SPAR Data Handling Utilities," NASA TM-78701, Sept. 1978.

[7]Vanderplaats, G.N., "The Computer for Design and Optimization," *Computing in Applied Mechanics,* edited by R.F. Hartung, AMD - Vol. 18, American Society of Mechanical Engineers, 1976, pp. 25-48.

[8]"NOS Version 1 Reference Manual, NOS 1.2," Control Data Corporation; CDC No. 60445300, March 1978.

[9]Fox, R.L., *Optimization Methods for Engineering Design,* Addison-Wesley Pub. Co., Reading, Mass., 1971.

[10]Storaasli, O.O. and Sobieszczanski, J., "On the Accuracy of the Taylor Approximation for Structure Resizing," *AIAA Journal,* Vol. 12, Feb. 1974, pp. 231-233.

[11]Schmit, L.A. and Miura, H., "Approximation Concepts for Efficient Structural Synthesis," NASA CR-2552, March 1976.

[12]Sobieszczanski, J., "Sizing of Complex Structures by the Integration of Several Different Optimal Design Algorithms," *AGARD Lecture Series No. 70 on Structural Optimization,* AGARD-LS-70, Sept. 1974.

[13]Sobieszczanski, J., "Building a Computer Aided Design Capability Using a Standard Time Share Operating System," *Proceedings of the ASME Winter Annual Meeting, Integrated Design and Analysis of Aerospace Structures,* Houston, Texas, Nov.-Dec. 1975, pp. 93-112.

[14]Starnes, J.H. and Haftka, R.T., "Preliminary Design of Composite Wings for Buckling, Strength and Displacement Constraints," AIAA Paper 78-466, *Proceedings of the AIAA/ASME 19th Structures, Structural Dynamics and Materials Conference,* Bethesda, Md., April 1978.

[15]Anderson, M. S. and Stroud, W.J., "A General Panel Sizing Computer Code and Its Application to Composite Structural Panels," AIAA Paper 78-467, *Proceedings of the AIAA/ASME 19th Structures, Structural Dynamics and Materials Conference,* Bethesda, Md., April 1978.

[16]Giles, G.L., "Procedure for Automating Aircraft Wing Structural Design," *J. of Str. Div. ASCE,* Vol. 97, No. ST1, 1971, pp. 99-113.