

# **ADS - A FORTRAN PROGRAM FOR AUTOMATED DESIGN SYNTHESIS**

**VERSION 2.01**

**January, 1987**

by  
G. N. Vanderplaats

## **ACKNOWLEDGMENTS**

The original ADS program was developed under sponsorship of NASA. Additional enhancements contained herein were funded by the Optimization Software Users, supported by EDO, Inc., Santa Barbara, CA.

ENGINEERING DESIGN OPTIMIZATION, Inc.  
1275 Camino Rio Verde, Santa Barbara, CA 93111

# Contents

## Section 1

<b>Introduction</b> .....	<b>1</b>
1.1 Program Enhancements and Modifications Since Version 1.00 .....	2

## Section 2

<b>Program Options</b> .....	<b>4</b>
2.1 Strategy .....	4
2.2 Optimizer .....	5
2.3 One Dimensional Search .....	6
2.4 Allowable Combinations of Algorithms .....	6

## Section 3

<b>Program Flow Logic</b> .....	<b>9</b>
---------------------------------	----------

## Section 4

<b>Program Options</b> .....	<b>14</b>
4.1 Calling Statement .....	14
4.2 Definitions of Parameters in the ADS Calling Statement .....	14
4.3 Overriding ADS Default Parameters .....	18
4.4 User Supplied Gradients .....	23
4.5 Restarting ADS .....	24
4.6 Choosing an Algorithm .....	25

## Section 5

<b>Examples</b> .....	<b>27</b>
-----------------------	-----------

## Section 6

<b>Main Program for Simplified Usage of ADS</b> .....	<b>39</b>
---	-----------

## Section 7

<b>References</b> .....	<b>41</b>
-------------------------	-----------

## Appendix A

<b>Quick Reference to ADS Options</b> .....	<b>44</b>
---	-----------

<b>Appendix B</b>	
<b>Useful Information Stored in Arrays WK and IWK. . . . .</b>	<b>46</b>
<b>Appendix C</b>	
<b>Subroutines Needed for ISTRAT, IOPT and IONED . . . . .</b>	<b>48</b>
<b>Appendix D</b>	
<b>ADS Subroutines . . . . .</b>	<b>50</b>
<b>Appendix F</b>	
<b>In Case of Difficulty . . . . .</b>	<b>53</b>
<b>Appendix G</b>	
<b>ADS Internal Parameter Description . . . . .</b>	<b>55</b>

# Figures

Figure 1. Program Usage; All Default Parameters and Finite Difference Gradients . . . . .	10
Figure 2. Program Flow Logic; Override Default Parameters, Finite Difference Gradients . .	11
Figure 3. Program Flow Logic; Override Default Parameters and Provide Gradients . . . . .	13
Figure 4. Restarting ADS . . . . .	24
Figure 5. Three Bar Truss. . . . .	27
Figure 6. BETAMC Concept . . . . .	54
Figure 7. Relationship Between Constraint G and the Parameters CT and CTMIN . . . . .	55

# Tables

Table 1.	Strategy to be Used.....	4
Table 2.	Optimizer Options .....	5
Table 3.	One-Dimensional Search Options .....	6
Table 4.	Combinations of Algorithms Allowed .....	7
Table 5.	Parameters in the ADS Argument List.....	14
Table 6.	Real Parameters Stored in Array WK.....	18
Table 7.	Definitions of Real Parameters Stored in Array WK.....	20
Table 8.	Integer Parameters Stored in Array IWK.....	22
Table 9.	Definitions of Integer Parameters Contained in Array IWK .....	23
Table 10.	Sequence of Algorithms .....	26
Table 11.	Changes in Parameters .....	29
Table 12.	Real Parameters Stored in Arrays WK.....	46
Table 13.	Integer Parameters Stored in Array IWK.....	47

# Sample Code

Sample Code 1.	Example 1 - All Default Parameters .....	30
Sample Code 2.	Example 1 - Output .....	31
Sample Code 3.	Example 2 - Modify Default Parameters.....	32
Sample Code 4.	Example 2 - Output .....	33
Sample Code 5.	Example 3 - Gradients Supplied by the User.....	35
Sample Code 6.	Example 3 - Output .....	38
Sample Code 7.	Program for Simplified Usage of ADS .....	40

# Section 1

## Introduction

ADS is a general purpose numerical optimization program containing a wide variety of algorithms. The problem solved is:

Minimize  $F(X)$

Subject to:

$$G_j(X) \leq 0 \quad j=1, m$$

$$H_k(X) = 0 \quad k=1, L$$

$$X_{L_i} \leq X_i \leq X_{U_i} \quad i=1, n$$

The solution of this general problem is separated into three basic levels:

1. **Strategy** - For example, Sequential Unconstrained Minimization or Sequential Linear Programming. The purpose of a strategy is to convert the original constrained problem into a sequence of approximate problems using various techniques. A strategy is not used for unconstrained problems. In that case, the parameter, ISTRAT, is set to zero.
2. **Optimizer** - For example, Variable Metric methods for unconstrained minimization or the Method of Feasible Directions for constrained minimization. The optimizer performs the actual function minimization of either the original problem (if ISTRAT=0) or the approximate problem (if ISTRAT is greater than zero).
3. **One-Dimensional Search** - For example, Golden Section or Polynomial Interpolation. The one-dimensional search is called by the optimizer and, in some cases, the strategy. By choosing the Strategy, Optimizer and One Dimensional Search, the user is given considerable flexibility in creating an optimization program which works well for a given class of design problems.

Additionally, we may consider another component to be problem formulation. It is assumed that the engineer makes every effort to formulate the problem in a format amenable to efficient solution by numerical optimization. This aspect is perhaps the most important ingredient to the efficient use of the ADS program for solution of problems of practical significance.

This manual describes the use of the ADS program and the available program options. Section 1.1 describes the enhancements and modifications to the ADS program subsequent to Version 1.00 (ref. 1). Section 2 identifies the available optimization strategies, optimizers and one-dimensional search algorithms. Section 3 defines the program organization, and Section 4 gives user instructions. Section 5 presents several simple examples to aid the user in becoming familiar with the ADS program. Section 6 gives a simple main program that is useful for general design applications.

## 1.1 Program Enhancements and Modifications Since Version 1.00

Since the release of Version 1.00 in May of 1984, numerous modifications and enhancements have been made to the program. Many of these are minor and are transparent to the casual user. These include various formatting changes, internal logic enhancements to improve program flow, and a few actual bugs in the FORTRAN. Because of the robustness of the basic program, where bugs exist, their correction often is detected only in special test cases. Examples of this are correction of an error in using the absolute convergence criteria and correction of polynomial one-dimensional search when a constraint is being followed. Other enhancements include checking to insure the initial design does not violate any side constraints, and checking to be sure the combinations of strategy, optimizer and one-dimensional search are valid.

Enhancements to the program, beyond the original capability, include addition of equality constraint capability throughout the program and addition of a new strategy.

Equality constraints are now available in all options of the program, whereas in Version 1.00 they were only available when using penalty function strategies. Specifically, equality constraints have been added to optimizers 4 and 5. Here, two approaches were investigated. The first was to formally treat them in a mathematical sense. This requires considerable program logic and usually insures rather precise following of the constraints, but at some efficiency cost. The second approach, and that used here, was to treat equality constraints via a linear penalty function and an equivalent inequality constraint. The basic concept is to first change the sign on the constraint, if necessary, so that the scalar product of the gradient of the constraint with the gradient of the objective function is negative. The constraint is then converted to a non-positive inequality constraint and a linear penalty is added to the objective. The penalty, together with the conversion to an Inequality constraint have the effect of driving the original equality constraint to zero at the optimum, but without demanding precise accuracy, with its corresponding inefficiency. This is in keeping with the general philosophy of ADS of finding a near optimum design quickly.

A new strategy (ISTRAT=9), called Sequential Convex Programming, developed by Fleury and Briabant (ref. 2), has been added to ADS. The basic concept of this strategy is that a linear approximation to the objective and constraint functions is first created, just as in sequential linear programming. However, during the approximate optimization sub-problem, either direct or reciprocal variables are used, depending on the sign of the corresponding components of the gradients. This creates a conservative convex approximation to the optimization problem in comparison to a simple linearization. In reference 2, the method was applied to structural optimization problems in which all design variables were positive.

It was shown that move limits were not required during the sub-problem and that the method converged quickly to the optimum. When incorporating the algorithm into ADS, move limits were included, but they are less stringent than for sequential linear programming. This is based on the experience that the design space can become ill-conditioned in some general applications. Also, reciprocal variables are only used if the design variable is positive.

In earlier versions of ADS, when scaling was performed, the scaled constraints were printed. In this version, the constraints are unscaled prior to printing. In the one-dimensional search, the variables and function values are now unschooled prior to printing. Also, in all printing, a number, followed by a decimal are now used instead of the earlier Exx.xx format, to improve readability.

Perhaps the most significant program modification is in the scaling algorithm itself. The original scaling algorithm appeared quite sophisticated and, when it worked, it seemed very good. However, in those cases where it produced poor scaling, the results were often disastrous. Unfortunately, it was not possible to predict when it would or would not work. A particularly disturbing feature was that, sometimes the scaled constraints were satisfied within a small tolerance during optimization, but at the end when the unscaled values were printed, they were greatly violated. This provided the important information that the user had probably not carefully scaled the constraints to begin with. However, this is not obvious to most users and so it often led to practical difficulties when using ADS.

A completely new scaling algorithm has been used in Version 2.00 which is in many ways similar to the time honored normalization method used in the old CONMIN program. However, in addition to normalizing the design variables, the objective and constraints are also scaled. If the problem is naturally well scaled, the scale factor will be unity, but if the function and gradient information suggests a better scaling, this will be attempted. On test problems, this has been found to be a significant improvement over the previous scaling routine.

## Section 2

# Program Options

In this section, the options available in the ADS program are identified. At each of the three solution levels, several options are available to the user.

### 2.1 Strategy

Table 1 lists the strategies available. The parameter ISTRAT will be sent to the ADS program to identify the strategy the user wants. The ISTRAT=0 option would indicate that control should transfer directly to the optimizer. This would be the case, for example, when using the Method of Feasible Directions to solve constrained optimization problems because the optimizer works directly with the constrained problem. On the other hand, if the constrained optimization problem is to be solved by creating a sequence of unconstrained minimizations, with penalty functions to deal with constraints, one of the appropriate strategies would be used.

ISTRAT	Strategy to be Used
0	None - Go directly to the optimizer.
1	Sequential unconstrained minimization using the exterior penalty function method (refs. 3, 4).
2	Sequential unconstrained minimization using the linear extended interior penalty function method (refs. 5-7).
3	Sequential unconstrained minimization using the quadratic extended interior penalty function method (refs 8, 9).
4	Sequential unconstrained minimization using the cubic interior penalty function method (ref 10).
5	Augmented Language Multiplier method (refs. 11-15).
6	Sequential Linear Programming (refs. 16, 17).
7	Method of Centers (method of inscribed hyperspheres), (ref. 18).
8	Sequential Quadratic Programming (refs. 13, 19, 20).
9	Sequential Convex Programming (ref. 2).

Table 1. Strategy to be Used

## 2.2 Optimizer

Table 2 lists the optimizers available. IOPT is the parameter used to indicate the optimizer desired.

IOPT	Optimizer Options
0	None - Go directly to the one-dimensional search. This option should be used only for program development.
1	Fletcher-Reeves algorithm for unconstrained minimization (ref. 21).
2	Davidon-Fletcher-Powell (DFP) variable metric method for unconstructed minimization (refs. 22, 23).
3	Broydon-Fletcher-Goldfarb-Shanno (BFGS) variable metric method for unconstructed minimization (refs. 24-27).
4	Method of Feasible Directions (MFD) for constrained minimization (refs. 28, 29).
5	Modified Method of Feasible Directions for constrained minimization (ref. 30).

*Table 2. Optimizer Options*

In choosing the optimizer (as well as strategy and one-dimensional search) it is assumed that the user is knowledgeable enough to choose an algorithm consistent with the problem at hand. For example, a variable metric optimizer would not be used to solve constrained problems unless a strategy is used to create the equivalent unconstrained minimization task via some form of penalty function.

## 2.3 One Dimensional Search

Table 3 lists the one-dimensional search options available for unconstrained and constrained problems. Here IONED identifies the algorithm to be used.

IONED	One-Dimensional Search Options (refs. 3, 31, 32)
1	Find the minimum of an unconstrained function using the Golden Section Method.
2	Find the minimum of an unconstrained function using the Golden Section Method followed by polynomial interpolation.
3	Find the minimum of an unconstrained function by first finding bounds and then using polynomial interpolation.
4	Find the minimum of an unconstrained function by polynomial interpolation/extrapolation without first finding bounds on the solution.
5	Find the minimum of a constrained function using the Golden Section method.
6	Find the minimum of a constrained function using the Golden Section Method followed by polynomial interpolation.
7	Find the minimum of a constrained function by first finding bounds and then using polynomial interpolation.
8	Find the minimum of a constrained function by polynomial interpolation/extrapolation without first finding bounds on the solution.

Table 3. One-Dimensional Search Options

## 2.4 Allowable Combinations of Algorithms

Not all combinations of strategy, optimizer and one-dimensional search are meaningful. For example, constrained one-dimensional search is not meaningful when minimizing unconstrained functions.

Table 4 identifies the combinations of algorithms which are available in the ADS program. In this table, an X is used to denote an acceptable combination of strategy, optimizer and one-dimensional search. An example is shown by the heavy line on the table which indicates that constrained optimization is to be performed by the Augmented Lagrange Multiplier Method (ISTRAT=5), using the BFGS optimizer (IOPT=3) and polynomial interpolation with bounds for the one-dimensional search (IONED=3). From the table, it is clear that a large number of possible combinations of algorithms are available.

Strategy	Optimizer				
	1	2	3	4	5
0	X	X	X	X	X
1	X	X	X	0	0
2	X	X	X	0	0
3	X	X	X	0	0
4	X	X	X	0	0
5	X	X	X	X	X
6	0	0	0	X	X
7	0	0	0	X	X
8	0	0	0	X	X
9	0	0	0	X	X
<b>One-Dimensional Search</b>					
1	X	X	X	0	0
2	X	X	X	0	0
3	X	X	X	0	0
4	X	X	X	0	0
5	0	0	0	X	X
6	0	0	0	X	X
7	0	0	0	X	X
8	0	0	0	X	X

Table 4. Combinations of Algorithms Allowed

Appendix A contains an annotated version of Table 4 for convenient reference once the user is familiar with ADS.

To conserve computer storage, it may be desirable to use only those subroutines in the ADS system needed for a given combination of ISTRAT, IOPT and IONED. Appendix C provides the information necessary for this. Appendix D lists the ADS subroutines with a very brief description of each.

In writing a program to call ADS, the user should be aware that subroutine names should not be duplicated. This is seldom a problem with ADS because each routine begins with the letters ADS, followed by a three digit number. The exception is the ADS routine itself, which has no trailing numbers. Thus, the user need only be sure not to use subroutines with this numbering sequence.

## Section 3

# Program Flow Logic

ADS is called by a user-supplied calling program. ADS does not call any user-supplied subroutines. Instead, ADS returns control to the calling program when function or gradient information is needed. The required information is evaluated and ADS is called again. This provides considerable flexibility in program organization and restart capabilities.

The algorithms in ADS are called gradient based methods. That is they require the calculation of the gradients of the objective and constraint functions. In most applications, the user does not choose to calculate gradient information (often it is not possible because of the implicit nature of the problem). Therefore, the default case is that ADS will calculate all needed gradient information using a first forward finite difference scheme. The exception to this is that, if a variable is at its upper bound, a first backwards finite difference step is taken. This is because the bounds on the design variables are considered to be absolute and ADS will not consider a design outside the specified bounds, even during gradient computations. The exception to this is that, if the bounds are nearly equal, the resulting finite difference step may violate the lower bound.

Also, ADS has numerous internal parameters that control the optimization process. These all have default values that are used unless the user specifically changes them.

Thus, ADS can be used in four principal modes:

1. Default control parameters and finite difference gradients.
2. Override default parameters, use finite difference gradients.
3. Default control parameters and user-supplied gradients.
4. Override default parameters and user-supplied gradients.

The first mode is the simplest “black box” approach. In the second mode, the user overrides the default parameters to “fine tune” the program for efficiency. In modes 3 and 4, the user supplies all needed gradient information to the program.

Figure 1 is the program flow diagram for the simplest use of ADS. The user begins by defining the basic control parameters and arrays (to be described in Section 4). The gradient computation parameter, IGRAD, is set to zero to indicate that finite difference gradients will be used. The information parameter, INFO, is initialized to zero and ADS is called for optimization. Whenever the values of the objective, OBJ, and constraints, G(I), I=1, NCON, are required, control is returned to the user with INFO=1. The functions are then evaluated and ADS is called again. When INFO=0 is returned to the user, the optimization is complete.

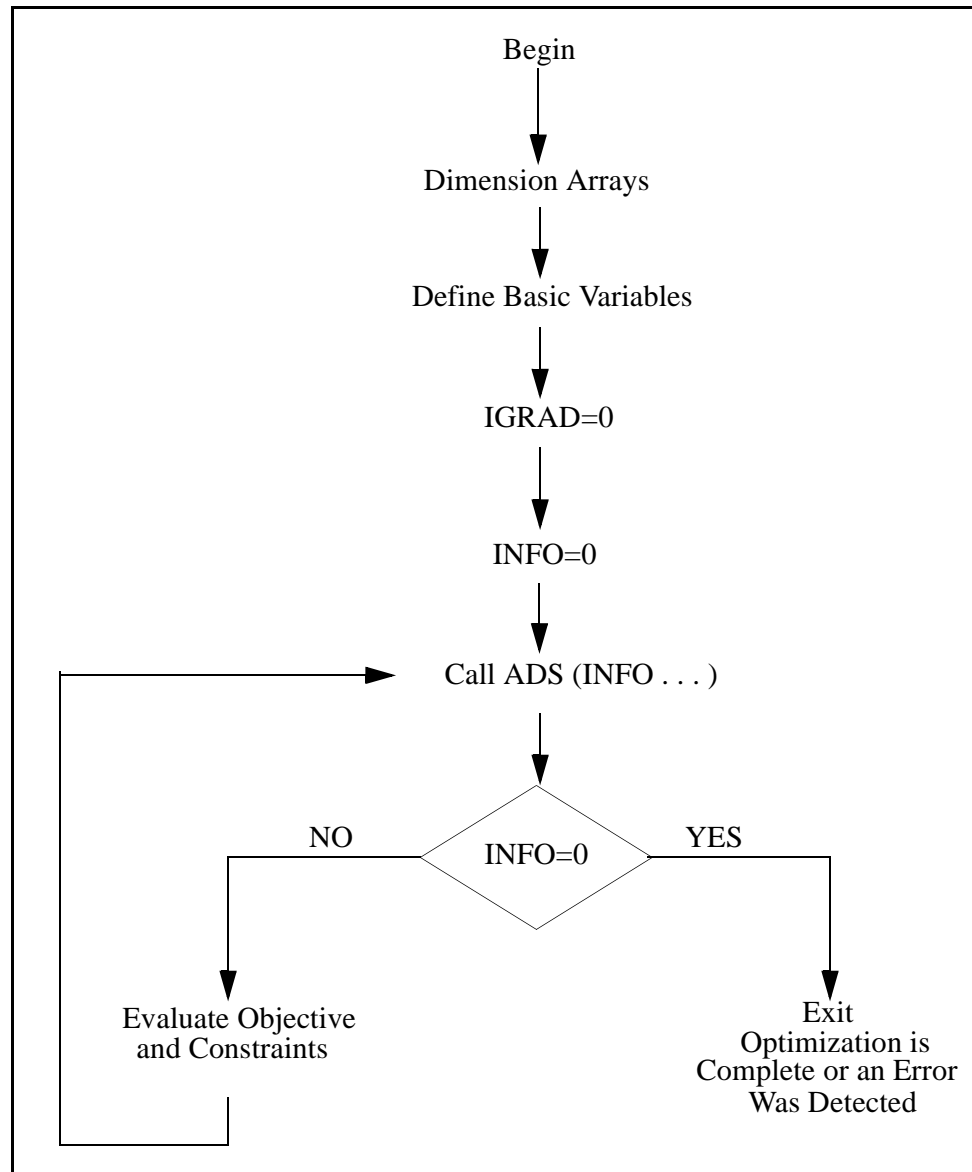


Figure 1. Program Usage; All Default Parameters and Finite Difference Gradients

Figure 2 is the program flow diagram for cases where the user wishes to override one or more internal parameters, such as convergence criteria or maximum number of iterations. After, initialization of basic parameters and arrays, the information parameter, INFO, is set to -2. ADS is called to initialize all internal parameters to their default values and allocate storage space for internal arrays. Control is returned to the user, at which point these parameters, for example convergence criteria, can be overridden. At this point, the information parameter, INFO, will have a value of -1 and must not be changed. ADS is called again and the optimization proceeds. Section 4.3 provides a list of internal parameters which may be modified, along with their locations in the work arrays **WK** and **IWK**. A more detailed explanation of these parameters is given in Appendix F.

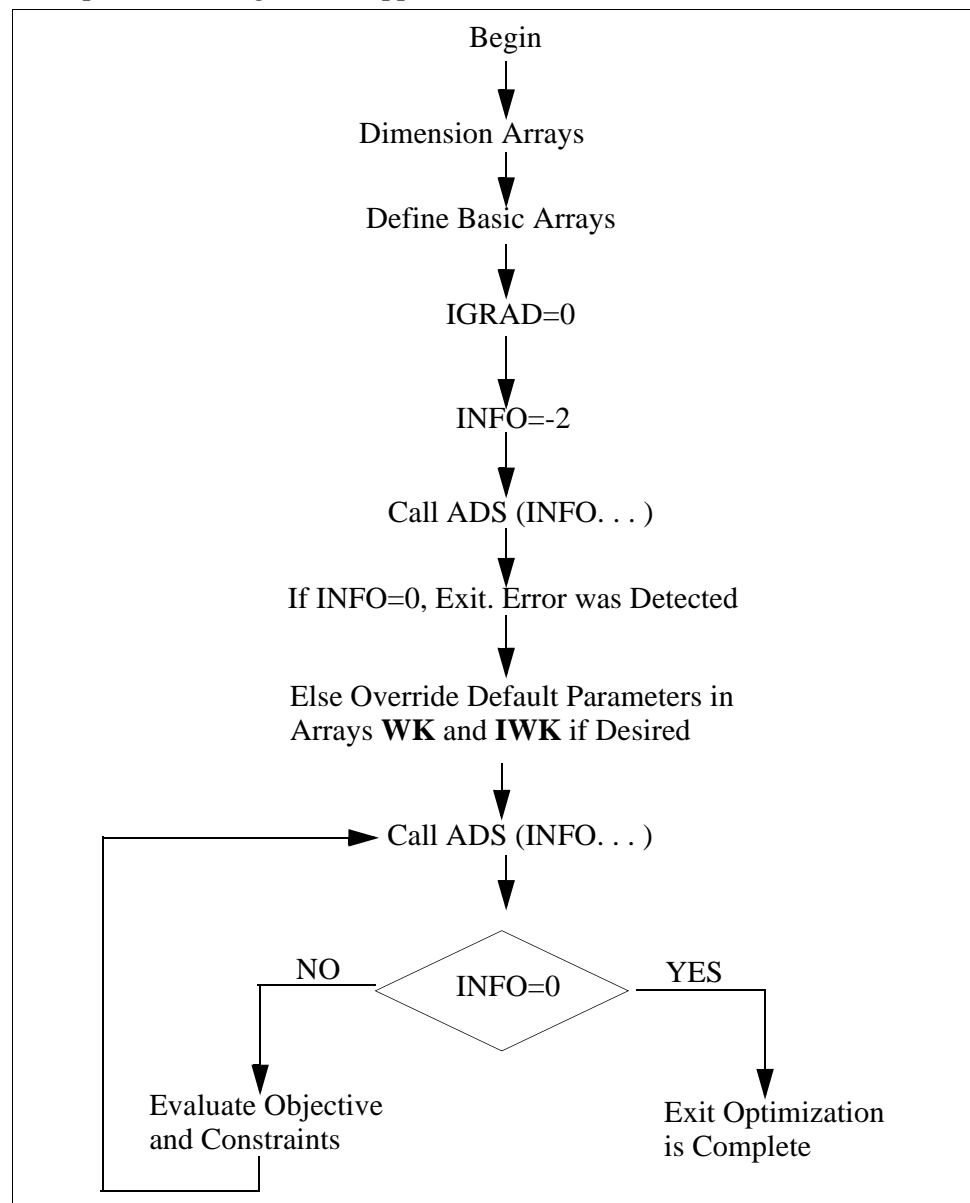


Figure 2. Program Flow Logic; Override Default Parameters, Finite Difference Gradients

Figure 3 is the flow diagram for the case where the user wishes to provide gradient information to ADS, rather than having ADS calculate this information using finite difference methods. In Figure 3, it is also assumed that the user will override some internal parameters, so the difference between Figures 2 and 3 is that IGRAD is now set to 1 and the user will now provide gradients during optimization. If the user does not wish to override any default parameters, INFO is initialized to zero and the first call to ADS is omitted (as in Figure 1). Now, when control is returned to the user, the information parameter will have a value of 1 or 2 (if INFO = 1, the optimization is complete, as before). If INFO = 1, the objective and constraint functions are evaluated and ADS is called again, just as in Figure 2. If INFO = 2, the gradient, DF, of the objective function is evaluated as well as the gradients of NGT constraints defined by vector IC.

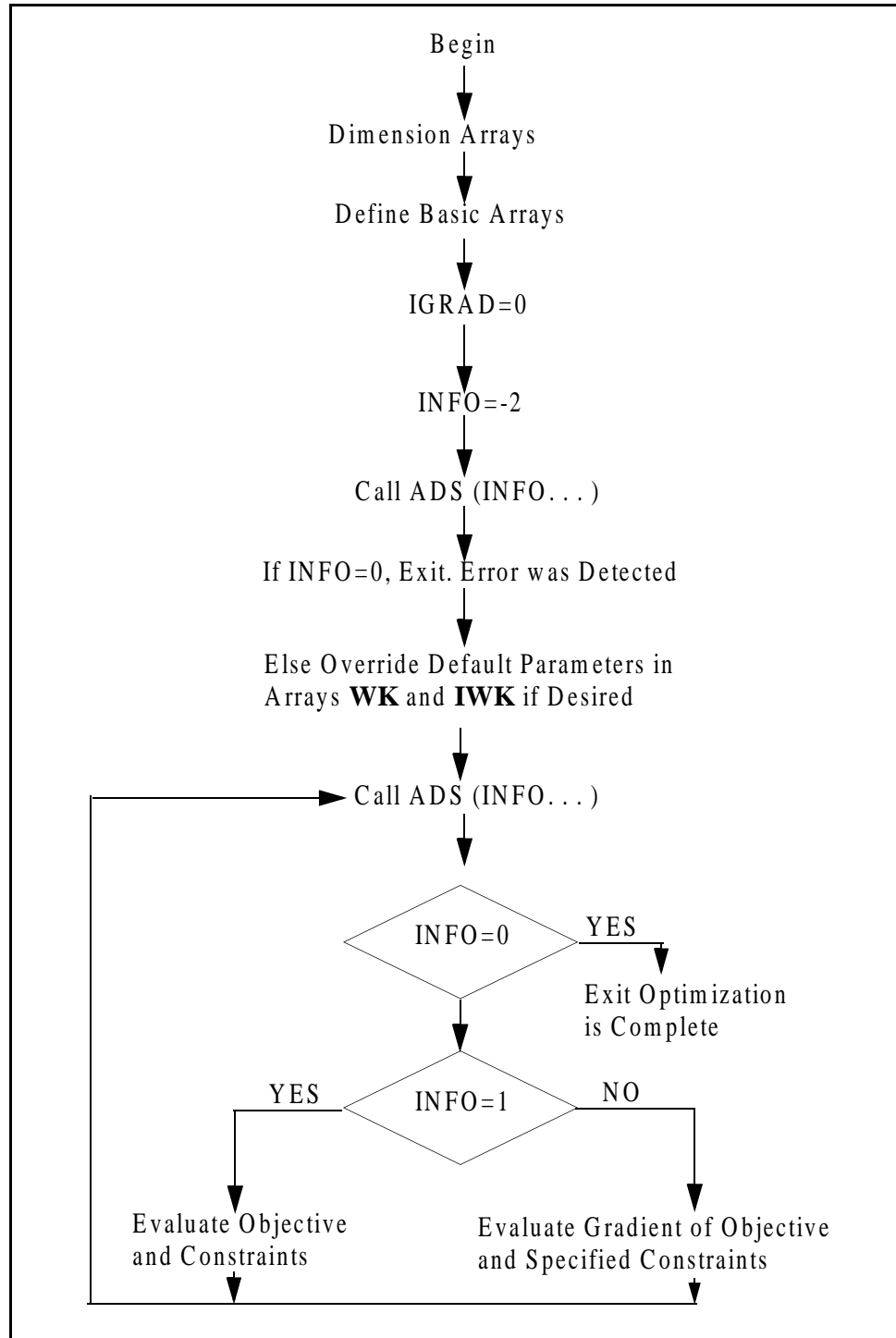


Figure 3. Program Flow Logic; Override Default Parameters and Provide Gradients

# Section 4

## Program Options

In this section the use of the ADS program is outlined. The FORTRAN Call statement given to ADS is given first, and then the parameters in the calling statement are defined. Section 4.3 identifies parameters that the user may wish to override to more effective use of ADS. Arrays are designated by bolding print.

### 4.1 Calling Statement

ADS is invoked by the following FORTRAN calling statement in the user's program:

```
CALL ADS (INFO, ISTRAT, IOPT, IONED, IPRINT, IGRAD, NDV, NCDN,  
X, VLB, VUB, OBJ, G, IDG, NGT, IC, DF, A, NRA, NCOLA, WK, NRWK,  
IWK, NRIWK)
```

### 4.2 Definitions of Parameters in the ADS Calling Statement

Table 5 lists the parameters in the calling statement to ADS. Where arrays are defined, the required dimension size is given as the array argument.

Parameter	Definition
INFO	Information parameter. On the first call to ADS, INFO=0 or -2. INFO=0 is used if the user does not wish to over-ride internal parameters and INFO = -2 is used if internal parameters are to be changed. When control returns from ADS to the calling program, INFO will have a value of 0, 1, or 2. If INFO=0, the optimization is complete. If INFO = 1, the user must evaluate the objective, OBJ, and constraint functions, G(I), I = 1, NCON, and call ADS again. If INFO = 2, the user must evaluate the gradient of the objective and the NGT constraints identified by the vector IC, and call ADS again. If the gradient calculation control, IGRAD = 0, INFO = 2 will never be returned from ADS, and all gradient information is calculated by finite difference within ADS.

Table 5. Parameters in the ADS Argument List (Page 1 of 4)

Parameter	Definition
ISTRAT	Optimization strategy to be used. Available options are identified in Tables 1 and 4.
IOPT	Optimizer to be used. Available options are identified in Tables 2 and 4.
IONED	One-dimensional search algorithm to be used. Available options are identified in Tables 3 and 4.
IPRINT	<p>A four-digit print control. IPRINT = IJKL where I, J, K and L have the following definitions:</p> <p>I ADS system print control.</p> <p>0 - No print.  1 - Print initial and final information.  2 - Same as 1 plus parameter values and storage needs.  3 - Same as 2 plus scaling information calculated by ADS.</p> <p>J Strategy print control.</p> <p>0 - No print.  1 - Print initial and final optimization information.  2 - Same as 1 plus OBJ and <b>X</b> at each iteration.  3 - Same as 2 plus <b>G</b> at each iteration.  4 - Same as 3 plus intermediate information.  5 - Same as 4 plus gradients of constraints.</p> <p>K Optimizer print control.</p> <p>0 - No print.  1 - Print initial and final optimization information.  2 - Same as 1 plus OBJ and <b>X</b> at each iteration.  3 - Same as 2 plus constraints at each iteration.  4 - Same as 3 plus intermediate optimization and one dimensional search information.  5 - Same as 4 plus gradients of constraints.</p> <p>L One-Dimensional search print control (debug only).</p> <p>0 - No print.  1 - One-dimensional search debug information.  2 - More of the same.</p> <p>Example: IPRINT=3120 corresponds to I=3, J=1, K=2 and L=0.  NOTE: IPRINT can be changed at any time control is returned to the user.</p>
IGRAD	Gradient calculation control. If IGRAD = 0 is input to ADS, all gradient computations are done within ADS by first forward finite difference. If IGRAD = 1, the user will supply gradient information as indicated by the value of INFO.

Table 5. Parameters in the ADS Argument List (Page 2 of 4)

Parameter	Definition
NDV	Number of design variables contained in vector <b>X</b> . NDV is the same as in the mathematical problem statement.
NCON	Number of constraint values contained in array <b>G</b> . NCON is the same $m + L$ in the mathematical problem statement given in Section 1.0. NCON = 0 is allowed.
<b>X</b> (NDV+1)	Vector containing the design variables. On the first call to ADS, this is the user's initial estimate to the design. On return from ADS, this is the design for which function or gradient values are required. On the final return from ADS (INFO=0 is returned), the vector <b>X</b> contains the optimum design.
<b>VLB</b> (NBV+1)	Array containing lower bounds on the design variables, <b>X</b> . If no lower bounds are imposed on one or more of the design variables, the corresponding component(s) of <b>VLB</b> must be set to a large negative number say -1.0E+15.
<b>VUB</b> (NDV+1)	Array containing upper bounds on the design variables, <b>X</b> . If no upper bounds are imposed on one or more of the design variables, the corresponding component(s) of <b>VUB</b> must be set to a large positive number, say 1.0E+15.
OBJ	Value of the objective function corresponding to the current values of the design variables contained in <b>X</b> . On the first call to ADS, OBJ need not be defined. ADS will return a value of INFO=1 to indicate that the user must evaluate OBJ and call ADS again. Subsequently, any time a value of INFO=1 is returned from ADS, the objective, OBJ, must be evaluated for the current design and ADS must be called again. OBJ has the same meaning as $F(X)$ in the mathematical problem statement given in Section 1.0.
<b>G</b> (NCON)	Array containing NCON constraint values corresponding to the current design contained in <b>X</b> . On the first call to ADS, the constraint values need not be defined. On return from ADS, if INFO=1, the constraints must be evaluated for the current <b>X</b> and ADS called again. If NCON=0, array <b>G</b> should be dimensioned to unity, but no constraint values need to be provided.
<b>IDG</b> (NCON)	Array containing identifiers indicating the type of the constraints contained in array <b>G</b> . <b>IDG</b> (I) = -2 for linear equality constraint. <b>IDG</b> (I) = -1 for nonlinear equality constraint. <b>IDG</b> (I) = 0 or 1 for nonlinear inequality constraint. <b>IDG</b> (I) = 2 for linear inequality constraint.

Table 5. Parameters in the ADS Argument List (Page 3 of 4)

Parameter	Definition
NGT	Number of constraints for which gradients must be supplied. NGT is defined by ADS as the minimum of NCOLA and NCON and is returned to the user.
IC(NGT)	Array identifying constraints for which gradients are required. <b>IC</b> is defined by ADS and returned to the user. If INFO=2 is returned to the user, the gradient of the objective and the NGT constraints must be evaluated and stored in arrays <b>DF</b> and <b>A</b> , respectively, and ADS must be called again.
DF(NDV + 1)	Array containing the gradient of the objective corresponding to the current <b>X</b> . Array <b>DF</b> must be defined by the user when INFO= 2 is returned from ADS. This will not occur if IGRAD=0, in which case array <b>DF</b> is evaluated by ADS.
A(NRA, NCOLA)	Array containing the gradients of the NGT constraints identified by array <b>IC</b> . That is, column J of array A contains the gradient of constraint number K, where $K = IC(J)$ . Array A must be defined by the user when INFO=2 is returned from ADS and when $NGT.GT.0$ . This will not occur if occur if IGRAD = 0, in which case, array A is evaluated by ADS. NRA is the dimensioned rows of array A. NCOLA is the dimensioned columns of array A.
NRA	Dimensioned rows of array A. NRA must be at least $NDV + 1$
NCOLA	NCOLA is the dimensioned columns of array <b>A</b> . NCOLA should be at least the minimum of NCON and $2*NDV$ . If enough storage is available, and the gradients are easily provided or are calculated by the finite difference, then $NCOLA = NCON + NDV$ is ideal.
WK(NRWK)	User provided work array for real variables. Array <b>WK</b> is used to store internal scalar variables and arrays used by ADS. <b>WK</b> must be dimensioned at least 100, but usually much larger. If the user has not provided enough storage, ADS will print the appropriate message and terminate the optimization.
NRWK	Dimensioned size of work array <b>WK</b> . A good estimate is $NRWK = 500 + 10 * (NDV + NCON) + (NCOLA + 3) + N*(N/2+1)$ , where $N = MAX(NDV, NCOLA)$
IWK (NRIWK)	User provided work array for integer variables. Array <b>IWK</b> is used to store internal scalar variables and arrays used by ADS. <b>IWK</b> must be dimensioned at least 200, but usually much larger. If the user has not provided enough storage, ADS will print the appropriate message and terminate the optimization.

Table 5. Parameters in the ADS Argument List (Page 4 of 4)

## 4.3 Overriding ADS Default Parameters

Various internal parameters are defined on the first call to ADS which work well for the “average” optimization task. However, it is often desirable to change these in order to gain maximum utility of the program. This mode of operation is shown in Figures 2 and 3. After the first call to ADS, various real and integer scalar parameters are stored in arrays **WK** and **IWK** respectively. Those which the user may wish to change are listed in Tables 6 through 9, together with their default values and definitions. If the user wishes to change any of these, the appropriate component of **WK** or **IWK** is simply re-defined after the first call to ADS. For example, if the relative convergence criterion, is to be changed to 0.002, this is done with the FORTRAN statement.

**WK**(12) = 0.002

because **WK**(12) contains the value of DELOBJ

Parameter	Location	Default	Modules Where Used		
			ISTRAT	IOPT	IONED
ALAMDC	1	0.0	5	-	-
BETAMC	2	0.0	7	-	-
CT(1)	3	-0.03	-	4, 5	-
CTL	4	-0.005	-	4, 5	-
CTLMN	5	0.001	-	4, 5	-
CTMIN	6	0.004	-	4, 5	-
DABALP(2)	7	0.0001	-	ALL	-
DABOBJ	8	ABS(FZ)/ 1000	ALL	-	-
DABOBM	9	ABS(FZ)/ 500	ALL	-	-
DABSTR	10	ABS(FZ)/ 1000	ALL	-	-
DELALP(3)	11	0.005	-	-	1, 2, 5, 6
DELOBJ	12	0.001	-	ALL	-
DELOBM	13	0.01	ALL	-	-
DELSTR	14	0.001	ALL	-	-

Table 6. Real Parameters Stored in Array **WK** (Page 1 of 2)

Parameter	Location	Default	Modules Where Used		
			ISTRAT	IOPT	IONED
DLOBJ1	15	0.1	-	ALL	-
DLOBJ2	16	1000.0	-	ALL	-
DX1	17	0.01	-	ALL	-
DX2	18	0.2	-	ALL	-
EPSPEN	19	-0.05	2, 3, 4	-	-
EXTRAP	20	5.0	-	-	ALL
FDCH	21	0.01	-	ALL	-
FDCHM	22	0.001	-	ALL	-
GMULTZ	23	10.0	8	-	-
PSAIZ	24	0.95	8	-	-
RMULT	25	5.0	1, 5	-	-
RMVLMZ	26	0.2	6, 7, 8, 9	-	-
RP	27	10.0	1, 5	-	-
RPMAX	28	1.0E+10	1, 5	-	-
RPMULT	29	0.2	1, 5	-	-
RPMIN	30	1.0E-10	2, 3, 4	-	-
RPPRIM	31	100	2, 3, 4	-	-
SCFO	32	1.0	ALL	ALL	ALL
SCLMIN	33	0.001	ALL	ALL	ALL
STOL	34	0.001	-	4, 5	-
THETAZ	35	0.1	-	4, 5	-
XMULT	36	2.618034	-	-	1,2,3 5,6,7
ZRO	37	0.00001	ALL	ALL	ALL
PMLT	38	10.0	6, 7, 8, 9	4, 5	-

Table 6. Real Parameters Stored in Array **WK** (Page 2 of 2)

- 1 IF IOPT = 4, CT = -0.1
- 2 If IONED = 3 or 8, DABALP = 0.001
- 3 If IONED = 3 or 8, DELALP = 0.05
- 4 If ISTRAT = 9, RMVLMZ = 0.4

FZ the objective function value for the initial design

Parameter	Definition
ALAMDZ	Initial estimate of the Lagrange Multipliers in the Augmented Lagrange Multiplier Method.
BETAMC	Additional steepest descent fraction in the method of centers. After moving to the center of the hypersphere, a steepest descent move is made equal to BETAMC times the radius of the hypersphere.
CT	Constraint tolerance in the Method of Feasible Directions or the Modified Method of Feasible Directions. A constraint is active if its numerical value is more positive than CT.
CTL	Same as CT, but for linear constraints.
CTLMIN	Same as CTMIN, but for linear constraints.
CTMIN	Minimum constraint tolerance for nonlinear constraints. If a constraint is more positive than CTMIN, it is considered to be violated.
DABALP	Absolute convergence criteria for the one-dimensional search when using the Golden Section method.
DABOBJ	Maximum absolute change in the objective between two consecutive iterations to indicate convergence in optimization.
DABOBM	Absolute convergence criterion for the optimization sub-problem when using sequential minimization techniques.
DABSTR	Same as DABOBJ, but used at the strategy level.
DELALP	Relative convergence criteria for the one-dimensional search when using the Golden Section method.
DELOBJ	Maximum relative change in the objective between two consecutive iterations to indicate convergence in optimization.
DELOBM	Relative convergence criterion for the optimization sub-problem when using sequential minimization techniques.
DELSTR	Same as DELOBJ, but used at the strategy level.
DLOBJ1	Relative change in the objective function attempted on the first optimization iteration. Used to estimate initial move in the one-dimensional search. Updated as the optimization progresses.
DLOBJ2	Absolute 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.

Table 7. Definitions of Real Parameters Stored in Array **WK** (Page 1 of 3)

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

Table 7. Definitions of Real Parameters Stored in Array **WK** (Page 2 of 3)

Parameter	Definition
SCLMIN	Minimum numerical value of any scale factor allowed.
STOL	Tolerance on the components of the calculated search direction to indicate that the Kuhn-Tucker conditions are satisfied.
THETAZ	Nominal value of the push-off factor in the Method of Feasible Directions.
XMULT	Multiplier on the move parameter, ALPHA, in the one-dimensional search to find bounds on the solution.
ZRO	Numerical estimate of zero on the computer. Usually the default value is adequate. If a computer with a short word length is used, ZRO = 1.0E-4 may be preferred.
PMLT	Penalty multiplier for equality constraints when IOPT = 4 or 5.

Table 7. Definitions of Real Parameters Stored in Array *WK* (Page 3 of 3)

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

Table 8. Integer Parameters Stored in Array *IWK*

Parameter	Definition
ICNDIR	Restart parameter for conjugate direction and variable metric methods. Unconstrained minimization is restarted with a steepest descent direction every ICNDIR iterations.
ISCAL	Scaling parameter. If ISCAL = 0, no scaling is done. If ISCAL=1, the design variables, objective and constraints are scaled automatically.
ITMAX	Maximum number of iterations allowed at the optimizer level.
ITRMOP	The number of consecutive iterations for which the absolute or relative convergence criteria must be met to indicate convergence at the optimizer level.
ITRMST	The number of consecutive iterations for which the absolute or relative convergence criteria must be met to indicate convergence at the strategy level.
JONED	The one-dimensional search parameter (IONED) to be used in the Sequential Quadratic Programming method at the strategy level
JTMAX	Maximum number of iterations allowed at the strategy level.

Table 9. Definitions of Integer Parameters Contained in Array **IWK**

## 4.4 User Supplied Gradients

If it is convenient to supply analytic gradients to ADS, rather than using internal finite difference calculations, considerable optimization efficiency is attainable. If the user wishes to supply gradients, the flow logic given in Figure 3 is used. In this case, the information parameter, INFO, will be returned to the user with a value of INFO=2 when gradients are needed. The user calculates the NGT gradients of the constraints identified by array IC and stores these in the first NGT columns of array **A**. That is column I of **A** contains the gradient of constraint J, where  $J=IC(I)$ .

## 4.5 Restarting ADS

When solving large and complex design problems, or when multi-level optimization is being performed, it is often desirable to terminate the optimization process and restart from that point at a later time. This is easily accomplished using the ADS program. Figure 4 provides the basic flowchart for this process. Whenever control is returned from ADS to the calling program, the entire contents of the parameter list are written to disk (or a file in a database management system). The program is then stopped at this point. Later, the program is restarted by reading the information back from disk and continuing from this point. If optimization is performed as a sub-problem within analysis, the information from the system level optimization is written to disk and the analysis is called. The analysis module can then call ADS to perform the sub-optimization task. Then, upon return from analysis, the system level information is read back from storage and the optimization proceeds as usual. From this, it is seen that considerable flexibility exists for multi-level and multi-discipline optimization with ADS, where the ADS program is used for multiple tasks within the overall design process.

The user may wish to stop the optimization at specific times during the process. The parameter IMAT is array **IWK** gives general information regarding the progress of the optimization. Appendix B provides details of this parameter as well as other parameters stored in **WK** and **IWK** which may be useful to the experienced user of ADS.

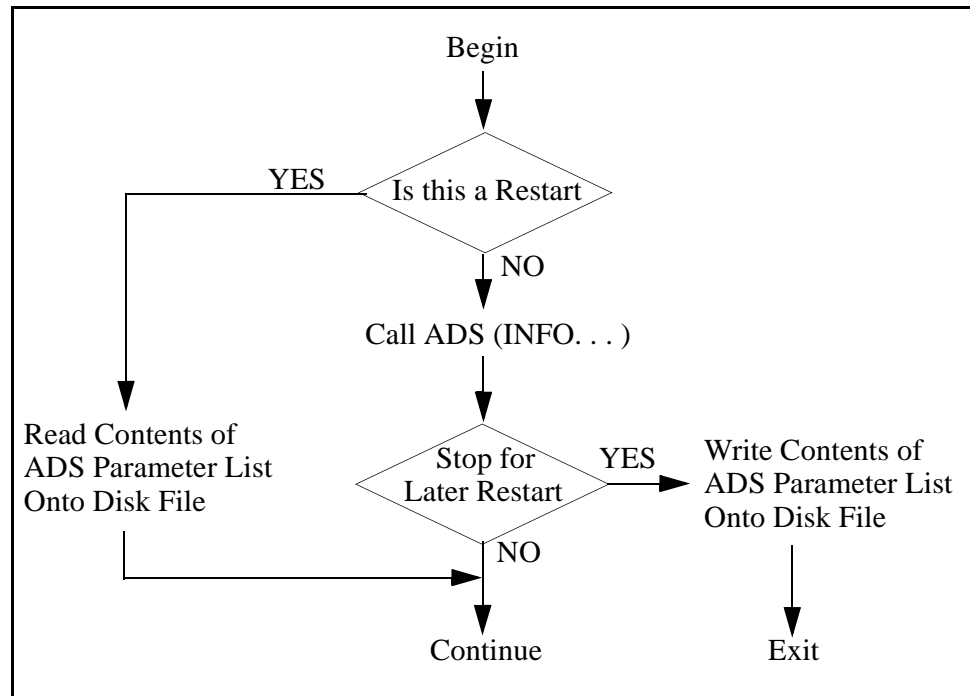


Figure 4. Restarting ADS

## 4.6 Choosing an Algorithm

One difficulty with a program such as ADS, which provides numerous options, is that of picking the best combination of algorithms to solve a given problem. While it is not possible to provide a concise set of rules, some general guidelines are offered here based on the author's experience. The user is strongly encouraged to try many different options in order to gain familiarity with ADS and to improve the probability that the best combination of algorithms is found for the particular class of problems being solved.

### Unconstrained Functions (NCON=0, Side Constraints OK)

ISTRAT = 0

Is computer storage very limited?

- **Yes - IOPT = 1. Are function evaluations expensive?**
  1. Yes - Is the objective known to be approximately quadratic?
    - A. Yes - IONED=4
    - B. No - IONED=3
  2. No - IONED=1 or 2
- **No - Is the analysis iterative?**
  1. Yes - IOPT=3. Are function evaluations expensive?
    - A. Yes - Is the objective known to be approximately quadratic?
      - a. Yes - IONED=4 No - IONED=3
      - b. No- IONED=3
    - B. No - IONED=1 or 2
  2. No - IOPT=2 or 3. Are function evaluations expensive?
    - A. Yes - Is the objective known to be approximately quadratic?
      - a. Yes - IONED=4
      - b. No - IONED=3
    - B. No - IONED=1 or 2

## Constrained Functions (NCON.GT.0)

Are relative minima known to exist?

- **Yes - ISTRAT = 1, IOPT = 3. Are functions expensive?**
  1. Yes - IONED = 3
  2. No - IONED = 1 or 2
- **No - Are the objective and/or constraints highly nonlinear?**
  1. Yes - Are function evaluations expensive?
    - A. Yes - ISTRAT = 0, IOPT = 4, IONED = 7
    - B. No - ISTRAT = 2, 3 or 5, IOPT = 2 or 3?, IONED = 1 or 2
  2. No - Is the design expected to be fully-constrained?  
(i.e., NDV active constraints at the optimum)
    - A. Yes - ISTRAT = 6, IOPT = 5, IONED = 6
    - B. No - Is the analysis iterative?
      - a. Yes - ISTRAT=0, IOPT=4, IONED=7 or  
ISTRAT=8, IOPT=5, IONED=7 or  
ISTRAT=9, IOPT=5, IONED=7
      - b. No - ISTRAT=0, IOPT=5, IONED=7 or  
ISTRAT=8, IOPT=5, IONED=7 or  
ISTRAT=9, IOPT=5, IONED=7

## General Applications

Often little is known about the nature of the problem being solved. Based on experience with a wide variety of problems, a very direct approach is given here for using ADS. The following table of parameters is offered as a sequence of algorithms. When using ADS the first few times, the user may prefer to run the cases given here, rather than using the decision approach given above. It is assumed here that a constrained optimization problem is being solved. If the problem is unconstrained, ISTRAT=0, IOPT=3 and IONED=2 or 3 is recommended.

ISTRAT	IOPT	IONED	IPRINT
8	5	7	2200
0	5	7	2020
0	4	7	2020
9	5	7	2200
6	5	6	2200
5	3	3	2200
2	3	3	2200
1	3	3	2200

Table 10. Sequence of Algorithms

# Section 5

## Examples

Consider the following two-variable optimization problem with two nonlinear constraints:

$$\text{Minimize } \text{OBJ} = 2 * \text{SQRT}(2) * X1 + X2$$

$$\text{Subject to: } G(1) = \frac{2 * X1 + \text{SQRT}(2) * X2}{2 * X1 * [X1 + \text{SQRT}(2) * X2]} - 1 \leq 0$$

$$G(2) = \frac{1}{2 * [X1 + \text{SQRT}(2) * X2]} - 1 \leq 0$$

$$0.01 \leq X_i \leq 1.0E + 20 \quad i = 1, 2$$

This is actually the optimization of the classical 3-bar truss shown in Figure 5 where, for simplicity, only the tensile stress constraints in members 1 and 2 under load P1 are included. The loads, P1 and P2, are applied separately and the material specific weight is 0.1 lb per cubic inch. The structure is required to be symmetric so X(1) corresponds to the cross-sectional area of members 1 and 3 and X(2) corresponds to the cross-sectional area of member 2.

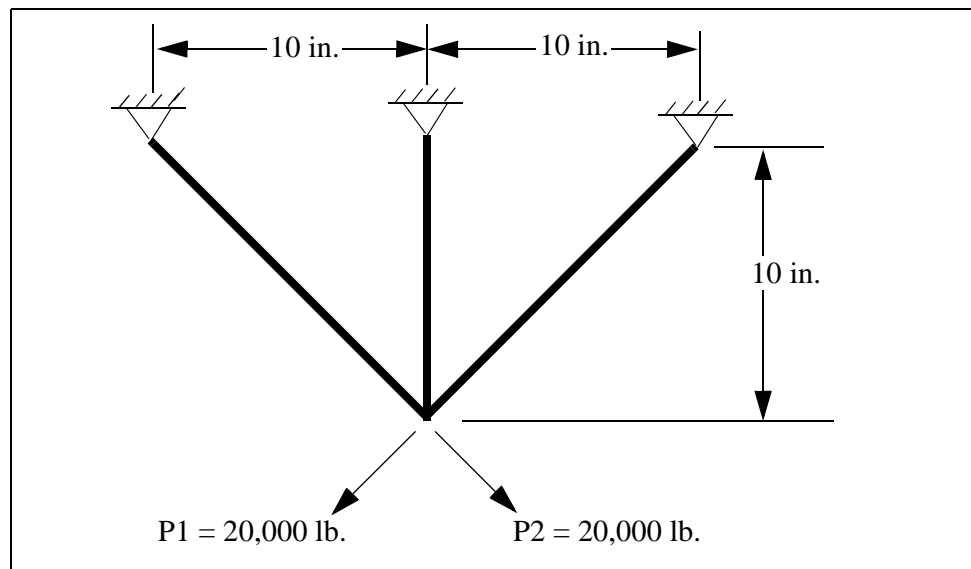


Figure 5. Three Bar Truss

In the source listings for the examples, the arrays are dimensioned sufficiently large to solve 10 design variable problems with 20 constraints. This allows the user to create larger problems using these programs as a basis. Note that the required array dimensions given in this manual are minimums. The arrays can be dimensioned larger than needed, just as is done here.

### Example 1 — All Default Parameters

Sample Code 1 gives the FORTRAN program to be used with ADS to solve this problem. Only one line of data is read by this program to define the values of ISTRAT, IOPT, IONED and IPRINT and the FORMAT is 4I5. When the optimization is complete, another case may be run by reading a new set of data. The program terminates when ISTRAT=-1 is read as data.

Sample Code 2 gives the results obtained with ISTRAT=0, IOPT=5, IONED=7 and IPRINT=1000. The reader is encouraged to experiment with this program using various combinations of the options from Table 4.

### Example 2 — Initial Parameters Are Modified

The 3-bar truss designed in Example 1 is now designed with the following changes in the internal parameters:

Parameter	New Value	Location in WK	Location in IWK
CT	-0.05	3	-
CTMIN	0.001	6	-
FDCH	0.001	21	-
ITRMOP	2	-	-

Table 11. Changes in Parameters

The FORTRAN program used here is shown in Sample Code 3 and the results are given in Sample Code 4.

### Example 3 — Gradients Supplied by the User

The 3-bar truss designed in Examples 1 and 2 is designed here with user supplied gradients. The parameters CT, CTMIN, THETAZ and ITRMOP are overridden as in Example 2. Also, now IPRINT=2020 to provide a more typical level of optimization output.

The FORTRAN program associated with this example is given in Sample Code 5. Sample Code 6 gives the results.

```
C   SIMPLIFIED USAGE OF ADS. THE THREE-BAR TRUSS.
C   REQUIRED ARRAYS.
      DIMENSION X(11), VLB(11), VUB(11), G(20), IDG(20), IC(20), DF(11),
      1 A(11, 20), WK(1000), IWK(500)
C   ARRAY DIMENSIONS.
      NRA=2
      NCOLA=2
      NRWK=1000
      NRIWK=500
C   PARAMETERS.
      IGRAD=0
      NDV=2
      NCON=2
C   INITIAL DESIGN.
      X(1)=1.
      X(2)=1.
C   BOUNDS.
      VLB(1)=.01
      VLB(2)=.01
      VUB(1)=1.0E+20
      VUB(2)=1.0E+20
C   IDENTIFY CONSTRAINTS IDG(1)=0
      IDG(2)=0
C   INPUT. READ(5,30) ISTRAT, IOPT, IONED, IPRINT
C   OPTIMIZE. INFO=0
30  CALL ADS (INFO,ISTRAT,IOPT IONED,IPRINT,IGRAD, NDV,NCON,X,VLB
      1 VUB, OBJ, G, IDG, NGT, IC, DF, A, NRA, NCOLA, WK,
      NRWK, IWK, NRIWK)
      IF (INFO.EQ.0) GO TO 20
C   EVALUATE OBJECTIVE AND CONSTRAINTS.
      OBJ=2.*SQRT(2.)*X(1)+X(2)
      G(1)=(2.*X(1)+SQRT(2.)*X(2))/(2.*X(1)*(X(1)+SQRT(2.)*X(2))-1.
      G(2)=.5/(X(1)+SQRT(2.)*X(2))-1.
C   GO CONTINUE WITH OPTIMIZATION.
      GO TO 10
20  CONTINUE
C   PRINT RESULTS.
      WRITE(6, 40) OBJ, X(1), X(2), G(1), G(2)... STOP
30  FORMAT (415)
40  FORMAT (/5X, 7H0PTIMUM, 5X, 5H0BJ =, E12.5//5X, 6HX(1) =, E12.5, 5X,
      1 6HX(2) =, E12.5/5X, 6HG(1) =, E12.5, 5X, 6HG(2) =, E12.5)
      END
```

*Sample Code 1. Example 1 - All Default Parameters*

**FORTRAN Program for Automated Design Synthesis**

(C) COPYRIGHT, EDO, INC., 1987  
ALL RIGHTS RESERVED, WORLDWIDE  
VERSION 3.00

**Control Parameters**

ISTRAT = 0    IOPT = 5    IONED = 7    IPRINT = 1000  
IGRAD = 0    NDV = 2    NCON = 2

**Optimization Results**

Objective Function Value 2.62899E+00

**Design Variables**

Variable	Lower Bound	Value	Upper Bound
1	1.00000E-02	7.82696E-01	1.00000E+20
2	1.00000E-02	4.15190E-01	1.00000E+20

**Design Constraints**

1.3.8170E-03 -6.3500E-01

**Function Evaluations** = 26

**Optimum**      OBJ = .26290E+01

**X**(1) = .78270E+00 **X**(2) = .41519E+00

**G**(1) = .38170E-02 **G**(2) = -.63500E+00

*Sample Code 2. Example 1 - Output*

```

C   USAGE OF ADS. OVERRIDING DEFAULT PARAMETERS. THE THREE
    BAR TRUSS.
    DIMENSION X(11), VLB(11), VUB(11), G(20), IDG(20), IC(20), DF(11),
    1 A(11, 20), WK(1000), IWK(500)
C   ARRAY DIMENSIONS.
    NRA=2
    NCOLA=2
    NRWK= 1000
    NRIWK=500
C   PARAMETERS. IGRAD=0
    NDV=2
    NCON=2
C   INITIAL DESIGN.
    X(1)=1.
    X(2)=1.
C   BOUNDS.
    VLB(1)=.01
    VLB(2)=.01
    VUB(1)=1.OE+20
    VUB(2)=1.OE+20
C   IDENTIFY CONSTRAINTS AS NONLINEAR, INEQUALITY.
    IDG(1)=0
    IDG(2)=0
C   INPUT.
    READ(5, 30) ISTRAT, IOPT, IONED, IPRINT
C   INITIALIZE INTERNAL PARAMETERS.
    INFO=-2
    CALL ADS (INFO,ISTRAT,IOPT,IONED,IPRINT,IGRAD, NDV,NCON,X, VLB
    1 VUB, OBJ, G, IDG, NGT, IC, DF, A, NRA, NCOLA, WK, NRWK,IWK,NRIWK)
C   OVERRIDE DEFAULT VALUES OF CT, CTMIN, THETAZ AND ITRMOP.
    WK(3)=-0.05
    WK(6)=0.001
    WK(21)=0.001
    IWK(4)=2
C   OPTIMIZE.
10  CALL ADS (INFO,ISTRAT,IOPT,IONED,IPRINT,IGRAD,NDV,NCON, X, VLB,
    1 VUB, OBJ, G, IDG, NGT, IC, DF, A, NRA, NCOLA, WK, NRWK,IWK,NRIWK)
    IF (INFO.EQ.0) GO TO 20
C   EVALUATE OBJECTIVE AND CONSTRAINTS.
    OBJ=2.*SQRT(2.)*X(1)+X(2)
    G(1)=(2.*X(1)+SQRT(2.)*X(2))/(2.*X(1)*(X(1)+SQRT(2.)*X(2)))-1.
    G(2)=.5/(X(1)+SQRT(2.)*X(2))-1.
C   GO CONTINUE WITH OPTIMIZATION.
    GO TO 10
20  CONTINUE
C   PRINT RESULTS.
    WRITE(6, 40) OBJ, X(1), X(2), G(1), G(2)
    STOP
30  FORMAT (4I5)
40  FORMAT (//5X, 7HOPTIMUM, 5X, 5HOBJ =, E12.5//5X, 6HX(1)=, E12.5, 5X,
    1 6HX(2) =, E12.5/5X, 6HG(1) =, E12.5, 5X, 6HG(2) =, E12.5)
END

```

*Sample Code 3. Example 2 - Modify Default Parameters*

## **FORTRAN Program for Automated Design Synthesis**

(C) COPYRIGHT, EDO, INC., 1987  
ALL RIGHTS RESERVED, WORLDWIDE  
VERSION 3.00

### **Control Parameters**

ISTRAT = 0    IOPT = 5    IONED = 7    IPRINT = 1000  
IGRAD = 0    NDV = 2    NCON = 2

### **Optimization Results**

Objective Function Value 2.63726E+00

### **Design Variables**

Variable	Lower Bound	Value	Upper Bound
1	1.00000E-02	7.86349E-01	1.00000E+20
2	1.00000E-02	4.13130E-01	1.00000E+20

### **Design Constraints**

1.6.5273E-04 -6.3520E-01

**Function Evaluations** = 29

**Optimum**      OBJ = .26373E+01

**X**(1) = .78635E+00 **X**(2) = .41313E+00

**G**(1) = .65273E-02 **G**(2) = -.63520E+00

*Sample Code 4. Example 2 - Output*

```

C   USAGE OF ADS. OVERRIDING DEFAULT PARAMETERS, AND
    PROVIDING GRADIENTS. THE THREE BAR TRUSS.
C   REQUIRED ARRAYS.
    DIMENSION X(11),VLB(11),VUB(11),G(20),IDG(20),IC(20), DF(11)
    1 A(11, 20), WK(1000), IWK(500)
    DIMENSION B(2, 2)
C   ARRAY DIMENSIONS.
    NRA=2
    NCOLA=2
    NRWK= 1000
    NRIWK=500
C   PARAMETERS.
    IGRAD=1
    NDV=2
    NCON=2
C   INITIAL DESIGN.
    X(1)=1.
    X(2)=1.
C   BOUNDS.
    VLB(1)=.01
    VLB(2)=.01
    VUB(1)=1.0E+20
    VUB(2)=1.0E+20
C   IDENTIFY CONSTRAINTS AS NONLINEAR, INEQUALITY
    IDG(1)=0
    IDG(2)=0
C   INPUT.
    READ(5, 70) ISTRAT, IOPT, IONED, IPRINT
C   INITIALIZE INTERNAL PARAMETERS.
    INFO=-2
    CALL ADS (INFO,ISTRAT,IOPT,IONED,IPRINT,IGRAD,NDV,NCON,X, VLB,
    1 VUB, OBJ, G,IDG,NGT,IC,DF,A,NRA,NCOLA,WK,NRWK,IWK, NRIWK)
C   OVERRIDE DEFAULT VALUES OF CT, CTMIN, THETAZ AND ITRMOP.
    WK(3)=-0.05
    WK(6)=0.001
    WK(21)=0.001
    IWK(4)=2
C   OPTIMIZE.
10  CALL ADS (INFO, ISTRAT, IOPT, IONED, IPRINT, IGRAD, NDV,
    NCON, X, VLB
    1 VUB, OBJ, G, IDG, NGT, IC, DF, A, NRA, NCOLA, WK, NRWK,
    IVK, NRIWK)
    IF (INFO.EQ.0) GO TO 60
    IF (INFO.GT.1) GO TO 20
C   EVALUATE OBJECTIVE AND CONSTRAINTS.
    OBJ=2.*SQRT(2.)*X(1)+X(2)
    G(1)=(2.*X(1)+SQRT(2.)*X(2))/(2.*X(1)*(X(1)+SQRT(2.)*X(2))-1.
    G(2)=.5/(X(1)+SQRT(2.)*X(2))-1.
C   GO CONTINUE WITH OPTIMIZATION.
    GO TO 10
20  CONTINUE
C   GRADIENT OF OBJ.
    DF(1)=2.*SQRT(2.)
    DF(2)=1.0
    IF (NGT.EQ.0) GO TO 10

```

```

C   CONSTRAINT GRADIENTS. USE ARRAY B FOR TEMPORARY STORAGE.
    D1=(X(1).SQRT(2.)*X(2))**2
C   G(1).
    B(1,1)=-2.*X(1)*X(1)+2.*SQRT(2.)*X(1)*X(2)+2.*X(2)*X(2)/
    1 (2.*X(1)*X(1)*D1)
    B(2,1)=-1./(SQRT(2.)*D1)
C   G(2)
    B(1,2)=-0.5/D1
    B(2,2)=SQRT(2.)*B(1,2)
C   STORE APPROPRIATE GRADIENTS IN ARRAY A.
    DO 30 J=1, NGT
    K=IC(J)
    A(1, J)=B(1,K)
30  A(2, J)=B(2, K)
    GO TO 10
60  CONTINUE
C   PRINT RESULTS.
    WRITE (6, 80) OBJ, X(1), X(2), G(1), G(2)
    STOP
70  FORMAT (4I5)
80  FORMAT (//5X, 7HOPTIMUM, 5X, 5HOBJ =, E12.5//5X, 6HX(1)=, E12.5, 5X,
    1 6HX(2) =, E12.5/5X, 6HG(1) =, E12.5,5X, 6HG(2) =, E12.5)
    END

```

*Sample Code 5. Example 3 - Gradients Supplied by the User*

## **FORTRAN Program for Automated Design Synthesis**

(C) COPYRIGHT, EDO, INC., 1987  
ALL RIGHTS RESERVED, WORLDWIDE  
VERSION 3.00

### **Control Parameters**

ISTRAT = 0      IOPT = 5      IONED = 7      IPRINT = 2020  
IGRAD = 1      NDV = 2      NCON = 2

### **Scalar Program Parameters**

#### **Real Parameters**

1. ALAMDZ =	.000000E+00	20. EXTRAP =	5.000000E+00
2. BETMAC =	.000000E+00	21. FDCH =	1.000000E-03
3. CT =	-5.000000E-02	22. FDCHM =	1.000000E-03
4. CTL =	-5.000000E-03	23. GMULTZ =	1.000000E+01
5. CTLMIN =	1.000000E-03	24. PSAIZ =	9.500000E-01
6. CTMIN =	1.000000E-03	25. RMULT =	5.000000E+00
7. DABALP =	1.000000E-04	26. RMVLMZ =	2.000000E-01
8. DABOBJ =	3.82843E-03	27. RP =	1.000000E+01
9. DABOBM =	7.65685E-03	28. RPMAX =	1.000000E+10
10. DABSTR =	3.82843E-03	29. RPMULT =	2.000000E-01
11. DELALP =	5.000000E-03	30. RPMIN =	1.000000E-10
12. DELOBJ =	1.000000E-03	31. RPPRIM =	1.000000E+02
13. DELOBM =	1.000000E-02	32. SCFO =	1.000000E+00
14. DELSTR =	1.000000E-03	33. SCLMIN =	1.000000E-03
15. DLOBJ1 =	1.000000E-01	34. STOL =	1.000000E-03
16. DLOBJ2 =	1.000000E+03	35. THETAZ =	1.000000E-01
17. DX1 =	1.000000E-02	36. XMULT =	2.61803E+00
18. DX =	2.200000E-01	37. ZR0 =	1.000000E-05
19. EPSPEN =	-5.000000E-02	38. PMLT =	1.000000E+01

#### **Integer Parameters**

1. ICNDIR =	3	4. ITRMOP =	2	6. JONED =	7
2. ISCAL =	1	5. ITRMST =	2	7. JTMAX =	20
3. ITMAX =	40				

#### **Array Storage Requirements**

	Dimensioned	Required
Array	Size	Size
<b>WK</b>	1000	199
<b>IWK</b>	500	184

\*\*\*\*\*  
*IOPT =5; Modified Method of Feasible Directions*  
\*\*\*\*\*

--Initial Design

OBJ = 3.82843E+00

Decision Variables (X-Vector)

1. 1.00000E+00 1.00000E+00

Lower Bounds on the Decision Variables (VLB-Vector)

1. 1.00000E-02 1.00000E-02

Upper Bounds on the Decision Variables (VUB-Vector)

1. 1.00000E+20 1.00000E+20

Constraint Values (G-Vector)

1. 2.92893E-01-7.92893E-01

--Iteration 1 OBJ = 2.79647EE+00

Decision Variables (X-Vector)

1. 6.75687E-01 8.85338E-01

-- Iteration 2 OBJ = 2.63882EE+00

Decision Variables (X-Vector)

1. 7.98080EE-01 3.81510E-01

-- Iteration 3 OBJ = 2.63724E+00

Decision Variables (X-Vector)

1. 7.86367E-01 4.13059E-011)

Final Optimization Results

Number of Iterations = 4

Objective = 2.63724E+00

**Decision Variables (X-Vector)**

1. 7.86367E-01 4.13059E-01

**Constraint Values (G-Vector)**

1. 6.60856E -6.35175E)

**Constraint Tolerance**

CT = -2.500000E-02 CTL = -2.50000E-03

There are 1 Active Constraints and 0 Violated Constraints

Constraint Numbers

1

There are 0 Active Side Constraints

**Termination Criteria**

KUHN-TUCKER PARAMETER, BETA = 9.65595E-06 is less than 1.00000E-03

**Optimization Results**

Objective Function Value 2.63724E+00

**Design Variables**

Variable	Lower Bound	Value	Upper Bound
1	1.00000E-02	7.863679E-01	1.00000E+20
2	1.00000E-02	4.13059E-01	1.00000E+20

**Design Constraints**

1.6.6086E-04 -6.3517E-01

**Function Evaluations** = 21

**Gradient Evaluations** = 4

**Optimum** OBJ = .26372E+01

X(1) = .78637E+00 X(2) = .41306E+00

G(1) = .66086E-03 G(2) = -.63517E+00

*Sample Code 6. Example 3 - Output*

## Section 6

# Main Program for Simplified Usage of ADS

Sample Code 7 is a general-purpose calling program for use with ADS. The arrays are dimensioned sufficient to solve problems of up to 20 design variables and 100 constraints. Arrays **IC** and **A** are dimensioned to allow for evaluation of 20 constraint gradients. Wherever a question mark (?) is given, it is understood that the user will supply the appropriate information. Note that the statement:  $X(I)=?$ ,  $I=1$ , **NDV** is not an implied FORTRAN DO LOOP, but simply denotes that the value of the **NDV** design variables must be defined here.

Subroutine **EVAL** is the user-supplied subroutine for evaluating functions and gradients (if user-supplied). The calling statement is:

```
CALL EVAL (INFO, NDV, NCON, OBJ,X, G, DF, NGT, IC, A, NRA)
```

The parameters **INFO**, **NDV**, **NCON**, **X**, **NGT**, **IC** and **NRA** are input to Subroutine **EVAL**, while **OBJ**, **C**, **DF** and **A** are output. Depending on the user needs, this may be simplified. For example, if **IGRAD=0** and **NDV** and **NCON** are not required by the analysis. the calling statement may be:

```
CALL EVAL (OBJ, X, G)
```

Also, a print control may be added so, after the optimization is complete, **EVAL** can be called again to print analysis information.

```

C   SIMPLIFIED USAGE OF THE ADS OPTIMIZATION PROGAM.
    DIMENSION X(21),VLB(21),VUB(21),G(100),IDG(100),IC(30),DF(21),
    * A(21, 30),WK(10000), IWK(2000)
    NRA=21
    NCOLA=30
    NRWK=10000
    NRIWK=2000
C   INITIALIZATION.
    IGRAD=?
    NDV=?
    NCON=?
    X(I)=?, I=1, NDV
    VLB(I)=?, I=1, NDV
    VUB(I)=?, I=1, NDV
    IDG(I)=?, I=1, NCON
    ISTRAT=?
    IOPT=?
    IONED=?
    IPRINT=?
    INFO=0
10  CALL ADS (INFO,ISTRAT,IOPT,IONED,IPRINT,IGRAD,NDV,NCON, X,
    *VLB,VUB,OBJ,G,IDG,NGT,IC,DF,A,NRA,NCOLA,WK,NRWK,IWK, NRIWK)
    CALL EVAL (INFO, NDV, NCON, OBJ, X, G, DF, NGT, IC, A, NRA)
    IF (INFO.GT.0) GO TO 10
C   OPTIMIZATION IS COMPLETE. PRINT RESULTS.
    STOP
    END

```

*Sample Code 7. Program for Simplified Usage of ADS*

# Section 7

## References

1. Vanderplaats, G. N., "ADS - A FORTRAN Program for Automated Design Synthesis," NASA CR 172460, Oct. 1984.
2. Fleury, C. and Braibant, V., "Structural Optimization; A New Dual Method Using Mixed Variables," *LTAS Report SA-115*, University of Liege, Leige, Belgium, March 1984.
3. Fox, R. L., "Optimization Methods for Engineering Design," Addison-Wesley, 1971.
4. Fiacco, A.V. and McCormick, G. P., *Nonlinear Programming: Sequential Unconstrained Minimization Techniques*, John Wiley and Sons, 1968.
5. Kavlie, D. and Moe, J., "Automated Design of Frame Structures," *ASCE Journal of Structural Div.*, Vol. ST1, Jan. 1971, pp. 33-62.
6. Cassis, J. H., "Optimum Design of Structures Subjected to Dynamic Loads," Ph.D. Thesis, University of California, Los Angeles, 1974.
7. Cassis, J. H. and Schmit, L. A., "On Implementation of the Extended Interior Penalty Function," *International Journal of Numerical Methods in Engineering*, Vol. 10, No. 1, 1976, pp. 3-23.
8. Haftka, R. T. and Starnes, J. H., Jr., "Applications of a Quadratic Extended Interior Penalty Function for Structural Optimization," *AIAA Journal*, Vol.14, June 1976, pp. 718-724.
9. Prasad, B. and Haftka, R. T., "Optimum Structural Design with Plate Finite Elements," *ASCE Journal of Structural Div.*, Vol.5Th, Nov. 1979, pp. 2367-2382.
10. Prasad, B., "A Class of Generalized Variable Penalty Methods for Nonlinear Programming," *Journal of Optimization Theory and Applications*, Vol.35, No.2, Oct. 1981, pp. 159-182.
11. Rockafellar, R. T., "The Multiplier Method of Hestenes and Powell Applied to Convex Programming," *Journal of Optimization Theory and Application*, Vol. 12, No. 6, 1973, pp. 555-562.
12. Pierre, D. A. and Lowe, M. J., "Mathematical Programming Via Augmented Lagrangians," *Applied Mathematics and Computation Series*, Addison-Wesley, 1975.

13. Powell, M.J.D., "Algorithms for Nonlinear Constraints that use Lagrangian Functions," *Mathematical Programming*, Vol. 14, No. 2, 1978, pp. 224-248.
14. Imai, K., "Configuration Optimization of Trusses by the Multiplier Method," Ph.D. Thesis, University of California, Los Angeles, 1978.
15. Imai, K. and Schrnit, L. A., "Configuration Optimization of Trusses," *Journal of the Structural Division, ASCE*, Vol. 107, No. ST5, May 1981, pp. 745-756.
16. Kelley, J. E., "The Cutting Plane Method for Solving Convex Programs," J. SIAM, 1960, pp. 703-712.
17. Moses, F., "Optimum Structural Design Using Linear Programming," *Proc. A.S.C.E.*, Vol. 90, ST6, 1964, pp. 89-104.
18. Baldur, R., "Structural Optimization by Inscribed Hyperspheres," *Journal of Engineering Mechanics, ASCE*, Vol. 98, No. EM3, June 1972, pp. 503-508.
19. Powell, M.J.D., "The Convergence of Variable Metric Methods for Nonlinearly Constrained Optimization Calculations," Proc. Nonlinear Programming Symposium 3, Madison, Wisconsin.
20. Powell, M.J.D., "A Fast Algorithm for Nonlinearly Constrained Optimization Calculations." *Report DAMTP77/NA2*, University of Cambridge, England.
21. Fletcher R. and Reeves, C. M., "Function Minimization by Conjugate gradients," *Computer Journal*, Vol. 7, No. 2, 1964, pp. 149-154.
22. Davidon, W. C., "Variable Metric Method for Minimization," *Argonne National Laboratory, ANL-5990 Rev.*, University of Chicago, 1959.
23. Fletcher, R. and Powell, M.J.D., "A Rapidly Convergent Method for Minimization," *Computer Journal*, Vol. 6, No. 2, 1963, pp. 163-168.
24. Broydon, C. G., "The Convergence of a Class of Double Rank Minimization Algorithms," Parts I and II, *J. Inst. Maths. Applns. Mzl.* 6, 1970, pp. 76-90 and 222-231.
25. Fletcher, R., "A New Approach to Variable Metric Algorithms," *Computer Journal*, Vol. 13, 1970, pp. 317-322.
26. Goldfarb, D., "A Family of Variable Metric Methods Derived by Variational Means," *Maths. Comput.*, Vol. 24, 1970, pp. 23-36.
27. Shanno, D. F., "Conditioning of Quasi-Newton Methods for Function Minimization," *Maths. Comput.*, Vol. 24, 1970, pp. 647-656.
28. Zoutendijk, M., *Methods of Feasible Directions*, Elsevier Publishing Co., Amsterdam, 1960.

29. Vanderplaats, C. N. and Moses, F., "Structural Optimization by Methods of Feasible Directions," *Journal of Computers and Structures*, Vol. 3, Pergamon Press, July 1973, pp. 739-755.
30. Vanderplaats, G. N., "An efficient Feasible Directions Algorithm for Design Synthesis," *AIAA J.*, Vol. 22, No. 11, Oct. 1984, pp. 1633-1640.
31. Himmelblau, D. M., *Applied Nonlinear Programming*, McGraw-Hill, 1972.
32. Vanderplaats, G. N., *Numerical Optimization Techniques for Engineering Design: With Applications*, McGraw-Hill, 1984.

# Appendix A

# Quick Reference to ADS Options

## **IOPT OPTIMIZER**

- 1 Fletcher-Reeves
- 2 Davidon-Fletcher-Powell (DEP)
- 3 Broydon-Fletcher-Goldfarb-Shanno (BFGS)
- 4 Method of Feasible Directions
- 5 Modified Method of Feasible Directions

<b><u>STRATEGY</u></b>	<b><u>ISTRAT</u></b>	<b><u>1</u></b>	<b><u>2</u></b>	<b><u>3</u></b>	<b><u>4</u></b>	<b><u>5</u></b>
None	0	X	X	X	X	X
SUMT, Exterior	1	X	X	X	0	0
SUMT, Linear Extended Interior	2	X	X	X	0	0
SUMT, Quadratic Extended Interior	3	X	X	X	0	0
SUMT, Cubic Extended Interior	4	X	X	X	0	0
Augmented Lagrange Multiplier Meth.	5	X	X	X	0	0
Sequential Linear Programming	6	0	0	0	X	X
Method of Centers	7	0	0	0	X	X
Sequential Quadratic Programming	8	0	0	0	X	X
Sequential Convex Programming	9	0	0	0	X	X

<b><u>ONE-DIMENSIONAL SEARCH</u></b>	<b><u>IONED</u></b>					
Golden Section Method	1	X	X	X	0	0
Golden Section + Polynomial	2	X	X	X	0	0
Polynomial Interpolation (bounded)	3	X	X	X	0	0
Polynomial Extrapolation	4	X	X	X	0	0
Golden Section Method	5	0	0	0	X	X
Golden Section + Polynomial	6	0	0	0	X	X
Polynomial Interpolation (bounded)	7	0	0	0	X	X
Polynomial Extrapolation	8	0	0	0	X	X

\*\*\*\*\*  
*An X denotes an allowed combination of algorithms.*  
 \*\*\*\*\*

# Appendix B

## Useful Information Stored in Arrays WK and IWK

Arrays **VK** and **IWK** contain information calculated by ADS which is sometimes monitoring the progress of the optimization. Tables B-1 and B-2 identify parameters which may be of interest to the user. Note that these parameters must not be changed by the user during the optimization process

Parameter	Location	DEFINITION
ALPHA	52	Move parameter in the one-dimensional search.
ALPHA3	53	ALPHA at the strategy level for ISTRAT=8.
PENALT	82	The value of the penalty in SUMT methods.
SLOPF	85	The slope of the OBJ versus ALPHA function in the one-dimensional search.

*Table 12. Real Parameters Stored in Arrays WK*

Parameter	Location	Definition
IDAB	23	Number of consecutive times the absolute convergence criterion has been satisfied at the optimization level.
IDAB3	24	Same as IDAB, but at the strategy level.
IDEL	25	Number of consecutive times the relative convergence criterion has been satisfied at the optimization level.
IDEL3	26	Same as IDEL, but at the strategy level.
IFCALL	28	The number of times the objective and constraint functions have been evaluated.
IGCALL	29	The number of times analytic gradients have been evaluated.
IMAT	34	Pointer telling the status of the optimization process. 0 - Optimization is complete. 1 - Initialization Is complete and control is being returned to the user to override default parameters. 2 - Initial function evaluation. 3 - Calculating analytic gradients. 4 - Calculating finite difference gradients. NXFD identifies the design variable being changed. 5 - One-dimensional search is being performed. See LGOTO.
ITER	45	Iteration number at the optimization level.
JTER	46	Iteration number at the strategy level.
LGOTO	54	Location in one-dimensional search. 1 - Finding bounds on the solution. 2 - Golden Section method. 3 - Polynomial interpolation after Golden Section 4 - Polynomial interpolation after getting bounds 5 - Polynomial interpolation/extrapolation.
NAC	58	Number of active constraints.
NACS	59	Number of active side constraints.
NVC	68	Number of violated constraints.
NXFD	69	Design variable being perturbed during finite difference gradients

Table 13. Integer Parameters Stored in Array **IWK**

# Appendix C

## Subroutines Needed for ISTRAT, IOPT and IONED

Depending on the combination of ISTRAT, IOPT and IONED, only a subset of subroutines contained in the ADS system are used. Therefore, if computer memory is limited, it may be desired only to load those routines which are actually used. This will result in “unsatisfied externals” at run time, but on most systems the program can be executed anyway since the unsatisfied external routines are not actually called. Below is a list of the routines needed for a given combination of algorithms. In some cases, slightly more routines are included than are absolutely necessary, but they are short and a more precise list would be unduly complicated.

Always Load the Following Subroutines:

ADS, ADS001, ADS002, ADS004, ADS005, ADS006, ADS007, ADS009, ADS010, ADS102, ADS103, ADS105, ADS112, ADS122, ADS201, AD5206, ADS211, AD5216, ADS236, AD5237, ADS401, AD5402, AD5403, AD5420, ADSS03, ADS504, AD5506, ADS510

### Strategy Level

Depending on the value of ISTRAT, the following subroutines are also required:

ISTRAT = 0, No strategy routines are added. Go to the optimizer level.

ISTRAT = 1, Add: ADS008, ADS301, ADS302, ADS508I

ISTRAT = 2, Add: ADS008, AD5302, ADS303, AD5308, ADS508

ISTRAT = 3, Add: ADS008, AD5302, ADS304, ADS308, ADS508

ISTRAT = 4, Add: ADS008, ADS302, AD5305, ADS308, ADS508

ISTRAT = 5 Add: ADS008, ADS302, ADS306, ADS307, ADS508

ISTRAT = 6, Add: ADS320, ADS321, AD5323, ADS333

ISTRAT = 7, Add: ADS323, ADS330, AD5331, ADS333

ISTRAT = 8, Add: ADS207, AD5217, ADS218, AD5221, ADS223, ADS310, AD5333,  
AD5371, ADS375, AD5376, AD5377, AD5378, ADS404, ADS507,  
ADSS08, ADS509

ISTRAT = 9, Add: ADS207, ADS17, ADS218, AD5221, AD5223, AD5325, ADS326,  
ADS509

### **Optimizer Level**

Depending on the value of IOPT, the following subroutines are also required:

IOPT = 0, No strategy routines are added. Go to the optimizer level.

IOPT = 1, Add: ADS204, ADS213, ADS214, ADS5509

IOPT = 2, Add: ADS213, ADS214, ADS235, ADS404, ADS503, ADS509

IOPT = 3, Add: ADS213, ADS214, ADS235, ADS404, ADS503, ADS509

IOPT = 4, Add: ADS201, ADS205, ADS207, ADS217, ADS218, ADS223,  
ADS507

IOPT = 5, Add: ADS201, ADS202, ADS203, ADS207, ADS209, ADS217, ADS218,  
ADS221, ADS235, ADS507

### **One-Dimensional Search Level**

Depending on the value of IONED, the following subroutines are also required:

IONED = 1-4, Add: ADS116, ADS117, ADS118, ADS121, ADS126, ADS127

IONED = 5-8, Add: ADS101, ADS104, ADS106, ADS108, ADS109, ADS110, ADS111,  
ADS115, ADS119, ADS123, ADS124, ADS125,  
ADS502

# Appendix D

## ADS Subroutines

The subroutines in the ADS system are listed here with a very brief description of each. Most subroutines are internally documented, and the user is referred to the program listing for more details

Generally, ADS001-ADS099 are control level routines, ADS201-ADS299 are optimum level routines and ADS301-ADS399 are strategy level routines. ADS401-ADS499 are print routines and ADS501-ADS599 are utility routines.

<b>Routine</b>	<b>Purpose</b>
ADS	- Main control routine for optimization.
ADS001	- Control one-dimensional search level.
ADS002	- Control optimizer level.
ADS003	- Control strategy level
ADS004	- Define work array storage allocations.
ADS005	- Initialize scalar parameters to their default values.
ADS006	- Initialize scale factors.
ADS007	- Calculate scale factors, scale, unscale.
ADS008	- Calculates gradients of pseudo-objective for ISTRAT=1-5.
ADS009	- Re-order <b>IC</b> and <b>A</b> arrays.
ADS010	- Calculates convergence criteria parameters.
ADS101	- Coefficients of linear polynomial.
ADS102	- Coefficients of quadratic polynomial.
ADS103	- Coefficients of cubic polynomial.
ADS104	- Zeros of polynomial to third-order.
ADS105	- Minimums of polynomial to third order.
ADS106	- Evaluate n-th order polynomial
ADS108	- Find minimum of a function by polynomial interpolation.
ADS109	- Find zeroes of a function by polynomial interpolation.
ADS110	- Evaluate slope of n-th order polynomial.
ADS111	- Polynomial interpolation for constraint boundaries
ADS112	- Find ALPMAX so NDV side constraints are encountered

- ADS115 - Control one-dimensional search for constrained functions.
- ADS116 - Control one-dimensional search for unconstrained functions.
- ADS117 - Polynomial interpolation of unconstrained function, within bounds.
- ADS118 - Polynomial interpolation of unconstrained function, no bounds given.
- ADS119 - polynomial interpolation of constrained function, no bounds given
- ADS121 - Find bounds on minimum of unconstrained function.
- ADS122 - Initial interior points for Golden Section method.
- ADS123 - Constrained one-dimensional search by Golden Section method.
- ADS124 - Update bounds and get new interior point in Golden Section method, constrained.
- ADS125 - Find bounds on minimum of constrained function.
- ADS126 - Unconstrained one-dimensional search by Golden Section method.
- ADS127 - Update bounds and get new interior point by Golden Section method, unconstrained.
- ADS201 - Identify NGT most critical constraints.
- ADS202 - Invert matrix B and store back in B.
- ADS203 - Delta-X back to boundary in Modified Method of Feasible Directions.
- ADS204 - Fletcher-Reeves unconstrained minimization.
- ADS205 - Method of Feasible Directions.
- ADS206 -  $X = X_{old} + \text{ALPHA} * S$ , subject to side constraints.
- ADS207 - Maximum component (magnitude) of each column of **A**.
- ADS209 - Calculate  $B = A\text{-Transpose} \text{ times } A$ .
- ADS211 - Update convergence parameters IDEL and IDAB.
- ADS213 - Calculate initial ALPHA for one-dimensional search based on objective value.
- ADS214 - Calculate initial ALPHA for one-dimensional search based on X-values.
- ADS216 - Finite difference gradients of objective and constraints.
- ADS217 - Solve direction-finding task for Methods of Feasible Directions.
- ADS218 - Solve special LP sub-problem from ADS217.
- ADS221 - Push-off factors for Methods of Feasible Directions.
- ADS223 - Identify active side constraints.
- ADS231 - Modified Methods of Feasible Directions.
- ADS235 - Variable Metric Methods, IOPT=2, 3
- ADS236 - Search direction for Variable Metric Methods

ADS237 - Penalty for equality constraints, IOPT=4, 5  
 ADS301 - Exterior Penalty Function Method, ISTRAT=1.  
 ADS302 - Calculates penalty for penalty function methods, ISTRAT=1-5.  
 ADS303 - Linear Extended Penalty Function Method, ISTRAT=2.  
 ADS304 - Quadratic Extended Penalty Function Method, ISTRAT=3.  
 ADS305 - Cubic Extended Penalty Function Method, ISTRAT=4.  
 ADS306 - Augmented Lagrange Multiplier Method, ISTRAT=5.  
 ADS307 - Update Lagrange Multipliers, ISTRAT=5.  
 ADS308 - Calculate penalty parameters, ISTRAT=5.  
 ADS310 - Sequential Quadratic Programming, ISTRAT=8.  
 ADS320 - Sequential Linear Programming, ISTRAT=6.  
 ADS321 - Control solution of LP sub-problem, ISTRAT=6.  
 ADS323 - Update move limits, ISTRAT=6, 7.  
 AD5325 - Sequential Convex Programming, ISTRAT=9.  
 AD5326 - Solve convex sub-problem, ISTRAT=9.  
 ADS330 - Method of Centers, ISTRAT=7.  
 ADS331 - Control solution of LP sub-problem, ISTRAT=7.  
 ADS333 - Calculate maximum constraint value.  
 ADS371 - Control solution of QP sub-problem, ISTRAT=8.  
 ADS375 - Temporary objective, ISTRAT=8.  
 ADS376 - Gradient of pseudo-objective for one-dimensional search, ISTRAT=8.  
 ADS377 - Change in objective gradients, ISTRAT--8.  
 ADS378 - Update Hessian matrix, ISTRAT=8.  
 ADS401 - Print arrays.  
 AD5402 - Print array title and array. Calls ADS401.  
 ADS403 - Print scalar control parameters.  
 AD5404 - Print Hessian matrix.  
 AD5420 - Print final optimization results.  
 ADS501 - Evaluate scalar product of two vectors.  
 ADS502 - Find maximum component of vector.  
 AD5503 - Equate two vectors.  
 ADS 504 - Matrix-vector product.  
 ADS506 - Initialize symmetric matrix to the identity matrix.  
 AD5507 - Normalize vector by dividing by maximum component.  
 ADS508 - Calculate gradient of pseudo-objective for ISTRAT=1-5.  
 Called by ADS008.  
 ADS509 - Identify active side constraints.  
 ADS510 - Scale, unscale the X-vector.

# Appendix F

## In Case of Difficulty

The ADS program is relatively robust, and there should seldom be a case where no progress is made during the optimization. Also, numerous internal checks are made to avoid exponent overflows, divide by zero, and similar run time errors.

Usually, when something seems wrong, it can be traced to the basic setup of the optimization problem or (more often) simple programming errors. Thus, while it is difficult to project all possible errors, some are common enough to be able to offer the following short list of items to check.

1. Check all array dimension statements. Be sure the values of NRA, NCOLA, NRWK and NRIWK are correct. ADS is written in single precision and double precision should really not be needed at the optimization level. If the analysis program is written in double precision, be sure to transfer all variables and arrays to equivalent single dimension values before calling ADS and transfer them back on return. This effects very few parameters and arrays, but is sometimes overlooked, and is very difficult to debug.
2. Check the parameter list for calling ADS. Be sure all parameters are present and in the proper order. A common error is to create a program with an editor that allows 80 column lines, while using a compiler that ignores all characters after column 72.
3. Turn off the automatic scaling and try again. Use the override capability and set **IWK(2)=0**. Sometimes the scaling actually makes the conditioning of the problem worse, although in Version 2.00 it is greatly improved from before. If the difficulties still exist, leave the scaling turned off during further testing.
4. Set the print control, IPRINT to 3500 if ISTRAT is greater than zero or 3050 if ISTRAT is equal zero. This will cause gradient information to be printed during the optimization. If the gradient of the objective or any constraint function has all zeroes, this parameter is not a function of the design variables. While it is theoretically possible to have a zero gradient, it is extremely rare on a digital computer. Check problem formulation.

5. Check the order of magnitude of the components of the gradients. A well conditioned problem will have roughly the same order of magnitude values (within a factor of 100). If one term is several orders of magnitude greater than the others it may help to scale this design variable by dividing by a number of that order of magnitude. A common error in problem formulation is to have a function, say  $Q$  that must be less than  $QQ$ , where  $QQ$  is on the order of 10,000. In creating the constraint (which is required to be less than or equal to zero) we may write  $G(I) = Q - QQ$ . This will make the constraint very difficult to deal with by ADS, because  $Q$  must equal about 9,999.95 before the constraint is considered active. Therefore, it is important to scale the constraint as  $G(I) = Q/QQ - 1$ . Now a constraint value of -0.01 will identify the constraint as being within one percent of being critical.
6. As a last resort, turn on the one-dimensional search print control (set the last digit of IPRINT to 5). Plot the objective and constraint functions versus the move parameter, ALPHA. If one or more are extremely nonlinear, reformulation of the problem by dividing that function by a large number is indicated. Another possibility here is that the finite difference gradient parameters, FDCH and FDCHM are either too large or too small. If the analysis is iterative, it often helps to try  $FDCH = 0.02$  or larger and  $FDCHM = 0.01$  or larger. This will mask the inaccuracies in the analysis. On the other hand, if the analysis is calculated very precisely as functions of the design variables, an order of magnitude smaller than the default value is indicated.
7. If the last resort fails, call EDO. We will do our best to help.

# Appendix G

## ADS Internal Parameter Description

In this appendix a description of the ADS internal parameters is given.

The parameters are listed in alphabetical order. If it is unlikely that the parameter should be changed from its default value, this is stated. Reference 32 describes most of the algorithms contained in ADS, and may be referred to for a more detailed description of how a parameter is used in a given algorithm.

### **Real Parameters Contained in WK**

**ALAMDZ** - Used for ISTRAT = 5. Initial values for the Lagrange Multipliers for the Augmented Lagrange Multiplier method. Applies to all constraints. Usually the default values are adequate.

**BETAMC** - Used only with ISTRAT = 7. This provides an additional steepest descent move in the method of centers beyond the move to the center of the hypersphere. The basic method moves to the center of the hypersphere bounded by the linear approximation to the current objective function and constraints. In problems that are not too highly nonlinear, this may be quite conservative. Using BETAMC, it is possible to move an additional amount in a steepest descent direction in order to speed convergence. If the move is too far (it violates constraints) it will be automatically reduced, but at the expense of a function evaluation. The general concept shown in Figure F.1, where the initial move is to the center of the circle (a hypersphere in two-dimensional space is a circle). The additional move is in the direction negative to the gradient of the objective function. BETAMC = 1.0 will move to the edge of the circle. A larger value is usually too optimistic, while a value of 0.5 will often be about right.

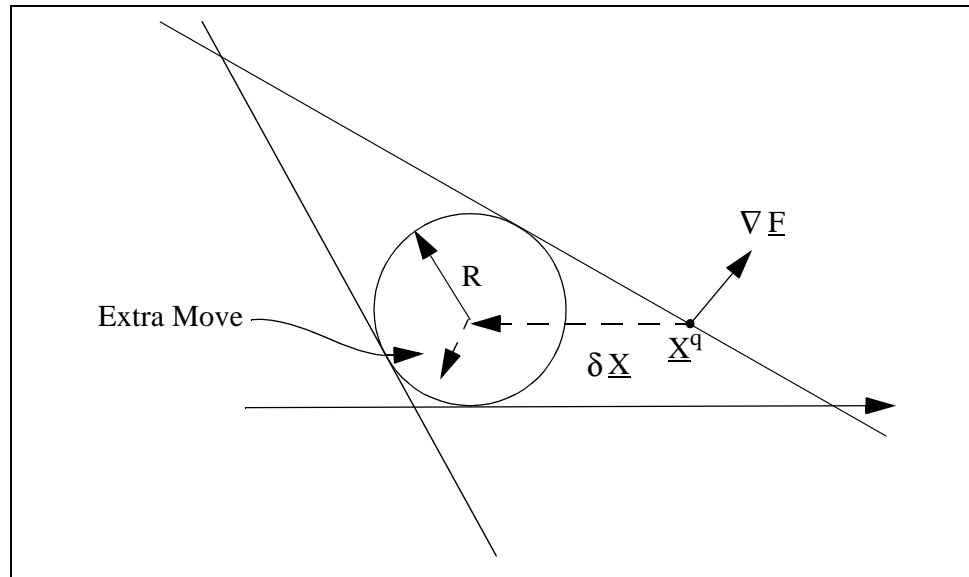


Figure 6. BETAMC Concept

CT - Used with IOPT=4 or 5. Also used with ISTRAT = 1-9 to a slightly lesser degree. Constraint tolerance for nonlinear inequality constraints. This parameter defines when a constraint is considered active, and is perhaps the most important parameter for nonlinear constrained optimization.

One of the key issues in constrained optimization is determining when a constraint is numerically “critical.” If a constraint,  $G(I)$  is numerically greater than CT, it is considered critical for purposes of finding a new search direction or deciding if the optimum has been found. This is also why constraint should be normalized to order of magnitude of unity. Thus if  $G(I)$  is numerically greater than CT (say -0.03) then it is assumed to be within 3 percent of being critical. Numerically, this is considered to be an “active” constraint.

For highly nonlinear constraints, it is often helpful to make CT more negative, say -0.10. By this method, the constraint is “trapped” sooner and the optimization process will direct the design away from this constraint. On the other hand, if the constraint is nearly linear, it may help to make CT closer to zero, say -0.01. Then, when interpolating for  $G(I)=0$  a more precise value of  $G(I)$  is obtained. In either case, the value of CT is progressively reduced during optimization to a value of -CTMIN, which is the value at which a constraint becomes strongly critical. In fact, if  $G(I)$  exceeds CTMIN (a positive) number the constraint is considered to be violated. See the definition of CTMIN.

For IOPT = 4 and 5, if a constraint repeatedly becomes active on one iteration and inactive on the next, CT should be increased in magnitude (try CT = -0.1 or -0.15), or the offending constraint should be divided by a factor of ten to reduce its sensitivity.

Note that in ADS, equality constraints are converted to equivalent inequality constraints. Therefore, the definitions of CT, CTMIN, CTL and CTLMIN apply equality constraints as well.

CTMIN - Used with IOPT=4 or 5. Also used with ISTRAT = 1-9 to a slightly lesser degree. Constraint tolerance defining when nonlinear inequality constraints are violated. CTMIN is a positive number. A constraint is considered inactive if its value is more negative than CT and active if its value is between CT and CTMIN. If the constraint value is more positive than CTMIN, it is considered violated. This is perhaps the second most important parameter for nonlinear constrained optimization.

Since, mathematically, an inequality is violated any time it's value is greater than zero, there may be a temptation to set CTMIN = 0. However, this should not be done because the optimization algorithms interpolate on zero and some numerical bandwidth should be provided to allow for numerical accuracies. The default value allows for about a half of a percent constraint violation for normalized constraints.

The geometric relationship between a constraint,  $G$ , and the parameters CT and CTMIN is shown in Figure F.2.

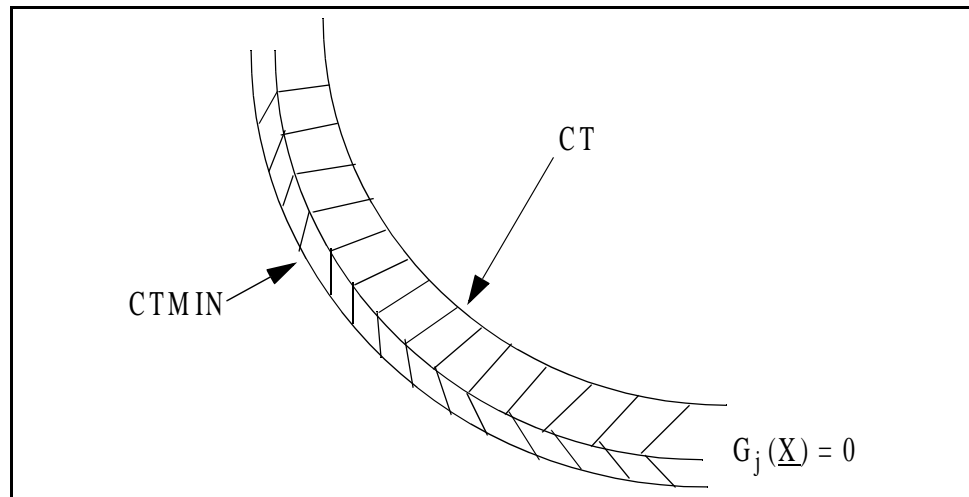


Figure 7. Relationship Between Constraint  $G$  and the Parameters CT and CTMIN

CTL,CTLMIN - These parameters have the same definition as CT and CTMIN, but for strictly linear constraints. Because numerical interpolation is more precise for linear constraints, these values are smaller in magnitude than CT and CTMIN. CTL is reduced during the optimization process to a magnitude approaching CTLMIN, but opposite in sign.

Caution: Do not define a constraint as linear unless you are absolutely sure it is. If a linear constraint is treated as nonlinear, efficiency is only slightly reduced, but if a nonlinear constraint is treated as linear, the result may be non-convergence.

DABALP - Used in IONED = 1 and 5. Convergence criteria in the Golden Section Method for the one-dimensional search. If IONED = 2 or 6, a larger value is used (by a factor of 100), since the Golden Section search will be followed by a cubic polynomial interpolation using the final four points.

If it is desired to find a very precise solution to the one-dimensional search, DABALP can be reduced. Alternatively, a larger value will give a less precise answer. It is normally not desired to change DABALP. The default value gives high precision on the assumption that function values are cheap, or else the Golden.Section method would not be used.

DABOBJ - Used in all IOPT options. Absolute convergence criteria for optimization. If the objective function is changed by less than this value for ITRMOP iterations, the optimization will terminate. If the objective function changes by more than one order of magnitude during optimization, the default value for DABOBJ will probably cause premature convergence. In this case, it is usually desirable to set DABOBJ to a small number, say 0.001, and let the optimization process converge based on the relative change criteria defined by DELOBJ.

DABOBM - Used with all strategies. This is the value of DABOBJ used during the optimization sub-problem and is larger than DABOBJ. The reason for this relaxed convergence criteria is that the optimizer will be called repeatedly by the strategy. Therefore, the solution of the sub-problem during the early stages is not as critical as if a strategy is not used. The rules for changing DABOBJ apply here also.

DABSTR - Used with all strategies. This is the overall absolute convergence criteria. If the objective function is changed by less than this value for ITRMST iterations by the strategy, the optimization will terminate. This has the same general meaning as DABOBJ and the same rules apply.

DELALF, DELOBJ, DELOBM, DELSTR - These parameters are used where their counterparts DABxxx are used above. However, here the convergence is tested on the relative change in the objective function. The combination DABxxx and DELxxx work together to form the diminishing returns convergence criteria in ADS. Here by relative change we mean the fractional change in the value of the objective function between successive iterations.

If the objective function is quite small in magnitude, a relative change, of say one percent, may not be meaningful and so the absolute criteria are relied on to detect convergence. On the other hand, for large values of the objective function, the absolute change is considered of lesser importance and the relative criteria tend to control the optimization convergence.

DLOBJ1 - Used in all one-dimensional searches. On the first search, it is difficult to estimate a desirable move parameter, ALPHA, because the optimization process has no history. DLOBJ1 is used to estimate the ALPHA which will reduce the objective function by this fraction, based on a linear approximation to the problem. Thus, for DLOBJ1= 0.1, the first step in the one-dimensional search will attempt to reduce the objective by ten percent.

If the problem is highly nonlinear, so that the calculated ALPHA is consistently less than the proposed ALPHA, efficiency will be improved by reducing DLOBJ1. Alternatively, if the calculated ALPHA is consistently greater than the proposed ALPHA, it is desirable to increase DLOBJ1.

DLOBJ2 - Used in all one-dimensional searches. If the objective function is quite large in magnitude, a move to reduce the objective by the fraction DLOBJ1 may be too large. In this case, DLOBJ2 is used to limit the change in the objective function to the magnitude of DLOBJ2. In other words, DLOBJ1 is a fractional change and DLOBJ2 is an absolute change. As with DLOBJ1, if the proposed moves are too large, DLOBJ2 may be reduced and if they are too small, DLOBJ2 may be increased.

Both DLOBJ1 and DLOBJ2 are updated during the optimization process by keeping track of progress. Therefore, their initial values are usually not too critical except for highly nonlinear problems where no progress can be made due to very large estimates for ALPHA.

- DX1, DX2 - Used in all one-dimensional searches. These parameters have an equivalent meaning to DLOBJ1 and DLOBJ2, but here are applied to each component of the X vector. The same general rules apply. The purpose of DX1 and DX2 is to prevent very large initial changes in the components of the X vector. DX1 and DX2 are also updated during the optimization process.
- EPSEN- Used in ISTRAT=2, 3 and 4. Initial transition point from interior to exterior penalty function. EPSPEN is a small negative number, and is updated during optimization. If significant constraint violations are observed in the initial stages, this should be made more negative. The basic concept is that, if the design is feasible, a penalty is imposed for each constraint proportional to one over the constraint value as the design approaches the feasible boundary (G approaches zero from the negative side). When a  $G = EPSPEN$ , the form of the constraint penalty changes to a linear (ISTRAT = 2), quadratic (ISTRAT = 3) or cubic (ISTRAT=4) function of the constraint.
- EXTRAP - Used in IONED=4 and 8. The maximum polynomial extrapolation allowed. These one-dimensional search routines do not require that bounds first be found on the minimum of the function, but instead extrapolate for the solution. Because extrapolation is relatively unreliable, EXTRAP is used to limit the amount of extrapolation. If the objective and constraints are nearly linear or quadratic, extrapolation is usually reliable, and may even be increased. If the objective and/or constraints are highly nonlinear, this is ill-conditioned and EXTRAP should be reduced. If this occurs, it is recommended to use IONED=3 or 7 instead.
- FDCH - Used if IGRAD=0 for internal gradient calculations by ADS. Gradients are calculated by first forward finite difference unless a variable is at its upper bound. In this case, a first backwards finite difference step is taken and no check is made to insure that the resulting design variable is above its lower bound. FDCH is the finite difference step size as a fraction of the design variable being perturbed. If high precision is available and required in evaluating the objective and constraint functions, this should be reduced. If the analysis is iterative, with its own internal convergence parameters, FDCH may have to be increased. For iterative analysis, a value of FDCH up to 0.05 may be appropriate for constrained problems, but FDCH=0.02 is a more reasonable limit for unconstrained problems.

The reason for this is that ADS seeks the point where the gradient is zero for unconstrained problems, and if FDCH is large, this is numerically difficult and will lead to false gradient information.

On the other hand, for constrained problems, the gradients of the objective and critical constraints are usually non-zero at the upturning and so precision in their calculation is not as important.

- FDCHM- Used if IGRAD=0 for internal gradient calculations by ADS. This is the minimum absolute steplength for gradient calculations. This is used if the component of X is near zero since a fractional change may not be meaningful. The same general rules apply as with FDCH.
- GMULTZ - Used with ISTRAT=8. Initial penalty parameter. If the design stays well inside the feasible region, this can be reduced. If the design moves well outside the feasible region, this should be increased.
- PMLT - Penalty multiplier for equality constraints. ADS treats equality constraints by adding a linear multiplier times the constraint values to the objective and then treating the constraint as an inequality. If the equality constraints are not sufficiently close to zero at the optimum, increase PMLT. If convergence is very slow because the optimization is trying to follow this constraint too closely, decrease PMLT.
- PSIAZ - Used with ISTRAT=8. Used to avoid constraint violations. This has little effect because of algorithmic modifications made to ADS and the fact that the ADS optimizers can deal well with constraint violations.
- RMULT - Used with ISTRAT=1 and ISTRAT=2 - 5 for equality constraints. Penalty factor multiplier for the exterior penalty function method. If the strategy iterations progress slowly from far outside the feasible region, RMULT should be increased. If the design seems to become near feasible quickly, but then converge poorly, RMULT should be decreased. RMULT should never be less than about 1.1.
- RMVLMZ - Used with strategies 6 through 9. Initial relative move limits. If the design variables alternately go from + to - the move limits, this should be reduced. If the design variables repeatedly hit one side (upper or lower limit), this should be increased. Also increase RMVLMZ if the problem is known to be nearly linear or if the optimum is always fully constrained (has as many active constraints as there are design variables).

- RP - Used with ISTRAT=1 and 5 and for ISTRAT=2, 3 and 4 for equality constraints. Initial penalty parameter for the exterior penalty function method and the Augmented Lagrange Multiplier Method and for equality constraints for exterior and extended interior penalty function methods. If the optimum of the first unconstrained sub-problem is well outside the feasible region, increase RP. If the optimum of the first unconstrained sub-problem is feasible or very near feasible for ISTRAT=1, reduce RF.
- RPMAX - Used with ISTRAT=1 and 5 and for ISTRAT=2, 3 and 4 for equality constraints. Maximum value of RP to be used. If optimum is significantly outside the feasible region, increase RPMAX. If constraints are satisfied much more precisely at the optimum than required, reduce RPMAX.
- RPMULT- Used with ISTRAT=2, 3 and 4. Multiplier on RPPRIM for consecutive iterations. Increase if convergence is very slow but reliable. Decrease if convergence is far from (expected) optimum.
- RPPMIN- Used with ISTRAT=2, 3 and 4. Minimum value of RPPRIM to be used. If optimum is well inside the feasible region, reduce. If constraints are more precisely satisfied than required, increase.
- RPPRIM - Used with ISTRAT=2, 3 and 4. Initial penalty parameter for extended penalty function methods. If the result of the first unconstrained sub-problem is well inside the feasible region, reduce. If the result is right at the constraint boundaries, increase. RPPRIM is reduced on each iteration by a factor RPMULT.
- STOL - Used by all optimizers. Tolerance on the components of the search direction to indicate convergence by the Kuhn-Tucker conditions. The Kuhn-Tucker conditions are the mathematical conditions that are satisfied at a precise optimum. These cannot generally be used as the only convergence criteria since this is numerically difficult to achieve. However, when the Kuhn-Tucker conditions are met, it is used as a convergence criterion which supersedes all others. Reducing STOL imposes a more stringent convergence criterion.
- THETAZ - Used with IOPT=4 and 5. Normally should not be changed if IOPT=5. THETAZ is the nominal “push-off” factor for the method of feasible directions. If the constraints are highly nonlinear, increase THETAZ. If constraints are nearly linear, reduce THETAZ. There is an interaction between the constraint tolerance CT and THETAZ. If constraints are highly nonlinear, it is usually preferable to increase the magnitude of CT (make CT more negative).

ZRO - Numerical “zero” to indicate reasonable machine accuracy. Primarily used internally by ADS to prevent floating point divide or to indicate that the numerical zero of a function has been found. Normally should not be changed.

### **Integer Parameters Contained on IWK**

ICNDIR - Used by all optimizers. Conjugate direction or variable metric restart parameter to restart with a steepest descent direction if the objective is currently unconstrained (no constrains are active or violated). The default is usually adequate. If no progress is being made, ADS will automatically override ICNDIR and restart with a steepest descent direction.

It is a worthwhile exercise to solve an unconstrained problem with ICNDIR=1. This will use a steepest descent direction on every iteration. This is the classical steepest descent method and a comparison of this with the other unconstrained minimization methods in ADS will indicate the power of modern methods.

ISCAL - Turns automatic scaling on/off. If the problem has been carefully scaled, set ISCAL=0. Also, In general, if the optimization progress is slow, it is worthwhile to try ISCAL=0 to see if the automatic scaling in ADS is actually causing some ill-conditioning. The present scaling routine in ADS is much improved from the original one and so should not cause difficulty.

ITMAX - Maximum number of iterations in the optimizer. If function evaluations are extremely expensive, reduce ITMAX. In the extreme case ITMAX=1 or 2 is justified because the first few iterations are where most progress is made. If function evaluations are not expensive and the optimization terminates by reaching ITMAX, it should be increased. When using a strategy, ITMAX should be at least 10 to insure reasonable solution of the sub-problem. When using ISTRAT=6, 7 or 9, ITMAX should not be reduced because the optimizer is only solving a simple and inexpensive approximate sub-problem. In these cases, the optimizer does not call for detailed function evaluations.

- ITRMOP - Used by all optimizers. The number of consecutive iterations that must satisfy the absolute or relative convergence criteria before optimization is terminated. Usually ITRMOP should be at least 2 because it is common to make little progress on one iteration, only to make major progress on the next. Therefore, ITRMOP=.2 will allow a second try before terminating.
- ITR~ST - Used by all strategies. The number of consecutive sub-optimizations that must satisfy the absolute or relative strategy convergence criteria before optimization is terminated. The same rules apply as to ITRMOP, except ITRMST = 1 may be used. This is because, the sub-problem cannot make progress, and therefore solving an additional sub-optimization problem will probably not help.
- ITRMST - Used by all strategies. The number of consecutive sub-optimizations that must satisfy the absolute or relative strategy convergence criteria before optimization is terminated. The same rules apply as to ITRMOP, except ITRMST=1 may be used. This is because, the sub-problem cannot make progress, and therefore solving an additional sub-optimization problem will probably not help.
- JONED - Used with ISTRAT= 8. This strategy performs an additional one-dimensional search. Normally the one-dimensional search defined by IONED is used. If a different one is desired, it is defined by JONED. Sometimes efficiency or reliability can be improved by using IONED=5 or 6 and JONED=7. This is because the optimization sub-problem does not call for detailed function evaluations and so can use a less efficient, but more precise one-dimensional search.
- JTMAX - Maximum number of strategy iterations to be allowed. Reduce if optimization is very expensive. Increase if optimization is stopped by reaching the maximum number of strategy iterations and function evaluations are cheap.