A Constraint Management Approach for Optimal Design of Mechanical Systems

DISSERTATION

Presented in Partial Fulfillment of the Requirements for
the Degree Doctor of Philosophy in the Graduate
School of The Ohio State University

by

Rajiv Agrawal, B. Tech., M.S.

The Ohio State University
1991

Dissertation Committee
Dr. G. L. Kinzel
Dr. K. Ishii
Dr. D. R. Houser

Approved by
G. L. Kinzel, Advisor
K. Ishii, Co-Advisor
Department of Mechanical Engineering
To My Parents
ABSTRACT

Design of mechanical systems is a complex process. Often the design can be modelled as a network of equality and inequality constraints. Though the solution strategy to handle these constraints varies from one design scenario to another, the underlying concepts of constraint management are applicable to most designs. This thesis uses these concepts to develop a generic framework for the interactive design of mechanical systems. The goal of the research is to provide a constraint management shell where the mathematical model for the design can be expressed in a declarative manner. A nonlinear occurrence matrix is used to represent the governing equality and inequality constraints. Constraint management algorithms for design decomposition, forward and backward dependency are presented by using the nonlinear occurrence matrix. Further, the occurrence matrix approach is also used to handle the initial basis selection and basis interchange issues in the Generalized Reduced Gradient (GRG) method for constraint optimization. The optimization phase also considers discrete variables. Discrete variables represent design variables that are either integer-valued, or have to adhere to standard dimensions. A Branch and Bound algorithm is used with the GRG method to handle the discrete variables. Robust numerical tools are developed to handle the solution of simultaneous nonlinear equations, an essential part of the both the GRG method for optimization and general constraint satisfaction. As an illustration of the design framework, a design method for spur and helical gears based on the American Gear Manufacturers Association (AGMA) standards is presented.

Keywords: Constraint Management, Nonlinear Constrained Discrete Optimization, Branch and Bound Algorithm, Generalized Reduced Gradient Method, Simultaneous Nonlinear Equations, Gear Design.
ACKNOWLEDGEMENTS

I wish to express sincere gratitude to Professor Gary L. Kinzel for his guidance and insight throughout the research. I thank Dr. Kos Ishii for bringing in new ideas, giving me the computer hardware, and helping me with my thesis. I also want to thank Prof. Donald R. Houser for his advice and comments in the gear design area.

The research for this thesis was funded by the GOCAD group of companies. A part of the research was supported by the Gear Laboratory, an NSF Grant No. DMC8810824, and NSF Grant No. DDM9157213. This support is gratefully acknowledged. I also express my sincerest thanks to the good folks of the Robinson Computer Graphics Laboratory. Without the computer support of RCGL, this thesis would not have been possible.

Thanks also go out to all my project colleagues over the years, especially my good friend Raman who started work on the Design Shell. I also thank the whole Design Shell team, Pavithran, Sridhar, Ramki, and Junhua, for their valuable suggestions and criticisms.

I want to thank all my friends that I made during my stay at The Ohio State University for easing the pains of the graduate student life. Finally, I thank my parents and my brother for their constant encouragement, love and understanding.
VITA

June 5, 1963 .............................................. Born - Mathura, U.P., India
1984 ................................................................. B. Tech., Indian Institute of Technology,
                                             Delhi, India
1984-85 ......................................................... University Fellow, The Ohio State
                                             University, Columbus, Ohio
1986 ................................................................. M.S., The Ohio State University,
                                             Columbus, Ohio
1986 - Present ........................................ Graduate Research Associate, Department
                                             of Mechanical Engineering, The Ohio State
                                             University, Columbus, Ohio

PUBLICATIONS

E. Raghavacharyulu, Rajiv Agrawal, and Sandeep Gupta, "Computer-Aided Mechanism
Design with Computer Graphics," Proceedings of National Conference on Machines and

Rajiv Agrawal and Gary L. Kinzel, "The Optimum Design of Spur and Helical Gears based
on AGMA Strength and Wear Criteria," Proceedings of ASME Computers in Engineering,

Raman V. Srinivasan, Rajiv Agrawal, and Gary Kinzel, "Design Shell: A Framework for

Rajiv Agrawal, Biplab Sarkar, Gary L. Kinzel, and Donald R. Houser, "Optimal Design of
Bevel Gears," Proceedings of the AGMA Fall Technical Meeting, Paper No. 90-FTM-15,

Rajiv Agrawal and Gary L. Kinzel, "A Comparison of the Performance of Non-linear
Equation Solvers for Machine Design Problems," Accepted for publication in the ASME
Journal of Mechanical Design.

Rajiv Agrawal, Gary L. Kinzel, Raman V. Srinivasan, and Kos Ishii, "Engineering
Constraint Management Based on an Ocurrence Matrix Approach," Accepted for
publication in the ASME Journal of Mechanical Design.


**FIELDS OF STUDY**

Major Field: Mechanical Engineering

# TABLE OF CONTENTS

Dedication .......................................................................................................................... ii  
Abstract .............................................................................................................................. iii  
Acknowledgements ............................................................................................................ iv  
Vita ........................................................................................................................................ v  
List of Tables ..................................................................................................................... xi  
List of Figures ................................................................................................................... xii  
Nomenclature .................................................................................................................... xiv

## CHAPTER PAGE

### I. INTRODUCTION

1.0 Background and Motivation ...................................................................................... 1  
1.1 Research Objectives ................................................................................................. 4  
1.2 Thesis Organization ................................................................................................. 5

### II. LITERATURE REVIEW

2.0 Introduction .............................................................................................................. 7  
2.1 Constraint Management (CM) ................................................................................. 7  
2.2 Nonlinear Constrained Discrete Optimization ......................................................... 10  
2.2.1 Generalized Reduced Gradient (GRG) method ................................................. 10  
2.2.2 Discrete Optimization ......................................................................................... 11  
2.3 Nonlinear Equation Solving .................................................................................... 13  
2.4 Gear Design ............................................................................................................ 14

### III CONSTRAINT MANAGEMENT

3.0 Introduction .............................................................................................................. 17  
3.1 Constraint Representation ....................................................................................... 20  
3.1.1 Nonlinear Occurrence Matrix ........................................................................... 20
3.1.2 Design State .................................................. 24
3.1.3 Block Information ........................................... 25
3.1.4 Level Information ............................................ 27
3.1.5 Order Information ........................................... 27
3.1.6 Solution Sequence ........................................... 27
3.2 Constraint Management Algorithms ........................................... 29
  3.2.1 Design Decomposition Algorithm (DDA) ......................... 29
    3.2.1.1 Stage 1: Determining \([A^*]\) from \([A]\) ..................... 32
    3.2.1.2 Stage 2: Determining \([A_s]\) using \([A^*]\) ................. 33
    3.2.1.3 Stage 3: Permuting \([A_s]\) to a Zero-Free Diagonal Form .... 34
    3.2.1.4 Stage 4: Permuting \([P_1A_s]\) to a Block Triangular Form .... 34
  3.2.2 Forward Dependency Algorithm (FDA) .......................... 35
    3.2.2.1 Respecification Problem ................................... 37
    3.2.2.1 Unspecification Problem ................................... 37
    3.2.3 Backward Dependency Algorithm (BDA) ......................... 38
  3.3 Hidden constraints, Inconsistency and Redundancy ................ 42
  3.4 Determining an Optimum Form of the Equations .................... 44
  3.5 Example - Design of a Torsion Bar Spring ......................... 51

IV. NONLINEAR EQUATION SOLVER ............................................. 61
  4.0 Introduction .................................................. 61
  4.1 Newton's Method ............................................. 62
  4.2 Levenberg-Marquardt Method ................................... 63
  4.3 Powell's Method ............................................. 64
  4.4 Brent's Method ............................................. 66
  4.5 Brown's Method ............................................. 68
  4.6 Secant Method with Broyden Update ............................. 69
  4.7 Scaling ..................................................... 70

V. NONLINEAR OPTIMIZATION ............................................ 72
  5.0 Introduction .................................................. 72
  5.1 Modeling a standard NLP into a Constraint Management Problem (EOP model) ........................................... 75
  5.2 Constraint Management Based GRG (CMB-GRG) Algorithm ......... 76
5.2.1  Broyden-Fletcher-Goldfarb-Shanno (BFGS) Update ........... 79
5.2.2  Initial Basis Selection Algorithm .......................... 80
5.2.3  Basis Interchange Algorithm .............................. 82
5.2.4  Example ..................................................... 84
5.3  Discrete Optimization........................................ 86
  5.3.1  Current Methods ........................................ 87
  5.3.2  Branch and Bound Method ............................... 89
  5.3.3  Subproblem Selection .................................. 90
  5.3.4  Branching Variable Selection .......................... 91
  5.3.5  Example - Design of a Gear Train ....................... 92

VI.  DESIGN SHELL FRAMEWORK ..................................... 96
    6.0  Introduction ............................................... 96
    6.1  Design Shell Modules ..................................... 97
    6.2  Design Shell User Interface ............................... 97
      6.2.1  The FILE Menu ....................................... 99
      6.2.2  The EDIT Menu ....................................... 100
      6.2.3  The COMMANDS menu ................................. 101
      6.2.4  The OPTIMIZE menu .................................. 102
      6.2.5  The UTILITIES menu ................................ 103
    6.3  The Equations Module ..................................... 103
    6.4  The Graphics Module ..................................... 104
    6.5  Implementation details ................................... 104
    6.6  Design Shell Configuration ............................... 104
      6.6.1  Cases file ........................................... 106
      6.6.2  Variable description file ............................ 107
      6.6.3  Archive file ......................................... 111
      6.6.4  Help files ........................................... 111
      6.6.5  File of File Names ................................... 111
    6.7  Design Shell Examples .................................... 112
      6.7.1  Design of a Torsion Bar Spring ....................... 112
      6.7.2  Optimization of a Torsion Bar Spring ................ 118
      6.7.3  Optimization of a Helical Compression Spring ........ 120
VII. CONSTRAINT-BASED GEAR DESIGN .................................................. 126
  7.0 Introduction .............................................................................. 126
  7.1 Gear Design Model Development .......................................... 127
    7.1.1 Geometry ........................................................................... 127
    7.1.2 AGMA Strength and Wear Calculations .............................. 130
    7.1.3 Other constraints ............................................................... 133
  7.2 Design Shell Formulation ....................................................... 137
  7.3 Gear Design Examples ............................................................ 137
    7.3.1 Example 1 ......................................................................... 137
    7.3.2 Example 2 ......................................................................... 145
    7.3.3 Example 3 ......................................................................... 145

VIII. CONCLUSION ............................................................................... 148
  8.0 Summary .................................................................................. 148
  8.1 Research Contributions .......................................................... 150
  8.2 Limitations ............................................................................... 151
  8.1 Future Research ........................................................................ 152

Bibliography .................................................................................. 154

APPENDICES
  A. Design Shell Files for Helical Gear Design ............................... 169
  B. Equations Module for Helical Gear Design ............................... 179
  C. Graphics Module for Helical Gear Design ............................... 199
# LIST OF TABLES

<table>
<thead>
<tr>
<th>TABLE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1 Blocks, I/O, Level, and Order corresponding to the Torsion Bar example for the input sequence ( {x_{11}, x_3, x_9, x_1, x_2} )</td>
<td>57</td>
</tr>
<tr>
<td>5.1 Results from the discrete optimization of the gear train</td>
<td>95</td>
</tr>
<tr>
<td>7.1 Recommended Diametral Pitches</td>
<td>136</td>
</tr>
<tr>
<td>7.2 Input variables for an example analysis</td>
<td>138</td>
</tr>
<tr>
<td>7.3 Output variables for an example analysis</td>
<td>139</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>FIGURE</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>2</td>
</tr>
<tr>
<td>1.2</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>22</td>
</tr>
<tr>
<td>3.2</td>
<td>26</td>
</tr>
<tr>
<td>3.3</td>
<td>31</td>
</tr>
<tr>
<td>3.4</td>
<td>40</td>
</tr>
<tr>
<td>3.5</td>
<td>51</td>
</tr>
<tr>
<td>5.1</td>
<td>73</td>
</tr>
<tr>
<td>5.2</td>
<td>74</td>
</tr>
<tr>
<td>5.3</td>
<td>92</td>
</tr>
<tr>
<td>5.4</td>
<td>94</td>
</tr>
<tr>
<td>6.1</td>
<td>98</td>
</tr>
<tr>
<td>6.2</td>
<td>99</td>
</tr>
<tr>
<td>6.3</td>
<td>105</td>
</tr>
<tr>
<td>6.4</td>
<td>107</td>
</tr>
<tr>
<td>6.5</td>
<td>109</td>
</tr>
<tr>
<td>6.6</td>
<td>110</td>
</tr>
<tr>
<td>6.7</td>
<td>113</td>
</tr>
<tr>
<td>6.8</td>
<td>113</td>
</tr>
</tbody>
</table>
6.9 New Specification of $x_2$ ........................................... 114
6.10 Solution sequence .................................................. 114
6.11 Respecification of $x_1$ ............................................. 115
6.12 Unspecification of $x_1$ ............................................ 115
6.13 Reverse Specification of $x_7$ ..................................... 116
6.14 Convergence to an extraneous root ............................. 116
6.15 Changing initial values ............................................ 117
6.16 Convergence to the correct solution ........................... 117
6.17 Setting lower limits for minimization .......................... 119
6.18 Minimization of volume ........................................... 119
6.19 Specifications before optimization ............................. 123
6.20 Setting up the lower limits ....................................... 123
6.21 Minimization of weight .......................................... 124
6.22 Weight optimization results ..................................... 124
6.23 Minimization of free volume .................................... 125
6.24 Volume optimization results .................................... 125
7.1 Values of the gear design variables ............................. 141
7.2 Scaled picture of the helical gears ............................... 143
7.3 Function residual values .......................................... 143
7.4 Solution sequence .................................................. 144
7.5 Graph of bending safety factor vs horsepower ............... 144
7.6 Design variables from example 2 ............................... 146
7.7 Solution sequence from example 2 ............................. 146
7.8 Design variables from example 3 ............................... 147
7.9 Solution sequence from example 3 ............................. 147
NOMENCLATURE

[0] null matrix
[A] occurrence matrix
[A'] active occurrence matrix
[B] nonlinear occurrence matrix
C_f surface finish factor for wear
C_H hardness factor for wear
C_p elastic coefficient
a_p, a_G addendum constants for pinion and gear
b_p, b_G dedendum constants for pinion and gear
C_d center distance (in)
[D] partial derivatives of h(x) with respect to z (decision variables)
d, D pitch diameters of pinion and gear (in)
d_b, D_b base diameters of pinion and gear (in)
d_o, D_o outside diameters of pinion and gear (in)
d_r, D_r root diameters of pinion and gear (in)
E_P, E_G moduli of elasticity for pinion and gear (psi)
F face width
f(x) complete constraint vector, dimension M
g(x) inequality constraint vector, dimension p
[G] occurrence matrix for the inequality constraints
h(x) equality constraint vector, dimension m
[H] occurrence matrix for the equality constraints
h_a hob addendum
I wear geometry factor
[I] identity matrix
J strength geometry factor
K_a, C_a application factors for strength and wear
K_L, C_L life factors for strength and wear
K_m, C_m load distribution factors for strength and wear
K_R, C_R reliability factors for strength and wear
K_S, C_S size factors for strength and wear
K_T, C_T temperature factors for strength and wear
K_v, C_v internal dynamic factors for strength and wear
m number of equality constraints
m_G gear ratio
M total number of constraints, equality and inequality, (m+p)
N dimension of the augmented design vector including slack variables, (n+p)
n number of design variables
N_G number of teeth on the gear
N_P number of teeth on the pinion
n_p pinion shaft speed (rpm)
N_Pmin minimum number of teeth to prevent undercutting
P horsepower (hp)
p number of inequality constraints. Same as number of slack variables
p_c transverse circular pitch (in)
p_b transverse base pitch (in)
p_{bN} normal base pitch (in)
\[ P_d \] transverse diametral pitch (1/in)
\[ P_d \] transverse diametral pitch (1/in)
\[ r_T \] hob tip radius (in)
\[ s_{at} \] allowable bending stress number (psi)
\[ SF_t \] safety factor for strength
\[ SF_c \] safety factor for wear
\[ [S] \] Jacobian of \( h(x) \) with respect to \( y \) (state variables)
\[ T_P \] pinion torque (lb-in)
\[ U \] set of the unknown variables
\[ W_t \] transmitted tangential load
\[ x \] design vector, dimension \( N \)
\[ X \] set of the design vector
\[ y \] dependent (basis) vector, dimension \( m \)
\[ Y \] set of the output variables
\[ z \] independent (non-basic) vector, dimension \( (n-p) \)
\[ Z \] set of input variables
\[ \gamma_P \] Poisson's ratio of pinion
\[ \gamma_G \] Poisson's ratio of gear
\[ \mu_P \] Poisson's ratio of pinion
\[ \mu_G \] Poisson's ratio of gear
\[ \phi_n \] normal pressure angle
\[ \phi_t \] transverse pressure angle
\[ \psi \] helix angle
\[ \psi_b \] base helix angle
\[ \cup \] set union
\[ \cap \] set intersection
\[ \Phi \] null set
\[ \in \] is a member of a set
\[ \notin \] is not a member of a set
\[ \wedge \] logical AND operator
\[ \exists \] there exists
\[ \forall \] for all
\[ \Omega \] solution sequence for a given design state

Additional terms:

BDA Backward Dependency Algorithm
CM Constraint Management
CMB-GRG Constraint Management Based Generalized Reduced Gradient
DDA Design Decomposition Algorithm
DS Design Shell
FDA Foward Dependency Algorithm
GRG Generalized Reduced Gradient
Chapter I

INTRODUCTION

1.0 Background and Motivation

During the last twenty years, the process of mechanical design has slowly moved from the engineer's drawing table to powerful mainframe and microcomputers. Numerous computer programs and packages are now available to help the designer in analyzing existing designs. There is, however, a critical need for computer algorithms that perform the task of synthesis and are capable of locating optimum configurations. With the growth in the field of Artificial Intelligence (AI), expert systems for the design of specific machine components have also started becoming available. But, with this proliferation, there is a common theme that evolves which can be used to create a framework that would be generic for any mechanical system.

During the course of an ongoing research project at The Ohio State University, several computer programs were developed that were aimed at integrating device independent graphics into machine design. Separate modules dedicated to the design of different machine elements, e.g. gears, springs, cams, etc. were written. This project, henceforth referred to as the Graphics-oriented Computer Aided Design (GOCAD) project, was aimed at providing CAD tools for machine design for several different computer platforms under a consistent user-interface.

The GOCAD project identified that there are a number of common constraint management principles that can be used to tackle the complex tasks of design. It was thus thought that a generic framework should be created that can function as a different design module depending upon the knowledge it had. The idea is similar to an expert system which
functions primarily on its knowledge base. Since the main thrust of the GOCAD project was to provide numerical design tools, the "numerical knowledge base for such a shell should contain the governing constraints for design. At a later stage, the numerical knowledge base could be complemented with empirical domain knowledge to simulate an intelligent design activity.

Based on the above ideas, a conceptual model of the Design Shell was created. The model is shown in Fig. 1.1. As stated earlier, the design shell derives all the necessary information about a particular mechanical system from its knowledge base. Once the mathematical model becomes part of the design shell, the resulting system performs as a mechanical design program. Some supporting files that describe the mechanical system are required for the proper functioning of the program.

![Diagram](image)

Fig. 1.1: Conceptual Model of the Design Shell

The numerical knowledge base for the shell has to be defined in a concise manner so that it can be used as an effective design tool. It was seen that most often the mathematical model for a given mechanical system can be characterized by $m$ nonlinear equalities and $p$ inequalities in $n$ unknowns:
\[ h_i(x_1, x_2, \ldots, x_n) = 0, \quad i = 1, 2, \ldots, m \]  
\[ g_i(x_1, x_2, \ldots, x_n) \leq 0, \quad i = 1, 2, \ldots, p \]  

In addition, there might be limit constraints:

\[ x^l \leq x \leq x^u \]

where some of the limits could be \(-\infty\) or \(+\infty\) signifying that the variable is unbounded in a certain direction.

The above representation is especially well adapted for an optimization framework where any variable can be designated as the objective function. Optimization is always a crucial part of any design process since the designer wants an optimal configuration that satisfies the design specifications. In machine design, both the objective function and the constraints are often nonlinear. Linear programming techniques such as the Simplex method are not suitable for nonlinear problems. Hence, nonlinear optimization techniques such as the Generalized Reduced Gradient method or the Sequential Quadratic Programming have been conventionally used in machine design.

In the past, a significant amount of research has concentrated on nonlinear programming techniques for continuous variables. In mechanical design problems, we often encounter variables that cannot take continuous values. Such variables are referred to as discrete variables. Discreteness in the design of mechanical components can come about due to a number of reasons. Sometimes, it is required by the physical constraint of the problem e.g. integer number of gear teeth. Also, due to manufacturing limitations (or reducing manufacturing costs), it might be necessary for some design variable to take on only discrete choices e.g. standard diametral pitch for gear design. The selection of material and the choice of a manufacturing method are other examples of discrete variables.

In the past, engineers have often tried to ignore the discrete variables. For example, in gear design, to tackle the problem of integer number of gear teeth, one can assume the variable to be continuous and truncate (or round up) the optimum value. But the problem becomes complex when there are several discrete variables. It is not hard to see that very soon we
would run into combinatorial optimization. To handle discrete variables, we need to enhance the existing techniques.

1.1 Research Objectives

The aim of the current research is to develop a machine Design Shell capable of handling routine design problems. The shell will derive all its knowledge from a numerical knowledge base which will be the computer representation of the governing constraint network. Though the underlying constraint network will be assumed to be static, the shell will be constructed in such a way that updating the governing equations will be extremely easy.

Constraint Management (CM) algorithms will be developed based on an occurrence matrix representation of the constraints. The following four key issues for interactive engineering design will be addressed using the occurrence matrix approach:

1. **New Specification Problem**: Determine if any new variables can be computed based upon the updated constraint network.
2. **Respecification Problem**: Determine how a change in the value of an input variable affects the dependent computed variables.
3. **Unspecification Problem**: Determine which variables can no longer be computed when one of the previous specifications is removed.
4. **Reverse Specification Problem**: When a computed variable is changed to a specification, determine which of the original specifications should be released.

Since constraint satisfaction relies heavily on a nonlinear equation solver, six robust methods will be studied. The six methods that will be discussed in the thesis are:

1. Newton's method
2. Levenberg-Marquardt method
3. Powell's method
4. Brent's method
5. Brown's method
6. Secant method with Broyden’s Jacobian update

To go beyond constraint satisfaction, the current research will use the CM algorithms for nonlinear constrained optimization. The Generalized Reduced Gradient (GRG) method has been selected as it lends itself to the use of a nonlinear occurrence matrix. Some unresolved issues of initial basis selection and basis interchange in the GRG method will be studied. Discreteness constraints will also be considered when numerical optimization is performed. A Branch and Bound algorithm will be developed over the GRG method to perform discrete optimization.

Based on the CM and optimization algorithms, a Design Shell will be constructed. Fig. 1.2 shows the system architecture of the design shell. The different components are the constraint manager, optimizer, nonlinear equation solver, and a graphical user interface (GUI).

All the sub-components of the design shell derive information from the pre-compiled mathematical model and the supporting data files. In order to understand the implementation issues of the design shell, a prototype for the design of spur and helical gears will be developed. The synthesis of a gear set to meet a set of specifications is one of the most complex and interesting machine design problems. Gear design also has a number of discrete variables such as integer number of gear teeth, standard diametral pitches, and standard pressure angles. Several other machine components will be used as simpler examples to illustrate the CM algorithms.

1.2 Thesis Organization

The rest of the thesis is organized into seven major chapters. Chapter II provides a review of literature on existing constraint management, Generalized Reduced Gradient method for constraint optimization, discrete optimization techniques, and simultaneous nonlinear equation solving methods, and also presents a summary of research done in the area of gear design. Chapter III presents the complete mathematical development of constraint management ideas and presents all the CM algorithms. Chapter IV describes six nonlinear
equation solving methods that form the backbone for constraint satisfaction and optimization. Chapter V is devoted to the study of nonlinear constraint optimization and it describes the enhancements to the GRG method with emphasis on discrete variable handling. Chapter VI describes in detail the constraint management shell developed using the techniques described in Chapter III, IV, and V. Chapter VII presents the mathematical background used in conventional gear design and describes the program configuration under the design shell. Chapter VIII gives a summary of the entire thesis and presents recommendations for future research.
Chapter II

LITERATURE REVIEW

2.0 Introduction

This chapter provides a literature review to support the algorithms developed in the current research. Since the development of generic design framework involves a number of building blocks, each of these blocks had to be researched. The literature reviewed has been organized into four major sections - Constraint Management, Nonlinear Constrained Discrete Optimization, Nonlinear Equation Solving, and Helical Gear Design.

2.1 Constraint Management (CM)

Constraint management techniques have been used in a number of fields such as variational geometry based Computer-Aided Drafting systems, mechanical design, and chemical process synthesis. Most of the work done in the field of constraint management is closely related to sparse matrix research since the underlying matrix representations of the designs tend to be quite sparse.

Steward (1962, 1965) was one of the earliest investigators who examined the structure of simultaneous equations. He presented algorithms for partitioning and tearing systems of simultaneous equations. Rudd and Watson (1968) used structural arrays to devise elegant input variable selection algorithms to avoid the large number of simultaneous equations. Most of their work was done for chemical process synthesis where the governing constraints are often linear and the occurrence matrix is quite sparse. Algorithms to partition and reorder directed graphs were shown by Christensen and Rudd (1969).
Himmelblau (1973) observed different patterns in occurrence matrices in the domain of nonlinear equation processing, linear and dynamic programming.

Soylmez and Seider (1973) presented an algorithm called Structural Analysis with Substitution (SWS) to determine decompositions in the Jacobian to reduce the burden on a simultaneous equation solver. A nonlinear structural matrix was used to represent and study the governing algebraic equations of chemical process engineering. Duff (1977) presented an excellent survey paper on sparse matrix research. Later, Duff and Reid (1978a, 1978b) described an algorithm to convert square sparse matrices into a block triangular form (BTF) in order to determine simultaneous equation sets, also referred to as strong components in graph theory. Their algorithm was based on work done by Tarjan (1972). A recent work done by Pothen and Fan (1990) is also aimed at converting a sparse matrix to its block triangular form. In order to permute a sparse matrix to a BTF, it is essential to convert the matrix to a zero-free diagonal form. Duff (1981a, 1981b, 1986, 1988) has compared various algorithms to achieve this configuration. Stadtherr and Wood (1984a, 1984b) compared a number of algorithms to reorder sparse matrix structures occurring in chemical process flowsheets.

Friedman and Leondes (1969a, 1969b, 1969c) provided a clear treatment of constraint theory using mathematical set operations. Shacham (1984) presented a new method to partition simultaneous nonlinear algebraic equations into smaller irreducible subsets. Derman and Wyk (1984) and later Derman and Sheppard (1985) developed a hierarchical equations solver based on directed graphs. Prasad and Kinzel (1986) presented an algorithm to perform decomposition of incidence matrices as the number of degrees of freedom is reduced in a mathematical model. Akin (1990) refers to the occurrence matrix as a Boolean Inference Array and presents an algorithm for the detection of solvable sets of nonlinear equations which is quite similar to the work of Prasad and Kinzel.

Sutherland (1963) was amongst the first researchers to treat 2-dimensional drafting as a constraint satisfaction problem. Borning (1977, 1981) and Borning and Duisberg (1986) described a simulation laboratory called ThingLab, which allowed a user to model objects based on constraint networks. Works of V. C. Lin (1981, Lin et al., 1981), Light (1980, 1982), Light and Gossard (1982, 1983), and Gallaher (1984) studied the application of constraint management techniques to problems from variational geometry. Serrano and
Gossard (1986) and Serrano (1984, 1987, 1989, 1990) used directed acyclic graphs (DAGs) to determine dependencies among variables and to identify sets of simultaneous equations. Serrano developed a constraint modeler that allowed interactive addition and deletion of constraints. Holtz (1983) also presented a constraint modeler called CONMAN which handled both equality and inequality constraints.

A commercial implementation of graph-theory based CM was done in the Mechanical Advantage 1000 package (Steinke, 1985; Cognition, 1990; Gifford, 1991). Chung and Schussel (1990) compared parametric and variational geometry environments and described a commercial package. Brown and Halpern (1990) have also brought out the differences between parametric and variational design. A recent article by Bissell (1991) reviewed two software packages that implement constraint management ideas on personal computers.

Kannapan and Marshek (1987) introduced the term design diagram based on graph theory to represent parametric machine design problems. Ishii (1987) described a framework for applying Active-constraint deduction (ACD) to mechanical system design. Sappossnek (1989) described an object-oriented constraint-based design system and highlighted the distinctions between parametric and constraint-based environments. Srinivasan (Srinivasan 1989; Srinivasan et al. 1990) presented a preliminary constraint manager called the Design Shell for interactive engineering design. Agrawal et al. (1991) extended the ideas presented in the Design Shell and developed algorithms that allowed interactive constraint manipulation using an active occurrence matrix approach.

Some research based on the ideas of constraint management has also been reported in the context of optimization. The Method of Optimum Design (MOD) developed by R. Johnson (1979) finds the best formulation of the constraints such that the solution effort is minimized. Circle diagrams are used to enumerate the possible formulations. Hammond and G. Johnson (1988, 1989) extended MOD by using Monotonicity Analysis (Papalambros and Wilde, 1980) to devise the Method of Alternate Formulations (MAF). Hammond's work was based on symbolic manipulation of constraints and all possible partitions of the design variables were studied. The partitioning of the design vector into state and decision variables exhibits a close similarity to the Generalized Reduced Gradient method for constraint optimization.
2.2 Nonlinear Constrained Discrete Optimization

Optimization is always an essential part of the design process. In order to solve, a general continuous optimization problem, the Generalized Reduced Gradient method has been found to be a very successful technique. Since, we are dealing with problems containing discrete variables, some discrete optimization techniques were also studied. Section 2.2.1 presents the work done by previous researchers on the GRG method and Section 2.2.2 describes the research done in the discrete optimization area.

2.2.1 Generalized Reduced Gradient (GRG) method

Generalized Reduced Gradient technique is an efficient method for handling equality as well as inequality constraints during design optimization. The earliest work on the Reduced Gradient method is most often attributed to Wolfe (1963, 1967) who proposed the method for solving nonlinear programming problems with linear constraints. Wolfe's ideas were very similar to the Simplex method where the variables are divided into basic and nonbasic sets.

Abadie and Carpentier (1969) extended Wolfe's ideas and developed the first computer implementation. Their code, GRGA, ranked first among 30 different algorithms on a set of Colville problems. Gabay and Luenberger (1976) discussed three variations of the Abadie's strategy by implementing different ways to perform unconstrained minimization. The use of a Lagrangian step size was suggested to speed up the line search.

Gabriele and Ragsdell (1977; Gabriele, 1975) implemented the GRG method in a nonlinear programming code called OPT. The program was tested for a number of design problems and later Gabriele (1980) extended the research to large structural problems. Lafrance et al. (1977) used the Davidon-Fletcher-Powell method for the unconstrained optimization phase of the GRG method and presented numerical results comparing their method with other nonlinear optimization techniques. Haggag (1981) implemented several methods to solve the unconstrained problem and presented a computer code called GRGAH. He tried four
different schemes and concluded that the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method was the best.

Lasdon et al. (1978, 1983) developed GRG2 code which was another successful implementation of the GRG method. Recently, some more researchers (Fisher, 1990; Smith, 1991) have adapted the GRG method for large scale problems. Fisher (1990) developed an algorithm that combines the GRG and Sequential Linear Programming technique. Smith (1991) extended the GRG2 code for sparse large scale problems.

Parkinson and Wilson (1988) developed a hybrid SQP-GRG algorithm for nonlinear programming. The SQP method does not need to maintain feasibility at intermediate iterations and requires less analysis than the GRG to reach an optimum. The proposed hybrid method replaces the reduced gradient with the SQP search direction, and the GRG line search is used to maintain feasibility at intermediate iterations.

A set of guidelines and rules of thumb have been proposed in the past to address the initial basis selection problem (Gabriele, 1975; Beightler et al., 1979; Li and Papalambros, 1988). Heltné et al. (1983) proposed that the Jacobian required to solve for the state variables can be decomposed into smaller irreducible sets. They also presented an algorithm to generate the decomposition for every basis change operation without starting from the original occurrence matrix.

Gabriele and Beltrachi (1987) and later Renaud and Gabriele, 1989) have addressed the issues of cycling and degeneracy in the GRG method that can occur depending upon the starting point for the optimization. A number of researchers (Lasdon et al., 1978; Schittkowski, 1980; Sandgren and Ragsdell, 1982) have compared the GRG method to other optimization methods and found it to be extremely fast and robust.

### 2.2.2 Discrete Optimization

Conventionally, discrete optimization has been successfully applied to operations research problems such as production planning, machine scheduling, and job assignment. Problems such as transportation and plant layout also fall in the category of integer
programming (which is a subset of discrete optimization). Travelling salesman (later generalized to Multiple travelling salesman), minimum spanning tree, set covering and knapsack problem are also classic examples which require the variables to take on discrete (generally 0-1) values. Design of structural elements is another area where discrete optimization has been used with success. In the design of structures, the designer is looking for standard section sizes.

Discrete programming techniques, particularly integer programming, are quite well developed in the linear programming domain (Murty, 1976; Garfinkel, 1972; Beale, 1988; Parker, 1988). Different algorithms have been used to incorporate discrete variables into the optimization process for engineering design. Fibonacci search has been shown to work for small optimization problems (Wilde, 1978). Sidall (1982) suggested an extension of the Box's complex method. Fu et al. (1989) used a variation of the penalty function approach to satisfy the discreteness requirement on the variables. Some other researchers (Givsvid and Moe, 1972; Pappas and Allentuch, 1974) have also presented different algorithms for discrete optimization.

Cha (1987, 1989a, 1989b) gave an excellent mathematical basis for the discrete optimization problem and used a variation of the Recursive Quadratic Programming (RQP) method to solve the problem. Olsen and Vanderplaats (1989) have proposed a Sequential Linear Discrete Programming (SLDP) so that existing integer programming techniques can be used. Recently, Loh and Papalambros (1990a, 1990b) have also used a sequential linearization approach to solve the discrete optimization problems and have done a significant numerical testing.

By far, the most popular technique to tackle the discrete optimization remains the "Branch-and-Bound" algorithm. The traditional branch-and-bound method was proposed by Land and Doig (1960) for a linear programming problem. Later researchers (Dakin, 1965; Martin et al., 1985) have suggested modifications to improve the efficiency and storage requirements for the tree-like search method. Mitra (1973) and Barki (1976, 1977) have employed mathematical heuristics to speed up the branch and bound search.

Though most of the research for branch-and-bound algorithm seems to be concentrated in the operations research area (Mitten, 1970; Myers, 1984), some researchers (Gupta, 1980; ...

During optimization, the use of AI has been suggested by using Monotonicity Analysis (Choy, 1986; Agogino and Almgren, 1987; Papalambros and Wilde, 1988). Monotonicity analysis can be used to determine active constraint sets while performing design optimization. Arora and Baenziger (1986) have suggested the use of production rules to determine suitable optimization techniques. Human intervention is also suggested to enhance the decision-making of the program. Glover (1986) has reviewed the use of Artificial Intelligence to tackle the integer programming problems and presents the advantages of new methods such as Simulated Annealing and Tabu Search.

2.3 Nonlinear Equations Solving

Simultaneous solution of non-linear equations has remained an active research topic in a number of fields. Newton's method has been used in the majority of cases since it exhibits quadratic convergence and is easy to program on a computer. However, Newton's method requires a reasonably good initial estimate for the solution procedure to converge. A significant number of other methods have been proposed and most of these have now been subjected to rigorous testing. Five other methods have been studied in the proposed research. An optimization-based method such as Powell's Hybrid method (Powell, 1970) has been shown to be quite successful for a number of numerical test problems. So does the method based on nonlinear least squares such as Levenberg-Marquardt method. Another method based on the Gaussian elimination of the Jacobian is discussed by Brown (1969). Brent (1973) has proposed a method similar to Brown's but his algorithm relies on orthogonal triangularization. Secant method with Broyden forward Jacobian update is another method that shows quadratic convergence and is easy to implement on a computer (Dennis and Schnabel, 1983).
Some researchers have tested the performance of nonlinear solvers and discussed the merits and limitations (Brent, 1973; Moré and Cosnard, 1979; Hiebert, 1982). However, in the past, most of the performance testing of these methods is done on mathematical problems. Though such test problems help in evaluating the strengths and weaknesses of a particular method, they are not necessarily representative of the everyday design problems. Recently, some testing of the nonlinear solvers has also been done for mechanical design problems (Padmanabhan, 1985; Agrawal and Kinzel, 1991).

A number of other methods that have been reported in literature but they have been excluded in the study. Some of these methods are Shamanskii’s (Brent, 1973), Method of steepest descent (Broyden, 1970), Method with Traub Iteration function (Traub, 1964), Secant method with Barnes update (Broyden, 1970), the Secant method with Broyden inverse update (Broyden, 1970), and weighted simplex method (Price, 1979). These methods have not been used extensively by the scientific community, and do not seem to have generally better characteristics than the methods chosen for this study.

2.4 Gear Design

A proper selection of gears has always been an important part in the design of a machine assembly. In the past, some researchers have looked into the computer-aided design of gears. Amongst the early works, Kinsman (1955) presented a comprehensive analytical design procedure for nonstandard involute spur gears. Similar work was done by Genson (1960) who presented a step-by-step procedure to determine gear load ratings and drawing dimensions for various gear applications. Though, most of his work is relevant for analysis, he has also studied long and short addendum designs. Willis (1963) analyzed different compound gear trains and presented charts and equations for minimum weight designs. Kron (1972) presented excellent guidelines for the design of spur and helical gears. Kron analyzed the effects of various gear geometry parameters to suggest an optimum design.

Merticaru and Atanasiu (1979) presented an optimum design of rack shift coefficients to realize gears with low tooth numbers. Tucker (1980) presented a thoughtful overview on the complete design process. His work provides a number of guidelines for synthesizing a
gear unit for a new or unique application. Hughson (1980) described a gear optimization system called GODA5 which was aimed mainly at tractor and automobile gears. Estrin (1980) presented an optimization method for the gear mesh parameters. Different objectives such as the minimization of root compressive stress, maximization of contact ratio, or maximization of tip thickness were considered by using a Sequential Linear Programming method.

Buchhorn et al. (1981) devised an optimization technique to design spur and helical gears based on the Australian gear standard AS B61. Gears with high addendum modification coefficients were also analyzed. Savage et al. (1982a; Savage, 1984) have suggested a concise procedure based on optimization considerations. Savage used empirical chart values for the computation of geometry factor (J) to simplify the gear bending strength calculation. Both external and internal spur gear sets were considered in their research. Later, Savage et al. (1982b) extended their previous work to the optimal design of non-standard spur gear sets. Optimal calculations of the tool shift were done to design long and short addendum gear sets.

Herscovici (1983) provided essential design formulae for spur and helical gears. Arabyan et al. (1984) described a microcomputer-based method for the design of hobbed standard spur gears. Carroll (1984) and later Carroll and Johnson (1984) extended Savage's strategy to incorporate geometry factor and dynamic factor calculations to design minimum size gear sets. In a later work, Carroll and Johnson (1989) showed that the optimum value of the gear geometry was independent of the load and speed requirements. Chapman (1984) suggested a method of gear design using Box's method of optimization. Zarak (1985) developed an optimization method to design minimum weight spur gears with a 25° pressure angle. He considered all the AGMA factors in the model and three modes of failure, namely, bending, pitting and scoring, were considered in his design model. Cardillo et al. (1985) used the number of pinion teeth and helix angle as the independent parameters to generate optimal helical gears based on AGMA 218.01 standards. Colletti (1985) extended the work done by Carroll to helical gears.

Agrawal (1986) and later Agrawal and Kinzel (1987) used the complete AGMA standards in formulating an optimization scheme for optimal design of gears. Several different optimization methods were explored and it was found that Sequential Linear Programming
was the best for gear design. Chicurel and Escheverria-Villagomez (1987) formulated an optimization problem for maximizing the contact ratio of helical gear sets. They used the R. C. Johnson's (1979) method to determine the intersection points of various constraints. Gitchel (1987) suggested the use of expert systems approach to gear design to alleviate the burden on the designer to supply a number of factors required by the AGMA standards. Yelle and Gauvin (1988) presented a computer program written to compute all the AGMA correction factors used in the standard.

Janninck (1987, 1988) presented an enumerative procedure to design spur gear sets. A computer program was described that takes in a fixed number of specifications and computes several alternative designs. Prayoonrat and Walton (1988) described a heuristic procedure to design a coaxial double speed reduction gear train with the objective of minimizing the center distance. Vanderplaats et al. (1988) used a standard optimization package to design spur gear sets. Several different objective functions such as maximization of gear life and minimization of dynamic load are considered in their study. A nonlinear optimization technique was used by Onwubiko (1989) to design spur gears with minimum tooth deflection. A Flexible Tolerance method was used in his study. Errichello (1989) presented a closed-form procedure to design minimum weight spur and helical gears. The three modes of failure, namely, bending fatigue, Hertzian contact stress and scoring were considered.

The K-factor method suggested by Dudley (1984) was used by Zarefar and Lawley (1989) to design minimum size spur gears. Their approach is very similar to an expert system methodology developed by Lin and Johnson (K. C. Lin, 1988, 1989). Standard helical gear design was presented by Jog and Pande (1989) with the objective of minimizing the center distance. They used an interior penalty function approach to satisfy the kinematic and stress constraints used in their model. Rogers et al. (1990a, 1990b) developed the mathematical basis for designing spur gears cut by hobs and shapers. Gear backlash was also incorporated in their design model to design both standard and nonstandard spur gears.
Chapter III

CONSTRAINT MANAGEMENT

3.0 Introduction

As mentioned in Chapter I, the mathematical model for a given mechanical system can be characterized by \( m \) nonlinear equalities and \( p \) inequalities in \( n \) unknowns:

\[
\begin{align*}
  h_i (x_1, x_2, \ldots, x_n) &= 0, & i &= 1, 2, \ldots, m \\
  g_i (x_1, x_2, \ldots, x_n) &\leq 0, & i &= 1, 2, \ldots, p
\end{align*}
\] (3.1) (3.2)

In addition, there might be limit constraints:

\[
x_l \leq x \leq x_u
\] (3.3)

where some of the limits could be \(-\infty\) or \(+\infty\) signifying that the variable is unbounded in a certain direction.

If the inequality constraints are converted to equality constraints using \( p \) non-negative slack variables, the total number of design variables increases from \( n \) to \( n + p \). Thus, in general, we have the following formulation:

\[
\begin{align*}
  f_i (x_1, x_2, \ldots, x_N) &= 0, & i &= 1, 2, \ldots, M \\
  x_i^l &\leq x_i & x_i^u \quad &i &= 1, 2, \ldots, n \\
  x_i &\geq 0 & \quad &i &= n+1, n+2, \ldots, N \quad \text{(slack variables)}
\end{align*}
\] (3.4)
where \( N = n + p \) and \( M = m + p \).

Depending on the values of \( M \) and \( N \), we can classify a design system into one of the following three categories:

- Case 1: \( M < N \) : Under-determined system
- Case 2: \( M = N \) : Fully determined system
- Case 3: \( M > N \) : Over-determined system

Most representations of mechanical design problems are under-determined, and they exhibit \((N-M)\) degrees of freedom. These degrees of freedom translate into \((N-M)\) input variables before the design is complete. The input variables can be the design specifications e.g. load requirement, safety factor, etc. or decision variables selected by an optimization algorithm, e.g. wire diameter, number of coils, etc. In this thesis, we shall restrict our discussion to under-determined systems.

**Definition 3.1**: A design case is defined as a set of \((N-M)\) input variables. A valid design case is a case which has a set of \((N-M)\) input variables such that \(M\) variables can be computed.

Typically, each equation does not contain all of the design variables, so that only certain combinations of variables can be input. In a fully determined case, when \( M \) is equal to \( N \), there is only one possible design case. But, in general, the number of possible cases, \( A_t \), can be computed as the total number of combinations of \( N \) variables taken \((N-M)\) at a time, given as follows:

\[
A_t = \binom{N}{N-M} = \frac{N!}{(N-M)! \, M!}
\]  

(3.5)

In most design problems, the number of valid cases is smaller than \( A_t \) because certain combinations of variables will lead to direct redundancy problems.

**Example**:

Let us take a simple example to illustrate the concept of direct redundancy.
\[ x_3 = x_1 + x_2 \]
\[ x_4 = x_3^2 \]

Since there are 4 variables in 2 equations, \( A_t = \binom{4}{2} = 6 \) cases. But the case which selects \((x_3 \text{ and } x_4)\) as input variables leads to direct redundancy. Thus there are only 5 valid design cases.

In a simple constraint network, as the one above, direct redundancies are easy to identify. But, when the size of the constraint network increases, determination of direct redundancies becomes a complex problem. Interactive design programs built over the constraints of a mechanical system should permit the user to input any valid combination of design variables. Thus, we need to determine which variables are dependent upon the known variables, and to determine the most efficient solution scheme to solve for these dependent variables.

Assuming that several variables have already been input and other variables have been solved, an interactive constraint manager has to deal with the following situations:

1. **New Specification Problem**: Input of any unknown variable. Determine if there are any dependent variables and solve them.
2. **Respecification Problem**: Change in the value of the variable which has already been input.
3. **Unspecification Problem**: Deletion of the value of a variable which was previously specified as an input.
4. **Reverse Specification Problem**: Input of a value for a variable which has been computed.

Each of these situations can be handled since the structure of the equations is assumed known, and each is addressed separately in this chapter. Section 3.1 deals with the problems of constraint representation, and explains the various matrices and vectors used to capture an interactive design. Section 3.2 presents all the constraint management (CM) algorithms. Section 3.3 explains the concepts of hidden constraints, constraint conflict, and redundancy. Section 3.4 deals with the development of an optimum formulation of
constraints. Section 3.5 presents a complete example of an engineering design using the concepts developed in this chapter.

3.1 Constraint Representation

In order to store the information on the constraints, we need a few matrices and some vectors. A nonlinear occurrence matrix is used to store the structure of the equations. Also a BLOCK vector is used to differentiate disjoint sets among the constraints. LEVEL and ORDER information is used to store the solution sequence.

3.1.1 Nonlinear Occurrence matrix

The structure of a set of algebraic equations can be described in a precise manner either by an undirected graph or a Boolean matrix. Although, the literature shows that a number of the past researchers have used directed graphs, the Boolean matrix approach offers a compact representation which is mathematically simple and easy to implement (Friedman and Leondes, 1969a). The matrix is called an occurrence matrix. Several different names for occurrence matrix such as structural matrix, incidence matrix, dependency matrix can be found in literature.

The elements of an occurrence matrix [A], which is an (M x N) matrix, are defined as follows:

\[ A_{ij} = 1 \quad \Rightarrow \text{Variable } x_j \text{ appears in equation } f_i \]
\[ A_{ij} = 0 \quad \Rightarrow \text{Variable } x_j \text{ does not appear in equation } f_i \]

The elements of the occurrence matrix can be determined by examining the equations symbolically and determining if a variable is present in an equation. Alternatively, a design variable \( x_j \) is present in \( f_i \), if for distinct values of \( x_j \), the constraint function value is also distinct. Mathematically,

\[
A_{ij} = \begin{cases} 
1 & \text{if } f_i(x_1, x_2, ..., x_{j-1}, x'_j, x_{j+1}, ..., x_n) \neq f_i(x_1, x_2, ..., x_{j-1}, x_j, x_{j+1}, ..., x_n), \quad x'_j \neq x_j \\
0 & \text{otherwise}
\end{cases}
\]
The numerical scheme can fail if by sheer coincidence $f_i(x_j^*) = f_i(x_j)$. We could use the minimum and maximum values of $x_j$ in the above formula to determine $A_{ij}$. In the current research, we have used the symbolic approach to determine the occurrence matrix.

Fig 3.1 shows the constraint network and the corresponding occurrence matrix for a set of constraints taken from gear analysis. As can be seen, the occurrence matrix for the constraint representation is quite sparse. This is generally true for a large number of design problems.

Since we are dealing with nonlinear constraints, the occurrence matrix information can be enhanced by including a nonlinear term instead of just 0 or 1. The resulting matrix is called a nonlinear occurrence matrix and is denoted by $[B]$. Thus,

$$B_{ij} = A_{ij} + \text{(degree of nonlinearity)}$$

Some guidelines for the entries for $[B]$ were suggested by Solyemez and Seider (1973). The degree of nonlinearity is somewhat subjective, but it is generally equal to the 0 if $x_j$ appears linearly in equation $f_i$. In other cases, a higher degree is assigned depending upon the nature of the variable. $B_{ij} = 2$ when $x_j$ is moderately nonlinear in equation $i$ and 3 or higher when $x_j$ is very nonlinear. Also, if $A_{ij}$ is zero, then the corresponding $B_{ij}$ must also be zero. Therefore, if all of the equations are linear in the variables, the matrix $B$ will be equal to $A$.

Fig. 3.1 also shows the constraints expressed in a generic form with all the denominator terms removed.
Constraints

\[ f_1: \quad m_G = \frac{N_G}{N_p} \]
\[ f_2: \quad \tan \phi_i = \frac{\tan \phi_n}{\cos \psi} \]
\[ f_3: \quad P_i = P_n \cos \psi \]
\[ f_4: \quad d = \frac{N_p}{P_i} \]

Constraints in a generic form

\[ f_1: \quad x_1 x_2 - x_3 = 0 \]
\[ f_2: \quad \tan(x_4) \cos(x_6) - \tan(x_7) = 0 \]
\[ f_3: \quad x_5 - x_4 \cos(x_6) = 0 \]
\[ f_4: \quad x_5 x_9 - x_2 = 0 \]

Constraint Network

Occurrence matrix

<table>
<thead>
<tr>
<th></th>
<th>(x_1 = m_G)</th>
<th>(x_2 = N_p)</th>
<th>(x_3 = N_G)</th>
<th>(x_4 = P_n)</th>
<th>(x_5 = P_t)</th>
<th>(x_6 = \psi)</th>
<th>(x_7 = \phi_n)</th>
<th>(x_8 = \phi_t)</th>
<th>(x_9 = d)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(f_1)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(f_2)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>(f_3)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>(f_4)</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 3.1: A typical set of constraints for gear design
The nonlinear occurrence matrix can be given as:

<table>
<thead>
<tr>
<th></th>
<th>$x_1=m_G$</th>
<th>$x_2=N_P$</th>
<th>$x_3=N_G$</th>
<th>$x_4=P_n$</th>
<th>$x_5=P_l$</th>
<th>$x_6=\Psi$</th>
<th>$x_7=\phi_n$</th>
<th>$x_8=\phi_l$</th>
<th>$x_9=d$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$f_2$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>$f_3$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$f_4$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

If we assume that all the inequalities are placed at the end of the equalities, then the $[B]$ matrix will be composed of four sub-matrices $[H]$, $[0]$, $[G]$, and $[I]$ and will have following form:

$$
\begin{bmatrix}
[H] & [0] \\
[G] & [I]
\end{bmatrix}
$$

The matrix $[H]$ represents the $m$ equalities and is an $(m \times n)$ matrix. Its elements are defined as follows:

$$
H_{ij} = 1 \Rightarrow \text{Variable } x_j \text{ appears in the equality } h_i \\
H_{ij} = 0 \Rightarrow \text{Variable } x_j \text{ does not appear in the equality } h_i
$$

The matrix $[G]$ represents the $p$ functional inequalities and is a $(p \times n)$ matrix. Its elements are defined as:

$$
G_{ij} = 1 \Rightarrow \text{Variable } x_j \text{ appears in the inequality } g_i \\
G_{ij} = 0 \Rightarrow \text{Variable } x_j \text{ does not appear in the inequality } g_i
$$

The matrix $[0]$ is the $(m \times p)$ null matrix signifying the absence of slack variables in any of the equality constraints. Finally, $[I]$ is the $(p \times p)$ identity matrix denoting the presence of a unique slack variable for each of the $p$ inequalities.
3.1.2 Design State

At any design state, we have a certain number of input variables, output variables and unknown variables. Using terminology from the Generalized Reduced Gradient (GRG) method (described in Chapter 5), the input variables are also referred to as decision variables or non-basic variables. Similarly, output variables are also called state variables or basic variables. In constraint management literature, input variables are also called independent variables, and output variables as dependent variables.

The class of unknown variables is special to an interactive design process where there could be variables have not been assigned to either the input or the output class.

If $X$ denotes the set of all the variables, we have,

$$X = Z \cup Y \cup U$$

where,

$Z$ set of input (decision, nonbasic, or independent) variables

$Y$ set of output (state, basic, or dependent) variables

$U$ set of unknown variables

Since the set of input and output variables is disjoint, $Z \cap Y = \emptyset$ (empty set), we have,

$$U = X - (Z \cup Y)$$

The design case is complete when, $U = \emptyset$ i.e. there are no unknowns left.

An IO vector is used to store the information about the state of a particular variable. The values of the vector are defined as follows:

$I0(x_i) = 0$ if $x_i \in U$.

$I0(x_i) = 1$ if $x_i \in Z$.

$I0(x_i) = -1$ if $x_i \in Y$. 
Example:

Let there be 2 equations in 4 unknowns and let us say that $x_3$ has already been input.

\[ x_3 = x_1 + x_2 \]
\[ x_4 = x_3^2 \]

Since we can solve $x_4$ using equation 2, we represent the current status for the variables as:

\[ Z = \{x_3\} \]
\[ Y = \{x_4\} \]
\[ U = \{x_1, x_2\} \]

3.1.3 Block Information

In multi-component mechanical design, it is possible to have disjoint blocks of equations and variables. In such a case, one group of equations and variables may be completely independent of the others. Computationally, it is efficient to consider these independent groups separately. The block information can be determined from the occurrence matrix prior to any actual design calculations. This can be done in a variety of ways, and one possible procedure is given in Fig. 3.2. The algorithm starts with the variable $x_1$ and finds all of the equations involving it. The other variables in these equations are then identified, and all of the equations involving these variables are identified. The new set of equations are then checked for new variables, and any additional equations which contain these new variables are identified. The procedure continues until no more new variables or equations are found. These equations and variables form the first block.

The algorithm then checks if all of the variables in the original formulation have been identified. If not, the variables and equations for the second block are identified. The procedure continues until all variables and equations have been included in one of the blocks. The block information for variables and equations is represented as the BLOCK vector.
Fig. 3.2: A Depth First Procedure for Identifying Disjointed Blocks of Equations
3.1.4 Level Information

In a typical design problem, the variables can be specified as inputs in any order. After a sufficient number of variables are input, some of the outputs can be computed. The level information indicates at what stage of the design process, input variables are used and output variables are computed. The current level number remains zero until some equations can be solved.

As soon as it is detected that some variables can be solved, these output variables and the corresponding equations are given level 1. The input variables in the equations solved are also assigned level 1. The second level includes the output variables and the new input variables (i.e. input variables with their level=0) which are required to compute the second group of outputs. Subsequent levels are defined in a similar manner. It should be noted that for the input variables, the level number indicates the first time they were used. Even if the input variables are respecified at a later stage, the level number remains unchanged for them. The level information for variables and equations is stored in the LEVEL vector.

3.1.5 Order Information

The order information gives the order in which the equations are solved and the variables determined. If several equations are solved simultaneously, the equations and variables involved are of the same order designation. The order index is specific to a particular level i.e. at a particular level there might be more than one order. The order information indicates the way in which equations are solved at a particular level. Since the input variables are not part of the equation solution process, their order is always set to zero. The order information for variables and equations is stored in the ORDER vector.

3.1.6 Solution Sequence

The relationship between output variables and input variables is given by a solution sequence.
**Definition 3.2:** A solution sequence is a collection of ordered sets of equations and variables which describes a systematic way to compute output variables from the input variables.

The solution sequence is denoted as \( \Omega \). At any stage of the design, the solution sequence can be determined by collecting equations and variables of the same order in an ascending manner.

To represent the solution sequence in a symbolic form, we will use the square brackets \([\) to separate the different levels. Curly braces \(\{\) will be used to identify a particular order inside a level. Brackets \((\) will denote groups of equations and variables that are solved simultaneously.

**Example:**

Consider the following set of constraints

\[
\begin{align*}
    f_1: & \quad x_1 + x_2 + x_3 = 0 \\
    f_2: & \quad x_2 - x_3 - 15 = 0 \\
    f_3: & \quad x_4 + 2x_5 = 0
\end{align*}
\]

Let us say the input sequence is \( x_4 \) followed by \( x_1 \). Then the solution sequence can be represented as:

\[
\Omega = [(f_3, x_5)] [(f_1, f_2), (x_2, x_3)]
\]

which implies that “Solve equation \( f_3 \) for variable \( x_5 \); then solve equations \( f_1 \) and \( f_2 \) simultaneously to find \( x_2 \) and \( x_3 \).”

The only information missing in the above solution sequence is the place where the input variables get activated. Once, that information is added, we can represent the whole design state symbolically. We indicate the inputs just after the left square bracket for each level. Thus, for the above example, we get:
\[ [x_4, \{f_3, x_5\}] [x_1, \{(f_1, f_2), (x_2, x_3)\}] \]

### 3.2 Constraint Management Algorithms

In interactive design programs which permit the user to input any valid combination of design variables, there is the need to determine which variables are dependent on variables already input or computed. Once the dependent variables have been identified, it is necessary to determine the most efficient sequence for computing these variables, and once the dependent variables are computed, changes in both the input and output variables must be accommodated. This indicates the need for algorithms which will determine the following:

a) The most efficient solution sequence for the dependent variables (Design decomposition)
b) Output variables dependent on a given input variable (Forward dependency)
c) All input variables affecting a given output variable (Backward dependency)

#### 3.2.1 Design Decomposition Algorithm (DDA)

When the value for a particular variable $x_k$ becomes known (i.e., it is either input or computed), there is a need to identify any dependent variables that can be solved using the constraint equations. This requires an algorithm that will systematically cycle through the remaining equations to determine if any equation can be solved individually or in a set of simultaneous equations. Such an algorithm is given in the following.

For any known variable, $x_k$, the $k$th column of the occurrence matrix is zeroed, i.e.,

\[ A_{ik} = 0 \quad i = 1, 2, \ldots, M \]

This process is referred to as updating the occurrence matrix. The input process is considered complete when all variables have been either specified (input) or computed (output), i.e., the occurrence matrix is identically zero.
If the occurrence matrix is not zero, then the entries in each row of the occurrence matrix are summed in a vector \textbf{ROWSUM}, i.e.

\[
\text{ROWSUM}_i = \sum_{j=1}^{N} A_{ij}
\]

If the \textbf{ROWSUM} vector is equal to 0, then there are no more variables remaining, and the design is complete.

To find equations which can be solved individually, the procedure is quite trivial. A simple check is made to see if one or more entries in the \textbf{ROWSUM} vector are equal to 1. These are recorded and the corresponding entries in \([A]\) are also zeroed out.

Figure 3.3 shows the block diagram involved in finding the variables which have to be solved using simultaneous equations. The algorithm involves the following four stages.

Stage 1: Determine an active occurrence matrix \([A^*]\) from the current \([A]\).
Stage 2: Find a square matrix emanating from one of the rows of \([A^*]\). Call this \([A_3]\).
Stage 3: Determine a row permutation, \([P_1]\), to convert \([A_3]\) to a matrix with a zero-free diagonal.
Stage 4: Find the symmetric permutation which changes \([P_1A_3]\) into a block triangular form.

\[
C = PA_sQ = Q^T(P_1A_s)Q = \begin{bmatrix}
C_{11} & 0 \\
C_{21} & C_{22} \\
C_{31} & C_{32} & C_{33} \\
\vdots & \vdots & \vdots & \ddots \\
C_{r1} & C_{r2} & C_{r3} & \cdots & C_{rr}
\end{bmatrix}
\]

Then the simultaneous sets of equations (strong components) can be read off as the blocks of \([C]\). If \(r = 1\), only one block of equations is found, and the matrix \([A_3]\) is said to be irreducible. In general, for \(r > 1\), \(C_{11}\) contains the equations and variables that need to be
solved simultaneously as the first set, $C_{22}$ contains variables and equations in the second set, and so on.

Stages 1 and 2 of DDA are referred to as the coarse decomposition phase. Stages 3 and 4 are termed as the fine decomposition phase.

**Note 1:** If no $[A_8]$ can be found from $[A^*]$, then DDA is terminated.

**Note 2:** Whenever the number of unsolved equations is equal to the number of unsolved variables, $[A_8]$ is immediately determined. In this case Stage 1 and 2 of the algorithm are skipped. We go directly to the fine decomposition.

**Note 2:** If $[A_8]$ is found and then subsequently block triangularized, $[A]$ is updated and the DDA is repeated from the beginning.
Each of these stages are described in detail in the following sections. An example is given later in the chapter to illustrate the different stages.

3.2.1.1 Stage 1: Determining $[A^*]$ from $[A]$  

The first stage of the Design Decomposition Algorithm is to extract an active occurrence matrix from the current $[A]$. An Active Occurrence Matrix $[A^*]$ consists of active equations and active variables.

**Definition 3.3:** An Active Equation is defined as an unsolved equation containing at least one known variable.

$$AE = \{ f_i \mid \sum_{k=1}^{N} A_{ik} \neq 0 \land \exists j \ (A_{ij}^o \neq 0 \land x_j \in U) \}$$

where $[A^o]$ is the original occurrence matrix without any updation and $U$ is the set of unknown variables. An active equation is an equation that has a potential of getting solved since one of its defining variable is known.

An easy way to find an active equation is to compare the ROWSUM of an equation in $[A^o]$ and $[A]$. The ROWSUM value reduces because as soon any variable becomes known, its column is zeroed out and the equation becomes active. A ROWSUM value of zero indicates the equation has been solved and thus is not active.

Mathematically,

$$AE = \{ f_i \mid 0 < \sum_{j=1}^{N} A_{ij} < \sum_{j'=1}^{N} A_{ij}' \}$$

**Definition 3.4:** An Active Variable is defined as any unknown variable appearing in the active equations.
AV = \{ x_j \mid x_j \in U \land \exists i (A_{ij}^* \neq 0) \}

It should be noted that as the design proceeds the dimensions of \([A^*]\) will change. In the beginning the \([A^*]\) will be small and will then grow until some variables can be solved. Typically, near the end of the process after some of the equations are solved, \([A^*]\) will reduce in size.

3.2.1.2 Stage 2: Determining \([A_5]\) using \([A^*]\)

Since we are dealing with under-determined systems (i.e. \(M < N\)), we can safely assume that any set of solvable simultaneous equations will emanate from an active equation. If this were not true, then we will have sets of equations that are fully determined before the design begins. Such sets can be identified in the beginning and are excluded from interactive design.

Hence, we need an algorithm that will examine every active equation and check to see if it can lead to a simultaneous set. The following algorithm is used to determine such a set and the corresponding matrix is extracted in \([A_5]\). A temporary copy of the current \([A]\) is made for the algorithm.

Step 0: Select the first active equation.
Step 1: Assign zero to NOEQNS.
Step 2: Examine the active equation and add all the new variables present in the equation to a candidate list.
Step 3: Increase the NOEQNS counter by the number of new variables.
Step 4: Zero out the columns of \([A]\) for the new found variables.
Step 5: Check to see if the number of zero ROWSUM entries is equal to NOEQNS. If so, then \([A_5]\) has been found. End the algorithm.
Step 6: If not, continue to add an equation that has the minimum ROWSUM among the unsolved equations and has at least one variable from the candidate list. If all the equations have been exhausted, then move on to the next active
equation and start from Step 1. If all active equations has been examined, then there are no solvable simultaneous sets. Terminate the algorithm.

Step 7: Add all the new variables in the last picked equation to the candidate list. Go back to Step 3.

If a matrix $[A_s]$ was found, it is possible that it can be decomposed further since the above algorithm cannot ensure the finest possible decomposition. All the equations and variables associated with $[A_s]$ are assigned the current level number in the LEVEL vector. Any input variables occurring in these equations which have a level value of zero are also assigned the current level number.

3.2.1.3 Stage 3 : Permuting $[A_s]$ to a Zero-free Diagonal Form

Before a fine decomposition of $[A_s]$ can be accomplished, it is necessary to permute the matrix such that it has no zero diagonal entry. Such a permutation is also referred to as an assignment problem, maximum matching or transversal form (See Duff, 1981; 1988). We use algorithm MC21A from the Harwell subroutine library to achieve the maximum transversal. A complete description of the algorithm can be found in (Duff, 1986). The permutation matrix is stored in $[P_1]$.

Once, the matrix $[A_s]$ is in the zero-free diagonal form, there is at least one designated output variable for each equation. If the matrix cannot be permuted, then there is some singularity in the constraint model.

3.2.1.4 Stage 4 : Permuting $[P_1A_s]$ to a Block Triangular Form

To convert a matrix to a block triangular form, an excellent algorithm was presented by Tarjan (1972). An implementation of the Tarjan's algorithm is given by Duff (1978, 1986). We use the subroutine MC13D from the Harwell library to find the symmetric permutation to the block lower triangular form.
3.2.2 Forward Dependency Algorithm (FDA)

Forward dependency for a given input variable refers to the set of its dependent output variables. Serrano (1987) refers to it as causality. Forward dependency algorithm is used in an interactive design process when the user wishes to respecify (change) or unspecify (delete) the value of an input variable.

Mathematically, we determine:

\[ \mathcal{F}(x_i) = \{ y \mid (x_i \in U \Rightarrow y \in U) \land (x_i \in Z \Rightarrow y \in Y) \} \]

Here \( \mathcal{F}(x_i) \) represents all the variables dependent on the value of \( x_i \) for their existence. If \( x_i \) is undefined, then all the elements of \( \mathcal{F}(x_i) \) are also unknown.

An immediate corollary of the above definition is that

\[ \mathcal{F}(x_i) \subset Y \]

since by definition all the dependent variables belong to the output set.

\( \mathcal{F}(x_i) \) can also be an empty set if the value of \( x_i \) has not been used yet to compute anything.

It is tempting to define the forward dependency of a variable in the following recursive way:

\[ \mathcal{F}(x_i) = (\text{direct outputs}) \cup \mathcal{F}(\text{direct outputs}) \]

But the above definition has a serious flaw. It should be noted that all output variables that are connected to an input variable (direct outputs) might not depend on the existence of the input variable for their value. Consider the following three constraints:

\[ f_1: \ x_3 + x_4 + x_5 = 0 \]
\[ f_2: \ x_5 + x_6 = 0 \]
\[ f_3: \ x_1 + x_2 + x_3 = 0 \]
Let us assume that the sequence of input is $x_4$, $x_1$ and then $x_2$. Then, the solution sequence would be,

$$\Omega = \{ \{ f_3, x_3 \}, \{ f_1, x_5 \}, \{ f_2, x_6 \} \}$$

Now, we need to find, $\mathcal{F}(x_4)$. Since, all the equations are solved at the same level, all the output variables are candidates for $\mathcal{F}(x_4)$. But a simple observation reveals that $x_3$ does not depend on $x_4$ for its existence. The simplest way to determine that is to make the status of $x_4$ as unknown and perform a design decomposition with just $x_1$ and $x_2$ as inputs. The new solution sequence is,

$$\Omega' = \{ \{ f_3, x_3 \} \}$$

Therefore,

$$\mathcal{F}(x_4) = \{ x_3, x_5, x_6 \} - \{ x_3 \} = \{ x_5, x_6 \}$$

Thus, to determine $\mathcal{F}(x_i)$, define

$$Z^* = Z - \{ x_i \}$$

Find $Y^*$ for $Z^*$ using the DDA. Then,

$$\mathcal{F}(x_i) = Y - Y^*$$

Since, the design state stores the current level and order information for the variables, we can increase the efficiency of the above computation. We first determine the block and level of the input variable $x_i$. Then, a sub-occurrence matrix is formed from the equations that are in the same block and at the same level or higher than $x_i$. $x_i$ is then removed from the set of the inputs and the design decomposition algorithm (described in Section 3.2.1) is invoked to determine the output variables which can still be solved. Such output variables are not dependent on $x_i$. The rest of the output variables, having level equal or higher than $x_i$, belong to $\mathcal{F}(x_i)$. 


To illustrate the use of level numbers, let us take the previous example and try to find \( \mathcal{F}(x_4) \). Let us say the input sequence in the above example is changed to \( x_1, x_2 \) followed by \( x_4 \). Now we would have the design state split into two levels:

\[
[x_1, x_2, \{f_3, x_3\}] [x_4, \{f_1, x_5\}, \{f_2, x_6\}]
\]

In this case, we would not consider \( x_3 \), since it lies at a level lower than that of \( x_4 \). The DDA would be used to determine if any variables can be solved using equations \( f_1 \) and \( f_2 \) without the knowledge of \( x_4 \). The outcome of that would be an empty set and we will get the following result:

\[
\mathcal{F}(x_4) = \{x_5, x_6\} \cdot \{\} = \{x_5, x_6\}
\]

Section 3.2.2.1 describes the use of the FDA in the respecification problem and Section 3.2.2.2 describes the unspecification problem.

### 3.2.2.1 Respecification Problem

If the value of an input variable is changed, then all of the output variables which depend on the input variable must be recomputed. The dependent output variables can be identified using the FDA presented in the previous section. The rest of the output variables are retained at their previous values. The solution sequence to compute the dependent variables is found by subtracting the solution sequence for the non-dependent output variables from the original sequence.

### 3.2.2.2 Unspecification Problem

After a number of input variables are specified and some of the output variables are computed, it may be desirable to unspecify or delete the value for a given input variable (unforced unspecification). Another case of input variable unspecification occurs when an output variable is changed to an input status by fixing it to a specific value. When this
occurs, one of the original input variables must be unspecified (forced unspecification). Whenever an unspecification occurs, the values of all of the output variables which depend on the unspecified input variable are no longer known. The FDA is used to determine all such output variables. The status of the identified output variables is then changed to an unknown (free) state.

3.2.3 Backward Dependency Algorithm (BDA)

For a complicated problem, the designer may know typical values for the output variables, and he/she may find the computed values to be unacceptable. One option in such cases is to specify the value for one of the variable which was calculated (Reverse Specification Problem). When this is done, the problem becomes over-determined, and it is necessary to unspecify one of the input variables (forced unspecification). It is therefore necessary to determine those input variables which influence the given output variable (backward dependency). Usually, this will be some subgroup of the original group of input variables.

Backward dependency problem is almost the inverse of forward dependency. Here we are interested in finding out all the variables that affect a given output variable. Although the variables that form this set might be both input and output, we are interested in only the input variables for the reverse specification problem.

Thus, in a general case:

Given \( y_i \in Y \) and the current design state

Find:

\[ \mathcal{B}(y_i) = \{ x \mid (x \in Z) \land (y_i \in \mathcal{F}(x)) \} \]

An immediate corollary of the above definition is that

\[ \mathcal{B}(y_i) \subseteq Z \]
Note that unlike $\mathcal{F}(x_i)$, $\mathcal{B}(y_i)$ cannot be an empty set since we assume that none of equations can be solved without any input. This is also a direct result of the fact that all solution sets emanate from active equations.

A brute force method to determine $\mathcal{B}(y_i)$ would be to check all the input variables and see if $y_i$ belongs to their forward dependency set. But such a check would take a long time since the determination of $\mathcal{F}$ uses the DDA. However, a simple observation helps us to simplify the problem.

$$\mathcal{B}(y_i) = \text{(direct inputs)} \cup \mathcal{B}(\text{direct outputs})$$

In the above definition, direct inputs and direct outputs refer to the variables that share common equations with the specified variable $y_i$. The definition is recursive and is analogous to reverse tree traversal. Since, a variable can have share equations with a number of direct outputs, a stack is maintained for the direct outputs which have not been explored. Care has to be taken to avoid an infinite loop and always check any new direct variable before addition to the set i.e. automatically the union of sets is performed. The procedure terminates when no new variables are added to the backward dependency set and there are no output variables on the unexplored stack.

As was the case in the forward dependency algorithm, the efficiency of the above algorithm can be increased by using the LEVEL and ORDER vectors. To determine the backward dependency, the level and order of the output variable is determined. Only those equations at the same order or lower and those input variables at the same level or lower need to be considered to determine direct inputs and direct outputs. If the variables and equations which have not been used yet are designated to be at the zero level, any inactive equations will automatically be excluded from consideration.

A procedure for finding the backward dependency is shown in Fig. 3.4. First the equations which are at the same level and order as the designated output variable($y_i$) are found, and the input variables in these equations are identified. If these are all of the inputs at the same level or lower, the procedure terminates. To identify all of the input variables which potentially influence the designated output variable, indirect dependencies also must be checked. To determine indirect dependencies, the other output variables in the group of
Fig. 3.4: Backward dependency algorithm
equations being investigated are identified. The level and order of these variables are determined, and variables that have a level and order strictly less than \( y_i \) are explored in exactly the similar fashion. The algorithm terminates if any of the following conditions are met:

1) All the input variables that have a level value equal to or lower than \( y_i \) have been marked.

2) There are no more direct outputs left to be explored.

The BDA is also used in a number of other places where the roles of the input and output variable have to be reversed. A prime example of this is when an inequality constraint becomes violated, and its corresponding slack variable is set to a value of zero. In such a case, one of the backwardly-dependent input has to be released (See Sridhar et al., 1991a, 1991b). A similar situation also occurs when a discrete output variable is not at its prescribed discrete value. The BDA is also a part of the basis interchange algorithm for the GRG method described in Chapter 5.
3.3 Hidden constraints, Inconsistency and Redundancy

Using the CM algorithms, the earlier sections showed that one can identify the variables that depend on input variables, and the solution sequence to solve for the dependent variables. This way direct redundancies in the input procedure can be eliminated. For example, if there are two equations:

Formulation I:
\[ f_1: x_3 = x_1 + x_2 \]
\[ f_2: x_4 = x_3^2 \]

It is clear that there are 2 equations in 4 variables and thus \( ^4C_2 = 6 \) possible input combinations. Then, the occurrence matrix approach will prevent the input case of \( x_3 \) and \( x_4 \). But what if the system is changed to:

Formulation II:
\[ f_1: x_3 = x_1 + x_2 \]
\[ f_2: x_4 = (x_1 + x_2)^2 \]

Now, it is not so obvious that \( x_3 \) and \( x_4 \) cannot be input together. In fact, the occurrence matrix approach will allow the user to input them and then try to solve \( f_1 \) and \( f_2 \) simultaneously for \( x_1 \) and \( x_2 \). There will not be any solution to the problem unless \( x_3 \) and \( x_4 \) are given values which satisfy the original \( f_2 \). Such a situation is quite common when a parametric system i.e. a system assuming the input of \( x_1 \) and \( x_2 \), is put inside a constraint management framework where the set of inputs is totally arbitrary. Such formulations will be referred to as hidden constraint problems. The only solution to the above problems is to use symbolic algebra and reduce the problem after every input. Numerically, this problem cannot be solved in an obvious manner. One has to identify these cases and provide logic to switch between the two formulations in case of special input combinations. An other alternative is to always use the formulation I.
Let us now examine as to what happens if the hidden constraint is added to the second formulation.

Formulation III:
\[ f_1: x_3 = x_1 + x_2 \]
\[ f_2: x_4 = (x_1 + x_2)^2 \]
\[ f_3: x_4 = x_3^2 \]

The number of possible input cases now becomes \( 4C_1 = 4 \). So, as soon as \( x_3 \) is input, \( x_4 \) gets computed. But, then both \( f_1 \) and \( f_2 \) are stating the same thing and the design decomposition algorithm will indicate to solve them simultaneously for \( x_1 \) and \( x_2 \). This will lead to a redundant constraint problem. The above problem will have a singular Jacobian and an infinite number of solutions. The design case will be complete only when some further information is provided, which in this case would be either \( x_1 \) or \( x_2 \). The redundancy problem can be resolved symbolically if the equations are simple enough and the superfluous constraint can be identified.

Another source of trouble for the occurrence matrix approach stems when there are inconsistent constraints. For example:

\[ f_1: x_1 - x_2 - 12 = 0 \]
\[ f_2: x_1 - x_2 - 14 = 0 \]

Now the occurrence matrix approach will try to solve for 2 equations simultaneously and fail. But such a situation is not apparent from just looking at the structure of equations. During, the solution process, the Jacobian for the system of equations will be singular.

To solve the redundancy and inconsistency problems, a numerical approach would be to evaluate the Jacobian for the system of equations for all possible cases of input and see if the Jacobian becomes singular. Such a processing can be done before the set of equations is plugged into the constraint management framework.
Although, the above examples are extremely simple, they illustrate the problems in the occurrence matrix approach. Sometimes, a relatively simple set of equations will exhibit these problems when the set of input variables is changed. Consider the problem of the torsion bar design:

\[
\begin{align*}
    f_1 &= x_6 - \frac{x_3 x_5}{x_2 x_8} = 0 \\
    f_2 &= x_1 - \frac{x_3 x_4 x_7}{2x_8} = 0 \\
    f_3 &= x_8 - \frac{x_4}{32} = 0 \\
    f_4 &= x_{11} - \frac{\pi x_4^2 x_5}{4} \\
    f_5 &= x_9 - \frac{x_3}{x_6} = 0 \\
    f_6 &= x_{10} - \frac{x_1}{x_6} = 0 \\
\end{align*}
\]

A cursory look at the system of equations does not reveal any possible problems. But upon close examination, it is observed that the input of \(x_2, x_5, x_8\) and \(x_9\) will lead to a singular Jacobian. This is because of the hidden constraint:

\[
f_7 = x_9 - \frac{x_2 x_8}{x_5} = 0
\]

which comes from the elimination of \(x_3\) and \(x_6\) from \(f_1\) and \(f_5\).

### 3.4 Determining an Optimum Form of the Equations

When we are concerned with a static set of equations, it is often advantageous to explore alternate formulations (Watton, 1990; Watton and Rinderle, 1990a, 1990b). The following outlines the problems with rearranging equations or introducing intermediate variables. The solution sequence changes and so do the dependencies between the variables.

Let us take some equations from the torsion bar design formulated in three different forms.

**Case 1:**

In its simplest form the equations for the design of a torsion bar spring can be written as follows:
Stress \((f_1)\):
\[ \tau = \frac{TDn}{2J} \]

Twist angle \((f_2)\):
\[ \theta = \frac{TL}{GJ} \]

Polar moment of inertia \((f_3)\):
\[ J = \frac{\pi D^4}{32} \]

There are eight variables \(\tau, G, \theta, D, n, T, L, J\) and 3 equations. Thus we can input 5 variables. Let the input sequence be \(\tau, T, n, G\) and \(\theta\). As soon as the first three variables \(\tau, T\) and \(n\) are input, we can solve for \(D\) and \(J\) by solving equations \(f_1\) and \(f_3\) simultaneously. Then the input of \(G\) and \(\theta\) leads to the solving of equation \(f_2\) for the last variable \(L\). Hence, the complete design state can be represented in the following compact form:

\[ [\tau, T, n, ((f_1, f_3), (D, J))] \]

\[ [G, \theta, (f_2, L)] \]

From the solution sequence, it is clear that the shear modulus \(G\) and twist \(\theta\) have no effect on the diameter of the torsion bar. The biggest trouble with the above representation is that when \(f_1\) and \(f_3\) are solved simultaneously, the equation solvers tend to converge to an extraneous root of \(D=0\) and \(J=0\). Though this is not an actual root, it shows up if \(f_1\) is rewritten as:

\[ 2J\tau - TDn = 0 \]

**Case 2:**

Let us now rearrange the equations as follows. In this case, we have used \(f_2\) to substitute for \(T\) in \(f_1\). Also, \(J\) has been eliminated from \(f_2\). \(f_3\) remains the same.
Stress ($f_1$):
\[ \tau = \frac{\theta D G n}{2 L} \]

Twist angle ($f_2$):
\[ \theta = \frac{32 T L}{\pi D^4 G} \]

Polar moment of inertia ($f_3$):
\[ J = \frac{\pi D^4}{32} \]

Now if we input the first three variables $\tau$, $T$ and $n$, we cannot solve for anything. Next we input $G$. Still nothing can be solved. Last we input $\theta$. This leads to simultaneous solving of $f_1$ and $f_2$ for $D$ and $L$. Then equation $f_3$ has to be solved for $J$. Hence the complete design state is:

$[\tau, T, n, G, \theta, \{(f_1, f_2), (D, L)\}, \{f_3, J\}]$

From this solution sequence, we cannot make the same conclusion as we did in Case 1. It looks like $G$ and $\theta$ do have an effect on $D$. Plotting a graph between $G$ and $D$, it does show that $D$ does not depend on $G$. Also, the sensitivity matrix entry for $\partial D/\partial G = 0.0$.

This case can in fact lead to problems of Jacobian singularity (pathological case) if instead of $G$ we input $D$. Assume the set of inputs now be $\tau$, $T$, $n$, $D$ and $\theta$. The equation $f_1$ then becomes:

\[ \frac{G \theta}{L} = 2\tau \frac{D n}{D n} \]

Equation $f_2$ also has a similar form:

\[ \frac{G \theta}{L} = 32T \frac{\pi D^4}{\pi D^4} \]
Thus, we must have:

\[
\frac{2\tau}{Dn} = \frac{32T}{\pi D^4}
\]

This is the extra equation that indicates that \( \tau, T, n, \) and \( D \) are not independent. As soon as any three are input (\( \tau, T, n \) in this case), we must compute the value of \( D \). Thus, the conclusion from case 1 get hidden and leads to problems later on. Numerically, it is tough to tackle the problems of hidden constraints. Once, the Jacobian becomes singular, one has to look at the equations and find out the hidden constraint. An algebraic manipulation of the constraints after every input might be the only solution.

**Case 3:**

Let the equations be arranged in the following way. Here we have just eliminated \( J \) from both \( f_1 \) and \( f_2 \). \( f_3 \) remains the same.

**Stress \((f_1)\):**

\[
\tau = \frac{16Tn}{\pi D^3}
\]

**Twist angle \((f_2)\):**

\[
\theta = \frac{32T L}{\pi G D^4}
\]

**Polar moment of inertia \((f_3)\):**

\[
J = \frac{\pi D^4}{32}
\]

Now if we input the first three variables \( \tau, T, \) and \( n, \) we can solve for \( D \) using equation \( f_1 \). Using the computed value of \( D \), we can solve for \( J \) using \( f_3 \). Then upon the input of \( G \) and \( \theta \), the last variable \( L \) can be computed using equation \( f_2 \). Hence the complete design state is represented as:
[τ, T, n, (f₁, D), (f₃, J)] [G, θ, (f₂, L)]

From this sequence we can again draw the same conclusion as we did in Case 1 i.e. G and θ have no effect on D. But in this case we have to solve for D using just one equation. This happens because of the fact that J has been eliminated from equation f₁.

Thus, out of the three different cases presented, Case 3 seems to be the best representation. Case 1 suffers from the problem of an extraneous root, and Case 2 can lead to a singular Jacobian.

Let us take all the six equations for the torsion bar case and explore different formulations.

**Formulation 1**: This is the formulation in its raw form and is used later to show a complete design sequence.

Twist angle (f₁):
\[ \theta = \frac{T L}{G J} \]

Stress (f₂):
\[ \tau = \frac{T D}{2 J} n \]

Polar moment of inertia (f₃):
\[ J = \frac{\pi D^4}{32} \]

Volume (f₄):
\[ V = \frac{\pi D^2 L}{4} \]

Stiffness (f₅):
\[ K = \frac{T}{\theta} \]
Stress rate ($f_6$):
\[ S = \frac{\tau}{\theta} \]

Since there are 6 equations in 11 variables, the total number of possible combinations is 462. Out of these 223 are invalid due to direct redundancy problems. For the valid 239 combinations, the number of simultaneous equations needed to solve a particular case is given as follows:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>223</td>
<td>163</td>
<td>41</td>
<td>26</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>462</td>
</tr>
</tbody>
</table>

The column 0 indicates the invalid combinations due to direct redundancies.

**Formulation II:** This is taken from the best formulation done before. $J$ has been eliminated.

Twist angle ($f_1$):
\[ \theta = \frac{32 \cdot TL}{\pi GD^4} \]

Stress ($f_2$):
\[ \tau = \frac{16Tn}{\pi D^3} \]

Polar moment of inertia ($f_3$):
\[ J = \frac{\pi D^4}{32} \]

Volume ($f_4$):
\[ V = \frac{\pi D^2L}{4} \]

Stiffness ($f_5$):
\[ K = \frac{T}{\theta} \]
Stress rate ($f_6$):

$$s = \frac{3}{\theta}$$

As before there are 462 possible combinations. For the valid 239 combinations, the number of simultaneous equations needed to solve a particular case is:

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>223</td>
<td>189</td>
<td>29</td>
<td>17</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>462</td>
</tr>
</tbody>
</table>

Notice that now there are no combinations that require the simultaneous solution of 6 equations. Also, the number of 5 equation cases has reduced from 3 to 1 and the number of cases requiring the solution of a single equation has increased from 163 to 189. An empirical measure for the "goodness" of a formulation could be the number of valid cases that lead to single equations. The number under the column 1 is higher in formulation II than formulation I, showing its superiority. It should be noted that any formulation for which the number of direct redundancy cases decreases signals the existence of pathological cases leading to singular Jacobians.
3.5 Example - Design of a Torsion Bar Spring

As an example, consider the torsion bar spring shown in Fig. 3.5.

![Torsion Bar Example](image)

Fig. 3.5: Torsion Bar Example

Two separate formulations for the torsion bar design were presented in the previous section. Let us consider Formulation I, and express the equations in a general homogenous form as:

\[
\begin{align*}
    f_1 &= x_6 - \frac{x_3 x_5}{x_2 x_8} = 0 \\
    f_2 &= x_1 - \frac{x_3 x_4 x_7}{2 x_8} = 0 \\
    f_3 &= x_8 - \frac{x_4}{32} = 0 \\
    f_4 &= x_{11} - \frac{\pi x_4^2 x_5}{4} = 0 \\
    f_5 &= x_9 - \frac{x_3}{x_6} = 0 \\
    f_6 &= x_{10} - \frac{x_4}{x_6} = 0
\end{align*}
\]

where

\[
\begin{align*}
    x_1 &= \tau &= \text{Shear strength of bar material} \\
    x_2 &= G &= \text{Shear modulus of bar} \\
    x_3 &= T &= \text{Torque on bar} \\
    x_4 &= D &= \text{Diameter of bar} \\
    x_5 &= L &= \text{Length of bar}
\end{align*}
\]
\[ x_6 = 0 = \text{Angular twist in bar} \]
\[ x_7 = n = \text{Factor of safety for bar} \]
\[ x_8 = J = \text{Polar moment of inertia} \]
\[ x_9 = K = \text{Torsional stiffness of bar} \]
\[ x_{10} = S = \text{Stress rate in bar} \]
\[ x_{11} = V = \text{Volume} \]

Since we are only illustrating the CM algorithms, we will only concern ourselves with the linear occurrence matrix \([A]\), which is given as:

\[
\begin{bmatrix}
 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
 2 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
 3 & 0 & 0 & 0 & 1 & 0 & 0 & 6 & 1 & 0 & 0 \\
 4 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \\
 5 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
 6 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\]

The above matrix will also be denoted as \([A^0]\) since it is the original occurrence matrix without any updation. Using the disjoint block determination algorithm, we see that the all the equations and variables are part of a single block i.e. there are no disjointed blocks. However, in general this might not be true for multi-component machine design. To illustrate the other constraint management algorithms, we simulate the following design session.

Let us assume that first input is \(x_{11}\) i.e. volume has been specified. This will lead to the zeroing of the 11th column. It can be easily verified that no equations can be solved yet. Next we specify \(x_5\) (torque load) and then \(x_9\) (torsional stiffness). The updated occurrence matrix now looks like the following along the with their rowsums.
<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>ROWSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>

It can be seen that there is a row with $\text{ROWSUM} = 1$ which implies one equation in one variable. Thus we can solve for the output variable $x_6$ using equation $f_5$. Since, $x_6$ is now known, column 6 is also zeroed out. Next we input $x_1$ (shear strength). The occurrence matrix now looks like:

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>ROWSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Again, we can solve one equation in one variable, namely $x_{10}$ using $f_6$. Next we input $x_2$ (shear modulus). After zeroing out column 10 and 2, we get the following occurrence matrix. This leads to the most interesting case to solve.

<table>
<thead>
<tr>
<th>i</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>ROWSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
The ROWSums indicate that the minimum number of simultaneous equations is 2. To determine if any simultaneous equation sets exist in the above matrix, we have to first extract the active occurrence matrix \([A^*]\). As mentioned earlier, a comparison of the ROWSUM values of the equations will be sufficient for this determination. Equations 5 and 6 are already solved and thus are inactive. A simple computation reveals that only \(f_3\) has the current ROWSUM equal to the ROWSUM in \([A^0]\), and is thus marked inactive. The other equations, \(f_1, f_2, f_4\), have lower ROWSUM values compared to \([A^0]\), and hence are the active equations. Thus, \([A^*]\) is given as:

\[
\begin{array}{c|cccc}
\text{AV}(i) \\
4 & 5 & 7 & 8 \\
\hline
1 & 0 & 1 & 0 & 1 \\
2 & 1 & 0 & 1 & 1 \\
4 & 1 & 1 & 0 & 0 \\
\end{array}
\]

Next, we have to find \([A_8]\) emanating from one of the rows of \([A^*]\) by using the algorithm given in Stage 2 of the DDA. Starting from the first active equation, \(f_1\), we add two variables \(x_5\) and \(x_8\) to a candidate list and zero out their columns. It should be noted that a copy of the current \([A]\) is updated during this search stage. The updated \([A]\) with only the unsolved equations is as follows:

\[
\begin{array}{cccccccccccc}
\text{Variable Number,} j & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 \\
\hline
\text{Eq. no. 1} & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
3 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\text{ROWSUM} & & & & & & & & & 0 & <= & \\
2 & & & & & & & & & 2 & & \\
1 & & & & & & & & & 1 & & \\
1 & & & & & & & & & 1 & & \\
\end{array}
\]

Since there are two variables in the candidate list, and only one equation, \(f_1\) with a zero ROWSUM, we continue with our search. Next, we have add equation \(f_3\) to the list of potential equations since it has the lowest ROWSUM. A new variable, \(x_4\), is identified and added to the candidate variable list. Column 4 is then zeroed out. The new matrix now looks as follows:
Since there are three entries in the candidate variable list and we have three equations with zero ROWSUMs, we have found the square matrix \([A_3]\):

\[
\begin{array}{c|cccccccccc}
\text{Eq. no., } j & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & \text{ROWSUM} \\
\hline
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

which indicates that equations \(f_1, f_3, f_4\) can be solved simultaneously to compute \(x_4, x_5,\) and \(x_8\). However, we are still not sure that \([A_3]\) cannot be decomposed further. So, we permute it to make it a zero-free diagonal matrix, which looks like:

\[
\begin{array}{c|ccc}
\text{Eq. no., } j & 4 & 5 & 8 \\
\hline
1 & 0 & 1 & 1 \\
3 & 1 & 0 & 1 \\
4 & 1 & 1 & 0 \\
\end{array}
\]

The above matrix cannot be reduced any further. Thus, the equations 1, 3 and 4 are solved, the occurrence matrix is updated, and the process is continued. The updating of the matrix involves zeroing out the columns 4, 5, and 8 (as variables 4, 5, 8 are solved by equations 1, 3, and 4). The updated occurrence matrix looks like:
where the symbol "\(<=\)" identifies the last equation with one unknown which can be solved. Thus, finally we solve for \(x_7\) using \(f_2\) to complete the overall solution.

The above solution strategy is for the input variables \(x_1, x_2, x_3, x_9, x_{11}\). Now the sequence in which the input variables are specified determines the LEVEL and ORDER information for variables and equations. The input sequence that was used for the illustration of the design decomposition algorithm was \(\{x_{11}, x_3, x_9, x_1, x_2\}\). Then the BLOCK, IO, LEVEL, and ORDER values for the variables and equations are given in the Table 3.1. Note that if the variables were input in any other order, both LEVEL and ORDER would change.

In a symbolic form, the final solution sequence is given as follows:

\[
\Omega = [\{f_5, x_6\}], \{f_6, x_10\}, [\{(f_1, f_3, f_4), (x_4, x_5, x_8)\}, \{f_2, x_7\}]
\]

The complete design state, can be represented as:

\[
[x_3, x_9, \{f_5, x_6\}], [x_1, \{f_6, x_{10}\}], [x_2, x_{11}, \{(f_1, f_3, f_4), (x_4, x_5, x_8)\}], \{f_2, x_7\}]
\]
Table 3.1: Block, Input/Output, Level, and Order corresponding to the Torsion Bar example for the input sequence \( \{x_{11}, x_3, x_9, x_1, x_2\} \)

<table>
<thead>
<tr>
<th>Variable or Equation</th>
<th>BLOCK</th>
<th>IO</th>
<th>LEVEL</th>
<th>ORDER</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( x_4 )</td>
<td>1</td>
<td>-1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( x_5 )</td>
<td>1</td>
<td>-1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( x_6 )</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( x_7 )</td>
<td>1</td>
<td>-1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>( x_8 )</td>
<td>1</td>
<td>-1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( x_9 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( x_{10} )</td>
<td>1</td>
<td>-1</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>( x_{11} )</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>( f_1 )</td>
<td>1</td>
<td>-</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( f_2 )</td>
<td>1</td>
<td>-</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>( f_3 )</td>
<td>1</td>
<td>-</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( f_4 )</td>
<td>1</td>
<td>-</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>( f_5 )</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( f_6 )</td>
<td>1</td>
<td>-</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Example for Forward dependency:

As an example, assume $x_1$ from Table 3.1 is to be "unspecified". Equations $f_1$, $f_2$, $f_3$, $f_4$, and $f_6$ from Block 1 correspond to the same Level 2 as that of $x_1$ or higher. Output variable $x_6$, which is at Level 1, can still be solved and is thus excluded from consideration. A sub-occurrence matrix is formed from the corresponding rows of the original occurrence matrix with only $x_2$, $x_3$, $x_9$, and $x_{11}$ as inputs.

<table>
<thead>
<tr>
<th>Variable Number, J</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>ROWSUM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eq. no., i</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

Using the design decomposition algorithm, we find that $x_4$, $x_5$, and $x_8$ can still be computed using equations $f_1$, $f_3$, and $f_4$ simultaneously. The details of this determination are not documented for brevity. This leaves only variables $x_7$ and $x_{10}$ that cannot be computed as a result of the deletion of $x_1$. Thus,

$\mathcal{F}(x_1) = \{x_7, x_{10}\}$

In a similar way, if the value of $x_1$ had been changed i.e. $x_1$ is "respecified", then we need to recompute only $x_7$ and $x_{10}$. The solution sequence is again determined using the design decomposition algorithm. At this stage, the occurrence matrix will have the columns corresponding to $x_1$, $x_2$, $x_3$, $x_4$, $x_5$, $x_6$, $x_8$, $x_9$, and $x_{11}$ zeroed out.
The solution will thus be to use equation $f_2$ to solve for $x_7$, and then use equation $f_6$ to solve for $x_{10}$.

$$\Omega = \{ [f_2, x_7], [f_6, x_{10}] \}$$

It can be verified that the forward dependency for other input variables is as follows:

$$\mathcal{F}(x_2) = \{ x_4, x_5, x_7, x_8 \}$$
$$\mathcal{F}(x_3) = \{ x_4, x_5, x_6, x_7, x_8, x_{10} \}$$
$$\mathcal{F}(x_9) = \{ x_4, x_5, x_6, x_7, x_8, x_{10} \}$$
$$\mathcal{F}(x_{11}) = \{ x_4, x_5, x_7, x_8 \}$$

**Example for Backward dependency:**

As an example of the procedure, assume that safety factor ($x_7$) which was one of the output variables is to be designated as an input variable. This variable is in Block 1 and has a Level and Order of 3 and 4, respectively. Equation 2 has the same Level and Order, and the input variables appearing in equation 2 are $x_1$ and $x_3$; therefore, $x_1$ and $x_3$ are directly marked as candidates which can be unspecified. To check for other candidates, the other outputs $x_4$ and $x_8$ in equation 2 are considered. Examining $x_4$ in the same manner as was used for $x_7$ indicates that $x_4$ is at Level 3 and Order 3 which involves equations $f_1$, $f_3$ and $f_4$. These equations contain input variables $x_2$ and $x_{11}$ which are then marked as candidates. The examination of output variable $x_6$ reveals the input variable $x_1$. Examining
\(x_8\) produces no new input variables as candidates. Therefore, any of the input variables can be unspecified when \(x_7\) is made an input.

\[ \mathcal{B}(x_7) = \{x_1, x_2, x_3, x_9, x_{11}\} \]

Thus, by the backward dependency algorithm, we are able to establish that \(x_7\) directly or indirectly depends on the \(x_1, x_2, x_3, x_9,\) and \(x_{11}\) input variables. Once one of these input variables is unspecified, then the forward dependency check discussed previously is made and all of the output variables which are dependent on the unspecified input are released. One of these released output variables will be \(x_7\). A value may then be designated for the chosen "output" variable \(x_7\).

It can be verified that the backward dependency for other output variables is as follows:

\[ \mathcal{B}(x_4) = \{x_2, x_3, x_9, x_{11}\} \]
\[ \mathcal{B}(x_5) = \{x_2, x_3, x_9, x_{11}\} \]
\[ \mathcal{B}(x_6) = \{x_3, x_9\} \]
\[ \mathcal{B}(x_8) = \{x_2, x_3, x_9, x_{11}\} \]
\[ \mathcal{B}(x_{10}) = \{x_1, x_3, x_9\} \]
Chapter IV

NONLINEAR EQUATION SOLVER

4.0 Introduction

One of the major tasks of the current research was to develop a robust nonlinear equation solver. The following six methods were studied.

1. Newton's method
2. Levenberg-Marquardt method
3. Powell's method
4. Brent's method
5. Brown's method
6. Secant method with Broyden's Jacobian update

In each case, the solution procedures are iterative and an initial starting point \( x^{(0)} \) must be provided. In most cases, subsequent estimates of \( x \) are provided by the iterative relationship

\[
x^{(i+1)} = x^{(i)} + k^{(i)} \Delta x^{(i)}
\]  

where \( \Delta x^{(i)} \) is the correction vector, and \( k^{(i)} \) is a damping term selected such that either

\[
|f_j(x^{(i+1)})| < |f_j(x^{(i)})| \quad \text{for } j = 1, 2, \ldots, n
\]

or

61
\[
\sum_{j=1}^{n} \left[ f_j(x^{(i+1)}) \right]^2 < \sum_{j=1}^{n} \left[ f_j(x^{(i)}) \right]^2
\]

(4.3)

In general, \( 0 < k^{(i)} \leq 1 \), and 1 is commonly chosen. The iterations are continued until a suitable stopping criteria is reached. The stopping criterion used in most of the cases is based on the Euclidean norm of the function residuals, i.e.

\[
F(x) = \| f(x) \| = \sum_{i=1}^{n} \left[ f_i(x) \right]^2 \leq \epsilon_f
\]

(4.4)

where, \( \epsilon_f \) is a convergence epsilon. Another termination criterion that is usually used is based on the convergence of variables to a certain precision. This can be expressed as:

\[
\left| \frac{x_j^{(i+1)} - x_j^{(i)}}{x_j^{(i)}} \right| \leq \epsilon_{x_j}
\]

for all \( x_j, j=1, 2, ..., n \)

(4.5)

The above criterion is used to stop the iterations when the design variables are no longer progressing to the solution satisfying the design equations. The criterion is not necessarily a condition indicating convergence; and, therefore, it is not used to identify convergence for our research.

A comparison study was conducted for the six methods for a number of mechanical design problems and the results were published in Agrawal and Kinzel (1991). The rest of the chapter describes the six solution techniques.

4.1 Newton's Method

Start with \( x^{(1)}, f^{(1)} = f(x^{(1)}), [J]^{(1)}, \) and \( k^{(1)} \). Then iterate using

\[
\Delta x^{(i)} = -[J^{-1}]^{(i)} f^{(i)}
\]

\[
x^{(i+1)} = x^{(i)} + k^{(i)} \Delta x^{(i)}
\]
\( [J] \) is the Jacobian matrix for \( f \) given by

\[
\begin{bmatrix}
\frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\
\frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n}
\end{bmatrix}
\]

The terms of the Jacobian matrix can be approximated using the forward difference formula

\[
J_{jk} = \frac{\partial f_j}{\partial x_k} = \frac{f_j(x_1, x_2, \ldots, x_{k-1}, x_k + h_k, x_{k+1}, \ldots, x_n) - f_j(x_1, x_2, \ldots, x_n)}{h_k}
\]

where \( h_k \) is a small perturbation in \( x_k \).

4.2 Levenberg-Marquardt Method (Broyden, 1970)

Start with \( x^{(1)}, f^{(1)} = f(x^{(1)}), [J]^{(1)}, \) and \( k^{(i)} \). Then iterate using

\[
\Delta x_i = -\left([J^T]^{(i)}[J]^{(i)} + \lambda^{(i)}[I])^{-1}[J^T]^{(i)}f^{(i)}\right) \quad \lambda^{(i)} \geq 0
\]

Where \( \lambda_i \) is a scalar which is large when \( i \) is small, and \( \lambda_i \) approaches 0 as \( i \) increases.

\[
x^{(i+1)} = x^{(i)} + k^{(i)}\Delta x^{(i)}
\]
4.3 Powell’s Method (Powell, 1970)

Start with $x^{(i)}$ and $f^{(i)} = f(x^{(i)})$, $[J]^{(i)}$, $\Delta$, $\Delta_{\text{max}}$ and $h^{(i)}$, where $\Delta$ is the initial step length ($h \leq \Delta \leq \Delta_{\text{max}}$), $\Delta_{\text{max}}$ is the maximum allowed step length, and $h^{(i)}$ is the perturbation of each parameter used to approximate the partial derivatives. Then iterate using

$$x^{(i+1)} = x^{(i)} + \Delta x^{(i)}$$

**Calculation of $\Delta x^{(i)}$**

$$[B]^{(i)} = [J]^{(i)}$$

$$[H]^{(i)} = [-J]^{-1}[J]^{(i)}; \quad v^{(i)} = -[H]^{(i)}f^{(i)}; \quad g^{(i)} = -[B]^{(i)} f^{(i)}$$

If

$$\|v^{(i)}\| \leq \Delta$$

then $\Delta x^{(i)} = v^{(i)}$; otherwise check

$$\|g^{(i)}\| \geq \Delta \text{ where } \mu = \frac{\|g^{(i)}\|^2}{\|B\|g^{(i)}|^2}$$

If the second inequality is satisfied, then

$$\Delta x^{(i)} = \frac{\Delta g^{(i)}}{\|g^{(i)}\|}$$

If neither inequality is satisfied, then

$$\Delta x^{(i)} = (1 - \theta)\mu g^{(i)} + \theta v^{(i)} \quad \text{for } i = 1, 2, \ldots, n$$

where
\[ \theta = \frac{\Delta^2 - \| \mu_{g(i)} \|^2}{\mu_{v(i)} - \mu_{g(i)}}^T g(i) + \sqrt{\left[ \mu_{v(i)}^T g(i) - \Delta^2 \right]^2 + \left[ v(i)^T v(i) - \Delta^2 \right]^2} \]

Revision of \( \Delta \)

\[ \tau_i = 1; \quad p^{(i)} = \{ r^{(i)} + [B]^{(i)} \Delta x^{(i)} \}; \quad \phi^{(i)} = p^{(i)} T p^{(i)} \]

If,

\[ \left[ r^{(i+1)} \right]^T r^{(i+1)} > \left[ r^{(i)} \right]^T - 0.1 \left[ r^{(i)} \right]^T r^{(i)} - \phi^{(i)} \]

replace \( \Delta \) by the maximum of \( (\Delta/2,h) \) and set \( t = 1 \). Otherwise compute

\[ \chi^2 = 1 + \frac{DM}{SP + \sqrt{SP^2 + (DM)(SS)}} \]

where

\[ DM = \sum_{i=1}^{n} \left[ r^{(i)} \right]^2 - 0.1 \sum_{i=1}^{n} \left[ r^{(i)} \right]^2 - \phi^{(i)} \right] - \sum_{i=1}^{n} \left[ r^{(i+1)} \right]^2 \]

\[ SP = \left[ r^{(i+1)} \right]^T \left[ r^{(i+1)} - p^{(i)} \right] \quad \text{and} \quad SS = \left[ r^{(i+1)} - p^{(i)} \right]^T \left[ r^{(i+1)} - p^{(i)} \right] \]

After two values of \( \lambda \) have been computed, compute

\[ \mu = \min(2,\lambda,\tau) \quad \text{and} \quad \tau = \lambda / \mu \]

and replace \( \Delta \) by the minimum of \( (\mu \Delta, \Delta_{\max}) \).

Revision of \([B]\) and \([H]\)

\[ [B]^{(i+1)} = [B]^{(i)} + \omega \left( y^{(i)} - [B]^{(i)} \Delta x^{(i)} \right) \left[ \Delta x^{(i)} \right]^T \]

\[ \left[ \Delta x^{(i)} \right]^T \Delta x^{(i)} \]
\[
[H]^{i+1} = [H]^{(i)} + \alpha \frac{[\Delta x^{(i)}]^T - [H]_i y_i [\Delta x^{(i)}]^{T} [H]_i}{\alpha [\Delta x^{(i)}]^T [H]_i \Delta x^{(i)} + (1 - \alpha) [\Delta x^{(i)}]^T \Delta x^{(i)}}
\]

where \( y^{(i)} = f^{(i+1)} - f^{(i)} \) and \( \alpha = 1 \) if \( \left\| [\Delta x^{(i)}]^T [H]_i y^{(i)} \right\| \geq [\Delta x^{(i)}]^T \Delta x^{(i)} \); otherwise, \( \alpha = 0.8 \).

The original procedure proposed by Powell [1970] also ensures that the last \( n \) search directions \([d's] \) are linearly independent.

### 4.4 Brent's Method (Brent, 1973; Moré, 1979)

Brent's procedure is based on the orthogonal triangularization of the Jacobian. For an improved efficiency of the Brent's method, the equations should be arranged from the least nonlinear to the most nonlinear. Start with \( i=1 \), \( x^{(1)} \), \( y^{(1)} = x^{(1)} \) and \([Q]^{(1)} = [I] \). Then iterate using the following:

1. Compute the perturbation vector \( h \) from

\[
h_i = \begin{cases} 
\varepsilon |x_i| & \text{if} |x_i| > \varepsilon \\
\varepsilon & \text{otherwise}
\end{cases}
\]

for \( i = 1, \ldots, n \)

where \( \varepsilon \) is the square root of the machine precision.

2. For \( k=1,2,3,\ldots,n \), use the following to compute \( y^{(k+1)} \) from \( y^{(k)} \).

a. Compute \( w \), an approximation to the gradient of \( f_k \) without its first \( k-1 \) elements, from
\[ w = \begin{bmatrix} 0 \\ \vdots \\ f_k(y^{(k)} + h_k [Q]^{(k)} e_k) - f_k(y^{(k)}) \\ f_k(y^{(k)} + h_{k+1} [Q]^{(k)} e_{k+1}) - f_k(y^{(k)}) \\ \vdots \\ f_k(y^{(k)} + h_n [Q]^{(k)} e_n) - f_k(y^{(k)}) \\ h_n \\ \end{bmatrix} \]

where \( e_j \) is the \( j^{\text{th}} \) column of the identity matrix \([I]\).

b. Compute a \((n-k+1) \times (n-k+1)\) sub-matrix \([U]^{(k)}\). This can be a either a single Householder transformation or a product of \((n-k)\) plane rotation matrices. We use the following Householder transformation (Stewart, 1973).

\[ [U]^{(k)} = [I] - \pi^{-1} vv^T \]

where,

\[ v = w + \sigma e_k \]

\[ \sigma = \text{sign}(w_k) \sqrt{w^T w} \], where \( w_k \) is the \( k^{\text{th}} \) component of \( w \).

and

\[ \pi = \frac{1}{2} v^T v = \sigma v_1 \]

c. Form the orthogonal matrix \([U]^{(k)}\) as

\[ [U]^{(k)} = \begin{bmatrix} [I]_{k-1} & 0 \\ -\frac{1}{\pi} & -\frac{1}{\pi} \\ 0 & [U]^{(k)} \end{bmatrix} \]
d. Compute \( \alpha \) from the expression \( [U]^{(k)} w = \alpha e_k \). Because \( [U]^{(k)} \) was derived using a Householder transformation, \( \alpha = -\sigma \).

e. Compute \( [Q]^{(k+1)} = [Q]^{(k)} [U]^{(k)} \) and

\[
y^{(k+1)} = y^{(k)} - \frac{f_k(y^{(k)})}{\alpha} [Q]^{(k+1)} e_k
\]

3. Set \( x^{(i+1)} = y^{(n+1)} \), \( y^{(i+1)} = x^{(i+1)} \), \( [Q]^{(1)} = [I] \) (or alternately use \( [Q]^{(1)} = [Q]^{(n+1)} \)), increment \( i \), and continue iterating from Step i until the termination criteria are satisfied.

4.5 Brown's Method (Brown, 1969; Cosnard, 1975)

Brown's method can be described in terms of a sequence of \( n+1 \) matrices \( R^{(1)}, \ldots, R^{(n+1)} \). The columns of \( R \) are denoted by \( u_1, \ldots, u_n \). Start with \( x^{(1)} \), and \( f^{(1)} = f(x^{(1)}) \). Then iterate using the following:

1. Set \( i = 1 \), and let \( y^{(1)} = x^{(1)} \), \( u_j = e_j \) for \( j = 1, \ldots, n \), where \( e_j \) is the jth column of the identity matrix \([I]\).

2. Compute the perturbation vector \( h \) from

\[
h_i = \begin{cases} 
|e^T x_i| & \text{if } |x_i| > \varepsilon \\
\varepsilon & \text{otherwise}
\end{cases} \text{ for } i = 1, 2, \ldots, n
\]

where \( \varepsilon \) is the square root of the machine precision.

3. Enter a sub-iteration. For \( k = 1, 2, \ldots, n \)

a. Compute \( w \), an approximation to the gradient of \( f_k \) without its first \( k-1 \) elements, from
\[ w = \begin{bmatrix} 0 \\ \vdots \\ \frac{f_k(y^{(k)} + h_k u_k) - f_k(y^{(k)})}{h_k} \\ \frac{f_k(y^{(k)} + h_{k+1} u_{k+1}) - f_k(y^{(k)})}{h_{k+1}} \\ \vdots \\ \frac{f_k(y^{(k)} + h_n u_n) - f_k(y^{(k)})}{h_n} \end{bmatrix} \]

b. Compute $j^*$ as the index corresponding to the maximum component of $w$.

c. If $j^* \neq k$, then
   - exchange $w_{j^*}$ and $w_k$
   - exchange $u_{j^*}$ and $u_k$

d. Compute

\[ y^{(k+1)} = y^{(k)} - \frac{f_k(y^{(k)})}{w_k} u_k \]

\[ u_\ell = u_\ell - \frac{w_\ell}{w_k} u_k \quad \ell = k+1, \ldots, n \]

4. Set $x^{(i+1)} = y^{(n+1)}$, $y^{(1)} = x^{(i+1)}$, increment $i$, and continue iterating from Step 2 until the termination criteria are satisfied.

4.6 **Secant Method with Broyden Update** (Dennis and Schnabel, 1983)

Start with $x^{(1)}$, $f^{(1)} = f(x^{(1)})$, $[J]^{(1)}$, and $k^{(i)}$.

\[ [B]^{(1)} = [J]^{(1)} \]

Then iterate using
\[ \Delta x^{(i)} = -[B^{-1}]^T f^{(i)} \]

\[ x^{(i+1)} = x^{(i)} + k^{(i)} \Delta x^{(i)} \]

\[ q^{(i)} = x^{(i+1)} - x^{(i)} \]

\[ [B]^{(i+1)} = [B]^{(i)} - ([f]^{(i+1)} - [f]^{(i)} - [B]^{(i)} q^{(i)} ) \frac{[q]^{(i)}T}{[q]^{(i)}T q^{(i)}} \]

### 4.7 Scaling

Two types of scaling have been suggested for the simultaneous solution of nonlinear equations. These are function scaling and variable scaling. Intuitively, one would guess that the application of both the scalings should improve the solvers. To incorporate function scaling, the initial residual values are used as typical values for the functions.

In the case of variable scaling, there is a change of variables such that

\[ u_i = A_i x_i + B_i \]

and

\[ x_i = \frac{u_i - B_i}{A_i} \]

where typical values for \( A_i \) and \( B_i \), respectively, are commonly taken as

\[ A_i = \frac{1}{x_{\text{max}_i} - x_{\text{min}_i}} \]

and
\[ B_i = \frac{-x_{\min_i}}{x_{\max_i} - x_{\min_i}} \]

although sometimes, \( B_i \) is set to 0 and \( A_i \) is taken to be a typical value of the variable (Dennis, 1983). Here, \( x_{\min_i} \) and \( x_{\max_i} \) are the minimum and maximum expected values, respectively, for \( x_i \).
Chapter V
NONLINEAR OPTIMIZATION

5.0 Introduction

This chapter describes the development of algorithms based on constraint management to handle continuous and discrete optimization. There are several different approaches to handle constraints in nonlinear optimization and the Generalized Reduced Gradient (GRG) method has been used very effectively for this purpose (Gabriele, 1980; Li, 1988). In spite of its popularity, GRG method has some troublesome issues such as the selection of the initial basis and the problem of basis interchange. These issues can be resolved using the constraint management techniques presented in Chapter 3 with the help of the nonlinear occurrence matrix.

This chapter presents a heuristic algorithm to determine an initial partition of the state and decision variables. The procedure not only ensures that the Jacobian for the state variables is nonsingular, but is also aimed at minimizing the nonlinear component in the set of state variables. Such a selection will lead to the reduction of the computational effort during the feasibility restoration phase of the GRG method. Further, this chapter presents another algorithm to automate the basis interchange by using a linearized model of the governing equations and a backward dependency procedure. An improved canonical form for the general nonlinear programming problem that has the objective function embedded in the set of equality constraints is also presented in this chapter.

The GRG strategy for nonlinear optimization treats all the variables as continuous. Due to a number of reasons, some of the variables may be allowed to take on only non-continuous or discrete values. Machine design also offers varied discreteness in variables. Often,
designers tend to ignore the discreteness constraints and round-off the values of the continuous optimum. There are several pitfalls to this approach (Fu et al., 1989). Fig. 5.1 shows an optimization case where the values of the discrete optimum are significantly different from the continuous optimum. Fig. 5.2 depicts a design case where the neighboring values near the continuous optimum might all be infeasible.

Fig. 5.1: Infeasible points around a continuous optimum
Fig. 5.2: Discrete optimum far away from the continuous optimum
Thus, we need a systematic procedure to locate the discrete optimum. This chapter presents a branch-and-bound algorithm that works with the GRG algorithm to ensure that the discreteness constraints are satisfied at the end of the optimization.

5.1 Modeling a standard NLP into a Constraint Management Problem (EOF Model)

A standard nonlinear programming (NLP) problem can be stated as

Minimize \( F(x) \)

subject to
\[
\begin{align*}
  f(x) &= 0 \\
  x^l &\leq x \leq x^u
\end{align*}
\]  

(5.1)

where \( F(x) \) represents the objective function, and \( f(x) \) represents a set of constraints. It is assumed that all the inequality constraints have been converted to equality constraints by the addition of slack variables. In order to use the constraint management algorithms presented in Chapter 3, we need to introduce a dummy variable, \( x_{N+1} \), which will represent the objective function. Thus, our new formulation is:

Minimize \( x_{N+1} \)

\[
\begin{align*}
  f(x) &= 0 \\
  F(x) - x_{N+1} &= 0 \\
  x^l &\leq x \leq x^u
\end{align*}
\]  

(5.2)

There is only a subtle difference between formulation (5.1) and (5.2). But, now the objective function is embedded into the design model. In fact, with the above formulation, we need not have an explicit objective function. So in general we assume that the objective function is modeled just like any other equality constraint. For the rest of the discussion, it will be assumed that one of the variables from the \( x \) vector, designated as \( x_{\text{obj}} \), is the objective function being minimized. This model is referred to an Embedded Objective Function (EOF) model.
5.2 Constraint Management Based Generalized Reduced Gradient (CMB-GRG) Algorithm

The GRG method relies on a robust unconstrained optimizer, an efficient algorithm for minimizing a function along a straight line, and a nonlinear equation solver. The optimizer is used to select search directions while the nonlinear equation solver is used to restore the feasibility of any design point by solving for state variables.

For the form of the GRG method considered here, a search direction is found such that any active constraints are satisfied for small moves in the search direction. The following steps describe the algorithm used.

**Step 1:** Convert the problem to the following canonical form.

\[
\begin{align*}
\text{Minimize } & \quad x_{\text{obj}}, \\
\text{S.t. } & \quad f_i(x_1, x_2, \ldots, x_N) = 0, \quad i = 1, \ldots, M \\
& \quad x_i \geq x_i^{l} \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad i = 1, \ldots, n \\
& \quad x_i \leq x_i^{u} \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad i = 1, \ldots, n \\
& \quad x_i \geq 0 \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad i = n+1, \ldots, N \quad \text{(slack variables)}
\end{align*}
\]

**Step 2:** Partition the \( x \) vector into \( M \) state variables, \( y \), and \( (N-M) \) decision variables, \( z \).

An algorithm to select a partition is presented in a later section.

\[
x = [z, y]^T
\]

Since \( x_{\text{obj}} \) is an output variable, it is also a part of the vector \( y \).

**Step 3:** For iteration \( i \), find the matrix \([S]\) of the gradients of \( f(x) \) with respect to \( y \) at the point \( x^{(i)} \).
Step 4: For iteration \( k \), find the matrix \([D]\) of the gradients of \( f(x) \) with respect to \( z \) at the point \( x^{(k)} \).

\[
[S] = \begin{bmatrix}
\nabla_y f_1(x) \\
\nabla_y f_2(x) \\
\vdots \\
\nabla_y f_m(x)_{(M \times M)} \\
\end{bmatrix}
\]

\[
[D] = \begin{bmatrix}
\nabla_z f_1(x) \\
\nabla_z f_2(x) \\
\vdots \\
\nabla_z f_m(x)_{(M \times (N-M))} \\
\end{bmatrix}
\]

Step 5: Compute the reduced gradient at the current point.

\[
g_r^{(i)} = \nabla_z x_{\text{obj}} - [S^{-1}D]^T \nabla_y x_{\text{obj}}
\]

For the first iteration, \( g^{(1)} = g_r^{(1)} \)

Step 6: Determine a line search direction based on the current \( g \) vector and values of the decision variables by using the following rules (Reklaitis et al., 1983)

\[
s_i^k = \begin{cases} 
0 & \text{if } z_i = z_i^a \text{ and } g_i < 0 \\
0 & \text{if } z_i = z_i^b \text{ and } g_i > 0 \\
-g_i & \text{otherwise}
\end{cases}
\]

Step 7: Minimize \( F(x) \) i.e. \( x_{\text{obj}} \) in a one dimensional search by directly adjusting only the decision variables. Thus,

\[
z^{\text{new}} = z^{\text{old}} + \alpha s^k
\]

where \( \alpha \) is the step length parameter. If any of the limit constraints are violated for the \( z \) vector, the value of \( \alpha \) is adjusted.
**Step 7a:** After each step involving a change in \( \alpha \), the state variables are recomputed to maintain feasibility by solving the equality constraints.

\[ f(y, z) = 0 \]

**Step 7b:** If any of the \( y_l \)'s violate the limit constraints, a basis change is initiated. Go to Step 3.

**Step 7c:** Continue changing \( \alpha \) until \( F(x) \) is minimized along \( s^k \). The following criteria is used to terminate the line search:

\[
| x_{\text{new}}^N - x_{\text{old}}^N | \leq \varepsilon_f
\]

which simply states that the objective function converges to within a certain \( \varepsilon_f \).

**Step 8:** Compute the new reduced gradient vector using the formula in Step 5. Update it using the Broyden-Fletcher-GoldFarb-Shanno (BFGS) formula (See next section) to obtain the new \( g \) vector.

\[
g^{(i+1)} = [H]^{(i+1)}g^{(i+1)}
\]

**Step 9:** Continue iterating from Step 6 until one of the following termination criteria is reached.

1. \( \| s^k \| \leq \varepsilon_s \)
2. \( \| x^{k+1} - x^k \| \leq \varepsilon_x \)
3. \( | x_{N}^{k+1} - x_{N}^k | \leq \varepsilon_f \)

With this approach, the equality and the inequality constraints are satisfied for every change in the optimization variables. In Step 8, the generalized reduced gradient could also be
used as the new search direction; however, in this study, it has been found that the BFGS update is more efficient.

It should be noted that the matrices $[D]$ and $[S]$ are never explicitly computed in the implementation of the GRG method. The reduced gradient is computed by perturbing the decision variables and computing all the state variables. One of the state variables will be the objective function. A simultaneous nonlinear solver is used to determine the state variables whenever the decision variables are changed. To reduce the computational time, the occurrence matrix for the state variables is decomposed into smaller sets using the Design Decomposition Algorithm presented in Chapter 3.

In Step 9, the first termination criterion stops at the optimum if the search direction is zero to a specified tolerance. If the optimum is an interior optimum, then the reduced gradient at this point will also be zero. In such a case, the Karush-Kuhn-Tucker (KKT) conditions for optimality are also satisfied (Gabriele, 1975).

5.2.1 Broyden-Fletcher-Goldfarb-Shanno (BFGS) Update

The Broyden-Fletcher-Goldfarb-Shanno method belongs to the general class of methods called variable metric methods. The matrix $[H]$ is an approximation to the inverse of the Hessian matrix at the point $x_i$. At the beginning of the search, the matrix $[H]$ is taken as the identity matrix $[I]$. That is,

$$[H]^{(1)} = [I]$$

For the next iteration, $[H]$ is updated according to the following equation

$$[H]^{(i+1)} = [H]^{(i)} + \frac{\sigma + \tau}{\sigma^2} \Delta x \Delta x^T - \frac{1}{\sigma} \left( [H]^{(i)} \Delta g \Delta x^T + \Delta x [H]^{(i)} \Delta g^T \right)$$

where the change vectors $\Delta x$ and $\Delta g$ are defined by
\[ \Delta x = x_i - x_{i-1} \]
\[ \Delta g = \nabla F(x_i) - \nabla F(x_{i-1}) \]

and the scalars \( \sigma \) and \( \tau \) are defined by

\[ \sigma = \Delta x \cdot \Delta g \]
\[ \tau = \Delta g^T [H_j] \Delta g \]

### 5.2.2 Initial Basis Selection Algorithm

The initial basis selection is still one of the problems being investigated for the GRG method. As pointed out earlier, there are \((N-M)\) degrees of freedom in the optimization problem. The problem that needs to be resolved is the choice of the \((N-M)\) decision variables. Any of the \( N \) variables except \( x_{\text{obj}} \) are candidates for the choice. The maximum number of possible choices is given by the combinations, \( nc \), of \((N-1)\) distinct items taken \((N-M)\) at a time or

\[ nc = \binom{(N-1)}{(N-M)} = \frac{(N-1)!}{(M-1)!(N-M)!} \]  

(5)

In a general machine design problem, the theoretical limit for the number of combinations will be less than this due to the direct redundancy problems discussed in Chapter 3. This means that if one or more equations contains \((N-M)\) or fewer variables, all of the variables in those equations cannot be chosen independently.

Ideally, every choice for the initial basis would give the same optimum design. However, this is unlikely because of the multimodal nature of design problems. Furthermore, different choices of variables can make the optimization more or less difficult to solve. R. Johnson (1979) presented a method for explicitly enumerating all the choices and studying the problem. Several researchers have pointed out the following conditions to select state variables:

1. State variables should lie away from the bounds.
2. The resulting Jacobian to compute the state variables should be non-singular.

The first condition is simple to satisfy and just depends on the starting point of the optimization. The second condition is much more complex. The use of a nonlinear occurrence matrix \([B]\) coupled with the Design Decomposition Algorithm presented in Chapter 3 is helpful in ensuring that an independent set of variables is selected as the non-basic variables. This following presents a heuristic algorithm that attempts to reduce the nonlinearity component of the state variables.

**Step 0:** Zero out the columns for any input or solved variables.

**Step 1:** Compute the \(\text{COLSUM}\) vector by summing the columns of all the variables in \([B]\).

\[
\text{COLSUM}_j = \sum_{i=1}^{M} B_{ij}
\]

**Step 2:** If the \(\text{COLSUM}\) vector is equal to 0, stop.

**Step 3:** Select the variable with the highest column sum. This variable is the most nonlinear in the set of equations. Designate this as a decision variable. In case of a tie between two variables having equal column sum, a variable that appears in more equations is selected first. If that number is also tied, then pick the variable with the lower index.

**Step 4:** Perform a redundancy check to determine if any variables can be solved due to selection of the decision variable. A Design Decomposition Algorithm presented in Chapter 3 is used for this purpose. All the solved variables are designated as state variables.

**Step 5:** Zero out the columns for the decision variable and the dependent state variables. It is possible that for a certain decision variable, there might not be any new dependent state variables.
Step 6: Return to Step 2.

Since, redundancy checking is performed after the selection of every decision variable, we are guaranteed that cases which lead to a permanently singular Jacobians will be eliminated. The algorithm also ensures that variables that are highly nonlinear are picked as decision variables, thus reducing the computational effort to determine the feasible state variables at every step of the GRG method.

5.2.3 Basis Interchange Algorithm

Basis change or respecification of state and decision variable is one of the most important aspects of the GRG method. During the line search stage of the optimization, it is possible to arrive at a point where a state variable does not lie between the specified bounds. In case of slack variables, such a condition often indicates the violation of the corresponding inequality constraint. To start another iteration, the violated state variable must be specified as a new decision variable, and a proper decision variable must be made a state variable. The violated state variable is then set to the appropriate limit value.

To find the proper decision variable to exchange with the violated state variable, say $y_s$, we find all of the decision variables which directly affect $y_s$ by using the Backward Dependency Algorithm (See Chapter 3). There are several guidelines to select which decision variable should enter the basis (Abadie and Carpentier, 1969; Gabriele and Ragsdell, 1977). One heuristic is to pick a variable that is far from its bounds and another one is to take the variable that is least sensitive to $y_s$. Combining the two heuristics, Abadie and Carpentier (1969) suggests the following expression to determine the entering basic variable, $z_D$:

$$z_D = \max_i \left( \left| \frac{\partial y_s}{\partial z_i} \right| \min \left[ (z_i^u - z_i), (z_i - z_i^l) \right] \right)$$

Using the above formula, we would choose a decision variable that is least likely to violate limits in the subsequent iterations. It should be emphasized that using the backward
dependency algorithm, we automatically eliminate the computation of unnecessary partial
derivatives.

In our research we use another heuristic called the "design deviation" to select a decision
variable to enter the basis. Design deviation gives a measure of the distance between the
current infeasible point to the nearest feasible point. A linearized model is used to compute
the estimated changes in the design variables. The decision variable that results in the
minimum value for the design deviation is picked.

There can be a number of ways to compute the design deviation. We use the sum of the
squares of the scaled deviations of the design variables from their current values. Let us
assume that the variable $y_s$ violates the specified bounds by an amount $\Delta y_s$ and that there
are $n_y$ state variables. Then, mathematically, design deviation is defined as:

$$\Psi_{z_i} = \left( \frac{\Delta z_i}{z_i} \right)^2 + \sum_{j=1}^{n_y} \left( \frac{\Delta y_j}{y_j} \right)^2 \quad \forall z_j \in \mathcal{B}(y_s)$$

where,

$$\Delta z_i = \frac{\partial y_s}{\partial z_i} \Delta y_s$$

represents the amount of change required in the backwardly-dependent
decision variable to make a change of $\Delta y_s$ in the violated state variable.

$$\Delta y_j = \Delta z_i \left( \frac{\partial y_j}{\partial z_i} \right)$$

represents the change in the other state variables as a result of $\Delta z_i$
change in the backwardly-dependent decision variable.

In a compact form, we get,

$$\Psi_{z_i} = \frac{\Delta y_s}{z_i} \frac{\partial y_s}{\partial z_i} + \sum_{j=1}^{n_y} \frac{\Delta z_i}{y_j} \left( \frac{\partial y_j}{\partial z_i} \right)$$
5.2.4 Example

To illustrate the basis selection and interchange algorithms, let us consider the following example.

Minimize: \[ F(x) = (x_1 - 3)^2 + (x_2 - 3)^2 \]
Subject to:
\[
\begin{align*}
g_1(x) &= 2x_1 - x_2^2 - 1 \geq 0 \\
g_2(x) &= 9 - 0.8x_1^2 - 2x_2 \geq 0 \\
0 \leq x_1 &\leq 4 \quad 0 \leq x_2 \leq 4
\end{align*}
\]

Converting the first two inequality constraints into equality constraints with the addition of slack variables and making the objective function as the fifth variable, we derive the following EOF formulation:

Minimize \[ x_5 \]
Subject to:
\[
\begin{align*}
f_1(x) &= 2x_1 - x_2^2 - 1 - x_3 = 0 \\
f_2(x) &= 9 - 0.8x_1^2 - 2x_2 - x_4 = 0 \\
f_3(x) &= (x_1 - 3)^2 + (x_2 - 3)^2 - x_5 = 0 \\
0 \leq x_1 &\leq 4 \\
0 \leq x_2 &\leq 4 \\
x_3 \geq 0 \quad x_4 \geq 0
\end{align*}
\]

In the above EOF form, we can optimize any variable which is involved in this set of equations. Before the optimization, we need to define the occurrence matrix \([A]\) and the nonlinear occurrence matrix \([B]\) as follows:

\[
A = \begin{bmatrix}
1 & 1 & 1 & 0 & 0 \\
1 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 1
\end{bmatrix} \quad \begin{bmatrix}
g_1 \\
g_2 \\
g_3
\end{bmatrix}
\]
\[
\begin{bmatrix}
  x_1 & x_2 & x_3 & x_4 & x_5 \\
  1 & 2 & 1 & 0 & 0 & g_1 \\
  2 & 1 & 0 & 1 & 0 & g_2 \\
  2 & 2 & 0 & 0 & 1 & g_3 
\end{bmatrix}
\]

For this example, \( N = 5 \) and \( M = 3 \). The degrees of freedom in the system is \((N-M) = 2\). To select an initial basis, we observe that we can select 2 variables out of 4. Hence, there are \( ^4C_2 = 6 \) possible bases. By examining the \([B]\) matrix, we find out that \( x_1 \) and \( x_2 \) appear in the equations in a more nonlinear fashion than do \( x_3 \) and \( x_4 \). Therefore, we can pick \( x_1 \) as the first decision variable. After performing redundancy checking, it is seen that no equations can be solved. Next, \( x_2 \) is added in the decision variable list. Then, the rest of the variables can be found by the following solution sequence:

\[
\Omega = \{ (f_1, x_3), (f_2, x_4), (f_3, x_5) \}
\]

The solution sequence should be read as "solve \( x_3 \) using \( f_1 \), then solve \( x_4 \) using \( f_2 \) and finally \( x_5 \) using \( f_3 \)."

Start from \( x^{(1)} = (1, 1, 0, 6.2, 4) \)

The first basis change occurs when \( x^{(1)} = (1.7071, 1.7071, -0.49999, 3.2544, 3.3431) \). Here \( x_3 \) violates the lower bound. The basis interchange algorithm is used to determine the new basic variable. Using the backward dependency algorithm, we find that,

\( \mathcal{B}(x_3) = \{ x_1, x_2 \} \)

After performing the sensitivity calculations, we get,

\[
\frac{\partial x_3}{\partial x_1} = 2.0 \quad \frac{\partial x_3}{\partial x_2} = -3.4
\]

Since, both \( x_1 \) and \( x_2 \) are equidistant from the limits, we pick \( x_2 \) as the new state variable.

The new set of decision variables is \( (x_1, x_3) \). The solution sequence changes to:
\[ \Omega = [(f_1, x_2), (f_2, x_4), (f_3, x_5)] \]

The second basis change happens when \( x^{(2)} = (2.7071, 2.1010, 0, -1.0647, 0.8939) \). In this case, \( x_4 \) violates the lower bound. The basis interchange algorithm designates \( x_1 \) to be the new basis variable since \( x_3 \) is already at the limit. Thus the new set of decision variables is \((x_3, x_4)\) and the solution sequence is:

\[ \Omega = [((f_1, f_2), (x_3, x_4)), (f_3, x_5)] \]

Here, we have to solve equations \( f_1 \) and \( f_2 \) simultaneously to evaluate the slack variables \( x_3 \) and \( x_4 \). Then, \( f_3 \) is used to solve \( x_5 \).

After the second basis change, both \( x_3 \) and \( x_4 \) are fixed at their lower limit and the search direction vector has a norm of 0. The procedure then terminates.

The total number of function calls required to reach the optimum was 15 and the optimum point is \( x^* = (2.5, 2, 0, 0, 1.25) \). Since \( x_5 \) is the objective function variable, the optimum value is 1.25. At the optimum, both the inequality constraints are active, since \( x_3 = x_4 = 0 \).

### 5.3 Discrete Optimization

The GRG method presented in the earlier sections is well suited for continuous optimization. But, as pointed out, mechanical design has a number of discrete variables. In the most general sense, a discrete variable can be defined as a variable that takes on values from a set. The set of choices for a discrete variable can be finite or it may be infinite. Once the bounds on all the variables are specified, one can assume the set to become finite. So, for example, for integer number of gear teeth, one might consider tooth number between 7 and 200 and not the entire integer domain.

Depending on the cardinality of the set, the following sub-classification of discrete variables has been done. These are:
Bivalent - These variables can take on either the value 0 or 1. They are sometimes referred to as binary variables or decision variables. A classic example of bivalent variables is the Generalized Assignment Problem (GAP). Sometimes a variables of other types can be converted to bivalent variables and any of the 0-1 procedures can be used. One often encounters such variables in design when trying to answer yes/no type of questions or choose between two finite states.

Finitely discrete - Such variables take on values from a finite set. For example, a design variable that is restricted to standard dimensions.

Infinitely discrete - Such variables take on values from an infinite set. For example, integer variables. Discretized continuous variables can also be thought of as belonging to this category.

In a broader sense, once the bounds of the variables have been specified and the continuous variables have been discretized, all the variables fall into the category of finitely discrete.

5.3.1 Current Discrete Optimization methods

There are a number of methods designed specially for discrete optimization problems. Integer programming problems have been around for a long time. Traditional procedures have often handled only linear constraints, though now there are methods that can handle non-linear objective function as well as constraints. Branch and bound methods with some kind of mathematical heuristic are the most popular algorithms.

Special algorithms exist for bivalent variables. 0-1 programming has also been used to deal with finitely-discrete variables. Each of the choice is made into a separate variable.

\[ x_1 \in \{s_1, s_2, \ldots, s_n\} \]

Convert the problem to
and add a constraint:

$$\sum_{i=1}^{n} x_{ii} = 1$$

But it is clear that 0-1 methods cannot be used to handle integer variables.

Mathematically exhaustive searches have also been used for small problems. Guided by mathematical heuristic, such searches can locate the global optimum in a reasonable amount of time.

Thus we need a combination of existing methods to solve the problem. A number of different procedures come to mind when dealing with discreteness. As mentioned earlier, one could convert all the variables into the finitely-discrete type. Then an algorithm could be developed to handle just these variables. This idea would mean discretizing all the continuous variables.

Another method that can be explored is to convert all the continuous variables into integer variables and solve a purely integer programming problem. But the problem with this method might be the fact, that most integer programming procedures start off from a continuous optimum. So we are going a step back.

Discretizing continuous variables and making all the discrete choices into bivalent variables is a technique that holds some promise. This might mean increase the dimension of the design space by a very huge amount, but simplifying the nature of the problem.

The best solution seems to be to develop a mixed-variable algorithm that can handle both the continuous and discrete variable. Branch and bound is a method that starts from the continuous optimum and successively refines to generate the discrete optimum. The next section describes the branch and bound method.
5.3.2 Branch-and-Bound Method

The branch-and-bound method as proposed by Land and Doig (1960) is well suited for traditional linear programming environment. Several researchers (Dakin, 1965; Gupta and Ravindran, 1980; Sandgren, 1988) have tried to enhance the procedure for nonlinear problems. The branch and bound method falls under the category of implicit enumeration or sometimes called partial enumeration techniques (Parker, 1988). Following is an overview of the branch-and-bound method used in our research:

**Step 1:** Determine the continuous optimum to the nonlinear programming problem by relaxing the discreteness requirements.

**Step 2:** Form a list of all the discrete variables in the current solution that are not at their prescribed discrete value. If this list is empty, then we have found an "incumbent solution." This solution will serve as a guide for the rest of the optimization search. Perform "pruning" of the unexplored subproblems list. This is done by examining the value of the incumbent solution and eliminating all the subproblems with a higher objective function value (assuming that we are solving a minimization problem). Such an elimination is also referred to as fathoming. Proceed directly to Step 6, otherwise go to Step 3.

**Step 3:** Select one of the variables from the list created in Step 2 using a criterion discussed in Section 5.3.4. Let this variable be \( x_j \). Create two sub-problems. In one sub-problem, add the constraint,

\[ x_j \leq x_{j_k} \]

and in the other sub-problem, add the constraint,

\[ x_j \geq x_{j_{k+1}} \]

where \( x_{j_k} \) is the discrete choice just below \( x_j \) and \( x_{j_{k+1}} \) is just above \( x_j \).

**Step 4:** Check if the branching variable is a decision variable. If the branching variable is a state variable, then perform a basis interchange. Thus, we ensure that the branching variable is always a decision variable.
Step 5: Perform feasibility check on both problems created in Step 3. Compute the continuous optimum for the feasible subproblems and add them to an unexplored subproblem list.

Step 6: Check if the unexplored subproblem list is empty. If so, the algorithm terminates and the current incumbent is the discrete optimum. Otherwise, proceed to Step 7.

Step 7: Select a problem to explore from the list of subproblems using a criterion discussed in Section 5.3.3. Mark the selected subproblem as fathomed.

Step 8: Go back to Step 2.

There are different forms of the branch and bound algorithm, some of which are storage efficient while others are computationally expensive. Some researchers (Ibarki, 1977; Glover, 1986) have also suggested use of heuristic functions to assist the search algorithm. There are two key problems associated with the branch and bound method. These are:

1. Subproblem selection: Which subproblem to select from the design space for further exploration?
2. Branching variable selection: Amongst the discrete variables which are not at their prescribed discrete value, which one should be selected for branching?

Both these problems are addressed in the following sections.

5.3.3 Subproblem selection

As noted in step 7 of the branch-and-bound algorithm, we have to select the subproblem out of a list of unexplored subproblems. There are two strategies that can be adopted:

1. **Depth First**: Select the last subproblem that was added to the unexplored list. This strategy picks the top subproblem of a stack.

2. **Breadth First**: Select a subproblem having the lowest objective function value (assuming a minimization problem is being solved).
In general, the depth first strategy will be storage efficient since we would not need to store complete subproblems. But, a breadth first strategy might reach the discrete optimum earlier. Owing to the interactive nature of the design presented in this thesis, we have selected the breadth first strategy in our implementation.

5.3.4 Branching variable selection

Determination of the branching variable from the selected subproblem is perhaps the most difficult part of branch and bound. Past research (Mitra, 1973) has suggested several different strategies for determining the branching variable. Though such strategies were mentioned for integer variables, they can be easily extended to a general discrete variable as follows.

Strategy 1: Branch on the first discrete variable i.e. the variable with the lowest index.

Strategy 2: Select a variable that is closest to the a discrete value, whether lower or upper.
Let \( x_{jk} \) be the discrete value just below \( x_j \) and \( x_{jk+1} \) be the discrete value just above. Compute,

\[
\begin{align*}
    f_{1j} & = x_j - x_{jk} \\
    f_{2j} & = x_{jk+1} - x_j \\
    f_j & = \min (f_{1j}, f_{2j})
\end{align*}
\]

Branch on the variable \( x_m \) that satisfies:

\[
    f_m = \min \{ f_j \}
\]

Strategy 3: Select a variable that is farthest from a discrete value, whether lower or upper.
Compute \( f_{1j} \) and \( f_{2j} \) as given in Strategy 2.

\[
    f_j = \min (f_{1j}, f_{2j})
\]

Then branch on variable \( x_m \) that satisfies:

\[
    f_m = \max \{ f_j \}
\]
Strategy 4: Compute the sensitivity of the objective function with respect to the discrete variables. Then branch on the variable with the highest sensitivity. In a linear programming problem, this means to select a variable with the largest cost coefficient.

We have used strategy 3 in our implementation of the branch-and-bound method. It has been seen by Gupta (1980) that selecting the most fractional variable results in the best performance of the branch-and-bound method. Strategy 3 generates subproblems that are quite far from the continuous optimum. This results in getting to an incumbent solution that fathoms a number of unexplored nodes (Myers, 1984).

5.3.5 Example - Design of a Gear Train

The following example is taken from the work of Lee (1983). It later appeared in Refs. (Sandgren, 1990b; Fu et al., 1989). The objective in the design is to produce a gear ratio of a compound gear train as close to 1/6.931. For each gear, the number of teeth can lie between 12 to 60.

![Diagram of a compound gear train]

Fig. 5.3: Compound Gear Train
Let,

\[ x = [T_d, T_b, T_a, T_f] = [x_1, x_2, x_3, x_4] \]

where \( T_a, T_b, T_d, T_f \) are the number of teeth on gears A, B, D, and F, respectively.

Then the optimization problem is expressed as:

\[
\text{Min } F(x) = \left[ \frac{1}{6.931} \left( \frac{T_d T_b}{T_a T_f} \right) \right]^2 = \left[ \frac{1}{6.931} \left( \frac{x_1 x_2}{x_3 x_4} \right) \right]^2
\]

\[ 12 \leq x_i \leq 60 \quad i = 1, 2, 3, 4 \]

The continuous optimum is first obtained as \( x^* = (16.117, 12.74, 34.03, 41.83) \) with the objective function value of 7.44 E-13. Fig. 5.4 shows the branch-and-bound tree search required to reach the discrete optimum. The objective function value is given next to each node. It should be noted that since we dealing with values in the range of 1E-6 to 1E-12, there are some nodes where the parent node has a higher objective function value than its branches. Such a situation occurs due to the fact that the convergence tolerances for the GRG code are of the order of 1E-6.

The branch-and-bound search shown in Fig. 5.4 terminates on Node 8, where all the integer requirements for the gear teeth are satisfied. As soon as Node 8 is generated, it is marked as an incumbent solution. A pruning operation is done and all unexplored nodes with objective function values higher than the incumbent are marked as fathomed. The pruning operation eliminates unexplored nodes 1, 3, and 8. Thus, we are left with only the incumbent solution which is then the discrete optimum. It should be noted that the node numbers on the search tree are used only for the purpose of illustration. In the actual computer program, the node numbers will be different since we always use the fathomed slots to store new problems before generating a new node.

Table 5.1 summarizes the results of the branch-and-bound search. It is clear from the Table 5.1 that we have generated a better discrete optimum than the one found by earlier researchers.
Fig. 5.4: Branch-and-bound search tree for the compound gear train
Table 5.1  Results from the discrete optimization of the gear train

<table>
<thead>
<tr>
<th>Design Variables</th>
<th>Starting Values</th>
<th>Optimum (Sandgren, 1990)</th>
<th>Optimum (Fu et al., 1989)</th>
<th>Optimum Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1 (T_d)$</td>
<td>18</td>
<td>18</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>$x_2 (T_b)$</td>
<td>15</td>
<td>22</td>
<td>29</td>
<td>13</td>
</tr>
<tr>
<td>$x_3 (T_a)$</td>
<td>33</td>
<td>45</td>
<td>47</td>
<td>34</td>
</tr>
<tr>
<td>$x_4 (T_f)$</td>
<td>41</td>
<td>60</td>
<td>59</td>
<td>42</td>
</tr>
<tr>
<td>$F(x)$</td>
<td>0.03056</td>
<td>5.7 E-6</td>
<td>4.5 E-6</td>
<td>1.9 E-6</td>
</tr>
<tr>
<td>Gear Ratio</td>
<td>0.199556</td>
<td>0.146666</td>
<td>0.146411</td>
<td>0.145658</td>
</tr>
<tr>
<td>%Error</td>
<td>38.3</td>
<td>1.65</td>
<td>1.47</td>
<td>0.956</td>
</tr>
</tbody>
</table>
Chapter VI

DESIGN SHELL FRAMEWORK

6.0 Introduction

This chapter describes the computer implementation of the design shell based on the concepts and theory presented in the earlier chapters. Emphasis is placed on the following aspects:

a) Providing a generic framework for design which is not specific to any mechanical system.

b) Ease of creating "what-if" scenarios. The user should be able to change a number of design specifications and alternatives in an interactive manner. Such a requirement forces the shell to be extremely fast in doing numerical computations.

c) An intuitive and user-friendly interface that provides a simplified man-machine interaction and a logical environment for design.

The rest of the chapter describes the various building blocks of the design shell. Since the design shell has to be configured for a particular mechanical system before it can be used, a section of the chapter describes the configuration process in detail.
6.1 Design Shell Modules

The design shell is composed of several modules. In a broad sense, these can be classified into the following:

1) Constraint management module
2) Constraint satisfaction module
3) Optimization module
4) Post Processing module
5) Help module
6) Equations module
7) Graphics module

The constraint management module ensures that all equalities and inequalities are satisfied at any design stage. It is responsible for determining the solution sequence whenever any change is made to the design vector. The constraint satisfaction module provides an interface to the nonlinear equation solvers described in Chapter 5. The optimization module is an implementation of the GRG method for continuous optimization and branch-and-bound method for discrete optimization. It works with the nonlinear solver to obtain values of the objective function and constraints. The post processing module is a collection of routines which help the user with additional information about the design. The help module is used to provide context sensitive help and is available at any stage of the design. This module gets its information from the help files.

The other two modules, namely the equations module and the graphics module, are specific to the mechanical system being designed. These modules are written by the person who configures the shell (configurer). Both the modules are written in FORTRAN and are described in detail in the configuration section.

6.2 Design Shell User Interface

The design shell provides the facility to be configured for designing multiple machine elements or various types of the same machine element (for example a spur gear, helical gear, and a bevel gear are all gear types). These are referred to as design cases. The
design selection module provides a menu of the available selections to the user. It obtains
the data about the cases from the database which is specific to a particular configuration.

The graphical user interface is the heart of the design shell. Once the case is defined, the
module draws a "spreadsheet" of the design variables through which inputs and outputs are
handled. This module coordinates the complete design phase. In addition, it provides a
number of utilities that can help the user perform the design with ease.

Figure 6.1 shows a typical screen from an interactive design session. It is comprised of the
design variable spreadsheet, a graphics window (which appears only if the user so
desires), a message region and the menu header. These are the key elements in the user
interface, which is primarily mouse driven. The module runs on an event loop; waiting for
the user to make a selection and then transferring control to the corresponding routine to
perform the selected task.

![Variable Spreadsheet]

**Fig. 6.1:** A typical screen from the design shell
The design shell's menu structure is shown in Figure 6.2. A brief description of the functions of the various menu items is presented below. All data is file oriented, and hence the program is devoid of any hard-wired details.

![Menu Structure](image)

**Fig. 6.2: Design shell menu structure**

### 6.2.1 The FILE Menu

The FILE menu is the starting point of any design process. If a new design has been initiated, this menu processes the cases file from which the various design cases are read. This data is used to construct the sub-menus. Once a valid design case has been selected, the variable description file is processed from which the design variable names, units, default values, upper and lower limits, and occurrence matrices associated with the case are obtained. These files are configuration specific and must be provided by the configurer. Alternatively, if an old design is opened, all the relevant design information will be read from the specified file.
In addition, this menu has the provisions to store the current design in a file or exit the shell altogether. The various items in this menu and their respective functions are given below:

NEW  ( start a fresh design )
OPEN ( open a stored design )
SAVE ( save current design )
SAVE AS ( save current design as .. )
ARCHIVE ( save the current design in an archive )
QUIT ( exit the design shell )

The archive feature of the FILE menu is used to store designs into a database. Such a database is then used to generate good defaults values of the design variables (See Balasubramanian, 1991).

6.2.2 The EDIT Menu

The EDIT menu is accessible once the design case has been defined either through the new or the open item of the FILE menu. Basically this menu handles the various attributes of the variables such as default values, upper limits, lower limits, and units. There are also options available to remove an input variable from the spreadsheet or clear the spreadsheet altogether. The various menu items and their functions are stated below:

INITIAL VALUES (edit the default value of any variable )
UPPER LIMITS (edit the maximum value of any variable )
LOWER LIMITS (edit the minimum value of any variable )
CLEAR (clear the spreadsheet of all variable values )
REMOVE (remove a specified input variable and the dependent outputs )
CHANGE UNITS (change the unit of any variable )
6.2.3 The COMMANDS Menu

The COMMANDS menu is accessible only after a design case has been defined. Basically this menu provides the user with information regarding the design procedure. An option to view a scaled picture of the machine element is also available. The various menu items and their respective functions are given below:

- **DEPENDENCY** (information on dependency among variables)
- **SEN. MATRIX** (computes and displays the sensitivity matrix)
- **LIN. ADVICE** (provides linear advice)
- **COR. ADVICE** (provides correct advice)
- **GRAPH** (plot of any output variable against any input variable)
- **VIOLATIONS** (current values of all constraints)
- **SOLUTION SEQ** (information on the solution sequence)
- **SHOW PIC.** (display scaled drawing of machine element)

The sensitivity matrix computation is performed around a solution point by perturbing the input design variables one at a time and re-solving the system of equations. Each time, the ratios of the resulting changes in the other variables to the amount of perturbation are computed. This gives rise to a matrix of the form shown below. For example, let the constraint system consist of 5 variables and 3 equations. If \( x_1 \) and \( x_4 \) are inputs and \( x_2, x_3, \) and \( x_5 \) are computed, the sensitivity matrix would be:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>x1</th>
<th>x4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\partial x_2}{\partial x_1} )</td>
<td>( \frac{\partial x_3}{\partial x_1} )</td>
<td>( \frac{\partial x_5}{\partial x_1} )</td>
</tr>
<tr>
<td>( \frac{\partial x_2}{\partial x_4} )</td>
<td>( \frac{\partial x_3}{\partial x_4} )</td>
<td>( \frac{\partial x_5}{\partial x_4} )</td>
</tr>
</tbody>
</table>

The ratios denote the rates at which an output variables change with respect to the inputs at the current design point.
The sensitivity matrix can be used to make linearized forecasts and thus give the user some advice. In order to change an output variable, $x_j$, by an amount, $\Delta x_j$, we can compute the linearized change, $\Delta x_i$, in the input variable $x_i$, as follows:

$$\Delta x_i = \left( \frac{\partial x_j}{\partial x_i} \right) \Delta x_j$$

The two forms of advice options are provided the user with a guidance mechanism before making changes in the design specifications. When, any of the advice option is selected, the user is prompted to select an output variable whose value needs to be changes. In case of the Linear Advice option, a linearized forecast is presented indicating the changes that should be made to the input variables to achieve the desired value of the selected output. In the Correct Advice option, several subproblems are solved and actual changes in the inputs are displayed. The Correct Advice option is definitely computationally intensive since it changes the input/output assignment for every subproblem that is solved.

6.2.4 The OPTIMIZE Menu

The OPTIMIZE menu is used to select a variable to be optimized. There are three items under the menu.

- **Min/Max** - Allows the user to specify if the variable has to be minimized or maximized.
- **Variable** - Selects and optimizes a design variable.
- **Results** - This menu item is used to get details about the optimization process. It gives a summary of the number of function calls, number of line searches, etc. needed to converge to the optimum.
6.2.5 The UTILITIES Menu

The Utilities Menu is provided for non-design related functions. There are three items under the menu:

Help - This option reads a help file and provides textual information to guide the user. Currently this option is only available on the VAX 8550. Two help files are used with design shell. One file describes the functions of the design shell and the other file is specific to the mechanical system that has been configured.

Snapshot - Provides a high quality screen dump from the program. This option is available on only the VAX 8550.

About DS - This menu item describes the design shell version being used.

6.3 The Equations Module

The equations module contains the equations for all of the design cases. These equations are written in terms of generic variables. However, this makes it difficult for the configurer to code and edit the module. Hence, at the start of each set of equations, the generic variables should be copied into meaningful variable names and the equations written in terms of the latter.

The equations module is invoked from the constraint satisfaction module whenever a constraint function must be evaluated. It is also used when the nonlinear solver computes the values of output variables by solving simultaneous equations. All of the equations are coded in a subroutine called EQUNS and the format of that subroutine is explained in the configuration section.
6.4 The Graphics Module

The graphics module is another configuration specific module. It is invoked to display a scaled, parameterized drawing in the graphics viewport. The argument list for this module consists of the case number and the design variable array. It is entirely up to the configurator's discretion how this module handles graphics. The module could be a dummy routine if no drawing is necessary. However, a routine called IOCDRW (even if it is just a dummy routine) must be present.

6.5 Implementation details

In the current form the design shell is restricted to handle up to 80 variables in 60 equations. The program is completely written in FORTRAN 77 and depends on two additional libraries. These libraries are PLTPAK which provides the necessary graphics support and UTLPAK which contains all the nonlinear solver routines. Since PLTPAK is a device-independent graphics package, the design shell is also device independent. At present, the design shell has been successfully ported to several computer platforms which include VAX/VMS, HP workstations running UNIX, IBM PCs running DOS and Macintosh II computers.

Several different mechanical systems have been configured under the design shell. Some of them include springs, brakes, clutches, flywheels, powerscrews, keys, pins, couplings, and gears. The configuration of a simple torsion bar spring is presented in the next section and that of gears is described in Chapter 7.

6.6 Design Shell Configuration

This section describes how to configure a specific machine element to work under the design shell. Let us consider the example used in Chapter 3 to design a torsion bar spring. The equations must first be coded in FORTRAN. Fig. 6.3 shows the FORTRAN code needed to declare all the constraints. Note the use of named variables to transfer values from a generic x vector. Such a transfer makes the code easier to read and debug.
SUBROUTINE EQUNS (IEQ, NMAX, X, VALF, IERF)
IMPLICIT DOUBLE PRECISION (A-H, O-Z)
INTEGER IEQ, NMAX, IERF
DOUBLE PRECISION VALF
DOUBLE PRECISION X(NMAX)
COMMON /ACASE/ ICASE

C IERF = 0
C PI = 4.0*ATAN (1.0)
C convert the variables into named entities
C TAU = X(1)
C G = X(2)
C TORQUE = X(3)
C DIA = X(4)
C RL = X(5)
C TWIST = X(6)
C SF = X(7)
C RJ = X(8)
C STIFF = X(9)
C SRATE = X(10)
C VOLUME = X(11)

C IF (IEQ .EQ. 1) THEN
C    twist angle
C     VALF = TWIST*G*RJ - TORQUE*RL
C ELSEIF (IEQ .EQ. 2) THEN
C    stress
C     VALF = 2*TAU*RJ - TORQUE*DIA*SF
C ELSEIF (IEQ .EQ. 3) THEN
C    moment of inertia
C     VALF = RJ - PI*DIA**4/32
C ELSEIF (IEQ .EQ. 4) THEN
C    volume
C     VALF = VOLUME - 0.25*PI*DIA*DIA*RL
C ELSEIF (IEQ .EQ. 5) THEN
C    stiffness
C     VALF = TORQUE - TWIST*STIFF
C ELSEIF (IEQ .EQ. 6) THEN
C    stress rate
C     VALF = TAU - TWIST*SRATE
ENDIF
RETURN
END

Fig. 6.3: FORTRAN code for the torsion bar design
Next, we need to generate configuration-specific data files. There are five files that are needed to run the design shell. These are:

1. Cases file
2. Variable description file
3. Design Archive file
4. Shell Help file
5. Help file specific to the machine component being designed.

6.6.1 Cases File

The purpose of this file is to provide information for setting up the menus. The file for the torsion bar spring example used in this chapter is shown below with each entry explained.

'EXAMPLE' Title that appears in the About DS panel
1 Total number of cases
1 Number of categories into which cases are divided
'Springs' Name of the first category
1 Number of cases under this category
'Torsion bar' Name of the first case

This file is in two levels, first the categories and then the cases in each category. The need for this hierarchical structure is not evident from this example but when the number of cases is large, a classification of cases is helpful. Figure 6.4 shows a hierarchical structure that could be used for brakes and clutches.
6.6.2 Variable Description File

This first line of the file contains the total number of cases. The following information is contained in this file for each design case:

a) Begin case marker (# sign) followed by the case number on the next line.
b) Case title.
c) Number of design variables (N).
d) Number of design constraints (M). Here the constraints include all equality and inequality constraints. The constraints manager in the design shell has been structured so that the inequalities can be specified by the user in the same manner as the equations are structured. The constraint manager assumes that the user defined equations routine (subroutine EQUNS) contains the inequalities at the end of all equations. It is also assumed that the numerical values for the inequality constraints are nonnegative when the constraints are satisfied.

e) Number of inequality constraints. It is important that all the inequalities be specified at the end of all the equality constraints.
f) Maximum number of equations that are simultaneously solvable,

g) Variable name, Spreadsheet position, X position, Variable type.
The next line contains the Unit, Default value, Lower Limit, Upper Limit
Number of Equations in which the variable appears, The equation numbers themselves,
and the nonlinear occurrence matrix entry in the same order as the equation numbers
appear.

h) Constraint names

Each case is separated by a line with a single character (#). Fig. 6.5 shows the variable
description file for the torsion bar spring design case.

A number of variable types have been created to identify the special nature of the variables.
Variable type designation is helpful to distinguish between a continuous and a discrete
variable. The type of the variable is also used to define variables that have to be necessarily
input before any design can be done. Variable type number 5 is used to indicate variables
that are necessarily computed. Such variables may lead to solution problems when
designated as input variables. A judicious assignment of variable types is an essential part
of the configuration process. The following numbers are used to indicate different variable
types:

<table>
<thead>
<tr>
<th>Variable type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Continuous variable</td>
</tr>
<tr>
<td>2</td>
<td>Integer variable</td>
</tr>
<tr>
<td>5</td>
<td>Output only variable</td>
</tr>
<tr>
<td>9</td>
<td>Slack variable</td>
</tr>
<tr>
<td>12</td>
<td>Discrete variable</td>
</tr>
<tr>
<td>13</td>
<td>Necessarily input - continuous</td>
</tr>
<tr>
<td>14</td>
<td>Necessarily input - integer</td>
</tr>
<tr>
<td>15</td>
<td>Necessarily input - discrete</td>
</tr>
</tbody>
</table>
1
#
1 'Torsion Bar Design'
11
6
0
5
'Max. Stress', 1, 1, 12
'MATERL.DAT', 'STRS'
'psi', 0.1890E+06, 1.000, 0.1000E+11
2 2 6 1 1
'Shear Modulus', 2, 2, 1
'MATERL.DAT', 'SMOD'
'psi', 0.9615E+07, 100.0, 0.1000E+11
1 1 1
'Max. Torque Load', 3, 3, 1
'lb-in', 100.0, 1.000, 0.1000E+06
3 1 2 5 2 2 1
'Diameter', 4, 4, 1
'in', 0.1750, 0.1000E-01, 0.1000E+05
3 2 3 4 1 4 2
'Active Length', 5, 5, 1
'in', 5.000, 0.1000E-01, 0.1000E+05
2 1 4 1 1
'Twist', 6, 6, 1
'rad', 0.4690, 0.1000E-04, 100.0
3 1 5 6 2 2 2
'Safety Factor', 7, 7, 1
', 2.000, 1.0, 100.0
1 1 1
'Moment of Inertia', 8, 8, 1
'in**4', 0.9200E-04, 0.1000E-09, 0.1000E+11
1 3 1
'Stiffness', 9, 9, 1
'lb-in/rad', 214.0, 0.1000E-01, 0.1000E+11
1 5 1
'Stress Rate', 10, 10, 1
'psi/rad', 0.4033E+06, 100.0, 0.1000E+11
1 6 1
'Volume', 11, 11, 1
'in**3', 0.1200, 0.1000E-09, 0.1000E+07
1 4 1
'Twist Angle'
'Stress'
'Moment of Inertia'
'Volume'
'Stiffness'
'Stress rate'

Fig 6.5: Variable description file for torsion bar design
All discrete variable values are read from a file. The file is separated by using keywords. For example, let the Shear Modulus be a discrete material property. Then the file entries for Shear Modulus are given in Fig. 6.6 (a). Note the use of variable type 12 to indicate that the values for the variable are to be read from a file. The filename and the corresponding keyword appear on the next line. The entries in the discrete data file MATERLDAT for the discrete material property are given in Fig. 6.6 (b).

```
'Shear Modulus', x, x, 12
'MATERLDAT', 'SMOD'
rest of the variable information
```

(a) Discrete variable designation

```
'SMOD'
4
'Material 1', value 1
'Material 2', value 2
'Material 3', value 3
'Material 4', value 4
```

(b) Discrete data file format

Fig. 6.6: Discrete variable handling using data files
6.6.3 Archive file

The archive file is used to store designs that the user thinks might be helpful later. The archived designs are used to generate initial values for the nonlinear solver by comparing the current design to an existing design in the archive file. The reader is referred to the work by Balasubramanian (1991) for further details.

6.6.4 Help Files

The help files are text files that provide textual information about the design shell and the mechanical system being designed. The design shell file is common to all the configurations and provides help about the functions of the various menu items. The second file is specific to the machine element that has been configured. It provides information about the design variables and constraints used in the equations module. This file is supplied by the configurer.

The format of the help files is documented in the PLTPAK manual (1991) under the KHELP routine.

6.6.5 File of File Names

This file contains the names of all the files required by the design shell listed in a particular sequence as shown below:

- Cases file
- Archive file
- Variable description file
- Help file #1
- Help file #2

This file itself has to be named FILES.DAT.
6.7 Design Shell Examples

This section presents hardcopies from some machine elements configured with the Design Shell. Section 6.7.1 presents a simulated design of a torsion bar spring. Section 6.7.2 considers the optimization of a torsion bar spring. Section 6.7.3 presents optimization of a helical compression spring with two different objective functions.

6.7.1 Design of a Torsion Bar Spring

The mathematical model and an example design session for a torsion bar spring was presented in Section 3.6. This section presents the actual screen copies of the program run. Fig. 6.7 shows the New Specification of $x_{11}$ (volume), $x_3$ (torque load), and $x_9$ (torsional stiffness). The input variables on the spreadsheet are indicated with an 'i' marker after the units column. It can be seen that $x_6$ (twist) can be computed based on the current inputs. Fig. 6.8 shows the New Specification for $x_1$ (shear strength). We can then compute $x_{10}$ (stress rate). Fig. 6.9 depicts the New Specification of $x_2$ (shear modulus) and then the rest of the variables get solved. The solution sequence is shown in Fig. 6.10. The solution sequence shows that we have to solve 3 equations simultaneously to compute the values of diameter, length, and moment of inertia. We next respecify the value of $x_1$ (shear strength) from 1.89E5 to 2E5. Fig. 6.11 shows the respecification. Notice that only the values of $x_7$ (safety factor) and $x_{10}$ (stress rate) get changed. Fig. 6.12 shows how the unspecification of $x_1$ leads to the deletion of $x_7$ and $x_{10}$. Next we look at the Reverse Specification of $x_7$ (safety factor). Fig. 6.13 shows the backwardly dependent inputs of $x_7$. We then release the input of $x_9$ (torsional stiffness) and specify the value of $x_7$ (safety factor) as 1.0. Fig. 6.14 shows that the new problem converges to an extraneous root where the diameter is zero. Fig. 6.15 shows that the initial value of diameter is changed to 0.5. Fig. 6.16 shows that the problem then converges to a good solution where diameter is 0.2942.
### TORSION BAR SPRING - SOLID CYL.

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter</th>
<th>Unit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shear Strength</td>
<td>psi</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Shear Modulus</td>
<td>psi</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Torque Load</td>
<td>lb-in</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>Diameter</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Length</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Twist</td>
<td>rad</td>
<td>0.1</td>
</tr>
<tr>
<td>7</td>
<td>Safety Factor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Moment of Inertia</td>
<td>in-in^4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Torsional Stiffness</td>
<td>lb-in/ra</td>
<td>10000</td>
</tr>
<tr>
<td>10</td>
<td>Stress Rate</td>
<td>psi/rad</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>Volume</td>
<td>in-in^3</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 6.7: New Specification of $x_{11}$, $x_3$, and $x_9$

### TORSION BAR SPRING - SOLID CYL.

<table>
<thead>
<tr>
<th>No.</th>
<th>Parameter</th>
<th>Unit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shear Strength</td>
<td>psi</td>
<td>$1.89E+05$</td>
</tr>
<tr>
<td>2</td>
<td>Shear Modulus</td>
<td>psi</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Torque Load</td>
<td>lb-in</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>Diameter</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Length</td>
<td>in</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Twist</td>
<td>rad</td>
<td>0.1</td>
</tr>
<tr>
<td>7</td>
<td>Safety Factor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Moment of Inertia</td>
<td>in-in^4</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>Torsional Stiffness</td>
<td>lb-in/ra</td>
<td>10000</td>
</tr>
<tr>
<td>10</td>
<td>Stress Rate</td>
<td>psi/rad</td>
<td>$1.89E6$</td>
</tr>
<tr>
<td>11</td>
<td>Volume</td>
<td>in-in^3</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 6.8: New Specification of $x_1$
Fig. 6.9: New Specification of $x_2$

Fig. 6.10: Solution sequence
### Torsion Bar Spring - Solid Cyl

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Unit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shear Strength</td>
<td>psi</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Shear Modulus</td>
<td>psi</td>
<td>3.615E+06</td>
</tr>
<tr>
<td>3</td>
<td>Torque Load</td>
<td>lb-in</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>Diameter</td>
<td>in</td>
<td>0.4879</td>
</tr>
<tr>
<td>5</td>
<td>Length</td>
<td>in</td>
<td>5.3408</td>
</tr>
<tr>
<td>6</td>
<td>Twist</td>
<td>rad</td>
<td>0.1</td>
</tr>
<tr>
<td>7</td>
<td>Safety Factor</td>
<td></td>
<td>4.5608</td>
</tr>
<tr>
<td>8</td>
<td>Moment of Inertia</td>
<td>in^4</td>
<td>0.0056</td>
</tr>
<tr>
<td>9</td>
<td>Torsional Stiffness</td>
<td>lb-in/ra</td>
<td>10000</td>
</tr>
<tr>
<td>10</td>
<td>Stress Rate</td>
<td>psi/rad</td>
<td>2.0E6</td>
</tr>
<tr>
<td>11</td>
<td>Volume</td>
<td>in^3</td>
<td>1</td>
</tr>
</tbody>
</table>

**Fig. 6.11: Respecification of x₁**

### Torsion Bar Spring - Solid Cyl

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Unit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shear Strength</td>
<td>psi</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Shear Modulus</td>
<td>psi</td>
<td>9.615E+06</td>
</tr>
<tr>
<td>3</td>
<td>Torque Load</td>
<td>lb-in</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>Diameter</td>
<td>in</td>
<td>0.4879</td>
</tr>
<tr>
<td>5</td>
<td>Length</td>
<td>in</td>
<td>5.3408</td>
</tr>
<tr>
<td>6</td>
<td>Twist</td>
<td>rad</td>
<td>0.1</td>
</tr>
<tr>
<td>7</td>
<td>Safety Factor</td>
<td></td>
<td>4.5608</td>
</tr>
<tr>
<td>8</td>
<td>Moment of Inertia</td>
<td>in^4</td>
<td>0.0056</td>
</tr>
<tr>
<td>9</td>
<td>Torsional Stiffness</td>
<td>lb-in/ra</td>
<td>10000</td>
</tr>
<tr>
<td>10</td>
<td>Stress Rate</td>
<td>psi/rad</td>
<td>2.0E6</td>
</tr>
<tr>
<td>11</td>
<td>Volume</td>
<td>in^3</td>
<td>1</td>
</tr>
</tbody>
</table>

**Fig. 6.12: Unspecification of x₁**
### Torsion Bar Spring - Solid Cyl.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shear Strength</td>
<td>psi</td>
<td>i</td>
<td>225</td>
</tr>
<tr>
<td>2</td>
<td>Shear Modulus</td>
<td>psi</td>
<td>i</td>
<td>3.615E+06</td>
</tr>
<tr>
<td>3</td>
<td>Torque Load</td>
<td>lb-in</td>
<td>i</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>Diameter</td>
<td>in</td>
<td></td>
<td>0.4879</td>
</tr>
<tr>
<td>5</td>
<td>Length</td>
<td>in</td>
<td></td>
<td>5.3488</td>
</tr>
<tr>
<td>6</td>
<td>Twist</td>
<td>rad</td>
<td></td>
<td>0.1</td>
</tr>
<tr>
<td>7</td>
<td>Safety Factor</td>
<td></td>
<td></td>
<td>4.5606+</td>
</tr>
<tr>
<td>8</td>
<td>Moment of Inertia</td>
<td>in=4</td>
<td></td>
<td>0.0656</td>
</tr>
<tr>
<td>9</td>
<td>Torsional Stiffness</td>
<td>lb-in/ra</td>
<td>i</td>
<td>10000</td>
</tr>
<tr>
<td>10</td>
<td>Stress Rate</td>
<td>psi/rad</td>
<td></td>
<td>2.0E6</td>
</tr>
<tr>
<td>11</td>
<td>Volume</td>
<td>in=3</td>
<td>i</td>
<td>1</td>
</tr>
</tbody>
</table>

Output Variable chosen for Input
5 Dependent Inputs Found
Select one of those indicated by <

---

**Fig. 6.13: Reverse Specification of x7**

### Torsion Bar Spring - Solid Cyl.

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Shear Strength</td>
<td>psi</td>
<td>i</td>
<td>225</td>
</tr>
<tr>
<td>2</td>
<td>Shear Modulus</td>
<td>psi</td>
<td>i</td>
<td>3.615E+06</td>
</tr>
<tr>
<td>3</td>
<td>Torque Load</td>
<td>lb-in</td>
<td>i</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>Diameter</td>
<td>in</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Length</td>
<td>in</td>
<td></td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>Twist</td>
<td>rad</td>
<td></td>
<td>0.162</td>
</tr>
<tr>
<td>7</td>
<td>Safety Factor</td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Moment of Inertia</td>
<td>in=4</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Torsional Stiffness</td>
<td>lb-in/ra</td>
<td>i</td>
<td>2132.196</td>
</tr>
<tr>
<td>10</td>
<td>Stress Rate</td>
<td>psi/rad</td>
<td></td>
<td>4.2E15</td>
</tr>
<tr>
<td>11</td>
<td>Volume</td>
<td>in=3</td>
<td>i</td>
<td>1</td>
</tr>
</tbody>
</table>

Lower limit violation for Diameter

---

**Activate(U)/Ignore(I)/Automate(A) ?**

**Fig. 6.14: Convergence to an extraneous root**
### Fig. 6.15: Changing initial values

<table>
<thead>
<tr>
<th></th>
<th>Initial Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Shear Stress</td>
<td>Diameter + 0.5</td>
</tr>
<tr>
<td>2 Shear Modulus</td>
<td>Length 5</td>
</tr>
<tr>
<td>3 Torque Load</td>
<td>Diameter 0.459</td>
</tr>
<tr>
<td>4 Diameter</td>
<td>Length 0.2</td>
</tr>
<tr>
<td>5 Length</td>
<td>Twist 2.0001</td>
</tr>
<tr>
<td>6 Twist</td>
<td>Moment of Inertia 214</td>
</tr>
<tr>
<td>7 Safety Factor</td>
<td>Torsional Stiffness 4.033E5</td>
</tr>
<tr>
<td>8 Moment of Inertia</td>
<td>Stress Rate</td>
</tr>
<tr>
<td>9 Torsional Stiffness</td>
<td>Volume OK</td>
</tr>
</tbody>
</table>

### Fig. 6.16: Convergence to the correct solution

<table>
<thead>
<tr>
<th></th>
<th>psi</th>
<th>lb-in</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Shear Strength</td>
<td>285</td>
<td>3.615E6</td>
</tr>
<tr>
<td>2 Shear Modulus</td>
<td>1000</td>
<td>0.294</td>
</tr>
<tr>
<td>3 Torque Load</td>
<td>14.7101</td>
<td>2.0001</td>
</tr>
<tr>
<td>4 Diameter</td>
<td>1</td>
<td>0.0007</td>
</tr>
<tr>
<td>5 Length</td>
<td>lb-in/rad</td>
<td>1</td>
</tr>
<tr>
<td>6 Twist</td>
<td>Stress Rate</td>
<td>9.615E4</td>
</tr>
<tr>
<td>7 Safety Factor</td>
<td>Volume</td>
<td></td>
</tr>
<tr>
<td>8 Moment of Inertia</td>
<td>Torsional Stiffness</td>
<td></td>
</tr>
<tr>
<td>9 Torsional Stiffness</td>
<td>psi/ rad</td>
<td></td>
</tr>
<tr>
<td>10 Stress Rate</td>
<td>in^3</td>
<td></td>
</tr>
<tr>
<td>11 Volume</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
6.7.2 Optimization of a Torsion Bar Spring

This section shows the optimization of a torsion bar spring. Let us assume that the following are already specified.

Shear strength = 2E5 psi
Shear modulus = 9.615E6 psi
Torque load = 1000 lb

We first set the lower limits for the optimization and minimize the volume of the torsion bar (See Fig. 6.17). Fig. 6.18 shows the optimum configuration. The design shell picks the length and safety factor as the decision variables and converges to the lower limits. The minimum volume is given as 0.3399.
Fig. 6.17: Setting lower limits for optimization

Fig. 6.18: Minimization of volume
6.7.3 Optimization of a Helical Compression Spring

A helical compression spring was configured under the design shell. The governing constraints are as follows:

f₁: Stress equation

\[ K_w = \frac{4C - 1}{4C - 4} + \frac{0.615}{C} \]

\[ \tau = \frac{16 \text{(S.F)} P R K_w}{\pi d^3} \]

f₂: Spring index equation

\[ C = \frac{2R}{d} \]

f₃: Deflection equation

\[ \delta = \frac{4NPC^4}{GR} \]

f₄: Stiffness equation

\[ K = \frac{P}{\delta} \]

f₅: Free length

\[ L_f = \delta + (N + 2)d \]

f₆: Weight
\[ W = \frac{\rho \pi^2 d^2 R N}{2} \]

**f7: Packaging volume**

\[ V = \frac{\pi}{4} (2R + d)^2 L_f \]

**f8: Energy**

\[ U = \frac{2P^2 NC^4}{GR} \]

**f9: Natural Frequency**

\[ \nu = \sqrt{\frac{K(386.0)}{\left(\frac{W}{3} + P\right)}} \]

**f10: Buckling load**

\[ t_1 = 2\pi^2 \left( \frac{1 - \frac{G}{E}}{1 + \frac{2G}{E}} \right) \left( \frac{2R}{L_f} \right)^2 \]

\[ t_2 = \begin{cases} 1 - \sqrt{i - t_1} & \text{if } t_1 \leq 1 \\ 1 & \text{if } t_1 > 1 \end{cases} \]

\[ P_{\alpha} = \frac{i}{2} K L_f \left( 1 - \frac{G}{E} \right) t_2 \]

where,

- **E** Elastic modulus
- **G** Shear modulus
- **\( \rho \)** Density
\( \tau \)  Failure stress
\( P \)  Load
\( R \)  Coil Radius
\( d \)  Wire diameter
\( \delta \)  Deflection
\( N \)  No. of coils
\( C \)  Spring Index
\( SF \)  Safety factor
\( K \)  Spring constant
\( L_f \)  Free length
\( W \)  Weight
\( V \)  Package volume
\( U \)  Stored Energy
\( \nu \)  Natural Frequency
\( P_{cr} \)  Buckling load

The helical compression spring example has 18 variables in 10 equations. Thus we have 8 degrees of freedom. Let us say that we already specify the material, load, and deflection on the spring. Fig. 6.19 shows that the specification of 6 input variables. That leads to the computation of spring constant. Thus, we have 2 degrees of freedom left. Next, we specify the lower limits for the design variables (See Fig. 6.20). Fig. 6.21 and Fig. 6.22 show the results of optimization if the weight of the spring is minimized. The final decision variables are \( x_9 \) (number of coils) and \( x_{11} \) (safety factor). As can be seen from the results, the optimum lies at the lower limits and two basis changes are required before reaching the optimum. It should be noted that discrete optimization was used to ensure that the number of coils is an integer.

Since, we can choose any unknown variable as an objective function for optimization, we next examine the minimization of \( x_{15} \) (free volume). Fig. 6.23 and Fig. 6.24 show the optimization results. Note that it required four basis changes to reach the optimum. The optimum configuration is different from the minimum weight case.
### Fig. 6.19: Specifications before optimization

<table>
<thead>
<tr>
<th>HELICAL COMPRESSION SPRING - ROUND WIRE</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic Modulus</td>
<td>psi</td>
</tr>
<tr>
<td>Shear Modulus</td>
<td>psi</td>
</tr>
<tr>
<td>Density</td>
<td>lb/in^3</td>
</tr>
<tr>
<td>Failure Stress</td>
<td>psi</td>
</tr>
<tr>
<td>Load P</td>
<td>lb</td>
</tr>
<tr>
<td>Coil Radius R</td>
<td>in</td>
</tr>
<tr>
<td>Wire Diameter</td>
<td>in</td>
</tr>
<tr>
<td>Deflection</td>
<td>in</td>
</tr>
<tr>
<td>No. of Coils M</td>
<td></td>
</tr>
<tr>
<td>Spring Index C</td>
<td></td>
</tr>
<tr>
<td>Safety Factor FS</td>
<td></td>
</tr>
<tr>
<td>Spring Constant k</td>
<td>lb/in</td>
</tr>
<tr>
<td>Free Length Lf</td>
<td>in</td>
</tr>
<tr>
<td>Weight W</td>
<td>lb</td>
</tr>
<tr>
<td>Free Volume V</td>
<td>in^3</td>
</tr>
<tr>
<td>Energy Stored U</td>
<td>lb-in</td>
</tr>
<tr>
<td>Natural Freq.</td>
<td>Hz</td>
</tr>
<tr>
<td>Buckling Load Per</td>
<td>lb</td>
</tr>
</tbody>
</table>

### Fig. 6.20: Setting up the lower limits

<table>
<thead>
<tr>
<th>HELICAL COMPRESSION SPRING - ROUND WIRE</th>
<th>Lower Limits</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Elastic Modulus</td>
<td>Elastic Modulus</td>
<td>100</td>
</tr>
<tr>
<td>Shear Modulus</td>
<td>Shear Modulus</td>
<td>100</td>
</tr>
<tr>
<td>Density</td>
<td>Density</td>
<td>0.01</td>
</tr>
<tr>
<td>Failure Stress</td>
<td>Failure Stress</td>
<td>100</td>
</tr>
<tr>
<td>Load P</td>
<td>Load P</td>
<td>10</td>
</tr>
<tr>
<td>Coil Radius R</td>
<td>Coil Radius R</td>
<td>0.1</td>
</tr>
<tr>
<td>Wire Diameter</td>
<td>Wire Diameter</td>
<td>0.05</td>
</tr>
<tr>
<td>Deflection</td>
<td>Deflection</td>
<td>0.0001</td>
</tr>
<tr>
<td>No. of Coils M</td>
<td>No. of Coils M</td>
<td>6</td>
</tr>
<tr>
<td>Spring Index C</td>
<td>Spring Index C</td>
<td>4</td>
</tr>
<tr>
<td>Safety Factor FS</td>
<td>Safety Factor FS</td>
<td>1</td>
</tr>
<tr>
<td>Spring Constant k</td>
<td>Spring Constant k</td>
<td>0.01</td>
</tr>
<tr>
<td>Weight W</td>
<td>Weight W</td>
<td>0.001</td>
</tr>
<tr>
<td>Free Length Lf</td>
<td>Free Length Lf</td>
<td>0.001</td>
</tr>
<tr>
<td>Free Volume U</td>
<td>Free Volume U</td>
<td>0.001</td>
</tr>
</tbody>
</table>

OK + CANCEL PgUp PgDn
<table>
<thead>
<tr>
<th>Elongation</th>
<th>psi</th>
<th>3.8E7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shear Modulus</td>
<td>psi</td>
<td>1.15E7</td>
</tr>
<tr>
<td>Density</td>
<td>lb/in^3</td>
<td>0.205</td>
</tr>
<tr>
<td>Fatigue Stress</td>
<td>psi</td>
<td>1.89E5</td>
</tr>
<tr>
<td>Load P</td>
<td>lb</td>
<td>60</td>
</tr>
<tr>
<td>Coil Radius R</td>
<td>in</td>
<td>0.2843</td>
</tr>
<tr>
<td>Wire Diameter</td>
<td>in</td>
<td>0.0624</td>
</tr>
<tr>
<td>Deflection</td>
<td>in</td>
<td>1</td>
</tr>
<tr>
<td>No. of Coils M</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Spring Index C</td>
<td></td>
<td>6.3001</td>
</tr>
<tr>
<td>Safety Factor FS</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Spring Constant K</td>
<td>lb/in</td>
<td>60</td>
</tr>
<tr>
<td>Free Length Lf</td>
<td>in</td>
<td>1.6591</td>
</tr>
<tr>
<td>Weight W</td>
<td>lbm</td>
<td>0.0163</td>
</tr>
<tr>
<td>Free Volume V</td>
<td>in^3</td>
<td>0.5523</td>
</tr>
<tr>
<td>Energy Stored U</td>
<td>lb-in</td>
<td>30</td>
</tr>
<tr>
<td>Natural Freq.</td>
<td>Hz</td>
<td>19.646</td>
</tr>
<tr>
<td>Buckling Load Pcr</td>
<td>lb</td>
<td>45.4899</td>
</tr>
</tbody>
</table>

Fig. 6.21: Minimization of weight

<table>
<thead>
<tr>
<th>Elongation</th>
<th>psi</th>
<th>3.8E7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shear Modulus</td>
<td>psi</td>
<td>1.15E7</td>
</tr>
<tr>
<td>Density</td>
<td>lb/in^3</td>
<td>0.205</td>
</tr>
<tr>
<td>Fatigue Stress</td>
<td>psi</td>
<td>1.89E5</td>
</tr>
<tr>
<td>Load P</td>
<td>lb</td>
<td>60</td>
</tr>
<tr>
<td>Coil Radius R</td>
<td>in</td>
<td>0.2843</td>
</tr>
<tr>
<td>Wire Diameter</td>
<td>in</td>
<td>0.0624</td>
</tr>
<tr>
<td>Deflection</td>
<td>in</td>
<td>1</td>
</tr>
<tr>
<td>No. of Coils M</td>
<td></td>
<td>6</td>
</tr>
<tr>
<td>Spring Index C</td>
<td></td>
<td>6.3001</td>
</tr>
<tr>
<td>Safety Factor FS</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>Spring Constant K</td>
<td>lb/in</td>
<td>60</td>
</tr>
<tr>
<td>Free Length Lf</td>
<td>in</td>
<td>1.6591</td>
</tr>
<tr>
<td>Weight W</td>
<td>lbm</td>
<td>0.0163</td>
</tr>
<tr>
<td>Free Volume V</td>
<td>in^3</td>
<td>0.5523</td>
</tr>
<tr>
<td>Energy Stored U</td>
<td>lb-in</td>
<td>30</td>
</tr>
<tr>
<td>Natural Freq.</td>
<td>Hz</td>
<td>19.646</td>
</tr>
<tr>
<td>Buckling Load Pcr</td>
<td>lb</td>
<td>45.4899</td>
</tr>
</tbody>
</table>

Fig. 6.22: Weight optimization results
### Fig. 6.23: Minimization of free volume

<table>
<thead>
<tr>
<th>HELICAL COMPRESSION SPRING - ROUND WIRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Elastic Modulus (psi)</td>
</tr>
<tr>
<td>2. Shear Modulus (psi)</td>
</tr>
<tr>
<td>3. Density (lb/in^3)</td>
</tr>
<tr>
<td>4. Failure Stress (psi)</td>
</tr>
<tr>
<td>5. Load (lb)</td>
</tr>
<tr>
<td>6. Coil Radius (in)</td>
</tr>
<tr>
<td>7. Wire Diameter (in)</td>
</tr>
<tr>
<td>8. Deflection (in)</td>
</tr>
<tr>
<td>9. No. of Coils</td>
</tr>
<tr>
<td>10. Spring Index C</td>
</tr>
<tr>
<td>11. Safety Factor SF</td>
</tr>
<tr>
<td>12. Spring Constant K (lb/in)</td>
</tr>
<tr>
<td>13. Free Length Lf (in)</td>
</tr>
<tr>
<td>14. Weight W (lb)</td>
</tr>
<tr>
<td>15. Free Volume V (in^3)</td>
</tr>
<tr>
<td>16. Energy Stored U (lb-in)</td>
</tr>
<tr>
<td>17. Natural Freq. (Hz)</td>
</tr>
<tr>
<td>18. Buckling Load Per (lb)</td>
</tr>
</tbody>
</table>

### Fig. 6.24: Volume optimization results

<table>
<thead>
<tr>
<th>HELICAL COMPRESSION SPRING - ROUND WIRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Elastic Modulus (psi)</td>
</tr>
<tr>
<td>2. Shear Modulus (psi)</td>
</tr>
<tr>
<td>3. Density (lb/in^3)</td>
</tr>
<tr>
<td>4. Failure Stress (psi)</td>
</tr>
<tr>
<td>5. Load (lb)</td>
</tr>
<tr>
<td>6. Coil Radius (in)</td>
</tr>
<tr>
<td>7. Wire Diameter (in)</td>
</tr>
<tr>
<td>8. Deflection (in)</td>
</tr>
<tr>
<td>9. No. of Coils</td>
</tr>
<tr>
<td>10. Spring Index C</td>
</tr>
<tr>
<td>11. Safety Factor SF</td>
</tr>
<tr>
<td>12. Spring Constant K (lb/in)</td>
</tr>
<tr>
<td>13. Free Length Lf (in)</td>
</tr>
<tr>
<td>14. Weight W (lb)</td>
</tr>
<tr>
<td>15. Free Volume V (in^3)</td>
</tr>
<tr>
<td>16. Energy Stored U (lb-in)</td>
</tr>
<tr>
<td>17. Natural Freq. (Hz)</td>
</tr>
<tr>
<td>18. Buckling Load Per (lb)</td>
</tr>
</tbody>
</table>
Chapter VII
CONSTRANT-BASED GEAR DESIGN

7.0 Introduction

This chapter discusses the design methodology for spur and helical gears in a manner suitable for implementation under a constraint management framework. All the equality and inequality constraints for gear design are given in this chapter. The complete constraint model is then implemented under the design shell. Such an implementation gives the gear designer an immense flexibility in the selection of known and unknown parameters. Some gear analyses presented in the chapter will show the benefits of using the design shell.

In general the designer seeks an optimum set of gears that will perform to the given system requirements. Since, the design shell is based on the Embedded Objective Function formulation, we can choose a number of objective functions. In general, for a given set of load and space requirements, we can try to design gears with minimum weight. Sometimes, if the center distance for the gears is not fixed, one can also minimize the size based on the center distance. Discreteness occurs in the design of gears in a number of ways. Firstly, we have integer variables, namely, the number of gear teeth. Secondly, we might want to limit ourselves to standard manufacturing dimensions i.e. standard diametral pitch and standard pressure angle.

Section 7.1 describes all the equality and inequality constraints used in the gear model. Section 7.2 presents the design shell formulation. Section 7.3 and Section 7.4 present examples of analyzing and designing gears using the design shell.
7.1 Gear Design Model Development

This section presents the mathematical formulae used to design and analyze spur and helical gears. Since spur gears form a subset of helical gears, all the mathematical formulation is presented for helical gears only. First, the formulae for basic gear geometry are given, followed by the AGMA derating factor calculations. Section 7.1.3 presents additional constraints used in gear design.

7.1.1 Geometry

Most of the basic formulae for helical gear geometry can be found in Dudley (1984). A summary of all the equations is presented in the following. The nomenclature is given in the list presented earlier in the thesis.

The gear ratio is defined as:

\[ m_G = \frac{N_G}{N_P} \]  

(7.1)

The relation between normal pressure angle \( \phi_n \), and the helix angle \( \psi \), the transverse pressure angle is given as:

\[ \tan \phi_t = \frac{\tan \phi_n}{\cos \psi} \]  

(7.2)

The transverse diametral pitch is given as:

\[ P_t = P_n \cos \psi \]  

(7.3)

The transverse circular pitch is:

\[ p_c = \frac{\pi}{P_t} \]  

(7.4)
The pitch diameters can be determined as follows:

\[ d = \frac{N_p}{P_t} \]  
\[ D = \frac{N_G}{P_t} \]  
(7.5)  
(7.6)

The center distance can be computed as:

\[ C_d = \frac{d + D}{2} \]  
(7.7)

The outside diameters can be computed as:

\[ d_o = d + \frac{2a_P}{P_t} \]  
\[ D_o = D + \frac{2a_G}{P_t} \]  
(7.8)  
(7.9)

where \( a_P \) and \( a_G \) are the addendum constants.

The root diameters can be found as:

\[ d_r = d - \frac{2b_P}{P_t} \]  
\[ D_r = D - \frac{2b_G}{P_t} \]  
(7.10)  
(7.11)

where \( b_P \) and \( b_G \) are the dedendum constants.

The base diameters can be computed as:

\[ d_b = d \cos \phi_t \]  
\[ D_b = D \cos \phi_t \]  
(7.12)  
(7.13)
The base helix angle can be found as:

$$\tan \psi_b = \tan \psi \cos \phi_t$$  \hspace{1cm} (7.14)

The transverse base pitch and the normal base pitch can be computed using the following relations:

$$p_b = \frac{\pi d_b}{N_p}$$  \hspace{1cm} (7.15)

$$p_{bN} = p_b \cos \psi$$  \hspace{1cm} (7.16)

The length of line of action is computed as:

$$Z = \rho_P + \rho_G - C_d \sin \phi_t$$  \hspace{1cm} (7.17)

where the $\rho_P$ and $\rho_G$ are the radii of curvature at the tip for the pinion and gear, respectively, and are defined as:

$$\rho_P = \sqrt{d_0^2 - d_b^2}$$

$$\rho_G = \sqrt{D_0^2 - D_b^2}$$  \hspace{1cm} (7.18)

The profile contact ratio is computed as:

$$m_p = \frac{Z}{p_b}$$  \hspace{1cm} (7.19)

The face contact ratio is found as:

$$m_F = \frac{F \tan \psi_b}{p_b}$$  \hspace{1cm} (7.20)

The arc tooth thicknesses at the pitch diameter are computed as:
\[
    t_p = \frac{p_c - B}{2} + 2 \left( \frac{a_p}{p_d} - \frac{h_k}{2} \right) \tan \phi \\
    t_g = \frac{p_c - B}{2} + 2 \left( \frac{a_g}{p_d} - \frac{h_k}{2} \right) \tan \phi
\]

where \( B = \) backlash, \( h_k = \) working depth.

The weight of the gear set can be approximated as follows:

\[
    W = \frac{\pi}{4} F \left( \gamma_p d^2 + \gamma_g D^2 \right)
\]  

(7.21)

The weight formula is just an approximation to the actual weight of the gear set as the gears are modelled as two disks with the diameter equal to the pitch diameter and the width equal to the face width.

7.1.2 AGMA Strength and Wear Calculations

Once the basic geometry of the gears is computed, a rating procedure similar to the one suggested by AGMA (1988) can be followed to determine the performance of the gear set under load. AGMA suggests two kinds of ratings, namely strength and wear. All of the derating factors used in the strength and wear rating equations address a particular aspect of gear design. A detailed description of each one of them and their calculation procedure is discussed in Refs. (AGMA, 1988; Agrawal, 1986). A brief description of each of these factors is as follows:

I  
- Wear geometry factor adjusts the calculated contact stress for the effects of Hertzian contact stresses. The Hertzian contact stress depend on the radii of curvature of the contacting profiles, helix angle, pressure angle and the gear ratio.

J  
- Strength geometry factor corrects the calculated bending stress to take into account the shape of the mating teeth, the position at which the most
damaging load is applied, load sharing between teeth, and the stress concentration due to the geometric shape.

\( K_a, C_a \) - Application factors correct the calculated stresses for the roughness or smoothness of operation of the gears. This factor makes allowance for momentary overload torques greater than those determined from the nameplate ratings.

\( K_L, C_L \) - Life factors adjust the allowable stresses to meet the minimum desired number of contact life cycles. The life factor is a function of the number of contact cycles and the Brinell Hardness number.

\( K_m, C_m \) - Load distribution factors correct the calculated stress for the effects of non-distribution of load along the lines of contact. They depend upon the alignment of axes of rotation, gear tooth manufacturing accuracy, tooth stiffness, elastic deflections of shafts, gearings and housings, tooth crowning, and Hertzian contact and bending deformations at the tooth surface.

\( K_R, C_R \) - Reliability factors.

\( K_S, C_S \) - Size factors reflect the non-uniformity of material properties and adjust the calculated contact stress. They depend on the tooth size, gear diameters, ratio of tooth size to gear diameter, and face width.

\( K_T, C_T \) - Temperature factors compensate the allowable stress for the adverse affects of high operating temperatures.

\( K_V, C_V \) - Dynamic factors adjust the calculated stresses for the effects of tooth spacing and profile errors, pitchline and rotational speeds, inertia and stiffness of the rotating elements, transmitted load per inch of face, and the tooth stiffness.
\( C_f \) - Surface condition factor modifies the calculated contact stress for the effects of surface finish (cutting, shaving, lapping, grinding, shot peening, etc.), residual stresses, and plasticity effects (work hardening).

\( C_H \) - Hardness ratio factor corrects the allowable contact stress for the effects of the gear ratio and the relative hardness between the pinion and the gear.

\( C_p \) - Elastic coefficient adjusts the calculated contact stress to take into account the elastic material properties of the gears such as Poisson's ratio and Young's modulus of elasticity.

**Bending Strength**: The AGMA uses the tensile bending stress at the root of the tooth in the strength criterion. The rating formula consists of three groups of terms: load, tooth size, and stress distribution. The tensile bending stress at the root of the tooth is computed as follows:

\[
 s_t = \frac{W_t K_s}{K_v} \frac{P_d}{F} \frac{K_s K_m}{J} 
\]

(7.22)

In the above formula, the tangential load, \( W_t \), is determined using the following equation:

\[
 W_t = \frac{2T_p}{d} 
\]

(7.23)

where \( T_p \) is the pinion torque given as:

\[
 T_p = \frac{63025 P}{n_p} 
\]

(7.24)

Using, the computed value of the tensile stress and the allowable bending stress, we can derive an expression for the safety factor for bending strength as follows:
\[ F_{S_t} = \frac{s_{at} \left( \frac{K_L}{K_T K_R} \right)}{s_t} \] (7.25)

**Pitting Strength**: The wear capacity of spur and helical gears is determined by the AGMA using the gear tooth's surface pitting durability. As with the basic strength rating, the surface durability formula can be broken down into three distinct groups of terms: load, tooth size, and stress distribution. The surface contact stress can be computed as:

\[ s_c = C_p \sqrt{\frac{W_c}{C_v}} \frac{C_s}{dF} \frac{C_m C_f}{I} \] (7.26)

<table>
<thead>
<tr>
<th>LOAD</th>
<th>TOOTH</th>
<th>STRESS</th>
<th>DISTRIBUTION</th>
</tr>
</thead>
</table>

Similar to the bending strength computation, we can derive an expression for the safety factor for contact stress as follows:

\[ F_{S_c} = \frac{s_{ac} \left( \frac{C_L C_H}{C_T C_R} \right)}{s_c} \] (7.27)

It should be noted that the stress computations are done separately for both pinion and gear.

### 7.1.3 Other Constraints

This section presents some more considerations helpful in designing high quality helical gears.

**Contact ratio**: To ensure a quiet mesh, there should be a certain minimum limit on the profile contact ratio \( (m_p) \). A value of 1.2 to 1.4 has been commonly stated. Gears should never be designed with contact ratios less than 1.2 since machining inaccuracies might
reduce the contact ratio even more, thereby increasing noise as well as the possibility of impact between the teeth.

Thus, the lower limit for \( m_p \) is designated as 1.2.

**Tooth Interference:** The next constraint imposed on the problem is that of tooth interference. Interference is defined as the contact of tooth profiles that are not conjugate. The actual effect of this contact is that the involute tip or face of the driven gear tends to dig out the the non-involute flank of the driver. Interference can be reduced by using a larger pressure angle. This makes the base circle smaller, thus increasing the involute profile of the tooth. The lower bound on the number of pinion teeth to avoid interference during mating is given as:

\[
(N_p)_{\text{Interference}} = \frac{2a_p \cos \psi}{\sin^2 \phi_t}
\]  
(7.28)

Thus the interference constraint can be written as:

\[
g_1: N_p - (N_p)_{\text{Interference}} \geq 0
\]  
(7.29)

**Undercutting:** When gears are produced by a generation process, a hob for example, interference can also occur during the manufacturing process. This is called undercutting and can severely weaken the tooth. Mitchiner et al. (1983) derived a minimum tooth number to avoid undercutting in hobbed spur gears. The undercutting constraint can be written as:

\[
(N_p)_{\text{Undercut}} = \frac{2(h_a + r_T (\sin \phi_n - 1))}{\sin^2 \phi_n}
\]  
(7.30)

Both \( h_a \) and \( r_T \) in the above expression are given for unit diametral pitch i.e. these are normalized values. The undercutting constraint can be written as:

\[
g_2: N_p - (N_p)_{\text{Undercut}} \geq 0
\]  
(7.31)
**Scoring**: Another form of gear failure is known as scoring, which results from excessive compressive stress (Dudley, 1984). If a gear set is let to run after pitting, excessive surface wear results which manifests itself in radial scratch lines. A PVT formula has been generally used as a design guideline to prevent scoring. The factors in PVT are as follows (Dudley, 1984):

\[ P = \text{Hertz contact pressure} \]
\[ V = \text{Sliding velocity in feet per second} \]
\[ T = \text{Distance along the line of action from the pitch point to the point where P is calculated.} \]

Often, the PVT formula is used at the tip of the gears. To compute the Hertz contact pressure, we use the following relations:

\[ P_P = 5740 \frac{T_P}{FZN_P} \frac{C \sin \phi_n}{\rho_P (C \sin \phi_t - \rho_P)} \]  
(7.32)
\[ P_G = 5740 \frac{T_P}{FZN_P} \frac{C \sin \phi_n}{\rho_G (C \sin \phi_t - \rho_G)} \]  
(7.33)

Then the PVT formula is given as:

\[ PVT_P = \frac{\pi n_P}{360} \left( 1 + \frac{N_p}{N_G} \right) \left( \rho_P - \frac{d \sin \phi_t}{2} \right)^2 P_P \]  
(7.34)
\[ PVT_G = \frac{\pi n_P}{360} \left( 1 + \frac{N_p}{N_G} \right) \left( \rho_G - \frac{D \sin \phi_t}{2} \right)^2 P_G \]  
(7.35)

A design limit of 1,500,000 has been suggested as a safe limit on PVT. Thus, we can represent this requirement by specifying upper limits on PVT\textsubscript{P} and PVT\textsubscript{G}.

**Facewidth constraint**: The face width of the gear set is usually proportional to the pinion pitch diameter to ensure that the tooth load is uniform. Gear sets having face width greater than the pitch diameter tend to have a non-uniform distribution of the load across the the face of the tooth because of the torsional deflection of the gear and shaft, machining
inaccuracies, and the necessity of maintaining very accurate and rigid bearing mountings (Dudley, 1984). Gears with a smaller facewidth tend to be quite large, require more space and are more expensive to manufacture. Earlier works (Savage et al., 1982; Carroll and Johnson, 1984) have taken the face width to pinion pitch diameter ratio, also called the aspect ratio, as 0.25. In general, for a given aspect ratio, $m_a$, we have the following constraint,

$$ g_1: F - m_a d \leq 0 $$

(7.36)

Discreteness Constraints: The following three variables were identified as the discrete design variables.

1. Number of pinion teeth, $N_p$
2. Number of gear teeth, $N_G$
3. Diametral pitch, $P_d$

Clearly, both the number of pinion and gear teeth have to be integers. Furthermore, the diametral pitch is generally selected from one of the standard manufacturing numbers and hence it is discrete. Dudley (1962) gives a table of standard diametral pitches (See Table 7.1).

### Table 7.1: Recommended Diametral Pitches (Dudley, 1962)

<table>
<thead>
<tr>
<th>Coarse pitch</th>
<th>Fine pitch</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0 4 12</td>
<td>20 48 120</td>
</tr>
<tr>
<td>2.25 6 16</td>
<td>24 64 150</td>
</tr>
<tr>
<td>2.5 8</td>
<td>32 80 200</td>
</tr>
<tr>
<td>3 10</td>
<td>40 96</td>
</tr>
</tbody>
</table>
7.2 Design Shell Formulation

In order to configure the helical gear design problem into the design shell, all of the equations presented in Section 7.1 are treated as equality constraints. To reduce the number of variables in the design and to simplify both the bending and pitting stress computation, we have taken most of the AGMA correction factors as unity. In the current formulation, we compute I, J, Kv, Cv, and Cp. The constraint network has 79 variables (including 2 slack variables) in 47 equality constraints and 2 inequality constraints. The stress constraints are specified by setting the lower limit for safety factors as 1.0. The configuration-specific data files required by the design shell are given in Appendix A. The FORTRAN subroutines that are used to describe the complete model are given as Appendix B. Appendix C presents the graphics subroutine for drawing a gear set.

7.3 Gear design examples

This section presents numerical examples from the helical gear module under the design shell. Though, the number of possible design cases is huge, we present three representative examples to show the flexibility in specifying the inputs.

7.3.1 Example 1

Let us first use the design shell to analyze a helical gear set. Since there are 79 variables in 49 equations, we have 30 degrees of freedom. The set of inputs taken from Dudley (1984) is given in Table 7.2. The corresponding outputs are given in Table 7.3.
Table 7.2  Input variables for an example analysis

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of pinion teeth</td>
<td>25</td>
</tr>
<tr>
<td>Number of gear teeth</td>
<td>96</td>
</tr>
<tr>
<td>Normal diametral pitch</td>
<td>8.466667</td>
</tr>
<tr>
<td>Normal pressure angle</td>
<td>20 deg</td>
</tr>
<tr>
<td>Helix angle</td>
<td>15 deg</td>
</tr>
<tr>
<td>Face width</td>
<td>3.07 in</td>
</tr>
<tr>
<td>Backlash</td>
<td>0</td>
</tr>
<tr>
<td>Working depth constant</td>
<td>2</td>
</tr>
<tr>
<td>Aspect ratio</td>
<td>2</td>
</tr>
<tr>
<td>Elastic modulus (pinion)</td>
<td>$30 \times 10^6$ psi (205 GPa)</td>
</tr>
<tr>
<td>Poisson's ratio</td>
<td>0.3</td>
</tr>
<tr>
<td>Bending strength (pinion)</td>
<td>60 ksi (414 MPa)</td>
</tr>
<tr>
<td>Contact strength (pinion)</td>
<td>200 ksi (1.38 GPa)</td>
</tr>
<tr>
<td>Density (pinion)</td>
<td>0.3</td>
</tr>
<tr>
<td>Elastic modulus (gear)</td>
<td>$30 \times 10^6$ psi (205 GPa)</td>
</tr>
<tr>
<td>Poisson's ratio (gear)</td>
<td>0.3</td>
</tr>
<tr>
<td>Bending strength (gear)</td>
<td>60 ksi (414 MPa)</td>
</tr>
<tr>
<td>Contact strength (gear)</td>
<td>200 ksi (1.38 GPa)</td>
</tr>
<tr>
<td>Density (gear)</td>
<td>0.3</td>
</tr>
<tr>
<td>Outside diameter (pinion)</td>
<td>3.293</td>
</tr>
<tr>
<td>Root diameter (pinion)</td>
<td>2.738</td>
</tr>
<tr>
<td>Outside diameter (gear)</td>
<td>11.974</td>
</tr>
<tr>
<td>Root diameter (gear)</td>
<td>11.419</td>
</tr>
<tr>
<td>Transmitted power</td>
<td>20 hp</td>
</tr>
<tr>
<td>Operating speed</td>
<td>1260.5 rpm</td>
</tr>
<tr>
<td>Cutter radius</td>
<td>$0.3/P_d$</td>
</tr>
<tr>
<td>Hob addendum</td>
<td>$1.35/P_d$</td>
</tr>
<tr>
<td>Hob shift</td>
<td>0</td>
</tr>
<tr>
<td>Hob protuberance</td>
<td>0</td>
</tr>
<tr>
<td>AGMA Quality number</td>
<td>7</td>
</tr>
<tr>
<td>Output Variables for an Example Analysis</td>
<td></td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td></td>
</tr>
<tr>
<td>Gear ratio</td>
<td>3.84</td>
</tr>
<tr>
<td>Addendum constant (pinion)</td>
<td>0.9995</td>
</tr>
<tr>
<td>Dedendum constant (pinion)</td>
<td>1.35</td>
</tr>
<tr>
<td>Addendum constant (gear)</td>
<td>0.9969</td>
</tr>
<tr>
<td>Dedendum constant (gear)</td>
<td>1.3526</td>
</tr>
<tr>
<td>Center distance</td>
<td>7.3977 in</td>
</tr>
<tr>
<td>Circular pitch</td>
<td>0.3841</td>
</tr>
<tr>
<td>Transverse diametral pitch</td>
<td>8.1782</td>
</tr>
<tr>
<td>Transverse base pitch</td>
<td>0.3595</td>
</tr>
<tr>
<td>Normal base pitch</td>
<td>0.3472</td>
</tr>
<tr>
<td>Axial pitch</td>
<td>1.4336</td>
</tr>
<tr>
<td>Transverse pressure angle</td>
<td>20.6469 deg</td>
</tr>
<tr>
<td>Base helix angle</td>
<td>14.076 deg</td>
</tr>
<tr>
<td>Addendum (pinion)</td>
<td>0.118</td>
</tr>
<tr>
<td>Dedendum (pinion)</td>
<td>0.1595</td>
</tr>
<tr>
<td>Addendum (gear)</td>
<td>0.118</td>
</tr>
<tr>
<td>Dedendum (gear)</td>
<td>0.1595</td>
</tr>
<tr>
<td>Pitch diameter (pinion)</td>
<td>3.0569</td>
</tr>
<tr>
<td>Base diameter (pinion)</td>
<td>2.8606</td>
</tr>
<tr>
<td>Pitch diameter (gear)</td>
<td>11.7385</td>
</tr>
<tr>
<td>Base diameter (gear)</td>
<td>10.9846</td>
</tr>
<tr>
<td>Face contact ratio</td>
<td>2.141</td>
</tr>
<tr>
<td>Profile contact ratio</td>
<td>1.6417</td>
</tr>
<tr>
<td>Load sharing ratio</td>
<td>0.6193</td>
</tr>
<tr>
<td>Length of line of action</td>
<td>0.5901</td>
</tr>
<tr>
<td>Pinion torque</td>
<td>1000 lb-in</td>
</tr>
<tr>
<td>Pitchline velocity</td>
<td>1008.777 fpm</td>
</tr>
<tr>
<td>Tangential load</td>
<td>653.994 lb</td>
</tr>
<tr>
<td>$C_p$</td>
<td>2290.604</td>
</tr>
<tr>
<td>$K_v$, $C_v$</td>
<td>0.7476</td>
</tr>
<tr>
<td>I-factor</td>
<td>0.2115</td>
</tr>
<tr>
<td>J-factor (pinion)</td>
<td>0.5243</td>
</tr>
<tr>
<td>J-factor (gear)</td>
<td>0.5926</td>
</tr>
<tr>
<td>Radius of curvature, tip (pinion)</td>
<td>0.8156</td>
</tr>
<tr>
<td>Radius of curvature, tip (gear)</td>
<td>2.383</td>
</tr>
<tr>
<td>Scoring $PVT_p$</td>
<td>37654</td>
</tr>
<tr>
<td>Scoring $PVT_g$</td>
<td>79762</td>
</tr>
<tr>
<td>AGMA Bending stress (pinion)</td>
<td>4444.76 psi</td>
</tr>
<tr>
<td>AGMA Contact stress (pinion)</td>
<td>4.6332 E4 psi</td>
</tr>
<tr>
<td>AGMA Bending stress (gear)</td>
<td>3932.144 psi</td>
</tr>
<tr>
<td>AGMA Contact stress (gear)</td>
<td>24297.19 psi</td>
</tr>
</tbody>
</table>
Table 7.3 (contd)  Output variables for an example analysis

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weight</td>
<td>106.4322</td>
</tr>
<tr>
<td>Minimum number of teeth</td>
<td>15.5288</td>
</tr>
<tr>
<td>Bending stress safety factor (pinion)</td>
<td>13.499</td>
</tr>
<tr>
<td>Pitting safety factor (pinion)</td>
<td>4.3167</td>
</tr>
<tr>
<td>Bending stress safety factor (gear)</td>
<td>15.2589</td>
</tr>
<tr>
<td>Pitting safety factor (gear)</td>
<td>8.2314</td>
</tr>
<tr>
<td>Undercutting slack</td>
<td>9.47</td>
</tr>
<tr>
<td>Facewidth slack</td>
<td>3.0438</td>
</tr>
</tbody>
</table>

Fig. 7.1 shows some hardcopies from the analysis of helical gears under the design shell. Since we are dealing with a large constraint network, all the variables fit on four spreadsheet pages. The input variables are marked with the letter 'i' before the numerical value. Fig. 7.2 shows a scaled drawing of the helical gear set. Fig. 7.3 gives the values of some of the function residuals which shows that all the equations have been solved. Fig. 7.4 shows a part of the solution sequence that was used to compute the outputs. Fig. 7.5 shows the effect of horsepower on the bending safety factor.
### External Helical Gear Set

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gear ratio</td>
<td>3.84</td>
</tr>
<tr>
<td>2</td>
<td>No of teeth (pinion)</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>No of teeth (gear)</td>
<td>96</td>
</tr>
<tr>
<td>4</td>
<td>Normal diam. pitch</td>
<td>0.4667</td>
</tr>
<tr>
<td>5</td>
<td>Normal press. angle</td>
<td>20</td>
</tr>
<tr>
<td>6</td>
<td>Helix angle</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Face width</td>
<td>3.07</td>
</tr>
<tr>
<td>8</td>
<td>Add on const. (pinion)</td>
<td>0.9995</td>
</tr>
<tr>
<td>9</td>
<td>Ded on const. (pinion)</td>
<td>1.25</td>
</tr>
<tr>
<td>10</td>
<td>Add on const. (gear)</td>
<td>0.9969</td>
</tr>
<tr>
<td>11</td>
<td>Ded on const. (gear)</td>
<td>1.3526</td>
</tr>
<tr>
<td>12</td>
<td>Center Distance</td>
<td>7.3977</td>
</tr>
<tr>
<td>13</td>
<td>Backlash</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>Working depth const.</td>
<td>2</td>
</tr>
<tr>
<td>15</td>
<td>Aspect ratio</td>
<td>2</td>
</tr>
<tr>
<td>16</td>
<td>Elastic mod. (pinion)</td>
<td>3.0E7</td>
</tr>
<tr>
<td>17</td>
<td>Pois. ratio (pinion)</td>
<td>0.3</td>
</tr>
<tr>
<td>18</td>
<td>All. bend stress (pinion)</td>
<td>6.0E4</td>
</tr>
<tr>
<td>19</td>
<td>All. contact str (pinion)</td>
<td>2.0E5</td>
</tr>
<tr>
<td>20</td>
<td>Density (pinion)</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>21</td>
<td>Elastic modulus (gear)</td>
<td>3.0E7</td>
</tr>
<tr>
<td>22</td>
<td>Pois. ratio (gear)</td>
<td>0.3</td>
</tr>
<tr>
<td>23</td>
<td>All bend stress (gear)</td>
<td>6.0E4</td>
</tr>
<tr>
<td>24</td>
<td>All contact str (gear)</td>
<td>2.0E5</td>
</tr>
<tr>
<td>25</td>
<td>Density (gear)</td>
<td>0.3</td>
</tr>
<tr>
<td>26</td>
<td>Circular pitch</td>
<td>0.3841</td>
</tr>
<tr>
<td>27</td>
<td>Trans. diam pitch</td>
<td>0.1762</td>
</tr>
<tr>
<td>28</td>
<td>Normal base pitch</td>
<td>0.3472</td>
</tr>
<tr>
<td>29</td>
<td>Trans. base pitch</td>
<td>0.3595</td>
</tr>
<tr>
<td>30</td>
<td>Axial pitch</td>
<td>1.4336</td>
</tr>
<tr>
<td>31</td>
<td>Trans. press. angle</td>
<td>20.6469</td>
</tr>
<tr>
<td>32</td>
<td>Base helix angle</td>
<td>14.0761</td>
</tr>
<tr>
<td>33</td>
<td>Addendum (pinion)</td>
<td>0.110</td>
</tr>
<tr>
<td>34</td>
<td>Dedendum (pinion)</td>
<td>0.1595</td>
</tr>
<tr>
<td>35</td>
<td>Addendum (gear)</td>
<td>0.1177</td>
</tr>
<tr>
<td>36</td>
<td>Dedendum (gear)</td>
<td>0.1598</td>
</tr>
<tr>
<td>37</td>
<td>Outside dia (pinion)</td>
<td>3.293</td>
</tr>
<tr>
<td>38</td>
<td>Pitch dia (pinions)</td>
<td>3.0569</td>
</tr>
<tr>
<td>39</td>
<td>Base dia (pinion)</td>
<td>2.8666</td>
</tr>
<tr>
<td>40</td>
<td>Root dia (pinion)</td>
<td>7.7390</td>
</tr>
</tbody>
</table>

---

**Fig. 7.1:** Values of the gear design variables
### External Helical Gear Set

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>41</td>
<td>Outside dia (gear)</td>
<td>11.974</td>
</tr>
<tr>
<td>42</td>
<td>Pitch dia (gear)</td>
<td>11.7305</td>
</tr>
<tr>
<td>43</td>
<td>Base dia (gear)</td>
<td>10.9846</td>
</tr>
<tr>
<td>44</td>
<td>Root dia (gear)</td>
<td>11.413</td>
</tr>
<tr>
<td>45</td>
<td>Profile contact rat.</td>
<td>1.6419</td>
</tr>
<tr>
<td>46</td>
<td>Face contact ratio</td>
<td>2.1414</td>
</tr>
<tr>
<td>47</td>
<td>Load sharing ratio</td>
<td>0.6193</td>
</tr>
<tr>
<td>49</td>
<td>Line of action</td>
<td>0.5902</td>
</tr>
<tr>
<td>49</td>
<td>Horsepower</td>
<td>20</td>
</tr>
<tr>
<td>50</td>
<td>Pinion torque</td>
<td>1000.006</td>
</tr>
<tr>
<td>51</td>
<td>Pinion speed</td>
<td>1260.5</td>
</tr>
<tr>
<td>52</td>
<td>Pitchline velocity</td>
<td>1000.773</td>
</tr>
<tr>
<td>53</td>
<td>Tangential load</td>
<td>563.9967</td>
</tr>
<tr>
<td>54</td>
<td>Normalized Hub add</td>
<td>1.35</td>
</tr>
<tr>
<td>55</td>
<td>Normalized Hub rad</td>
<td>0.3</td>
</tr>
<tr>
<td>56</td>
<td>Normalized Hub Shift</td>
<td>0</td>
</tr>
<tr>
<td>57</td>
<td>Normalized Hub Proto</td>
<td>0</td>
</tr>
<tr>
<td>58</td>
<td>Quality Number</td>
<td>7</td>
</tr>
<tr>
<td>59</td>
<td>Elastic coeff. CP</td>
<td>2290.694</td>
</tr>
<tr>
<td>60</td>
<td>Dynamic factor, CV</td>
<td>0.7476</td>
</tr>
</tbody>
</table>

---

### External Helical Gear Set

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>60</td>
<td>Dynamic factor, CV</td>
<td>0.7476</td>
</tr>
<tr>
<td>61</td>
<td>J-factor</td>
<td>0.2151</td>
</tr>
<tr>
<td>62</td>
<td>J-factor (pinion)</td>
<td>0.5243</td>
</tr>
<tr>
<td>63</td>
<td>J-factor (gear)</td>
<td>0.5926</td>
</tr>
<tr>
<td>64</td>
<td>Rad. of curv. tip F in</td>
<td>0.8556</td>
</tr>
<tr>
<td>65</td>
<td>Rad. of curv. tip G in</td>
<td>2.3831</td>
</tr>
<tr>
<td>66</td>
<td>Scoring PVI (Pin)</td>
<td>3.766454</td>
</tr>
<tr>
<td>67</td>
<td>Scoring PVI (Gear)</td>
<td>7.976284</td>
</tr>
<tr>
<td>68</td>
<td>AGMA Bend str (pinion)</td>
<td>4444.7642</td>
</tr>
<tr>
<td>69</td>
<td>AGMA Contact str (pinion)</td>
<td>4.633224</td>
</tr>
<tr>
<td>70</td>
<td>AGMA Bend str (gear)</td>
<td>3932.144</td>
</tr>
<tr>
<td>71</td>
<td>AGMA Contact str (gear)</td>
<td>24297.1895</td>
</tr>
<tr>
<td>72</td>
<td>Bending S.F. (pinion)</td>
<td>13.499</td>
</tr>
<tr>
<td>73</td>
<td>Pitting S.F. (pinion)</td>
<td>4.3167</td>
</tr>
<tr>
<td>74</td>
<td>Bending S.F. (gear)</td>
<td>15.2509</td>
</tr>
<tr>
<td>75</td>
<td>Pitting S.F. (gear)</td>
<td>8.2314</td>
</tr>
<tr>
<td>76</td>
<td>Min. No. of teeth</td>
<td>15.5296</td>
</tr>
<tr>
<td>77</td>
<td>Weight</td>
<td>206.4322</td>
</tr>
<tr>
<td>78</td>
<td>Slack 1</td>
<td>5.4704</td>
</tr>
<tr>
<td>79</td>
<td>Slack 2</td>
<td>3.0430</td>
</tr>
</tbody>
</table>
Fig. 7.2: Scaled picture of the helical gears

<table>
<thead>
<tr>
<th>No</th>
<th>Function name</th>
<th>Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Diagonal pitch</td>
<td>-2.4522E-07</td>
</tr>
<tr>
<td>2</td>
<td>Pitch dia pinion</td>
<td>-6.366E-08</td>
</tr>
<tr>
<td>3</td>
<td>Pitch dia gear</td>
<td>4.111E-07</td>
</tr>
<tr>
<td>4</td>
<td>Center distance</td>
<td>-9.696E-07</td>
</tr>
<tr>
<td>5</td>
<td>Outside dia (pin)</td>
<td>-2.220E-09</td>
</tr>
<tr>
<td>6</td>
<td>Root dia (pin)</td>
<td>2.187E-06</td>
</tr>
<tr>
<td>7</td>
<td>Outside dia (gear)</td>
<td>-9.755E-07</td>
</tr>
<tr>
<td>8</td>
<td>Root dia (gear)</td>
<td>-9.293E-07</td>
</tr>
<tr>
<td>9</td>
<td>Pressure angle (pin)</td>
<td>7.184E-08</td>
</tr>
<tr>
<td>10</td>
<td>Base dia (pinion)</td>
<td>3.189E-07</td>
</tr>
<tr>
<td>11</td>
<td>Base dia (gear)</td>
<td>7.942E-07</td>
</tr>
<tr>
<td>12</td>
<td>Gear ratio</td>
<td>2.145E-06</td>
</tr>
<tr>
<td>13</td>
<td>Circular pitch</td>
<td>-3.4945E-07</td>
</tr>
<tr>
<td>14</td>
<td>Length of line of ac</td>
<td>9.999E-07</td>
</tr>
<tr>
<td>15</td>
<td>Base pitch calc.</td>
<td>-1.067E-07</td>
</tr>
<tr>
<td>16</td>
<td>Normal base pitch ca</td>
<td>3.639E-08</td>
</tr>
<tr>
<td>17</td>
<td>Base helix angle</td>
<td>9.749E-08</td>
</tr>
<tr>
<td>18</td>
<td>Profile contact ratio</td>
<td>4.230E-08</td>
</tr>
<tr>
<td>19</td>
<td>i-factor routine</td>
<td>-1.469E-07</td>
</tr>
<tr>
<td>20</td>
<td>Upward trend</td>
<td>0.099E+00</td>
</tr>
</tbody>
</table>

Fig. 7.3: Function residual values
### Solution Sequence

<table>
<thead>
<tr>
<th>Gear ratio</th>
<th>3.84</th>
</tr>
</thead>
</table>

#### Equations

- Elastic coeff, C£: $4.4240E-05$
- Tangential load: $-9.1159E-04$
- Base helix angle: $9.7498E-06$
- Axial pitch: $2.1020E-07$
- Pitchline velocity: $2.9185E-03$

### Graph of Bending Safety Factor vs Horsepower

Fig. 7.5: Graph of bending safety factor vs horsepower
7.3.2 Example 2 - Specification of the center distance

To change the set of inputs, we specify the addendum and dedendum constants for the gears. We also specify the center distance and let the design shell compute the helix angle. Fig. 7.6 shows some of the inputs and outputs from this design case. Fig. 7.7 shows the solution sequence from this case. Notice that we now have to solve three equations simultaneously to compute pitch diameters and the transverse diametral pitch.

7.3.3 Example 3 - Specification of the center distance and gear ratio

To illustrate the flexibility in the specifications, let us specify the gear ratio and center distance. The partial design is presented in Fig. 7.8 which shows that the number of teeth are computed by the design shell. The solution sequence presented in Fig. 7.9 shows that we have to solve four equations simultaneously to compute the values of number of teeth and pitch diameters.
### Fig. 7.6: Design variables from example 2

<table>
<thead>
<tr>
<th>No.</th>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Gear ratio</td>
<td>3.84</td>
</tr>
<tr>
<td>2</td>
<td>No of teeth (pinion)</td>
<td>25</td>
</tr>
<tr>
<td>3</td>
<td>No of teeth (gear)</td>
<td>48</td>
</tr>
<tr>
<td>4</td>
<td>Normal diam. pitch</td>
<td>20</td>
</tr>
<tr>
<td>5</td>
<td>Normal press. angle deg</td>
<td>19.0362</td>
</tr>
<tr>
<td>6</td>
<td>Face width</td>
<td>3.07</td>
</tr>
<tr>
<td>7</td>
<td>Addm const. (pinion)</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Bedm const. (pinion)</td>
<td>1.35</td>
</tr>
<tr>
<td>9</td>
<td>Bedm const. (gear)</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Center Distance in</td>
<td>8</td>
</tr>
<tr>
<td>11</td>
<td>Backlash</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>Working depth const. in</td>
<td>2</td>
</tr>
<tr>
<td>13</td>
<td>Elastic mod. (pinion) psi</td>
<td>3.0E7</td>
</tr>
<tr>
<td>14</td>
<td>Pois. ratio (pinion) psi</td>
<td>0.3</td>
</tr>
<tr>
<td>15</td>
<td>All. bend str (pin) psi</td>
<td>6.0E4</td>
</tr>
<tr>
<td>16</td>
<td>All. contact str (pin) psi</td>
<td>2.0E5</td>
</tr>
<tr>
<td>17</td>
<td>Density (pinion) lb/in^3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

### Fig. 7.7: Solution sequence from example 2

```
<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1458E-06</td>
</tr>
<tr>
<td>4.4246E-05</td>
</tr>
<tr>
<td>-5.5171E-08</td>
</tr>
<tr>
<td>0.0000E+00</td>
</tr>
<tr>
<td>2.4390E-08</td>
</tr>
<tr>
<td>0.0000E+00</td>
</tr>
</tbody>
</table>
```
### Fig. 7.8: Design variables from example 3

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>i</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>15.4548</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>77.2741</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Fig. 7.9: Solution sequence from example 3

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>i</td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>15.4548</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>77.2741</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**SOLUTION SEQUENCE**

1. Mo of teeth (pinion) = 15.4548
2. Mo of teeth (gear) = 77.2741
3. Normal diam. pitch = 8
4. Normal press. angle = 20 degrees
5. Helix angle = 15 degrees
6. Face width = 0.5
7. Addm const. (pinion) = 1
8. Bedm const. (pinion) = 1
9. Bedm const. (gear) = 1
10. Center Distance = 6
11. Backlash = 1
12. Working depth const. = 2
13. Elastic mod. (pinion) = psi
14. Pois. ratio (pinion) = psi
15. All. bend str (pinion) = psi
16. All. contact str (pin) = psi
17. Density (pinion) = 1/lb/in^3
CHAPTER VIII
CONCLUSION

8.0 Summary

The purpose of this dissertation was to develop the mathematical tools required to build a framework for mechanical design. It was essential for this framework to be completely domain-independent so that it could be used as a design tool for a number of mechanical systems. Since design is primarily concerned with the management of constraints, the developed framework called the Design Shell derives all its knowledge from a numerical constraints module. All the constraints are written in a declarative manner making it easy to configure the Design Shell.

The main tasks of the research were to develop different constraint management (CM) algorithms based on the occurrence matrix representation. Three CM algorithms were developed to handle different design scenarios of an interactive CAD environment. The interactive situations that commonly occur in design were classified into four categories—New Specification, Respecification, Unspecification, and Reverse Specification. It was shown how all these four can be handled using the CM algorithms.

Since, most constraint managers rely heavily on a robust nonlinear equation solver, the development of a simultaneous equation solver was also discussed in the thesis. Six different methods were researched and programmed. Out of six, two methods namely Brent and Brown methods, were seen to be extremely robust methods.

Nonlinear optimization capability was provided so that the designer does not need to iterate to achieve optimum design configurations. The Generalized Reduced Gradient (GRG)
method was developed for a constraint optimization problem that has the objective function embedded inside the constraint equations. Such a formulation was found to be ideally suited for the design shell framework. The use of nonlinear occurrence matrix to generate an initial feasible basis was done to reduce the burden on the nonlinear solver. Also, the basis interchange algorithm relies on the constraint manager to find a set of backwardly-dependent decision variables.

Since practical design problems often involve discrete variables, the constraint optimization procedure was supplemented with a branch and bound algorithm to generate optimal discrete configurations. The branch and bound algorithm selects a branching variable that is closest to its prescribed discrete value. A basis change in then performed if the branching variable is not a decision variable. GRG method was used to generate the optimum for the left and right branches of the branch and bound tree search.

Using the methodology presented in this thesis, it was seen that the design shell framework is a good test bed to implement new ideas in mechanical design. A number of different mechanical systems were configured and tested during the course of the research. Some of these are summarized in the following table, where M defines the number of equality constraints and N is the number of variables in a design case.

<table>
<thead>
<tr>
<th>Description</th>
<th>Number of types</th>
<th>Maximum M</th>
<th>Maximum N</th>
</tr>
</thead>
<tbody>
<tr>
<td>Springs</td>
<td>5</td>
<td>10</td>
<td>18</td>
</tr>
<tr>
<td>Brakes</td>
<td>12</td>
<td>13</td>
<td>22</td>
</tr>
<tr>
<td>Flywheels</td>
<td>10</td>
<td>9</td>
<td>22</td>
</tr>
<tr>
<td>Keys</td>
<td>1</td>
<td>6</td>
<td>10</td>
</tr>
<tr>
<td>Pins</td>
<td>1</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>Couplings</td>
<td>1</td>
<td>5</td>
<td>11</td>
</tr>
<tr>
<td>Powerscrews</td>
<td>20</td>
<td>8</td>
<td>21</td>
</tr>
</tbody>
</table>

The above table shows that most of the problems configured under the design shell were of a modest dimension. In order to test the Design Shell for a large constraint network,
design of helical gears was modelled as a constraint management (CM) problem. The choice of a gear program as a prototype was based on the fact that gear design is one of the most complicated and complex design procedures. The gear design problem had 49 equality constraints in 79 variables.

8.1 Research Contributions

A number of new ideas were presented in this thesis. This section outlines all the research contributions.

1) Constraint management algorithms for large constraint networks were presented. The concept of active equations was introduced which simplifies the search for solution sets in an under-determined system of constraints (rectangular occurrence matrix).

2) Four different interactive design scenarios, namely, New Specification, Respecification, Unspecification, and Reverse Specification, were identified and separate procedures were presented to handle each one of them.

3) A study of hidden, redundant, and inconsistent constraints was done. Knowledge of these problems helps in good model development.

4) A measure for good formulations was presented. Alternate formulations for a mathematical model should always be studied to see if they can eliminate the problems of extraneous solutions, multiple solutions, and hidden singularities.

5) An Embedded Objective Function (EOF) model was presented which is used to convert any conventional nonlinear programming problem into a constraint management setup. The EOF model gives flexibility to the user to select any variable as an objective function.

6) Constraint management algorithms were used in the Generalized Reduced Gradient method for optimization. The use of the Design Decomposition Algorithm and the
nonlinear occurrence matrix to generate an initial basis is unique. A good initial basis reduces the computational effort needed in the state variable restoration phase.

7) A minimum design deviation principle was presented during the basis interchange phase of the GRG method. Using the design deviation principle, we can reduce the possibility of non-convergence of the nonlinear solver, since the design vector does not change radically in one iteration.

8) A basis change step was introduced in the branch-and-bound search for the discrete optimum. Such an addition helps the convergence of the algorithm by ensuring that the branching variable is a decision variable.

9) A significant study was conducted on the simultaneous nonlinear equation solver. The nonlinear occurrence matrix was used to reorder the equations in increasing order of nonlinearity. Such a reordering helps the Brent and Brown methods which work on the simultaneous equations in a top-down fashion.

10) A large constraint network for helical gear design was studied. A constraint management approach to gear design provides immense flexibility to the designer in choosing alternative sets of specifications.

11) A computer implementation based on the constraint management and optimization theory was developed. The software program called the Design Shell was written in FORTRAN-77 over a device independent graphics package.

8.2 Limitations

There are some limitations of the numerical approach to constraint management. Since the constraints are coded as a FORTRAN subroutine, the constraint management framework does not provide the flexibility of adding and deleting constraints in an interactive manner. In addition to that, the CM framework lacks any symbolic algebra capability to identify cases that lead to singular Jacobians due to hidden constraints. Thus, after configuration,
the design shell depends a lot on the robustness of the mathematical model of the design being done.

During the constraint satisfaction stage, the nonlinear solver is heavily dependent on the initial guesses for the output variables. For bad initial guesses, the solver might not converge or converge to physically meaningless roots. Since, most of the mathematical models for routine design problems exhibit a significant amount of nonlinearity, we often come across cases having multiple solutions. There is no provision in the design shell to automatically generate all the solutions and pick the best one.

Currently, the design shell can handle only continuous constraints. Though discontinuous constraints can be present in the mathematical model, the convergence of the solver is guaranteed only if the output variables present in such constraints are not specified as inputs.

A diagnostic scheme is absent from the design shell that can monitor the progress of the design and explain any problems during the solution stage. Such a scheme will be helpful in identifying redundant or inconsistent constraints and let the user know about the deficiencies in the mathematical model.

8.2 Future Research

Some of the limitations of the current research were presented in the previous section. Based on that, this section discusses several research issues which need to be addressed in any future work.

There is a lot of scope for research in the constraint management area. The current research was restricted to mathematical continuous constraints. In the future, one would come across constraints that are empirical in nature. For example, domain knowledge is generally expressed in an empirical form. Other constraints, such as, conditional constraints, discontinuous constraints, and constraints that allow multiple input/output (MIMO) should also be researched. The handling of these constraints might involve a hybrid symbolic-numeric approach.
In CM, additional research needs to be done to develop an even faster Design Decomposition Algorithm by employing heuristics. It was observed during the course of the research that DDA spends a lot of time determining the non-existence of solution sets whenever a new input variable is specified. Such a determination might become very time consuming for large constraint networks in an interactive design environment. Unresolved issues of constraint confliction and redundant constraints have to be researched. Preliminary research was done on the identification of hidden constraints that lead to singular Jacobians. But some more work is required to handle these hidden constraints. In order to simplify the configuration process of the design shell, a mechanism that automatically generates the nonlinear occurrence matrix should be developed.

In the area of the constraint optimization, there is work that needs to be done in resolving the degeneracy and singularity problems in the GRG method. The use of nonlinear occurrence matrix to prevent direct redundant cases was discussed in the thesis, but cases that lead to singular Jacobians during the optimization process have to be fully understood.

In the domain of nonlinear equation solving, there is still a lot of potential for future work. Newer methods are being developed everyday and one has to do a lot of experimentation to find out which methods are good for mechanical design. Homotopy-based continuation methods are also gaining popularity and these should be researched. Research needs to be done to identify multiple solutions of nonlinear equations and prevent the design from converging to solutions that lie in the infeasible region.

Finally, the model for gear design can be improved by adding the constraints for all the AGMA factors. The constraints for gear noise can be added to ensure quieter running gears. Models for other gears such as bevel, spiral bevel and worm gears can also be developed.
BIBLIOGRAPHY


Balasubramanian, Ramakrishnan, 1991, Initial Design Selection and Discrete Variable Issues in Mechanical Design, MS Degree, The Ohio State University, Columbus, OH.


Gabriele, G. A., 1975, *Application of the Reduced Gradient Method to Optimal Engineering Design*, M.S. Thesis, School of Mechanical Engineering, Purdue University, Lafayette, IN.


APPENDIX A

Design Shell Files for Helical Gear Design
Appendix A

Design Shell Files for Helical Gear Design

A.1 File of filenames (FILES.DAT)

'GCASES.DAT'
'GEARS.ARC'
'GEARS.DAT'
'DS.HLP'
'DESIGN.HLP'

A.2 Cases files (GCASES.DAT)

'GEARS'

1
1

'HELICAL'
1

'External'

A.3 Variable Description File (GEARS.DAT)

1
#
1

'External Helical Gear Set'
79
49
0
0

'Gear ratio', 1, 1, 1
', 2.000 , 1.000 , 50.00
2 12 20 1 1

'No of teeth (pinion)', 2, 2, 2
', 25.00 , 7.000 , 200.0
9 2 12 15 21 34 37 38 39 48
1 1 1 3 1 3 3 2 1

'No of teeth (gear) ', 3, 3, 2
', 50.00 , 7.000 , 200.0
6 3 12 21 37 38 39
1 1 3 2 2 2
'Normal diam. pitch', 4, 4, 1
' ', 8.000, 0.0000E+00, 1000.
  7 1 5 6 7 8 21 39
  1 2 2 2 2 3 2

'Normal press. angle', 5, 5, 1
'deg', 20.00, 14.50, 30.00
  5 9 21 37 38 39
  1 3 3 3 3

'Helix angle', 6, 6, 1
'deg', 0.0000E+00, 0.0000E+00, 45.00
  11 1 9 16 17 19 20 21 29 30 31 39
  1 1 1 1 3 3 4 2 2 3

'Face width', 7, 7, 1
'in', 0.5000, 0.1000, 100.0
  13 19 20 21 22 23 24 31 37 38 39 40 41 49
  1 1 3 1 1 1 1 2 3 3 3 1 1 1

'Addm const. (pinion)', 8, 8, 1
' ', 1.0000, 0.1000, 2.0000
  3 5 21 29 1 1 1

'Dedm const. (pinion)', 9, 9, 1
' ', 1.2500, 0.1000, 2.0000
  1 6 1

'Addm const. (gear)', 10, 10, 1
' ', 1.0000, 0.1000, 2.0000
  2 7 39 1 1

'Dedm const. (gear)', 11, 11, 1
' ', 1.2500, 0.1000, 2.0000
  1 8 1

'Center Distance', 12, 12, 1
'in', 4.6888, 0.0000E+00, 1000.
  7 4 14 20 21 37 38 39
  1 1 1 3 3 3 3

'Backlash', 13, 13, 1
'in', 0.0, 0.0, 0.025
  2 21 39 1 1

'Working depth const.', 14, 14, 1
'in', 2.0, 0.0, 5.0
  2 21 39 1 1

'Aspect ratio', 15, 15, 1
' ', 0.25, 0.1, 5.0
  1 49 1

'Elastic mod. (pinion)', 16, 16, 12
'GHAT.DAT', 'YMOD'
'psi', 30E6, 1E6, 1E10
'Pois. ratio (pinion)', 17, 17, 12
'GMAT.DAT', 'POIS'
', 0.3, 0.2, 0.6
1 32 2

'All. bend str (pinion)', 18, 18, 12
'GMAT.DAT', 'SAT'
'psi', 0.20000E+05, 10.00, 0.10000E+11
1 25 1

'All. contact str (pinion)', 19, 19, 12
'GMAT.DAT', 'SAC'
'psi', 0.60000E+05, 10.00, 0.10000E+11
1 26 1

'Density (pinion)', 20, 20, 12
'GMAT.DAT', 'DENS'
'lb/in**3', 0.0, 0.3, 10.
1 24 1

'Elastic modulus (gear)', 21, 21, 12
'GMAT.DAT', 'YMOD'
'psi', 30E6, 1E5, 1E10
1 32 2

'Pois. ratio (gear)', 22, 22, 12
'GMAT.DAT', 'POIS'
', 0.3, 0.2, 0.6
1 32 2

'All bend stress (gear)', 23, 23, 12
'GMAT.DAT', 'SAT'
'psi', 0.20000E+05, 10.00, 0.10000E+11
1 42 1

'All contact str (gear)', 24, 24, 12
'GMAT.DAT', 'SAC'
'psi', 0.60000E+05, 10.00, 0.10000E+11
1 43 1

'Density (gear)', 25, 25, 12
'GMAT.DAT', 'DENS'
'lb/in**3', 0.0, 0.3, 10.
1 24 1

'Circular pitch', 26, 26, 1
'in', 0.4, 0., 10.
2 13 19 2 2

'Trans. diam pitch', 27, 27, 1
', 8.0000, 0.00000E+00, 1000.
7 1 2 3 13 22 34 40
1 1 1 1 1 1 1
'Normal base pitch ', 28, 28, 1
'in ', 0.5000 , 0.1000E-01, 100.0
1 16 1

'Trans. base pitch ', 29, 29, 1
'in ', 0.5000 , 0.1000E-01, 100.0
5 15 16 18 20 30
1 1 1 1 1

'Axial pitch', 30, 30, 1
' ', 0.5, -0.1, 100.
2 30 31 1 2

'Trans. press. angle ', 31, 31, 1
'deg ', 20.00 , 14.50 , 30.00
11 9 10 11 14 17 20 21 29 37 38 39
1 1 1 1 1 1 3 4 3 3 3

'Base helix angle ', 32, 32, 1
'deg ', 0.0000E+00 , 0.0000E+00, 50.00
6 17 20 21 30 31 39
1 3 3 1 2 3

'Addendum (pinion)', 33, 33, 1
'in ', 0.2, 0., 10.
1 44 1

'Dedendum (pinion)', 34, 34, 1
'in ', 0.2, 0., 10.
1 45 1

'Addendum (gear)', 35, 35, 1
'in ', 0.2, 0., 10.
1 46 1

'Dedendum (gear)', 36, 36, 1
'in ', 0.2, 0., 10.
1 47 1

'Outside dia (pinion)', 37, 37, 1
'in ', 3.375 , 0.0000E+00, 1000.
6 5 20 21 23 35 44
1 1 3 1 2 1

'Pitch dia (pinion) ', 38, 38, 1
'in ', 3.125 , 0.0000E+00, 1000.
13 2 4 5 6 10 20 24 27 28 37 44 45 49
1 1 1 1 1 1 1 2 1 1 2 1 1 1

'Base dia (pinion) ', 39, 39, 1
'in ', 2.937 , 0.0000E+00, 1000.
4 10 15 20 35
1 1 1 2

'Root dia (pinion) ', 40, 40, 1
'in ', 2.813 , 0.0000E+00, 1000.
'Outside dia (gear) ', 41, 41, 1
'in ', 6.500, 0.0000E+00, 1000.
7 7 20 21 36 39 41 46
1 1 3 2 2 1 1

'Pitch dia (gear) ', 42, 42, 1
'in ', 6.250, 0.0000E+00, 1000.
9 3 4 7 8 11 20 24 38 47
1 1 1 1 1 1 1 2 1

'Base dia (gear) ', 43, 43, 1
'in ', 5.873, 0.0000E+00, 1000.
2 11 36 1 2

'Root dia (gear) ', 44, 44, 1
'in ', 5.938, 0.0000E+00, 1000.
2 8 47 1 1

'Profile contact rat.', 45, 45, 1
' ', 1.000, 0.1000, 5.000
2 18 31 1 1

'Face contact ratio ', 46, 46, 1
' ', 0.0000E+00, 0.0000E+00, 5.000
3 19 20 31 1 1 2

'Load sharing ratio', 47, 47, 1
', 1.4, 0., 100.
4 20 21 31 39 3 3 1 3

'Line of Action', 48, 48, 1
'in ', 3.000, 0.0000E+00, 200.0
6 14 18 20 31 37 38
1 1 3 1 3 3

'Horsepower', 49, 49, 1
'HP ', 20.00, 0.1000E-01, 1000.
2 27 34 1 1

'Pinion torque', 50, 50, 1
'lb-in', 1000.0, 0.1, 1E6
3 27 37 38 1 3 3

'Pinion speed', 51, 51, 1
'RPM ', 1120., 0.1000, 0.1000E+05
4 28 34 37 38
1 1 2 2

'Pitchline velocity', 52, 52, 1
'fpm', 10., 0.1, 1E10
3 27 28 33 1 1 2

'Tangential load', 53, 53, 1
'lb ', 400.0, 0.1000, 0.1000E+06
'Scoring PVT (Gear)', 67, 67, 5
    , 1E6, 1E2, 1E7
1 38 1

'AGMA Bend str (pinion)', 68, 68, 5
'psi', 0.2400E+05, 0.1000 , 0.1000E+09
2 22 25 1 1

'AGMA Contact str (pinion)', 69, 69, 5
'psi', 0.4000E+05, 0.1000 , 0.1000E+09
2 23 26 1 1

'AGMA Bend str (gear)', 70, 70, 5
'psi', 0.2400E+05, 0.1000 , 0.1000E+09
2 40 42 1 1

'AGMA Contact str (gear)', 71, 71, 5
'psi', 0.4000E+05, 0.1000 , 0.1000E+09
2 41 43 1 1

'Bending S.F. (pinion)', 72, 72, 5
    , 1.000 , 1.0, 100.0
1 25 1

'Pitting S.F. (pinion)', 73, 73, 5
    , 1.000 , 1.0, 100.0
1 26 1

'Bending S.F. (gear)', 74, 74, 5
    , 1.000 , 1, 100.0
1 42 1

'Pitting S.F. (gear)', 75, 75, 5
    , 1.000 , 1., 100.0
1 43 1

'Min. No. of teeth', 76, 76, 5
    , 17, 5, 100
2 29 48 1 1

'Weight', 77, 77, 1
'lb', 1.000 , 0.1000E-02, 1.0E10
1 24 1

'Undercutting slack', 78, 78, 1
    , 0., 0., 1e6
1 48 1

'Facewidth slack', 79, 79, 1
    , 0., 0., 1e6
1 49 1

'Diametral pitch'
'Pitch dia pinion'
'Pitch dia gear'
'Center distance'
A.4 Discrete variable file (GMAT.DAT)

$$SAT
7
'Steel BHN 180 Harden, Temper', 26500
'Steel BHN 300 Harden, Temper', 37500
'Steel BHN 450 Harden, Temper',47500
'Steel R55C Case carburized',60000
'Cast Iron AGMA 20 as cast',5000
'Cast Iron AGMA 30 as cast',8500
'Cast Iron AGMA 40 as cast',13000
$SSAC
7
'Steel BHN 180 Harden, Temper',95000
'Steel BHN 300 Harden, Temper',135000
'Steel BHN 450 Harden, Temper',190000
'Steel R55C Case carburized',200000
'Cast Iron AGMA 20 as cast',30000
'Cast Iron AGMA 30 as cast',50000
'Cast Iron AGMA 40 as cast',65000
$SYMOD
7
'Steel BHN 180 Harden, Temper',30E6
'Steel BHN 300 Harden, Temper',30E6
'Steel BHN 450 Harden, Temper',30E6
'Steel R55C Case carburized',30E6
'Cast Iron AGMA 20 as cast',19E6
'Cast Iron AGMA 30 as cast',19E6
'Cast Iron AGMA 40 as cast',19E6
$SPOIS
7
'Steel BHN 180 Harden, Temper',0.3
'Steel BHN 300 Harden, Temper',0.3
'Steel BHN 450 Harden, Temper',0.3
'Steel R55C Case carburized',0.3
'Cast Iron AGMA 20 as cast',0.255
'Cast Iron AGMA 30 as cast',0.255
'Cast Iron AGMA 40 as cast',0.255
$SBHN
7
'Steel BHN 180 Harden, Temper',180
'Steel BHN 300 Harden, Temper',300
'Steel BHN 450 Harden, Temper',450
'Steel R55C Case carburized',300
'Cast Iron AGMA 20 as cast',200
'Cast Iron AGMA 30 as cast',200
'Cast Iron AGMA 40 as cast',200
$DENS
7
'Steel BHN 180 Harden, Temper',0.3
'Steel BHN 300 Harden, Temper',0.3
'Steel BHN 450 Harden, Temper',0.3
'Steel R55C Case carburized',0.3
'Cast Iron AGMA 20 as cast',0.25
'Cast Iron AGMA 30 as cast',0.25
'Cast Iron AGMA 40 as cast',0.25
APPENDIX B

Equations Module for Helical Gear Design
Appendix B

Equations Module for Helical Gear Design

SUBROUTINE EQUNS (IEQN, NMAX, X, VALF, IERR)
C
INTEGER NMAX, IERR
DOUBLE PRECISION X(NMAX)
DOUBLE PRECISION VALF
C
DOUBLE PRECISION PI, DEGRAD
C
COMMON /ACASE/ ICASE
COMMON /ACONST/ PI, DEGRAD
C
constants
C
PI = 4.0*ATAN(1.0)
DEGRAD = PI/180.0
C
IF (ICASE.EQ.1) THEN
   CALL HECAL (IEQN, NMAX, X, VALF, IERR)
ENDIF
C
RETURN
END
C
SUBROUTINE HECAL (IEQN, NMAX, X, VALF, IERR)
INTEGER IEQN, NMAX, IERR
DOUBLE PRECISION X(NMAX)
DOUBLE PRECISION VALF
C
DOUBLE PRECISION NP, NG, TNMIN
DOUBLE PRECISION GRATIO
DOUBLE PRECISION PN, PT, PC, PBT, PBN, PX
DOUBLE PRECISION PHIN, PHIT
DOUBLE PRECISION PSI, PSIB
DOUBLE PRECISION ADDP, DEDP, ADDG, DEDG, ADCP, DDCP, ADCG, DDCG
DOUBLE PRECISION ODP, ODG, PDP, PDG, BDP, BDG, RDP, RDG
DOUBLE PRECISION FWIDTH, CD, WEIGHT
DOUBLE PRECISION CRP, CRF, ZLC, LSR
DOUBLE PRECISION HORSEP, RPMP, TANLOD, PLV, TORQP
DOUBLE PRECISION CP, AKV, AIFACT, XIP, XJP, XJG
DOUBLE PRECISION EMP, EMG, POISP, POISG, DENS, DENG
DOUBLE PRECISION SATP, SACP, SATG, SACG
DOUBLE PRECISION STP, STG, SCP, SCG, SFTP, SFTC, SFTG, SFGC
DOUBLE PRECISION HOBFSFT, HOBADD, HOBRAD, PROTUB, BLASH
DOUBLE PRECISION QN
DOUBLE PRECISION RHTP, RHTT
DOUBLE PRECISION HSTRP, HSTRG
DOUBLE PRECISION PVTP, PVTT
DOUBLE PRECISION ASPRAT
DOUBLE PRECISION HK
DOUBLE PRECISION SLACK1, SLACK2

DOUBLE PRECISION PI, DEGRAD
COMMON /ACONST/ PI, DEGRAD

abnormal return error flag
IERR = 1

Geometry variables

GRATIO = X(1)
NP = X(2)
NG = X(3)
PN = X(4)
PHIN = X(5)*DEGRAD
PSI = X(6)*DEGRAD
FWIDTH = X(7)
ADCP = X(8)
DDCP = X(9)
ADCG = X(10)
DDCG = X(11)
CD = X(12)
BLASH = X(13)
HK = X(14)
ASPRAT = X(15)

Material properties

EMP = X(16)
POISP = X(17)
SATP = X(18)
SACP = X(19)
DENSE = X(20)
EMG = X(21)
POISG = X(22)
SATG = X(23)
SACG = X(24)
DENSG = X(25)

PC = X(26)
PT = X(27)
PBN = X(28)
PBT = X(29)
PX = X(30)
PHIT = X(31)*DEGRAD
PSIB = X(32)*DEGRAD

ADDP = X(33)
DEDP = X(34)
ADDG = X(35)
DEDG = X(36)
ODP = X(37)
PDP = X(38)
BDP = X(39)
RDP = X(40)
ODG = X(41)
PDG = X(42)
BDG = X(43)
RDG = X(44)

C
CRP = X(45)
CRF = X(46)
LSR = X(47)
ZLC = X(48)

C
loading and velocity
HORSEP = X(49)
TCRQP = X(50)
RPMP = X(51)
PLV = X(52)
TANLOD = X(53)

C
HOBADD = X(54)
HOBRAD = X(55)
HOBSFT = X(56)
PROTUB = X(57)

C
QN = X(58)
CP = X(59)
AKV = X(60)
XIP = X(61)
XJP = X(62)
XJG = X(63)

C
RHOTP = X(64)
RHOTG = X(65)
PVTG = X(66)
PVTG = X(67)

C
STP = X(68)
SCP = X(69)
STG = X(70)
SCG = X(71)
SFTP = X(72)
SFCP = X(73)
SFTG = X(74)
SFCG = X(75)
TNMIN = X(76)
WEIGHT = X(77)
SLACK1 = X(78)
SLACK2 = X(79)

C
transverse diametral pitch
IF (IEQN .EQ. 1) THEN
   VALF = PT - PN*COS(PSI)
C
pitch diameters
ELSEIF (IEQN .EQ. 2) THEN
  VALF = PDP - (NP/PT)
ELSEIF (IEQN .EQ. 3) THEN
  VALF = PDG - (NG/PT)

center distance

ELSEIF (IEQN .EQ. 4) THEN
  VALF = CD - 0.5*(PDP+PDG)

outside and root diams

ELSEIF (IEQN .EQ. 5) THEN
  VALF = ODP - (PDP + 2*ADCP/PN)
ELSEIF (IEQN .EQ. 6) THEN
  VALF = RDP - (PDP - 2*DDCP/PN)
ELSEIF (IEQN .EQ. 7) THEN
  VALF = ODG - (PDG + 2*ADCG/PN)
ELSEIF (IEQN .EQ. 8) THEN
  VALF = RDG - (PDG - 2*DDCG/PN)

transverse pressure angle

ELSEIF (IEQN .EQ. 9) THEN
  VALF = COS(FSI)*TAN(PHIT)-TAN(PHIN)

base diameters

ELSEIF (IEQN .EQ. 10) THEN
  VALF = BDP - PDP*COS(PHIT)
ELSEIF (IEQN .EQ. 11) THEN
  VALF = BDG - PDG*COS(PHIT)

gear ratio

ELSEIF (IEQN .EQ. 12) THEN
  VALF = NG - GRATIO*NP

Circular pitch

ELSEIF (IEQN.EQ.13) THEN
  VALF = PC*PT - PI

length of line of action

ELSEIF (IEQN. EQ. 14) THEN
  VALF = ZLC - (RHOTP + RHTG - CD*SIN(PHIT))

CALCULATION OF transverse BASE PITCH

ELSEIF (IEQN. EQ. 15) THEN
  VALF = PBT - (PI*BDP)/NP

CALCULATION OF normal transverse BASE PITCH

ELSEIF (IEQN. EQ. 16) THEN
VALF = PBN - PBT*COS(PSI)

calculation of the base helix angle

ELSEIF (IEQN. EQ. 17) THEN
    VALF = TAN(PSIB) - TAN(PSI)*COS(PHI)

calculation of the profile contact ratio

ELSEIF (IEQN. EQ. 18) THEN
    VALF = CRF*PBT - ZLC

calculation of the face contact ratio

ELSEIF (IEQN. EQ. 19) THEN
    VALF = CRF*PC - FWIDTH*TAN(PSI)

I-factor : XIP

ELSEIF (IEQN. EQ. 20) THEN
    XIP = AIFACT(CD,CRF,FWIDTH,GRATIO,
*     PBT,PHIT,PSI,PSIB,
*     BDP,ODG,ODP,PDG,PDP,ZLC,LSR,
*     IERR)
    IF (IERR.EQ.0) THEN
        X(61) = XIP
        VALF = 0.0
    ELSE
        VALF = 1E3
    ENDF

J-factor for pinion : XJP

ELSEIF (IEQN. EQ. 21) THEN
    CALL AJFACT(1, PN, NP, NG, ODP, ODG, CD, FWIDTH,
*     PHIN, PHIT, PSI, PSIB, LSR, ADCP, HK,
*     HOBST, HOBADD, HOBRAD, PROTUB, BLASH,
*     XJP, IERR)
    IF (IERR.EQ.0) THEN
        VALF = 0.0
        X(62) = XJP
    ELSE
        VALF = 1E3
    ENDF

bending stress (pinion) : STP

ELSEIF (IEQN. EQ. 22) THEN
    VALF = STP*FWIDTH*XJP*AKV - TANLOD*PT

contact stress (pinion) : SCP

ELSEIF (IEQN. EQ. 23) THEN
    VALF = ODP*FWIDTH*XIP*AKV*(SCP/CP)**2 - TANLOD

Weight of the gears
ELSEIF (IEQN .EQ. 24) THEN
   VALF = WEIGHT - 0.25*PI*WIDTH*(DENSP*PDP*PDP + DENS*PDG*PDG)

safety factor in bending (pinion) : SFTP

ELSEIF (IEQN .EQ. 25) THEN
   VALF = SFTP*STP - SATP

safety factor in pitting (pinion) : SFCP

ELSEIF (IEQN .EQ. 26) THEN
   VALF = SFCP*SCP - SACP

torque and horsepower

ELSEIF (IEQN .EQ. 27) THEN
   VALF = TORQP*PLV - HORSEP*PDP*16500.0

Fitchline velocity and RPM

ELSEIF (IEQN .EQ. 28) THEN
   VALF = PI*PDP*RPM* - PLV*12.0

Minimum number of teeth for undercutting

ELSEIF (IEQN .EQ. 29) THEN
   VALF = TNMIN - 2*ADCP*COS(PSI)/(SIN(PHI)*SIN(PHI))

axial pitch

ELSEIF (IEQN .EQ. 30) THEN
   IF (PSI.LT.0.002) THEN
      VALF = PX
   ELSE
      VALF = PX*TAN(PSIB) - PBT
   ENDIF

Approximate load sharing ratio (exact for spur gears).

ELSEIF (IEQN .EQ. 31) THEN
   IF (PSI.LT.0.002) THEN
      VALF = LSR - 1.0
   ELSE
      SINBPS = SIN(PSIB)
      COSBPS = COS(PSIB)
      CRPF = CRF - INT(CRF)
      CRFF = CRF - INT(CRF)
      IF ( (1 - CRPF) .GE. CRFF ) THEN
         ALMIN = (CRP*WIDTH - CRPF*CRFF*PX)/COSBPS
      ELSE
         ALMIN = (CRF*WIDTH - (1 - CRFF)*(1 - CRPF)*PX)/COSBPS
      ENDIF
      IF (CRF.LE.2.0) THEN
         VALF = LSR - WIDTH/ALMIN
      ELSE
          (IEQN .EQ. 36) THEN
VALF = LSP - PBH/(0.95*ZLC)
ENDIF
ENDIF

C
Elastic coefficient
C
ELSEIF (IEQN .EQ. 32) THEN
  DENOM = (((1 - POISP*POISP))/EMP) + ((1 - (POISG*POISG))
  /EMG)
  RADICL = 1/(PI*DENOM)
  IF (RADICL .LE. 0.0) THEN
    IERR = 1
    RETURN
  ELSE
    VALF = CP - SQRT(RADICL)
  ENDIF
C
Dynamic factor
C
ELSEIF (IEQN .EQ. 33) THEN
  For Quality numbers higher than 11 AKV=CV=0.94.
  IF (QN .GT. 11) THEN
    VALF = AKV - 0.94
  ELSE
    TPLV = PLV
    IF (QN .LT. 6) THEN
      IF (TPLV .GT. 2500) TPLV = 2500.
      VALF = AKV - 50. / (50. + SQRT(TTPLV) )
    ELSE
      B = ( (12.0 - QN) **0.667) /4.0
      A = 50.0 + 56.0* (1.0 - B)
      PLVX = (A + (QN - 3.0 ) ) **2
      IF (TPLV .GT. PLVX) TPLV = PLVX
      VALF = AKV - (A/ (A + SQRT(TTPLV) ) ) **B
    ENDIF
  ENDIF
C
tangential load
C
ELSEIF (IEQN. EQ. 34) THEN
  VALF = 0.001*TANLOD*NPR*EMP - 126*HORSEP*PT
C
radius of curvatures at the tip
C
ELSEIF (IEQN . EQ. 35) THEN
  IF (ODP.LT.BDP) THEN
    IERR=1
    RETURN
  ENDIF
  VALF = RHOTP - 0.5*SQRT(ODP**2 - BDP**2)
ELSEIF (IEQN . EQ. 36) THEN
  IF (ODG.LT.BDG) THEN
    IERR=1
    RETURN
ENDIF
VALF = RHOST - 0.5*SQR(ODG**2 - BDG**2)

Scoring factor computation. (PVT Formula)
Hertz compressive stress

ELSEIF (IEQN .EQ. 37) THEN
   HSTRP = 5740*SQR(TOPQ*CD*ZIN(PHIN)/(FWIDTH*ZLC*
   * NP*RHOST*(CD*ZIN(PHIT) - RHOST))
   VALF = PVTP - (PI*RPMP/360.)*(1+NP/NG)*HSTRP*
   * (RHOST - 0.5*PDP*ZIN(PHIT))**2
ELSEIF (IEQN .EQ. 38) THEN
   HSTRG = 5740*SQR(TOPQ*CD*ZIN(PHIN)/(FWIDTH*ZLC*
   * NP*RHOST*(CD*ZIN(PHIT) - RHOST))
   VALF = PVTP - (PI*RPMP/360.)*(1+NP/NG)*HSTRG*
   * (RHOSTG - 0.5*FDG*ZIN(PHIT))**2

J-factor for gear

ELSEIF (IEQN . EQ. 39) THEN
   CALL AJFACT(2, PN, NP, NG, ODF, ODG, CD, FWIDTH,
   * PHIN, PHIT, PSI, PSIB, LSH, ADCG, HK,
   * HOBST, HOBADD, HOBRADE, PROTUB, BIASH,
   * XJG, IERR)
   IF (IERR.EQ.0) THEN
      VALF = 0.0
      X(63) = XJG
   ELSE
      VALF = 1E3
   ENDIF

bending stress (gear)

ELSEIF (IEQN .EQ. 40) THEN
   VALF = STG*FWIDTH*XJG*AKV - TANLOD*PT

contact stress (gear)

ELSEIF (IEQN . EQ. 41) THEN
   VALF = ODG*FWIDTH*XIP*AKV*(SCG/CP)**2 - TANLOD

safety factor in bending (gear)

ELSEIF (IEQN .EQ. 42) THEN
   VALF = SFTG*STG - SATG

safety factor in pitting (gear)

ELSEIF (IEQN .EQ. 43) THEN
   VALF = SFCG*SCG - SACG

addendum and dedendum for pinion

ELSEIF (IEQN .EQ. 44) THEN
   VALF = 2*ADD + PDP - ODP
ELSEIF (IEQN .EQ. 45) THEN
\[
\text{VALF} = 2\times\text{DEDP} + \text{RDP} - \text{PDP}
\]

addendum and dedendum for gear

\[
\text{ELSEIF (IEQN .EQ. 46) THEN}
\]
\[
\text{VALF} = 2\times\text{ADDG} + \text{PDG} - \text{ODG}
\]

\[
\text{ELSEIF (IEQN .EQ. 47) THEN}
\]
\[
\text{VALF} = 2\times\text{DEDG} + \text{RDG} - \text{PDG}
\]

inequality constraint 1: tooth interference

\[
\text{ELSEIF (IEQN .EQ. 48) THEN}
\]
\[
\text{VALF} = \text{NP} - \text{TNMIN} - \text{SLACK1}
\]

inequality constraint 2: face width to pitch diameter

\[
\text{ELSEIF (IEQN .EQ. 49) THEN}
\]
\[
\text{VALF} = \text{ASPRAT}\times\text{PDP} - \text{FWIDTH} - \text{SLACK2}
\]

ENDIF

IERR = 0
RETURN
END

***********************************************************************
GOCAD
***********************************************************************

SUBROUTINE AJFACT (IGORP, PN, TNP, TNG, ODP, ODG, CD,
*     FWS, PHIN, PHIT, PSI, PSIb, LSR, ADC, HK,
*     X0, HAO, RA0, DELO, BLASH,
*     XJ, IERR)

***********************************************************************
GOCAD
***********************************************************************

AJFACT - Determines strength geometry factors (XJP,XJG).

AUTHOR - Rajiv Agrawal 1-APR-91

REVISED- Rajiv Agrawal 23-SEP-91

PROGRAM DESCRIPTION
This subroutine determines the AGMA bending strength geometry
factor and form factor for spur and helical gear teeth. Helical
gears are transformed into equivalent spur gears.

The algorithm for this program is based on a technical paper by
Robert Errichello "An Efficient Algorithm for Obtaining the Gear
Strength Geometry Factor For Shaper Cut Gears", Paper No.
P139.05, American Gear Manufacturers Association.

ACCESS
CALL AJFACT (IGORP, PN, TNP, TNG, ODP, ODG, CD,
*     FWS, PHIN, PHIT, PSI, PSIb, LSR, ADC, HK,
*     X0, HAO, RA0, DELO, BLASH,
*     XJ, IERR)

IGORP -IN VBL-USD- FLAG WHETHER PINION (=1) OR GEAR (=2)
PN  -DP VBL-xxx- Normal diametral pitch.
TNP -DP VBL-xxx- Number of pinion teeth.
TNG -DP VBL-xxx- Number of gear teeth.
C GDP -DP VBL-xxx- Outside diameter (pinion).
C ODG -DP VBL-xxx- Outside diameter (gear).
C Cd -DP VBL-xxx- Center distance.
C FWS -DP VBL-xxx- Face width.
C PHIN -DP VBL-xxx- Normal pressure angle.
C PHT -DP VBL-xxx- Transverse pressure angle.
C PSI -DP VBL-xxx- Helix angle.
C PS1B -DP VBL-xxx- Base helix angle.
C LSR -DP VBL-xxx- Load sharing ratio.
C ADC -DP VBL-xxx- Addendum constant.
C NK -DP VBL-xxx- Working depth.
C X0 -DP VBL-xxx- Addendum mod. coeff. of tool.
C HA0 -DP VBL-xxx- Nominal tool addendum.
C RAD0 -DP VBL-xxx- Tool tip radius.
C DEL0 -DP VBL-xxx- Amount of protuberance on cutter.
C BLASH -DP VBL-xxx- Backlash.
C KX -DP VBL-RTD- Agma bending strength geometry factor.
C IEPR -IN VBL-xxx- Error code.

COMMON BLOCK VARIABLES

AKF -DP VBL-WRK-/DERIV -- Temporary variable.
BETN -DP VBL-WRK-/DERIV -- Temporary variable.
DEGRAD -DP VBL-xxx-/ACONST-- Degrees to radians conversion.
DRA0 -DP VBL-WRK-/FUNCT -- Tool tip radius.
ENF -DP VBL-WRK-/DERIV -- Abscissa of critical point.
GNS -DP VBL-WRK-/FUNCT -- Angle to center of tool tip radius.
HP -RL VBL-WRK-/DERIV -- Height of Lewis parabola.
MNF -DP VBL-WRK-/DERIV -- Ordinate of critical point.
P1 -DP VBL-xxx-/ACONST-- The constant 3.14159.
RN2 -DP VBL-WRK-/FUNCT -- Gen. pitch radius of virtual spur.
RNLI -DP VBL-WRK-/FUNCT -- Virtual load radius.
RNO2 -DP VBL-WRK-/FUNCT -- Gen. pitch radius of virtual tool.
RNS0 -DP VBL-WRK-/FUNCT -- Radius to center of tool tip radius.
TN -DP VBL-WRK-/FUNCT -- Virtual tooth number of gear/pinion.
TNO -DP VBL-WRK-/FUNCT -- Virtual tooth number of tool.
UNO -DP VBL-WRK-/DERIV -- Temporary variable.

ERROR CONDITIONS

NONE

EXTERNAL REFERENCES

AFUNF - To compute function for critical point iteration.
AXINV - Determines the involute of an angle.
USDMMY - Dummy UTILPAK routine for USCLVS routine.
USOLVS - Solve for roots of 1 dimensional linear equation.
UXPOKR - Check for error in indicated real exponentiation.

COMMENTS

NONE

LOCAL VARIABLES

AAA -DP VBL-Work variable.
ALPHN -DP VBL-Angle of surface normal.
BBB -DP VBL-Work variable.
C1 -DP VBL-Distance along line of action.
C4 -DP VBL-Distance along line of action.
C C6 -DP VBL- DISTANCE ALONG LINE OF ACTION
C CR -DP VBL- OPERATING CENTER DISTANCE
C DSN -DP VBL- Amount of thinning for backlash
C F -DP VBL- EFFECTIVE FACE WIDTH
C H -RL VBL- PARAMETER FOR STRESS CORRECTION FACTOR
C HLEWIS -DP VBL- Height of Lewis parabola.
C KF -DP VBL- Dolan stress conc.
C L -RL VBL- PARAMETER FOR STRESS CORRECTION FACTOR
C M -RL VBL- PARAMETER FOR STRESS CORRECTION FACTOR
C PB1 -DP VBL- TRANSVERSE BASE PITCH
C PHIN1 -DP VBL- SUCCESSIVE GENERATING PRESSURE ANGLE
C PHIN2 -DP VBL- GENERATING PRESSURE ANGLE
C PHIN2I -DP VBL- INVOLUTE OF GENERATING PRESSURE ANGLE
C PHIND -DP VBL- Normal pressure angle (deg)
C PHINL -DP VBL- Pressure angle at load
C PHINR -DP VBL- OPERATING NORMAL PRESSURE ANGLE
C PHINS -DP VBL- PRESSURE ANGLE AT CENTER OF TOOL TIP RADIUS
C PHINW -DP VBL- PRESSURE ANGLE AT LOAD APPLICATION POINT
C PHIR -DP VBL- OPERATING TRANSVERSE PRESSURE ANGLE
C PSI1 -DP VBL- OPERATING HELIX ANGLE
C RI -DP VBL- PITCH RADIUS OF PINION
C RB1 -DP VBL- BASE RADIUS OF PINION
C RN2 -DP VBL- BASE RADIUS OF GEAR
C RF -RL VBL- MINIMUM RADIUS OF CURVATURE OF FILLET CURVE
C RN -DP VBL- REFERENCE PITCH RADIUS OF VIRTUAL SPUR GEAR
C RNA -DP VBL- VIRTUAL OUTSIDE RADIUS
C RN8 -DP VBL- VIRTUAL BASE RADIUS
C RNO -DP VBL- VIRTUAL BASE RADIUS OF TOOL
C RNO -DP VBL- REFERENCE PITCH RADIUS OF VIRTUAL TOOL
C RG1 -DP VBL- OUTSIDE RADIUS OF PINION
C RG2 -DP VBL- OUTSIDE RADIUS OF GEAR
C SN -DP VBL- REFERENCE NORMAL CIRCULAR TOOTH THICKNESS
C SN0 -DP VBL- REFERENCE NORMAL CIRCULAR TOOTH THICK OF TOOL
C TN1 -DP VBL- PINION TOOTH NUMBER
C TN2 -DP VBL- GEAR TOOTH NUMBER
C TN3 -DP VBL- Number of teeth on cutter.
C W -DP VBL- ANGLE OF INCLINATION OF HELICAL CONTACT LINE
C X1 -DP VBL- addendum modification of the tool
C XGO -DP VBL- GENERATING RACK SHIFT COEFFICIENT
C Y -DP VBL- Lewis form factor.
C
C IMPLICIT DOUBLE PRECISION (A-Z)
C
C DOUBLE PRECISION PI, DEGRAD
C DOUBLE PRECISION PH, TNP, TNG
C DOUBLE PRECISION CDP, CDP, CD, CML
C DOUBLE PRECISION PHN, PHIT, PSI, PSIB, LSR
C DOUBLE PRECISION ADC, HK, BLASH
C DOUBLE PRECISION TNC, X0, X1, HA0, RAO, DEL0, DSN
C INTEGER IGR, FZ, IBRR
C DOUBLE PRECISION XJ
C
C DOUBLE PRECISION RNO, RSN0, RNN, TNO, TN, GNS, RNL
C DOUBLE PRECISION UNO, AKF, BETH, ENF, MNN
C DOUBLE PRECISION ALPHN
REAL H, L, M, HF, RF, SF
LOGICAL UXPOKR

C
COMMON /ACONST/ PI, DEGRAD
COMMON /DERIV/ UNO, AKF, BETN, ENF, MNF, HF
COMMON /FUNCT/ DRA0, RNO2, RSNO, RN2, TNO, TN, GNS, RNL

C
EXTERNAL AXINV, AFUNJF
EXTERNAL USDMMY, UXPOKR

C
IERR = 1
TNC = 10000
F2 = 2
PHIND = PHIN*180.0/PI
DRA0 = RA0

C
Determine whether pinion or gear is being analyzed.

C
IF (IGORP .EQ. 1) THEN
   TN1 = TNP
   TN2 = TNG
   RO1 = 0.5*ODP
   RO2 = 0.5*ODG
ELSE
   TN1 = TNG
   TN2 = TNP
   RO1 = 0.5*ODG
   RO2 = 0.5*ODP
ENDIF

C
NORMALIZE TO PN = 1.0

C
CR = CD*PN
F = FWS*PN
RO1 = RO1*PN
RO2 = RO2*PN

C
equal thinning of pinion and gear is assumed

C
DSN = 0.5*BLASH*PN

C
compute the modification on pinion/gear

C
X1 = ADC-0.5*HK

C
BASIC GEOMETRY

C
R1 = TN1/ (2*COS (PSI) )
RB1 = R1*COS (PH1)
RB2 = RB1*TN2/TN1
PB1 = 2.0*PI*RB1/TN1

IF (ABS ( (RB2 + RB1) /CR) .GT. 1.0) RETURN
PHIR = ACOS ( (RB2 + RB1) /CR)
C6 = CR*SIN (PHIR)
C1 = C6 - SQRT (RO2**2 - RB2**2)
C4 = C1 + PB1

C
IF (F2 .EQ. 1) THEN

Tip loaded spur gears

TN = TN1
RN = R1
RNB = RB1
PHINW = ATAN (C4/RNB)
ELSE

Load at HPSTC

TN = TN1/ ( (COS (PSI) ) **3)
RN = TN/2
RNB = RN*COS (PHIN)
RNA = RN + RO1 - R1
PHINW = ATAN (SQRT ( (RNA/RNB) **2 - 1 )
ENDIF

XGO = X1 - (DSN/ (2*TAN (PHIN) ) )

SN = REFERENCE NORMAL CIRCULAR TOOTH THICKNESS

SN = PI/2 + (2*XGO*TAN (PHIN) )
PHINL = TAN (PHINW) - TAN (PHIN) + PHIN - SN/TN
RNL = RNB/COS (PHINL)

Virtual shaper cutter geometry

TNO = TNC/ (COS (PSI) **3)
RNO = TNO/2
RNBO = RNO*COS (PHIN)
ESNO = RNO + HA0 + X0 - DRA0
PHINS = ACOS (RNBO/RSNO)
SNO = PI/2 + 2*X0*TAN (PHIN)
GNS = 2* (AXINV(PHIN) + SNO/TNO - AXINV(PHINS) +
* (DEL0 - DRA0) /RNBO)

Iteration for generating pressure angle

PHIN2I = AXINV(PHIN) + (2* (XGO + X0)*TAN(PHIN) ) / (TN + TNO)
IF ( PHIN2I .LE. 0. ) RETURN
PHIN2 = ( (3*PHIN2I) ** (1./3.) )

90 CONTINUE
PHIN1 = PHIN2 + (PHIN2I + PHIN2 - TAN(PHIN2) )/(TAN(PHIN2)**2)
IF (ABS (PHIN1 - PHIN2) .LE. 1.0E - 06) GO TO 100
PHIN2 = PHIN1
GO TO 90
100 CONTINUE
RN2 = RN*COS (PHIN) /COS (PHIN2)
RNO2 = RNO*COS (PHIN) /COS (PHIN2)

C
C Iteration for critical point
C
ALPHN = PI/4
CALL USOLVS (AFUNJF, ALPHN, 0.5, 2, 40, 0, USDMNY, 1.0D-6, 0.0,
* ALPHN, IERR)
IF (IERR .NE. 0) THEN
  RETURN
ENDIF

C
RF = DRA0 + (RNO2 - RSNO) **2/ (RN2*RNO2/ (RN2 + RNO2) -
  (RNO2 - RSNO) )
W = ATAN (TAN (PSI) *SIN (PHIN) )
W = W*180.0/PI
CH = 1.0/ (1.0 - SQRT (W* (1.0 - W/100.0) ) /10.0)
SF = 2*ENF

C
C COMPUTING DOLAN AND BROGHAMER STRESS CORRECTION FACTOR
C
IF (PHIND .EQ. 20.0) THEN
  H = 0.18
  L = 0.15
  M = 0.45
ELSE
  IF (PHIND .EQ. 14.5) THEN
    H = 0.22
    L = 0.20
    M = 0.40
  ELSE
    IF (PHIND .EQ. 25.0) THEN
      H = 0.14
      L = 0.11
      M = 0.50
    ELSE
      H = 0.18 - 0.008* (PHIND - 20.0)
      L = H - 0.03
      M = 0.45 + 0.01* (PHIND - 20.0)
    ENDIF
  ENDIF
ENDIF

C
IF ( .NOT. (UXPokr (SF/RF, L) .AND. UXPokr (SF/HF, M) ) ) RETURN
C
KF = H + ((SF/RF)**L) * ((SF/HF)**M)

C
IF (PSI .LT. 0.002) THEN
  PSIR = 0.0
  PHINR = PHIR
ELSE
  PSIR = ATAN (TAN (PSIB) /COS (PHIR) )
  PHINR = ACOS (SIN (PSIB) /SIN (PSIR) )
ENDIF

C
AAA = COS (PSIR) *COS (PSI)
BBB = COS(PHINL)/COS(PHINR) * (6*HF/ (SF**2*CH) - TAN(PHINL)/SF)
Y = AAA/BBB

DETERMINE THE AGMA GEOMETRY FACTOR (J)

XJ = Y/ (KF*LSR)

ALSO RETURN HEIGHT OF LEWIS PARABOLA

HLEWIS = HF

IERR = 0
RETURN
END

******************************************************************************** GOCAD ********************************************************************************

SUBROUTINE AFUNJF (ALPHN, Y, IERR)

******************************************************************************** GOCAD ********************************************************************************

AFUNJF - To compute function for critical point iteration.

AUTHOR - RAJIV AGRAWAL 19-JUN-86
REVISED- Rajiv Agrawal 23-SEP-91

PROGRAM DESCRIPTION
SUBROUTINE TO EVALUATE FUNCTION VALUE FOR JFACT PROGRAM.

ACCESS
CALL AFUNJF (ALPHN, Y, IERR)
ALPHN -DP VBL-USD- ANGLE OF SURFACE NORMAL.
Y -DP VBL-USD- ITERATION FUNCTION.
IERR -IN VBL-RTD- ERROR CODE FOR USOLVS.

COMMON BLOCK VARIABLES
AKF -DP VBL-RTD-/DERIV /* TEMPORARY VARIABLE.
BETN -DP VBL-RTD-/DERIV /* TEMPORARY VARIABLE.
DEGRAD -DP VBL-xxx/-ACONST/-
DRA0 -DP VBL-USD-/FUNCT /* TOOL TIP RADIUS
ENF -DP VBL-RTD-/DERIV /* ABCISSA OF CRITICAL POINT
GNS -DP VBL-USD-/FUNCT /* ANGLE TO CENTER OF TOOL TIP RADIUS
HF -RL VBL-RTD-/DERIV /* HEIGHT OF LEWIS PARABOLA
MNF -DP VBL-RTD-/DERIV /* ORDINATE OF CRITICAL POINT
PI -DP VBL-USD-/ACONST/- THE CONSTANT 3.14159.
RN2 -DP VBL-USD-/FUNCT /* GEN. PITCH RADIUS OF VIRTUAL SPUR.
RNL -DP VBL-USD-/FUNCT /* VIRTUAL LOAD RADIUS
RNO2 -DP VBL-USD-/FUNCT /* GEN. PITCH RADIUS OF VIRTUAL TOOL.
RSNO -DP VBL-USD-/FUNCT /* RADIUS TO CENTER OF TOOL TIP RADIUS
TN -DP VBL-USD-/FUNCT /* VITUAL TOOTH NUMBER OF GEAR/PINION
TNO -DP VBL-USD-/FUNCT /* VIRTUAL TOOTH NUMBER OF TOOL
UNO -DP VBL-RTD-/DERIV /* TEMPORARY VARIABLE.

ERROR CONDITIONS
NONE
EXTERNAL REFERENCES
NONE

COMMENTS
NONE

LOCAL VARIABLES
AKS -DP VBL- TEMPORARY VARIABLE.
TEMPRY -DP VBL- TEMPORARY VARIABLE.
THETN -DP VBL- TEMPORARY VARIABLE.

IMPLICIT DOUBLE PRECISION (A-Z)

COMMON /ACONST/ PI, DEGRAD
COMMON /DERIV/ UNO, AKF, BETN, ENF, MNF, HF
COMMON /FUNCT/ DRA0, RNO2, RSNO, RN2, TNO, TN, GNS, RNL

REAL HF
INTEGER IERR

IERR = 0

ALPHN SHOULD BE BETWEEN 0.0 AND PI/4

IF (ALPHN .LT. 0.0 .OR. ALPHN .GT. PI/4) THEN
   IERR = 1
   RETURN
ENDIF

TEMPRY = RNO2*COS (ALPHN) /RSNO
IF (ABS (TEMPRY) .GT. 1.0) THEN
   IERR = 1
   RETURN
ENDIF
UNO = ACOS (TEMPRY) - ALPHN

AKS = RNO2*SIN (ALPHN) - RSNO*SIN (ALPHN + UNO)
AKF = AKS - DRA0
THETN = TNO/TN* (UNO - GNS/2 + PI/TNO)
BETN = ALPHN - THETN
ENF = RN2*SIN (THETN) + AKF*COS (BETN)
MNF = RN2*COS (THETN) + AKF*SIN (BETN)
HF = RNL - MNF
Y = 2*HF*TAN (BETN) - ENF
RETURN
END

******************************************************************** GOCAD ********************************************************************

DOUBLE PRECISION FUNCTION AXINV(X)

******************************************************************** GOCAD ********************************************************************

AXINV - Determines the involute of an angle.

AUTHOR - RAJIV AGRAWAL  19-JUN-86
REVISED - Rajiv Agrawal 23-SEP-91

PROGRAM DESCRIPTION

THIS FUNCTION DETERMINES THE INVOLUTE OF THE ANGLE.

ACCESS

USE A FUNCTION REFERENCE OF THE FORM

...AXINV(X)
X -DP VBL-USD- AN ANGLE IN RADIANS.

COMMON BLOCK VARIABLES

NONE

ERROR CONDITIONS

NONE

EXTERNAL REFERENCES

NONE

COMMENTS

NONE

LOCAL VARIABLES

NONE

DOUBLE PRECISION X

AXINV = TAN(X) - X

RETURN

END

********************************************************************************** GOCAD **********************************************************************************

DOUBLE PRECISION FUNCTION AIFACT(CD, CRF, FWIDTH, GRATIO,
* PBT, PHIT, PSI, PSIB,
* BDP, ODG, ODP, PDG, PDP, ZLC, LSR,
* IERR)

********************************************************************************** GOCAD **********************************************************************************

AIFACT - Determines wear geometry factors (XIP, XIG).

AUTHOR - RAJIV AGRAWAL 19-JUN-86
REVISED - Rajiv Agrawal 23-SEP-91

PROGRAM DESCRIPTION

THIS SUBROUTINE DETERMINES THE AGMA GEOMETRY FACTOR FOR WEAR
ACCORDING TO THE STANDARDS.

ACCESS

USE A FUNCTION REFERENCE OF THE FORM

...AIFACT(CD, CRF, FWIDTH, GRATIO,
* PBT, PHIT, PSI, PSIB,
* BDP, ODG, ODP, PDG, PDP, ZLC, LSR,
* IERR)
CD  -DP VBL-xxx-  Center distance
CRF  -DP VBL-xxx-  Face contact ratio
FWIDTH  -DP VBL-xxx-  Face width
GRATIO  -DP VBL-xxx-  Gear ratio
PBT  -DP VBL-xxx-  Transverse base pitch
PHIT  -DP VBL-xxx-  Transverse pressure angle
PSI  -DP VBL-xxx-  Helix angle
PSIB  -DP VBL-xxx-  Base helix angle
BDP  -DP VBL-xxx-  Base dia (pinion)
ODG  -DP VBL-xxx-  Outside dia (gear)
ODP  -DP VBL-xxx-  Outside dia (pinion)
PDG  -DP VBL-xxx-  Pitch dia (gear)
PDP  -DP VBL-xxx-  Pitch dia (pinion)
ZLC  -DP VBL-xxx-  Length of line of action
LSR  -DP VBL-xxx-  Load sharing ratio
IERRE  -IN VBL-xxx-  Error code.

COMMON BLOCK VARIABLES

DEGRAD  -DP VBL-xxx-/ACONST/-  Degree to radian conversion.
PI  -DP VBL-xxx-/ACONST/-  THE CONSTANT 3.14159.

ERROR CONDITIONS

NONE

EXTERNAL REFERENCES

NONE

COMMENTS

NONE

LOCAL VARIABLES

R1  -RL VBL-  TEMPORARY VARIABLE.
R1H  -RL VBL-  TEMPORARY VARIABLE.
R2  -RL VBL-  TEMPORARY VARIABLE.
R2H  -RL VBL-  TEMPORARY VARIABLE.
RM  -RL VBL-  TEMPORARY VARIABLE.
XI  -RL VBL-  I-factor.
ZA  -RL VBL-  TEMPORARY VARIABLE.
ZC  -RL VBL-  TEMPORARY VARIABLE.
ZCH  -RL VBL-  TEMPORARY VARIABLE.

DOUBLE PRECISION CD, FWIDTH, GRATIO
DOUBLE PRECISION CRF, ZLC
DOUBLE PRECISION PBT
DOUBLE PRECISION PHIT, PSI, PSIB
DOUBLE PRECISION BDP, ODG, ODP, PDG, PDP
DOUBLE PRECISION LSR
INTEGER IERR

DOUBLE PRECISION PI, DEGRAD
COMMON /ACONST/ PI, DEGRAD

IERRE  = 1

RBP  = 0.5*BDP
ROG  = 0.5*ODG
ROP = 0.5*ODP
RPG = 0.5*PDG
RPP = 0.5*PDP

C

DEFINE THE LENGTH OF THE LINE OF ACTION.
C

IF (CRF .LE. 1.) THEN
    IF (ROP .LT. RBP .OR. RPP .LT. RBP) RETURN
    ZA = (SQRT(ROP*ROP - RBP*RBP) - SQRT(RPP*RPP - RBP*RBP))
    ZC = PBT - ZA
ELSE
    RM = (CD - ROG + ROP) /2.
    IF (RPP .LT. RBP .OR. RM .LT. RBP) RETURN
    ZC = SQRT (RPP*RPP - RBP*RBP) - SQRT (RM*RM - RBP*RBP)
ENDIF

C

DETERMINE THE GEOMETRY FACTOR FOR WEAR.
C

CG = GRATIO/(GRATIO + 1.)
CC = COS(PHIT)*SIN(PHIT)*CG/2.

C

PROP = RPP*SIN(PHIT)
PROG = RPG*SIN(PHIT)
R1 = PROP - ZC
R2 = PROG + ZC
IF (PROP .EQ. 0.0 .OR. PROG .EQ. 0.0) RETURN
CX = R1*R2/ (PROP*PROG)
CPSH = 1.0
IF (PSI .GE. 0.002 .AND. CRF .LT. 1.0) THEN

C

LOW FACE CONTACT RATIO (LCR) HELICAL GEARS
C

RM = (CD - ROG + ROP) /2.0
IF (RPP .LT. RBP .OR. RM .LT. RBP) RETURN
ZCH = SQRT(RPP*RPP - RBP*RBP) - SQRT(RM*RM - RBP*RBP)
R1H = PROP - ZCH
R2H = PROG + ZCH
CXH = R1H*R2H/ (PROG*PROP)
IF (CX .EQ. 0.0 .OR. FWIDTH .EQ. 0.0) RETURN
CPSH = 1.0 - CRF + CXH*ZLC*CPSH*CRF/ (CWX*FWIDTH*SIN(PSIB))
ENDIF
IF (LSR .EQ. 0.0) RETURN
XI = CC*CPSH/LSR

C

AIFACT = XI
IERR = 0

C

RETURN
END
APPENDIX C

Graphics Module for Helical Gear Design
Appendix C

Graphics Module for Helical Gear Design

SUBROUTINE IOCDRW ( X, NVARS)
REAL X(NVARS)

C
COMMON /ACASE/ ICASE

C
IF (ICASE .EQ. 1) THEN
   CALL DRWHEL(X,NVARS)
ENDIF

C
RETURN
END

C
SUBROUTINE DRWHEL(X,NVARS)
REAL X(NVARS)

C
CD = X(12)
ODP = X(37)
PDP = X(38)
BDP = X(39)
RDP = X(40)
ODG = X(41)
PDG = X(42)
BDG = X(43)
RDG = X(44)

C
XL = 1.5*A MAX1 (ODP, ODG)
IF (XL.EQ.0.) XL=1.0
YL = 1.2*(ODP+ODG)
IF (YL.EQ.0.0) YL=1.0
XSCLAE = 0.4/XL
YSCLAE = 0.4/YL
IF (XSCLAE.LT.YSCLAE) THEN
   XMAX = XL
   YMAX = XL
ELSE
   XMAX = YL
   YMAX = YL
ENDIF
CALL KWPRT (0.6,0.998,.24,0.638)
CALL KWINDO(0.,XMAX,0.,YMAX)

C
XCEN = 0.5*XMAX
YCEN = YMAX-0.65*ODP
CALL KMOVAB(XCEN,YCEN+0.55*ODP)
CALL KLINDX(7)
CALL KLNTYP('DASHDOT')
CALL KDRWRL(0.,-1.05*(ODP+ODG))

C
CALL KLNIDX(3)
CALL KLNTP('SOLID')
CALL KCIRCL(0.5*ODP,XCEN,YCEN,0.0,360.0)
CALL KLNIDX(2)
CALL KLNTP('DASHDOT')
CALL KCIRCL(0.5*PDP,XCEN,YCEN,0.0,360.0)
CALL KLNIDX(6)
CALL KLNTP('SOLID')
CALL KCIRCL(0.5*RDP,XCEN,YCEN,0.0,360.0)
CALL KLNTP('DOTTED')
CALL KCIRCL(0.5*BDP,XCEN,YCEN,0.0,360.0)

CALL KMOVAB(XCEN,YCEN)
CALL KMOVRL(-0.6*ODP,0.)
CALL KLNTP('DASHDOT')
CALL KLNIDX(7)
CALL KDRWRL(1.2*ODP,0.)

VCEN = YCEN-CD
CALL KLNIDX(3)
CALL KLNTP('SOLID')
CALL KCIRCL(0.5*ODG,XCEN,YCEN,0.0,360.0)
CALL KLNIDX(2)
CALL KLNTP('DASHDOT')
CALL KCIRCL(0.5*PDG,XCEN,YCEN,0.0,360.0)
CALL KLNIDX(6)
CALL KLNTP('SOLID')
CALL KCIRCL(0.5*RDG,XCEN,YCEN,0.0,360.0)
CALL KLNTP('DOTTED')
CALL KCIRCL(0.5*BDG,XCEN,YCEN,0.0,360.0)

CALL KMOVAB(XCEN,YCEN)
CALL KMOVRL(-0.6*ODG,0.)
CALL KLNTP('DASHDOT')
CALL KLNIDX(7)
CALL KDRWRL(1.2*ODG,0.)
RETURN
END