CHAPTER

7

Optimum Design with MATLAB®

Upon completion of this chapter, you will be able to:

- Use the capabilities of the Optimization Toolbox in MATLAB to solve both unconstrained and constrained optimization problems

MATLAB was used in chapter: Graphical Solution Method and Basic Optimization Concepts to graphically solve two-variable optimization problems. In chapter: Optimum Design Concepts: Optimality Conditions, it was used to solve a set of nonlinear equations obtained as Karush–Kuhn–Tucker (KKT) optimality conditions for constrained optimization problems. In this chapter, we describe the capabilities of the Optimization Toolbox in MATLAB to solve linear, quadratic, and nonlinear programming problems. We start by describing the basic capabilities of this toolbox. Some operators and syntax used to enter expressions and data are described. In subsequent sections, we illustrate the use of the program for unconstrained and constrained optimization problems. Then some engineering design optimization problems are solved using the program (original draft of this chapter was provided by Tae Hee Lee and his contribution to this book is highly appreciated).

7.1 INTRODUCTION TO THE OPTIMIZATION TOOLBOX

7.1.1 Variables and Expressions

MATLAB can be considered a high-level programming language for numerical computation, data analysis, and graphics for applications in many fields. It interprets and evaluates expressions entered at the keyboard. The statements are usually in the form `variable = expression`. The variables can be scalars, arrays, or matrices. Arrays may store many variables at a time. A simple way to define a scalar, array, or matrix is to use assignment statements as follows:

$$a = 1;\quad b = [1, 1];\quad c = [1, 0, 0;\quad 1, 1, 0;\quad 1, -2, 1] \tag{7.1}$$

Introduction to Optimum Design. http://dx.doi.org/10.1016/B978-0-12-800806-5.00007-X

Note that several assignment statements can be entered in one row as in Eq. (7.1). A semicolon (;) at the end of a statement prevents the program from displaying the results. The variable `a` denotes a scalar that is assigned a value of 1; the variable `b` denotes a 1 × 2 row vector, and the variable `c` denotes a 3 × 3 matrix assigned as follows:

$$\mathbf{b} = [1 \ \ 1]; \quad \mathbf{c} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & -2 & 1 \end{bmatrix} \tag{7.2}$$

The semicolons inside the brackets of the expression for c separate the rows, and the values in the rows can be separated by commas or blanks. MATLAB has a rule that the *variable name* must be a single word without spaces, and it must start with a letter followed by any number of letters, digits, or underscores. It is important to note that variable names are case sensitive. In addition, there are several built-in variables, for example, `pi` for the ratio of the circumference of a circle to its diameter; `eps` for the smallest number in the computer; `inf` for infinity, and so on.

7.1.2 Scalar, Array, and Matrix Operations

The arithmetic operators for scalars in MATALB are: addition (+), subtraction (−), multiplication (*), division (/), and exponentiation (^). Vector and matrix calculations can also be organized in a simple way using these operators. For example, multiplication of two matrices **A** and **B** is expressed as **A**.***B**. Slight modification of the standard operators with a "dot" prefix is used for element-by-element operations between vectors and matrices: (.*) for multiplication, (./) for division, and (.^) for exponentiation.

For example, element-by-element multiplication of vectors of the same dimension is accomplished using the operator".*" (dot star):

$$\mathrm{c} = a.*b = \begin{bmatrix} a_1 b_1 \\ a_2 b_2 \\ a_3 b_3 \end{bmatrix} \tag{7.3}$$

Here `a`, `b`, and `c` are column vectors with three elements. For addition and subtraction, element-by-element and usual matrix operations are the same. Other useful matrix operators are: $\mathbf{A}^2 = \mathbf{A}.{*}\mathbf{A}$, $\mathbf{A}^{-1} = \mathrm{inv}(\mathbf{A})$, determinant as det(**A**), and transpose as **A**′.

7.1.3 Optimization Toolbox

The Optimization Toolbox for MATLAB can solve unconstrained and constrained optimization problems. In addition, it has an algorithm to solve nonsmooth optimization problems. Some of the optimization algorithms implemented in the Optimization Toolbox are presented in chapters: Numerical Methods for Unconstrained Optimum Design; More on Numerical Methods for Unconstrained Optimum Design; Numerical Methods for Constrained Optimum Design; More on Numerical Methods for Constrained Optimum Design. These algorithms are based on the basic concepts of algorithms for smooth and nonsmooth problems that are presented in Section 6.1. That material should be reviewed at this point.

TABLE 7.1 Optimization Toolbox Functions

Problem type	Formulation	MATLAB function
One-variable minimization in fixed interval	Find $x \in [x_L\ x_U]$ to minimize $f(x)$	`fminbnd`
Unconstrained minimization	Find $\mathbf{x}$ to minimize $f(\mathbf{x})$	`fminunc` `fminsearch`
Constrained minimization: Minimize a function subject to linear inequalities and equalities, nonlinear inequalities and equalities, and bounds on the variables	Find x to minimize $f(\mathbf{x})$ subject to $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{Nx} = \mathbf{e}$ $g_i(\mathbf{x}) \le 0$, $i = 1$ to m $h_j = 0$, $j = 1$ to p $x_{iL} \le x_i \le x_{iU}$	`fmincon`
Linear programming: minimize a linear function subject to linear inequalities and equalities	Find x to minimize $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x}$ subject to $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{Nx} = \mathbf{e}$	`linprog`
Quadratic programming: Minimize a quadratic function subject to linear inequalities and equalities	Find $\mathbf{x}$ to minimize $f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{Hx}$ subject to $\mathbf{Ax} \le \mathbf{b}$, $\mathbf{Nx} = \mathbf{e}$	`quadprog`

The Optimization Toolbox must be installed in the computer in addition to the basic MATLAB program before it can be used. Table 7.1 shows some of the optimization functions available in the toolbox. Most of these functions require m-files (stored in the current directory) containing a definition of the problem to be solved; several such files are presented and discussed later. Default optimization parameters are used extensively; however, they can be modified through an options command available in the program.

The syntax of invoking an optimization function is generally of the form:

```
[x,FunValue,ExitFlag,Output]=fminX('ObjFun',...,options)
```
(7.4)

The left side of the statement represents the quantities returned by the function. These output arguments are described in Table 7.2. On the right side, `fminX` represents one of the functions given in Table 7.1. There can be several arguments for the function `fminX`, for example, starting values for the variables; upper and lower bounds for the variables; m-file names

TABLE 7.2 Explanation of Output From Optimization Function

Argument	Description
x	The solution vector or matrix found by the optimization function. If `ExitFlag` > 0 then x is a solution, otherwise x is the latest value from the optimization routine.
`FunValue`	Value of the objective function, `ObjFun`, at the solution x.
`ExitFlag`	The exit condition for the optimization function. If `ExitFlag` is positive then the optimization routine converged to a solution x. If `ExitFlag` is zero then the maximum number of function evaluations was reached. If `ExitFlag` is negative then the optimization routine did not converge to a solution.
`Output`	The `Output` structure contains several pieces of information about the optimization process. It provides the number of function evaluations (`Output.iterations`), the name of the algorithm used to solve the problem (`Output.algorithm`), and Lagrange multipliers for constraints, etc.

containing problem functions and their gradients; optimization algorithm–related data; and so on. Use of this function is demonstrated in sections 7.2 to 7.4 for various types of problems and conditions. For further explanation of various functions and commands, extensive online help is available in MATLAB.

7.2 UNCONSTRAINED OPTIMUM DESIGN PROBLEMS

In this section, we first illustrate the use of the fminbnd function for minimization of a function of single variable $f(x)$ with bounds on x as $x_L \le x \le x_U$. Then the use of the function fminunc is illustrated for minimization of a function $f(\mathbf{x})$ of several variables. The m-files for the problems, containing extensive comments, are included to explain the use of these functions. Example 7.1 demonstrates use of the function `fminbnd` for functions of single variable, and Example 7.2 demonstrates the use of functions `fminsearch` and `fminunc` for multivariable unconstrained optimization problems.

EXAMPLE 7.1 SINGLE-VARIABLE UNCONSTRAINED MINIMIZATION

Find x to
Minimize

$$f(x) = 2 - 4x + e^x, \; -10 \le x \le 10 \tag{a}$$

Solution

To solve this problem, we write an m-file that returns the objective function value. Then we invoke `fminbnd`, the single-variable minimization function, in fixed intervals through another m-file that is shown in Table 7.3. The file that evaluates the function, shown in Table 7.4, is called through `fminbnd`.

TABLE 7.3 m-File for Calling `fminbnd` to Find Single Variable Minimizer in Fixed Interval for Example 7.1

```
% All comments start with %
% File name: Example7_1.m
% Problem: minimize f(x) = 2 − 4x + exp(x)
  clear all
% Set lower and upper bound for the design variable
  Lb = -10; Ub = 10;
% Invoke single variable unconstrained optimizer fminbnd;
% The argument ObjFunction7_1 refers to the m-file that
% contains expression for the objective function
  [x,FunVal,ExitFlag,Output] = fminbnd('ObjFunction7_1',Lb,Ub)
```

TABLE 7.4 m-File for Objective Function for Example 7.1

% File name: ObjFunction7_1.m

% Example 7.1 Single variable unconstrained minimization

```
function f = ObjFunction7_1(x)
f = 2 - 4*x + exp(x);
```

The output from the function is

`x = 1.3863, FunVal = 0.4548, ExitFlag = 1 > 0` (ie, minimum was found),
`output` = (iterations: 14, funcCount: 14, algorithm: golden section search, parabolic interpolation).

EXAMPLE 7.2 MULTIVARIABLE UNCONSTRAINED MINIMIZATION

Consider a two-variable problem:
Minimize

$$f(\mathrm{x}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2 \text{ starting from } \mathrm{x}^{(0)} = (-1.2, 1.0) \tag{a}$$

Solve the problem using different algorithms available in the Optimization Toolbox.

Solution

The optimum solution for the problem is known as $\mathbf{x}^* = (1.0, 1.0)$ with $f(\mathbf{x}^*) = 0$ (Schittkowski, 1987). The syntax for the functions `fminsearch` and `fminunc` used to solve a multivariable unconstrained optimization problem is given as follows:

```
[x,FunValue,ExitFlag,Output]=fminsearch('ObjFun',x0,options)   (b)
```

```
[x,FunValue,ExitFlag,Output]=fminunc('ObjFun',x0,options)   (c)
```

where

`ObjFun` = the name of the m-file that returns the function value and its gradient if programmed
`x0` = the starting values of the design variables
`options` = a data structure of parameters that can be used to invoke various conditions for the optimization process

`fminsearch` uses the Simplex search method of Nelder–Mead, which does not require numerical or analytical gradients of the objective function (see Subsection 11.9.3 for details of this algorithm). Thus it is a *nongradient-based method* (*direct search method*) that can be used for problems where the cost function is not differentiable.

Since `fminunc` does require the gradient value, with the option `LargeScale` set to `off`, it uses the BFGS quasi-Newton method (refer to chapter: More on Numerical Methods for Unconstrained Optimum Design for details) with a mixed quadratic and cubic line search procedure. The DFP formula (refer to chapter: More on Numerical Methods for Unconstrained Optimum Design, for details), which approximates the inverse Hessian matrix, can be selected by setting the option `HessUpdate` to `dfp`. The steepest-descent method can be selected by setting option `HessUpdate` to `steepdesc`. `fminsearch` is generally less efficient than `fminunc`. However, it can be effective for problems for which the gradient evaluation is expensive or not possible.

To solve this problem, we write an m-file that returns the objective function value. Then, the unconstrained minimization function `fminsearch` or `fminunc` is invoked through execution of another m-file, shown in Table 7.5. The m-file for function and gradient evaluations is shown in Table 7.6.

The gradient evaluation option can be omitted if automatic evaluation of gradients by the finite difference method is desired. Three solution methods are used, as shown in Table 7.5. All methods converge to the known solution.

TABLE 7.5 m-File for Unconstrained Optimization Routines for Example 7.2

```
% File name: Example7_2
% Rosenbruck valley function with analytical gradient of
% the objective function
  clear all
  x0 = [-1.2 1.0]';  Set starting values
% Invoke unconstrained optimization routines
% 1. Nelder-Mead simplex method, fminsearch
  % Set options: medium scale problem, maximum number of function evaluations
  % Note that "…" indicates that the text is continued on the next line
    options = optimset('LargeScale', 'off', 'MaxFunEvals', 300);
            [x1, FunValue1, ExitFlag1, Output1] = …
             fminsearch ('ObjAndGrad7_2', x0, options)
% 2. BFGS method, fminunc, dafault option
  % Set options: medium scale problem, maximum number of function evaluations,
  % gradient of objective function
    options = optimset('LargeScale', 'off', 'MaxFunEvals', 300,…
            'GradObj', 'on');
    [x2, FunValue2, ExitFlag2, Output2] = …
            fminunc ('ObjAndGrad7_2', x0, options)
% 3. DFP method, fminunc, HessUpdate = dfp
  % Set options: medium scale optimization, maximum number of function evaluation,
  % gradient of objective function, DFP method
    options = optimset('LargeScale', 'off', 'MaxFunEvals', 300, …
            'GradObj', 'on', 'HessUpdate', 'dfp');
    [x3, FunValue3, ExitFlag3, Output3] = …
            fminunc ('ObjAndGrad7_2', x0, options)
```

TABLE 7.6 m-File for Objective Function and Gradient Evaluations for Example 7.2

% File name: ObjAndGrad7_2.m

% Rosenbrock valley function

```
function [f, df] = ObjAndGrad7_2(x)
```

% Re-name design variable x

```
x1 = x(1); x2 = x(2);  %
```

% Evaluate objective function

```
f = 100*(x2 - x1^2)^2 + (1 - x1)^2;
```

% Evaluate gradient of the objective function

```
df(1) = -400*(x2-x1^2)*x1 - 2*(1-x1);
df(2) = 200*(x2-x1^2);
```

7.3 CONSTRAINED OPTIMUM DESIGN PROBLEMS

The general constrained optimization problem treated by the function `fmincon` is defined in Table 7.1. The procedure for invoking this function is the same as for unconstrained problems except that an m-file containing the constraint functions must also be provided. If analytical gradient expressions are programmed in the objective function and constraint functions m-files, they are declared through the `options` command. Otherwise, `fmincon` uses numerical gradient calculations based on the finite difference method. Example 7.3 shows the use of this function for an inequality-constrained problem. Equalities, if present, can be included similarly.

EXAMPLE 7.3 CONSTRAINED MINIMIZATION PROBLEM USING *FMINCON* IN OPTIMIZATION TOOLBOX

Solve the problem:

Minimize

$$f(\mathrm{x}) = (x_1 - 10)^3 + (x_2 - 20)^3 \tag{a}$$

subject to

$$g_1(\mathrm{x}) = 100 - (x_1 - 5)^2 - (x_2 - 5)^2 \le 0 \tag{b}$$

$$g_2(\mathrm{x}) = -82.81 - (x_1 - 6)^2 - (x_2 - 5)^2 \le 0 \tag{c}$$

$$13 \le x_1 \le 100,\ 0 \le x_2 \le 100 \tag{d}$$

Solution

The optimum solution for the problem is known as $\mathbf{x} = (14.095, 0.84296)$ and $f(\mathbf{x}^*) = -6961.8$ (Schittkowski, 1981). Three m-files for the problem are given in Tables 7.7–7.9. The script m-file in Table 7.7 invokes the function `fmincon` with appropriate arguments and options. The function m-file in Table 7.8 contains the cost function and its gradient expressions, and the function m-file in Table 7.9 contains the constraint functions and their gradients.

The problem is solved successfully, and the output from the function is given as

`Active constraints`: 5, 6 [ie, g(1) and g(2)]
$\mathbf{x} = (14.095, 0.843)$, `FunVal` $= -6.9618e + 003$, `ExitFlag` $= 1 > 0$ (ie, minimum was found)
`output` = (iterations: 6, funcCount: 13, stepsize: 1, algorithm: medium scale: SQP, quasi-Newton, line-search).

Note that the `Active constraints` listed at the optimum solution are identified with their index counted as Lb, Ub, inequality constraints, and equality constraints. If `Display off` is included in the options command, the set of active constraints is not printed.

TABLE 7.7 m-File for Constrained Minimizer fmincon for Example 7.3

```
% File name: Example7_3
% Constrained minimization with gradient expressions available
% Calls ObjAndGrad7_3 and ConstAndGrad7_3
  clear all
% Set options; medium scale, maximum number of function evaluation,
% gradient of objective function, gradient of constraints, tolerances
% Note that three periods "..." indicate continuation on next line
  options = optimset ('LargeScale', 'off', 'GradObj', 'on',...
  'GradConstr', 'on', 'TolCon', 1e-8, 'TolX', 1e-8);
% Set bounds for variables
  Lb = [13; 0]; Ub = [100; 100];
% Set initial design
  x0 = [20.1; 5.84];
% Invoke fmincon; four [ ] indicate no linear constraints in the problem
  [x,FunVal, ExitFlag, Output] = …
  fmincon('ObjAndGrad7_3',x0,[ ],[ ],[ ],[ ],Lb, …
  Ub,'ConstAndGrad7_3',options)
```

TABLE 7.8 m-File for Objective Function and Gradient Evaluations for Example 7.3

```
% File name: ObjAndGrad7_3.m
  function [f, gf] = ObjAndGrad7_3(x)
% f returns value of objective function; gf returns objective function gradient
% Re-name design variables x
  x1 = x(1); x2 = x(2);
% Evaluate objective function
  f = (x1-10)^3 + (x2-20)^3;
% Compute gradient of objective function
  if nargout > 1
          gf(1,1) = 3*(x1-10)^2;
          gf(2,1) = 3*(x2-20)^2;
  end
```

TABLE 7.9 Constraint Functions and Their Gradients Evaluation m-File for Example 7.3

```
% File name: ConstAndGrad7_3.m
  function [g, h, gg, gh] = ConstAndGrad7_3(x)
% g returns inequality constraints; h returns equality constraints
% gg returns gradients of inequalities; each column contains a gradient
% gh returns gradients of equalities; each column contains a gradient
% Re-name design variables
  x1 = x(1); x2 = x(2);
% Inequality constraints
  g(1) = 100-(x1-5)^2-(x2-5)^2 ;
  g(2) = -82.81+ (x1-6)^2 + (x2-5)^2;
% Equality constraints (none)
  h = [ ];
% Gradients of constraints
  if nargout > 2
          gg(1,1) = -2*(x1-5);
          gg(2,1) = -2*(x2-5);
          gg(1,2) = 2*(x1-6);
          gg(2,2) = 2*(x2-5);
          gh = [];
  end
```

7.4 OPTIMUM DESIGN EXAMPLES WITH MATLAB

7.4.1 Location of Maximum Shear Stress for Two Spherical Bodies in Contact

Project/Problem Statement

There are many practical applications where two spherical bodies come into contact with each other, as shown in Fig. 7.1. We want to determine the maximum shear stress and its location along the z-axis for a given value of the Poisson's ratio of the material, $\nu = 0.3$.

Data and Information Collection

The shear stress along the z-axis σ_{zx} is calculated using the principal stresses as (Norton, 2000):

$$\sigma_{xz} = \frac{p_{\max}}{2}\left(\frac{1-2\nu}{2} + \frac{(1+\nu)\alpha}{\sqrt{1+\alpha^2}} - \frac{3}{2}\frac{\alpha^3}{\left(\sqrt{1+\alpha^2}\right)^3}\right) \tag{a}$$

where $\alpha = z/a$ and a represents the contact-patch radius, as shown in Fig. 7.1. The maximum pressure occurs at the center of the contact patch and is given as

$$p_{\max} = \frac{3P}{2\pi a^2} \tag{b}$$

It is well known that the peak shear stress does not occur at the contact surface but rather at a small distance below it. The subsurface location of the maximum shear stress is believed to be a significant factor in the surface fatigue failure called *pitting*.

Definition of Design Variables

The only design variable for the problem is the normalized depth for the location of the maximum shear stress α $(= z/a)$.

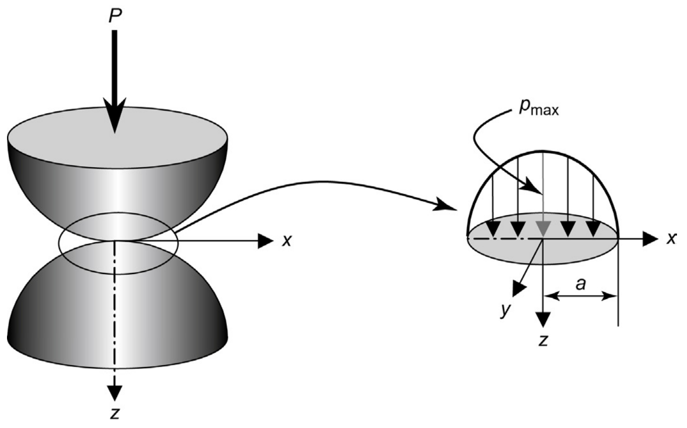

FIGURE 7.1 **Spherical bodies in contact and pressure distribution on contact patch.**

Optimization Criterion

The objective is to locate a point along the z-axis where the shear stress is maximum. Transforming to the standard minimization form and normalizing with respect to $p_{\max}$, the problem becomes finding α to minimize

$$f(\alpha) = -\frac{\sigma_{xz}}{p_{\max}} \tag{c}$$

Formulation of Constraints

There are no constraints for the problem except bounds on the variable α taken as $0 \le \alpha \le 5$.

Solution

The exact solution for the problem is given as

$$\left.\frac{\sigma_{xz}}{p_{\max}}\right|_{\max} = \frac{1}{2}\left(\frac{1-2\nu}{2} + \frac{2}{9}(1+\nu)\sqrt{2(1+\nu)}\right) \text{ at } \alpha = \sqrt{\frac{2+2\nu}{7-2\nu}} \tag{d}$$

This is a single-variable optimization problem with only lower and upper bounds on the variable. Therefore, the function `fminbnd` in the Optimization Toolbox can be used to solve the problem. Table 7.10 shows the script m-file that invokes the function `fminbnd`, and Table 7.11

TABLE 7.10 m-File to Invoke Function fminbnd for Spherical Contact Problem

```
% File name: sphcont_opt.m
% Design variable: ratio of the max shear stress location to
% size of the contact patch
% Find location of the maximum shear stress along the z-axis
  clear all
% Set lower and upper bound for the design variable
  Lb = 0; Ub = 5;
% Plot normalized shear stress distribution along the z-axis in spherical contact
  z = [Lb: 0.1: Ub]';
  n = size (z);
  for i = 1: n
         outz(i) = -sphcont_objf(z(i));
  end
  plot(z, outz); grid
  xlabel ('normalized depth z/a');
  ylabel ('normalized shear stress');
% Invoke the single-variable unconstrained optimizer
  [alpha, FunVal, ExitFlag, Output] = fminbnd ('sphcont_objf', Lb, Ub)
```

TABLE 7.11 m-File for Evaluation of Objective Function for the Spherical Contact Problem

```
% File name = sphcont_objf.m
% Location of max shear stress along z-axis for spherical contact problem
  function f = sphcont_objf(alpha)
% f = - shear stress/max pressure
  nu = 0.3;  % Poisson's ratio
  f = -0.5*( (1-2*nu)/2 + (1+nu)*alpha/sqrt(1+alpha^2) - …
        1.5*( alpha/sqrt(1+alpha^2) )^3 );
```

shows the function m-file that evaluates the function to be minimized. The script m-file also contains commands to plot the shear stress as a function of z, which is shown in Fig. 7.2. The optimum solution matches the exact solution for ν = 0.3 and is given as

$$\texttt{alpha} = 0.6374,\ \texttt{FunVal} = -0.3329\ \ [\alpha^* = 0.6374,\ f(\alpha^*) = -0.3329] \qquad \text{(e)}$$

7.4.2 Column Design for Minimum Mass

Project/Problem Statement

As noted in Section 2.7, columns are used as structural members in many practical applications. Many times such members are subjected to eccentric loads, such as those applied by a jib crane. The problem is to design a minimum-mass tubular column that is subjected to an

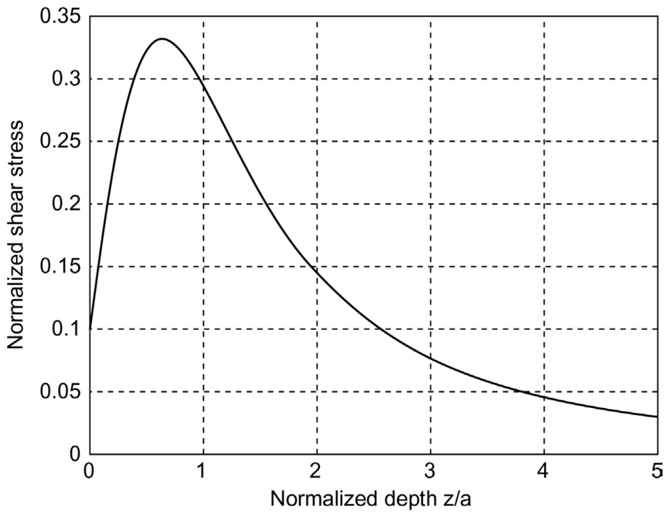

FIGURE 7.2 **Normalized shear stress along the z-axis.**

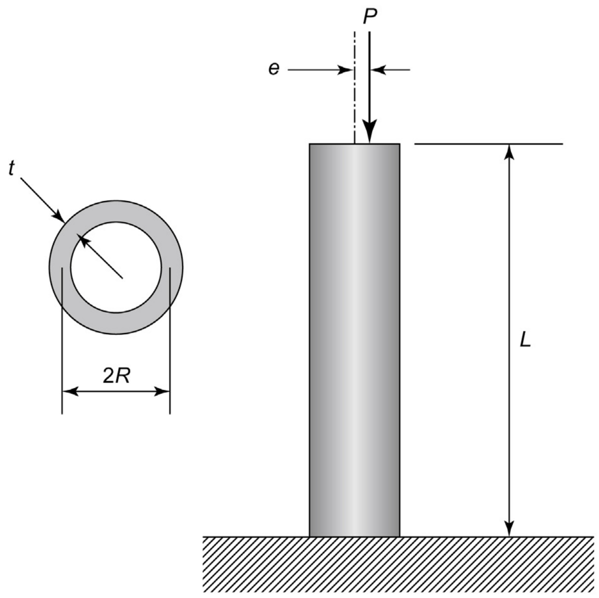

FIGURE 7.3 **Configuration of vertical column with an eccentric load.**

eccentric load, as shown in Fig. 7.3. The cross-section of the column is a hollow circular tube with R and t as the mean radius and wall thickness, respectively.

Data and Information Collection

The notation and the data for the problem are given as follows:

P	=	Load, 50 kN
L	=	Length, 5 m
R	=	Mean radius, m
E	=	Young's modulus, 210 GPa
σ_a	=	Allowable stress, 250 MPa
e	=	Eccentricity (2% of radius), $0.02R$, m
Δ	=	Allowable lateral deflection, 0.25 m
ρ	=	Mass density, 7850 kg/m^3
A	=	Cross-sectional area, $2\pi Rt$, m^2
I	=	Moment of inertia, $\pi R^3 t$, m^4
c	=	Distance to the extreme fiber, $R+\frac{1}{2}t$, m

An analysis of the structure yields the following design equations:

Normal stress:

$$\sigma = \frac{P}{A}\left[1+\frac{ec}{k^2}\sec\left(\frac{L}{k}\sqrt{\frac{P}{EA}}\right)\right],\ k^2 = \frac{I}{A} \tag{a}$$

Buckling load:

$$P_{cr} = \frac{\pi^2 EI}{4L^2} \tag{b}$$

Deflection:

$$\delta = e\left[\sec\left(\sqrt{\frac{P}{EI}}\right) - 1\right] \tag{c}$$

Definition of Design Variables

Two design variables for the problem are defined as

R, mean radius of the tube, m; t, wall thickness, m.

Optimization Criterion

The objective is to minimize the mass of the column, which is given as

$$f(\mathrm{x}) = \rho LA = (7850)(5)(2\pi Rt),\ \mathrm{kg} \tag{d}$$

Formulation of Constraints

Constraints for the problem are on performance of the structure, the maximum radius-to-thickness ratio, and the bounds on the radius and thickness:

Stress constraint:

$$\sigma \le \sigma_a \tag{e}$$

Buckling load constraint:

$$P \le P_{cr} \tag{f}$$

Deflection constraint:

$$\delta \le \Delta \tag{g}$$

Radius/thickness constraint, $\frac{R}{t} \le 50$:

$$R \le 50t \tag{h}$$

Bounds on variables:

$$0.01 \le R \le 1,\ 0.005 \le t \le 0.2,\ \mathrm{m} \tag{i}$$

Solution

Let us redefine the design variables and other parameters for MATLAB as

$$x_1 = R,\ x_2 = t \tag{j}$$

$$c = x_1 + \frac{1}{2}x_2,\ e = 0.02x_1 \tag{k}$$

$$A = 2\pi x_1 x_2,\ I = \pi x_1^3 x_2,\ k^2 = \frac{I}{A} = \frac{x_1^2}{2} \tag{l}$$

All constraints are normalized and rewritten using these redefined design variables. Therefore, the optimization problem is stated in the standard form as follows:

Minimize

$$f(\mathrm{x}) = 2\pi(5)(7850)x_1 x_2 \tag{m}$$

subject to

$$g_1(\mathrm{x}) = \frac{P}{2\pi x_1 x_2 \sigma_a}\left[1 + \frac{2\times 0.02(x_1 + 0.5x_2)}{x_1}\sec\left(\frac{\sqrt{2}L}{x_1}\sqrt{\frac{P}{E(2\pi x_1 x_2)}}\right)\right] - 1 \le 0 \tag{n}$$

$$g_2(\mathrm{x}) = 1 - \frac{\pi^2 E(\pi x_1^3 x_2)}{4L^2 P} \le 0 \tag{o}$$

$$g_3(\mathrm{x}) = \frac{0.02x_1}{\Delta}\left[\sec\left(L\sqrt{\frac{P}{E(\pi x_1^3 x_2)}}\right) - 1\right] - 1 \le 0 \tag{p}$$

$$g_4(\mathrm{x}) = x_1 - 50x_2 \le 0 \tag{q}$$

$$0.01 \le x_1 \le 1,\ 0.005 \le x_2 \le 0.2 \tag{r}$$

Note that the constraints in Eqs. (n) to (p) have been scaled with respect to their limit values. The problem is solved using the `fmincon` function in the Optimization Toolbox. Table 7.12 shows the script m-file for invoking this function and setting various options for the optimization process. Tables 7.13 and 7.14 show the function m-files for the objective and constraint functions, respectively. Note that analytical gradients are not provided for the problem functions.

The output from the function is given as

`Active constraints`: 2, 5, that is, the lower limit for thickness and `g(1)`.
x = (0.0537,0.0050), `FunVal = 66.1922`, `ExitFlag = 1`, `Output` = (*iterations:* 31, `funcCount`: 149, stepsize: 1, `algorithm`: medium-scale: SQP, Quasi-Newton, line-search).

7.4.3 Flywheel Design for Minimum Mass

Project/Problem Statement

Shafts are used in practical applications to transfer torque from a source point to another point. However, the torque to be transferred can fluctuate, causing variations in the angular

TABLE 7.12 m-File for Invoking Minimization Function for Column Design Problem

```
% File name = column_opt.m
  clear all
% Set options
  options = optimset ('LargeScale', 'off', 'TolCon', 1e-8, 'TolX', 1e-8);
% Set the lower and upper bounds for design variables
  Lb = [0.01 0.005]; Ub = [1 0.2];
% Set initial design
  x0 = [1 0.2];
% Invoke the constrained optimization routine, fmincon
  [x, FunVal, ExitFlag, Output] = ...
         fmincon('column_objf', x0, [], [], [], [], Lb, Ub, 'column_conf',
                 options)
```

TABLE 7.13 m-File for Objective Function for Minimum Mass Column Design Problem

```
% File name = column_objf.m
% Column design
  function f = column_objf (x)
% Rename design variables
  x1 = x(1); x2 = x(2);
% Set input parameters
  L = 5.0; % length of column (m)
  rho = 7850; % density (kg/m^3)
         f = 2*pi*L*rho*x1*x2; % mass of the column
```

speed of the shaft, which is not desirable. Flywheels are used on the shaft to smooth out these speed variations (Norton, 2000; Budynas and Nisbett, 2014). The purpose of this project is to design a flywheel to smooth out variations in the speed of a solid shaft of radius r_i. The flywheel-shaft system is shown in Fig. 7.4. The input torque function, which varies during a cycle, is shown in Fig. 7.5. The torque variation about its average value is shown there as a function of the shaft angle from 0 to 360 degrees. The kinetic energy due to this variation is obtained by integrating the torque pulse above and below its average value (shaded area in Fig. 7.5) during the cycle and is given as $E_k = 26{,}105$ in lb. The shaft is rotating at a nominal angular speed of $\omega = 800$ rad/s.

TABLE 7.14 m-File for Constraint Functions for the Column Design Problem

```
% File name = column_conf.m
  % Column design
function [g, h] = column_conf (x)
  x1 = x(1); x2 = x(2);
% Set input parameters
  P = 50000; % loading (N)
  E = 210e9; % Young's modulus (Pa)
  L = 5.0; % length of the column (m)
  Sy = 250e6; % allowable stress (Pa)
  Delta = 0.25; % allowable deflection (m)
% Inequality constraints
  g(1) = P/(2*pi*x1*x2)*(1 + …
      2*0.02*(x1+x2/2)/x1*sec( 5*sqrt(2)/x1*sqrt(P/E/(2*pi*x1*x2)) ) )/Sy - 1;
  g(2) = 1 - pi^3*E*x1^3*x2/4/L^2/P;
  g(3) = 0.02*x1*( sec( L*sqrt( P/(pi*E*x1^3*x2) ) ) - 1 )/Delta - 1;
  g(4) = x1-50*x2;
% Equality constraint (none)
  h = [];
```

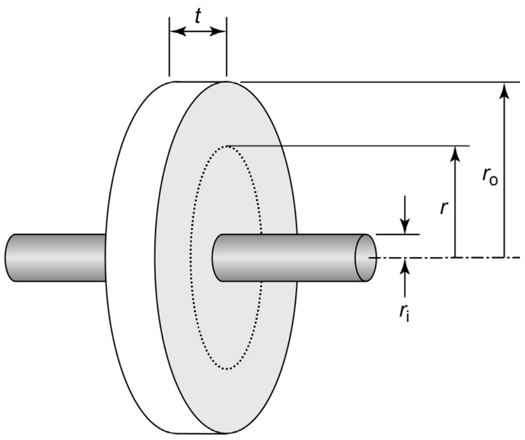

FIGURE 7.4 **Flywheel-shaft system.**

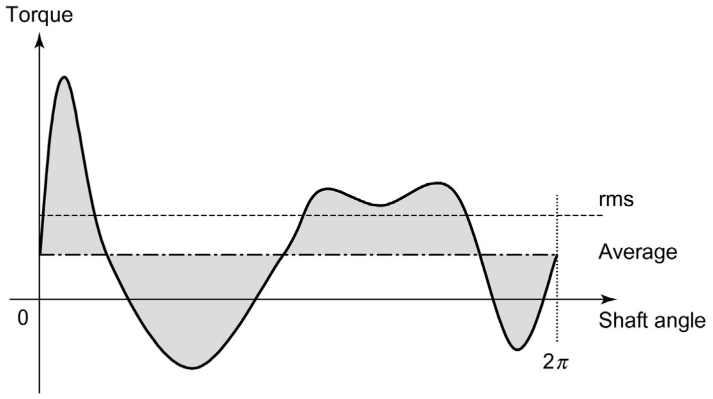

FIGURE 7.5 **Fluctuation about the average value of input torque for one cycle.**

Data and Information Collection

One cycle of torque variation shown in Fig. 7.5 is assumed to be repetitive and, thus, representative of the steady-state condition. The desired coefficient of fluctuation is assumed to be 0.05 (C_f). The coefficient of fluctuation represents the ratio of variation of angular velocity to the nominal angular velocity: $C_f = (\omega_{max} - \omega_{min})/\omega$. The system is assumed to be in continuous operation with minimal start–stop cycles. The minimum-mass moment of inertia for the flywheel is determined using the required change in kinetic energy, E_k, specified earlier, as

$$I_s = \frac{E_k}{C_f\omega^2} = \frac{26{,}105}{0.05(800)^2} = 0.816 \text{ lb in. s}^2 \tag{a}$$

The design data and equations needed to formulate the minimum-mass flywheel problem are given as

γ = Specific weight, 0.28 lb/in.^{3}
G = Gravitational constant, 386 in/s^2
S_y = Yield stress, 62,000 psi
ω = Nominal angular velocity, 800 rad/s
ν = Poisson's ratio, 0.28
r_i = Inner radius of flywheel, 1.0 in.
r_o = Outer radius of flywheel, in.
t = Thickness of flywheel, in.

Some useful expressions for the flywheel are

Mass moment of inertia of flywheel:

$$I_m = \frac{\pi}{2}\frac{\gamma}{g}(r_o^4 - r_i^4)t, \text{ lb in. s}^2 \tag{b}$$

Tangential stress in flywheel at radius r:

$$\sigma_t = \frac{\gamma}{g}\omega^2 \frac{3+\nu}{8}\left(r_i^2 + r_o^2 + \frac{r_i^2 r_o^2}{r^2} - \frac{1+3\nu}{3+\nu}r^2\right), \text{psi} \tag{c}$$

Radial stress in flywheel at radius r:

$$\sigma_t = \frac{\gamma}{g}\omega^2 \frac{3+\nu}{8}\left(r_i^2 + r_o^2 + \frac{r_i^2 r_o^2}{r^2} - r^2\right), \text{psi} \tag{d}$$

von Mises stress:

$$\sigma' = \sqrt{\sigma_r^2 - \sigma_r\sigma_t + \sigma_t^2}\text{, psi} \tag{e}$$

Definition of Design Variables

The two design variables for the problem are defined as

r_o, outer radius of the flywheel, in.; t, thickness of the flywheel, in.

Optimization Criterion

The objective of the project is to design a flywheel of minimum mass. Since mass is proportional to material volume, we want to minimize the volume of the flywheel, which is given as

$$f = \pi(r_o^2 - r_i^2)t,\ \text{in.}^3 \tag{f}$$

Formulation of Constraints

Performance and other constraints are expressed as

Mass moment of inertia requirement:

$$I_m \geq I_s \tag{g}$$

von Mises stress constraint:

$$\sigma' \leq \frac{1}{2}S_y \tag{h}$$

Limits on design variables:

$$4.5 \leq r_o \leq 9.0,\ \ 0.25 \leq t \leq 1.25\text{, in.} \tag{i}$$

Solution

The problem is solved using the `fmincon` function in the Optimization Toolbox. Table 7.15 shows the script m-file that invokes the `fmincon` function for the flywheel problem. Table 7.16 shows the function m-file that calculates the problem's objective function. Table 7.17 shows the function m-file for calculation of the constraints. Note that the von Mises stress constraint is imposed at the point of maximum stress. Therefore, this maximum is calculated using the

TABLE 7.15 m-File to Invoke Constrained Minimization Routine for Flywheel Design Problem

% File name = flywheel_opt.m

% Flywheel design

% Design variables: outside radius (ro), and thickness (t)

```
clear all
```

% Set options

```
options = optimset ('LargeScale', 'off');
```

% Set limits for design variables

```
Lb = [4.5, 0.25]; % lower limit
Ub = [9, 1.25]; % upper limit
```

% Set initial design

```
x0 = [6, 1.0];
```

% Set radius of shaft

```
ri = 1.0;
[x, FunVal, ExitFlag, Output] = ...
       fmincon('flywheel_objf', x0, [], [], [], [], Lb, Ub, 'flywheel_conf',
options, ri)
```

TABLE 7.16 m-File for Objective Function for Flywheel Design Problem

% File name = flywheel_objf.m

% Objective function for flywheel design problem

```
function f = flywheel_objf(x, ri)
```

% Rename the design variables x

```
ro = x(1);
t = x(2);
f = pi*(ro^2 - ri^2)*t; % volume of flywheel
```

`fminbnd` function. Table 7.18 shows the function m-file that calculates the von Mises stress. This file is called by the constraint evaluation function. Also note that all constraints are entered in the normalized "≤" form. The solution and other output from the function are given as follows:

Active constraints are 2 and 5 [ie, lower bound on thickness and `g(1)`]

$r_o^* = 7.3165$ in, $t^* = 0.25$ in, $f^* = 41.2579$ in^3, Output = [iterations: 8, funcCount: 37]

TABLE 7.17 m-File for Constraint Functions for Flywheel Design Problem

% Constraint functions for flywheel design problem

```
function [g, h] = flywheel_conf(x, ri)
```

% Rename design variables x

```
ro = x(1);
t = x(2);
```

% Constraint limits

```
Is = 0.816;
Sy = 62000; % yield strength
```

% Normalized inequality constraints

```
g(1) = 1 - pi/2*(0.28/386)*(ro^4 - ri^4)*t/Is;
```

% Evaluate maximum von Mises stress

```
options = [];
[alpha, vonMS] = fminbnd('flywheel_vonMs', ri, ro, options, ri, ro);
g(2) = -vonMS/(0.5*Sy) - 1;
```

% Equality constraint (none)

```
h = [];
```

TABLE 7.18 m-File for Computation of Maximum von Mises Stress at Current Design

% File name = flywheel_vonMS.m

% von Mises stress

```
function vonMS = flywheel_vonMS (x, ri, ro)
temp = (0.28/386)*(800)^2*(3+0.28)/8;
```

% Tangential stress

```
st = temp*(ri^2 + ro^2 + ri^2*ro^2/x^2 - (1+3*0.28)/(3+0.28)*x^2 );
```

% Radial stress

```
sr = temp*(ri^2 + ro^2 - ri^2*ro^2/x^2 - x^2); % radial stress
vonMS = -sqrt(st^2 - st*sr + sr^2); % von Mises stress
```

EXERCISES FOR CHAPTER 7*

Formulate and solve the following problems.

7.1 Exercise 3.34
7.2 Exercise 3.35
7.3 Exercise 3.36
7.4 Exercise 3.50
7.5 Exercise 3.51
7.6 Exercise 3.52
7.7 Exercise 3.53
7.8 Exercise 3.54
7.9 Consider the cantilever beam-mass system shown in Fig. E7.9. Formulate and solve the minimum weight design problem for the rectangular cross section so that the fundamental vibration frequency is larger than 8 rad/s and the cross-sectional dimensions satisfy the limitations

$$\begin{aligned} 0.5 \le b \le 1.0, \text{ in.} \\ 0.2 \le h \le 2.0, \text{ in.} \end{aligned} \tag{a}$$

Use a nonlinear programming algorithm to solve the problem. Verify the solution graphically and trace the history of the iterative process on the graph of the problem. Let the starting point be (0.5,0.2). The data and various equations for the problem are as shown in the following.
Cantilever beam with spring-mass at the free end.
Fundamental vibration frequency

$$\omega = \sqrt{k_e/m}, \text{ rad/s} \tag{b}$$

Equivalent spring constant k_e

$$\frac{1}{k_e} = \frac{1}{k} + \frac{L^3}{3EI} \tag{c}$$

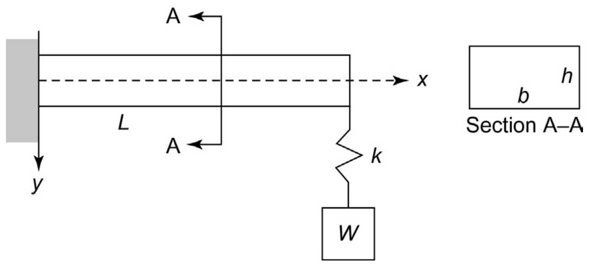

FIGURE E7.9 **Cantilever beam with spring-mass at the free end.**

Mass attached to the spring

$$m = W/g \tag{d}$$

Weight attached to the spring

$$W = 50 \text{ lb} \tag{e}$$

Length of the beam

$$L = 12 \text{ in.} \tag{f}$$

Modulus of elasticity

$$E = (3 \times 10^7)\,\text{psi} \tag{g}$$

Spring constant

$$k = 10 \text{ lb/in.} \tag{h}$$

Moment of inertia

$$I,\ \text{in.}^4 \tag{i}$$

Gravitational constant

$$g,\ \text{in./s}^2 \tag{j}$$

7.10 A prismatic steel beam with symmetric I cross-section is shown in Fig. E7.10. Formulate and solve the minimum weight design problem subject to the following constraints:

1. The maximum axial stress due to combined bending and axial load effects should not exceed 100 MPa.
2. The maximum shear stress should not exceed 60 MPa.
3. The maximum deflection should not exceed 15 mm.
4. The beam should be guarded against lateral buckling.
5. Design variables should satisfy the limitations $b \geq 100$ mm, $t_1 \leq 10$ mm, $t_2 \leq 15$ mm, $h \leq 150$ mm.

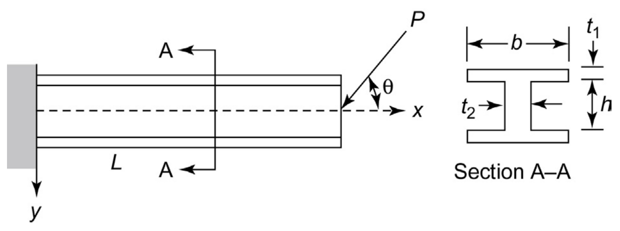

FIGURE E7.10 **Cantilever I beam.** Design variables b, t_1, t_2, and h.

Solve the problem using a numerical optimization method, and verify the solution using KKT necessary conditions for the following data:

Modulus of elasticity, $E = 200$ GPa
Shear modulus, $G = 70$ GPa
Load, $P = 70$ kN
Load angle, $\theta = 45$ degree
Beam length, $L = 1.5$ m

7.11 *Shape optimization of a structure.* The design objective is to determine the shape of the three-bar structure shown in Fig. E7.11 to minimize its weight (Corcoran, 1970). The design variables for the problem are the member cross-sectional areas A_1, A_2, and A_3 and the coordinates of nodes A, B, and C (note that x_1, x_2, and x_3 have positive values in the figure; the final values can be positive or negative), so that the truss is as light as possible while satisfying the stress constraints due to the following three loading conditions:

Condition no. j	Load P_j (lb)	Angle θ_j (degrees)
1	40,000	45
2	30,000	90
3	20,000	135

The stress constraints are written as

$$\begin{aligned} -5000 &\le \sigma_{1j} \le 5000, \text{ psi} \\ -20{,}000 &\le \sigma_{2j} \le 20{,}000, \text{ psi} \\ -5000 &\le \sigma_{3j} \le 5000, \text{ psi} \end{aligned}$$

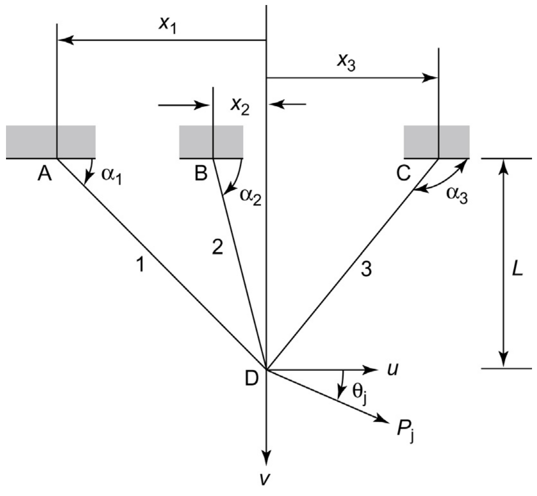

FIGURE E7.11 **A three-bar structure–shape optimization.**

where ***j*** = 1, 2, 3 represents the index for the three loading conditions and the stresses are calculated from the following expressions:

$$
\begin{aligned}
\sigma_{1j} &= \frac{E}{L_1}[u_j \cos\alpha_1 + v_j \sin\alpha_1] = \frac{E}{L_1^2}(u_j x_1 + v_j L) \\
\sigma_{2j} &= \frac{E}{L_2}[u_j \cos\alpha_2 + v_j \sin\alpha_2] = \frac{E}{L_2^2}(u_j x_2 + v_j L) \\
\sigma_{3j} &= \frac{E}{L_3}[u_j \cos\alpha_3 + v_j \sin\alpha_3] = \frac{E}{L_3^2}(-u_j x_3 + v_j L)
\end{aligned}
\tag{a}
$$

where L = 10 in and

$$
\begin{aligned}
L_1 &= \text{length of member } 1 = \sqrt{L^2 + x_1^2} \\
L_2 &= \text{length of member } 2 = \sqrt{L^2 + x_2^2} \\
L_3 &= \text{length of member } 3 = \sqrt{L^2 + x_3^2}
\end{aligned}
\tag{b}
$$

and $\boldsymbol{u_j}$ and $\boldsymbol{v_j}$ are the horizontal and vertical displacements for the jth loading condition determined from the following linear equations:

$$
\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix}\begin{bmatrix} u_j \\ v_j \end{bmatrix} = \begin{bmatrix} p_j \cos\theta_j \\ p_j \sin\theta_j \end{bmatrix}, \; j = 1,2,3 \tag{c}
$$

where the stiffness coefficients are given as ($\boldsymbol{E}$ = 3.0E + 07psi)

$$
\begin{aligned}
k_{11} &= E\left(\frac{A_1 x_1^2}{L_1^3} + \frac{A_2 x_2^2}{L_2^3} + \frac{A_3 x_3^2}{L_3^3}\right) \\
k_{12} &= E\left(\frac{A_1 L x_1}{L_1^3} + \frac{A_2 L x_2}{L_2^3} + \frac{A_3 L x_3}{L_3^3}\right) = k_{21} \\
k_{22} &= E\left(\frac{A_1 L^2}{L_1^3} + \frac{A_2 L^2}{L_2^3} + \frac{A_3 L^3}{L_3^3}\right)
\end{aligned}
\tag{d}
$$

Formulate the design problem and find the optimum solution starting from the point

$$
\begin{aligned}
&A_1 = 6.0,\; A_2 = 6.0,\; A_3 = 6.0 \\
&x_1 = 5.0,\; x_2 = 0.0,\; x_3 = 5.0
\end{aligned}
\tag{e}
$$

Compare the solution with that given later in Table 14.7.

7.12 *Design synthesis of a nine-speed gear drive*. The arrangement of a nine-speed gear train is shown in Fig. E7.12. The objective of the synthesis is to find the size of all gears from the mesh and speed ratio equations such that the sizes of the largest gears are kept to a minimum (Osman et al., 1978). Because of the mesh and speed ratio

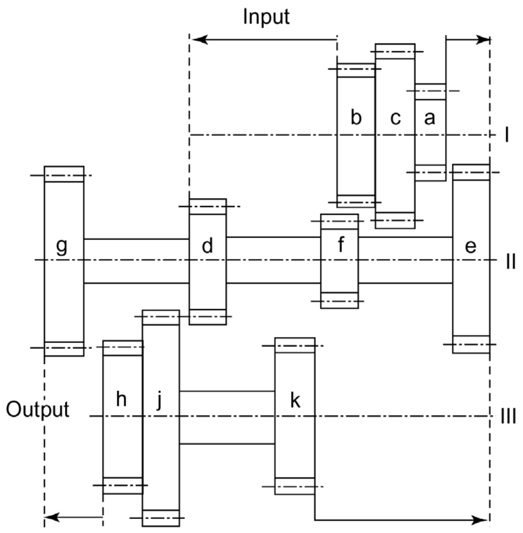

FIGURE E7.12 **Schematic arrangement of a nine-speed gear train.**

equations, it is found that only the following three independent parameters need to be selected:

$$x_1 = \text{gear ratio, } d/a$$
$$x_2 = \text{gear ratio, } e/a$$
$$x_3 = \text{gear ratio, } j/a$$

Because of practical considerations, it is found that the minimization of $|x_2 - x_3|$ results in the reduction of the cost of manufacturing the gear drive.
The gear sizes must satisfy the following mesh equations:

$$\phi^2 x_1(x_1 + x_3 - x_2) - x_2 x_3 = 0$$
$$\phi^3 x_1 - x_2(1 + x_2 - x_1) = 0 \tag{a}$$

where ϕ is the step ratio in speed. Find the optimum solution for the problem for two different values of ϕ as $\sqrt{2}$ and $(2)^{1/3}$.

References

Budynas, R., Nisbett, K., 2014. Shigley's Mechanical engineering design, tenth ed. McGraw- Hill, New York.
Corcoran, P.J., 1970. Configuration optimization of structures. Int. J. Mech. Sci. 12, 459–462.
Norton, R.L., 2000. Machine Design: An Integrated Approach, second ed. Prentice-Hall, Upper Saddle River, NJ.

Osman, M.O.M., Sankar, S., Dukkipati, R.V., 1978. Design synthesis of a multi-speed machine tool gear transmission using multiparameter optimization. ASME J. Mech. Des. 100, 303–310.

Schittkowski, K., 1981. The nonlinear programming method of Wilson, Han and Powell with an augmented Lagrangian type line search function, Part 1: Convergence analysis, Part 2: An efficient implementation with linear least squares subproblems. Numerische Mathematik 38, 83–127.

Schittkowski, K., 1987. More Test Examples for Nonlinear Programming Codes. Springer-Verlag, New York.