Optimum Design of Systems for Dynamics and Controls Using Sequential Quadratic Programming

C. H. Tseng* and J. S. Arora†
The University of Iowa, Iowa City, Iowa

This paper describes application of a sequential quadratic programming method for dynamic response optimization and optimal control problems. Several methods to treat time-dependent constraints are described. Numerical procedures to solve this class of problems are studied and promising ones are implemented. They are evaluated by solving numerous problems. Numerical experience with the solution procedures is described and conclusions that can be useful for practical applications are given.

Abbreviations

ALL = indicates conventional treatment of dynamic constraints
AVM = adjoint variable method
Con. para. = convergence parameter
COST = value of the cost function
CPU = central processor units in seconds
CPU* = central processor in seconds for solution of differential equations
CTRLCF = control routine for constraint function evaluation
CTRLCG = control routine for constraint function gradient evaluation
CTRLMF = control routine for merit function evaluation
CTRLMG = control routine for merit function gradient evaluation
DDM = direct differentiation method
DDEABM = subroutine based on Adam-Bashforth method
DDEBDF = subroutine based on backward-differentiation formula
DDERKF = subroutine based on fifth-order Runge-Kutta method
DEPAC = differential equation solver package
DSA = design sensitivity analysis
GAUSS = Gaussian quadrature rule for integral evaluation
HYB = hybrid treatment of dynamic constraints
IDDIFF = index for differential equation solver
IDESIGN = interactive design optimization package
IDINTG = integral evaluation indicator
IDINTU = index for interpolation scheme used for the control function
IDMETH = index for design sensitivity analysis method
IDPTWC = index for treatment of pointwise dynamic constraints
MAX = worst-case treatment of dynamic constraints
Max. vio. = maximum constraint violation
NAC = number of potential constraints
NCF = number of function evaluations
NGPT = number of grid points
NGPTU = number of grid points for the control function
NIT = number of iterations
NLP = nonlinear programming
NR = test run number
NTG = total number of gradient evaluations
NV = number of design variables
OCP = optimal control problem
PI = performance index
SIMPSN = Simpson's method for integral evaluation
SQP = sequential quadratic programming
USERCF = user-supplied constraint functions evaluation routine
USERCG = user-supplied constraint function gradient evaluation routine
USERMF = user-supplied merit function evaluation routine
USERMG = user-supplied merit function gradient evaluation routine

I. Introduction

In recent years, considerable interest has been shown in optimizing the dynamics and control of mechanical and structural systems. These include linear/nonlinear and time-invariant/variant-feedback control systems as well as vibration isolation and minimum time control problems. Considerable attention has also been paid to the extension of mathematical programming methods to solve engineering design optimization problems.[1] As a result, a powerful iterative optimization method called sequential quadratic programming (SQP) has been developed.[2-6] The SQP method with potential constraint strategy that generates second-order information about the Lagrange function for the problem has been adapted to solve optimization problems. This paper discusses application of the method to dynamic-response optimization problems.

A general formulation that encompasses dynamic response as well as control problems is given. Evaluation of the dynamic response and the constraints can require enormous computational effort. Also, design sensitivity analysis of dynamic constraints must be performed. Significant progress has been recently made in developing methods for treatment of dynamic constraints and their design sensitivity analysis.[7] However, both the mathematical programming methods and the design sensitivity analysis procedures need to be evaluated for solving general dynamic response optimization problems. Therefore, major objectives of the present paper are to study the optimal design formulation for dynamics and controls, identify numerical procedures needed to solve the problem, and evaluate some of them through implementation.

Received March 22, 1988; presented as Paper 88-2303 at the 29th Structures, Structural Dynamics and Materials Conference, Williamsburg, VA, April 18-22, 1988; revision received Oct. 27, 1988.

*Graduate Research Assistant, College of Engineering; currently Associate Professor, Mechanical Engineering, National Chiao Tung University, Taiwan.

†Professor, Civil and Environmental Engineering, Mechanical Engineering, College of Engineering. Member AIAA.

An effective solution method for realistic dynamic-response problems is to transform them into nonlinear programming (NLP) problems by introducing a discretized control model.[8-13] Two possible discretization models—admissible and discrete optimal control—are briefly discussed. The software package for solution of these problems should be flexible and have options for different numerical methods; such as, for integration of differential equations, interpolation, integral evaluation during analysis and design sensitivity analysis, treatment of dynamic constraints, design sensitivity analysis, etc. Such a software environment is described. The criteria to switch between the procedures automatically need to be developed. The procedures need to be evaluated in order to improve accuracy of the final solution and efficiency of the optimal design process.

II. Problem Statement

A general optimal control problem (OCP) that encompasses both dynamic response as well as control applications is defined as follows: Find a k-dimensional design variable vector $\boldsymbol{b}$, the m-dimensional control function $\boldsymbol{u}(t)$, and the terminal time t_f which minimize the performance index (PI):

$$\psi_0 = g_0[\boldsymbol{b},z(t_f),t_f] + \int_{t_0}^{t_f} F_0[\boldsymbol{b},\boldsymbol{u}(t),z(t),t]\,dt \tag{1}$$

subject to the state (or dynamic) equation

$$\dot{z}(t) = \boldsymbol{f}[\boldsymbol{b},\boldsymbol{u}(t),z(t),t], \qquad t_0 \le t \le t_f \tag{2}$$

with the initial conditions $z(t_0) = \boldsymbol{h}(\boldsymbol{b})$, the functional constraints

$$\psi_\alpha = g_\alpha[\boldsymbol{b},z(t_f),t_f] + \int_{t_0}^{t_f} F_\alpha[\boldsymbol{b},\boldsymbol{u}(t),z(t),t]\,dt \quad \begin{cases} = 0; & \alpha = 1,...,r' \\ \le 0; & \alpha = r'+1,...,r \end{cases} \tag{3}$$

and dynamic pointwise constraints for $t_0 \le t \le t_f$

$$\phi_\beta[\boldsymbol{b},\boldsymbol{u}(t),z(t),t] \quad \begin{cases} = 0; & \beta = 1,...,q' \\ \le 0; & \beta = q'+1,...,q \end{cases} \tag{4}$$

The state variable $z(t) \in R^n$ is assumed to be a continuous function of time, and $\boldsymbol{u}(t)$ to be bounded and measurable on the interval $[t_0,t_f]$.

The preceding definition extends the original Bolza problem to account for inequality constraints. The original formulation containing only equality constraints is not general for optimal control problems. It also does not treat design variables $\boldsymbol{b}$ which may serve a variety of useful purposes apart from obvious design parameters, e.g., stiffness, length, area. The differential equations for the system in Eq. (2) are written in the first-order form which can be nonlinear, even though $z(t)$ appears linearly. It is understood that the most efficient procedure, such as modal analysis for linear systems, shall be used in integrating the state equations. Advantages of such procedures can also be realized during design sensitivity analysis.[14] To maintain generality, however, we shall treat the first-order form of the state equations. Equation (4) represent constraints on the state and control variables. The functions g_0, F_0, g_α, $\boldsymbol{f}$, F_α, and ϕ_β are assumed to be at least twice differentiable.

It is important to note that the preceding formulation can treat min-max response,[14] minimum time optimal control, minimum control effort, and precision (minimum error in response) design optimization problems. It can also treat linear and nonlinear systems, and equality and inequality constraints on dynamic response at intermediate as well as terminal time points.

III. Transformation to the NLP Problem

The necessary and sufficient conditions for the problem using the maximum principle are derived by Bryson and Ho.[15] This approach leads to the solution of a boundary-value problem. A basic hypothesis of this paper is that for most practical problems, the foregoing approach is not viable. A better and more general approach is to use iterative methods of NLP where equations of motion can be treated as initial-value problems. To treat the continuum model as a discrete model, we must represent the infinite dimensional control function $\boldsymbol{u}(t)$ and state variable $z(t)$ by a finite set of parameters. Discretization of the control function is considered first. The entire time interval $[t_0,t_f]$ is subdivided into N equal intervals, $\Delta t = (t_f - t_0)/N$, and the grid is designated as $t_0,t_1,t_2,...,t_{N-1},t_N(t_f)$ which generate the parameter set

$$\boldsymbol{U} = [u_1(t_0),...,u_1(t),u_2(t_0),...,u_2(t),...,u_m(t)]^T = [U_1,...,U_{N+1},U_{N+2},...,U_{2N+2},...,U_{mN+m}]^T \tag{5}$$

This is also treated as a design variable vector resulting in a total of $(k + mN + m)$ design variables as

$$\boldsymbol{x} = [b_1,...,b_k,U_1,...,U_{N+1},U_{N+2},...,U_{2N+2},...,U_{mN+m}]^T \tag{6}$$

The coefficients of an interpolating function for the control function $\boldsymbol{u}(t)$ may be considered as design variables instead of U_i in Eq. (6). For example, the interpolation function based on a third-order polynomial $\boldsymbol{u}_1(t) = U_1 + U_2 t + U_3 t^2 + U_4 t^3$ can be used to represent the first component of the control forces in $\boldsymbol{u}(t)$; and U_1, U_2, U_3, and U_4 can be treated as design variables. Also, in some control problems, the number of the initial conditions is not enough for integration, or the initial conditions cannot be separated from the functional constraints in Eq. (3). Artificial variables can be added as design variables to treat the problem which expands the vector $\boldsymbol{x}$. Two discretization techniques for the state and control variables can be used to reformulate the problem.[9]

Discrete-Time Optimal Control Problem

The state variable $z(t)$ is discretized using a grid as $z(t_j)$ for $j = 1,...,N$ and the control variable is discretized as in Eq. (5). The state equations in Eq. (2) are rewritten in the finite-difference form and the integrals in Eqs. (1) and (3) are written as finite sums. The design variables are defined as in Eq. (6) and the time-dependent constraints in Eq. (4) are imposed at each time grid point. This type of formulation is rarely used because good programs to integrate state equations are available, and the time-dependent constraints are treated in one of the several ways discussed in the sequel.

Admissible Optimal Control Problem

The approximate trajectory $z(t)$ is generated by solving the initial-value problem defined in Eq. (2). Interpolation of the state variable $z(t)$ may be done internally in the equation solver, if a variable-step-size algorithm is used. Also, in each subinterval $[t_i,t_{i+1}]$, $i = 0,...,(N-1)$, the control forces $\boldsymbol{u}(t)$ are approximated by some interpolating functions $\boldsymbol{I}(t)$. (Note that the number of grid points for the state and control variables are usually different.) The control function is represented in the form $\boldsymbol{u}(t) = \boldsymbol{I}(\boldsymbol{U},t)$ and the state variable is written in the form $z(\boldsymbol{b},\boldsymbol{U},t)$. This is called the admissible optimal control problem which is stated in NLP form as follows: Find $\boldsymbol{x}$ to minimize the PI

$$\psi_0 = g_0[\boldsymbol{b},z(\boldsymbol{b},\boldsymbol{U},t_f),t_f] + \int_{t_0}^{t_f} F_0[\boldsymbol{b},\boldsymbol{I}(\boldsymbol{U},t),z(\boldsymbol{b},\boldsymbol{U},t),t]\,dt \tag{7}$$

subject to state equations

$$\dot{z}(t) = \boldsymbol{f}[\boldsymbol{b},\boldsymbol{I}(\boldsymbol{U},t),z(\boldsymbol{b},\boldsymbol{U},t),t], \qquad t_0 \le t \le t_f \tag{8}$$

with initial conditions $z(t_0) = \boldsymbol{h}(\boldsymbol{b})$, the functional constraints as

$$\psi_\alpha = g_\alpha[\boldsymbol{b},\boldsymbol{z}(\boldsymbol{b},\boldsymbol{U},t_f),t_f] + \int_{t_0}^{t_f} F_\alpha[\boldsymbol{b},\boldsymbol{I}(\boldsymbol{U},t), \boldsymbol{z}(\boldsymbol{b},\boldsymbol{U},t),t]\,dt \begin{cases} = 0; & \alpha = 1,...,r' \\ \le 0; & \alpha = r'+1,...,r \end{cases} \tag{9}$$

and dynamic constraints as

$$\phi_\beta[\boldsymbol{b},\boldsymbol{I}(\boldsymbol{U},t),\boldsymbol{z}(\boldsymbol{b},\boldsymbol{U},t),t] \begin{cases} = 0; & \beta = 1,...,q' \\ \le 0; & \beta = q'+1,...,q \end{cases} \tag{10}$$

Numerical Considerations

The major difference between discrete and admissible control problems is in the discretization technique for the state equations. A uniform grid is used to solve the finite-difference state equation to obtain the state trajectories. The state and control variables are evaluated at time grid points only, and they are not needed between the grid points. Therefore, interpolation schemes for the control function are also not necessary. This implementation of the discretized control problem is quite straightforward. The accuracy of the final solution is highly dependent on the time interval. If a small time interval is used, the computational effort increases which limits the application of the method to small problems.

Some good first-order differential equation solvers having step size and user-desired error control are available to integrate the state equations as well as the sensitivity equations. These can be used in the admissible optimal control formulation resulting in more accurate solution than that with the discrete-time optimal control formulation. Therefore, the present study concentrates only on the admissible optimal control formulation.

IV. Numerical Methods for Solving the NLP Problem

Dynamic-Constraint Treatment

The continuum dynamic constraints in Eq. (4) must be satisfied over the entire time interval. Some procedure to eliminate time from these constraints needs to be used. There are five possible treatments, as shown in Fig. 1:

1) Conventional formulation: The time interval $[t_0,t_f]$ is divided into N subintervals and the dynamic constraints of Eq. (10) are imposed at all of the grid points.

2) Equivalent functional formulation: Each dynamic constraint is transformed into an equivalent functional form by integrating them over the entire time interval.[14]

3) Worst-case design formulation: Each continuum constraint of Eq. (10) is replaced by several constraints imposed at the worst-response time points.[7] The numerical procedure is to locate all of the local maximum points for the constraint at a given design and impose the constraint here.

4) Subdomain functional formulation: Each dynamic constraint is transformed into several equivalent functional constraints by dividing the entire time domain into several subdomains each containing one local maximum point.[7]

5) Grid points adjacent to the maximum point: The last treatment is to impose constraints at the two grid points that bracket the local maximum point.

Conventional treatment can lead to a large number of constraints for the NLP problem. If a potential constraint strategy is not used in the optimization algorithm, the NLP problem may require excessive computer memory and time. Also, the constraint may be violated between the grid points especially when the time interval is large. Therefore, the NLP problem should be formulated and solved with a small time interval. With the equivalent formulation, the optimality conditions are different from those with the original constraints.[7] The formulation causes difficulties limiting the precision with which convergence can be obtained in numerical calculations. In the worst-case treatment, much computational effort is expended in accurately locating the local maximum points. Also, the number of maximum points for a dynamic constraint can change with design changes and it becomes tedious to keep track of all the information needed to impose constraints. Subdomain functional treatment combines ideas of the equivalent functional and worst-case formulations. This kind of an approach can avoid accurate location of the maximum points. It can result in a more efficient procedure compared to the worst-case formulation. The main advantage of grid points adjacent to maximum-point formulation is that it avoids the disadvantages of the functional, worst-case, and subdomain functional treatments. Also, it can be treated as a special case of the conventional formulation where only some severely violated constraints are imposed.

Design Sensitivity Analysis

In numerical methods of optimization, one must perform design sensitivity analysis (DSA); i.e., calculate design gradients of problem functions. In general, there are two ways for computing these gradients. A straightforward procedure is to use a finite-difference approximation. The other way is to differentiate implicit functions analytically. Two methods for calculating analytical derivatives of a constraint with respect to the design variables can be used. These are the direct differentiation method (DDM) and the adjoint variable method (AVM).[7] Design gradients of the functionals in Eqs. (7) and (9) can be routinely calculated as explained in Ref. 14. The dynamic constraints in Eq. (10) can be treated by several approaches that are described in Ref. 7.

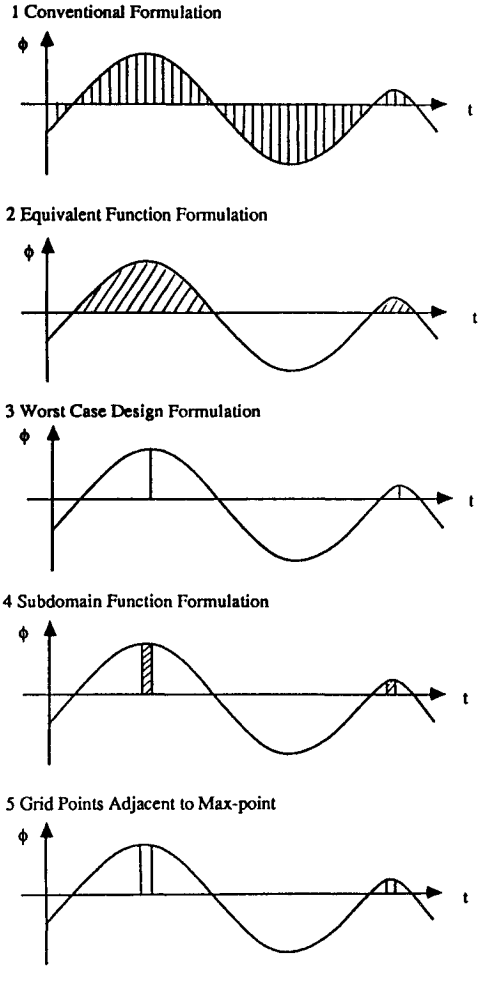

Fig. 1 Five possible treatments of dynamic constraints.

Equation Solver

For integrating state or design sensitivity equations, programs DDERKF, DDEABM, and DDEBDF are available in the package DEPAC.[16] These equation solvers, developed by Sandia Laboratory, use variable-step-size algorithms and have good error control. DDERKF is a fifth-order Runge-Kutta code; DDEAMB is a variable-order (1–12) Adams-Bashforth code; and DDEBDF is based on the variable-order (1–5) backward-differentiation formula. DDERKF and DDEAMB are primarily designed to solve nonstiff and mildly stiff differential equations, and DDEBDF is for stiff equations. If the differential equation is very stiff, DDERKF and DDEABM will be quite inefficient compared to DDEBDF, and may not even change. However, DDEBDF will be inefficient compared to DDERKF and DDEABM for nonstiff equations. Since it is not a priori known that the differential equations might be stiff, DDERKF and DDEABM may not converge. Therefore, DDEBDF needs to be used. To handle this situation, a subroutine that controls the use of differential equations solvers is developed. The subroutine first uses DDEABM and if the intermediate output shows the problem to be stiff, a switch is made to DDEBDF. With this implementation, the differential equation solvers can be used more reliably and efficiently.

Interpolation Schemes

Since variable-step-size differential equation solvers are used in the present study, values of the state and control variables between the grid points are needed to calculate the right-hand side of the sensitivity equations. Therefore, an interpolation scheme is needed to obtain the information between the grid points. Also, to integrate the PI and functional constraints, information between the grid points is needed for a variable-step-size integration rule, e.g., Gaussian quadrature formula. In the treatment of dynamic constraints, an interpolation scheme is needed to locate the maximum points for the worst-case formulation or to evalute the integral for the functional formulation. Thus, for the admissible optimal control formulation, interpolation schemes are needed at several places. The zero- and first-order interpolation functions and piecewise cubic-spline interpolation with natural boundary conditions are evaluated in this study.[17,18]

Numerical Integration Scheme

The Simson's rule and the Gaussian quadrature are evaluated for integrating the integral part of the functional constraints and the sensitivity coefficients with the AVM.

Optimization Algorithm

The optimization algorithm used in this study has been recently developed and evaluated.[4–6,19] It is based on the SQP idea where a potential constraint strategy has been incorporated. The method generates and uses approximate second-order information about the Lagrange function for the problem. At each iteration, a search direction $\boldsymbol{d}$ is determined by solving the following quadratic programming subproblem: minimize $(\boldsymbol{c},\boldsymbol{d}) + 0.5(\boldsymbol{d},\boldsymbol{Hd})$, subject to $(\boldsymbol{N},\boldsymbol{d}) = \boldsymbol{e}$ and $(\boldsymbol{A},\boldsymbol{d}) \leq \boldsymbol{a}$, where $(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^T\boldsymbol{y},\boldsymbol{c}$ is the gradient of the cost function, $\boldsymbol{H}$ is the approximate Hessian of the Lagrange function, $\boldsymbol{N}$ is the matrix whose each column is the gradient of an equality constraint, $\boldsymbol{A}$ is the matrix whose each column is the gradient of an inequality constraint in the potential set, $\boldsymbol{e}$ is the correction vector for the equality constraints, and $\boldsymbol{a}$ is the correction vector for the inequality constraints. Once the search direction has been determined, an appropriate step size is calculated by minimizing a descent function. The design is then updated and the entire process is repeated until convergence is obtained. More details of the algorithm for dynamic-response applications that can be found in Refs. 5, 6, and 19, where it has also been shown that the SQP method without potential constraint strategy is not applicable to dynamic-response problems.

V. Software Environment

A proper software environment must be created to evaluate various numerical procedures systematically. Such an environment can provide directions for future practical applications. Figure 2 shows a conceptual layout of the design environment in which the "designer" is in control of the iterative optimization process. He or she can call upon analysis capabilities to determine response of the system to a set of inputs. The response can be used to evaluate the given design against the specified failure conditions. If the design is not optimum, DSA of the cost and constraint functions can be performed and new design determined through an optimization algorithm.

A prototype software system called OCP having the preceding capabilities has been created. The system is developed using the design optimization software called IDESIGN, which is described in Ref. 20. The dynamic-response optimization capability is implemented through four subroutines as shown in Fig. 3: USERMF (merit function), USERCF (constraint functions), USERMG (merit function gradient), and USERCG (constraint function gradient). Each of the subroutines has a control subroutine CTRLMF, CTRLCF, CTRLMG, and CTRLCG. These subroutines control options of using various numerical procedures, e.g., integration of differential equations, interpolation, integral evaluation, DSA, and treatment of dynamic constraints. The system is modular, so a new version of a particular subroutine or new schemes can be easily incorporated. Three different equation solvers—DDERKF, DDEABM, and DDEBDF, two integration rules—Simpson's rule and the Gaussian quadrature formula, and three interpolation schemes—zero-order, first-order, and piecewise cubic spline have been implemented. Two formulations for treatment of dynamic constraints—conventional and worst case—are implemented. A treatment that combines both formulations is also developed, implemented, and evaluated. Two methods of analytical design sensitivity analysis—DDM and AVM—are implemented. Depending on the problem conditions, the system can automatically select either one; or the user can specify one.

VI. Example Problems

Several dynamic-response optimization and optimal control problems have been used in Ref. 19 as test problems to evaluate the performance of various numerical schemes. These

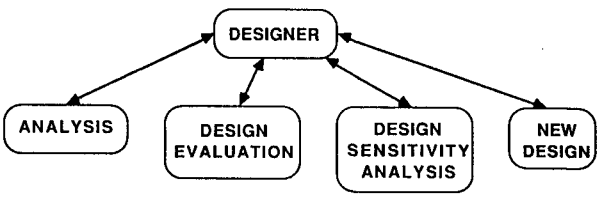

Fig. 2 Conceptual layout of design environment.

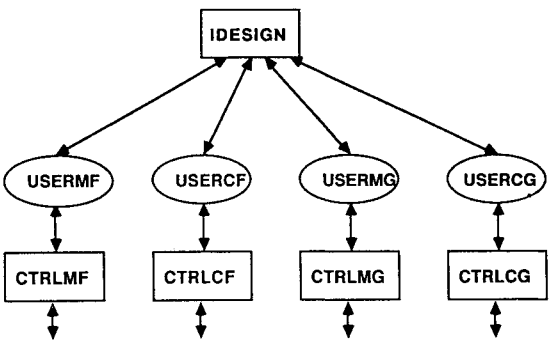

Fig. 3 Structure of the OCP software with optimizer IDESIGN.

problems and combination of various options give about 300 test cases. Many problems have equality and inequality constraints on the dynamic response, bounds on the design variables as well as on the control function. Each case is solved with three optimization algorithms by executing the program in the batch mode. No experimentation is done with the algorithmic parameters to fine-tune algorithms for each problem. Also, a very severe convergence criterion is used to obtain a precise optimum point. For most practical applications, a good solution can be obtained more efficiently with much relaxed stopping criterion. All of the data and the results for the various cases are given in Ref. 19. In this paper, only a few results are given and the performance of numerical procedures summarized.

Example 1: Single-Degree-of-Freedom Nonlinear Impact Absorber

Description of the problem, numerical data, and initial design are given in Ref. 14. Since the design case with $\omega = 4$ is more nonlinear than others, that case is used to study performance of various numerical schemes. More detailed results for other cases can be found in Ref. 19. To compare the performance between a coarse and fine mesh, 101 and 1001 time grid points are selected. DDERKF and DDEABM with an option to switch to DDEBDF are selected for solving first-order differential equations with relative scalar error of 1.E − 7 and absolute scalar error of 3.E − 7. The Simpson's rule (IDINTG = SIMPSN) and Gaussian quadrature formula (GAUSS) are used to carry out the numerical integration over the time interval. Conventional design (IDPTWC = ALL) and worst-case design (MAX) formulations for dynamic constraints, and DDM and AVM for sensitivity analysis are selected to evaluate their performance. Tolerance for the maximum violation and the convergence parameter for optimization algorithm are chosen as 1.E − 3.

Table 1 shows the kind of data collected for the 32 cases of this problem. It shows the number of iterations (NIT) for the SQP method, the number of calls to the function evaluation subroutine (NCF), the total number of gradient (NTG) eval-

Table 1 Summary of results with various options for nonlinear impact absorber ($\omega = 4$)

NGPT	IDDIFF	IDINTG	IDPTWC	IDMETH	NIT	COST	NCF	NTG	CPU	CPU*
101	DDERKF	SIMPSN	ALL	DDM	4	7.535E−01	4	104	93	82
101	DDERKF	SIMPSN	ALL	AVM	4	7.535E−01	4	104	237	146
101	DDERKF	SIMPSN	MAX	DDM	7	7.541E−01	16	18	186	154
101	DDERKF	SIMPSN	MAX	AVM	7	7.541E−01	16	18	125	75
101	DDERKF	GAUSS	ALL	DDM	4	7.535E−01	4	104	92	81
101	DDERKF	GAUSS	ALL	AMV	4	7.535E−01	4	104	261	150
101	DDERKF	GAUSS	MAX	DDM	7	7.541E−01	16	18	186	155
101	DDERKF	GAUSS	MAX	AVM	7	7.541E−01	16	18	121	75
101	DDEABM	SIMPSN	ALL	DDM	4	7.535E−01	4	104	83	70
101	DDEABM	SIMPSN	ALL	AVM	4	7.535E−01	4	104	259	161
101	DDEABM	SIMPSN	MAX	DDM	7	7.541E−01	16	18	175	147
101	DDEABM	SIMPSN	MAX	AVM	7	7.541E−01	16	18	123	72
101	DDEABM	GAUSS	ALL	DDM	4	7.535E−01	4	104	80	67
101	DDEABM	GAUSS	ALL	AVM	4	7.535E−01	4	104	281	166
101	DDEABM	GAUSS	MAX	DDM	7	7.541E−01	16	18	176	154
101	DDEABM	GAUSS	MAX	AVM	7	7.541E−01	16	18	118	70
1001	DDERKF	SIMPSN	ALL	DDM	4	7.541E−01	4	1001	536	243
1001	DDERKF	SIMPSN	ALL	AVM	4	7.541E−01	4	1001	13157	6150
1001	DDERKF	SIMPSN	MAX	DDM	7	7.541E−01	16	18	793	548
1001	DDERKF	SIMPSN	MAX	AVM	7	7.541E−01	16	18	803	332
1001	DDERKF	GAUSS	ALL	DDM	4	7.541E−01	4	1001	534	245
1001	DDERKF	GAUSS	ALL	AVM	4	7.541E−01	4	1001	8345	6149
1001	DDERKF	GAUSS	MAX	DDM	7	7.541E−01	16	18	793	542
1001	DDERKF	GAUSS	MAX	AVM	7	7.541E−01	16	18	601	327
1001	DDEABM	SIMPSN	ALL	DDM	4	7.541E−01	4	1001	487	197
1001	DDEABM	SIMPSN	ALL	AVM	4	7.541E−01	4	1001	10676	3648
1001	DDEABM	SIMPSN	MAX	DDM	7	7.541E−01	16	18	710	461
1001	DDEABM	SIMPSN	MAX	AVM	7	7.541E−01	16	18	741	256
1001	DDEABM	GAUSS	ALL	DDM	4	7.541E−01	4	1001	486	199
1001	DDEABM	GAUSS	ALL	AVM	4	7.541E−01	4	1001	5874	3658
1001	DDEABM	GAUSS	MAX	DDM	7	7.541E−01	16	18	710	468
1001	DDEABM	GAUSS	MAX	AVM	7	7.541E−01	16	18	541	269
Averages										
101	—	—	—	—	5.5	7.538E−01	10	61	162	114
1001	—	—	—	—	5.5	7.541E−01	10	509	2861	1481
—	DDERKF	—	—	—	5.5	7.540E−01	10	285	1678	965
—	DDEABM	—	—	—	5.5	7.540E−01	10	285	1345	629
—	—	SIMPSN	—	—	5.5	7.540E−01	10	285	1824	796
—	—	GAUSS	—	—	5.5	7.540E−01	10	285	1199	798
—	—	—	ALL	—	4.0	7.538E−01	4	552	2592	1338
—	—	—	MAX	—	7.0	7.541E−01	16	18	431	257
—	—	—	—	DDM	5.5	7.540E−01	10	285	382	238
—	—	—	—	AVM	5.5	7.540E−01	10	285	2641	1357

uations of constraint functions, the optimal cost function value COST, CPU time for the entire iterative process, net CPU time for integrating the first-order differential equations on an APOLLO DSP90A computer (indicated by CPU*). The lower part of the table includes the mean values for different conditions. To obtain these, we fix one condition and let others vary and compute a mean value. As the numerical results show, the test runs are successfully solved with the SQP method. The performance with all of the numerical schemes is quite accurate (the optimum cost function has a value of 0.754 in Ref. 7). But with the coarser mesh (101 grid points), the differential equation solver DDEAMB, integration with GAUSS, the worst-case design procedure (MAX), and the DSA with DDM give better performance with respect to efficiency.

Example 2: Linear Two-Degree-of-Freedom Vibration Isolator

This problem having five excitation frequencies is also taken from Ref. 14. The input parameters for the IDESIGN and OCP system are the same as for example 1. The total time interval is set to 2.0 s and the number of time grid points is set to 101, 251, and 1001. In addition, a hybrid (HYB) method for DSA which combines the conventional and worst-case treatments is added and tested. In numerical implementation of this hybrid treatment, if a local maximum point is found between two time grid points, the constraint information for the time grid point nearer to the maximum point is replaced by constraint information for the local maximum point. The example has 10 dynamic constraints and 10 other inequality constraints. Results obtained here and in Refs. 7 and 14 are given in Table 2. All of the test cases are successfully solved using the SQP method. The cost function at the optimum with fine mesh is quite accurate compared with that in the literature, and the NIT and the number of calls for function evaluation (NCF) are also smaller. The cost function with the conventional treatment is slightly smaller than that with the worst-case treatment. The CPU times for HYB and DDM with 101 and 251 points, and AVM with 1001 points are smaller.

Example 3: OCP

The third example is a constrained OCP from Ref. 21. with a dynamic (state variable) constraint. The conventional design (ALL), worst-case design (MAX), and hybrid (HYB) treatment are used. The parameters and conditions for the OCP and IDESIGN are the same as for the previous examples. The results, given in Table 3, show that the MAX needs more iterations and CPU time to converge. The solution of the test runs (NR) 11 and 12 oscillate near the end of the iterative process. These cases can converge with a more relaxed convergence parameter, (1.E-2) and those results are given in Table 3. The test case (NR) 10 fails due to the Hessian matrix being set to identity in two consecutive iterations. As the results show, cases with small number of grid points (NGPT) for the control function (NGPTU) converge in the less number of iterations and CPU time, but the PI has slightly higher value.

VII. Numerical Experience

Effect of NGPT

Since variable-step-size integration methods in DEPAC are used, the NGPT does not affect accuracy of the solution of the equations of motion. However, it affects the accuracy of the location of the maximum-response points for dynamic constraints, the interpolated values for the adjoint vector, and integration for sensitivity coefficients. A larger number of time grid points for the state variable gives more accuracy but is inefficient in calculating the final sensitivity coefficients. It is not easy to select the proper NGPT for general usage. As seen in Tables 1 and 2, the performance with respect to efficiency is very good with 101 points, whereas the accuracy is almost the same as with 1001 points. This is also true for the third example. Use of a large NGPT results in very accurate solution, but it makes the entire design process inefficient.

Since optimal control function in the third example is a smooth function, a small NGPTU can be used, which means smaller number of design variables for the SQP method. In general, this results in good performance with respect to efficiency. As seen in Table 3, the NIT, NCF, and CPU time are quite small with smaller NGPTU, but the PI at optimum is slightly higher. The accuracy of the solution with smaller NGPTU and cubic-spline interpolation is very good.

Comparison of Interpolation Schemes for Control Function

For the third example shown in Table 3, the final PI value varies somewhat with the zero-order interpolation scheme when the NGPTU is increased. However, with the other two schemes, it is almost the same when the NGPTU or the treatment for the dynamic constraints is different. The performance with respect to efficiency with zero-order interpolation scheme is good, but the accuracy is poor. The cubic-spline interpolation scheme appears to be the best one among the three investigated for the present problems.

Comparison of Differential Equation Solvers

The equations of motion are solved by forward integration

Table 2 Optimal solutions for dynamic absorber

NGPT	IDPTWC	IDMETH	b_1	b_2	COST	NIT	CPU	NCF	NTG	Max. vio.	Con. para.
101	ALL	DDM	9.218E−01	1.549E−01	4.279E+00	7	5075	8	605	3.31E−05	9.08E−04
	ALL	AVM	9.220E−01	1.549E−01	4.278E+00	5	17221	5	389	5.03E−05	6.37E−04
	ALL	HYB	9.218E−01	1.549E−01	4.279E+00	7	5093	8	605	3.31E−04	9.08E−04
	MAX	DDM	9.093E−01	1.566E−01	4.452E+00	8	5924	8	263	4.52E−05	4.28E−05
	MAX	AVM	9.093E−01	1.566E−01	4.452E+00	11	16896	12	390	1.96E−04	1.07E−04
	MAX	HYB	9.093E−01	1.566E−01	4.452E+00	8	5937	8	263	4.52E−05	4.28E−05
251	ALL	DDM	9.212E−01	1.569E−01	4.259E+00	7	8372	7	1320	3.91E−05	8.10E−06
	ALL	AVM	9.213E−01	1.570E−01	4.259E+00	11	60421	17	977	3.33E−06	1.08E−04
	ALL	HYB	9.212E−01	1.569E−01	4.259E+00	7	8390	7	1320	3.91E−05	8.10E−06
	MAX	DDM	9.209E−01	1.546E−01	4.296E+00	6	6910	6	118	3.58E−05	4.24E−05
	MAX	AVM	9.209E−01	1.546E−01	4.297E+00	6	6264	6	123	5.23E−05	6.45E−04
	MAX	HYB	9.209E−01	1.546E−01	4.296E+00	6	5509	6	118	3.58E−05	4.24E−05
1001	MAX	DDM	9.213E−01	1.545E−01	4.296E+00	6	7300	6	85	3.73E−05	2.57E−05
	MAX	AVM	9.213E−01	1.545E−01	4.296E+00	6	6136	6	82	2.46E−05	3.71E−05
	MAX	HYB	9.213E−01	1.545E−01	4.296E+00	6	7301	6	85	3.73E−05	2.57E−05
(Equivalent functional)			9.231E−01	1.545E−01	4.291E+00	22	421.7[a]	N/A	(Ref. 14)		
1001	MAX	AVM	9.213E−01	1.545E−01	4.296E+00	11	1507.2[b]	11	(Ref. 22)		

Note: CPU APOLLO DSP90A. [a]IBM 360/65. [b]PRIME 750.

Table 3 Optimal solutions for example 3

NR	IDPTWC	NGPTU	IDINTU	NIT	PI	NCF	NTG	CPU(CPU*)	Max. vio.	Con. para
1	ALL	5	Zero	3	1.6723E+00	3	301	91(43)	3.342E−04	2.469E−04
2			First	7	1.6578E+00	7	704	238(94)	3.780E−06	5.801E−04
3			Cubic	7	1.6575E+00	7	704	260(106)	3.725E−06	4.683E−04
4		51	Zero	16	1.6230E+00	16	1621	7790(4701)	5.756E−06	6.356E−04
5			First	24	1.6560E+00	25	2432	12623(7778)	9.222E−06	8.788E−04
6			Cubic	24	1.6562E+00	24	2430	41956(31276)	2.373E−06	7.369E−04
7	MAX	5	Zero	30	1.6789E+00	53	76	1111(267)	4.890E−05	2.991E−04
8			First	39	1.6576E+00	119	81	1517(373)	1.007E−05	4.309E−04
9			Cubic	42	1.6575E+00	77	87	1882(305)	3.750E−06	9.458E−04
10		51	Zero	7	1.6718E+00	42	21	2491(836)	2.829E−03	3.621E−02
11			First	64	1.6571E+00	218	194	18925(2952)	2.291E−04	6.199E−03
12			Cubic	57	1.6564E+00	178	202	61940(2022)	3.211E−04	8.715E−03
13	HYB	5	Zero	4	1.6789E+00	4	399	115(44)	9.503E−06	2.560E−05
14			First	6	1.6577E+00	6	599	192(61)	5.308E−06	2.391E−04
15			Cubic	9	1.6575E+00	9	899	324(104)	1.077E−06	5.057E−04
16		51	Zero	16	1.6231E+00	16	1601	13698(10646)	7.966E−07	5.864E−04
17			First	22	1.6561E+00	22	2202	15901(11481)	9.527E−06	9.555E−04
18			Cubic	25	1.6562E+00	25	2503	61618(50640)	5.325E−06	7.511E−04

and the adjoint equations are solved by backward integration.[22] The results for the three examples show that the accuracy with DDERKF and DDEABM is almost the same, but the efficiency with DDEABM is better. Also, a substantial amount of CPU time is used in solving differential equations during analysis and DSA (more than 85% in some cases). Therefore, if efficiency of the differential equation solver can be improved, efficiency of the entire iterative design process can be substantially improved.

Comparison of Integration Schemes

The accuracy and efficiency of Simpson's rule and Gaussian quadrature formula depend on the number of integration points used in the entire interval. As noted before, the large number of integration points will be inefficient for integration rules because more function evaluations are needed. In general, the Gaussian quadrature formula requires fewer integration points than the Simpson's rule for the same accuracy of the results.

Methods for Treatment of Dynamic Constraints

The NIT and the number of calls for NCF are smaller but the NTG is larger with the conventional design treatment (ALL). The worst-case design treatment (MAX) with constraints imposed at local maxima has the opposite effect. The number of constraints in the potential set for the worst-case treatment is smaller than that with the conventional treatment at every iteration. Since less information is used in the quadratic programming subproblem, the search direction with the former may not be good for the original NLP problem. As seen in Tables 1–3, the NIT with the worst-case treatment is always larger than that with the conventional treatment. But, the NTG with the worst-case treatment is smaller. The worst-case treatment is more accurate than the conventional treatment but less efficient.

A disadvantage of the conventional treatment is that the NLP problem requires excessive computer memory and time when incorporated with the large number of time grid points. Also, the constraint between the grid points may still be violated at the optimum. The SQP method with potential set strategy works well with conventional treatment of dynamic constraints and is quite straightforward to implement. However, with the worst-case approach, the method is somewhat tedious to implement. The difficulty is in updating the Hessian of the Lagrange function because the difference in its gradients at the two successive points is needed. Since the number of maximum points as well as their location can change, it becomes tedious to keep track of all the information. Consistent procedures, however, have been developed to update the Hessian properly, as explained in Ref. 5.

Comparison of DSA Methods

The total number of differential equations needed to be solved during the entire iterative process is (NCF + NIT*NV) for DDM and (NCF + NTG) for AVM. This means that the number of differential equations for gradient evaluation are NV for DDM at every iteration and NAC for AVM. Based on this observation, a scheme which combines DDM and AVM is used in the last two examples. In this scheme, we compare NV and NAC at each iteration to decide which DSA method is to be used before the OCP system calls the gradient evaluation routine.

As observed from Table 3, the hybrid treatment is more accurate than the conventional treatment but less efficient. Since the dynamic response for the state-variable constraints is very smooth, the results with the conventional treatment are not too different from those with the hybrid treatment. In general, the HYB treatment is as efficient as the conventional treatment and as accurate as the worst-case treatment.

VIII. Summary and Conclusions

Based on the present study, various numerical procedures are evaluated as follows:

1) The variable-step-size differential equation solvers with interpolation schemes for the state variables and control functions require a large number of calculations during analysis as well as sensitivity analysis. This enormous computation makes the entire design process fairly inefficient. It is, however, quite accurate. Efficiency of the differential equation solvers needs to be improved. This way, efficiency of the entire iterative design process can be substantially improved. Although the new versions of differential equation solvers in DEPAC are robust and accurate, more efficient variable-step integrators need to be developed.

2) The Gaussian quadrature formula and the piecewise cubic-spline interpolation worked well for most of the examples in the present study. The NGPT for the state variables and the control functions need to be very carefully selected in dynamic problems. A larger NGPT can give better performance with respect to accuracy but not efficiency.

3) Dynamic constraints need to be satisfied over the entire time interval. Several treatments are presented to eliminate time from the constraints. They result into a different history of design change during the iterative process. The SQP method with potential constraint strategy and conventional treatment takes less computational effort. The SQP method with the worst-case treatment needs to be improved for the dynamic-response optimization problems.

4) The present study shows that a number of different numerical procedures can be used at different stages of the

optimal design process for dynamic systems. The software for design of such systems should have options for automatic selection of the procedures. It can be observed from the results presented in the paper that the computational effort can vary substantially based on the numerical procedures used at various stages (e.g., 80–13,157 s for example 1, 5075–60,421 s for example 2, and 91–61,940 s for example 3). Therefore, it is extremely important to select them judiciously. In this regard, use of interactive facilities, where various procedures and problem formulation can be interactively changed, are recommended. Such procedures are discussed in Refs. 19 and 23.

Based on the study of various numerical procedures, the following recommendations are made for general usage:

1) Equation solvers. If possible, fixed-step solvers should be used to integrate state as well as sensitivity differential equations. This way, the interpolation step for evaluating the right-hand side at points other than the grid points can be eliminated. The variable-step-size procedures should be used only when absolutely necessary. For linear systems, modal analysis is recommended. The software should have all of the procedures implemented which can then be automatically selected or specified by the user.

2) Treatment of dynamic constraints. If the optimization uses a potential set strategy, then conventional or the new hybrid treatment is recommended. For algorithms without potential set strategy, the new hybrid treatment should be preferred.

3) DSA procedure. Automatic selection of either DDM or AVM is recommended.

4) Interpolation procedure. The piecewise cubic-spline interpolation scheme among the three investigated is recommended.

5) Integral evaluation procedure. The Gaussian quadrature formula is recommended.

6) Optimization algorithm. The SQP algorithm with potential set strategy is recommended.

Acknowledgment

This research is sponsored in part by National Science Foundation Grant MSM 86-13314.

References

[1]Arora, J. S. and Thanedar, P. B., "Computational Methods for Optimal Design of Large Complex Systems," *Computational Mechanics*, Vol. 1, No. 2, 1986, pp. 221–224.

[2]Schittkowski, K. (ed.), *Computational Mathematical Programming*, Series F: Computer and System Sciences, Vol. 15, NATO-ASI, Springer-Verlag, New York, 1985.

[3]Thanedar, P. B., Arora, J. S., and Tseng, C. H., "A Hybrid Optimization Method and Its Role in Computer-Aided Design," *Computers and Structures*, Vol. 23, No. 3, 1986, pp. 305–314.

[4]Lim, O. K. and Arora, J. S., "An Active Set RQP Algorithm for Engineering Design Optimization," *Computer Methods in Applied Mechanics and Engineering*, Vol. 57, No. 1, 1986, pp. 51–65.

[5]Lim, O. K. and Arora, J. S., "Dynamic-Response Optimization Using an Active Set RQP Algorithm," *International Journal for Numerical Methods in Enginering*, Vol. 24, No. 10, Oct. 1987, pp. 1827–1840.

[6]Tseng, C. H. and Arora, J. S., "On Implementation of Computational Algorithms for Optimal Design 1: Preliminary Investigation; 2: Extensive Numerical Investigation," *International Journal for Numerical Methods in Engineering*, Vol. 26, No. 6, June 1988, pp. 1365–1402.

[7]Hsieh, C. C. and Arora, J. S., "An Efficient Method for Dynamic Response Optimization," *AIAA Journal*, Vol. 23, Sept. 1985, pp. 1454–1456.

[8]Tabak, D. and Kuo, B. C., *Optimal Control by Mathematical Programming*, Prentice-Hall, Englewood Cliffs, NJ, 1973.

[9]Sage, A. P. and White, C. C., *Optimum Systems Control*, 2nd ed. Prentice-Hall, Englewood Cliffs, NJ, 1977.

[10]Knox, J. R., "On the Application of Nonlinear Programming to the Solution of Optimal Output-Constrained Regulator Problems," *Control Applications of Nonlinear Programming*, Proceedings of the IFAC Workshop, Denver, CO, June 1979.

[11]Kraft, D., "Comparing Mathematical Programming Algorithms Based on Lagrangian Functions for Solving Optimal Control Problems," *Control Applications of Nonlinear Programming*, Proceedings of the IFAC Workshop, Denver, CO, June, 1979.

[12]Kraft, D., "Finite-Difference Gradients Vs Error-Quadrature Gradients in the Solution of Parameterized Optimal Control Problems," *Optimal Control Applications and Methods*, Vol. 2, April–June, 1981, pp. 191–199.

[13]Kraft, D., "On Converting Optimal Control Problems into Nonlinear Programming Problems," *Computational Mathematical Programming*, NATO ASI Series, Vol. F15, edited by K. Schittowski, Springer-Verlag, New York, 1985, pp. 261–280.

[14]Haug, E. J. and Arora, J. S., *Applied Optimal Design: Mechanical and Structural Systems*, Wiley, New York, 1979, Chap. 5.

[15]Bryson, A. E. and Ho, Y. O., *Applied Optimal Control*, Revised printing, Ginn & Co., Waltham, MA, 1975.

[16]Shampine, L. F. and Watts, H. A., "DEPAC—Design of User Oriented Package of ODE Solvers," SAND 79-2347, Sandia Laboratory, 1979.

[17]De Boor, C., "A Practical Guide to Spline," *Applied Mathematical Sciences*, Vol. 27, Springer-Verlag, New York, 1978.

[18]Atkinson, K. E., *An Introduction to Numerical Analysis*, Wiley, New York, 1978.

[19]Tseng, C. H. and Arora, J. S., "Optimal Design for Dynamics and Control by using a Sequential Quadrature Programming Algorithm," Optimal Design Laboratory, College of Engineering, The Univ. of Iowa, Iowa City, IA, TR ODL-87.10, Dec. 1978.

[20]Arora, J. S. and Tseng, C. H., "Interactive Design Optimization," *Engineering Optimization*, Vol. 13, No. 3, 1988, pp. 173–188.

[21]Di Pillo, G., Grippo, L., and Lampariello, F., "A Newton-Type Computing Technique for Optimal Control Problems," *Optimal Control Applications and Methods*, Vol. 5, No. 2, April–June, 1984, pp. 149–166.

[22]Hsieh, C. C. and Arora, J. S., "Design Sensitivity Analysis and Optimization of Dynamic Response," *Computer Methods in Applied Mechanics and Engineering*, Vol. 43, No. 2, April 1984, pp. 195–219.

[23]Tseng, C. H. and Arora, J. S., "Interactive Design Optimization of Dynamic Response," under review.