Improved Gradient Algorithm for Two-Point Boundary Value Problems

H. GARDNER MOYER*
Grumman Aerospace Corporation, Bethpage, N.Y.

Although variable final point problems can be readily solved by the gradient procedure, the two-point case is troublesome. In the present paper the original two-point boundary value problem is replaced by one with a variable endpoint. The minimum of the cost functional of the original formulation appears in the new problem as a free parameter. The two problems have identical solution extremals provided the parameter is given its correct value. The new problem is solved initially using an estimate for the parameter and a first order correction is then calculated. This process is repeated until all the boundary values lie within their tolerances. The efficacy of the method is shown by its application to three-dimensional aircraft trajectory optimization.

Introduction

THE gradient algorithm[1] is still widely used for control problems owing to its simplicity, reliability, and—at least in its conjugate gradient version[2]—the relative rapidity of its rate of convergence. Indeed, if the problem involves tabulated functions, most competing methods[3] (e.g., the indirect, quasi-linearization, and second variation) are at a severe disadvantage, either because the values of the control variables that maximize the Hamiltonian cannot be expressed explicitly, or because the numerical approximations to the various second-order partial derivatives are unreliable.

The gradient method is well suited to variable endpoint applications, but when it is used for two-point boundary value problems it must be supplemented by techniques[4-6] that seriously degrade one or more of the values mentioned above. The simplest technique augments the cost functional with penalty functions containing constants that are increased periodically.[1,7] As the constants approach infinity the boundary value errors approach zero. In the process, however, the penalty terms overwhelm the original cost functional so that in practice the latter approaches its minimum only when careful attention is paid to each problem.

The present paper shows how to make the latter technique reliable while retaining its simplicity and rapidity. The original two-point boundary value problem is replaced by a variable endpoint problem whose cost functional is the penalty terms mentioned. The value of the cost functional of the original problem appears in the new problem as a free parameter. Therefore, the two problems have identical solution extremals when the parameter is chosen correctly. Thus the task of the present algorithm is to find the right value of the parameter and then solve the variable endpoint problem. The efficacy of the algorithm will be illustrated by its application to a three-dimensional aircraft trajectory problem.

Time-Optimal Case

First the time-optimal case will be treated. The original problem will be called Problem A (see Table 1) to distinguish it from the variable endpoint problems that will be introduced later. The system whose state n-vector x and control m-vector u obey

$$dx/dt = f(x,u,t) \tag{1}$$

is to be transferred from the initial point

$$x = x_o, \qquad t = t_o \tag{2}$$

to the final point

$$x = x_f \tag{3}$$

via the path that minimizes the cost functional

$$P^A = \tau \tag{4}$$

Here τ is the final value of the independent time variable t. The value of τ defined by the extremal that solves Problem A will be called τ^A.

In order to keep $u(t)$ as smooth as possible, it is advisable to avoid extrapolation by replacing t with

$$\theta = (t-t_o)/(\tau-t_o) \tag{5}$$

whose range is always 0 to 1. The final time τ could now be regarded as a free parameter; however, the analysis will be a little clearer if we regard it instead as an additional state variable.[8] If x continues to represent only the original state variables, the system equations take on the form

$$dx/d\theta = (\tau-t_o)f[x,u,t_o+(\tau-t_o)\theta] \tag{6a}$$

$$d\tau/d\theta = 0 \tag{6b}$$

The algorithm begins with a preliminary stage whose purpose is merely to find an optimal trajectory that approximates the solution. This is accomplished by solving Problem B (see Table 1) which differs from Problem A in that the constraints [Eq. (3)] on the final point are ignored and the cost functional [Eq. (4)] is replaced by

$$P^B = \tau + \tfrac{1}{2}\|x(1)-x_f\|_K^2 \tag{7}$$

The error norm is defined as

$$\|x-x_f\|_K^2 = (x-x_f)^T K(x-x_f) \tag{8}$$

where K is a diagonal matrix of penalty function constants. The value of each of these constants is chosen by assuming a reasonable tradeoff between τ and the boundary value error.

To solve Problem B by the gradient method we introduce adjoint variables λ that obey

$$\frac{d}{d\theta}(\lambda_x{}^T\delta x + \lambda_\tau \delta\tau) = (\tau-t_o)\lambda_x{}^T \frac{\partial f}{\partial u}\delta u = \frac{\partial H}{\partial u}\delta u \tag{9}$$

Table 1 Problem formulations

Problem	Cost functional	Initial conditions	Final conditions
A	τ	$x = x_o$, τ open λ_x open, $\lambda_\tau = 0$	$x = x_f$ λ_x open, $\lambda_\tau = -1$
B	$\tau+\frac{1}{2}\|x-x_f\|_K^2$	same as above	$\lambda_x = -K(x-x_f)$ $\lambda_\tau = -1$
C	$\frac{1}{2}\|x-x_f\|_K^2$	$x = x_o$, τ fixed λ_x, λ_τ open	τ open, $\lambda_\tau = 0$ $\lambda_x = -K(x-x_f)$

Received June 11, 1974; revision received July 15, 1974.
Index category: Navigation, Control and Guidance Theory.
* Research Mathematician. Member AIAA.

The right member has been simplified by defining the generalized Hamiltonian H as

$$H \equiv (\tau - t_o)\lambda_x{}^T f[x, u, t_o + (\tau - t_o)\theta] \tag{10}$$

The differential equations that λ must satisfy when Eq. (9) holds can be found by expanding the left member of this equation and using the equations of variation of Eqs. (6).

$$\begin{aligned} d\lambda_x/d\theta &= -(\partial H/\partial x)^T \\ d\lambda_\tau/d\theta &= -\partial H/\partial \tau \end{aligned} \tag{11}$$

When the final values of the adjoint variables are defined as

$$\begin{aligned} \lambda_x{}^T(1) &= -\partial P^B/\partial x = -[x(1) - x_f]^T K \\ \lambda_\tau(1) &= -\partial P^B/\partial \tau = -1 \end{aligned} \tag{12}$$

the integral of Eq. (9) takes on the form

$$-\delta P^B = -[x(1) - x_f]^T K \delta x - \delta\tau = \lambda_x{}^T \delta x + \lambda_\tau \delta\tau|_{\theta=0} + \int_0^1 \frac{\partial H}{\partial u}\,\delta u\, d\theta = \lambda_\tau(0)\delta\tau + \int_0^1 \frac{\partial H}{\partial u}\,\delta u\, d\theta \tag{13}$$

The last member assumes

$$\delta x(0) = 0 \tag{14}$$

since all the trajectories obey the initial conditions [Eq. (2)]. In order to vary P^B in the direction of steepest descent, we choose

$$\delta\tau = \sigma\lambda_\tau(0), \qquad \delta u(\theta) = \sigma\left[\frac{\partial H}{\partial u}(\theta)\right]^T \tag{15}$$

Thus when σ is small enough to justify a first-order analysis, the change in the cost functional is

$$\delta P^B = -\sigma[\lambda_\tau{}^2(0) + \int (\partial H/\partial u)^2\, d\theta] \tag{16}$$

After the first descent the conjugate gradient version of Eq. (15) can be used.

$$\begin{aligned} \delta\tau_j &= \sigma_j\left[\lambda_{\tau_j}(0) + \beta_\tau\left(\frac{\delta\tau}{\sigma}\right)_{j-1}\right], \\ \delta u_j &= \sigma_j\left[\left(\frac{\partial H}{\partial u}\right)_j^T + \beta_u\left(\frac{\delta u}{\sigma}\right)_{j-1}\right] \end{aligned} \tag{17}$$

Here the subscript j represents the descent number and β_τ and β_u stand for

$$\begin{aligned} \beta_\tau &= \lambda_{\tau_j}^2(0)/\lambda_{\tau_{j-1}}^2(0), \\ \beta_u &= \left[\int_0^1 \left(\frac{\partial H}{\partial u}\right)_j^2 d\theta\right]\left[\int_0^1 \left(\frac{\partial H}{\partial u}\right)_{j-1}^2 d\theta\right]^{-1} \end{aligned} \tag{18}$$

The conjugate gradient algorithm proceeds by 1) computing a starting trajectory, 2) integrating Eq. (11) backward from Eq. (12), 3) incrementing τ and $u(\theta)$ in accordance with Eq. (17) ($\beta_u = \beta_\tau = 0$ on the first descent), and 4) integrating Eq. (6) forward from Eq. (2). Steps 3) and 4) are repeated several times using different values of σ and the trajectory with the smallest P^B provides the basis for the next backward integration. Steps 2), 3), and 4) comprise a "descent cycle" and are repeated until $\lambda_\tau(0)$ and $\partial H(\theta)/\partial u$ have become negligible. We see that the gradient algorithm is well suited to this variable final point problem because the state variables are assigned values only at the initial point and the adjoint variables only at the final point.

Next the results from Problem B will be used to obtain a first order estimate of τ^A. Equation (9) is integrated assuming only that its right side is zero.

$$\lambda_x{}^T \delta x + \lambda_\tau \delta\tau|_{\theta=1} = \lambda_x{}^T \delta x + \lambda_\tau \delta\tau|_{\theta=0} \tag{19}$$

This important relation holds between the endpoints of any extremal and those of any adjacent arc whose $\delta x(\theta)$, $\delta u(\theta)$, $\delta\tau$ obey the equations of variation of Eqs. (6). Let us evaluate it for the extremal of Problem B using the extremal of Problem A as the comparison curve. This implies $\delta x(0) = 0$ and $\delta x(1) = x_f - x(1)$, so that Eq. (19) becomes

$$\lambda_x{}^T(x_f - x) + \lambda_\tau \delta\tau|_{\theta=1} = \lambda_\tau \delta\tau|_{\theta=0} \tag{20}$$

On taking account of Eq. (12) and recalling from the previous paragraph that $\lambda_\tau(0) = 0$, we see that the first order estimate for the difference between the final times of the two extremals is[1]

$$\delta\tau^B = [x(1) - x_f]^T K[x(1) - x_f] \tag{21}$$

The algorithm now proceeds to the second stage which works with Problem C (Table 1). The cost functional

$$P^C = \tfrac{1}{2}\|x(1) - x_f\|_K^2 \tag{22}$$

is to be minimized with τ kept fixed at the following first-order estimate for τ^A.

$$\tau^C = \tau^B + \delta\tau^B \tag{23}$$

The solution procedure is the same as that given for Problem B except that the second of Eqs. (12) and the first of Eqs. (17) are replaced with

$$\lambda_\tau(1) = 0, \qquad \delta\tau(0) = 0, \tag{24}$$

respectively.

The above definition for the variable endpoint Problem C has been chosen so that its solution extremal will be identical to that of the fixed endpoint Problem A provided τ^C is exactly equal to τ^A. If τ^C turns out to be smaller than τ^A, the boundary value errors obtained from the solution to Problem C may exceed the tolerances of Problem A. A new increment in τ can then be obtained from Eq. (20) after substituting from the first of each of Eqs. (12) and (24).

$$\delta\tau^C = [x(1) - x_f]^T K[x(1) - x_f]/\lambda_\tau(0) \tag{25}$$

The τ^C defined by Eq. (23) might also be too large. In this case the algorithm converges to a nonextremal trajectory that makes the boundary value errors negligible. There is no way of estimating the magnitude of $\tau^C - \tau^A$. Therefore, the algorithm presented below recommends that the increments in τ given by Eqs. (21) and (25) be divided by two as a safety measure.

The algorithm is summarized in the following steps: a) Compute a starting trajectory. Set the descent counter j to zero. b) Compute a descent cycle for Problem B. $j+1 \to j$. c) Compute $\delta\tau$ from Eq. (21). d) If $j < 5$ go to Step b). e) If the conjugate gradient direction decreases τ [see Eq. (17a)], go to Step b). f) If the values of $\tau + \delta\tau$ computed from the three most recent descents do not agree to within a given tolerance, go to Step b). g) Compute a new starting trajectory with $\tau + \delta\tau/2 \to \tau$. $j = 0$. h) Compute a descent cycle for Problem C. $j+1 \to j$. i) Terminate the computation if all boundary value errors lie within given tolerances. j) Compute $\delta\tau$ from Eq. (25). k) If $j < 5$, go to Step h). l) If $\delta\tau < 0$, go to step h). m) If the values of $\delta\tau$ computed from the three most recent descents do not agree to within a given tolerance, go to Step h). n) Go to Step g).

The numbers 5 and 3 in Steps d), f), k), and m) were chosen arbitrarily. Step e) was inserted to help the program converge to a τ^B that is below τ^A. Step l) helps test for convergence because $\delta\tau$ takes on a positive value when $\tau^C < \tau^A$ and a negligible value when $\tau^C > \tau^A$.

Application to Aircraft Trajectories

The above conjugate gradient algorithm has been used to optimize the turning flight performance in a uniform gravitational field of the F-14A jet fighter.[9] This example was chosen to illustrate the ability of the algorithm to bring in a large number of boundary values for a complex problem. The dynamic equations are

$$\begin{aligned} dy_1/d\theta &= \tau V \cos\gamma \cos\chi \\ dy_2/d\theta &= \tau V \cos\gamma \sin\chi \\ dy_3/d\theta &= \tau V \sin\gamma \\ dV/d\theta &= \tau[T\cos(\alpha+\varepsilon) - \tfrac{1}{2}\rho V^2 S C_D - mg\sin\gamma]/m \\ d\gamma/d\theta &= \tau[T\sin(\alpha+\varepsilon)\cos\mu + \tfrac{1}{2}\rho V^2 S C_L \cos\mu - mg\cos\gamma]/mV \\ d\chi/d\theta &= \tau[T\sin(\alpha+\varepsilon) + \tfrac{1}{2}\rho V^2 S C_L]\sin\mu/mV\cos\gamma \\ d\tau/d\theta &= 0 \end{aligned} \tag{26}$$

Here y_1, y_2, y_3 are rectangular coordinates of the position with y_3 the altitude. The remaining state variables are the velocity magnitude V, the flight path angle γ, and the heading angle χ.

Table 2 Sequence of final values for an aircraft trajectory problem with three final conditions

	τ (sec)	y_1 (ft)	y_2 (ft)	y_3 (ft)
0 descents	20.000	28,712	4,071	41,204
8 descents	26.071	30,215	23,436	34,029
13 descents	28.379	29,192	28,101	34,876
18 descents	29.041	29,580	29,105	34,953
23 descents	29.361	29,768	29,585	34,983
28 descents	29.520	29.873	29,818	34,996
33 descents	29.599	29,936	29,929	35,000
36 descents	29.637	29,968	29,981	35,002
constraints		30,000	30,000	35,000

Table 3 Sequence of final values for an aircraft trajectory problem with five final conditions

	τ (sec)	y_1 (ft)	y_2 (ft)	y_3 (ft)	γ (deg)	χ (deg)
0 descents	29.599	29,907	29,945	34,983	9.8703	80.809
30 descents	29.111	28,959	29,110	34,910	0.2151	85.618
37 descents	29.787	29,463	29,916	34,955	−0.3428	87.088
70 descents	30.377	29,997	30,210	35,001	−0.4294	89.246
110 descents	30.377	29,995	30,059	34,999	−0.1285	89.737
158 descents	30.377	29,993	30,019	34,999	−0.0469	89.881
constraints		30,000	30,000	35,000	0.0000	90.000

The control variables are the angle-of-attack α and the bank angle μ. A curve fit is given for the atmospheric mass density $\rho(y_3)$. The mass m, reference area S, thrust cant ε, and acceleration of gravity g are constants. Bivariate tables are given for the thrust T(Mach, y_3), lift coefficient C_L(Mach, α), and drag coefficient C_D(Mach, C_L). These data are somewhat noisy owing to their experimental origin. The interpolation for the thrust is by spline in the mach direction and linear in the y_3 direction. The aerodynamic coefficients are obtained from a combination of curve fitting and linear interpolation.

The first problem is to find the time-optimal path from $t_0 = y_1 = y_2 = \gamma = \chi = 0$, $y_3 = 35{,}000$ ft, $V = 1{,}500$ fps to $y_1 = y_2 = 30{,}000$ ft, $y_3 = 35{,}000$ ft with the final values of the other state variables open. An open state variable simply means that the corresponding penalty function constant is zero. The results computed on the IBM 370/168 with double precision are shown in Table 2. Eight descents are applied to Problem B and τ is then incremented from 26.071 sec to 28.379 sec in accordance with Step g) of the algorithm. During the second stage τ is incremented after every five descents—the minimum allowed by Step k). The tolerances on the boundary values are satisfied after thirty-six descents.

Next a problem that specifies constraints on the final values of five state variables is attempted. The initial and final conditions are the same as before except that $\gamma(1)$ is required to be zero and $\chi(1)$ to be 90°. The starting trajectory is that generated on the thirtieth descent of the previous problem. The results are shown in Table 3. Thirty descents are required for Problem B and the final time is incremented again after the thirty-seventh descent. Further changes in the final time are prevented by Step l) of the algorithm. All the boundary value errors are within the tolerances after 158 descents.

Generalizations and Conclusions

The general principles that have been presented can of course be applied to other types of problems. For example, suppose that the cost function is $x_n(1)$ with the final values of the other state variables specified. The final time can be either fixed or open. Let us define $x_n{}^A$ as the value of $x_n(1)$ on the solution extremal. If $x_n{}^A$ were known, the gradient program could generate the solution extremal by minimizing the error norm (22) with $x_{n_f} = x_n{}^A$. But a first-order approximation for $x_n{}^A$ can be found from Eq. (19) using the adjoint variables defined by the extremal that minimizes

$$P = x_n(1) + \tfrac{1}{2}\sum_{i=1}^{n-1} k_i[x_i(1) - x_{i_f}]^2 \tag{27}$$

Further generalizations will be left to the motivated reader.

The procedures presented in Refs. 4–6 employ trajectories that satisfy the boundary conditions but not the extremal condition $\partial H/\partial u = 0$ to approach the minimum of the cost functional from above. The present algorithm takes the converse standpoint and employs extremals that do not satisfy the boundary conditions to approach the minimum from below. The latter is certainly simpler than other gradient methods and appears to be at least competitive with regard to time of computation.

References

[1] Kelley, H. J., "Methods of Gradients," *Optimization Techniques*, 1st ed., edited by G. Leitmann, Academic Press, New York, 1962, pp. 205–254.

[2] Lasdon, L. S., Mitter, S. K., and Waren, A. D., "The Conjugate Gradient Method for Optimal Control Problems," *IEEE Transactions on Automatic Control*, Vol. AC-12, No. 2, Feb. 1967, pp. 132–138.

[3] Kopp, R. E. and Moyer, H. G., "Trajectory Optimization Techniques," *Advances in Control Systems*, edited by C. T. Leondes, 1st ed., Vol. 4, Academic Press, New York, 1966, pp. 103–155.

[4] Miele, A., Pritchard, R. E., and Damoulakis, J. N., "Sequential Gradient Restoration Algorithm for Optimal Control Problems," *Journal of Optimization Theory and Applications*, Vol. 5, No. 4, 1970, pp. 235–282.

[5] Williamson, W. E. and Eller, T. J., "Riccati Transformation for Gradient Optimization Techniques," *AIAA Journal*, Vol. 11, No. 6, June 1973, pp. 881–883.

[6] Rajtora, S. G. and Pierson, B. L., "An Automated Gradient Projection Algorithm for Optimal Control Problems," *AIAA Journal*, Vol. 10, No. 7, July 1972, pp. 949–951.

[7] Kelley, H. J. and Falco, M., "Aircraft Symmetric Flight Optimization," *Controls and Dynamic Systems: Advances in Theory and Applications*, edited by C. T. Leondes, 1st ed., Vol. 10, Academic Press, New York, 1973, pp. 89–129.

[8] Long, R. S., "Newton-Raphson Operator: Problems with Undetermined Endpoints," *AIAA Journal*, Vol. 3, No. 7, July 1965, pp. 1351–1352.

[9] Moyer, H. G. and Hinz, H. K., "Three Dimensional Aircraft Trajectory Optimization," Research Dept. Rept., Grumman Aerospace, Bethpage, N.Y., in preparation.