Use of Approximate Gradients in Trajectory Optimization

Ping Lu*
Iowa State University, Ames, Iowa 50011

Introduction

THE nonlinear programming approach has proven quite successful in solving optimal control problems, especially for highly nonlinear complicated systems. It does not usually require a very good initial estimate of the solution and offers great flexibility when the system model changes. Also, considerably less programming efforts are needed when all gradient information is generated by finite differences. In some problems, such as trajectory optimization for multistage launch vehicles where discontinuities in the system are inherent, the application of the necessary conditions in the function space[1] is almost prohibitive. For those problems, the nonlinear programming approach has demonstrated great superiority over other methods.[2]

On the other hand, the efficiency of the nonlinear programming approach is not as good because additional $n+1$ (n = dimensionality of the problem) integrations of the system equations are required to compute the gradients in each iteration by finite differences. As a result, the dominant portion of computational time is spent on the gradient evaluations. There are situations where it is essential to obtain quickly a trajectory that satisfies all terminal constraints and any other imposed constraints when a slight penalty on the performance index is tolerable. Such a trajectory serves all of the practical purposes well even if it is just near optimal. In this Note, we shall explore the possibility of using approximate gradients to enhance the efficiency of the nonlinear programming approach. A formula for approximate gradients is derived that is more accurate than a well known result. The method is tested on trajectory optimization for an advanced launch system.

Approximate Gradients Generation

Suppose that after proper parameterization, the optimal control problem has been transformed into the following nonlinear programming problem:

$$\min F(\boldsymbol{x}) \tag{1}$$

$$\boldsymbol{h}(\boldsymbol{x})=0 \tag{2}$$

$$\boldsymbol{c}(\boldsymbol{x})\geq 0 \tag{3}$$

where we assume that $F:\boldsymbol{R}^n\to\boldsymbol{R}\in C^3$, $\boldsymbol{h}:\boldsymbol{R}^n\to\boldsymbol{R}^m\in C^3$, $\boldsymbol{c}:\boldsymbol{R}^n\to\boldsymbol{R}^l\in C^3$, $\boldsymbol{x}\in\boldsymbol{R}^n$. We have observed in many applications that after a number of iterations, changes in the parameter vector $\boldsymbol{x}$ become small from $\boldsymbol{x}_k$ to $\boldsymbol{x}_{k+1}$. Thus changes in $F(\boldsymbol{x})$, $\boldsymbol{h}(\boldsymbol{x})$, and $\boldsymbol{c}_x$ are small. For the derivations in the sequel, the equality sign "=" is used in some equations where it obviously means "$\approx$." Let us first consider the changes in $F(\boldsymbol{x})$. In the kth iteration, after the line search is completed, the next point $\boldsymbol{x}_{k+1}$ is found. The gradient of $F(\boldsymbol{x})$ at $\boldsymbol{x}_{k+1}$ is to be computed. Instead of using finite differences, we may try to obtain an estimate of the gradient by first approximating $F(\boldsymbol{x})$ in the neighborhood of $\boldsymbol{x}_{k+1}$ by the first-order expansion:

$$F(\boldsymbol{x})=F(\boldsymbol{x}_{k+1})+\boldsymbol{g}^T(\boldsymbol{x}_{k+1})(\boldsymbol{x}-\boldsymbol{x}_{k+1})$$

where $\boldsymbol{g}=\nabla F$. Let $\boldsymbol{x}=\boldsymbol{x}_k$, drop the arguments for simplicity, and rearrange the preceding equation:

$$\Delta F=F_{k+1}-F_k=\boldsymbol{g}_{k+1}^T(\boldsymbol{x}_{k+1}-\boldsymbol{x}_k)=\boldsymbol{g}_{k+1}^T\Delta\boldsymbol{x} \tag{4}$$

We look for an approximation of the form $\boldsymbol{g}_{k+1}=\boldsymbol{g}_{k+1}(\boldsymbol{g}_k,\Delta F,\Delta\boldsymbol{x})$. The following formula

$$\boldsymbol{g}_{k+1}=\boldsymbol{g}_k+\frac{(\Delta F-\boldsymbol{g}_k^T\Delta\boldsymbol{x})}{\Delta\boldsymbol{x}^T\Delta\boldsymbol{x}}\Delta\boldsymbol{x} \tag{5}$$

has the desired form and reduces to Eq. (4) when multiplied by $\Delta\boldsymbol{x}^T$ on both sides. It is known as the Broyden rank-one update[3] when $\boldsymbol{g}$ is replaced by the Jacobian. Furthermore, if we expand $F(\boldsymbol{x})$ in the neighborhood of $\boldsymbol{x}_{k+1}$ by the second-order approximation, we have

$$\Delta F=\boldsymbol{g}_{k+1}^T\Delta\boldsymbol{x}-\tfrac{1}{2}\Delta\boldsymbol{x}^T\boldsymbol{H}_{k+1}\Delta\boldsymbol{x} \tag{6}$$

where $\boldsymbol{H}_{k+1}$ is the Hessian at $\boldsymbol{x}_{k+1}$. The update

$$\boldsymbol{g}_{k+1}=\boldsymbol{g}_k+\frac{(\Delta F+0.5\Delta\boldsymbol{x}^T\boldsymbol{H}_{k+1}\Delta\boldsymbol{x}-\boldsymbol{g}_k^T\Delta\boldsymbol{x})}{\Delta\boldsymbol{x}^T\Delta\boldsymbol{x}} \tag{7}$$

conforms with Eq. (6). Equation (7), nonetheless, is not readily applicable because the unknown $\boldsymbol{H}_{k+1}$ is involved. Any approximations to $\boldsymbol{H}_{k+1}$ will require additional computation, which is against our original intention "in conducting" this investigation. However, by examining Eq. (7), we see that we only need the scalar $\Delta\boldsymbol{x}^T\boldsymbol{H}_{k+1}\Delta\boldsymbol{x}$, not $\boldsymbol{H}_{k+1}$ itself. So for small $\Delta\boldsymbol{x}$, we have the first-order approximation

$$\boldsymbol{g}_{k+1}-\boldsymbol{g}_k=\boldsymbol{H}_{k+1}\Delta\boldsymbol{x} \tag{8}$$

Multiplying Eq. (8) by $\Delta\boldsymbol{x}^T$ and approximating $\boldsymbol{g}_{k+1}-\boldsymbol{g}_k$ by Eq. (7) lead to

$$\tfrac{1}{2}\Delta\boldsymbol{x}^T\boldsymbol{H}_{k+1}\Delta\boldsymbol{x}=\Delta F-\boldsymbol{g}_k^T\Delta\boldsymbol{x} \tag{9}$$

Substituting Eq. (9) into Eq. (7) gives rise to

$$\boldsymbol{g}_{k+1}=\boldsymbol{g}_k+2\frac{(\Delta F-\boldsymbol{g}_k^T\Delta\boldsymbol{x})}{\Delta\boldsymbol{x}^T\Delta\boldsymbol{x}}\Delta\boldsymbol{x} \tag{10}$$

There is another rather interesting way of deriving Eq. (10). If we approximate $\boldsymbol{g}_{k+1}-\boldsymbol{g}_k$ in Eq. (8) by the simple expression of Eq. (5), instead of the more elaborate formula of Eq. (7), we have

$$\Delta\boldsymbol{x}^T\boldsymbol{H}_{k+1}\Delta\boldsymbol{x}=\Delta\boldsymbol{x}^T(\boldsymbol{g}_{k+1}-\boldsymbol{g}_k)=\Delta F-\boldsymbol{g}_k^T\Delta\boldsymbol{x} \tag{11}$$

Substituting Eq. (11) into Eq. (7),

$$\boldsymbol{g}_{k+1}=\boldsymbol{g}_k+\frac{3}{2}\frac{(\Delta F-\boldsymbol{g}_k^T\Delta\boldsymbol{x})}{\Delta\boldsymbol{x}^T\Delta\boldsymbol{x}} \tag{12}$$

We then use Eq. (12) again in Eq. (8) for $\boldsymbol{g}_{k+1}-\boldsymbol{g}_k$. The recursive substitution yields

$$\boldsymbol{g}_{k+1}^{(N)}=\boldsymbol{g}_k+\left(\frac{2^{N+1}-1}{2^N}\right)\frac{(\Delta F-\boldsymbol{g}_k^T\Delta\boldsymbol{x})}{\Delta\boldsymbol{x}^T\Delta\boldsymbol{x}}\Delta\boldsymbol{x},\quad N=1,2,3,\ldots \tag{13}$$

where N indicates the number of substitutions. Obviously, when $N\to\infty$, the limit of Eq. (13) is exactly Eq. (10).

Received April 23, 1991; revision received July 1, 1991; accepted for publication July 2, 1991.

*Assistant Professor, Department of Aerospace Engineering and Engineering Mechanics, 304 Town Engineering. Member AIAA.

Next we shall show that Eq. (10) is indeed a more accurate approximation to the gradient at x_{k+1} than the Broyden update (5) if $\|\Delta x\|$ is small, although Eqs. (5) and (10) only differ in the coefficient. Suppose that g_k^* and g_{k+1}^* are the accurate gradients at x_k and x_{k+1}, respectively. Consider the expansion of

$$\Delta x^T g_{k+1}^* = \Delta x^T (g_k^* + H_k \Delta x + \ldots)$$
$$= \Delta x^T g_k^* + \Delta x^T H_k \Delta x + \mathcal{O}(\|\Delta x\|^2) \tag{14}$$

where $\mathcal{O}(\|\Delta x\|^2)$ denotes the term of order higher than $\|\Delta x\|^2$. The second-order Taylor series for F at x_k gives

$$\Delta F = \Delta x^T g_k^* + \tfrac{1}{2}\Delta x^T H_k \Delta x + \mathcal{O}(\|\Delta x\|^2) \tag{15}$$

Combining Eqs. (14) and (15) results in

$$\Delta x^T g_{k+1}^* = 2\Delta F - \Delta x^T g_k^* + \mathcal{O}(\|\Delta x\|^2) \tag{16}$$

On the basis of ΔF, Δx, g_k, and Eq. (16), we look for g_{k+1}, an approximation to g_{k+1}^*, such that

$$\Delta x^T g_{k+1} = \alpha \Delta F + \beta \Delta x^T g_k \tag{17}$$

Notice that g_{k+1} given by Eq. (10) satisfies Eq. (17) with $\alpha = 2$ and $\beta = -1$. So for g_{k+1} obtained from Eq. (10),

$$\Delta x^T (g_{k+1}^* - g_{k+1}) = \mathcal{O}(\|\Delta x\|^2) \tag{18}$$

In other words,

$$\|g_{k+1}^* - g_{k+1}\| = \mathcal{O}(\|\Delta x\|) \tag{19}$$

On the other hand, g_{k+1} obtained from the Broyden update meets Eq. (17) with $\alpha = 1$ and $\beta = 0$. This only leads to

$$\|g_{k+1}^* - g_{k+1}\| = \mathcal{O}(\|\Delta x\|^0) \tag{20}$$

Equations (19) and (20) indicate that the difference between the accurate gradient g_{k+1}^* and the estimate g_{k+1} by Eq. (10) is proportional to $\|\Delta x\|^2$, whereas the difference between g_{k+1}^* and g_{k+1} from the Broyden rank one update (5) is only proportional to $\|\Delta x\|$.

Similarly, the Jacobian $\nabla h(x_{k+1})$ can be approximated by

$$\nabla h_{k+1}^T = \nabla h_k^T + 2\frac{(\Delta h^T - \Delta x^T \nabla h_k^T)}{\Delta x^T \Delta x}\Delta x \tag{21}$$

The Jacobian of those active inequality constraints in Eq. (3) also has the similar approximation.

The following example demonstrates that Eq. (10) gives the true gradient in the case when $F(x)$ is a quadratic function, regardless of the size of $\|\Delta x\|$. Let

$$F = \tfrac{1}{2} x^T x$$

Suppose that $g_k^* = x_k$ is known. It is easy to show that using Eq. (10) for an estimate of the true gradient g_{k+1}^* at x_{k+1} produces

$$g_{k+1} = g_k^* + 2\frac{(\Delta F - g_k^{*T}\Delta x)}{\Delta x^T \Delta x}\Delta x = x_k + \Delta x = x_{k+1} = g_{k+1}^*$$

whereas the Broyden rank-one update (5) yields

$$g_{k+1} = \tfrac{1}{2}(x_k + x_{k+1}) = \tfrac{1}{2}(g_k^* + g_{k+1}^*)$$

The advantage of using these formulas is to reduce the number of integrations spent on finite differences. Nevertheless, Eqs. (10) and (21) should not completely replace finite difference steps. First for a number of iterations where x, $F(x)$, and constraints undergo relatively large changes, finite differences should be employed for the gradient calculations. Experience also shows that even after the first few iterations it is better to use Eqs. (10) and (21) and finite differences alternatively to prevent accumulation of errors in the gradients.

Depending on which nonlinear programming algorithm is used, ∇F and gradients for constraints may or may not be required separately. For instance, if the augmented Lagrangian method is used, only the gradient of the augmented Lagrangian is needed, which may again be approximated by Eq. (10).

A final remark is that this approximate gradient update may also be used in some other problems, such as an iterative scheme for the zero-finding problem of a nonlinear algebraic system.

Trajectory Optimization for an Advanced Launch System

The advanced launch system (ALS) is a proposed concept for the next generation of heavy-lift vehicles. A conceptual model ALS-L[4] is used in this paper. It has an asymmetric configuration with a liquid rocket booster (LRB) attached to the core vehicle. The LRB has seven low-cost engines (LCE), and the core vehicle has three identical ones. Each LCE has a vacuum thrust of 2,580,457 N and a specific impulse of 430 s. At ignition, all 10 LCEs will be started. At the burnout of the LRB, the LRB is jettisoned and the core vehicle continues the flight. The mission is to insert the payload into a 148.16 km × 277.8 km Earth orbit at the perigee. Because of the complexity and discontinuity of the system, the trajectory optimization is best handled by the nonlinear programming approach. Assuming a spherical nonrotating Earth, the point-mass equations of motion are

$$\dot{r} = v \sin\gamma$$
$$\dot{\theta} = \frac{v\cos\gamma}{r}$$
$$\dot{v} = \frac{T\cos(\alpha-\epsilon) - D}{m} - \frac{\mu\sin\gamma}{r^2}$$
$$\dot{\gamma} = \frac{T\sin(\alpha-\epsilon) + L}{mv} + \left(\frac{v}{r} - \frac{\mu}{vr^2}\right)\cos\gamma$$
$$\dot{m} = -\frac{T}{g_0 I_{sp}} \tag{22}$$

In Eq. (22), r is the radius from the center of the Earth to the ALS, θ the polar angle, v the velocity, γ the flight-path angle, m the mass, and T the vacuum thrust. We neglected the back pressure term. The angle of attack α and nozzle gimbal angle ϵ represent the controls. The aerodynamic forces are the drag D and lift L. The drag coefficient C_D and lift coefficient C_L are given in tabular forms as functions of Mach number and α. Other relevant data are the following: For the core vehicle, payload + fairing = 72,176 kg, inert mass = 79,891 kg, maximum fuel = 678,669 kg. For the LRB, inert mass = 98,375 kg, maximum fuel = 679,930 kg. The objective is to minimize the takeoff mass. Since the inert weights and payload (plus fairing) are specified, it amounts to finding the optimal propellant distribution between the LRB and the core vehicle and the optimal control histories $\alpha(t)$ and $\epsilon(t)$. Two control constraints are imposed:

$$|\alpha(t)| \le 14 \text{ deg}$$
$$|\epsilon(t)| \le 9 \text{ deg}$$

It turns out that these constraints are not active. So $\alpha(t)$ and $\epsilon(t)$ are parameterized by cubic spline functions for each

Table 1 Summary of test results

Gradient generation	$m(0)$, kg	Iterations	Integrations
Eqs. (10) and (21)	1,572,202	61	699
Finite differences	1,568,089	65	1209

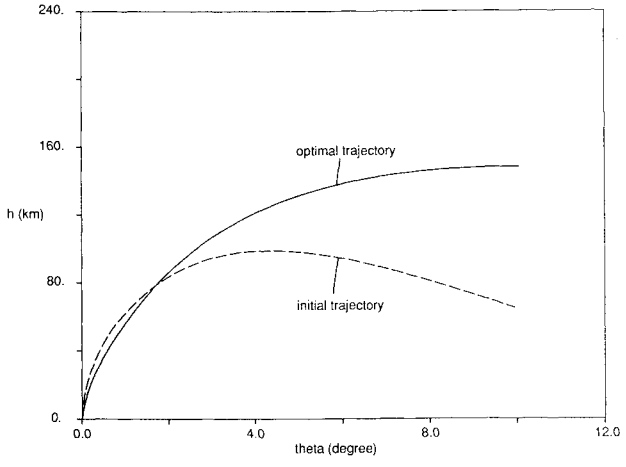

Fig. 1 Ascent trajectory: altitude vs downrange angle.

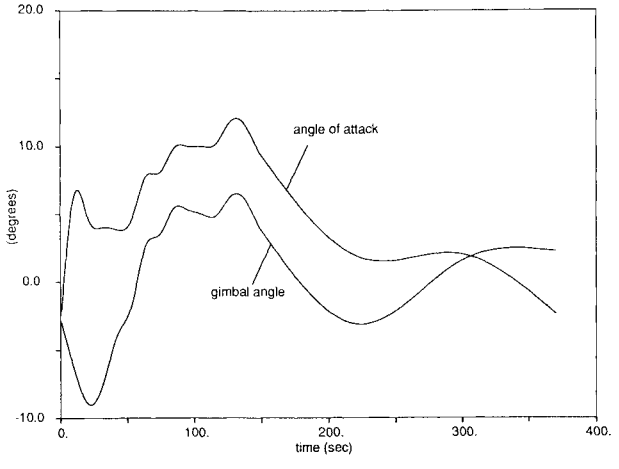

Fig. 2 Control histories: $\alpha(t)$ and $\epsilon(t)$.

stage of the flight. To demonstrate the point of using the approximate gradients, no inequality state-control constraints such as on dynamic pressure q and αq are imposed, although it is not necessary. The problem thus becomes a 20-parameter optimization problem with the equality constraints for orbital insertion

$$h_1 = [r(t_f) - r_0] - 148{,}160 \text{ m} = 0$$

$$h_2 = v(t_f) - 7855.1205 \text{ m/s} = 0$$

$$h_3 = \gamma(t_f) = 0 \tag{23}$$

The launch conditions are

$$r(0) = 6{,}376{,}400 \text{ m}$$

$$\theta(0) = 0$$

$$v(0) = 1 \text{ m/s}$$

$$\gamma(0) = 89.9 \text{ deg}$$

$$m(0) = \text{to be minimized} \tag{24}$$

A sequential quadratic programming (SQP) code[5] is used to solve the constrained parameter optimization problem. To test the approximate gradient formulas (10) and (21), after the first 10 iterations using finite differences, Eqs. (10) and (21) and finite differences are used alternatively in every other iteration for gradient update. The stopping criteria are either that the constraints are satisfied and $|\nabla L|$, the norm of the gradient of the Lagrangian

$$L = F + \sum_{i=1}^{3} \lambda_i h_i \tag{25}$$

is $< 10^{-5}$ or

$$\left| \frac{F_{k+1} - F_k}{F_k} \right| \leq 10^{-6} \tag{26}$$

$$\sqrt{\sum_{i=1}^{3} h_i^2} \leq 10^{-5} \tag{27}$$

are satisfied for two consecutive iterations. To compare the efficiency of the approximate gradient approach, the problem is also solved with all gradients generated by finite differences. Table 1 summarizes the comparison in terms of takeoff mass and computation efficiency.

From Table 1 it is clear that when Eqs. (10) and (21) are used, about 43% reduction in the number of integrations is achieved at the expense of approximately 4000 kg more takeoff mass. That is a 0.26% increase. As mentioned, when quick convergence is the prime concern, for all practical purposes this is a reasonable price to pay. Notice that the preceding results are obtained from an arbitrary initial guess. The number of iterations and integrations can be reduced dramatically in an onboard environment, for the preceding calculated trajectory can serve as a good initial solution.

It should be noted that the Broyden update (5) was also tried for the gradient update. The algorithm failed to converge due to inaccurate gradients.

Figure 1 shows the comparison of the optimal ascent history and the initial guessed history. The optimal control histories $\alpha(t)$ and $\epsilon(t)$ are depicted in Fig. 2.

Conclusion

Approximate gradient update formulas have been derived to enhance the efficiency of the nonlinear programming approach when used to solve the optimal control problem. They are shown to be more accurate than the well known Broyden rank one update. The application in trajectory optimization for an advanced launch system shows that the number of trajectory integrations is reduced by more than 40% with a slight increase of 0.26% in the performance index.

Acknowledgment

This research was supported by the University Research Grant of Iowa State University.

References

[1]Pontryagin, L. S., Boltyanskii, V. G., Gramkreledze, Q. V., and Mishchenko, E. F., *The Mathematical Theory of Optimal Processes,* Intersciences, New York, 1962.

[2]Gilbert, E. G., Howe, R. M., Lu, P., and Vinh, N. X., "Optimal Aeroassisted Intercept Trajectories At Hyperbolic Speeds," *Journal of Guidance, Control, and Dynamics,* Vol. 14, No. 1, 1991, pp. 123-131.

[3]Stoer, J., and Bulirsch, R., *Introduction to Numerical Analysis,* Springer-Verlag, New York, 1980.

[4]Pamadi, B. N., and Kevin, D., "An Aerodynamic Model for an ALS-L Vehicle," Proposed NASA TM, 1990.

[5]Pouliot, M. R., "CONOPT2: A Rapidly Convergent Constrained Trajectory Optimization Program for TRAJEX," General Dynamics, Convair Division, San Diego, CA, Rept. No. GDC-SP-82-008, 1982.