540:615 Nonlinear Programming
198:529 " " Algorithms

computational requirements, stopping criteria, and modifications for fixed end point problems. Table 6-4 summarizes these and other characteristics of the three iterative methods.

It should be emphasized that the numerical techniques we have discussed

Table 6-4 A COMPARISON OF THE FEATURES OF THREE ITERATIVE METHODS FOR SOLVING NONLINEAR TWO POINT BOUNDARY VALUE PROBLEMS

| *Feature* | *Steepest descent* | *Variation of extremals* | *Quasilinearization* |
|---|---|---|---|
| Initial guess | $\mathbf{u}(t),\ t \in [t_0, t_f]$ | $\mathbf{p}(t_0)$ [or $\mathbf{x}(t_f)$] | $\mathbf{x}(t), \mathbf{p}(t),\ t \in [t_0, t_f]$ |
| Iterate to satisfy | $\dfrac{\partial \mathscr{H}}{\partial \mathbf{u}} \equiv 0$ | $\mathbf{p}(t_f) = \dfrac{\partial h}{\partial \mathbf{x}}(\mathbf{x}(t_f))$ | State and costate equations |
| Importance of initial guess | Not usually crucial to convergence | Divergence may result from poor guess | Divergence may result from poor guess |
| Storage requirements | $\mathbf{u}^{(i)}(t), \mathbf{x}^{(i)}(t)$, and $\dfrac{\partial \mathscr{H}^{(i)}}{\partial \mathbf{u}}(t),\ t \in [t_0, t_f]$ | $2[n \times n]$ matrices, boundary conditions | $\mathbf{x}^{(i)}(t), \mathbf{p}^{(i)}(t),\ t \in [t_0, t_f]$, $n \times n$ matrix, boundary conditions, $\mathbf{c}$ |
| Convergence | Approaches a minimum rapidly, then slows down drastically | Once convergence begins (if it does), it is rapid | Converges quadratically in the vicinity of the optimum |
| Computations required | Integration of $2n$ differential equations, calculation of $\partial \mathscr{H}/\partial \mathbf{u}$, step size. | Integration of $2n(n + 1)$ first-order differential equations, inversion of an $n \times n$ matrix. | Integration of $2n(n + 1)$ first-order differential equations, inversion of an $n \times n$ matrix. |
| Modifications for fixed end point problems | Penalty function or see [B-5] | Adjust $\mathbf{p}(t_0)$ based on calculated values of $\mathbf{x}(t_f)$. | Solve for $\mathbf{c}$ from equation for $\mathbf{x}(t_f)$ |

may not always converge, and even if convergence occurs it may be only to a local minimum. By trying several different initial guesses, we can be reasonably sure of locating any other local minima that may exist, or, if the numerical procedure converges to the same control and trajectory for a variety of initial guesses, we have some assurance that a global minimum has been determined.

The difficulty of solving nonlinear two-point boundary-value problems has made iterative numerical techniques the subject of continuing research. When one is confronted with a problem of this type, it is useful to be familiar with many different techniques, perhaps trying several methods on a given problem, or a hybrid scheme may be useful. For example, the steepest descent method may be used as a starting procedure and quasilinearization to close in on the solution.

## 6.6 GRADIENT PROJECTION

In this section we shall discuss an alternative approach to optimization introduced by J. B. Rosen [R-4, 5, 6] which does not involve the solution of nonlinear two-point boundary-value problems. Rosen's method, called gradient projection, is an iterative numerical procedure for finding an extremum of a function of several variables that are required to satisfy various constraining relations. If the function to be extremized (called the *objective function*) and the constraints are linear functions of the variables, the optimization problem is referred to as a *linear programming problem*; when nonlinear terms are present in the constraining relations or in the objective function, the problem is referred to as a *nonlinear programming problem*.

We shall first discuss gradient projection as it applies to nonlinear programming problems that have linear constraints, but nonlinear objective functions. Then we shall show how the gradient projection algorithm can be used to solve optimal control problems.

### *Minimization of Functions by the Gradient Projection Method*

**Example 6.6-1.** To begin, let us consider a simple example. Let $f$ be a function of two variables $y_1$ and $y_2$ and $f(y_1, y_2)$ denote the value of $f$ at the point $(y_1, y_2)$. The problem is to find the point $(y_1^*, y_2^*)$ where $f$ has its minimum value. The variables $y_1$ and $y_2$ are required to satisfy the linear inequality constraints

$$y_1 \geq 0 \tag{6.6-1a}$$
$$y_2 \geq 0 \tag{6.6-1b}$$
$$2y_1 - 5y_2 + 10 \geq 0 \tag{6.6-1c}$$
$$-4y_1 - 7y_2 + 22.5 \geq 0 \tag{6.6-1d}$$
$$9y_1 - 2y_2 + 26.5 \geq 0 \tag{6.6-1e}$$

The set of points that satisfy all of these constraints is denoted by $R$ and called the *admissible region*.† For this example, $R$ is the interior and the boundary of the region whose boundary is determined by the lines labelled

† In the nomenclature of nonlinear programming the term *feasible* is used rather than *admissible*.

## 6.6 GRADIENT PROJECTION

85-962-11 ©

Now *Rosen's Technique* (Chap 6)
and *"Discrete-Time Dynamic Programming"* (Chap 3)
both start by: ***forming a discrete-time opt control problem.***

That is:

Represent time interval $[t_0, t_f]$
via $k = 0, 1, 2, \ldots, N$.

For example $t = t_0 \leftrightarrow k = 0$
$t = t_0 + \Delta t \leftrightarrow k = 1$
$\vdots$
$t = t_f = t_0 + N\Delta t \leftrightarrow k = N$

Approximate derivative and integral in
the state eqn and cost integral to obtain

$$\boxed{\begin{aligned} x(k+1) &= a_D\left(x(k), u(k), k\right) \\ J_D(u) &= h\left(x(N), N\right) + \Delta t \sum_{k=0}^{N-1} g\left(x(k), u(k), k\right) \end{aligned}}$$

COMMENTS:

1) Notice that again this uses $t_f$ fixed (prespecifie

2) The models may turn out simpler (linear, time-invariant, etc.) than the general case shown here.

3) Fixed $x(t_f)$ problems led to (chap 3) "back-to-front" sol
and "control law" tables [illegible]

85-962-11 Ⓓ

4) Fixed $x(t_0)$ problems could be done "front-to-back" to yield $u^*(k)$ tabular formula [Dyn Programming Chap 3].

5) Free $x(t_f)$ problems could be done as in 3) — just get larger grid.

6) Control Constraints $M_{i-} \leq u_i(t) \leq M_{i+}$
and
State Constraints $S_{i-} \leq x_i(t) \leq S_{i+}$
or
$x_i(t_j) = T_{ij}$ specified states

actually "helped" the computational solution via dynamic programming.

WITH ROSEN'S TECHNIQUE:

START WITH A GUESS of $u^*(k)$ which generates a state $x(k)$ on $[0, N]$.

LINEARIZE THE NONLINEAR STATE EQ ABOUT the known control-state history.

SINCE IT'S LINEAR, $x(k)$ can be expressed as functions of $x(0)$ and $u(i)$, $0 \leq i \leq k$.

SINCE $x(0)$ is given, this makes $J_D(u)$ a function

85-962-11 Ⓔ

The problem of minimizing $J_D$ subject to state & control constraints now becomes:

nonlinear objective function

subject to inequality & equality constraints.

Thus, it becomes a "Nonlinear Programming" Problem for which many routines are available

if constraints linear (which they are here) and objective function is nice (linear, quadratic, etc).

Rosen's is just one such procedure.