A Survey of Methods of Feasible Directions for the Solution of Optimal Control Problems

ELIJAH POLAK, MEMBER, IEEE

Abstract—Although the class of methods of feasible directions, which can be used for solving constrained optimization problems in $\mathbb{R}^n$, is quite large, only a few of these methods can be extended for the solution of optimal control problems. This paper reviews three of the most promising algorithms in this class: an extension of the Frank–Wolfe method due to Barnes, a dual method due to Pironneau and Polak, and a method due to Zoutendijk.

I. INTRODUCTION

THE CLASS of nonlinear programming algorithms known as methods of feasible directions, or as modified methods of centers, is quite large. All the algorithms in this class apply to discrete optimal control problems. In this paper we shall review three of the most promising methods for feasible directions for optimal control: an extension of the Frank–Wolfe method [5], which is a composite of algorithms proposed by Demyanov [4], Levitin and Polyak [7], Barnes [2], and Armijo [1], a dual method of feasible directions devised by Pironneau and Polak [8], and a Zoutendijk method [13].

From the point of view of feasible directions algorithms, continuous optimal control problems must be divided into four categories: 1) fixed time problems with fixed initial state, free terminal state, and simple constraints on the control; 2) fixed time problems with inequality constraints on both the initial and terminal state and no control constraints; 3) free time problems with inequality constraints on the initial and terminal states and simple constraints on the control; and, finally, 4) fixed time problems with inequality state-space constraints and constraints on the control.

We shall show that the earlier mentioned extension of the Frank–Wolfe method can be used for solving problems in category 1), that the Pironneau–Polak method can be used for solving problems in category 2), and that the Zoutendijk method can be used for solving discretized problems in category 4). The Pironneau–Polak method can also be used for solving problems in category 3). However, this requires a new modification of the method, based on a Valentine-type transformation. The interested reader will find the details of this in [10].

II. THE NONLINEAR PROGRAMMING ALGORITHMS

The three nonlinear programming algorithms, which are going to be adapted for the solution of optimal control problems, were originally intended to solve problems of the form

$$
\min \{f^0(\mathbf{z})| f^j(\mathbf{z}) \leq 0, j = 1, 2, \ldots, m \} \quad (2.1)
$$

where the $f^j: \mathbb{R}^n \rightarrow \mathbb{R}, j = 0, 1, \ldots, m$, are continuously differentiable.

We begin with a modification of the Frank–Wolfe method [5], which can be used only when the set

$$
\Omega = \{z | f^j(z) \leq 0, j = 1, 2, \ldots, m, \}
$$

is convex. The modification of the Frank–Wolfe algorithm below combines a direction finding subroutine proposed by Levitin and Polyak [5] and by Barnes [2], with an efficient step length subroutine due to Armijo [1]. Such “hybrids” are quite common in nonlinear programming.

Algorithm 2.3 (Modification of Frank–Wolfe Method)

- Step 0: Select a continuous, symmetric, positive semidefinite $n \times n$ matrix $D(\mathbf{z})$, an $\alpha \in (0,1)$, and a $\beta \in (0,1)$.
- Step 1: Compute a starting point $\mathbf{z}_0 \in \Omega$, as explained in (2.7), below, and set $i = 0$.
- Step 2: Compute a point $\mathbf{z}_i$ as a solution of the problem

$$
\min \{\langle \nabla f^0(\mathbf{z}_i), \mathbf{x} - \mathbf{z}_i \rangle + \langle \mathbf{z} - \mathbf{z}_i, D(\mathbf{z}_i)(\mathbf{z} - \mathbf{z}_i) \rangle \mid f^j(\mathbf{z}_i) \leq 0, j = 1, 2, \ldots, m \} \quad (2.4)
$$

and set $d(\mathbf{z}_i) = \mathbf{z}_i - \mathbf{z}_i$.
- Step 3: If $\mathbf{d}(\mathbf{z}_i) = 0$, stop; else, compute the smallest integer $k(\mathbf{z}_i) \geq 0$ such that

$$
f(\mathbf{z}_i + \beta^{k(\mathbf{z}_i)}d(\mathbf{z}_i)) - f^0(\mathbf{z}_i) - \beta^{k(\mathbf{z}_i)}\alpha(\nabla f^0(\mathbf{z}_i)d(\mathbf{z}_i)) \leq 0.
$$

The Frank–Wolfe method and its extensions belong to the class of feasible directions algorithms.

1 The choice $D(\mathbf{z}) = 0$ was used by Frank and Wolfe and results in slow convergence, proportional to $1/k$. $D(x) > 0$ can sometimes be chosen to obtain a linear rate of convergence, see [2]. Note that the algorithms we are about to state involve various parameters that must be preselected. We shall indicate a first choice for these parameters. However, this choice may not always be the best and the reader is encouraged to experiment a little.

2 Note that the need to solve (2.4) restricts this method to problems in which the $j_i = 1, 2, \ldots, m$, are affine, unless $D(\mathbf{z}) = 0$, in which case a single ($m = 1$) quadratic constraint can be accommodated.
Step 4: Set $z_{i+1} = z_i + \beta^k(\alpha_i) d(z_i)$.

Step 5: Set $i = i + 1$ and go to Step 2.

The function $d^p(\cdot)$ used in Algorithm (2.3) has the following properties. 1) $d^p(z) \leq 0$ for all $z \in \Omega$. 2) Supposing that $\Omega$ is convex and that $z_i \in \Omega$ is optimal for (2.1), then $d^p(z_i) = 0$ (i.e., $d^p(\Omega) = 0$ is a necessary condition of optimality). This result can be established by reasoning similar to that in [9, sec. 4.4.]. 3) When the set $\Omega$ satisfies the Kuhn-Tucker constraint qualification, its convergence properties can be summed up as follows (see [9, sec. 4.3]).

Theorem 2.8: Suppose that $\Omega$ is convex and compact, and that the sequence $\{x_i\}$ is constructed by Algorithm (2.3).

If $\{x_i\}$ is finite, then its last element $x_i$ satisfies $d^p(x_i) = 0$. If $\{x_i\}$ is infinite, then every accumulation point $\bar{x}$ of $\{x_i\}$ satisfies $d^p(\bar{x}) = 0$.

Remark 2.7: Algorithm (2.3) requires a starting point $x_0 \in \Omega$. Such a point can be computed by applying Algorithm (2.3) to the problem, in $\alpha^{k+1}$,

$$\min \{y^T f(y) - y^0 \leq 0, j = 1, 2, \cdots, m \}.$$  \hspace{1cm} (2.7')

A starting point $(y^0, y_0)$ for solving (2.7) is obtained by taking $y_0$ to be a good guess and then setting $y^0 = \max f(y)$. When the set $\{x_i\}$ is not empty, after a finite number of iterations, the algorithm will construct a $(y^0, y_0)$ such that $f(y) \leq 0, j = 1, 2, \cdots, m$, at which point we set $y_0 = y$. This is so since the optimal $y_0$ is strictly negative.

For the sake of saving space and so as to exhibit their common features, we state the following two algorithms as one, with a parameter $p$. When $p = 1$, the algorithm becomes a composite using the Zoutendijk procedure 1 direction finding subroutine [13] and the Armijo step size subroutine [1]. When $p = 2$, the algorithm becomes the Pironneau-Polak modified method of centers [8].

These two algorithms differ both in their direction finding and step length subroutines. Both of these algorithms require that the set $\Omega = \{x \in f(x) < 0, j = 1, 2, \cdots, m \}$ be nonempty; otherwise they jam up. Convexity of $\Omega$ is not required.

Algorithm 2.8 (Zoutendijk Method of Feasible Directions and Pironneau-Polak Modified Method of Centers)

Step 0: Select parameters $\lambda > 0$, $\epsilon' \in (0, e_0], \alpha \in (0, 1), \beta \in (0, 1), \gamma > 0$. Set $p = 1$ to obtain Zoutendijk procedure 1 type method of feasible directions; set $p = 2$ to obtain Pironneau-Polak modified method of centers. (It is difficult to recommend values for $e_0$ and $\gamma$, but try $e_0 = 0.1, \gamma = 0.1$; $\epsilon'$ is a precision parameter, try $\epsilon' = 10^{-4}$; try $\alpha = 0.25, \beta = 0.7$; try $\lambda = 2$ or $\lambda = 1$.)

Step 1: Compute a $z_0 \in \Omega$ by applying (2.8) to (2.7'), and set $i = 0, \epsilon = e_0$.

Step 2: Set

$$I(z_i, e) = \{j \in [1, 2, \cdots, m] \mid f(z_i) \geq -\epsilon \}$$  \hspace{1cm} (2.9)

$$J(z_i, e) = I(z_i, e) \cup \{0\}$$  \hspace{1cm} (2.10)

and go to Step 3.\(p = 1\) or 2).

Step 3.1: Compute $(d^p(z_i, e_i), d(z_i, e_i))$, where $d^p(z_i, e) \in \Omega^1$, $d(z_i, e) \in \Omega^*$, as a solution of the linear program

$$\phi(z_i, e) \triangleq \min \{d^p(\nabla f(z_i), d) - d^0 \leq 0,$$

$$j \in J(z_i, e), \mid d_j \leq 1, l = 1, 2, \cdots, n \}.$$  \hspace{1cm} (2.11)

and go to Step 4.

Step 3.2: Compute $(d^p(z_i, e_i), d(z_i, e_i))$, where $d^p(z_i, e) \in \Omega^1$, $d(z_i, e) \in \Omega^*$, as a solution of the quadratic program

$$\phi(z_i, e) = \min \{d^0 + \frac{1}{2} \|d\|^2 |\nabla f(z_i), d - d^0 \leq 0,$$

$$f(z_i) + \nabla f(z_i) d - d^0 \leq 0, j \in I(z_i, e) \}$$  \hspace{1cm} (2.12)

and go to Step 4.

Step 4: If $\phi(z_i, e) \leq -\gamma e^*$, go to Step 6; else, go to Step 5.

Step 5: If $\epsilon \leq \epsilon'$, stop; else, set $\epsilon = \beta \epsilon$ and go to Step 2.

Step 6.1: Compute the smallest integer $k(z_i, e) \geq 0$ such that

$$f(z_i + \lambda \beta^{k(z_i, e)} d(z_i, e)) \leq 0, \quad j = 1, 2, \cdots, m$$

and

$$f(z_i + \lambda \beta^{k(z_i, e)} d(z_i, e)) - f(z_i)$$

$$= \lambda \beta^{k(z_i, e)} \alpha \nabla f(z_i), d(z_i, e) \leq 0 \quad (2.13)$$

and go to Step 7.

Step 6.2: Compute the smallest integer $k(z_i, e) \geq 0$ such that

$$\max \{f(z_i + \lambda \beta^{k(z_i, e)} d(z_i, e)) - f(z_i)$$

$$= \lambda \beta^{k(z_i, e)} \alpha \nabla f(z_i), d(z_i, e) \leq 0 \}$$  \hspace{1cm} (2.14)

and go to Step 7.

Step 7: Set $z_{i+1} = z_i + \lambda \beta^{k(z_i, e)} d(z_i, e)$.

Step 8: Set $i = i + 1$ and go to Step 2.

Remark: Suppose that $z_i \in \Omega$ and that $\phi(z_0, 0)$ is defined by (2.11) or by (2.12). Then $\phi(z_0, 0) \leq 0$ and $\phi(z_0, 0) = 0$ if and only if there exist multipliers $\mu^0 \geq 0, \mu^i \geq 0, \cdots, \mu^m \geq 0, \sum_{i=0}^{m} \mu^i \nabla f(z_i) = 0, \mu^l j(z_i) = 0, j = 1, 2, \cdots, m, \sum_{i=0}^{m} \mu^i = 1$, i.e., $\phi(z_0, 0) = 0$ if and only if $z_i$ satisfies the John optimality condition [4] (see [9, sec. 4.3]).

The convergence properties of the two algorithms defined by (2.12) (which can be used even when the set $\{x \in f(x) < 0, j = 1, 2, \cdots, m \}$ is not convex, provided the set $\{x \in f(x) < 0, j = 1, 2, \cdots, m \}$ is nonempty) are stated as follows.

4 Relation (2.14) defines a step size subroutine of the "centres" type. It keeps the iterates $z_i$ in the interior of $\Omega$, a feature which is useful in optimal control when coarse integration is used in the early iterations. Step size rule (2.12) can also be used with $p = 2$, if preferred.
**Theorem 2.15:** Suppose that the set \(|z|f'(z) < 0, j = 1,2,\ldots, m|\) is nonempty. Then, for \(p = 1\) or \(p = 2\), and \(\varepsilon = 0\), Algorithm (2.8) either jams up at a point \(z_0\), in which case \(\phi(z_0) = 0\), or it constructs an infinite sequence \(\{z_j\}_{j=0}^{\infty}\) such that every accumulation point \(\xi\) of \(\{z_j\}_{j=0}^{\infty}\) satisfies \(\phi(\xi,0) = 0\). (See [13] for \(p = 1\) and [8] for \(p = 2\)).

Note that a sequence \(\{z_j\}_{j=0}^{\infty}\) constructed by (2.8) will always have accumulation points when the set \(\Omega(z_0) \neq \emptyset\) and \(f'(z_0) < 0, j = 1,2,\ldots, m|\) is bounded. Note also that, in using an algorithm such as (2.8) or (2.9), there is no need to extract a convergent subsequence of \(\{z_j\}_{j=0}^{\infty}\), since once the sequence \(\{z_j\}_{j=0}^{\infty}\) converges to the set \(\{z \in \Omega(z_0)\mid \phi(z,0) = 0\}\), \(\phi(z_0) = 0\), where \(K\) is the set of all positive integers.

### III. The Optimal Control Problems

For the purpose of applying the algorithms in Section II, we must state our optimal control problems in a form similar to (2.1). Thus, suppose that \(t_0 < t_1\) are given and that \(L^a[t_0,t_1]\) is the Banach space of equivalence classes of essentially bounded, measurable functions from \([t_0,t_1]\) into \(\mathbb{R}^a\), with norm

\[
\|u\|_a = \sup_{t \in [t_0,t_1]} \|u(t)\|
\]

where \(|\cdot|\) denotes the Euclidean norm. Suppose that \(h^a): \mathbb{R}^a \times \mathbb{R}^a \times \mathbb{R}^a \rightarrow \mathbb{R}^a\) and \(h): \mathbb{R}^e \times \mathbb{R}^a \times \mathbb{R}^a \rightarrow \mathbb{R}^e\) are continuously differentiable functions. Then we define \(f^a): \mathbb{R}^a \times L^a[t_0,t_1] \rightarrow \mathbb{R}^a\) by

\[
f^a(\xi,u) = \int_{t_0}^{t_1} h^a(x(t,\xi,u)(t),t) dt + \psi(x(t,\xi,u)) \tag{3.1}
\]

where \(x(t,\xi,u), t \in [t_0,t_1]\), is the solution of the differential equation

\[
\frac{d}{dt} x = h(x,u,t), \quad t \in [t_0,t_1] \tag{3.2}
\]

with \(x(t_0) = \xi\).

Next, let \(g^j): \mathbb{R}^a \rightarrow \mathbb{R}^j, j = 1,2,\ldots, m|\), and \(g^j): \mathbb{R}^e \rightarrow \mathbb{R}^j, j = 1,2,\ldots, m|\), be continuously differentiable functions. For \(j = 1,2,\ldots, m\), \(m = m_0 + m_1\), we define \(f^j): \mathbb{R}^j \times L^j[t_0,t_1] \rightarrow \mathbb{R}^j\) as follows:

\[
f^j(\xi,u) = g^j(\xi), \quad j = 1,2,\ldots, m_0 \tag{3.3}
\]

\[
f^{j+m_0}(\xi,u) = g^j(x(t,\xi,u)), \quad j = 1,2,\ldots, m_1. \tag{3.4}
\]

With these definitions we shall be able to solve the Problems P1, P2, and P3 below.

**Problem P1:**

\[
\text{min} \{f^a(\xi,u) \mid \xi = \xi_0, u(t) \in U \subset \mathbb{R}^a, t \in [t_0,t_1]\} \tag{3.5}
\]

with

\[
U = \{v \in \mathbb{R}^a \mid a^i \leq v^i \leq b^i, i = 1,2,\ldots, q\} \tag{3.6}
\]

or with

\[
U = \{v \in \mathbb{R}^a \mid \|v\| \leq 1\}. \tag{3.7}
\]

**Problem P2:**

\[
\text{min} \{f^a(\xi,u) \mid f^a(\xi,u) \leq 0, j = 1,2,\ldots, m\}. \tag{3.8}
\]

**Problem P3:**

\[
\text{min} \{f^a(\xi,u) \mid f^a(\xi,u) \leq 0, j = 1,2,\ldots, m\}. \tag{3.8}
\]

For Problem P2 we assume that either the set \(\{\xi_0\} \cup \mathbb{R}^e \rightarrow \mathbb{R}^j, j = 1,2,\ldots, m|\) is not empty (\(g^a_1, g^a_2,\ldots, g^a_m\), \(g^a_1, g^a_2,\ldots, g^a_m\)), or that \(\|\xi_0\| \leq L\), and the set \(\{u \mid g^a(x(t,\xi,u)) = 0\}\) is nonempty.

Finally, let \(N > 0\) be an integer and let \(\Delta = (t_f - t_0)/N\). For \(i = 0,1,2,\ldots, N\), let \(g_l^j: \mathbb{R}^j \rightarrow \mathbb{R}^j, j = 1,2,\ldots, m|\), be continuously differentiable. Then the discrete optimal control problem that we can solve is the following.

**Problem P3:**

\[
\text{min} \{f^a(\xi,u) \mid f^a(\xi,u) \leq 0, j = 1,2,\ldots, m, v_i \in U \subset \mathbb{R}^j\} \tag{3.9}
\]

with \(U\) as in (3.6) or (3.7), \(f^a\) as in (3.1) and with

\[
\pi(t) = 1, \quad t \in [0,\Delta]
\]

or

\[
\pi(t) = 0, \quad \text{otherwise.} \tag{3.10}
\]

For Problem P3, with \(U\) as in (3.6), we must assume that \(\{\xi_0, v_0, v_1,\ldots, v_{N-1}\} \cup \mathbb{R}^e \rightarrow \mathbb{R}^j, j = 1,2,\ldots, m|\), \(v_i \in U\) is nonempty or that the set \(\{\xi_0\} \cup \mathbb{R}^e \rightarrow \mathbb{R}^j, j = 1,2,\ldots, m|\), \(v_i \in U\) is not empty. When \(U\) is as in (3.7), replace \(U\) by \(\int U\) in the preceding conditions.

Note that the discretization in Problem P3 is only of the control and not of the differential equation (3.2). However, while Problem P1 and Problem P2 are problems on the infinite dimensional space \(\mathbb{R}^a \times L^a[t_0,t_1]\), Problem P3 is defined on the finite dimensional space \(\mathbb{R}^a \times \mathbb{R}^e \rightarrow \mathbb{R}^j (= \mathbb{R}^{j+k})\), and is obviously of the form of the problem (2.1), with \(n = s + N_2\) and \(m\) determined by the number of the \(g^j\) and the description of \(U\).

The algorithms in Section II make use of gradients and scalar products. For the problems stated in this section, we use the scalar product \(\langle \cdot, \cdot \rangle\) on \(\mathbb{R}^a \times L^a[t_0,t_1]\), defined by

\[
\langle \langle \xi_1, u_1 \rangle, \langle \xi_2, u_2 \rangle \rangle = \langle \xi_1, \xi_2 \rangle + \int_{t_0}^{t_1} \langle u_1(t), u_2(t) \rangle dt \tag{3.11}
\]

where, as before, \(\langle \cdot, \cdot \rangle\) denotes the Euclidean scalar product. The derivation of the gradients below can be found in [9 sec. 2.3]; here we shall merely state the formulas for their computation. These gradients have the same properties with respect to linear expansions as gradients in \(\mathbb{R}^a\). Note that, as defined below, \(\nabla f^a(\xi,u), j = 0,1,\ldots, m|\), is an element (a pair) in \(\mathbb{R}^a \times L^a[t_0,t_1]\). The first part of the pair is the gradient with respect to the initial state, while the second part is the gradient with respect to the control. Thus, for Problems P1 and P2,
\[ \nabla f^j(\xi, u) = (\nabla f^j(\xi, u), \nabla f^j(\xi, u)(\cdot)) = (-p_0(0, \xi, u), \]
\[ - \left[ \frac{\partial h}{\partial u} (x(t, \xi, u), u(t), \cdot) \right]^T p_0(t, \xi, u) + \left[ \frac{\partial h^\nu}{\partial u} (x(t, \xi, u), u(t), \cdot) \right]^T \]  (3.12)

(i.e., it is a pair consisting of a vector in \( \mathbb{R}^n \) and of a vector valued function in \( L_\infty([t_0, t_f]) \), where \( p_0(-, \xi, u) \) is defined by
\[ \frac{d}{dt} p_0(t, \xi, u) = - \left[ \frac{\partial h}{\partial x} (x(t, \xi, u), u(t), \cdot) \right]^T p_0(t, \xi, u) + \left[ \frac{\partial h}{\partial u} (x(t, \xi, u), u(t), \cdot) \right]^T, \quad t \in [t_0, t_f], \]  (3.13)
\[ p_0(t_f, \xi, u) = - \left[ \frac{\partial f^j}{\partial x} (x(t, \xi, u), \cdot) \right]^T. \]  (3.14)

Also for Problems P1 and P2 and \( j = 1, 2, \cdots, m \),
\[ \nabla f^j(\xi, u) = (\nabla f^j(\xi, u), \nabla f^j(\xi, u)(\cdot)) = \begin{pmatrix} \frac{\partial f^j}{\partial \xi}(\xi, u), 0 \end{pmatrix}, \quad j = 1, 2, \cdots, m, \]  (3.15)

where, for \( j = 1, 2, \cdots, m_f \), the \( p_j(-, \xi, u) \) are defined by
\[ \frac{d}{dt} p_j(t, \xi, u) = - \left[ \frac{\partial h}{\partial x} (x(t, \xi, u), u(t), \cdot) \right]^T p_j(t, \xi, u), \quad t \in [t_0, t_f], \]  (3.16)
\[ p_j(t_f, \xi, u) = - \left[ \frac{\partial f^j}{\partial x}(x(t, \xi, u), \cdot) \right]^T. \]  (3.17)

Thus, the discretization in Problem P3 does not remove the need for integrating differential equations. Its main advantage is that it results in a problem that we can solve, at least in principle, by Algorithm (2.8), with \( p = 1 \), whereas we do not know how to solve continuous time problems with control and state-space constraints by means of feasible directions algorithms.

## IV. EXTENSION OF NONLINEAR PROGRAMMING ALGORITHMS

We shall now show how to apply Algorithm (2.3) to the Problem P1, Algorithm (2.8) with \( p = 2 \), to Problem P2, and Algorithm (2.8), \( p = 1 \), to Problem P3.

**Problem P1 and the Modified Frank-Wolfe Method (2.3)**

Thus, consider Problem P1 and suppose that we have a control \( u_i(\cdot) \) such that \( u_i(t) \in U \) for \( t \in [t_0, t_f] \), where \( U \) is as in (3.6) or as in (3.7). To compute \( u_{i+1}(\cdot) \) according to Algorithm (2.3) we must first solve (2.4), where we associate \( z_i \) with \( u_i(\cdot) \). Following Barnes [21], for improved rate of convergence, we set \( D(u_i(t)) = \frac{\partial h}{\partial x}(x(t, \xi, u), u(t), \cdot) \), \( t \in [t_0, t_f] \), if the matrix is positive semidefinite; otherwise, to avoid singular subproblems, we set \( D(u_i(t)) = I \). If we calculate \( V_f^0(\xi, u) \) by linearization, rather than by formula (3.12), we find that (2.4) is equivalent to the quadratic cost optimal control problem
\[ d^f(u_i) = \min \left\{ \int_{t_0}^{t_f} \left[ \frac{\partial h}{\partial x}(x(t, \xi, u), u_i(t), \cdot) \right]^2 \right. \]  (4.1)
We solve (4.1) by means of the Pontryagin maximum principle [11] and denote the optimal control for (4.1) by $\delta u_i(\cdot)$. Next, we must compute the step size $\beta^{(i)}(u_i)$ as given by (2.5), which in this case becomes, because of (3.1) and (3.12),

$$
\int_0^{t_f} [\ell'(x(t, z, u_i) + \beta^{(i)}(u_i))_p u_i(t) + \beta^{(i)}(u_i) \delta u_i(t), t) - h(x(t, z, u_i))_p u_i(t) - \beta^{(i)}(u_i) \delta u_i(t)] dt \leq 0. \tag{4.2}
$$

Note that in solving (4.1) we have also computed $\nabla_f^0 (x, z, u_i) = -[\partial h/\partial u](x(t, z, u_i), u_i(t))_p u_i(t, z, u_i) + [\partial h/\partial u](x(t, z, u_i), u_i(t))_p u_i(t)$, since the adjoint equations for (4.1) coincide with (3.13), (3.14). The next control $u_{i+1}(\cdot)$ is then computed according to $u(t) = u(t) + \beta^{(i)}(u_i) \delta u_i(t)$, $t \in [t_0, t_f]$.

Now that we have computed the step size $\beta^{(i)}(u_i)$, we may we have to integrate the system (3.2) (with $\xi = \xi_0$) several times, once for each trial value of $k \geq 0$ that we wish to test for the condition in (4.2).

**Problem P2 and the Pironneau–Polak Algorithm (2.8), with $p = 2$**

Next, let us turn to Problem P2 for which we now adapt Algorithm (2.8) with $p = 2$. For this purpose, we must find a way for solving (2.12), with the gradients and scalar products as defined in Section III. This task is made easy by the fact that (2.12) has a convenient dual (see [5]), so that $\phi(z_t, \xi)$ and $d(z_t, \xi)$ can also be computed by solving the dual quadratic program

$$
\phi(z_t, \xi) = \max \left\{ \sum_{j \in J(z_t, \xi)} \mu^j f^j(z_t) - \frac{1}{2} \left\| \sum_{j \in J(z_t, \xi)} \mu^j \nabla f^j(z_t) \right\|^2 \right\}, \tag{4.3}
$$

where the $\mu^j, j \in J(z_t, \xi)$, solve (4.3). The importance of (4.3) is that the dimension of this problem depends only on the number of e-active constraints and not on the dimension of $z_t$. Consequently, even in the case of the infinite dimensional Problem P2, (4.3) remains a finite dimensional quadratic program. To be specific, given a feasible initial state $z_t$ and a feasible control $u_t(\cdot)$, which we associate with $z_t$ in (4.3), (4.4) according to $z_t = (x_0, u_t(\cdot))$, (4.3) becomes

$$
\phi(z_t, \xi) = \max \left\{ \sum_{j \in J(z_t, \xi)} \mu^j f^j(z_t, u_t) - \frac{1}{2} \left\| \sum_{j \in J(z_t, \xi)} \mu^j \nabla f^j(z_t, u_t) \right\|^2 \right\}
$$

where $f^j, j = 0, 1, 2, \ldots, m$, are defined as in (3.1), (3.3), and (3.4), $I(z_t) = \{ j \in \{1, 2, \ldots, m \} \mid f^j(z_t, u_t) \geq -\varepsilon \} = \{ l_1, l_2, \ldots, l_r \}, r \leq m, \mu = (\mu^{l_1}, \mu^{l_2}, \ldots, \mu^{l_r})^T$, $F_{l, j}(z, u_t)$ is a matrix with columns $\nabla f^j(z, u_t)$, $j \in J(z_t, \xi)$, the columns being ordered in the same way as the components of $\mu$, and $F_{l, j}(z, u_t)(t)$ is a matrix valued function of $t$, the columns of $F_{l, j}(z, u_t)(t)$ being $\nabla f^j(z, u_t)(t)$, $j \in J(z_t, \xi)$.

Thus, to use Algorithm (2.8) with $p = 2$, at each iteration we begin with a feasible pair $(z_t, \xi)$ and an $\varepsilon > 0$. Then we carry out the following operations.

1. We evaluate the functions $f^j(z_t, u_t), j = 0, 1, \ldots, m$.
2. We construct the index sets $J(z_t, \xi)$ and $J(z_t, \xi)$.
3. We compute the gradient $\nabla f^j(z_t, u_t), j \in J(z_t, \xi)$, according to (3.12)–(3.17).
4. We compute the coefficients of the quadratic form in (4.5).
5. We solve (4.5) [the dual of (2.12)] by a method such as Wolfe’s [12] to obtain a vector $\mu = (\mu^{l_1}, \mu^{l_2}, \ldots, \mu^{l_r})$ and $\phi(z_t, \xi)$.
6. We set

$$
\delta z_t = -\sum_{j \in J(z_t, \xi)} \mu^j \nabla f^j(z_t, u_t), \tag{4.5'}
$$

and

$$
\delta u_t(t) = -\sum_{j \in J(z_t, \xi)} \mu^j \nabla f^j(z_t, u_t)(t); \tag{4.5''}
$$

so that we associate $d(z_t, \xi)$ with the pair $(\delta z_t, \delta u_t(\cdot))$.

7. We then go through the tests in Steps 4 and 5 of Algorithm (2.8) until we reach Step 6.2, where we calculate the smallest integer $k(z_t, \xi)$ such that [see (2.14)]

$$
\max \left\{ \int_0^{t_f} \left[ h(x(z(t, \xi) + \lambda \delta^{k(z_t, \xi)}(\xi))_p u_t(t) + \lambda \delta^{k(z_t, \xi)}(\xi) \delta u_t(t) \right] dt; \right. \tag{4.6}
$$

where $\delta^{k(z_t, \xi)}(\xi), j = 1, 2, \ldots, m$.

8. (Step 7.) We set $z_{i+1} = z_t + \lambda \delta^{k(z_t, \xi)}(\xi)_p u_t + \lambda \delta^{k(z_t, \xi)}(\xi) \delta u_t$ and continue, with $i + 1$ replacing $i$ in all expressions.

**Problem P3 and the Zoutendijk-Type Algorithm (2.8) with $p = 1$**

Apart from a cumbersome evaluation of functions and derivatives, formulas for which were given in the preceding section, the application of Algorithm (2.8), with $p = 1$, to Problem P3 is straightforward, once the identification $z = (x_0, v_1, \ldots, v_{m-1})$ is made. We shall therefore elaborate no further.

Convergence

In the case of the optimal control Problems P1 and P2, the condition $d(\bar{z}) = 0$ (where $\bar{z} = u$, (4.5)) which the modified Frank–Wolfe method attempts to satisfy, and the condition $\phi(\bar{z}, 0) = 0$ (where $\bar{z} = (\bar{x}, \bar{u})$), which the Pironneau–Polak method tries to satisfy, can both be shown to be equivalent.
to the Pontryagin maximum principle in differential form, i.e., they imply the existence of an adjoint vector \( p(\cdot) \), satisfying the Pontryagin transversality conditions, such that for some \( p^0 \leq 0 \), \( \partial/\partial u(p^0 h(x(t),u(t),t) + \langle p(t), h(x(t),u(t),t) \rangle, \delta u) \leq 0 \) for all \( \delta u \in U \) (\( U = \mathbb{R}^n \) for Problem P2).

We can now summarize the convergence properties of the Algorithms (2.3) and (2.8) with respect to the Problems P1, P2, and P3. We find that these are slightly better than a direct extension of Theorems (2.6) and (2.15) would indicate. Thus the following.

1) Suppose that the sequence \{\( u_i \)\} constructed by Algorithm (2.3) in solving Problem P1 remains bounded (i.e., there is an \( M \in (0, \infty) \) such that \( \|u_i(t)\| < M \) for \( i = 0,1,2,\ldots \), and all \( t \in [t_0,t_f] \)). Then \( u_i(t) = 0 \) for all accumulation points \( \hat{u} = \hat{u}(\cdot) \) of \{\( u_i(\cdot) \}\}, where we may take \( \hat{u}(\cdot) \) to be an accumulation point of \{\( u_i \)\} in the \( L_\infty \cap L_2 \) sense (i.e., \( \|u_i(t)\| \leq M \), for some \( M < \infty \), \( i = 0,1,2,\ldots \), \( t \in [t_0,t_f] \), and for some infinite subset \( k \in \{0,1,2,\ldots \} \)).

\[ \lim_{i \to \infty} \int_{t_0}^{t_f} \|u_i(t) - \hat{u}(t)\|^p dt = 0, \]

which is somewhat more general than an accumulation point in \( L_\infty [t_0,t_f] \).

2) Assuming that the sequences \{\( \xi_i \)\} and \{\( u_i \)\} constructed by Algorithm (2.8), with \( p = 2 \), remain bounded (as in 1 above), \( \phi(\xi,0) = 0 \) for all accumulation points \( \hat{\xi} = \hat{\xi}(\cdot) \) of the sequence \{\( (\xi_i,u_i) \}\}, where we may construe \( \hat{\xi}(\cdot) \) to be an accumulation point of \{\( u_i \)\} in the \( L_\infty \cap L_2 \) sense.

3) When Algorithm (2.8), with \( p = 1 \), is applied to Problem P3, Theorem (2.15) remains valid without qualifications.

CONCLUSION

We have shown that certain methods of feasible directions can be extended for use in optimal control. It is to be remembered that in using methods of feasible directions in optimal control, the major cost is in the many integrations required per iteration. This cost can be reduced substantially by integrating coarsely when far from a solution and by refining the precision of integration adaptively as a solution is approached. The reader will find details of procedures for doing this in [9, Appendix A], and in [9], which deals specifically with the Pironneau–Polak method.

REFERENCES


Elijah Polak (S'61-M'62) was born in Bialystok, Poland, on August 11, 1931. He received the B.E.E. degree from the University of Melbourne, Melbourne, Australia, in 1956, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, in 1959 and 1961, respectively.


Dr. Polak is a member of the Society for Industrial and Applied Mathematics, of the editorial board of the SIAM Journal on Control, and of Sigma Xi. He is on the Joint Automatic Control Conference Program Committee for 1972.