- *box constraints:* $C = \{x \in \mathbb{R}^n : \ l \leq x \leq u\}$. *In this case we describe the projection component-wise*

$$\rho_C(\bar{x})_i = \begin{cases} l_i & \text{if } \bar{x}_i < l_i \\ \bar{x}_i & \text{if } l_i \leq \bar{x}_i \leq u_i \\ u_i & \text{if } \bar{x}_i > u_i \end{cases},$$

  *for* $i = 1, \ldots, n$.

*In both cases we have that projection costs* $\mathcal{O}(n)$.

## 5.3 Frank-Wolfe method

The Frank-Wolfe method (aka conditional gradient method or reduced gradient method) is an iterative first-order optimization algorithm originally proposed by Marguerite Frank and Philip Wolfe in 1956 to solve quadratic programming problems with linear constraints. It has seen an impressive revival recently due to its nice properties compared to projected gradient methods, in particular for machine learning applications. In this section, we will describe in depth the method and its theoretical properties. Furthermore, we will try to explain why people in data science use this method in practice. Here, we consider a problem of the form

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned} \tag{5.7}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a convex function with Lipschitz continuous gradient having constant $L > 0$ and $C \subseteq \mathbb{R}^n$ is a convex compact[4] set. We start by giving some definition that can help in the theoretical analysis. In particular, we define the *diameter* of the set $C$ as

$$D = \max_{x,y \in C} \|x - y\|_2.$$

From compactness of $C$, we have that $D$ is a finite value.

Now, we give a detailed description of the algorithm. We start with a feasible solution and, at each iteration, we define a descent direction in the current iterate $x_k$ by solving the problem:

$$\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$$

We notice that this is equivalent to minimize the linear approximation of $f$ in $x_k$:

$$\min_{x \in C} f(x_k) + \nabla f(x_k)^\top (x - x_k)$$

From compactness[5] of $C$, we have that there exists a solution $\hat{x}_k \in C$ for the linearized problem. If

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) = 0,$$

[4] A compact set is both closed and bounded.

[5] We use the Weierstrass theorem here. If we minimize a continuous function over a compact set, we can always be sure there exists a global minimum for the problem.

then we have

$$0 = \nabla f(x_k)^\top (\hat{x}_k - x_k) \leq \nabla f(x_k)^\top (x - x_k) \quad \forall\, x \in C$$

and $x_k$ satisfies first order optimality conditions. if

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) < 0$$

we have a new descent direction in $x_k$:

$$d_k = \hat{x}_k - x_k.$$

Thus we can have a new iterate

$$x_{k+1} = x_k + \alpha_k d_k$$

with $\alpha_k \in (0, 1]$ calculated by means of a line search. We report here the scheme of the method:

**Algorithm 11** `Frank-Wolfe method`

1 Choose a point $x_1 \in C$
2 For $k = 1, \ldots$
3 Set $\hat{x}_k = \arg\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k \in (0, 1]$ suitably chosen stepsize
6 End for

**Algorithm 1** `Gradient method`

1 Choose a point $x_1 \in \mathbb{R}^n$
2 For $k = 1, \ldots$
3 If $x_k$ satisfies some specific condition, then STOP
4 Set $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$, with $\alpha_k > 0$ a stepsize
5 End for

We prove convergence of the method with diminishing stepsize $\alpha = \dfrac{2}{k+1}$ in the next theorem:

**Theorem 5.13** *Let $f$ be a convex function with Lipschitz continuous gradient having constant $L > 0$. The Frank-Wolfe method with stepsize $\alpha_k = \frac{2}{k+1}$ satisfies the following inequality:*

$$f(x_{k+1}) - f(x^*) \leq \frac{2LD^2}{k+1}, \tag{5.8}$$

*for all $k \geq 1$.*

*Proof.* By using first order convexity properties, we have

$$f(x) \geq f(x_k) + \nabla f(x_k)^\top (x - x_k), \quad \forall\, x \in C.$$

Minimizing on both sides of the inequality over $C$, we get

$$f(x^*) \geq f(x_k) + \nabla f(x_k)^\top (\hat{x}_k - x_k),$$

that can be rewritten as

$$f(x^*) - f(x_k) \geq \nabla f(x_k)^\top (\hat{x}_k - x_k),$$

or equivalently as follows

$$-(f(x_k) - f(x^*)) \geq \nabla f(x_k)^\top (\hat{x}_k - x_k). \tag{5.9}$$

We consider the point

$$x_{k+1} = x_k + \alpha_k d_k = x_k + \alpha_k(\hat{x}_k - x_k)$$

and, using Lipschitz continuity of the gradient an inequality (5.9), we write

$$\begin{aligned}
f(x_{k+1}) - f(x^*) &\leq f(x_k) + \alpha_k \nabla f(x_k)^\top (\hat{x}_k - x_k) + \frac{\alpha_k^2 L}{2}\|x_k - \hat{x}_k\|^2 - f(x^*) \\
&\leq f(x_k) - f(x^*) - \alpha_k(f(x_k) - f(x^*)) + \frac{\alpha_k^2 L}{2}\|x_k - \hat{x}_k\|^2 \\
&\leq (1 - \alpha_k)(f(x_k) - f(x^*)) + \frac{\alpha_k^2 L}{2}\|x_k - \hat{x}_k\|^2 \\
&\leq (1 - \alpha_k)(f(x_k) - f(x^*)) + \frac{\alpha_k^2 L D^2}{2}.
\end{aligned}$$

We set $r_{k+1} = f(x_{k+1}) - f(x^*)$ and rewrite

$$r_{k+1} \leq (1 - \alpha_k) r_k + \frac{\alpha_k^2 L D^2}{2}. \tag{5.10}$$

By induction we can show

$$r_{k+1} \leq \frac{2LD^2}{k+1}.$$

We first prove that the inequality holds for $k = 1$. By means of inequality (5.10), we get

$$r_2 \leq (1 - \alpha_1) r_1 + \frac{\alpha_1^2 L D^2}{2}.$$

Since $\alpha_1 = 1$, we have

$$r_2 \leq \frac{LD^2}{2} \leq LD^2.$$

Now we assume that inequality

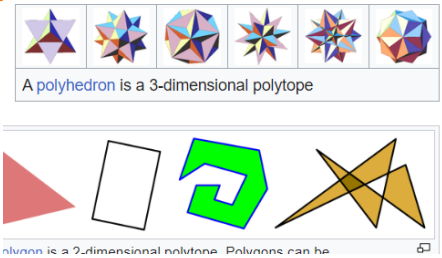
$$r_{k+1} \leq \frac{2LD^2}{k+1}$$

holds for any $k \geq 1$, we want to show that it holds also for $k + 1$. By using inequality (5.10) again, we get

$$\begin{aligned}
r_{k+2} &\leq (1 - \alpha_{k+1}) r_{k+1} + \frac{\alpha_{k+1}^2 L D^2}{2} \\
&\leq \left(1 - \frac{2}{k+2}\right)\frac{2LD^2}{k+1} + \frac{LD^2}{2}\left(\frac{2}{k+2}\right)^2 \\
&= 2LD^2\left(\frac{k}{(k+1)(k+2)} + \frac{1}{(k+2)^2}\right) \\
&\quad \text{(using the fact that } k+1 \leq k+2\text{)} \\
&\leq 2LD^2\left(\frac{k}{(k+1)(k+2)} + \frac{1}{(k+1)(k+2)}\right) \\
&= \frac{2LD^2}{k+2}.
\end{aligned}$$

Thus concluding the proof. □

This result implies that the convergence rate is $\mathcal{O}(\frac{1}{k})$. It is possible to improve the rate (i.e., getting a linear rate) if we make stronger assumptions on the problem, that is:




- feasible sets with special structure (like, e.g., polytopes);
- function is $\sigma$-strongly convex;
- optimal solution in the interior.



In Figure 5.4, we report an iteration of the Frank-Wolfe method.

For a given point $x \in C$ we can define a simple dual function:

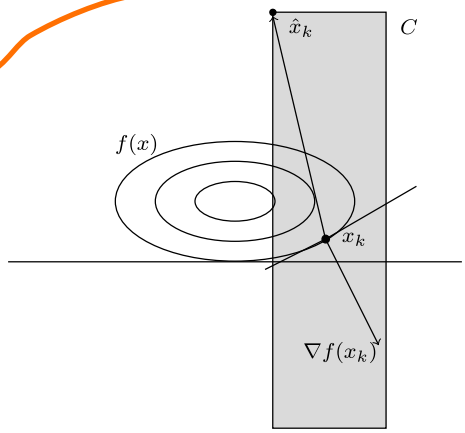



Figure 5.4: Iteration of the Frank-Wolfe method.

$$w(x) = \min_{z \in C} f(x) + \nabla f(x)^\top (z - x).$$

This minimum is always attained since $C$ is compact and the linear function is continuous in $z$. A weak duality result holds in this case

**Proposition 5.14** *For all pairs $x, y \in C$ it holds that*

$$w(x) \leq f(y).$$

*Proof.* We have:

$$\begin{aligned} w(x) &= \min_{z \in C} f(x) + \nabla f(x)^\top (z - x) \\ &\leq f(x) + \nabla f(x)^\top (y - x) \\ &\leq f(y). \end{aligned}$$

□

For a given point $x \in C$ we can define the duality gap:

$$g(x) = f(x) - w(x) = \max_{z \in C} \nabla f(x)^\top (x - z) = -\min_{z \in C} \nabla f(x)^\top (z - x).$$

By the weak duality result we have

$$g(x) \geq f(x) - f(x^*) \geq 0, \quad \forall\, x \in C.$$

Since primal error (i.e., $f(x) - f(x^\star)$) is not computable ($x^*$ unknown), duality gap represents a good optimality measure, e.g. as a stopping criterion. Keep in mind that, since we solve problem at Step 3 of the algorithm, we have $g(x)$ for free at each iteration. So, if we want a primal gap $0 \leq f(x) - f(x^*) \leq \epsilon$, we need

$$f(x) - f(x^*) \leq g(x) \leq \epsilon.$$

Thus we can stop the method when

$$\nabla f(x_k)^\top (\hat{x}_k - x_k) \geq -\epsilon.$$

The Frank-Wolfe method is really appealing in the machine learning context for two main reasons:

- the cost per iteration is much smaller than the one we have for projected gradient method (Frank-Wolfe method is a projection-free algorithm);
- the iterates keep desirable structure (like, e.g., sparsity).

We will discuss in depth these two facts. First of all, it is easy to see that solving problem at Step 2 of the algorithm costs less than projecting over $C$ in general. Indeed, if we think about a polyhedral feasible set, at each iteration we need to solve a linear program, while projection is equivalent to solve a quadratic programming problem.

### 5.3.1 Frank-Wolfe method for structured feasible sets

At each iteration of the algorithm we solve the problem:

$$\hat{x}_k = \arg\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$$

When $C$ is a polytope, we know, by means of the fundamental theorem of linear programming that one of the vertices is solution of the linear program. Hence, we get that the Frank-Wolfe iteration in some cases has linear cost:

**Unit simplex.** feasible set is

$$C = \{x \in R^n : \; e^\top x = 1, \; x \geq 0\} = conv(\{e_i, \; i = 1, \ldots, n\});$$

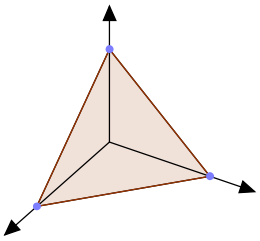
solution in this case is

$$\hat{x}_k = e_{i_k},$$

with $i_k = \arg\min_i \nabla_i f(x_k)$. It is easy to see that cost per iteration is $\mathcal{O}(n)$. A 3D simplex can be seen in Figure (5.5). The Frank-Wolfe scheme is

**Algorithm 12** `Frank-Wolfe method for unit simplex`

1 Set $x_1 = e_i$, with $i = 1, \ldots, n$
2 For $k = 1, \ldots$
3 Set $\hat{x}_k = e_{i_k}$, with $i_k = \arg\min_i \nabla_i f(x_k)$.
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k = \frac{2}{k+1}$
6 End for



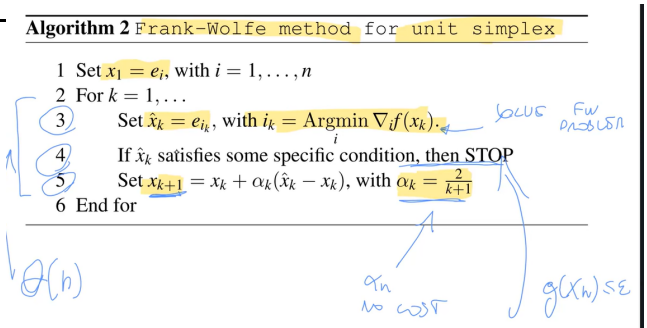



Figure 5.5: Unit simplex.

**$\ell_1$-ball.** feasible set is

$$C = \{x \in R^n : \ \|x\|_1 \leq 1\} = conv(\{\pm e_i, \ i = 1, \ldots, n\});$$

solution in this case is

$$\hat{x}_k = sign(-\nabla_{i_k} f(x_k)) \cdot e_{i_k},$$

with $i_k = \arg\max_i |\nabla_i f(x_k)|$. It is easy to see that cost per iteration is $\mathcal{O}(n)$. A 3D $\ell_1$ ball can be seen in Figure (5.6). The Frank-Wolfe scheme is

**Algorithm 13** `Frank-Wolfe method for` $\ell_1$ `ball`

1 Set $x_1 = \pm e_i$, with $i = 1, \ldots, n$
2 For $k = 1, \ldots$
3 Set $\hat{x}_k = sign(-\nabla_{i_k} f(x_k)) \cdot e_{i_k}$, with $i_k = \arg\max_i |\nabla_i f(x_k)|$.
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $x_{k+1} = x_k + \alpha_k(\hat{x}_k - x_k)$, with $\alpha_k = \frac{2}{k+1}$
6 End for

It is further easy to see that at most one new nonzero is included at each iteration in both cases. Support of the solution (i.e., number of nonzero component of $x_k$) is upper bounded by $k$. Frank-Wolfe can be seen in this case as a variant of coordinate descent (we use coordinate directions at each step).

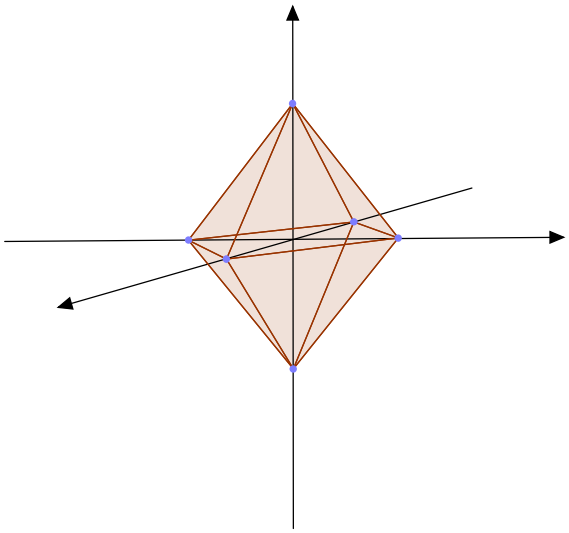


Figure 5.6: $\ell_1$ ball.

If we choose a re-parameterization of $C$ by a surjective linear or affine map $M : \hat{C} \to C$ then

$$\min_{x \in C} f(x) \equiv \min_{y \in \hat{C}} \hat{f}(y)$$

with $\hat{f}(y) = f(My)$. In this case is easy to see that every iteration of FW Algorithm remains the same (thanks to $\nabla \hat{f}(y) = M^\top \nabla f(My)$). Hence we have that Frank-Wolfe is invariant under "distortion" (existing approaches in optimization strongly depend on distortion of the domain). In order to better understand this concept, we consider problem

$$\begin{array}{l} \min f(x) \\ x \in C = conv\{v_1, \ldots, v_p\}, \end{array} \tag{5.11}$$

where $C$ is a polytope that can be described in terms of its vertices. We use *bary-centric coordinates* to reparameterize the problem. In this case $M$ contains the vertices as columns

$$M = [v_1 \ldots v_p]$$

and $\hat{C}$ is just the unit simplex. It is easy to see that solving the original problem using the Frank-Wolfe method is equivalent to solve (by means of Frank-Wolfe) the reparameterized problem

$$\begin{array}{ll} \min & f(My) \\ s.t. & e^\top y = 1 \\ & y \geq 0 \end{array} \tag{5.12}$$

Indeed, at each iteration we have

$$\begin{array}{ll} \min & \nabla \hat{f}(y_k)^\top (y - y_k) \\ s.t. & e^\top y = 1 \\ & y \geq 0 \end{array} \tag{5.13}$$

and by taking into account the expression of $\hat{f}$, we get

$$\nabla \hat{f}(y_k)^\top (\hat{y}_k - y_k) = \left(M^\top \nabla f(My_k)\right)^\top (\hat{y}_k - y_k) = \nabla f(My_k)^\top (M\hat{y}_k - My_k).$$

By further considering that $x_k = My_k$ and $\hat{x}_k = M\hat{y}_k$, we get

$$\nabla \hat{f}(y_k)^\top (\hat{y}_k - y_k) = \nabla f(x_k)^\top (\hat{x}_k - x_k).$$

### 5.3.2 Sublinear rate in Frank-Wolfe method

The sublinear rate in the Frank-Wolfe algorithm is due to the use a *linear minimization oracle* over $C$ that is defined as follows

$$LMO(y) = \arg\min_{x \in C} y^\top x.$$

A classic example of sublinear rate for the Frank Wolfe algorithm is obtained when $C$ is a polytope and $x^*$ is on the boundary of the feasible set. This is because the iterates of the algorithm start to zig-zag between the vertices defining the face containing the solution $x^*$. We report an example of the zig-zagging phenomenon in Figure 5.7.

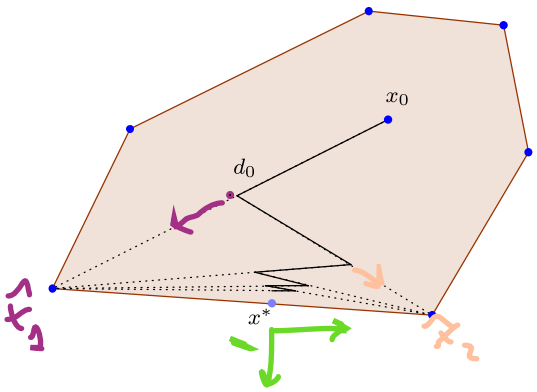



Figure 5.7: zig-zagging phenomenon.

While in general is not possible to improve the sublinear rate of the Frank-Wolfe algorithm, in the polyhedral case there exist some variants[6] of the Frank-Wolfe algorithm that guarantee (under suitable assumptions like, e.g. $\sigma$-strong convexity) convergence at a linear rate. We report here two well known variants.

[6]Those variants, as we will see, are still based on linear minimization oracles.

### 5.3.3 Away-step Frank-Wolfe method

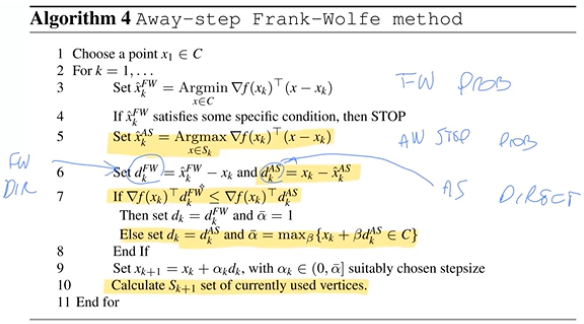
This modification of the Frank-Wolfe method was proposed by Wolfe (1970). As we have previously seen Frank-Wolfe directions are always directed towards extreme points. When we are close to the optimum (and the optimum is on the boundary) directions get more and more orthogonal to the gradient thus getting the so-called zig-zagging phenomenon. In order to avoid this, Wolfe suggested to include directions pointing away from extreme points. Linear convergence can be obtained in case $f$ is $\sigma$-strongly convex and $C$ is polyhedral. Here, we consider a problem of the form

$$\begin{aligned} &\min f(x) \\ &x \in C = conv\{v_1, \dots, v_p\} \end{aligned} \tag{5.14}$$

If we call $V = \{v_1, \dots, v_p\}$, we know that, at step $k$, iterate is represented as a sparse convex combination of at most $k$ vertices $S_k \subseteq V$. We report the scheme of the method below:

**Algorithm 14** `Away-step Frank-Wolfe method`

1 Choose a point $x_1 \in C$
2 For $k = 1, \dots$
3 Set $\hat{x}_k^{FW} = \arg\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$
4 If $\hat{x}_k^{FW}$ satisfies some specific condition, then S
5 Set $\hat{x}_k^{AS} = \arg\max_{x \in S_k} \nabla f(x_k)^\top (x - x_k)$
6 Set $d_k^{FW} = \hat{x}_k^{FW} - x_k$ and $d_k^{AS} = x_k - \hat{x}_k^{AS}$
7 If $\nabla f(x_k)^\top d_k^{FW} \leq \nabla f(x_k)^\top d_k^{AS}$
Then set $d_k = d_k^{FW}$ and $\bar{\alpha} = 1$
Else set $d_k = d_k^{AS}$ and $\bar{\alpha} = \max_\beta \{x_k + \beta d_k^{AS} \in$
8 End If
9 Set $x_{k+1} = x_k + \alpha_k d_k$, with $\alpha_k \in (0, \bar{\alpha}]$ suitably chosen stepsize
10 Calculate $S_{k+1}$ set of currently used vertices.
11 End for




Hence, at each iteration we calculate the classic Frank-Wolfe direction and the so-called away-step direction, that is a direction pointing away from the worst vertex (i.e., the one with highest value of the linearized function) describing the current iterate. Then we choose the best between the two and perform a line search along that direction (See Step 9). Finally, we update $S_k$. In Figure 5.8, we show how the away-step Frank-Wolfe works in practice. We notice that storing and updating $S_k$ might be costly in practice. Furthermore, there might be multiple ways to represent iterate $k$ as combination of vertices.

### 5.3.4 Pairwise Frank-Wolfe method

The Pairwise Frank-Wolfe method was first described by Mitchel et al. for the polytope distance problem. This method is strongly related to classic SMO

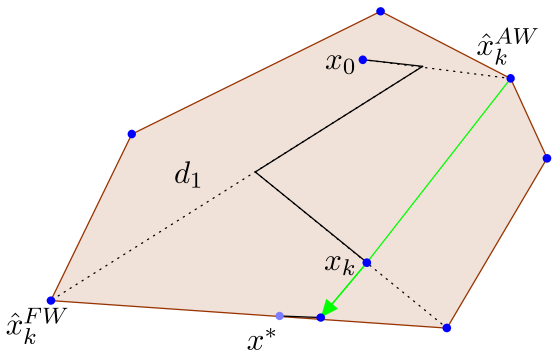



Figure 5.8: Behavior of the away-step Frank-Wolfe method.

algorithms in machine learning. The main idea is moving weight from the away vertex to the Frank-Wolfe vertex. In practice, at each iteration we use the search direction $d_k = d_k^{FW} + d_k^{AS}$. Linear convergence can be obtained under similar assumptions as the away-step Frank-Wolfe method. What we actually we get is that the linear rate is more loose than away-step variant. Anyway, the method is very efficient in practice. We report the scheme of the method below:

**Algorithm 15** `Pairwise Frank-Wolfe method`

1 Choose a point $x_1 \in C$
2 For $k = 1, \dots$
3 Set $d_k = d_k^{FW} + d_k^{AS}$ and $\bar{\alpha} = \max_\beta \{x_k + \beta d_k \in C\}$
4 Set $x_{k+1} = x_k + \alpha_k d_k$, with $\alpha_k \in (0, \bar{\alpha}]$ suitably chosen stepsize
5 Calculate $S_{k+1}$ set of currently used vertices.
6 End for

In Figure 5.9, we show how the pairwise Frank-Wolfe works in practice.

## 5.4 Fully corrective Frank-Wolfe method

In this section, we consider the more general problem:

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned} \tag{5.15}$$

where $f$ is continuously differentiable and $C$ is a compact convex set. The fully corrective Frank Wolfe method (aka Simplicial Decomposition) represents a class of methods used for dealing with convex problems. It was first introduced by Holloway (1970) and then further studied in other papers. The method basically uses an iterative *inner approximation* of the feasible set $C$. In practice, the feasible set is approximated with the convex hull of an ever expanding finite

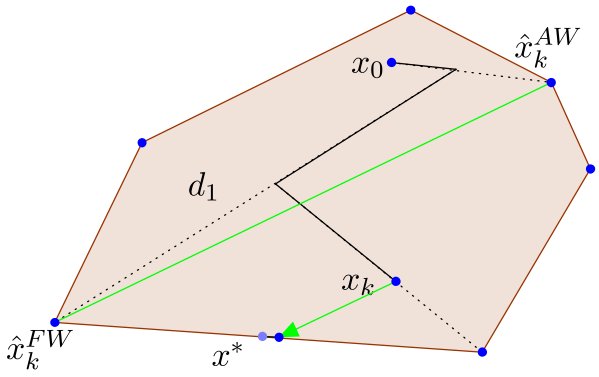



Figure 5.9: Behavior of the pairwise Frank-Wolfe method.

set $C_k = \{\hat{x}_0, \hat{x}_2, \ldots, \hat{x}_k\}$ where $\hat{x}_i$, $i = 0, \ldots,$ $k$ are extreme points of $C$. We denote this set with $conv(C_k)$:

$$conv(C_k) = \{x \mid x = \sum_{i=0}^{k} \lambda_i \hat{x}_i, \sum_{i=0}^{k} \lambda_i = 1, \lambda_i \geq 0\} \tag{5.16}$$

At each iteration, it is possible to add new extreme points to $C_k$ in such a way that a function reduction is guaranteed when minimizing the objective function over the convex hull of the new (enlarged) set of extreme points. If the algorithm does not find at least one new point, the solution is optimal and the algorithm terminates.

The use of the proposed method is particularly indicated when the following two conditions are satisfied:

1. Minimizing a linear function over $C$ is much simpler than solving the original nonlinear problem;

2. Minimizing the original objective function over the convex hull of a relatively small set of extreme points is much simpler than solving the original nonlinear problem (i.e., tailored algorithms can be used for tackling the specific problem in our case).

The first condition is needed due to the way a new extreme point is generated. Indeed, this new point is the solution of the following linear programming problem

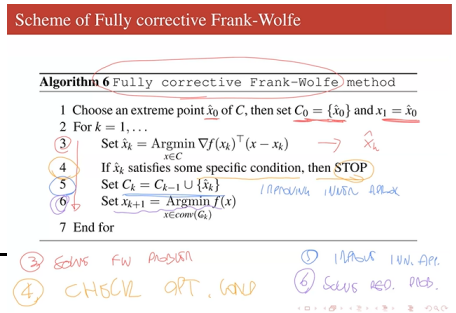
$$\begin{array}{ll} \min & \nabla f(x_k)^\top (x - x_k) \\ \text{s.t.} & x \in C \end{array} \tag{5.17}$$

where a linear approximation calculated at the last iterate $x_k$ (i.e., the solution obtained by minimizing $f$ over $conv(C_{k-1})$ ) is minimized over the original feasible set $C$.

Below, we report the detailed scheme related to the algorithm.

**Algorithm 16** `Fully corrective Frank-Wolfe method`

1 Choose an extreme point $\hat{x}_0$ of $C$, then set $C_0 = \{\hat{x}_0\}$ and $x_1 = \hat{x}_0$
2 For $k = 1, \dots$
3 Set $\hat{x}_k = \arg\min_{x \in C} \nabla f(x_k)^\top (x - x_k)$
4 If $\hat{x}_k$ satisfies some specific condition, then STOP
5 Set $C_k = C_{k-1} \cup \{\hat{x}_k\}$
6 Set $x_{k+1} = \arg\min_{x \in conv(C_k)} f(x)$
7 End for




We first generate an extreme point $\hat{x}_k$ by solving the linear program at Step 3. Then, we update $C_k$. Finally, we minimize $f$ over the set $conv(C_k)$, thus obtaining the new iterate $x_{k+1}$.

Finite convergence of the method is stated in the following Proposition:

**Proposition 5.15** *Fully corrective Frank-Wolfe algorithm obtains a solution of Problem* (5.15) *(when $C$ is a polytope) in a finite number of iterations.*

*Proof.* Extreme point $\hat{x}_k$, obtained by approximately solving linear problem at Step 3, can only satisfy one of the following conditions

1. $\nabla f(x_k)^\top (\hat{x}_k - x_k) \geq 0$. Hence we get

$$\min_{x \in C} \nabla f(x_k)^\top (x - x_k) = \nabla f(x_k)^\top (\hat{x}_k - x_k) \geq 0,$$

that is necessary and sufficient optimality conditions are satisfied and $x_k$ minimizes $f$ over the feasible set $C$;

2. $\nabla f(x_k)^\top (\hat{x}_k - x_k) < 0$, hence direction $d_k = \hat{x}_k - x_k$ is descent direction and

$$\hat{x}_k \notin conv(C_{k-1}). \tag{5.18}$$

Indeed, since $x_k$ minimizes $f$ over $conv(C_{k-1})$ it satisfies necessary and sufficient optimality conditions, that is $\nabla f(x_k)^\top (x - x_k) \geq 0$ for all $x \in conv(C_{k-1})$.

From (5.18) we thus have $\hat{x}_k \notin C_{k-1}$. Since our feasible set $C$ has a finite number of extreme points, case 2 occurs only a finite number of times, and case 1) will eventually occur. □

In practice, the method makes more progress per iteration and results in iterates that are combination of even fewer vertices (better sparsity) with respect to other Frank-Wolfe variants. Anyway, solving the inner problem, in some cases, is as hard as solving the original one. In Figure 5.9, we show how the fully corrective Frank-Wolfe works in practice.

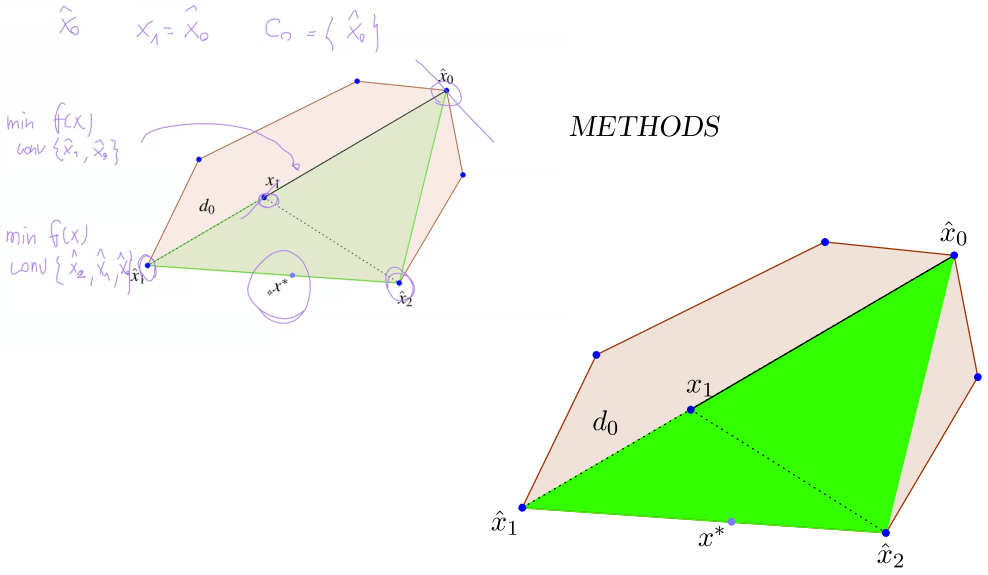



Figure 5.10: Behavior of the fully corrective Frank-Wolfe method.

## 5.5 Interior point methods

Interior point methods represent a class of approaches for solving linear and nonlinear programming problems. The first algorithm of this type was proposed by Karmarkar (1984). In his seminal paper the author described a polynomial time algorithm for solving linear programs and made some strong claims about its performance in practice. The algorithm was controversial at the time of its introduction, but there have been many improvements both in theory and practice since then. Interior point method is now considered to be better than simplex, especially on large LPs. The method, as we will see, can be used to solve general convex programs. This is the reason why, in this section, we consider problems having the following form:

$$\begin{aligned} &\min c^\top x \\ &x \in C \end{aligned} \tag{5.19}$$

where $C$ is a compact convex set with non-empty interior. We notice that any convex problem

$$\begin{aligned} &\min f(x) \\ &x \in C \end{aligned} \tag{5.20}$$

can be reformulated like Problem (5.19) by minimizing a linear function over the epigraph of the original objective $f$:

$$\begin{aligned} &\min_{x,y} y \\ &x \in C \\ &f(x) \leq y. \end{aligned} \tag{5.21}$$

In interior point methods, we add to the original objective function a term $B(x)$ defined in the interior of $C$. This function, usually called *barrier function*, is continuous and tends to $+\infty$ as the point approaches the boundary of the feasible set. The following assumptions are made on this term:

- the barrier should map the interior of the feasible space to the real space, that is $B(x) : int(C) \to \mathbb{R}$;