MATH 310 :
Degeneracy and Geometry in the Simplex Method

Fayadhoi Ibrahima *

December 11, 2013

1 Introduction

This project is exploring a bit deeper the study of the simplex method introduced in 1947 by George Dantzig [3] to solve Linear Programming problems. The simplex algorithm has been listed as one of the 10 most influential algorithms of the 20^{th} century. The lecture given in class was sufficient to understand the main lines of the algorithm and its usefulness. However, in practice, some issues arise when we try to implement the method. In particular, the phenomena of degeneracy occur naturally. Moreover, the simplex algorithm appears to have an exponential complexity for some specific problem in the worst case, preventing this method to be completely celebrated as a robust polynomial-time solver for LP. The aim of the project is to introduce the concept of degeneracy and try to give a geometric perspective on it whenever possible. Then some methods to alleviate the problems caused by degeneracy will be introduced : pivot strategy and perturbation method. Finally, we will shortly take a look at the Klee-Minty problem to explain why degeneracy is not really what prevents the simplex method to be robust. The book from Luenberger and Ye [7] has been used as a socle of knowledge in linear programming.

1.1 Linear Programming

A linear programming (LP) is a specific mathematical optimization problem with constraints where both the objective function and the constraints are linear. In standard form, it writes:

$$\begin{array}{ll} \text{minimize } c^T x & \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array} \tag{1}$$

Where $c \in \mathbb{R}^n$ is a "cost" vector, $A \in \mathbb{R}^{m \times n}$ is a "consumption" matrix, $b \in \mathbb{R}^m$ is a "resource" vector and $x \in \mathbb{R}^n$ is a "decision" vector. In other words, we can interpret a linear program as minimizing the cost of producing n items with fixed unit costs, given than we have m constraints on the required amount of resources we are going to consume.
We are going to focus on the case $m < n$ and will work with $\text{rank}(A) = m$ (full row rank) (if not, we can use Gaussian elimination to reduce A to a full row-ranked matrix (and modify b) without changing the polyhedron).

*ICME, Stanford University, Stanford, CA 94305. Email: fibrahim@stanford.edu

2 Simplex method

2.1 How it works

We have seen during the first part of the lecture that the simplex method uses the polyhedral nature of the feasible set. More specifically, the simplex method allows to move from one vertex to an adjacent one so that to reduce the cost.

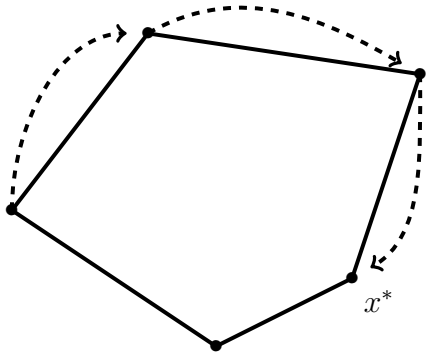

Figure 1: Moving from one vertex to an adjacent one using Simplex method until reaching an optimal vertex

Algebraically, let us recall that we are trying to come up with an algorithm to solve

$$(LP) \qquad \begin{array}{ll} \text{minimize } c^T x & \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array} \tag{2}$$

We also proved that in the non-degenerate case, if we have a point x for which $B = \{i : \ x_i > 0\}$ (the set of basic variables) is such that A_B has full rank, then x is a vertex of the corresponding polyhedron.
To go from a vertex to another, since two basic solutions are adjacent if they differ by exactly one basic variable, we only need to swap two columns of A (one corresponding to a basic solution and another one from the non-basic set) that allow to reduce the cost.

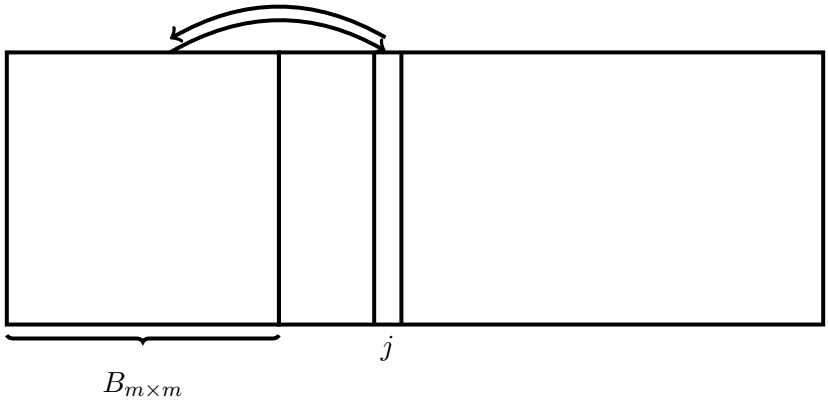

Figure 2: Swapping columns while updating basic feasible solution in simplex method

3 Degeneracy

Degeneracy is a simple concept, but has a lot of implications in the performance of the simplex algorithm, and has given rise to tons of research on improving the simplex method.

3.1 Degeneracy and geometry

3.1.1 Bases vs. extreme points

If $\mathcal{P}$ is a polyhedron, then there is two ways of viewing it. The first one is the geometric way, which means that $\mathcal{P}$ is a physical object living in an n dimensional space. If we somewhat were able to see in n dimensions and were given the polytope $\mathcal{P}$, then the solving algorithm could be simply as such :

1. take the hyperplane $\mathcal{H}$ defined by $c^T x$
2. slide it through the polytope $\mathcal{P}$ in the appropriate direction (whether you are trying to minimize or maximize)
3. get the optimal solution as (one of) the extreme point(s) you reach before $\mathcal{H} \cap \mathcal{P} = \emptyset$

However a polyhedron is usually described algebraically. Which means that, for instance in the standard form, there exists A and b such that $\mathcal{P} = \{x \in \mathbb{R}^n : \ Ax = b, x \geq 0\}$. We can first notice that there is now more ambiguity on the definition of $\mathcal{P}$ as for instance A and b might not be uniquely define (even up to a multiplicative constant).

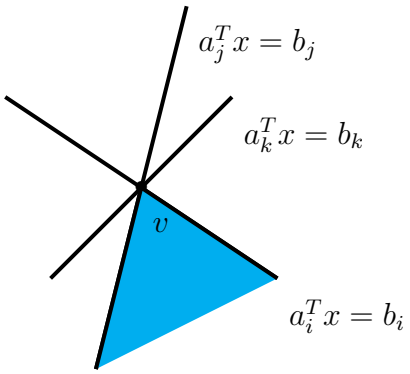

Figure 3: Degenerate extreme point v

From the previous part, we know that : x is a basic feasible solution (algebraic object) if and only if x is an extreme point (geometric object). However, this doesn't mean that there is a one-to-one correspondence between the set of bases $\mathcal{B}$ and the set of extreme points $\mathcal{V}$. Indeed, if we consider for instance $v \in \mathcal{V}$ from Figure 3, we can see that v can be defined with the help of $\{j, k\}$, $\{j, i\}$, or $\{k, i\}$, so that we have 3 bases corresponding to the same geometrical extreme point v. Therefore in general

$$|\mathcal{V}| \leq |\mathcal{B}| \tag{3}$$

And the main mechanism of the simplex algorithm is to jump from basis to basis, and not from extreme point to extreme point. Basic feasible solutions (BFSs) are in fact derived from the knowledge of the bases. Basic feasible solutions where at least one of the basic variables is zero are called degenerate. That is the reason why degeneracy is a crucial topic for the algorithm.

3.1.2 Redundant description of the polyhedron

As suggested in the previous part, degeneracy can happen when more than necessary hyperplanes are used to describe the polyhedron, and more precisely some of its extreme points. Let us illustrate it by an example.

Let's consider the following Linear Programming problem :

$$\begin{array}{ll} \text{maximize} & 3x_1 + 4x_2 \\ \text{subject to} & -2x_1 + 2x_2 \leq 2 \\ & 2x_1 - x_2 \leq 4 \\ & 0 \leq x_1 \leq 3,\ 0 \leq x_2 \leq 4 \end{array} \tag{4}$$

If we plot the corresponding polyhedron, we get the one in Figure 4. However, if we now consider the following LP :

$$\begin{array}{ll} \text{maximize} & 3x_1 + 4x_2 \\ \text{subject to} & -2x_1 + 2x_2 \leq 2 \\ & 2x_1 - x_2 \leq 4 \\ & -x_1 + x_2 \leq 3 \\ & x_1 + x_2 \leq 7 \\ & 0 \leq x_1 \leq 3,\ 0 \leq x_2 \leq 4 \end{array} \tag{5}$$

We can notice that the algebraic problem seems different, nevertheless the corresponding polyhedron is the same as shown in Figure 5, so we are dealing with the same problem geometrically speaking, but the algebraic view offers some (useless) redundant information. The issue in applications is that we are mostly dealing with large systems so that it is almost impossible to detect "redundant" hyperplanes.

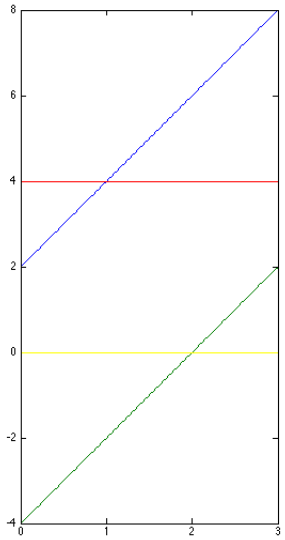

Figure 4: Polyhedron defined by the extreme points (0,0), (0,2), (1,4), (3,4), (3,2), (2,0)

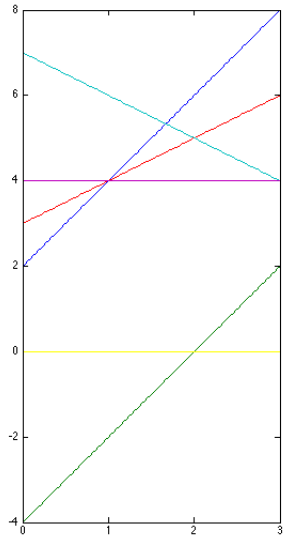

Figure 5: Same polyhedron, but red (resp. cyan) hyperplane crosses it at extreme point (1,4) (resp. (3,4))

3.1.3 Redundant variable

This second point is a bit more subtle. To illustrate it, let's consider an example. Suppose the corresponding polyhedron of an LP has characteristics :

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 1 \\ 0 \end{bmatrix}. \tag{6}$$

Then we can see that even though the problem is set in 3D, the polyhedron here lives in 2D as $x_3 = 0$, and corresponds to a triangle in the (x_1, x_2) plane with extreme points $\{(0,0,0), (1,0,0), (1,1,0)\}$.

However, we can verify that the corresponding LP has two bases :

1. $B_1 = (1,3)$ with basic feasible solution $x = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$
2. $B_2 = (2,3)$ with basic feasible solution $x = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

Here both BFSs are degenerate (since only one entry is non-zero), but the basis is in both cases uniquely determined. Therefore in this specific case, degeneracy is not an issue.

3.1.4 Inherent degeneracy

The third and last type of degeneracy is even more subtle to detect. The previous two types of degeneracy originate from a redundant description of the polyhedron. Technically, we could get rid of them by removing a row (redundant inequality) or a column (redundant variable). In dimension 2, any degeneracy can be expressed as one of those types. However, for dimension ≥ 3, some degeneracy are inherent to the geometry of the polyhedron. Let us illustrate that by the following example.

First consider the LP :

$$\begin{array}{ll} \text{maximize} & x_1 + 2x_2 + 3x_3 \\ \text{subject to} & x_1 + 2x_3 \leq 3 \\ & x_2 + 2x_3 \leq 2 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \end{array} \tag{7}$$

whose plot gives Figure 6. If we compute the BFSs, we will notice that they are all non-degenerate. And if we look at the plot, every extreme point is defined by (at most) 3 hyperplanes.

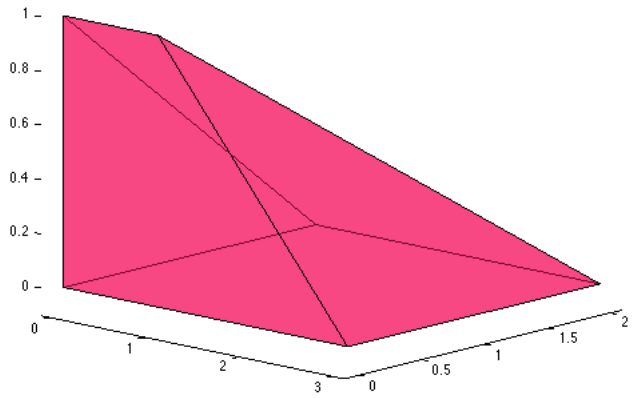

Figure 6: Simple polyhedron : each vertex is defined by 3 hyperplanes in a 3 dimensional space

However, if we consider the LP (which is in a sense a perturbation of the previous one):

$$\begin{array}{ll} \text{maximize} & x_1 + 2x_2 + 3x_3 \\ \text{subject to} & x_1 + 2x_3 \leq 2 \\ & x_2 + 2x_3 \leq 2 \\ & x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0 \end{array} \tag{8}$$

we can see from Figure 7 that no extra hyperplanes are used to define the polyhedron. However, the BFS $e_3 = (0, 0, 1)$ is degenerate. And we can notice that e_3 is actually defined by 4 hyperplanes, whereas all extreme points are defined by only 3.

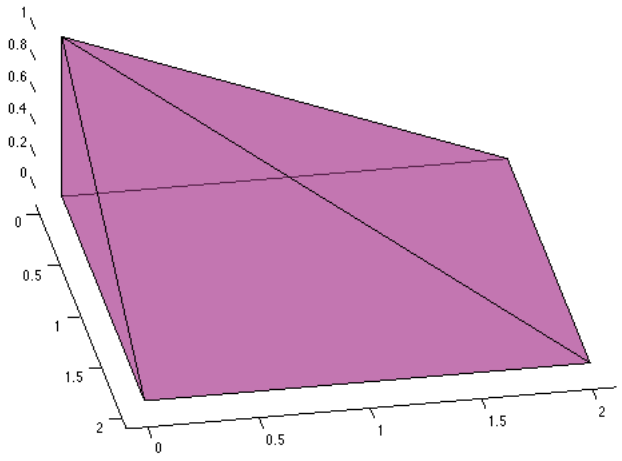

Figure 7: Degenerate polyhedron : 4 hyperplanes cross to define the vertex (0, 0, 1) in a 3 dimensional space

Therefore, for dimension $d \geq 3$, degeneracy can appear naturally in the description of the polyhedron (or polytope)[1] if more than d hyperplanes meet in a common point.

3.2 Effects: Stalling and cycling

Basic feasible solutions may result in pivots for which there is no improvement in the objective value. In this case there is no actual change in the solution but only a change in the set of basic variables. When several such pivots occur in succession, there is no improvement; in large industrial applications, degeneracy is common and such "stalling" is notable. Worse than stalling is the possibility the same set of basic variables occurs twice, in which case, the deterministic pivoting rules of the simplex algorithm will produce an infinite loop, or "cycle". While degeneracy is the rule in practice and stalling is common, cycling is rare in practice. [from Wikipedia, [2]]

3.3 Corrections

3.3.1 How to recognize LPs with degenerate BFS ?

Since the degeneracy is a big issue, is there a way of identifying (at a polynomial cost at most) if the problem has degenerate BFSs? The question will remain unanswered. However, we could state that philosophically the ultimate goal could be to be able to characterize completely the mapping $\phi_{\mathcal{P}} : \mathcal{V} \mapsto \mathcal{B}$ such that $\phi_{\mathcal{P}}(v) = \mathcal{B}_v$, for all $v \in \mathcal{V}$, where $\mathcal{B}_v$ is the set of all bases corresponding to v. Or more precisely in the case of the simplex algorithm, the inverse mapping $\psi_{\mathcal{P}} : \mathcal{B} \mapsto \mathcal{V}$, such that if we run into $B \in \mathcal{B}_v$, then the algorithm will no longer select any other $B' \in \mathcal{B}_v$. In other words, if we define the equivalence relation $\sim$ by :

$$B \sim B' \;\Leftrightarrow\; B, B' \in \mathcal{B}_v, \tag{9}$$

Then we would like to be able to build the mapping :

$$\tilde{\psi}_{\mathcal{P}} : \mathcal{B}/\sim \mapsto \mathcal{V}. \tag{10}$$

3.3.2 Different rules for pivoting

In the simplex algorithm, there can be a choice to be made for the pivot column when more than one entry on the reduced cost vector is negative (minimization case). There are at least 4 possible rules:

1. **Dantzig's rule**: choose the column corresponding to the most negative entry.
2. **Best improvement rule**: choose the column among those corresponding to negative entries in reduced cost, that will give you the largest decrease in objective function
3. **Bland's rule**: choose the lowest-numbered (i.e., leftmost) nonbasic column t with a positive cost. Now among the rows choose the one with the lowest ratio between the cost and the index in the matrix where the index is greater than zero. If the minimum ratio is shared by several rows, choose the lowest-numbered (i.e., topmost) one of them. [from Wikipedia, [1]]
4. **Random pivot**: choose randomly one column among the possible choices.

With Bland's rule (and more precisely the criss-cross algorithm, but also the lexicographic order rule), it is possible to get rid of the curse of cycling. And we are not going to give further detail on that part. Therefore it remains to have a closer look on stalling.

[1] plots obtained by `plotregion` function written by Per Bergström 2006-01-16

3.3.3 Modification a degenerate LP into an exact/approximate non-degenerate LP ?

Now suppose that we have an oracle that can tell us for sure if the LP is degenerate or not. Then if it is, can we design a procedure to transform the LP into a non-degenerate one ? Or do we need to find some new way to approximate the problem ?

3.3.4 Perturbation Method: Sensitivity analysis

We are then interested in the following questions. Suppose we perturb the data in (1) as:

$$c \rightsquigarrow c + \delta c, \quad b \rightsquigarrow b + \delta b, \quad A \rightsquigarrow A + \delta A, \tag{11}$$

Is the perturbed problem still feasible ? Is there still an optimal solution ? If yes, how would the optimal solution be updated ? And as importantly, how are the BFSs modified ?

Here we are only going to investigate on perturbations of the RHS b. The idea of perturbation is to nudge the constraints a little bit so that the new polytope becomes simple (i.e. no more degenerate bases exist) and therefore get a one-to-one correspondence between bases and extreme points ($\phi_{\mathcal{P}}$ bijective).

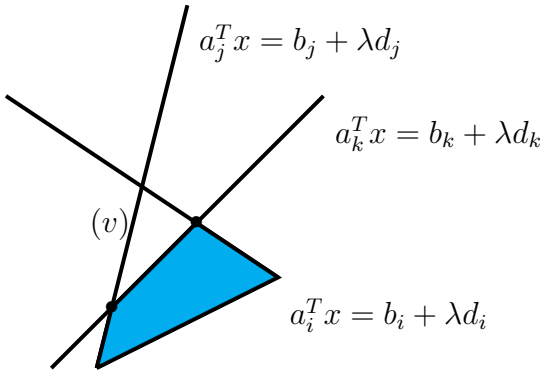

Figure 8: Nudging hyperplanes to get rid of degeneracy

Definition 1 (Perturbation method)**.** *Given $Ax = b,\ x \geq 0$, and B a basis determining a BFS, the perturbation method forms the system*

$$Ax = b',\ x \geq 0, \textit{where } b' = b + A_B \begin{bmatrix} \epsilon \\ \epsilon^2 \\ \vdots \\ \epsilon^m \end{bmatrix}, \tag{12}$$

where $\epsilon > 0$ is a "small enough" parameter.

If B' is a basis determining a BFS of the perturbed problem, then B' also determines a BFS of the original problem. Indeed it suffices to show that $A_{B'}^{-1} b \geq 0$. But since by definition of B' we have $A_{B'}^{-1} b' \geq 0$, it implies that

$$A_{B'}^{-1} b + A_{B'}^{-1} A_B \begin{bmatrix} \epsilon \\ \epsilon^2 \\ \vdots \\ \epsilon^m \end{bmatrix} \geq 0. \tag{13}$$

Now this gives a hint on how small we choose ϵ to be. For instance, if we suppose that the first entry $(A_{B'}^{-1}b)_1 = z_1 < 0$, then we could choose ϵ small enough so that the first entry of the second contribution (the one involving sum of powers of ϵ) would be $< |z_1|$, which leads to a contradiction and finishes the proof.
We actually have more : $x'_B = A_{B'}^{-1}b > 0$, which means that the BFS is non degenerate. Indeed by looking at Equation 13, we can notice that each entry has the form:

$$\alpha_0 + \alpha_1\epsilon + \cdots + \alpha_m\epsilon^m \geq 0. \tag{14}$$

Then let's consider the case $\alpha_0 + \alpha_1\epsilon + \cdots + \alpha_m\epsilon^m = 0$. Since ϵ is really small. the only way to have the sum equal to zero is to have $\alpha_0 = 0$ and $\alpha_1\epsilon + \cdots + \alpha_m\epsilon^m = 0$. Now we can divide the second equation by ϵ to get $\alpha_1 + \cdots + \alpha_m\epsilon^{m-1} = 0$. Now by repeating the same argument, we can see that $\alpha_0 = \alpha_1 = \cdots = \alpha_m = 0$. But then this means that one row of $A_{B'}^{-1}A_B$ is a zero row, which is impossible since the latter matrix is invertible.
Therefore using this power formulation allows to build non-degenerate BFSs in an elegant way.

Also, we have :

Theorem 1 (Sensitivity analysis for b)**.** *If b is replaced by $b + \lambda d$ (d direction of perturbation, λ magnitude of perturbation), then the optimal basis set B of Equation 1 remains optimal for the perturbed problem if*

$$A_B^{-1}(b + \lambda d) \geq 0 \quad \textit{and} \quad c - A^T(A_B^T)^{-1}c \geq 0 \tag{15}$$

which is straightforward to prove. Therefore the optimal solution is preserved when using the perturbation method.

Example.

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 4 & 0 & 2 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ 4 \end{bmatrix}. \tag{16}$$

$x^* = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 0 \end{bmatrix}$ is a BFS of $Ax = b,\ x \geq 0$, determined by the bases $B = \{1, 3\}$, or $B = \{2, 3\}$ or $B = \{3, 4\}$. Then choosing for instance the basis $B = \{3, 4\}$, we get :

$$\begin{aligned} b' &= \begin{bmatrix} 2 \\ 4 \end{bmatrix} + \begin{bmatrix} \epsilon \\ 2\epsilon + \epsilon^2 \end{bmatrix} = \begin{bmatrix} 2 + \epsilon \\ 4 + 2\epsilon + \epsilon^2 \end{bmatrix}, \\ x'^* &= \begin{bmatrix} 0 \\ 0 \\ 2 + \epsilon \\ \epsilon^2 \end{bmatrix} \end{aligned} \tag{17}$$

And we can see that the new BFS is non-degenerate.

However we can notice that we are then increasing the number of extreme points, and we can see the perturbation method as defining a transformation $\mathcal{T} : \phi_{\mathcal{P}}(\mathcal{B}, \mathcal{V}) \mapsto \bar{\phi}_{\bar{\mathcal{P}}}(\bar{\mathcal{B}}, \bar{\mathcal{V}})$ such that $\bar{\phi}_{\bar{\mathcal{P}}}$ bijective, $\mathcal{B} \subset \bar{\mathcal{B}}$ and $|\mathcal{V}| \leq |\bar{\mathcal{V}}|$.

4 Why is the standard Simplex method not sufficient ?

4.1 Polynomiality for simplex method in non-degenerate cases

Following the work by Ye [8], who proved in 2010 that the Markov Decision Problem (MDP) with a fixed discount rate is solved with a strongly polynomial-time using the simplex algorithm with Dantzig's rule, Kitahara and Mizuno [5] went back to the general LP problem, and proved that under the condition that the primal problem is non-degenerate, i.e. if there exist $\delta > 0$ and $\gamma > 0$ such that, for any BFS $\hat{x}$ and any $j \in 1, 2, \cdots, n$, if $\hat{x}_j \neq 0$, then $\delta \leq \hat{x}_j \leq \gamma$, the simplex method with Dantzig's rule terminates in at most $n\lceil m\frac{\gamma}{\delta} \log\left(m\frac{\gamma}{\delta}\right)\rceil$ iterations.

4.2 Worst case scenarii for simplex method

If we look at the simplex method, what is really does is, geometrically, in a first phase, find an extreme point/vertex of the polyhedron generated by the constraints, then in the second phase move to an adjacent vertex with lower cost function. Therefore we can already feel that the simplex method is performing local updates. And at least two issues pop up: the algorithm in general doesn't "see" a lot of vertices at each step, and it relies on the fact that the adjacent vertices can produce (strictly) lower objective value.
For the first issue, we can notice that the number of extreme points $|V|$ is bounded by $|V| \leq \sum_{k=0}^{m} \begin{pmatrix} n \\ k \end{pmatrix}$, and typically, if $m \sim \frac{n}{2}$, we can expect in the worst case that $|V| \sim 2^{n-1}$, which is an exponential number of vertices possibly to explore ! For the second issue, we see that it can appear if the polyhedron has some degenerated faces, and hence vertices, which would produce adjacent vertices with equal objective value.

4.2.1 Exponential running time

The first bad news with the simplex method is that there exist some LPs for which the running time is unfortunately not polynomial but exponential. One practical example is the one designed by Victor Klee and George Minty [6] in 1972:

$$\begin{aligned} &\text{maximize } \sum_{j=1}^{n} 10^{n-j} x_j \\ &\text{subject to } \quad 2\sum_{j=1}^{i-1} 10^{i-j} x_j + x_i \leq 100^{i-1}, \ \forall\, 1 \leq i \leq n, \\ &\qquad\qquad\quad x_j \geq 0, \ \forall\, 1 \leq i \leq n \end{aligned} \tag{18}$$

It can be shown that if we use the Dantzig's rule for the choice of the pivot in the simplex algorithm (i.e. in this case always choosing the pivot column to be the one with the largest reduced cost), then, starting from the obvious vertex $x = 0$, it takes exactly $2^n - 1$ pivot steps to get to the optimal solution, i.e. the algorithm will explore all possible vertices !

Here, let's illustrate the phenomenon with $n = 3$.

$$
\begin{array}{ll}
\text{maximize} & 100x_1 + 10x_2 + x_3 \\
\text{subject to} & x_1 \leq 1 \\
& 20x_1 + x_2 \leq 100 \\
& 200x_1 + 20x_2 + x_3 \leq 10000 \\
& x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0
\end{array}
\tag{19}
$$

In this case, we have three constraints and three variables. After adding slack variables, we get a problem in the standard form. The system has $m = 3$ equations and $n = 6$ nonnegative variables. The procedure is shown in Table 1.

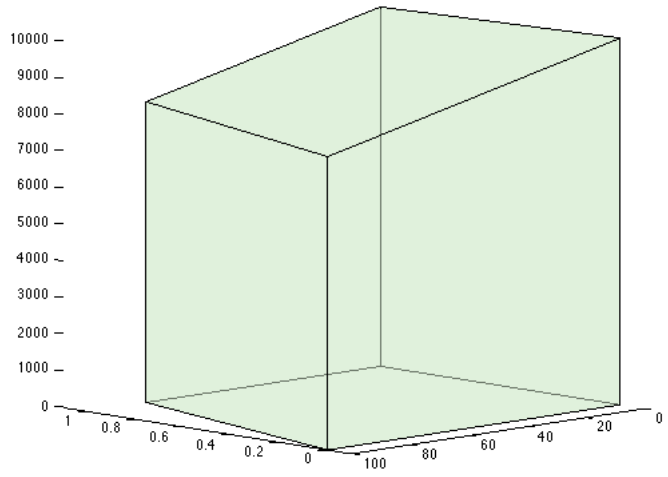

Figure 9: Klee-Minty cube in 3D. We can see that from the highest extreme point, we can find a strictly descending path that goes through all the extreme points. This is the path the standard simplex algorithm takes with the Dantzig's pivot rule.

Table 1: Simplex Steps

Table 2:

Basic	Nonbasic x_1	x_2	x_3	RHS
s_1	1^*			1
s_2	20	1		100
s_3	200	20	1	10000
$-z$	100	10	1	0

Table 3:

Basic	Nonbasic s_1	x_2	x_3	RHS
x_1	1			1
s_2	-20	1^*		80
s_3	-200	20	1	9800
$-z$	-100	10	1	-100

Table 4:

Basic	Nonbasic s_1	s_2	x_3	RHS
x_1	1^*			1
x_2	-20	1		80
s_3	200	-20	1	8200
$-z$	100	-10	1	-900

Table 5:

Basic	Nonbasic x_1	s_2	x_3	RHS
s_1	1			1
x_2	20	1		100
s_3	-200	-20	1^*	8000
$-z$	-100	-10	1	-1000

Table 6:

Basic	Nonbasic x_1	s_2	s_3	RHS
s_1	1^*			1
x_2	20	1		100
x_3	-200	-20	1	8000
$-z$	100	10	-1	-9000

Table 7:

Basic	Nonbasic s_1	s_2	s_3	RHS
x_1	1			1
x_2	-20	1^*		80
x_3	200	-20	1	8200
$-z$	-100	10	-1	-9100

Table 8:

Basic	Nonbasic s_1	x_2	s_3	RHS
x_1	1^*			1
s_2	-20	1		80
x_3	-200	20	1	9800
$-z$	100	-10	-1	-9900

Table 9:

Basic	Nonbasic x_1	x_2	s_3	RHS
s_1	1^*			1
s_2	20	1		100
x_3	200	20	1	10000
$-z$	-100	-10	-1	-10000

So we indeed go through $2^3 = 8$ tableaux before reaching the optimal solution.

Now we can go back to the result from Kitahara and Mizuno. The Klee-Minty example does not involve any degeneracy, so we should expect the conclusion to hold, i.e. the $n\lceil m\frac{\gamma}{\delta}\log\left(m\frac{\gamma}{\delta}\right)\rceil$ number of iterations bound, which is essentially polynomial in n and m. The caveat here is the ratio $\frac{\gamma}{\delta}$, which has more to do with the right-hand side b. Indeed, in our example, we can see that $\delta = 1$, while $\gamma = 10000$, so that $n\lceil m\frac{\gamma}{\delta}\log\left(m\frac{\gamma}{\delta}\right)\rceil = 1546345$, which is a really bad upper bound for the number of iterations, but emphasizes the fact that the geometry of the problem also matters.

4.3 Average case : Random pivot and KM game

We have seen in the previous part that in general, the worst case for the Klee-Minty example makes the simplex algorithm visits all the extreme points, i.e. $2^n - 1$, which is an exponential number of the data n (and we can find Klee-Minty constructions for any existing pivot rules). Trying to fix a definite rule for pivoting seems to be fruitless. So we might try to use some random rule instead. We are going to illustrate it by an example inspired from [4] and called the Klee-Minty game. Of course we are now evaluating the complexity in terms of average complexity and not worst case.

The principle is simple. Since basically in the KM case we can have a strictly decreasing path going through all the extreme points, it means that we can assume an ordering on them. And since in dimension n we have 2^n extreme points, we can use a binary representation of the extreme points. In other words, we can represent any extreme point v as

$$v = [\ b_1 \quad b_2 \quad \cdots \quad b_n\ ], \tag{20}$$

with $b_i \in \{0, 1\}$. And $v_i < v_j$ if the first left-most 1 in v_j is located before the one from v_i, and if we have equality, we look at the second left-most 1's, etc. (this is the lexicographic order). In terms of the simplex algorithm, $v_i < v_j$ means that we have made an improvement in the objective function by going from extreme point v_j to v_i.

We can now describe the game in the following algorithm (implemented in MATLAB) :

1. choose a dimension `n` for the problem
2. draw a random vector `v` of size `n` with entries `0` or `1`
3. while `v` is not the zero vector, repeat
   (a) choose a random `1` in the list of `v`
   (b) swap it, together with all the entries on its right (1 to 0, and 0 to 1) (so that we get a decrease in the value)

```
% Simplex algorithm
% dealing with average-case complexity
% Klee-Minty simulation with random improvement
% K-M game

% inspired from The Discrete Charm of geometry, by Gnter M. Ziegler
% http://www.mathematik.de/ger/information/forschungsprojekte/
    zieglergeometrie/zieglergeometrie.html

function iter = kmgame(n)

vec = ones(1,n).*(randn(1,n)>0); % start at a random vertex

iter = 0;

while sum(vec) ~= 0 % while we are not at the minimum
    % choose one 1 index at random
    ind = find(vec);
```

```
18      j = randi(length(ind),1);
19      randj = ind(j); % select a random 1 in vec
20      vec(1,randj:end) = 1 - vec(1,randj:end); % swap all the right-sided
            entries
21
22      iter = iter + 1;
23  end
```

Therefore, it means that we are choosing a random edge to go amongst the possible edges that give a strictly lower value of the objective function (random-edge rule). The algorithm is guaranteed to terminate since at each step we are making a strict progress and we have finitely many extreme points (2^n).

We can then repeat the experiment multiple times and get the average number of steps it takes to solve the problem, and this for different sizes. That is what is shown in Figure 10.

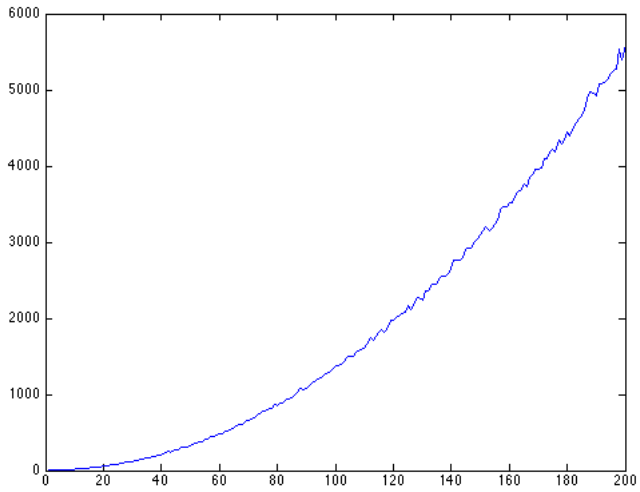

Figure 10: Average (over 200 simulations) complexity of the KM game. x-axis: size n of the problem, y-axis: average number of steps necessary to reach the optimum value.

We can now try to figure out what kind of behavior we have here. Let's call $c(n)$ the average number of steps to optimality in dimension n.

1. If we bet on a exponential profile, i.e. $c(n) = k^n$, we should have $k = c(n)^{1/n}$, which gives here $k \simeq 1.1$. The comparison of the plots is made in Figure 11. It suggests that this is not a good model at all.

2. If we bet on a polynomial profile, i.e. $c(n) = n^k$, we should have $k = \dfrac{\log(c(n))}{\log(n)}$, which gives here $k \simeq 1.6$. The comparison of the plots is made in Figure 12. The fitting is not too bad, even though we have some discrepancy.

3. Finally, going through the work in [4], it is proposed that $c(n) = k\dfrac{n^2}{\log(n)}$ (which means that the cost is somehow sub-polynomial), with some multiplicative constant k. Therefore

we can also try $k = \frac{c(n)\log(n)}{n^2}$, which gives $k \simeq 0.7$. The model fits now really well with the experiment, which suggests that :

$$c(n) = 0.7\frac{n^2}{\log(n)}. \tag{21}$$

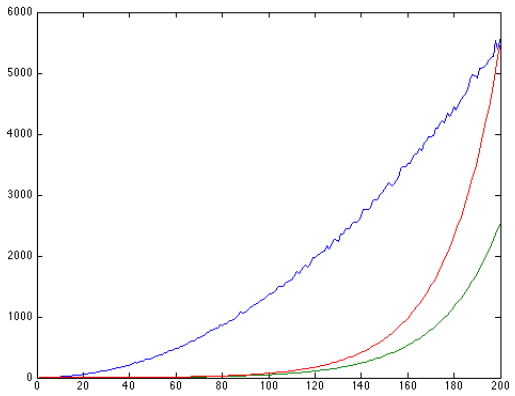

Figure 11: Model 1: exponential fitting. Red: $k = 1.1$. Green: $k = 1.05$.

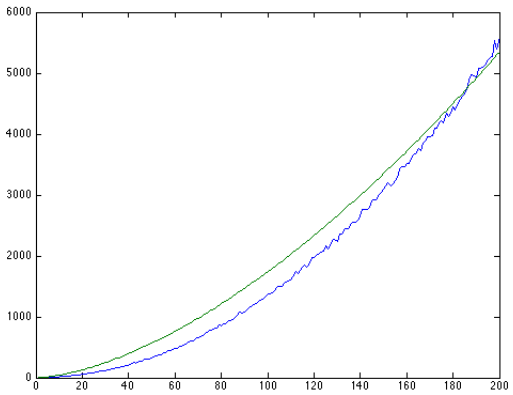

Figure 12: Model 2: polynomial fitting. $k = 1.6$.

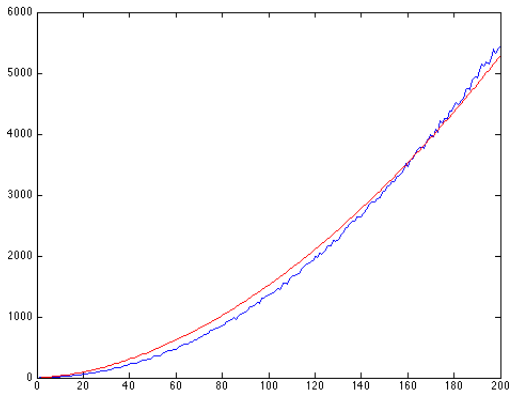

Figure 13: Model 3: sub-polynomial fitting. $k = 0.7$.

Strikingly, it seems to mean that as n increases, the probability of getting the worst-case scenario is rapidly decreasing to 0, as otherwise, we couldn't get such small values for the number of iterations, even for n as big as 200 ($2^{200} \simeq 10^{60}$).

5 Conclusions

The project is aimed to cover the essence of the simplex algorithm, without going into details in its implementation, but to point out the defects of it and where further research efforts should be put

on. We have seen that degeneracy can indeed be more understood by looking at the geometry of the polyhedra. Nonetheless, for dimension bigger than 3 the geometry cannot help us. Hence the need to deal exclusively with the algebraic representation and work on the side effects of degeneracy.
Indeed drawbacks in the simplex algorithm can be partially overcome : we can get rid of cycling by using appropriate pivot rules. We can design equivalent non-degenerate LPs from degenerate LPs using the perturbation method but we might still have some stalling effect (with regards to the original LP to solve) since we are increasing the number of BFSs, and no robust method has been invented yet to map clearly bases to BFS/extreme points.
However, the picture is actually darker than that for the simplex algorithm. Indeed for any choice of rule, it is possible to construct an example for which the simplex algorithm takes an exponential number of iteration to terminate. Therefore it may not be sufficient a strategy to get rid of the curse of non-polynomiality. The counter-example from Klee and Minty is an interesting one. What's more, the generated polyhedron is non-degenerate ! Which means that degeneracy is not really what causes the algorithm to have an exponential worst-case cost.
With that in mind, we have tried to see if there was a possible strategy to tackle the Klee-Minty puzzle, and random-edge pivot seems to give on average sub-polynomial running time.
Therefore all of these incentives suggest that the solution might come from an implementation of the simplex algorithm with some randomized aspects.

References

[1] Bland's rule. Wikipedia article. http://en.wikipedia.org/wiki/Bland%27s_rule.

[2] Simplex algorithm. Wikipedia article. http://en.wikipedia.org/wiki/Simplex_algorithm#cite_note-6.

[3] G. B. Dantzig. Application of the simplex method to a transportation problem. *Activity Analysis of Production and Allocation*, pages 359–373, 1951.

[4] B. Gärtner, M. Henk, and G. M. Ziegler. Randomized simplex algorithms on klee-minty cubes. Technical report, ETH Zürich and Technische Universität Berlin, 1998.

[5] T. Kitahara and S. Mizuno. A bound for the number of different basic solutions generated by the simplex method. *Math. Program.*, 137:579–586, 2013.

[6] V. Klee and G. J. Minty. How good is the simplex method. *Inequalities III*, 1972.

[7] D. G. Luenberger and Y. Ye. *Linear and Nonlinear Programming*. Springer, 2008.

[8] Y. Ye. The simplex and policy-iteration methods are strongly polynomial for the markov decision problem with a fixed discount rate. *Mathematics of Operations Research*, 36:593–603, 2011.