ORF 307: Lecture 4

Linear Programming: Chapter 3
Degeneracy

Robert Vanderbei

February 14, 2019

Slides last edited on February 14, 2019

Solve This...

$$
\begin{array}{lrcrcl}
\text{maximize} & 2x_1 & + & 3x_2 & & \\
\text{subject to} & x_1 & + & 2x_2 & \leq & 8 \\
 & x_1 & - & x_2 & \leq & 4 \\
 & -x_1 & + & x_2 & \leq & 4 \\
 & & & x_1,\ x_2 & \geq & 0.
\end{array}
$$

Solution

maximize ζ	=	0	+	2	x_1	+	3	x_2
w_1	=	8	-	1	x_1	-	2	x_2
w_2	=	4	-	1	x_1	-	-1	x_2
w_3	=	4	-	-1	x_1	-	1	x_2

$\Downarrow$ Enter: x_2, Leave: w_3

maximize ζ	=	12	+	5	x_1	+	-3	w_3
w_1	=	0	-	3	x_1	-	-2	w_3
w_2	=	8	-	0	x_1	-	1	w_3
x_2	=	4	-	-1	x_1	-	1	w_3

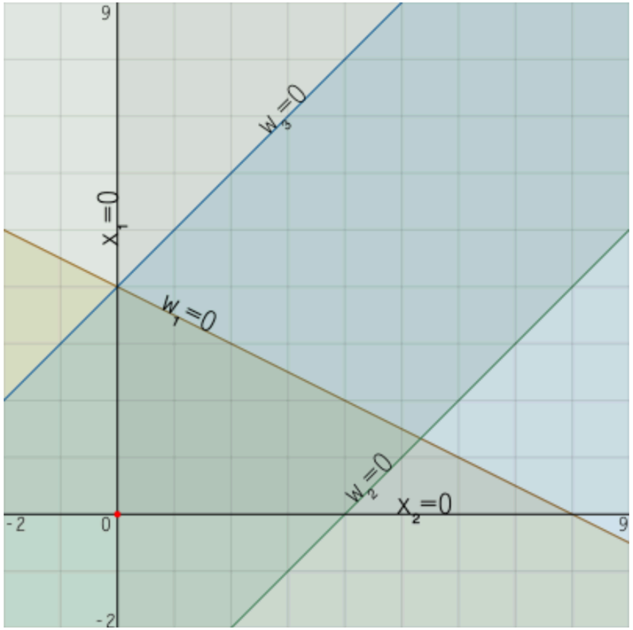
$\Downarrow$ Enter: x_1, Leave: w_1

maximize ζ	=	12	+	-5/3	w_1	+	1/3	w_3
x_1	=	0	-	1/3	w_1	-	-2/3	w_3
w_2	=	8	-	0	w_1	-	1	w_3
x_2	=	4	-	1/3	w_1	-	1/3	w_3

$\Downarrow$ Enter: w_3, Leave: w_2

maximize ζ	=	44/3	+	-5/3	w_1	+	-1/3	w_2
x_1	=	16/3	-	1/3	w_1	-	2/3	w_2
w_3	=	8	-	0	w_1	-	1	w_2
x_2	=	4/3	-	1/3	w_1	-	-1/3	w_2

Note: The horizontal axis, which one might call the x_1-axis, is where $x_2 = 0$ and is labeled as such.



In (x_1, x_2) coordinates, the pivots visit the following vertices:

$$(0,0) \Longrightarrow (0,1) \Longrightarrow (0,1) \Longrightarrow (4/3, 1/3).$$

Note that the second pivot went nowhere.

Degeneracy

Definitions.

A *dictionary is degenerate* if one or more “rhs”-value vanishes.

Example:

$$
\begin{array}{rcrcrcrcr}
\zeta & = & 6 & + & w_3 & + & 5x_2 & + & 4w_1 \\
\hline
x_3 & = & 1 & - & 2w_3 & - & 2x_2 & + & 3w_1 \\
w_2 & = & 4 & + & w_3 & + & x_2 & - & 3w_1 \\
x_1 & = & 3 & - & 2w_3 & & & & \\
w_4 & = & 2 & + & w_3 & & & - & w_1 \\
w_5 & = & 0 & & & - & x_2 & + & w_1
\end{array}
$$

A *pivot is degenerate* if the objective function value does not change.

Examples (based on above dictionary):

1. If x_2 enters, then w_5 must leave, pivot is degenerate.
2. If w_1 enters, then w_2 must leave, pivot is *not* degenerate.

Cycling

A *cycle* is a sequence of pivots that returns to the dictionary from which the cycle began.

Note: Every pivot in a cycle must be degenerate. Why?

Pivot Rules

A *pivot rule* is an explicit statement for how one chooses entering and leaving variables (when a choice exists).

Some Examples:

Largest-Coefficient Rule. (most common pivot rule for entering variable)

Choose the variable with the largest coefficient in the objective function.

Random Positive-Coefficient Rule.

Among all nonbasic variables having a positive coefficient, choose one at random.

First Encountered Rule.

In scanning the nonbasic variables, stop with the first one whose coefficient is positive.

Hope

Some pivot rule, such as the largest coefficient rule, will be proven never to cycle.

Hope

Some pivot rule, such as the largest coefficient rule, will be proven never to cycle.

Hope Fades

An example that cycles using the following pivot rules:

- entering variable: largest-coefficient rule.
- leaving variable: smallest-index rule.

$$
\begin{array}{rcrcrcrcrcr}
\zeta & = & & & x_1 & - & 2x_2 & & & - & 2x_4 \\
\hline
w_1 & = & & - & 0.5x_1 & + & 3.5x_2 & + & 2x_3 & - & 4x_4 \\
w_2 & = & & - & 0.5x_1 & + & x_2 & + & 0.5x_3 & - & 0.5x_4 \\
w_3 & = & 1 & - & x_1 & . & & & & &
\end{array}
$$

Here's a demo of cycling (ignoring the last constraint)...

ζ	=	0	+	1	x_1	+	-2	x_2	+	0	x_3	+	-2	x_4
w_1	=	0	-	1/2	x_1	-	-7/2	x_2	-	-2	x_3	-	4	x_4
w_2	=	0	-	1/2	x_1	-	-1	x_2	-	-1/2	x_3	-	1/2	x_4

$\Downarrow$ Enter: x_1, Leave: w_1

ζ	=	0	+	-2	w_1	+	5	x_2	+	4	x_3	+	-10	x_4
x_1	=	0	-	2	w_1	-	-7	x_2	-	-4	x_3	-	8	x_4
w_2	=	0	-	-1	w_1	-	5/2	x_2	-	3/2	x_3	-	-7/2	x_4

$\Downarrow$ Enter: x_2, Leave: w_2

ζ	=	0	+	0	w_1	+	-2	w_2	+	1	x_3	+	-3	x_4
x_1	=	0	-	-4/5	w_1	-	14/5	w_2	-	1/5	x_3	-	-9/5	x_4
x_2	=	0	-	-2/5	w_1	-	2/5	w_2	-	3/5	x_3	-	-7/5	x_4

$\Downarrow$ Enter: x_3, Leave: x_1

ζ	=	0	+	4	w_1	+	-16	w_2	+	-5	x_1	+	6	x_4
x_3	=	0	-	-4	w_1	-	14	w_2	-	5	x_1	-	-9	x_4
x_2	=	0	-	2	w_1	-	-8	w_2	-	-3	x_1	-	4	x_4

$\Downarrow$ Enter: x_4, Leave: x_2

$$
\begin{array}{rcrcrlcrlcrlcrl}
\zeta & = & 0 & + & 1 & w_1 & + & -4 & w_2 & + & -1/2 & x_1 & + & -3/2 & x_2 \\
x_3 & = & 0 & - & 1/2 & w_1 & - & -4 & w_2 & - & -7/4 & x_1 & - & 9/4 & x_2 \\
x_4 & = & 0 & - & 1/2 & w_1 & - & -2 & w_2 & - & -3/4 & x_1 & - & 1/4 & x_2
\end{array}
$$

$\Downarrow$ Enter: w_1, Leave: x_3

$$
\begin{array}{rcrcrlcrlcrlcrl}
\zeta & = & 0 & + & -2 & x_3 & + & 4 & w_2 & + & 3 & x_1 & + & -6 & x_2 \\
w_1 & = & 0 & - & 2 & x_3 & - & -8 & w_2 & - & -7/2 & x_1 & - & 9/2 & x_2 \\
x_4 & = & 0 & - & -1 & x_3 & - & 2 & w_2 & - & 1 & x_1 & - & -2 & x_2
\end{array}
$$

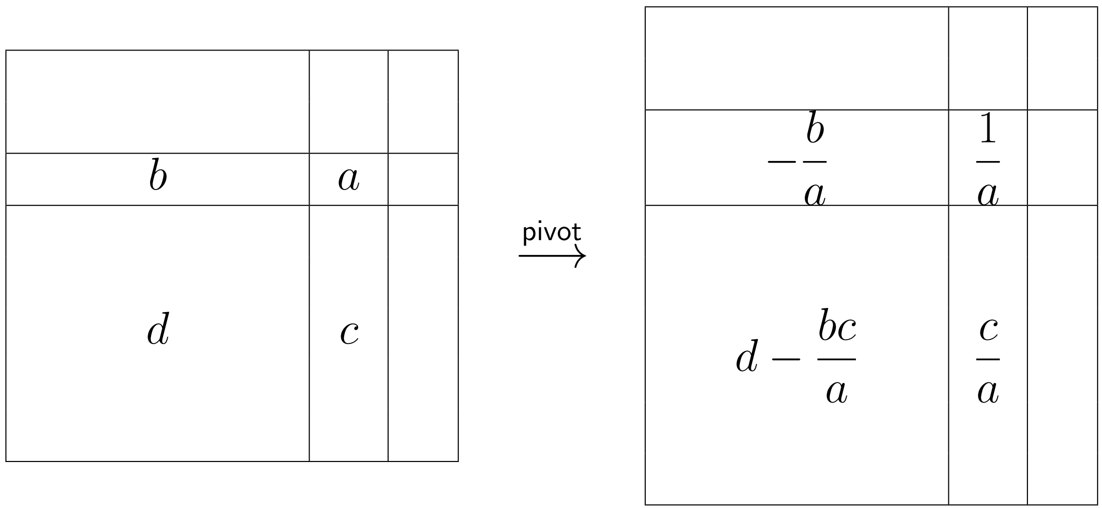
$\Downarrow$ Enter: w_2, Leave: x_4

$$
\begin{array}{rcrcrlcrlcrlcrl}
\zeta & = & 0 & + & 0 & x_3 & + & -2 & x_4 & + & 1 & x_1 & + & -2 & x_2 \\
w_1 & = & 0 & - & -2 & x_3 & - & 4 & x_4 & - & 1/2 & x_1 & - & -7/2 & x_2 \\
w_2 & = & 0 & - & -1/2 & x_3 & - & 1/2 & x_4 & - & 1/2 & x_1 & - & -1 & x_2
\end{array}
$$

Cycling is rare for small problems! A program that generates random 2×4 fully degenerate problems was run more than *one billion* times and did not find one example!

However, for larger problems with lots of zeros, cycling is common and can be a real problem.

Algebra of a Pivot



AMPL Code

```
param m := 2;
param n := 4;

param c {1..n};          param A {1..m, 1..n};
param nonbasics {1..n}; param basics {1..m};
param row;               param col;
param ii;                param jj;
param Arow {1..n};       param Acol {1..m};
param cj;                param bi;
param a;                 param ccol;
param iter;

for {k in 1..1000000000} {
   let {i in 1..m, j in 1..n} A[i,j] := Normal01();
   let {j in 1..n}            c[j]   := Normal01();
   let {j in 1..n}            nonbasics[j] := j;
   let {i in 1..m}               basics[i] := n+i;
   display k;

   let iter := 1;
   repeat while (max {j in 1..n} c[j] > 0) {
        let cj := 0;
        for {j in 1..n} {
            if (c[j] > cj) then {
                let col := j;
                let cj  := c[j];
            }
        }
        let jj := nonbasics[col];

        let bi := m+n+1;
        for {i in 1..m: A[i,jj] < -1e-8} {
                if (basics[i] < bi) then {
                        let bi := basics[i];
                        let row := i;
                }
        }
        if bi > m+n then {break;} # unbounded polytope
        let ii := basics[row];

        let {j in 1..n} Arow[j] := A[row,j];
        let {i in 1..m} Acol[i] := A[i,col];
        let                   a := A[row,col];

        let {i in 1..m, j in 1..n}
                 A[i,j] := A[i,j] - Acol[i]*Arow[j]/a;
        let {j in 1..n} A[row,j] := -Arow[j]/a;
        let {i in 1..m} A[i,col] :=  Acol[i]/a;
        let A[row,col] := 1/a;

        let ccol := c[col];
        let {j in 1..n} c[j] := c[j] - ccol*Arow[j]/a;
        let c[col] := ccol/a;

        let basics[row] := jj;
        let nonbasics[col] := ii;

        if iter > 15 then {
                display "found a cycling example";
                break;
        }

        let iter := iter+1;
   }
}
```

Perturbation Method

Whenever a vanishing "rhs" appears perturb it.
If there are lots of them, say k, perturb them all.
Make the perturbations at different *scales*:

$$\text{other data} \gg \epsilon_1 \gg \epsilon_2 \gg \cdots \gg \epsilon_k > 0.$$

An Example.

ζ	=	0	+	0	ϵ_1+	0	ϵ_2+	0	ϵ_3	+	2	x_1	+	4	x_2
w_1	=	0	+	1	ϵ_1+	0	ϵ_2+	0	ϵ_3	−	-1	x_1	−	1	x_2
w_2	=	0	+	0	ϵ_1+	1	ϵ_2+	0	ϵ_3	−	-3	x_1	−	1	x_2
w_3	=	0	+	0	ϵ_1+	0	ϵ_2+	1	ϵ_3	−	4	x_1	−	-1	x_2

Entering variable: x_2
Leaving variable: w_2

ζ	=	0	+	0	ϵ_1+	4	ϵ_2+	0	ϵ_3	+	14	x_1	+	-4	w_2
w_1	=	0	+	1	ϵ_1+	-1	ϵ_2+	0	ϵ_3	−	2	x_1	−	-1	w_2
x_2	=	0	+	0	ϵ_1+	1	ϵ_2+	0	ϵ_3	−	-3	x_1	−	1	w_2
w_3	=	0	+	0	ϵ_1+	1	ϵ_2+	1	ϵ_3	−	1	x_1	−	1	w_2

Perturbation Method—Example Con't.

Recall current dictionary:

ζ	=	0	+	0	ϵ_1+	4	ϵ_2+	0	ϵ_3	+	14	x_1	+	-4	w_2
w_1	=	0	+	1	ϵ_1+	-1	ϵ_2+	0	ϵ_3	−	2	x_1	−	-1	w_2
x_2	=	0	+	0	ϵ_1+	1	ϵ_2+	0	ϵ_3	−	-3	x_1	−	1	w_2
w_3	=	0	+	0	ϵ_1+	1	ϵ_2+	1	ϵ_3	−	1	x_1	−	1	w_2

Entering variable: x_1
Leaving variable: w_3

ζ	=	0	+	0	ϵ_1+	18	ϵ_2+	14	ϵ_3	+	-14	w_3	+	-18	w_2
w_1	=	0	+	1	ϵ_1+	-3	ϵ_2+	-2	ϵ_3	−	-2	w_3	−	-3	w_2
x_2	=	0	+	0	ϵ_1+	4	ϵ_2+	3	ϵ_3	−	3	w_3	−	4	w_2
x_1	=	0	+	0	ϵ_1+	1	ϵ_2+	1	ϵ_3	−	1	w_3	−	1	w_2

DONE!

Perturbation Method Applied to Cycling Example

ζ	=	0	+	0	ϵ_1+	0	ϵ_2	+	1	x_1	+	-2	x_2	+	0	x_3	+	-2	x_4
w_1	=	0	+	1	ϵ_1+	0	ϵ_2	−	1/2	x_1	−	-1	x_2	−	-1/2	x_3	−	1/2	x_4
w_2	=	0	+	0	ϵ_1+	1	ϵ_2	−	1/2	x_1	−	-7/2	x_2	−	-2	x_3	−	4	x_4

$\Downarrow$ x_1 enters, w_2 leaves

ζ	=	0	+	0	ϵ_1+	2	ϵ_2	+	-2	w_2	+	5	x_2	+	4	x_3	+	-10	x_4
w_1	=	0	+	1	ϵ_1+	-1	ϵ_2	−	-1	w_2	−	5/2	x_2	−	3/2	x_3	−	-7/2	x_4
x_1	=	0	+	0	ϵ_1+	2	ϵ_2	−	2	w_2	−	-7	x_2	−	-4	x_3	−	8	x_4

$\Downarrow$ x_2 enters, w_1 leaves

ζ	=	0	+	2	ϵ_1+	0	ϵ_2	+	0	w_2	+	-2	w_1	+	1	x_3	+	-3	x_4
x_2	=	0	+	2/5	ϵ_1+	-2/5	ϵ_2	−	-2/5	w_2	−	2/5	w_1	−	3/5	x_3	−	-7/5	x_4
x_1	=	0	+	14/5	ϵ_1+	-4/5	ϵ_2	−	-4/5	w_2	−	14/5	w_1	−	1/5	x_3	−	-9/5	x_4

$\Downarrow$ x_3 enters, x_2 leaves

ζ	=	0	+	8/3	ϵ_1+	-2/3	ϵ_2	+	2/3	w_2	+	-8/3	w_1	+	-5/3	x_2	+	-2/3	x_4
x_3	=	0	+	2/3	ϵ_1+	-2/3	ϵ_2	−	-2/3	w_2	−	2/3	w_1	−	5/3	x_2	−	-7/3	x_4
x_1	=	0	+	8/3	ϵ_1+	-2/3	ϵ_2	−	-2/3	w_2	−	8/3	w_1	−	-1/3	x_2	−	-4/3	x_4

$\Downarrow$ w_2 enters, problem unbounded!

Note: objective function increases with every pivot: $0 < 2\varepsilon_2 < 2\varepsilon_1 < \frac{8}{3}\varepsilon_1 - \frac{2}{3}\varepsilon_2$

Other Pivot Rules

Smallest Index Rule.

Choose the variable with the smallest index (the x variables are assumed to be "before" the w variables).

Note: Also known as *Bland's rule*.

No cycling (it's been proved).

Random Selection Rule.

Select at random from the set of possibilities.

No infinite cycles.

Greatest Increase Rule.

Pick the entering/leaving pair so as to maximize the increase of the objective function over all other possibilities.

Note: Too much computation.

Needs a tie-breaking rule.

Theoretical Results

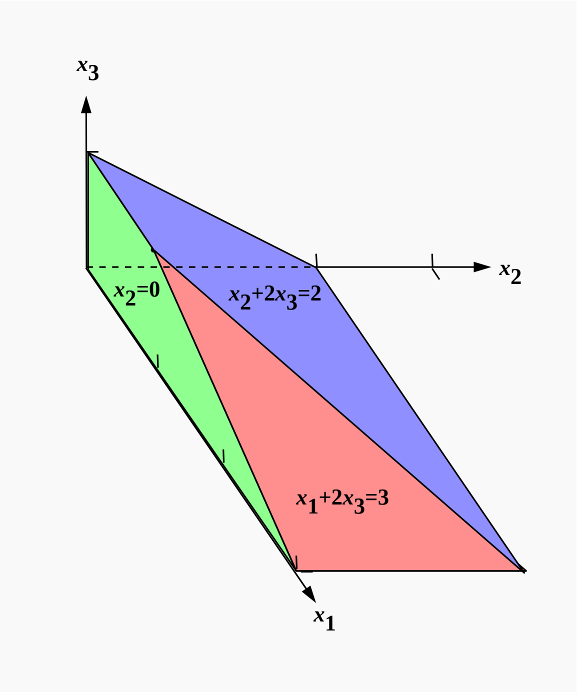
Cycling Theorem. If the simplex method fails to terminate, then it must cycle.

Why?

Fundamental Theorem of Linear Programming. For an arbitrary linear program in standard form, the following statements are true:

1. If there is no optimal solution, then the problem is either infeasible or unbounded.
2. If a feasible solution exists, then a basic feasible solution exists.
3. If an optimal solution exists, then a basic optimal solution exists.

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



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