ORF 307: Lecture 10

Linear Programming: Chapter 7
Parametric Self-Dual Simplex Method

Robert Vanderbei

March 26, 2019

Slides last edited on March 25, 2019

An Example

$$
\begin{array}{lrcrcrcr}
\text{maximize} & -3x_1 & + & 11x_2 & + & 2x_3 & & \\
\text{subj. to} & -x_1 & + & 3x_2 & & & \leq & 5 \\
 & 3x_1 & + & 3x_2 & & & \leq & 4 \\
 & & & 3x_2 & + & 2x_3 & \leq & 6 \\
 & -3x_1 & & & - & 5x_3 & \leq & -4 \\
 & & & & x_1, & x_2, \; x_3 & \geq & 0.
\end{array}
$$

Initial Dictionary:

$$
\begin{array}{rcrcrcrcr}
\zeta & = & & & -3x_1 & + & 11x_2 & + & 2x_3 \\
\hline
w_1 & = & 5 & + & x_1 & - & 3x_2 & & \\
w_2 & = & 4 & - & 3x_1 & - & 3x_2 & & \\
w_3 & = & 6 & & & - & 3x_2 & - & 2x_3 \\
w_4 & = & -4 & + & 3x_1 & & & + & 5x_3
\end{array}
$$

Note: neither primal nor dual feasible.

Perturb

Introduce a parameter μ and perturb:

$$
\begin{array}{rcrcrcrcrcrcr}
\zeta & = & & & & & -3\ x_1 & + & 11\ x_2 & + & 2\ x_3 \\
 & & & & & & -\mu x_1 & - & \mu x_2 & - & \mu x_3 \\
\hline
w_1 & = & 5 & + & \mu & + & x_1 & - & 3x_2 & & \\
w_2 & = & 4 & + & \mu & - & 3x_1 & - & 3x_2 & & \\
w_3 & = & 6 & + & \mu & & & - & 3x_2 & - & 2x_3 \\
w_4 & = & -4 & + & \mu & + & 3x_1 & & & + & 5x_3
\end{array}
$$

For μ large, dictionary is optimal.

Question: For which μ values is dictionary optimal?

Answer:

$$\begin{array}{rcccccc} -3 & - & \mu & \leq & 0 & \\ 11 & - & \mu & \leq & 0 & * \\ 2 & - & \mu & \leq & 0 & * \\ \hline 5 & + & \mu & \geq & 0 & \\ 4 & + & \mu & \geq & 0 & \\ 6 & + & \mu & \geq & 0 & \\ -4 & + & \mu & \geq & 0 & * \end{array}$$

Note: only those marked with (*) give inequalities that omit $\mu = 0$.

Tightest:

$$\mu \geq 11$$

Achieved by: objective row perturbation on x_2.

Let x_2 enter.

Who Leaves?

Do ratio test using current lowest μ value, i.e. $\mu = 11$:

$$\begin{array}{rcrcrcl} 5 & + & 11 & - & 3x_2 & \geq & 0 \\ 4 & + & 11 & - & 3x_2 & \geq & 0 \\ 6 & + & 11 & - & 3x_2 & \geq & 0 \\ -4 & + & 11 & & & \geq & 0 \end{array}$$

Tightest:

$$4 + 11 - 3x_2 \geq 0.$$

Achieved by: constraint containing basic variable w_2.

Let w_2 leave.

After the pivot:

$$
\begin{array}{rcrcrcrcrcr}
\zeta & = & \frac{44}{3} & + & \frac{11}{3}\mu & - & 14x_1 & - & \frac{11}{3} & w_2 & + & 2 & x_3 \\
 & & -\frac{4}{3}\mu & - & \frac{1}{3}\mu^2 & & & + & \frac{1}{3}\mu & w_2 & - & \mu & x_3 \\
\hline
w_1 & = & 1 & & & + & 4x_1 & + & & w_2 & & & \\
x_2 & = & \frac{4}{3} & + & \frac{1}{3}\mu & - & x_1 & - & & \frac{1}{3}w_2 & & & \\
w_3 & = & 2 & & & + & 3x_1 & + & & w_2 & - & & 2x_3 \\
w_4 & = & -4 & + & \mu & + & 3x_1 & & & & + & & 5x_3
\end{array}
$$

Advanced Pivot Tool

Using the *advanced* pivot tool, the original and current dictionaries are:

Current Dictionary:

maximize	ζ	=	0		+	0	μ	+	-3		x1	+	11		x2	+	2		x3
		+	0	μ	+	0	μ²	+	-1	μ	x1	+	-1	μ	x2	+	-1	μ	x3
w1		=	5		+	1	μ	-	-1		x1	-	3		x2	-	0		x3
w2		=	4		+	1	μ	-	3		x1	-	3		x2	-	0		x3
w3		=	6		+	1	μ	-	0		x1	-	3		x2	-	2		x3
w4		=	-4		+	1	μ	-	-3		x1	-	0		x2	-	-5		x3

$x_1, x_2, x_3, w_1, w_2, w_3, w_4 \geq 0$

$11 \leq \mu \leq \infty$

Current Dictionary:

maximize	ζ	=	44/3		+	11/3	μ	+	-14		x1	+	-11/3		w2	+	2		x3
		+	-4/3	μ	+	-1/3	μ²	+	0	μ	x1	+	1/3	μ	w2	+	-1	μ	x3
w1		=	1		+	0	μ	-	-4		x1	-	-1		w2	-	0		x3
x2		=	4/3		+	1/3	μ	-	1		x1	-	1/3		w2	-	0		x3
w3		=	2		+	0	μ	-	-3		x1	-	-1		w2	-	2		x3
w4		=	-4		+	1	μ	-	-3		x1	-	0		w2	-	-5		x3

$x_1, x_2, x_3, w_1, w_2, w_3, w_4 \geq 0$

$4 \leq \mu \leq 11$

Second Pivot

Here's the current dictionary:

Current Dictionary:

maximize ζ =	44/3		+	11/3	μ	+	-14	x_1	+	-11/3	w_2	+	2	x_3
+	-4/3	μ	+	-1/3	μ²	+	0	μ x_1	+	1/3	μ w_2	+	-1	μ x_3

w_1	=	1	+	0	μ	-	-4	x_1	-	-1	w_2	-	0	x_3
x_2	=	4/3	+	1/3	μ	-	1	x_1	-	1/3	w_2	-	0	x_3
w_3	=	2	+	0	μ	-	-3	x_1	-	-1	w_2	-	2	x_3
w_4	=	-4	+	1	μ	-	-3	x_1	-	0	w_2	-	-5	x_3

$x_1, x_2, x_3, w_1, w_2, w_3, w_4 \geq 0$

$4 \leq \mu \leq 11$

Question: For which μ values is this dictionary optimal? Answer:

$$\begin{array}{rcl|rcl} -14 & \leq & 0 & 1 & \geq & 0 \\ -\frac{11}{3}+\frac{1}{3}\mu & \leq & 0 & \frac{4}{3}+\frac{1}{3}\mu & \geq & 0 \\ 2-\mu & \leq & 0 \;\; * & 2 & \geq & 0 \\ & & & -4+\mu & \geq & 0 \;\; * \end{array}$$

Tightest lower bound: $\mu \geq 4$.

Achieved by: constraint containing basic variable w_4. Let w_4 leave.

Second Pivot–Continued

Who shall enter?

Recall the current dictionary:

Current Dictionary:

maximize	ζ =	44/3	+	11/3	μ	+	-14	x_1	+	-11/3	w_2	+	2	x_3	
	+	-4/3	μ +	-1/3	μ^2	+	0	μ x_1	+	1/3	μ w_2	+	-1	μ x_3	

w_1	=	1	+	0	μ	-	-4	x_1	-	-1	w_2	-	0	x_3
x_2	=	4/3	+	1/3	μ	-	1	x_1	-	1/3	w_2	-	0	x_3
w_3	=	2	+	0	μ	-	-3	x_1	-	-1	w_2	-	2	x_3
w_4	=	-4	+	1	μ	-	-3	x_1	-	0	w_2	-	-5	x_3

$x_1,\ x_2,\ x_3,\ \ w_1,\ w_2,\ w_3,\ w_4 \geq 0$

$4 \leq \mu \leq 11$

Do *dual-type* ratio test using current lowest μ value, i.e. $\mu = 4$:

$$\begin{aligned} 14 + 0 \cdot 4 - 3y_4 &\geq 0 \\ \tfrac{11}{3} - \tfrac{1}{3} \cdot 4 &\geq 0 \\ -2 + 1 \cdot 4 - 5y_4 &\geq 0 \end{aligned}$$

Tightest: $-2 + 1 \cdot 4 - 5y_4 \geq 0$.

Achieved by: objective term containing nonbasic variable x_3. Let x_3 enter.

Third Pivot

The current dictionary is:

Current Dictionary:

maximize ζ	=	244/15	+	49/15	μ	+	-76/5	x_1	+	-11/3	w_2	+	2/5	w_4
	+	-32/15	μ +	-2/15	μ²	+	3/5	μ x_1	+	1/3	μ w_2	+	-1/5	μ w_4
w_1	=	1	+	0	μ	-	-4	x_1	-	-1	w_2	-	0	w_4
x_2	=	4/3	+	1/3	μ	-	1	x_1	-	1/3	w_2	-	0	w_4
w_3	=	2/5	+	2/5	μ	-	-21/5	x_1	-	-1	w_2	-	2/5	w_4
x_3	=	4/5	+	-1/5	μ	-	3/5	x_1	-	0	w_2	-	-1/5	w_4

$x_1,\ x_2,\ x_3,\ \ w_1,\ w_2,\ w_3,\ w_4 \geq 0$

$2 \leq \mu \leq 4$

Question: For which μ is dictionary optimal? Answer:

$$
\begin{array}{rcrcl|rcrcl}
-\frac{76}{5} & + & \frac{3}{5}\mu & \leq & 0 & 1 & & & \geq & 0 \\
-\frac{11}{3} & + & \frac{1}{3}\mu & \leq & 0 & \frac{4}{3} & + & \frac{1}{3}\mu & \geq & 0 \\
\frac{2}{5} & - & \frac{1}{5}\mu & \leq & 0 \ * & \frac{2}{5} & + & \frac{2}{5}\mu & \geq & 0 \\
 & & & & & \frac{4}{5} & - & \frac{1}{5}\mu & \geq & 0
\end{array}
$$

Tightest lower bound: $\mu \geq 2$.

Achieved by: objective term containing nonbasic variable w_4. Let w_4 enter.

Third Pivot–Continued

Who shall leave? Recall the current dictionary:

Current Dictionary:

maximize	ζ =	244/15	+	49/15	μ	+	-76/5		x_1	+	-11/3		w_2	+	2/5		w_4
	+	-32/15 μ	+	-2/15	μ^2	+	3/5	μ	x_1	+	1/3	μ	w_2	+	-1/5	μ	w_4
w_1	=	1	+	0	μ	-	-4		x_1	-	-1		w_2	-	0		w_4
x_2	=	4/3	+	1/3	μ	-	1		x_1	-	1/3		w_2	-	0		w_4
w_3	=	2/5	+	2/5	μ	-	-21/5		x_1	-	-1		w_2	-	2/5		w_4
x_3	=	4/5	+	-1/5	μ	-	3/5		x_1	-	0		w_2	-	-1/5		w_4

$x_1, x_2, x_3, \ w_1, w_2, w_3, w_4 \geq 0$

$2 \leq \mu \leq 4$

Do *primal-type* ratio test using current lowest μ value, i.e. $\mu = 2$:

$$
\begin{array}{lllll}
1 & + & 0 \cdot 2 & & \geq 0 \\
\frac{4}{3} & + & \frac{1}{3} \cdot 2 & & \geq 0 \\
\frac{2}{5} & + & \frac{2}{5} \cdot 2 & - \ \frac{2}{5} w_4 & \geq 0 \\
\frac{4}{5} & - & \frac{1}{5} \cdot 2 & + \ \frac{1}{5} w_4 & \geq 0
\end{array}
$$

Tightest: $\frac{2}{5} + \frac{2}{5} \cdot 2 - \frac{2}{5} w_4 \geq 0$.

Achieved by: constraint containing basic variable w_3. Let w_3 leave.

Fourth Pivot

The current dictionary is:

Current Dictionary:

maximize ζ	=	50/3		+	11/3	μ	+	-11	x_1	+	-8/3	w_2	+	-1		w_3
	+	-7/3	μ	+	-1/3	μ^2	+	-3/2	μ x_1	+	-1/6	μ w_2	+	1/2	μ	w_3

w_1	=	1	+	0	μ	-	-4	x_1	-	-1	w_2	- 0 w_3
x_2	=	4/3	+	1/3	μ	-	1	x_1	-	1/3	w_2	- 0 w_3
w_4	=	1	+	1	μ	-	-21/2	x_1	-	-5/2	w_2	- 5/2 w_3
x_3	=	1	+	0	μ	-	-3/2	x_1	-	-1/2	w_2	- 1/2 w_3

$x_1, x_2, x_3, w_1, w_2, w_3, w_4 \geq 0$

$-1 \leq \mu \leq 2$

It's optimal! Also, the range of μ values includes $\mu = 0$:

$$
\begin{array}{rcrcl|rcrcl}
-11 & - & \frac{3}{2}\mu & \leq & 0 & 1 & & & \geq & 0 \\
-\frac{8}{3} & - & \frac{1}{6}\mu & \leq & 0 & \frac{4}{3} & + & \frac{1}{3}\mu & \geq & 0 \\
-1 & + & \frac{1}{2}\mu & \leq & 0 & 1 & + & 1\mu & \geq & 0 \\
 & & & & & 1 & & & \geq & 0
\end{array}
$$

That is, $-1 \leq \mu \leq 2$.

Range of μ values is shown at bottom of pivot tool. Invalid ranges are highlighted in yellow.

Top Ten Reasons to Like this Method

- Freedom to pick perturbation as you like.

Top Ten Reasons to Like this Method

- Freedom to pick perturbation as you like.
- Randomizing perturbation completely solves the degeneracy problem.

Top Ten Reasons to Like this Method

- Freedom to pick perturbation as you like.
- Randomizing perturbation completely solves the degeneracy problem.
- Perturbations don't have to be "small".

Top Ten Reasons to Like this Method

- Freedom to pick perturbation as you like.
- Randomizing perturbation completely solves the degeneracy problem.
- Perturbations don't have to be "small".
- In the optimal dictionary, perturbation is completely gone—no need to remove it.

Top Ten Reasons to Like this Method

- Freedom to pick perturbation as you like.
- Randomizing perturbation completely solves the degeneracy problem.
- Perturbations don't have to be "small".
- In the optimal dictionary, perturbation is completely gone—no need to remove it.
- The average-case performance can be analyzed (next lecture).

Top Ten Reasons to Like this Method

- Freedom to pick perturbation as you like.
- Randomizing perturbation completely solves the degeneracy problem.
- Perturbations don't have to be "small".
- In the optimal dictionary, perturbation is completely gone—no need to remove it.
- The average-case performance can be analyzed (next lecture).
- In some real-world problems, a "natural" perturbation exists (later).

Top Ten Reasons to Like this Method

- Freedom to pick perturbation as you like.
- Randomizing perturbation completely solves the degeneracy problem.
- Perturbations don't have to be "small".
- In the optimal dictionary, perturbation is completely gone—no need to remove it.
- The average-case performance can be analyzed (next lecture).
- In some real-world problems, a "natural" perturbation exists (later).

Okay, there are only 6 items in the list. SORRY.

AMPL Code

```
# generate random problem with an optimal solution
let {i in 1..m, j in 1..n} A[i,j]:=round(sigma*Normal01());
let {i in 1..m} w[i] := round(sigma*Uniform01());
let {j in 1..n} x[j] := round(sigma*Uniform01());
let {i in 1..m} y[i] := round(sigma*Uniform01());
let {j in 1..n} z[j] := round(sigma*Uniform01());
let {i in 1..m} b[i] := sum {j in 1..n} A[i,j]*x[j] + w[i];
let {j in 1..n} c[j] := sum {i in 1..m} A[i,j]*y[i] - z[j];
let {i in 1..m} bb[i] :=  sigma*Uniform01();
let {j in 1..n} cc[j] := -sigma*Uniform01();
let {i in 1..m, j in 1..n} A[i,j] := -A[i,j];

repeat while forever {

  # find entering (or leaving) variable
  let mu := -1/eps;
  let row := -1;
  let col := -1;
  for {j in 1..n} {
    let tmp := -c[j]/cc[j];
    if (c[j] > eps && tmp > mu) then {
       let col := j;
       let row := -1;
       let mu := tmp;
    }
  }
  for {i in 1..m} {
    let tmp := -b[i]/bb[i];
    if (b[i] < -eps && tmp > mu) then {
       let row := i;
       let col := -1;
       let mu := tmp;
    }
  }
```

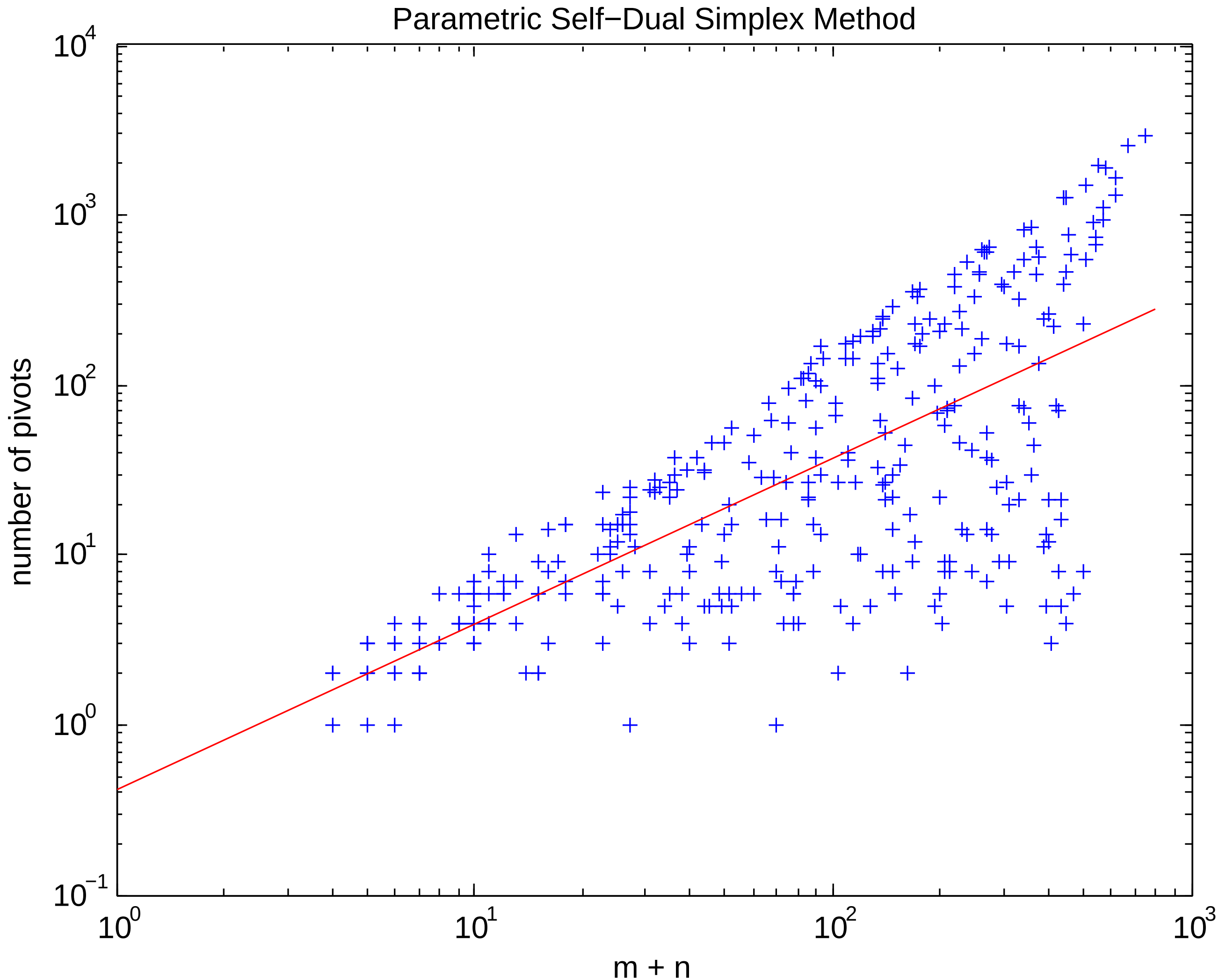
```
# if none, declare optimal
if (mu <= eps) then {
    let opt := 1;  # optimal;
    break;
}

# find leaving (or entering) variable
if (row == -1) then {
  let minratio := 1/eps;
  for {i in 1..m} {
    if (A[i,col] < -eps) then {
      if (-(b[i]+mu*bb[i])/A[i,col]
                   < minratio) then {
         let minratio := -(b[i]+mu*bb[i])/A[i,col];
         let row := i;
      }
    }
  }
  if minratio >= 1/eps then {
         let opt := -1; # dual infeas
         break;
  }
} else if (col == -1) then {
  let minratio := 1/eps;
  for {j in 1..n} {
    if (A[row,j] > eps) then {
      if (-(c[j]+mu*cc[j])/A[row,j] < minratio) then {
         let minratio := -(c[j]+mu*cc[j])/A[row,j];
         let col := j;
      }
    }
  }
  if minratio == 1/eps then {
         let opt := -1; # primal infeas
         break;
  }
}
            :
```

```
              .
              .
              .
   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 brow := b[row];
   let {i in 1..m} b[i] := b[i] - brow*Acol[i]/a;
   let b[row] := -brow/a;

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

   let brow := bb[row];
   let {i in 1..m} bb[i] := bb[i] - brow*Acol[i]/a;
   let bb[row] := -brow/a;

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

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

   let iter := iter+1;
}  # the end of forever
```

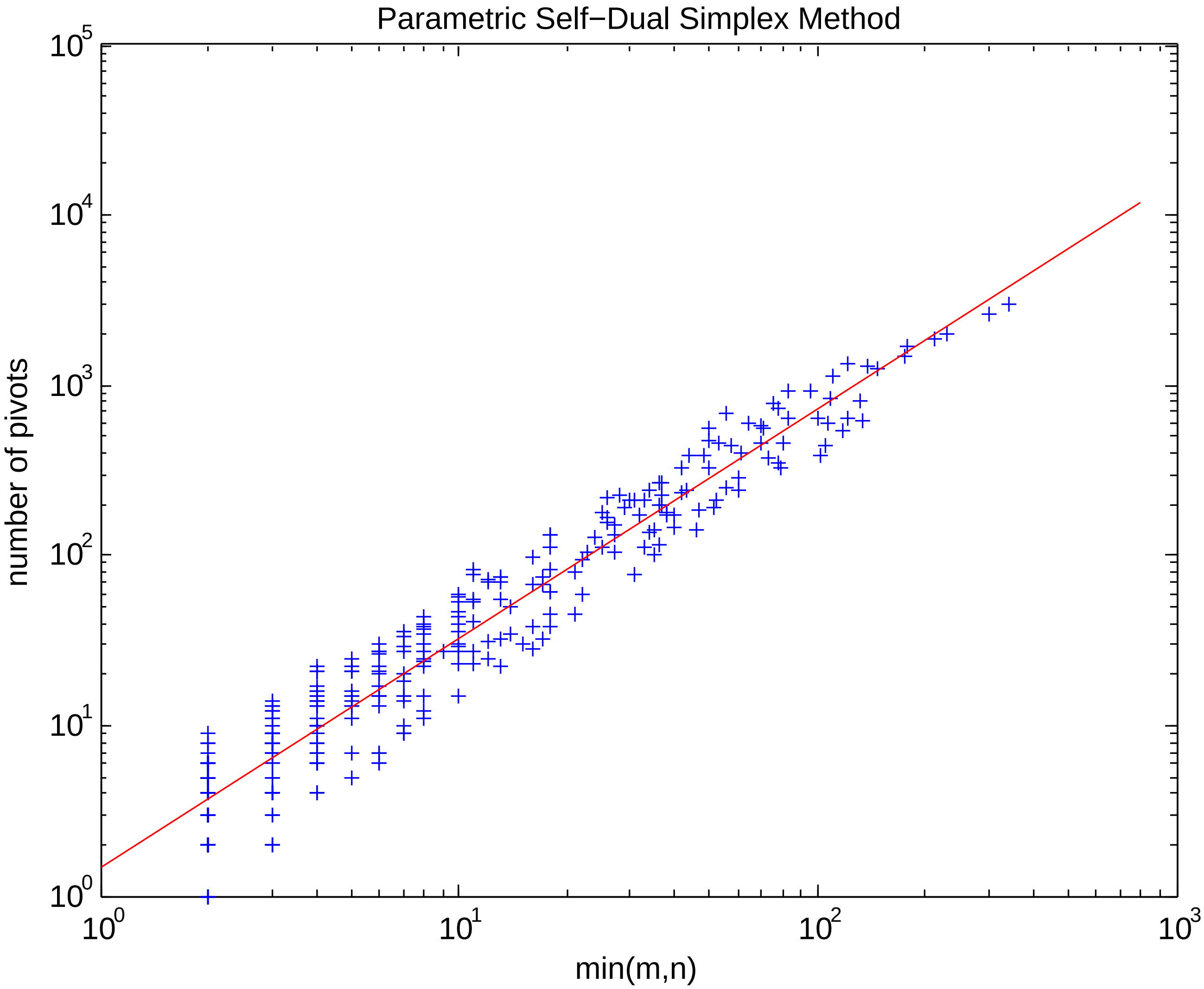
Parametric Self-Dual Simplex Method

Thought experiment:

- μ starts at ∞.
- In reducing μ, there are $n + m$ barriers.
- At each iteration, one barrier is passed—the others move about randomly.
- To get μ to zero, we must on average pass half the barriers.
- Therefore, on average the algorithm should take $(m + n)/2$ iterations.



$$\text{iters} = 0.4165(\mathsf{m} + \mathsf{n})^{0.9759}$$



$$\textsf{iters} = 1.4880 \min(\textsf{m}, \textsf{n})^{1.3434}$$