Solving Linear and Integer Programs

Robert E. Bixby

ILOG, Inc. and Rice University

Dual Simplex Algorithm

Some Motivation

- Dual simplex vs. primal: Dual > 2x faster
- Best algorithm of MIP
- There isn't much in books about implementing the dual.

Dual Simplex Algorithm
(Lemke, 1954: Commercial codes ~1990)

Input: A dual feasible basis B and vectors

$$X_B = A_B^{-1}b \quad \text{and} \quad \mathrm{D_N} = \mathrm{c_N} - \mathrm{A_N}^{\mathrm{T}}\mathrm{B}^{-\mathrm{T}}\mathrm{c_B}.$$

- **Step 1:** (Pricing) If $\mathrm{X_B} \geq 0$, stop, B is optimal; else let

$$i = argmin\{X_{Bk} : k \in \{1, \ldots, m\}\}.$$

- **Step 2:** (BTRAN) Solve $B^T z = e_i$. Compute $\alpha_N = -A_N^T z$.
- **Step 3:** (Ratio test) If $\alpha_N \leq 0$, stop, (D) is unbounded; else, let

$$j = argmin\{D_k / \alpha_k : \alpha_k > 0\}.$$

- **Step 4:** (FTRAN) Solve $\mathrm{A}_B y = A_j$.
- **Step 5:** (Update) Set $B_i = j$. Update X_B (using y) and D_N (using α_N)

Dual Simplex Algorithm

(Lemke, 1954: Commercial codes ~1990)

Input: A dual feasible basis B and vectors

$$X_B = A_B^{-1}b \quad \text{and} \quad D_N = c_N - A_N^T A_B^{-T} c_B.$$

- **Step 1:** (Pricing) If $X_B \geq 0$, stop, B is optimal; else let

$$i = argmin\{X_{Bk} : k \in \{1,\ldots,m\}\}.$$

- **Step 2:** (BTRAN) Solve $B^T z = e_i$. Compute $\alpha_N = -A_N^T z$.
- **Step 3:** (Ratio test) If $\alpha_N \leq 0$, stop, (D) is unbounded; else, let

$$j = argmin\{D_k / \alpha_k : \alpha_k > 0\}.$$

- **Step 4:** (FTRAN) Solve $A_B y = A_j$.
- **Step 5:** (Update) Set $B_i = j$. Update X_B (using y) and D_N (using α_N)

Implementing the Dual Simplex Algorithm

Implementation Issues for Dual Simplex

1. **Finding an initial feasible basis, or the concluding that there is none:** Phase I of simplex algorithm.
2. **Pricing:** Dual steepest edge
3. **Solving the linear systems**
   - LU factorization and factorization update
   - BTRAN and FTRAN – exploiting sparsity
4. **Numerically stable ratio test:** Bound shifting and perturbation
5. **Bound flipping:** Exploiting “boxed” variables to combine many iterations into one.

Issue 0
Preparation: Bounds on Variables

In practice, simplex algorithms need to accept LPs in the following form:

$$\begin{aligned} &\textit{Minimize} && c^T x \\ &\textit{Subject to} && Ax = b \qquad (\mathrm{P}_{\mathrm{BD}}) \\ & && l \le x \le u \end{aligned}$$

where l is an n-vector of **lower bounds** and u an n-vector of **upper bounds**. l is allowed to have -∞ entries and u is allowed to have +∞ entries. (Note that (P_{BD}) is in standard form if $l_j = 0,\ u_j = +\infty\ \forall\ j$.)

(Issue 0 – Bounds on variables) Basic Solution

A **basis** for (P_{BD}) is a triple (B,L,U) where B is an ordered m-element subset of $\{1,\ldots,n\}$ (just as before), (B,L,U) is a partition of $\{1,\ldots,n\}$, $l_j > -\infty\ \forall\ j \in L$, and $u_j < +\infty\ \forall\ j \in U$. $N = L \cup U$ is the set of **nonbasic** variables. The associated (**primal**) **basic solution** X is given by $X_L = l_L$, $X_U = u_U$ and

$$X_B = A_B^{-1}(b - A_L l_L - A_U u_U).$$

This solution is **feasible** if

$$l_B \le X_B \le u_B.$$

The associated **dual basic solution** is defined exactly as before: $D_B=0$, $\Pi^T A_B = c_B^T$, $D_N = c_N - A_N^T \Pi$. It is **dual feasible** if

$$D_L \ge 0 \ \ and\ \ D_U \le 0.$$

(Issue 0 – Bounds on variables)
The Full Story

- **Modify simplex algorithm**
  - Only the "Pricing" and "Ratio Test" steps must be changed substantially.
  - The complicated part is the ratio test
- **Reference:** See Chvátal for the primal

Issue 1
The Initial Feasible Basis – Phase I

- **Two parts to the solution**
  1. Finding some initial basis (probably not feasible)
  2. Modified simplex algorithm to find a feasible basis

Reference for Primal: **R.E. Bixby (1992). "Implementing the simplex method: the initial basis",** ***ORSA Journal on Computing*** 4**, 267—284.**

(Issue 1 – Initial feasible basis)
Initial Basis

- Primal and dual bases are the same. We begin in the context of the primal. Consider

$$\begin{array}{ll} \textit{Minimize} & c^T x \\ \textit{Subject to} & Ax = b \\ & l \le x \le u \end{array} \qquad (\mathrm{P_{BD}})$$

- **Assumption:** Every variable has some finite bound**.**
- **Trick:** Add **artificial variables** $x_{n+1}, \ldots, x_{n+m}$:

$$Ax + I \begin{pmatrix} x_{n+1} \\ \cdot \\ \cdot \\ x_{n+m} \end{pmatrix} = b$$

where $l_j = u_j = 0$ for $j = n+1, \ldots, n+m$.
- **Initial basis:** $B = (n+1, \ldots, n+m)$ and for each $j \notin B$, pick some finite bound and place j in L or U, as appropriate.
- **Free Variable Refinement:** Make free variables non-basic at value 0. This leads to a notion of a *superbasis*, where non-basic variables can be between their bounds.

(Issue 1 – Initial feasible basis)
Solving the Phase I

- If the initial basis is not dual feasible, we consider the problem:

$$\begin{aligned} &\mathit{Maximize} \;\; \Sigma\,(d_j : d_j < 0) \\ &\mathit{Subject\ to} \;\; A^T\pi + d = c \end{aligned}$$

- This problem is "locally linear": Define $\kappa \in \mathbf{R}^n$ by $\kappa_j = 1$ if $D_j < 0$, and 0 otherwise. Let

$$K = \{j: D_j < 0\} \;\; \mathit{and} \;\; \underline{K} = \{j: D_j \geq 0\}$$

Then our problem becomes

$$\begin{aligned} &\mathit{Maximize} \;\; \kappa^T d \\ &\mathit{Subject\ to} \;\; A^T\pi + d = c \\ &\qquad\qquad\qquad d_K \leq 0,\; d_{\underline{K}} \geq 0 \end{aligned}$$

- Apply dual simplex, and whenever d_j for $j \in K$ becomes 0, move it to $\underline{K}$.

Solving Phase I: An Interesting Computation

- Suppose d_{Bi} is the entering variable. Then $X_{Bi} < 0$ where X_B is obtained using the following formula:

$$X_B = A_B^{-1} A_N\, \kappa$$

- Suppose now that d_j is determined to be the leaving variable. Then in terms of the phase I objective, this means κ_j is replace by $\kappa_j + \varepsilon\, e_j$, where $\varepsilon \in \{0,+1,-1\}$. It can then be shown that

$$\underline{x}_{Bi} = X_{Bi} + \varepsilon\, \alpha_j$$

- **Conclusion:** If $x_{Bi} < 0$, then the current iteration can continue without the necessity of changing the basis.
- **Advantages**
  - Multiple iterations are combined into one.
  - x_{Bi} will tend not to change sign precisely when α_j is small. Thus this procedure tends to avoid unstable pivots.

Issue 2
Pricing

- The texbook rule is **TERRIBLE**: For a problem in standard form, select the entering variable using the formula

$$j = argmin\{X_{Bi} : i = 1,\ldots,m\}$$

- **Geometry is wrong:** Maximizes rate of change relative to axis; better to do relative to edge.
- Goldfard and Forrest 1992 suggested the following **steepest-edge** alternative

$$j = argmin\{X_{Bi}/\eta_i : i = 1,\ldots,m\}$$

where $\eta_i = \|e_i^T A_B^{-1}\|_2$, and gave an efficient update.
- **Note that there are two ingredients in the success of Dual SE:**
  - *Significantly reduced iteration counts*
  - *The fact that there is a very efficient update for* $\boldsymbol{\eta}_i$s

Example: Pricing

Model: dfl001

Pricing: Greatest infeasibility

```
Dual simplex - Optimal:  Objective =    1.1266396047e+07
Solution time = 1339.86 sec.  Iterations = 771647 (0)
```

Pricing: Goldfarb-Forrest steepest-edge

```
Dual simplex - Optimal:  Objective =    1.1266396047e+07
Solution time =   24.48 sec.  Iterations = 18898 (0)
```

Issue 3
Solving FTRAN, BTRAN

- **Computing LU factorization:** See Suhl & Suhl (1990). "Computing sparse LU factorization for large-scale linear programming basis", ORSA Journal on Computing 2, 325-335.
- **Updating the Factorization:** Forrest-Tomlin update is the method of choice. See Chvátal Chapter 24.
  - **There are multiple, individually relatively minor tweaks that collectively have a significant effect on update efficiency.**
- **Further exploiting sparsity:** This is the main recent development.

(Issue 3 – Solving FTRAN & BTRAN)

We must solve two linear systems per iteration:

FTRAN	BTRAN
$A_B y = A_j$	$A_B{}^T z = e_i$

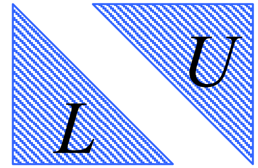
where

A_B	=	basis matrix	(very sparse)
A_j	=	entering column	(very sparse)
e_i	=	unit vector	(very sparse)

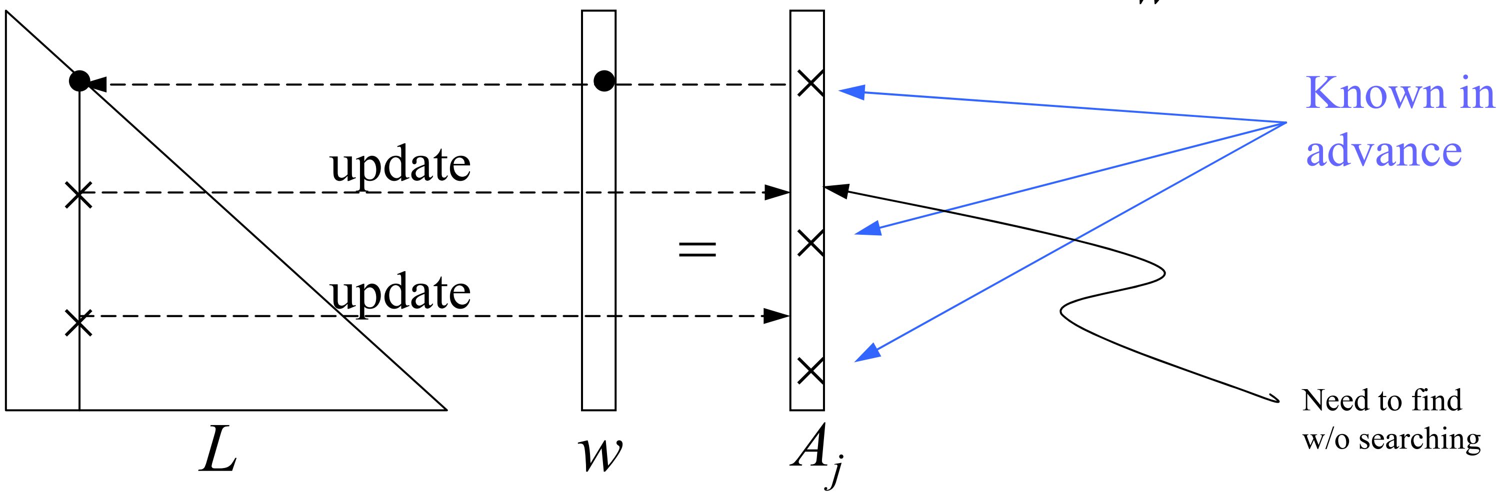
$\Rightarrow y$ an z are typically very sparse

Example: Model pla85900 (from TSP)

Constraints	85900
Variables	144185
Average |y|	15.5

$A_B =$ 

Triangular solve: $Lw = A_j \quad (A_B y = L(\underbrace{Uy}_{w}) = A_j)$



Graph structure: Define an acyclic digraph $D = (\{1,\ldots,m\}, E)$ where $(i,j) \in E \Leftrightarrow l_{ij} \neq 0$ and $i \neq j$.

Solving using *D*: Let $X = \{i \in V\colon A_{ij} \neq 0\}$. Compute

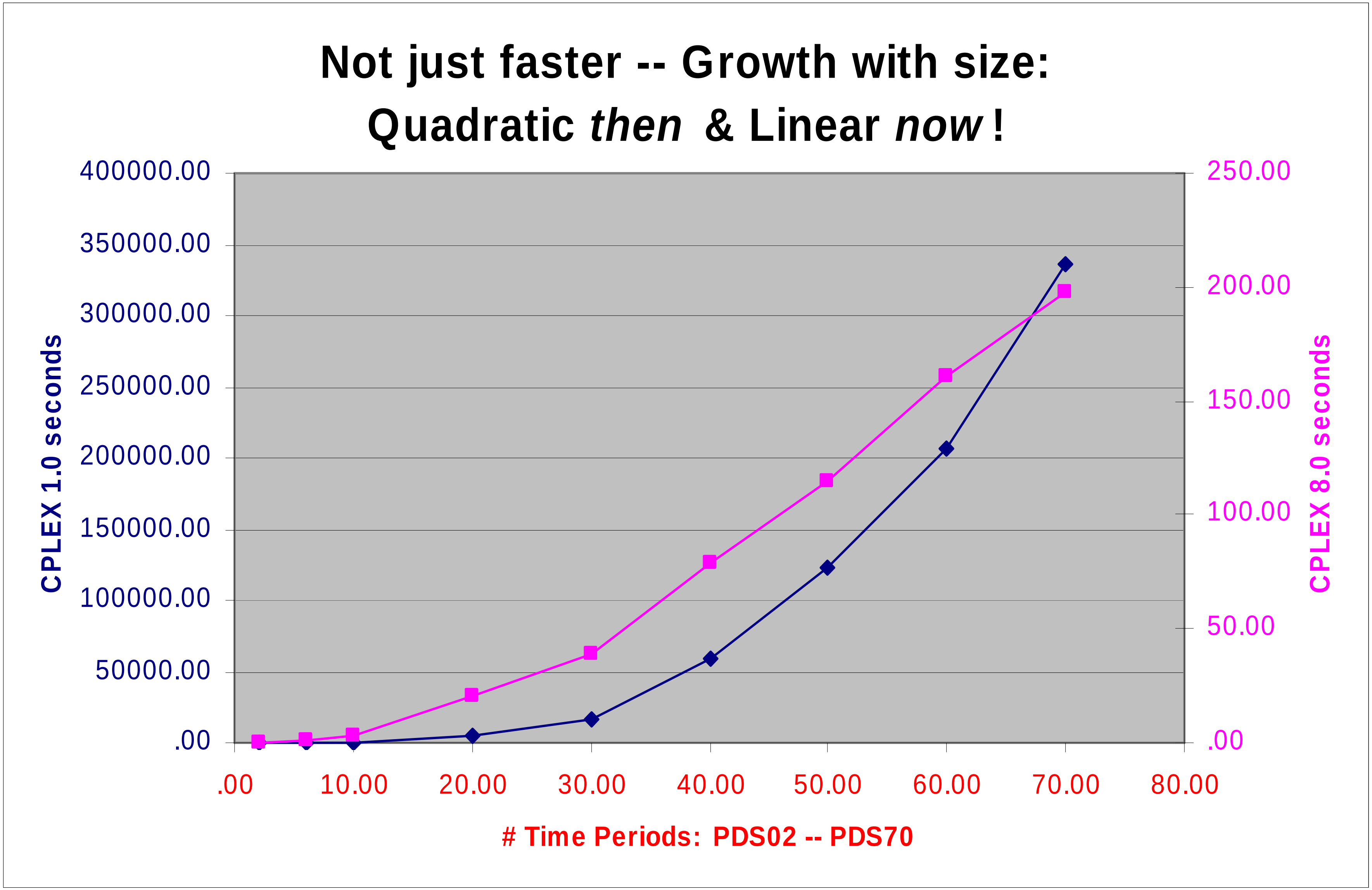
$$\underline{X} = \{j \in V\colon \exists \text{ a directed path from } j \text{ to } X\}.$$

$\underline{X}$ can be computed in time linear in $|E(\underline{X})| + |\underline{X}|$.

PDS Models

"Patient Distribution System": Carolan, Hill, Kennington, Niemi, Wichmann, *An empirical evaluation of the KORBX algorithms for military airlift applications*, Operations Research 38 (1990), pp. 240-248

		CPLEX1.0	CPLEX5.0	CPLEX8.0	SPEEDUP
MODEL	ROWS	1988	1997	2002	1.0→8.0
pds02	2953	**0.4**	**0.1**	**0.1**	**4.0**
pds06	9881	**26.4**	**2.4**	**0.9**	**29.3**
pds10	16558	**208.9**	**13.0**	**2.6**	**80.3**
pds20	33874	**5268.8**	**232.6**	**20.9**	**247.3**
pds30	49944	**15891.9**	**1154.9**	**39.1**	**406.4**
pds40	66844	**58920.3**	**2816.8**	**79.3**	**743.0**
pds50	83060	**122195.9**	**8510.9**	**114.6**	**1066.3**
pds60	99431	**205798.3**	**7442.6**	**160.5**	**1282.2**
pds70	114944	**335292.1**	**21120.4**	**197.8**	**1695.1**
		Primal Simplex	Dual Simplex	Dual Simplex	


Not just faster -- Growth with size:
Quadratic *then* & Linear *now*!
400000.00
350000.00
300000.00
250000.00
200000.00
150000.00
100000.00
50000.00
.00
CPLEX 1.0 seconds
250.00
200.00
150.00
100.00
50.00
.00
CPLEX 8.0 seconds
.00
10.00
20.00
30.00
40.00
50.00
60.00
70.00
80.00
Time Periods: PDS02 -- PDS70

Issue 4
Ratio Test and Finiteness

The "standard form" dual problem is

$$\begin{aligned} &\textit{Maximize} \quad b^T\pi \\ &\textit{Subject to} \quad A^T\pi + d = c \\ &\qquad\qquad\qquad\qquad\quad d \geq 0 \end{aligned}$$

Feasibility means

$$\boldsymbol{d \geq 0}$$

However, in practice this condition is replaced by

$$\boldsymbol{d \geq -\varepsilon\, e}$$

where $e^T=(1,\ldots,1)$ and $\varepsilon=10^{-6}$. Reason: Degeneracy. In 1972 Paula Harris proposed suggested exploiting this fact to improve numerical stability.

(Issue 4 – Ratio test & finiteness)

$\boxed{\text{STD. RATIO TEST}}$ $j_{enter} = argmin\{D_j / \alpha_j : \alpha_j > 0\}$

Motivation: Feasibility $\Rightarrow$ step length θ satisfies

$$D_N - \theta\alpha_N \geq 0$$

However, the bigger the step length, the bigger the change in the objective. So, we choose

$$\theta_{max} = min\{D_j/\alpha_j : \alpha_j > 0\}$$

Using ε, we have

$$\theta^{\varepsilon}_{max} = min\{(D_j+\varepsilon)/\alpha_j : \alpha_j > 0\} > \theta_{max}$$

$\boxed{\text{HARRIS RATIO TEST}}$ $j_{enter} = argmax\{\alpha_j : D_j / \alpha_j \leq \theta^{\varepsilon}_{max}\}$

(Issue 4 – Ratio test & finiteness)

- **Advantages**
  - Numerical stability – α_{jenter} = "pivot element"
  - Degeneracy – Reduces # of 0-length steps
- **Disadvantage**
  - $D_{jenter} < 0 \Rightarrow$ objective goes in wrong direction
- **Solution: BOUND SHIFTING**
  - If $D_{jenter} < 0$, we replace the lower bound on d_{jenter} by something less than its current value.
  - Note that this shift changes the problem and must be removed: 5% of cases, this produces dual infeasibility $\Rightarrow$ process is iterated.

Example: Bound-Shifting Removal

```
Problem 'pilot87.sav.gz' read.
Reduced LP has 1809 rows, 4414 columns, and 70191 nonzeros.

Iteration log . . .
Iteration:     1   Scaled dual infeas =             0.697540
Iteration:   733   Scaled dual infeas =             0.000404
Iteration:   790   Dual objective     =          -185.892207
...
Iteration: 16326   Dual objective     =           302.786794
Removing shift (3452).
Iteration: 16417   Scaled dual infeas =             0.207796
Iteration: 16711   Scaled dual infeas =             0.000021
Iteration: 16726   Dual objective     =           296.758656
Elapsed time =  104.36 sec. (17000 iterations).
Iteration: 17072   Dual objective     =           300.965492
...
Iteration: 17805   Dual objective     =           301.706409
Removing shift (76).
Iteration: 17919   Scaled dual infeas =             0.000060
Iteration: 17948   Dual objective     =           301.708660
Elapsed time =  114.42 sec. (18000 iterations).
Removing shift (10).
Iteration: 18029   Scaled dual infeas =             0.000050
Iteration: 18039   Dual objective     =           301.710058
Removing shift (1).

Dual simplex - Optimal:  Objective =   3.0171034733e+002
Solution time =  116.44 sec.  Iterations = 18095 (1137)
```

Shift 1: $\varepsilon = 10^{-7}$

Shift 2: $\varepsilon = 10^{-8}$

Shift 3: $\varepsilon = 10^{-9}$

(Issue 4 – Ratio test & finiteness)

Finiteness: Bound shifting is closely related to the "perturbation" method employed in CPLEX if no progress is being made in the objective.

"No progress" $\Rightarrow$

$$d_j \geq -\varepsilon \qquad j = 1,\ldots,n$$

is replaced by

$$d_j \geq -\varepsilon - \varepsilon_j \qquad j = 1,\ldots,n,$$

where ε_j is random uniform on $[0,\varepsilon]$.

Issue 5
Bound Flipping

- **A basis is given by a triple (B,L,U)**
  - L = non-basics at lower bound: Feasibility $D_L \geq 0$
  - U = non-basics at upper bound: Feasibility $D_U \leq 0$
- **Ratio test:** Suppose $\boldsymbol{X_{Bi}}$ is the leaving variable, and the step length is blocked by some variable $\boldsymbol{d_j}$**,** $\boldsymbol{j \in L}$**,** that is about to become negative and such that $u_j < +\infty$:
  - **Flipping means:** Move j from L to U.
  - **Check:** Do an update to see if X_{Bi} is still favorable (just as we did in Phase I!)
- Can combine many iterations into a single iteration.

Example: Bound Flipping

w/o flipping

```
Problem 'fit2d.sav.gz' read.
Initializing dual steep norms . . .

Iteration log . . .
Iteration:     1   Dual objective     =        -80412.550000
Perturbation started.
Iteration:   203   Dual objective     =        -80412.550000
Iteration:  1313   Dual objective     =        -80412.548666
Iteration:  2372   Dual objective     =        -77028.548350
Iteration:  3413   Dual objective     =        -71980.245530
Iteration:  4316   Dual objective     =        -70657.605570
Iteration:  5151   Dual objective     =        -68994.477061
Iteration:  5820   Dual objective     =        -68472.659371
Removing perturbation.

Dual simplex - Optimal:  Objective =  -6.8464293294e+004
Solution time =   18.74 sec.  Iterations = 5932 (0)
```

w/ flipping

```
Problem 'fit2d.sav.gz' read.
Initializing dual steep norms . . .

Iteration log . . .
Iteration:     1   Dual objective     =        -77037.550000

Dual simplex - Optimal:  Objective =  -6.8464293294e+004
Solution time =    1.88 sec.  Iterations = 201 (0)
```