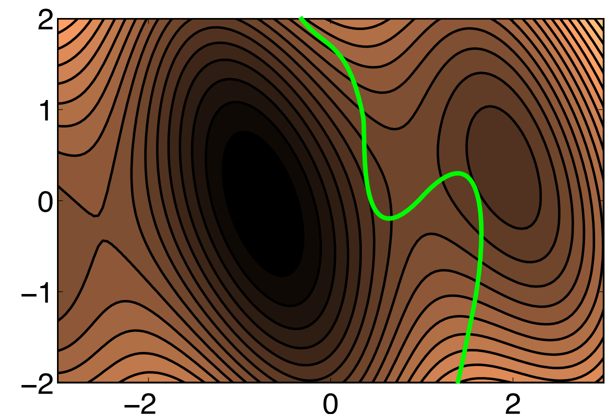
Linear programs

10-725 Optimization
Geoff Gordon
Ryan Tibshirani

Review



- Newton w/ equality constraints
- Examples:
  - bundle adjustment
  - MLE in exponential families
- Convergence: damped phase, quadratic phase
- Compare: Newton, FISTA, (stoch) (sub)gradient
- Variations: trust region, quasi-Newton, Gauss-Newton, Levenberg-Marquardt

Variations: Fisher scoring

- Recall Newton in exponential family

$$\mathrm{Var}[x \mid \theta] d\theta = E[x \mid \theta] - \bar{x}$$

- Can use this formula in place of Newton, even if not an exponential family
    - descent direction, even w/ no regularization
    - "Hessian" is independent of data
    - often a wider radius of convergence than Newton
    - can be superlinearly convergent

Administrivia

- HW2 due now
- HW3 out tonight (hopefully)
- Final project update
  - project milestone report requirements on web site
  - final poster session (3:30–6:30 12/12 NSH Atrium, 3PM setup)
  - set up meetings w/ TA mentors

Linear programs

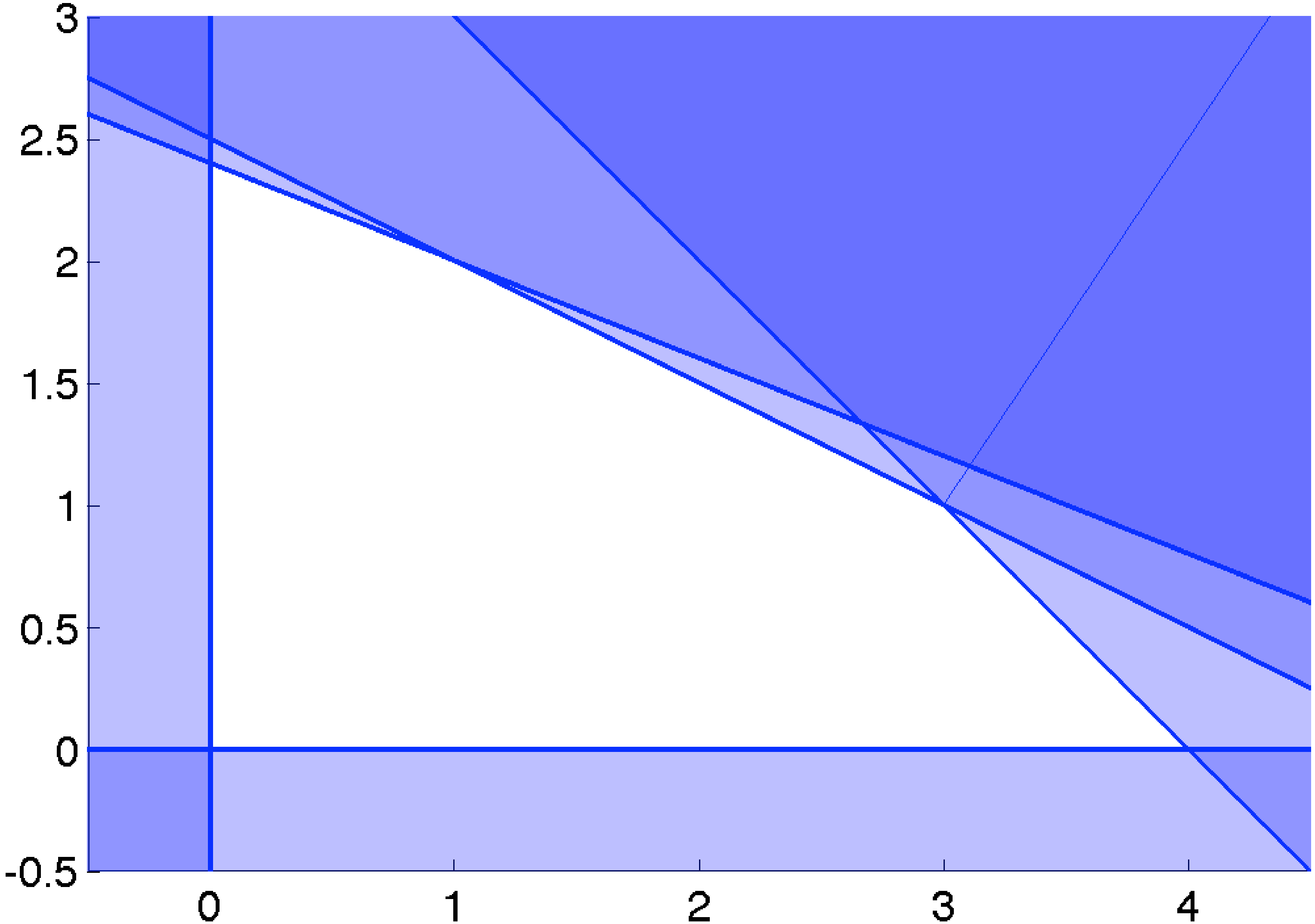
- n variables: $x = (x_1, x_2, \ldots, x_n)^T$
  - ranges: $[l_i, u_i]$
- Objective:
- m constraints (equality or inequality):
- Example:

Sketching an LP

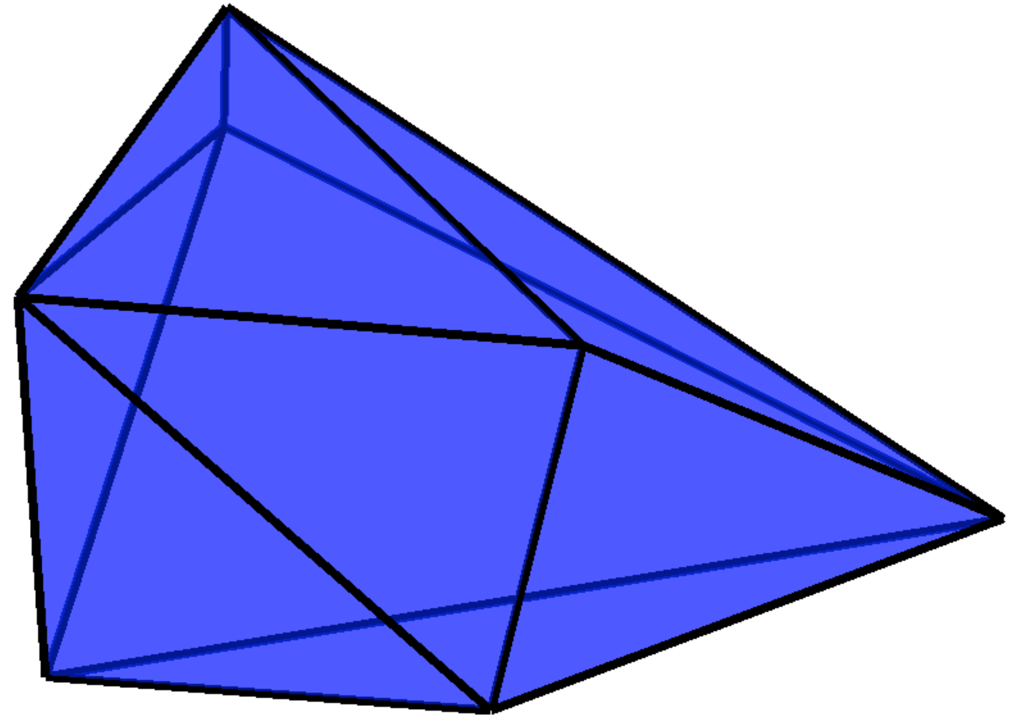
max $2x+3y$ s.t.

$$x + y \le 4$$
$$2x + 5y \le 12$$
$$x + 2y \le 5$$
$$x, y \ge 0$$

Did the prof get it right?



Polyhedra



- hull({points}) or
  ∩({halfspaces})
- Vertices, edges, faces
  - in general: d-faces
    - n vars: d-face = set of feasible points that make independent halfspace constrs tight
    - therefore, dimensionality =
  - n vars, m≥n halfspaces: can have -faces thru -faces

Matrix notation

- For a vector of variables v and a constant matrix A and vector b,
  - $Av \leq b$ [componentwise]
- Objective: $c^T v$
- E.g.:
  - A = b = c =

max $2x+3y$ s.t.
$x + y \leq 4$
$2x + 5y \leq 12$
$x + 2y \leq 5$
$x, y \geq 0$

Finding the optimum

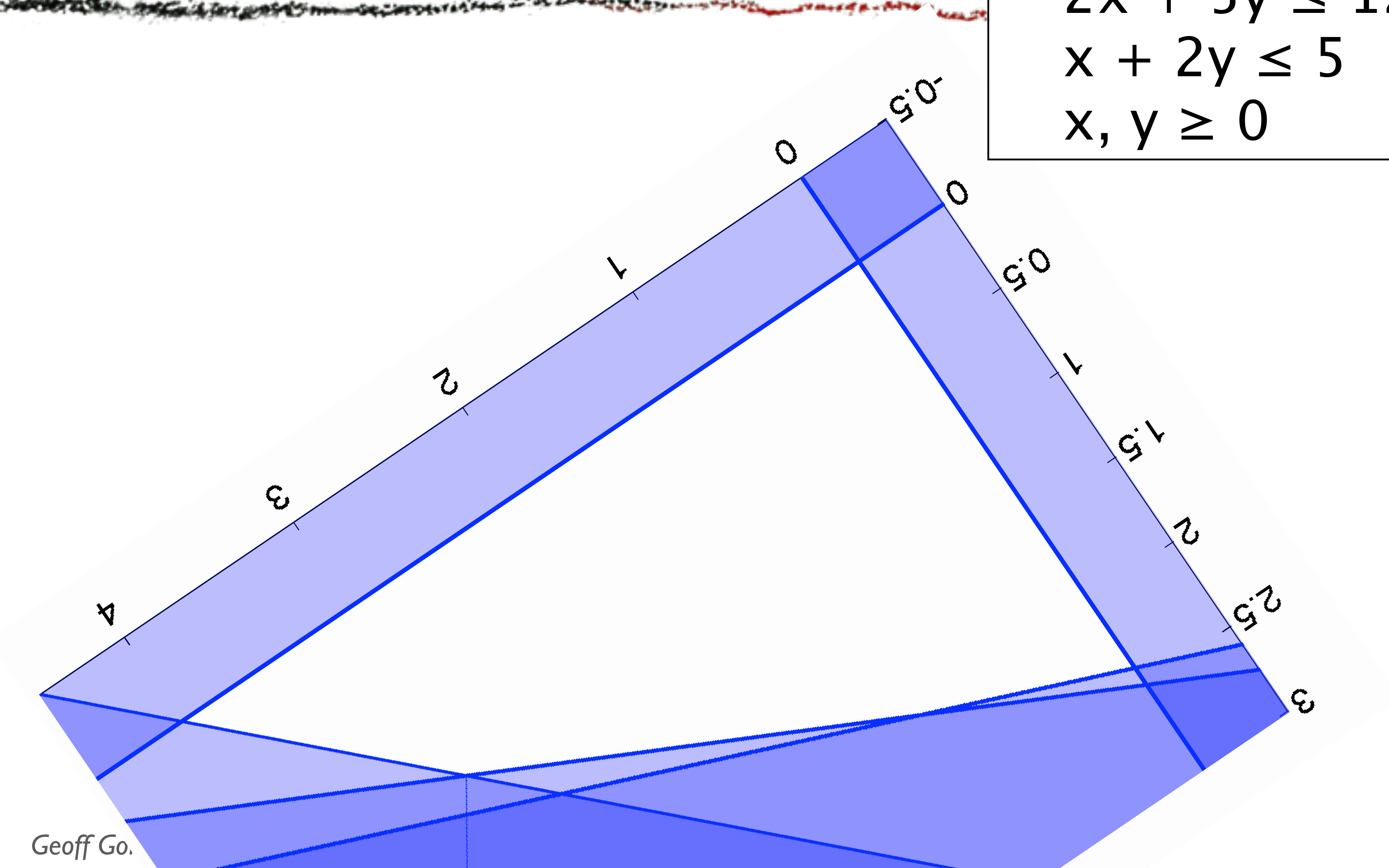
max $2x+3y$ s.t.

$x + y \leq 4$

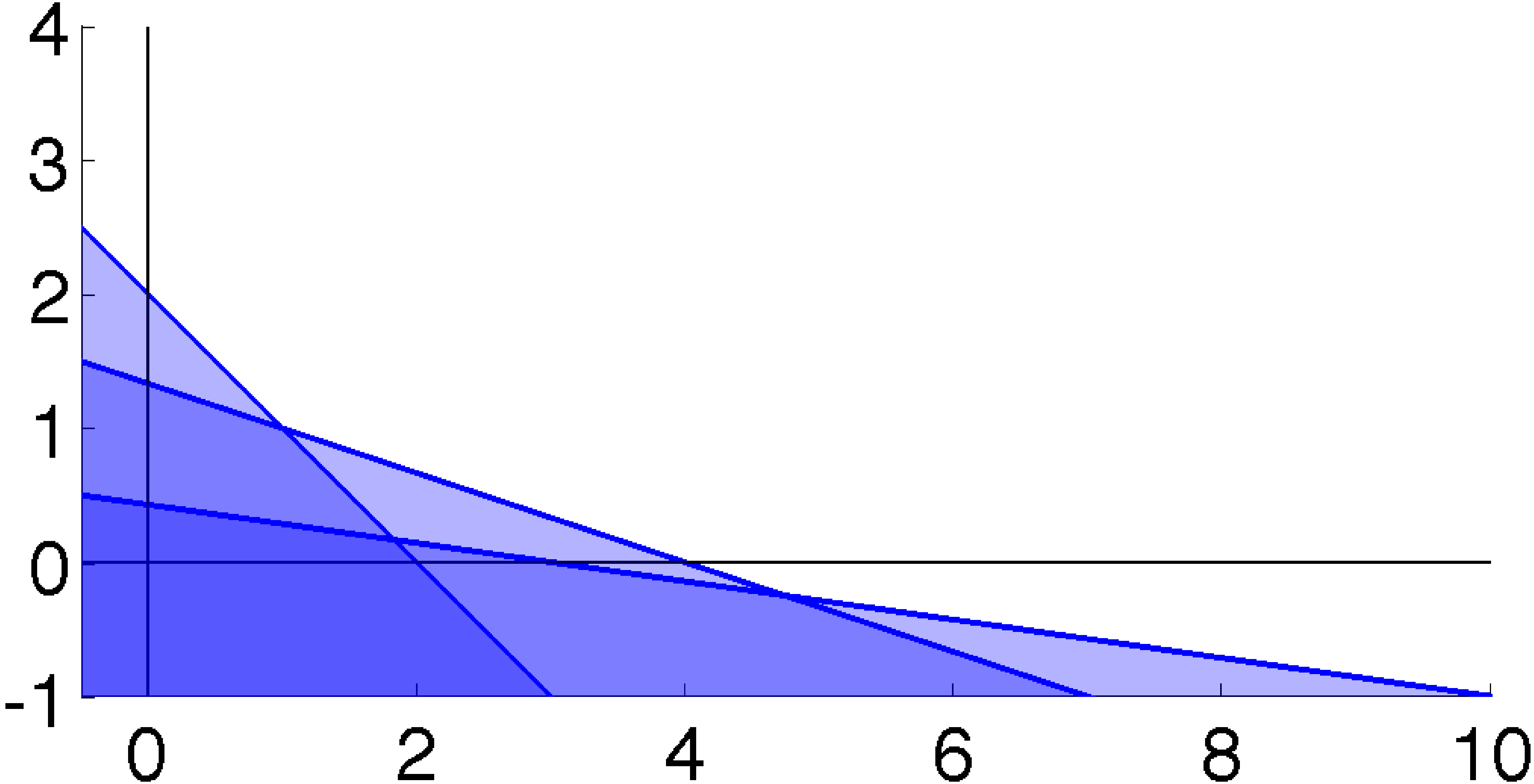
$2x + 5y \leq 12$

$x + 2y \leq 5$

$x, y \geq 0$



Where's my ball?



Unhappy ball

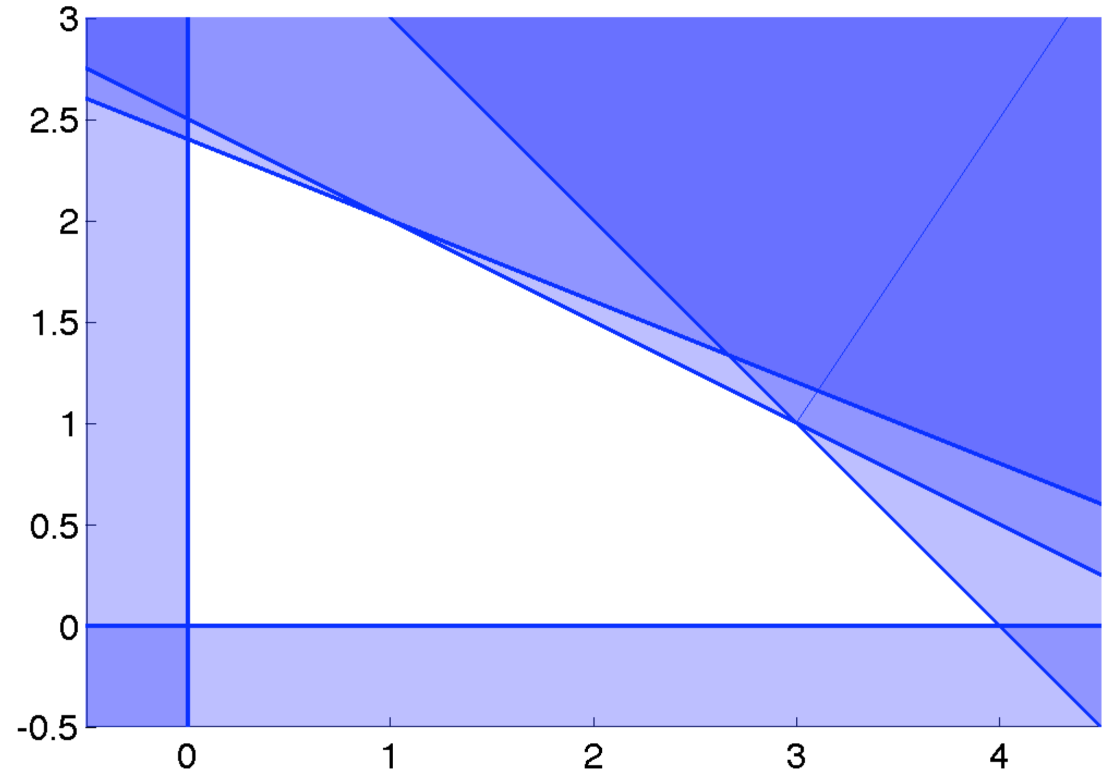
- max $2x + 3y$ subject to
- $x \geq 5$
- $x \leq 1$

Convention

- min over empty set =
- max over empty set =
- Adding an element always
    -
    -

Linear feasibility

- find (x, y) s.t.
  - $x + y \leq 4$
  - $2x + 5y \leq 12$
  - $x + 2y \leq 5$
  - $x, y \geq 0$

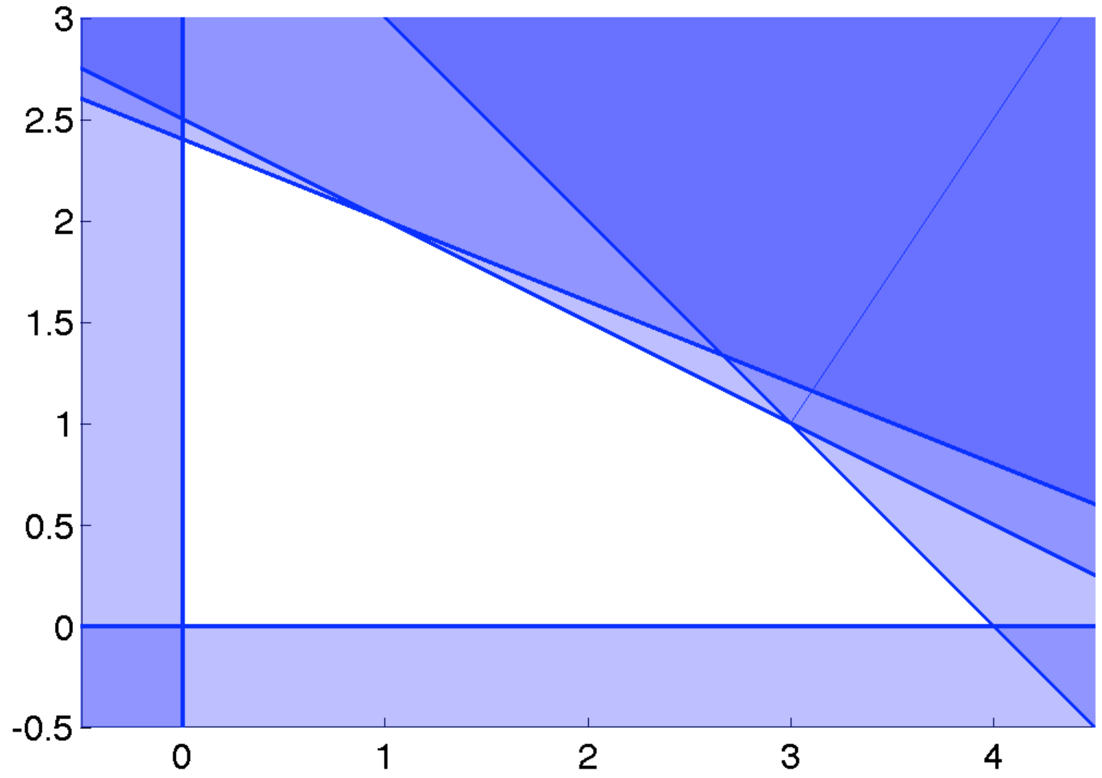


- Any easier than LP?

Binary search

- find (x, y) s.t.
  - $x + y \leq 4$
  - $2x + 5y \leq 12$
  - $x + 2y \leq 5$
  - $x, y \geq 0$

vs. max $2x+3y$ s.t. ↑



Transforming LPs

- Getting rid of inequalities (except variable bounds)

$$x + y \leq 4$$

- Getting rid of equalities

$$x + 2y = 4$$

Transforming LPs

- Getting rid of free vars

$$\begin{aligned} \max\ x + y \text{ s.t.} \\ 2x + y \le 3 \\ y \ge 0 \end{aligned}$$

- Getting rid of bounded vars

$$x \in [2, 5]$$

Standard form LP

$$\max\ 2x+3y \text{ s.t.}$$
$$x + y \le 4$$
$$2x + 5y \le 12$$
$$x + 2y \le 5$$
$$x, y \ge 0$$

- all variables are nonnegative
- all constraints are equalities
- E.g.: max c^Tq s.t. $Aq = b$, $q \ge 0$
  - $q = [x\ y\ u\ v\ w]^T$

tableau

Objective in tableau

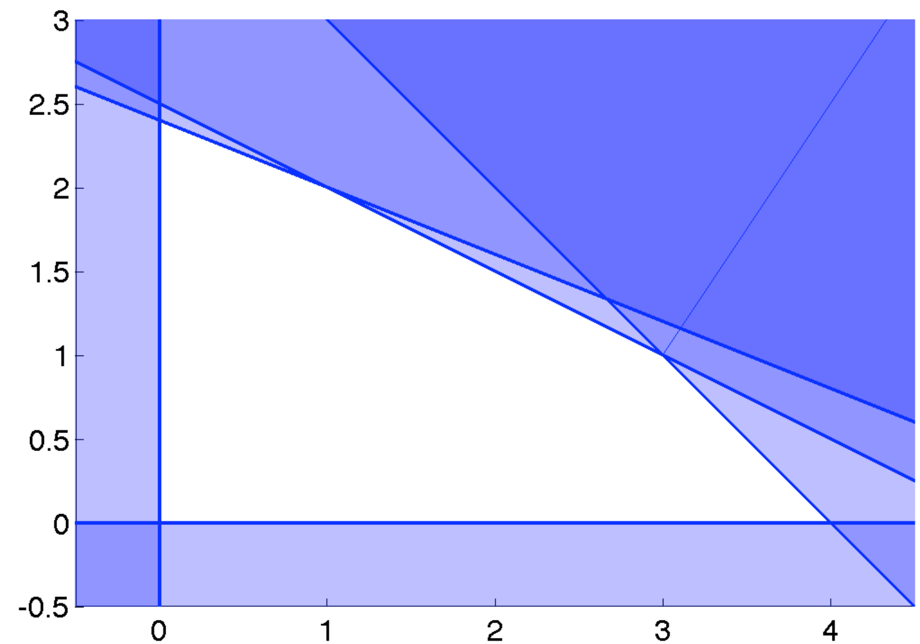
max $2x+3y$ s.t.
$x + y \leq 4$
$2x + 5y \leq 12$
$x + 2y \leq 5$
$x, y \geq 0$

- Add an extra variable z
  - constrain it to = the objective

x	y	u	v	w	z	RHS
1	1	1	0	0	0	4
2	5	0	1	0	0	12
1	2	0	0	1	0	5
–2	–3	0	0	0	1	0

Standard v. inequality forms

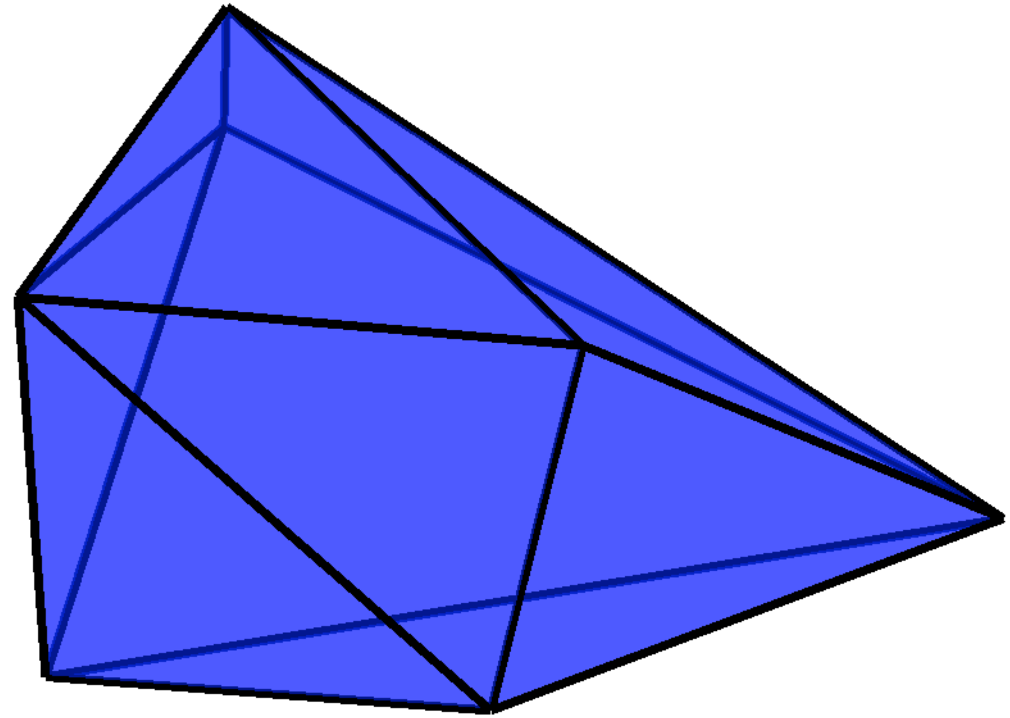
- max $2x+3y$ s.t.
  - $x + y \leq 4$
  - $2x + 5y \leq 12$
  - $x + 2y \leq 5$
  - $x, y \geq 0$



- or s.t.
  - $x + y + u = 4$
  - $2x + 5y + v = 12$
  - $x + 2y + w = 5$
  - $x, y, u, v, w \geq 0$

if std fm has *n* vars, *m* eqns then ineq form has *n–m* vars and *m*+(*n–m*)=*n* ineqs (here *m* = 3, *n* = 5)

Faces in standard form



- Inequality form
  - n vars, m≥n halfspaces: can have 0-faces thru n-faces
  - d-face makes n–d inequalities tight
- Standard form
  - n nonneg. vars, m≤n equalities: -faces thru -faces

Why is standard form useful?

- Can take linear combinations of constraints
- E.g., $x + 2y = 4$ & $2x + 3y = 5$

- Easy to manipulate via row operations
- Easy to find corners by Gaussian elimination

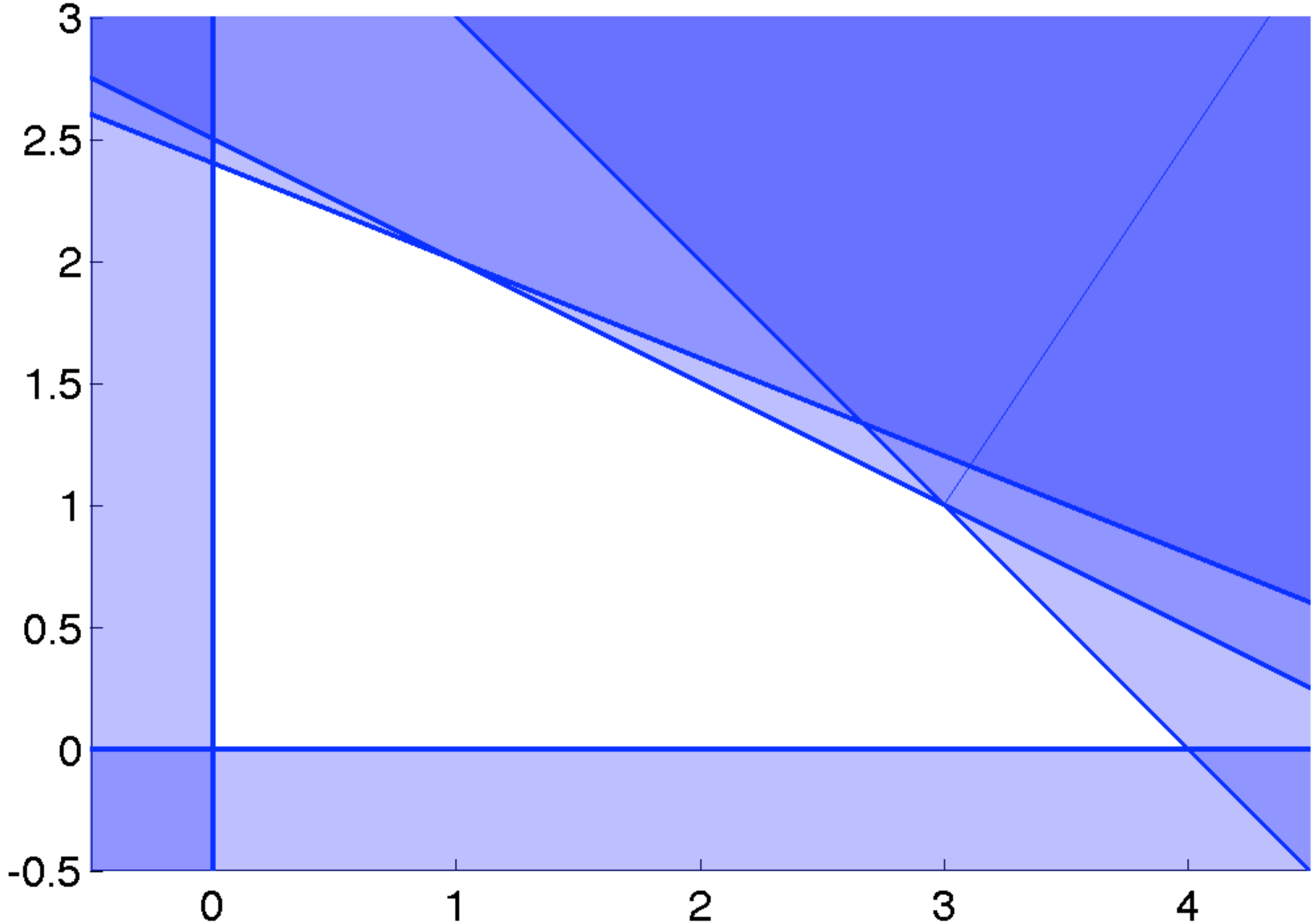
Example

1	1	1	0	0	4	set x, y = 0
2	5	0	1	0	12	
1	2	0	0	1	5	

1	1	1	0	0	4	set v, w = 0
2	5	0	1	0	12	
1	2	0	0	1	5	

1	1	1	0	0	4	set x, u = 0
2	5	0	1	0	12	
1	2	0	0	1	5	

What happened?



Row operations

- Can replace any row with linear combination of existing rows
  - as long as we don't lose independence
- Eliminate x from 2nd and 3rd rows

A					b
1	1	1	0	0	4
2	5	0	1	0	12
1	2	0	0	1	5

Presto change-o

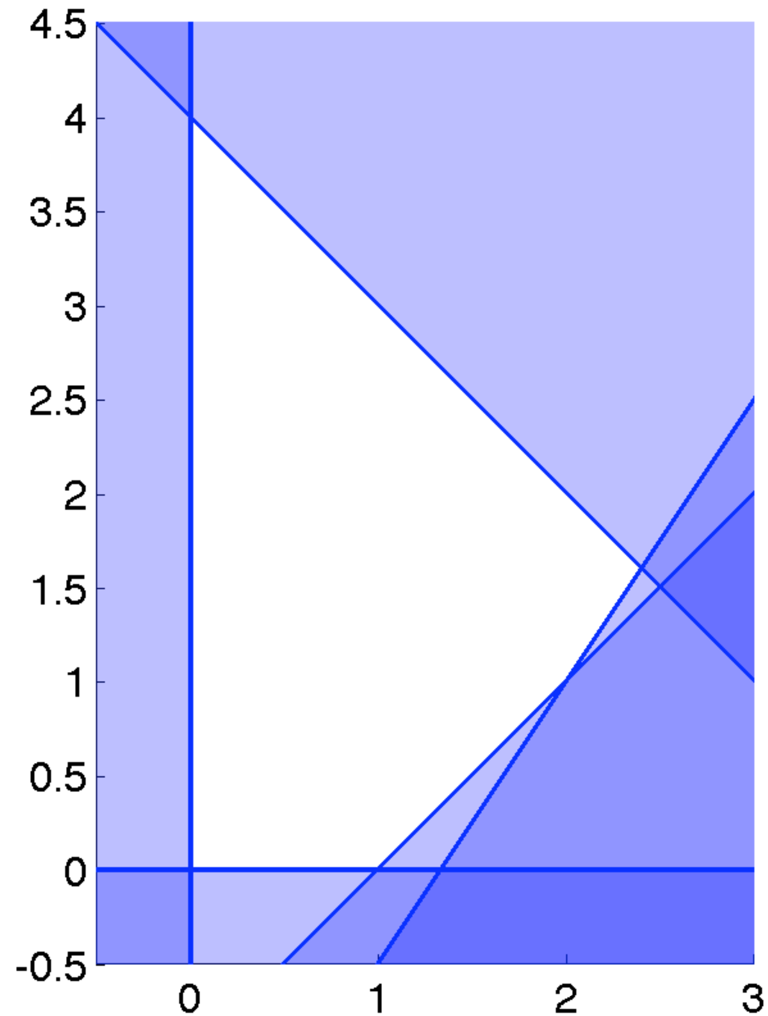
x	y	u	v	w	RHS
1	1	1	0	0	4
0	3	−2	1	0	4
0	1	−1	0	1	1

- Which are the slacks now?
  -
  -
- Eliminating x from all but one constr:
- Constraint we used to eliminate x:

The "new" LP

objective: was z = 2x + 3y

x	y	u	v	w	**z**	RHS
1	1	1	0	0	0	4
0	3	−2	1	0	0	4
0	1	−1	0	1	0	1
−2	−3	0	0	0	1	.



$y + u \leq 4$

$3y - 2u \leq 4$

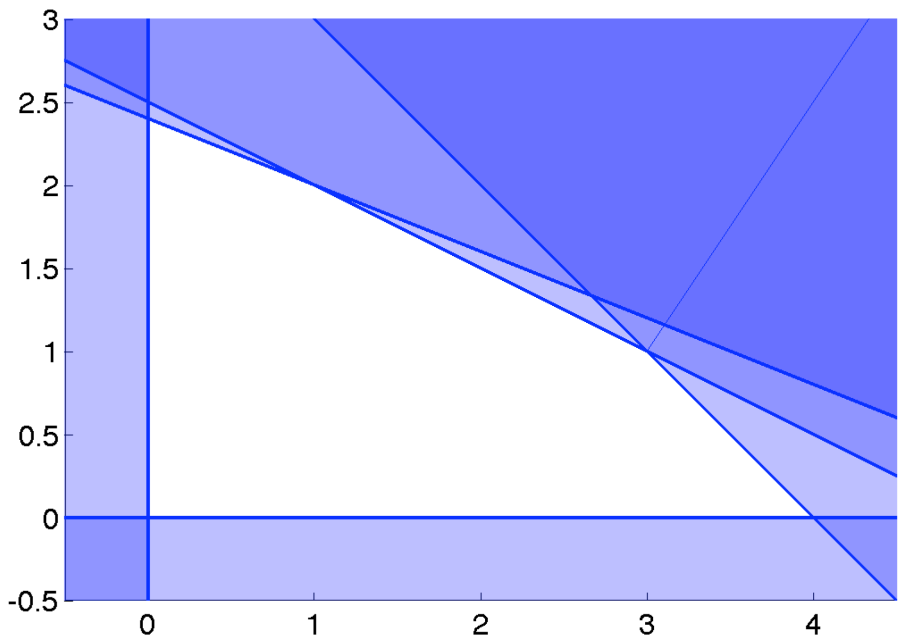
$y - u \leq 1$

$y, u \geq 0$

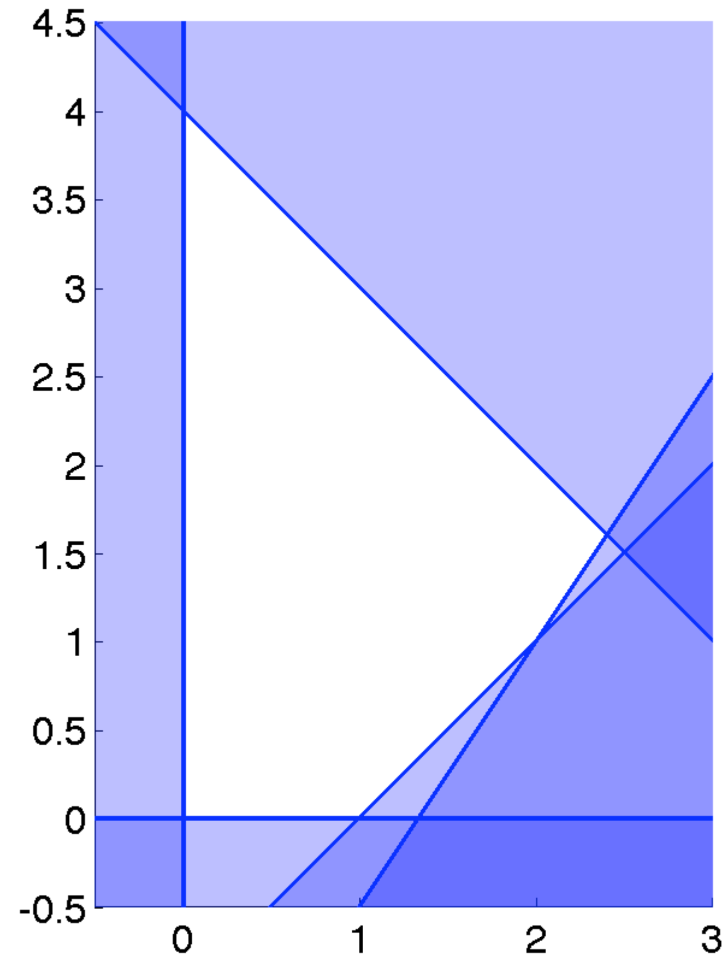
Sketching standard form

- Drop the slacks, sketch in inequality form
- May be several ways

$$\begin{array}{cccccc} 1 & 1 & 1 & 0 & 0 & 4 \\ 0 & 3 & -2 & 1 & 0 & 4 \\ 0 & 1 & -1 & 0 & 1 & 1 \end{array}$$



$$\begin{array}{cccccc} 1 & 1 & 1 & 0 & 0 & 4 \\ 2 & 5 & 0 & 1 & 0 & 12 \\ 1 & 2 & 0 & 0 & 1 & 5 \end{array}$$



What if there aren't slacks?

- Use row ops to ***make*** some:
  - $u, v, w \geq 0$
  - $u + 2v + w = 3$
  - $3u + v - w = 5$

Matlab version

```
A = [1  2  1;
     3  1 -1];
b = [3;5];
A(:,1:2) \ A
ans =
    1.0000         0   -0.6000
         0    1.0000    0.8000

A(:,1:2) \ b
ans =
    1.4000
    0.8000
```

Matlab with z

max z=2x+3y s.t.
x + y + u = 4
2x + 5y + v = 12
x + 2y + w = 5
x, y, u, v, w ≥ 0

- always pick z's column (here, col 6)
- remember z is unconstrained

A and b

x	y	u	v	w	z	RHS
1	1	1	0	0	0	4
2	5	0	1	0	0	12
1	2	0	0	1	0	5
−2	−3	0	0	0	1	0

result = A(:,[1 4 5 6]) \ [A b]

x	y	u	v	w	z	RHS
1	1	1	0	0	0	4
0	3	−2	1	0	0	4
0	1	−1	0	1	0	1
0	−1	2	0	0	1	8

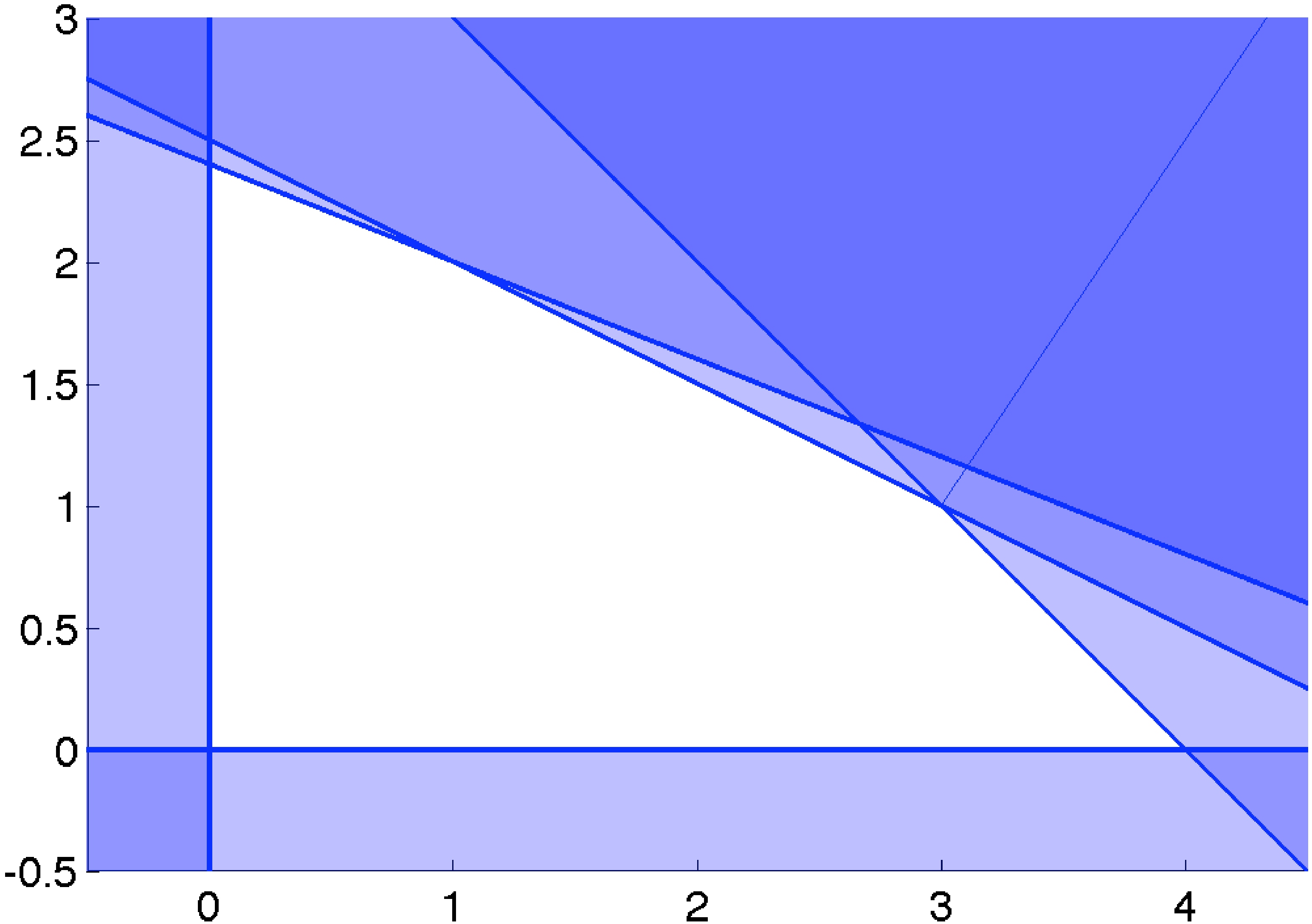
Basis

- {u, v, w, z} and {x, v, w, z} are ***bases***
  - elements = basic variables (always m of them)
  - easy to write values of basic variables in terms of non-basic ones
  - e.g., set x=y=0
  - e.g., set y=u=0

x	y	u	v	w	z	RHS
1	1	1	0	0	0	4
2	5	0	1	0	0	12
1	2	0	0	1	0	5
−2	−3	0	0	0	1	0

x	y	u	v	w	z	RHS
1	1	1	0	0	0	4
0	3	−2	1	0	0	4
0	1	−1	0	1	0	1
0	−1	2	0	0	1	8

Basic solutions



Bases ↔ corners

- Std form: n vars, m equations
  - fix ineq form: n–m vars, n ineqs
- Pick a basis B for std form
  - m basic vars (≥ 0), n–m nonbasic (set to 0)
- Each nonbasic var yields a tight inequality
  - var is either a slack or explicit in ineq fm
    - explicit: one of n–m "trivial" (x≥0) ineqs tight
    - slack: one of m "real" ineqs tight
- Ineq fm: n–m vars and n–m tight ineqs → corner

What if we can't pick basis?

- E.g., suppose A doesn't have full row rank
  - can't pick m linearly independent cols
- Ex:

What if we can't pick basis?

- E.g., suppose fewer vars than constraints
  - A taller than it is wide, $m \geq n$
  - can't pick enough cols of A to make a square matrix
- Ex:

Nonsingular

- We can assume
  - $n \geq m$ (at least as many vars as constrs)
  - A has full row rank
- Else, drop rows (maintaining rank) until it's true
- Called ***nonsingular*** standard form LP

Naive (slooooow) algorithm

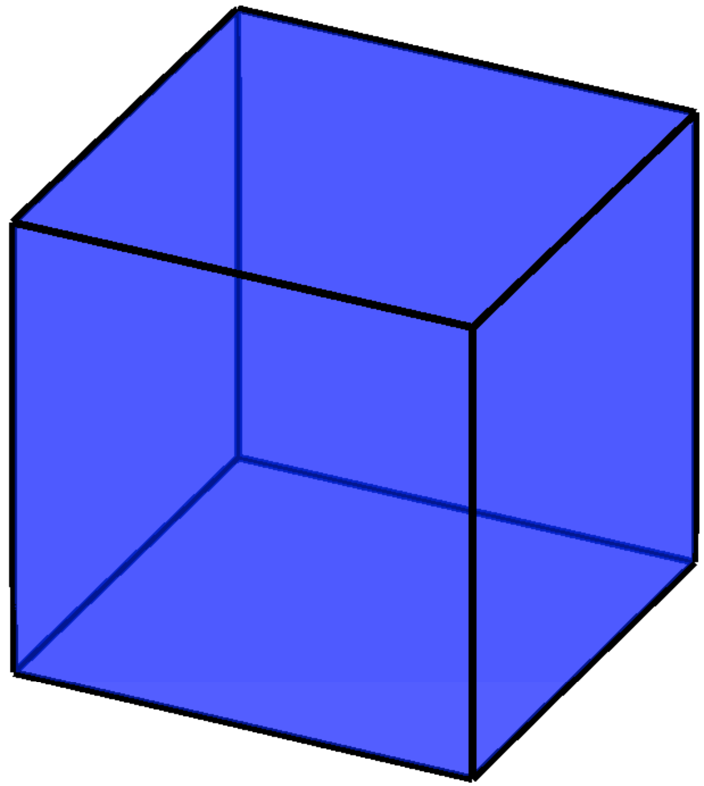
- Put in nonsingular standard form
- Iterate through all subsets of n vars
    - if m constraints, how many subsets?
- Check each for
    - full rank (“basis-ness”)
    - feasibility (RHS ≥ 0)
- If pass both tests, compute objective
- Maintain running winner, return at end

Improving our search

- Naive: enumerate all possible bases
- Smarter: maybe neighbors of good bases are also good?
- Simplex algorithm: repeatedly move to a neighboring basis to improve objective
  - continue to assume nonsingular standard form LP

Neighboring bases

- Two bases are ***neighbors*** if they share (m–1) variables
- Neighboring feasible bases correspond to vertices connected by an edge

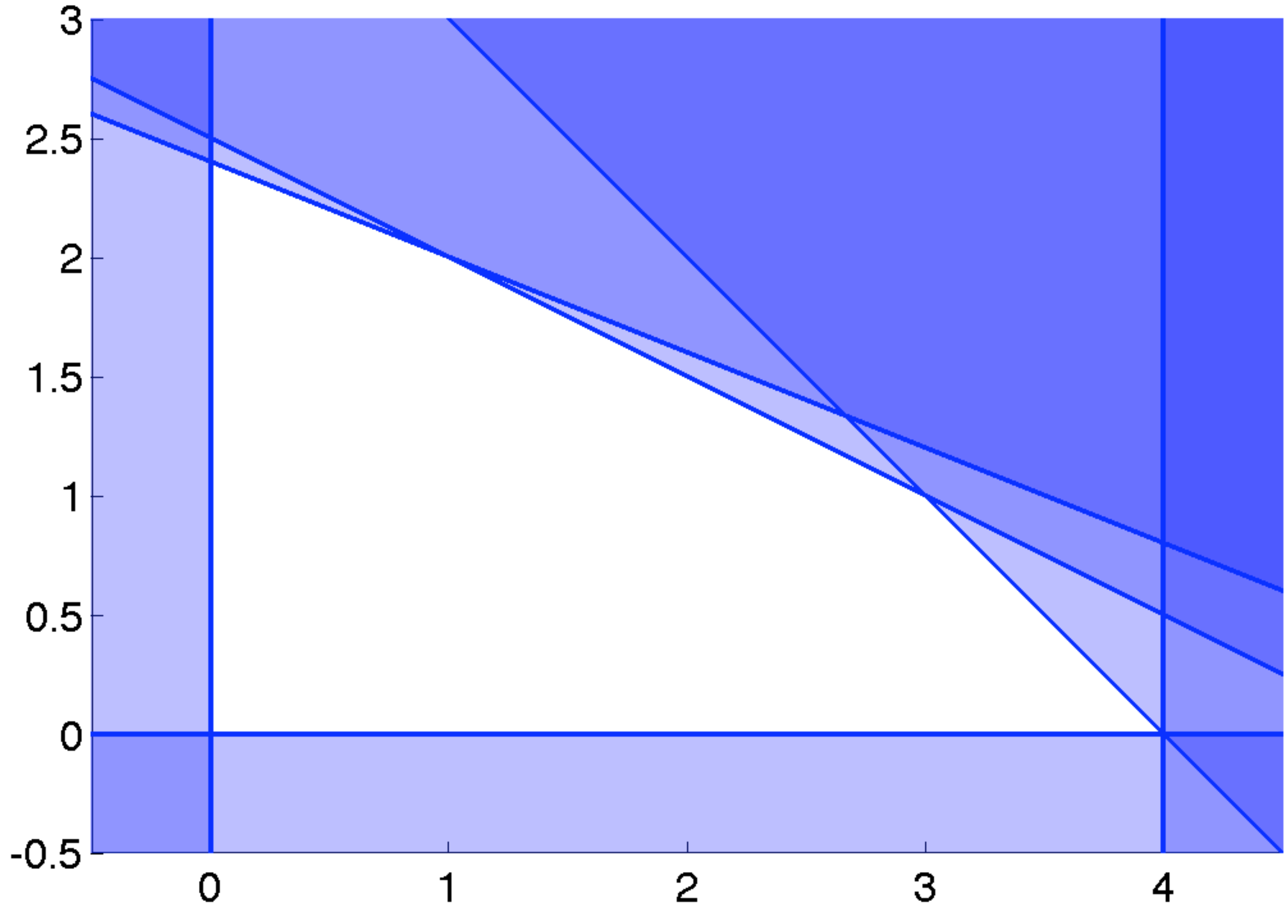


x	y	z	u	v	w	RHS
1	0	0	1	0	0	1
0	1	0	0	1	0	1
0	0	1	0	0	1	1

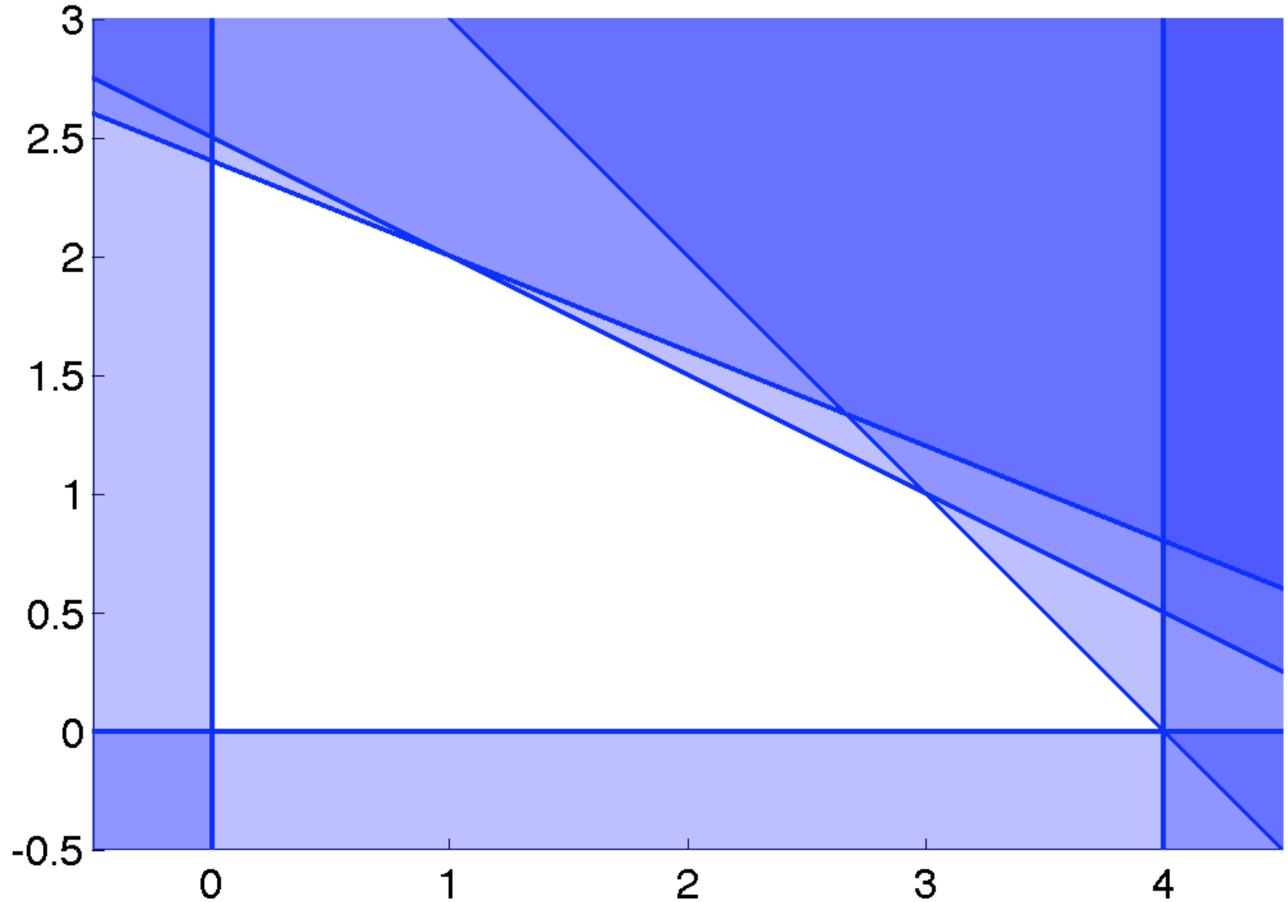
def'n: pivot, enter, exit

Example

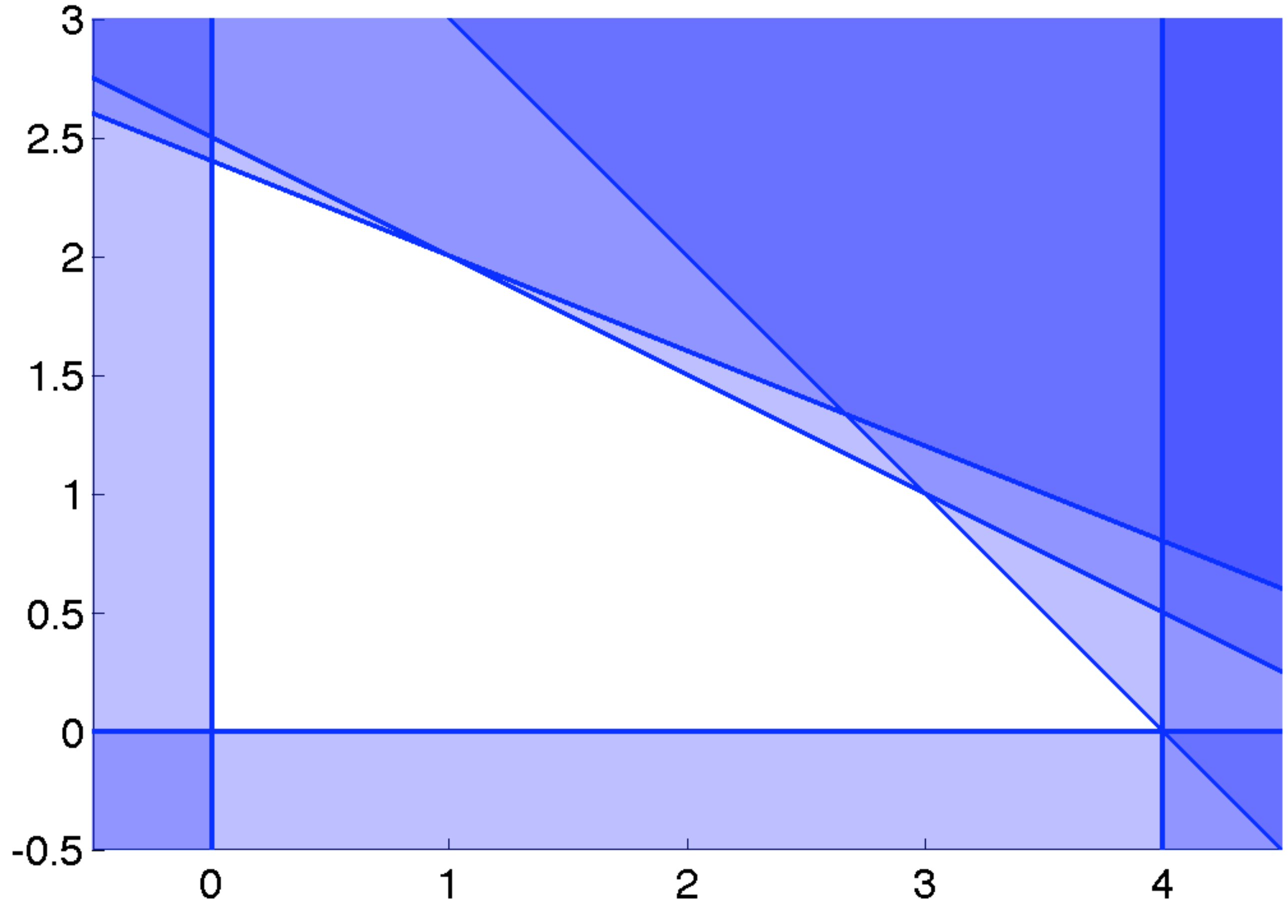
max z = 2x + 3y s.t.
$x + y \leq 4$
$2x + 5y \leq 12$
$x + 2y \leq 5$
$x \leq 4$



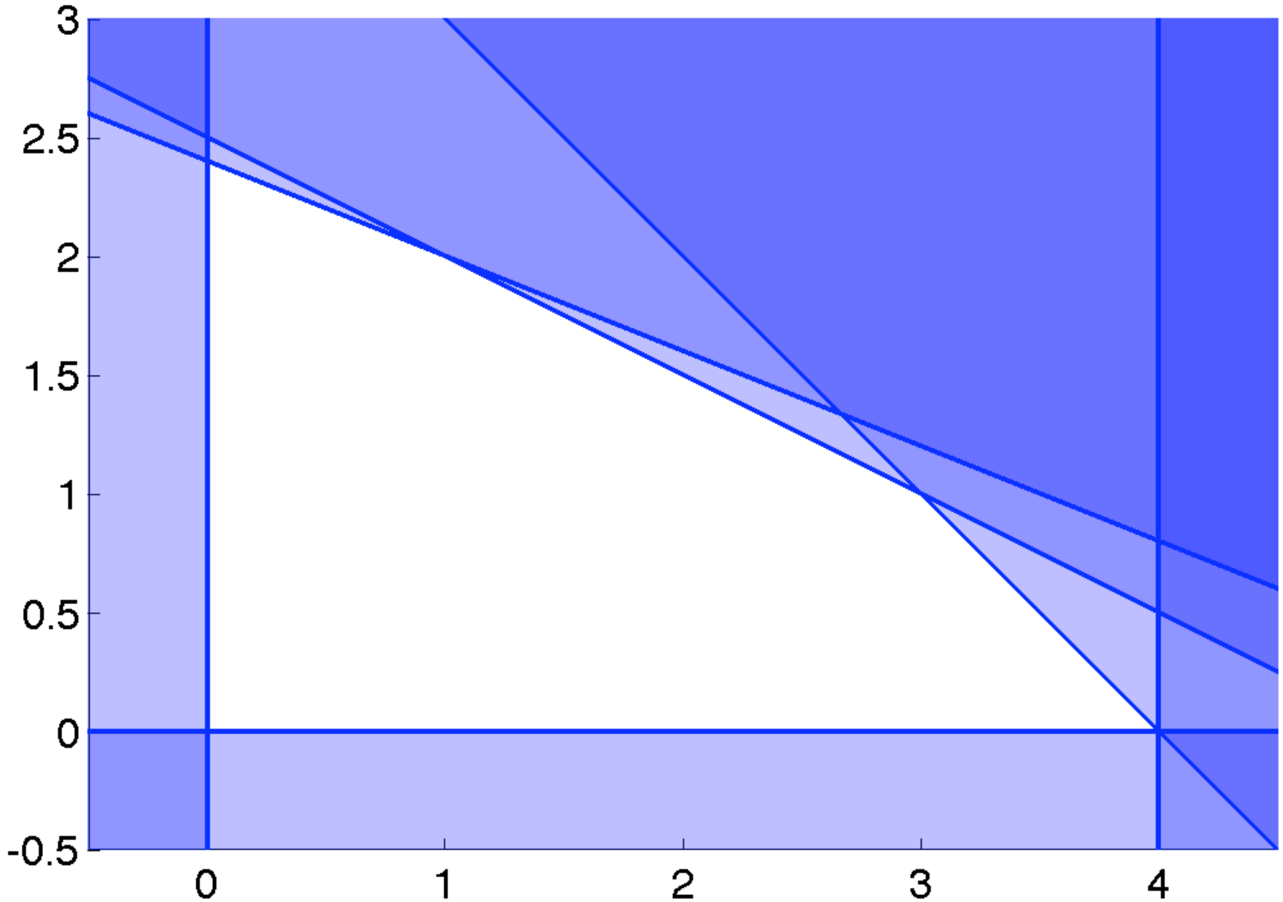
x	y	s	t	u	v	z	RHS
1	1	1	0	0	0	0	4
2	5	0	1	0	0	0	12
1	2	0	0	1	0	0	5
1	0	0	0	0	1	0	4
−2	−3	0	0	0	0	1	0



x	y	s	t	u	v	z	RHS
0.4	1	0	0.2	0	0	0	2.4
0.6	0	1	−0.2	0	0	0	1.6
0.2	0	0	−0.4	1	0	0	0.2
1	0	0	0	0	1	0	4
−0.8	0	0	0.6	0	0	1	7.2

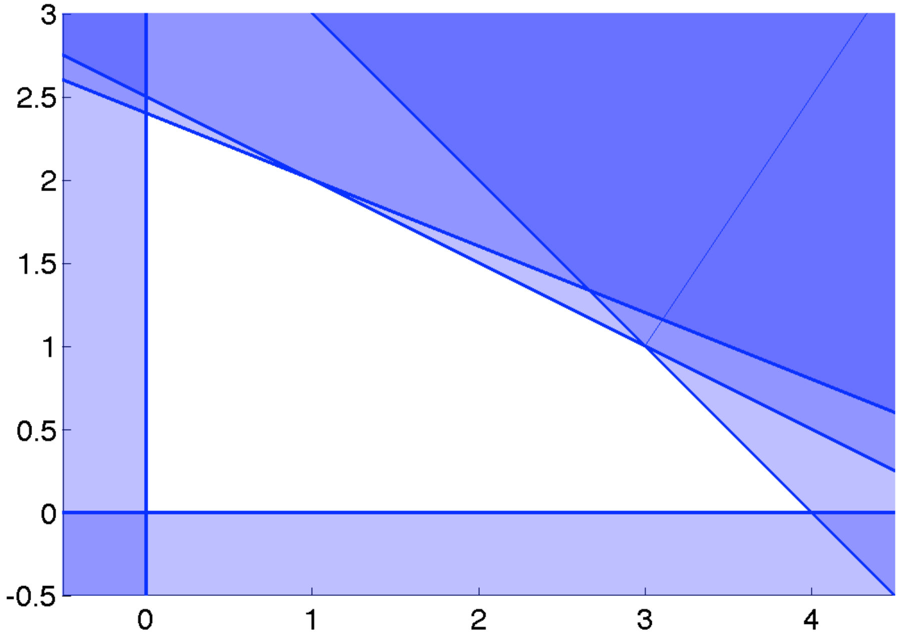


x	y	s	t	u	v	z	RHS
1	0	0	−2	5	0	0	1
0	1	0	1	−2	0	0	2
0	0	1	1	−3	0	0	1
0	0	0	2	−5	1	0	3
0	0	0	−1	4	0	1	8



x	y	s	t	u	v	z	RHS
1	0	2	0	−1	0	0	3
0	1	−1	0	1	0	0	1
0	0	1	1	−3	0	0	1
0	0	−2	0	1	1	0	1
0	0	1	0	1	0	1	9

Initial basis



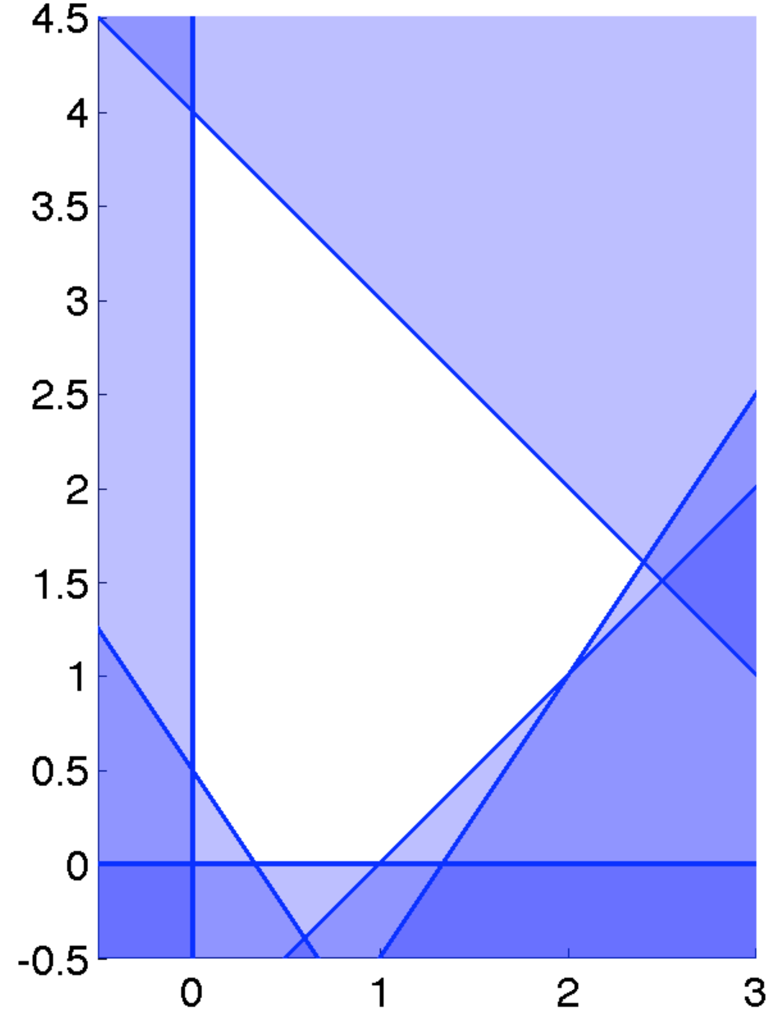
x	y	u	v	w	RHS
1	1	1	0	0	4
2	5	0	1	0	12
1	2	0	0	1	5

- So far, assumed we started w/ feasible basic solution—in fact, it was trivial to find one
- Not always so easy in general

Big M

```
0 ≤ x, y, s1..s6
max x - 2y
```

x	y	slacks				z	RHS
1	1	1	0	0	0	0	4
3	-2	0	1	0	0	0	4
1	-1	0	0	1	0	0	1
-3	-2	0	0	0	1	0	-1
-1	2	0	0	0	0	1	0



- Can make it easy: variant of slack trick
  - For each violated constraint, add var w/ coeff –1
  - Penalize in objective

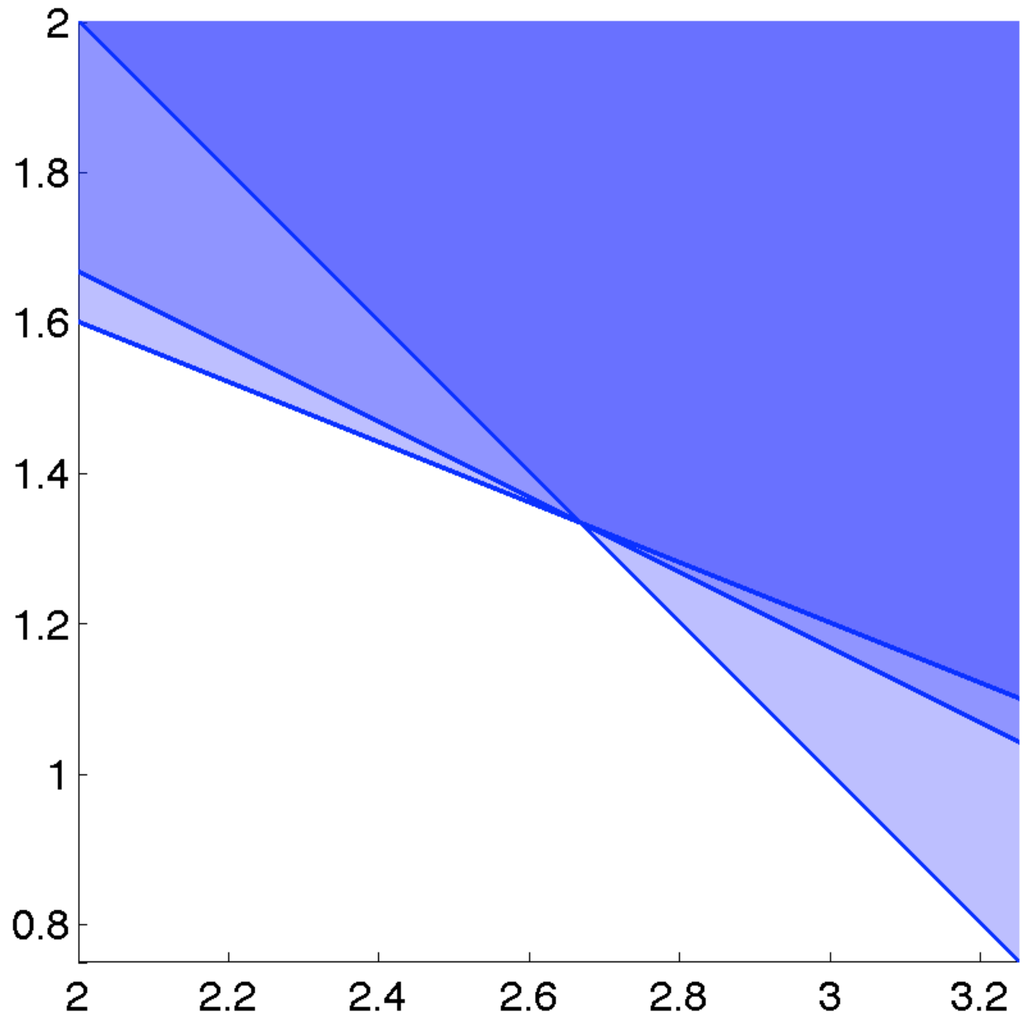
Simplex in one slide
(skipping degeneracy handling)

- Given a nonsingular standard-form max LP
- Start from a feasible basis and its tableau
  - big-M if needed
- Pick non-basic variable w/ coeff in objective ≤ 0
- Pivot it into basis, getting neighboring basis
  - select exiting variable to keep feasibility
- Repeat until all non-basic variables have objective ≥ 0

Degeneracy

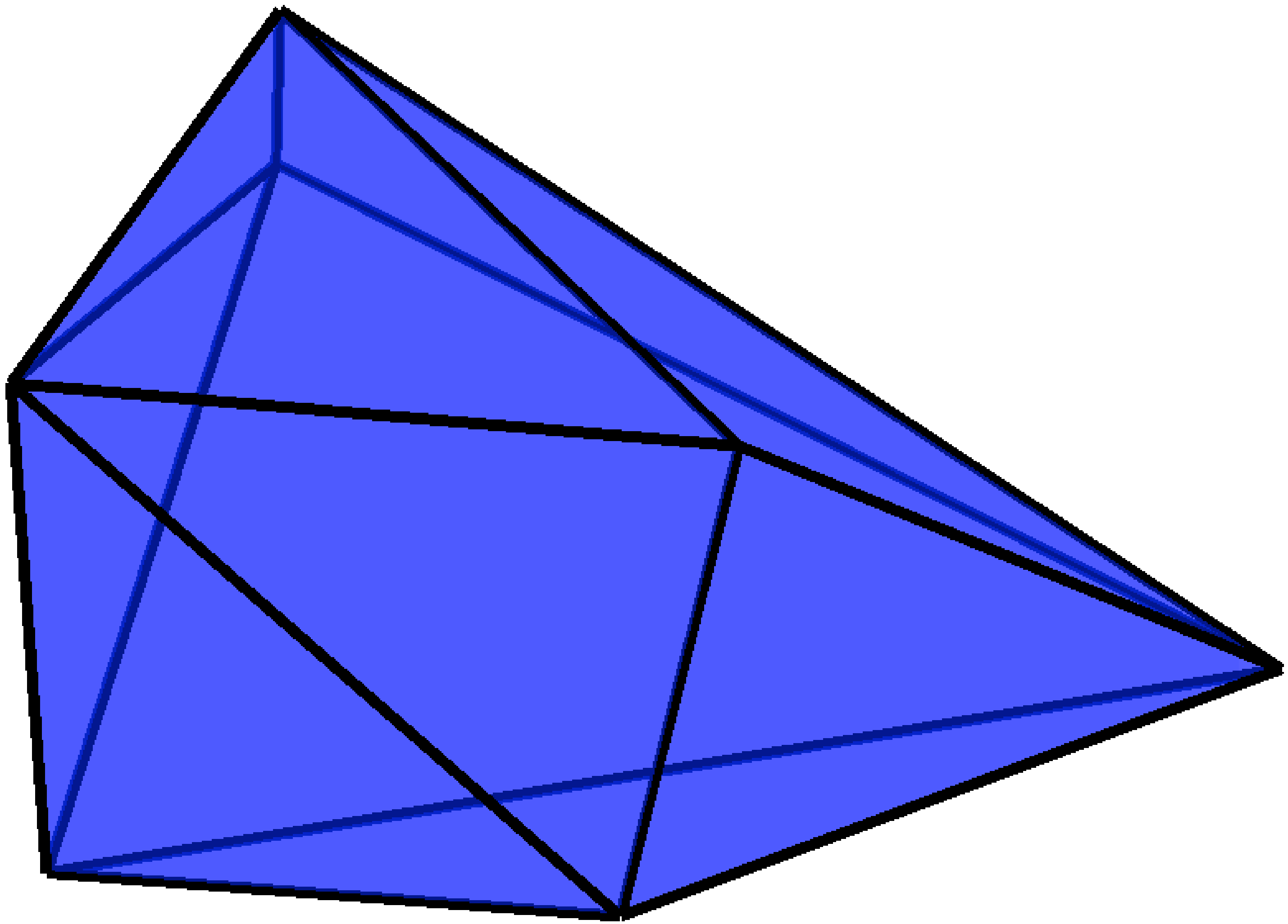
- Not every set of m variables yields a corner
    - some have rank < m (not a basis)
    - some are infeasible
- Can the reverse be true? Can two bases yield the same corner?

Degeneracy



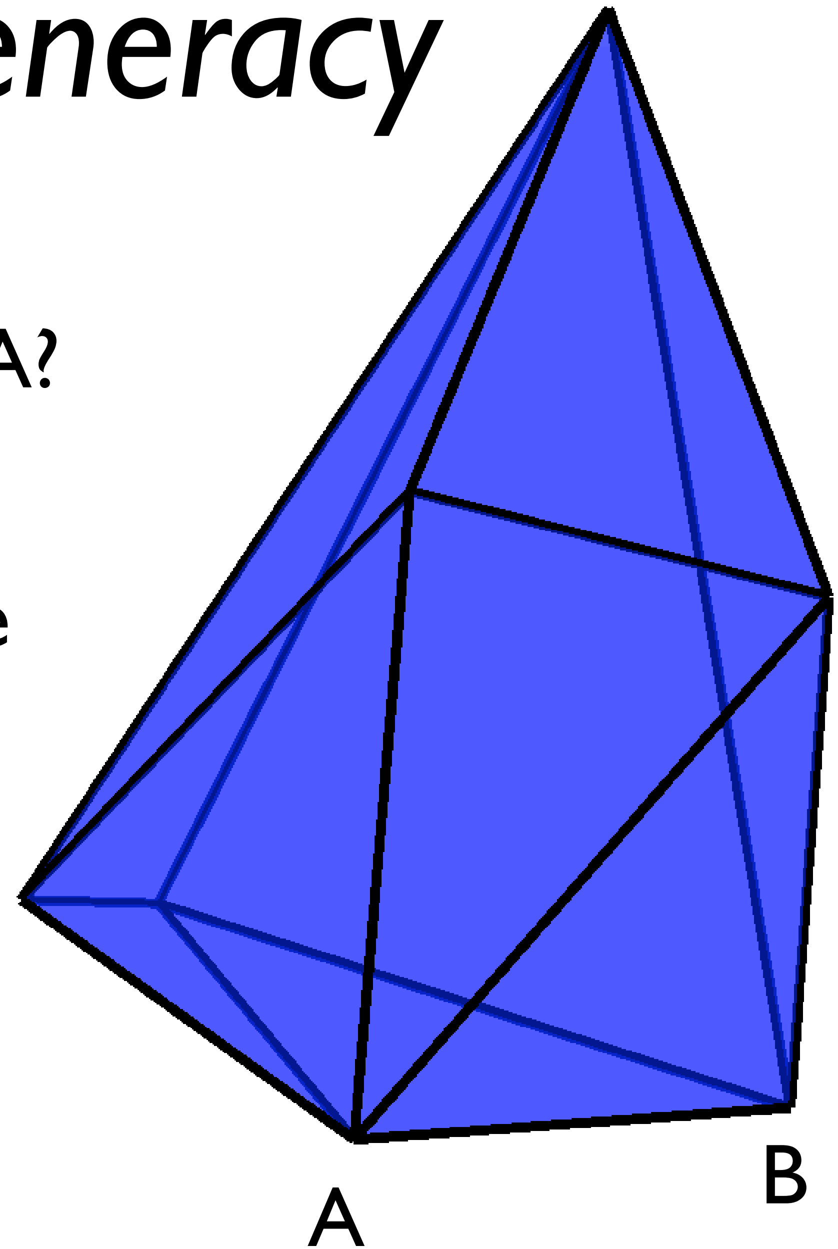
x	y	u	v	w	RHS
1	1	1	0	0	4
2	5	0	1	0	12
1	2	0	0	1	16/3
1	0	0	-2	5	8/3
0	1	0	1	-2	4/3
0	0	1	1	-3	0
1	0	2	0	-1	8/3
0	1	-1	0	1	4/3
0	0	1	1	-3	0

Degeneracy in 3D



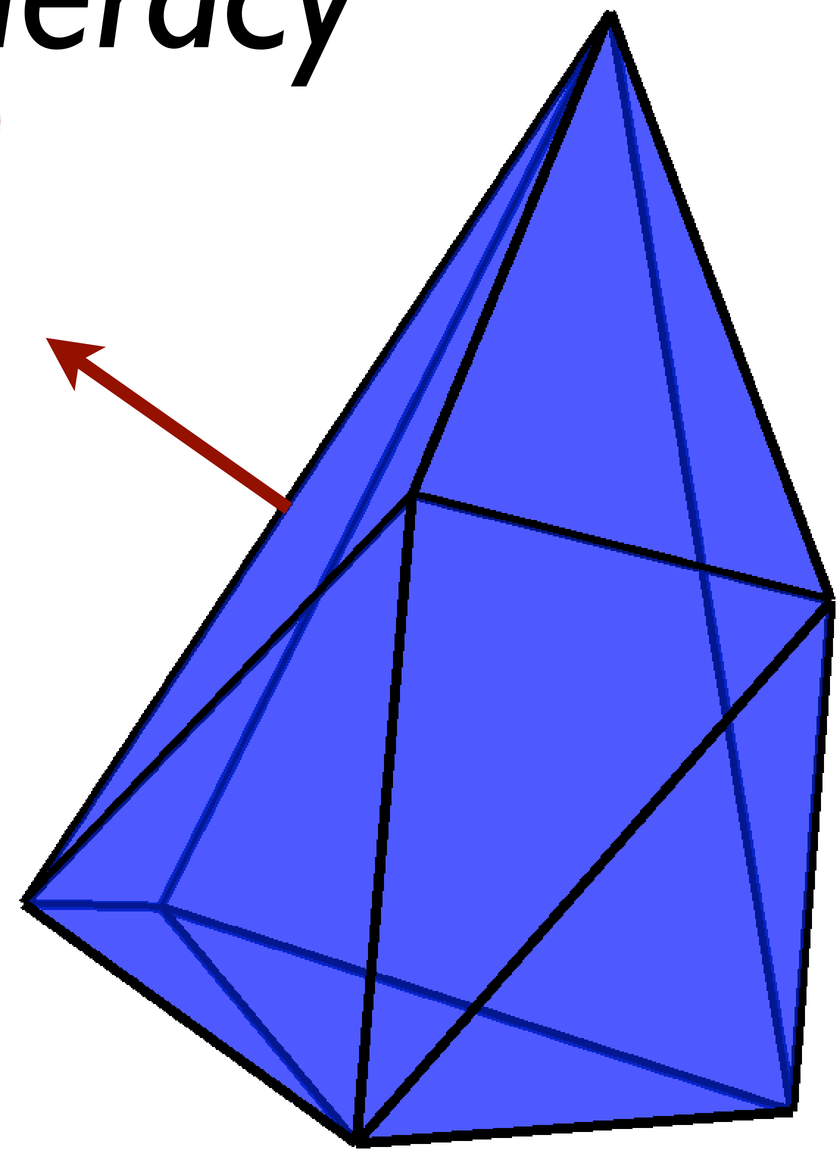
Bases & degeneracy

- How many bases for vertex A?
    -
- Are they all neighbors of one another?
    -
- Are they all neighbors of B?
    -



Dual degeneracy

- More than m entries in objective row = 0
  - so, a nonbasic variable has reduced cost = 0
  - objective orthogonal to a d-face for d ≥ 1



Handling degeneracy

- Sometimes have to make pivots that don't improve objective
  - stay at same corner (exiting variable was already 0)
  - move to another corner w/ same objective (coeff of entering variable in objective was 0)
- Problem of cycling
  - need an anti-cycling rule (there are many…)
  - e.g.: add tiny random numbers to obj, RHS