Computational Integer Programming

Lecture 7: Review of Linear Optimization

Dr. Ted Ralphs

A Quick Review of Linear Optimization

Definition 1. *A* polyhedron *is a set of the form* $\{x \in \mathbb{R}^n | Ax \geq b\}$*, where* $A \in \mathbb{R}^{m \times n}$ *and* $b \in \mathbb{R}^m$*.*

Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a given polyhedron.

Definition 2. *A vector* $x \in \mathcal{P}$ *is an* extreme point *of* $\mathcal{P}$ *if* $\nexists y, z \in \mathcal{P}, \lambda \in (0, 1)$ *such that* $x = \lambda y + (1 - \lambda) z$*.*

Definition 3. *A vector* $x \in \mathcal{P}$ *is an* vertex *of* $\mathcal{P}$ *if* $\exists c \in \mathbb{R}^n$ *such that* $c^\top x < c^\top y \; \forall y \in \mathcal{P}, x \neq y$*.*

Basic Solutions and Extreme Points

Consider a polyhedron $\mathcal{P} = \{x \in \mathbb{R}^n | Ax \geq b\}$ and let $\hat{x} \in \mathbb{R}^n$ be given.

Definition 4. *The vector $\hat{x}$ is a* basic solution *with respect to $\mathcal{P}$ if there exist n linearly independent, binding constraints at $\hat{x}$.*

Definition 5. *If $\hat{x}$ is a basic solution and $\hat{x} \in \mathcal{P}$, then $\hat{x}$ is a* basic feasible solution.

Theorem 1. *If $\mathcal{P}$ is nonempty and $\hat{x} \in \mathcal{P}$, then the following are equivalent:*

- *$\hat{x}$ is a vertex.*
- *$\hat{x}$ is an extreme point.*
- *$\hat{x}$ is a basic feasible solution.*

Example

$$
\begin{aligned}
\max \quad & 2x_1 + 5x_2 \\
\text{s.t.} \quad & -x_1 + 3.75x_2 \leq 14.375 \\
& -x_1 - 2x_2 \leq -2.5 \\
& -14x_1 + 8x_2 \leq 1 \\
& x_1 - 18x_2 \leq -2.5 \\
& 3.75x_1 - x_2 \leq 23.875 \\
& x_1 + x_2 \leq 12.7 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

Example

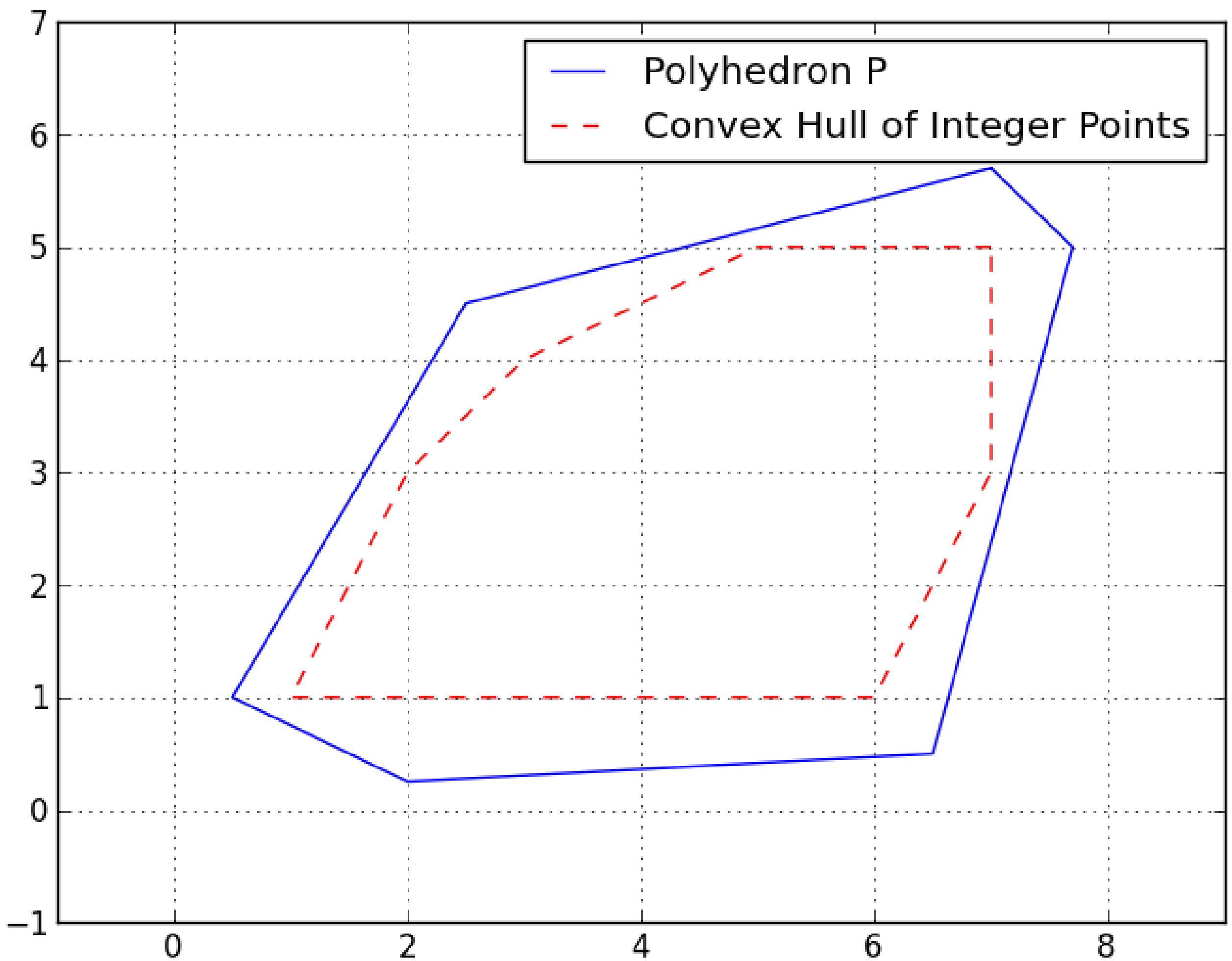


Figure 1: Feasible region for example

Polyhedra in Standard Form

- For the next few slides, we consider the standard form polyhedron $\mathcal{P} = \{x \in \mathbb{R}^n | \bar{A}x = b, x \geq 0\}$.
- Here, $\bar{A} = [A \mid I]$, where the additional columns are those corresponding to the slack variables.
- The feasible region of any linear optimization problem can be expressed equivalently in this form.
- We will assume that the rows of $\bar{A}$ are linearly independent $\Rightarrow m \leq n$.
- What does a basic feasible solution look like here?

Basic Feasible Solutions in Standard Form

- In standard form, the equations are always binding.
- To obtain a basic solution, we must set $n - m$ of the variables to zero (why?).
- We must also end up with a set of linearly independent constraints.
- Therefore, the variables we pick cannot be arbitrary.

Theorem 2. *Consider a polyhedron $\mathcal{P}$ in standard form with m linearly independent constraints. A vector $\hat{x} \in \mathbb{R}^n$ is a basic solution with respect to $\mathcal{P}$ if and only if $\bar{A}\hat{x} = b$ and there exist indices $B(1), \ldots, B(m)$ such that:*

- *The columns $\bar{A}_{B(1)}, \ldots, \bar{A}_{B(m)}$ are linearly independent, and*
- *If $i \neq B(1), \ldots, B(m)$, then $\hat{x}_i = 0$.*

Some Terminology

- If $\hat{x}$ is a basic solution, then $\hat{x}_{B(1)}, \ldots, \hat{x}_{B(m)}$ are the *basic variables*.
- The columns $\bar{A}_{B(1)}, \ldots, \bar{A}_{B(m)}$ are called the *basic columns*.
- Since they are linearly independent, these columns form a *basis* for $\mathbb{R}^m$.
- A set of basic columns form a *basis matrix*, denoted B. So we have,

$$B = \left[\bar{A}_{B(1)}\ \bar{A}_{B(2)} \cdots \bar{A}_{B(m)}\right], \quad x_B = \begin{bmatrix} x_{B(1)} \\ \vdots \\ x_{B(m)} \end{bmatrix}$$

Basic Solutions and Bases

- Given a basis matrix B, the values of the basic variables are obtained by solving $Bx_B = b$, whose unique solution is $x_B = B^{-1}b$.
- However, multiple bases can give the same basic solution.
- Two bases are *adjacent* if they differ in only one basic column.
- Two basic solutions are adjacent if and only if they can be obtained from two adjacent bases (proof is homework).

Example: Basis Inverse

Basis inverse and corresponding solution when non-basic variables are s_1 and s_6:

```
[ 0.21  0.    0.    0.    0.    0.21]
[ 0.21  1.    0.    0.    0.    1.21]
[-4.63  0.    1.    0.    0.    9.37]
[ 4.    0.    0.    1.    0.    3.  ]
[ 1.    0.    0.    0.    1.   -2.75]
[-0.21  0.    0.    0.    0.    0.79]
```

Example

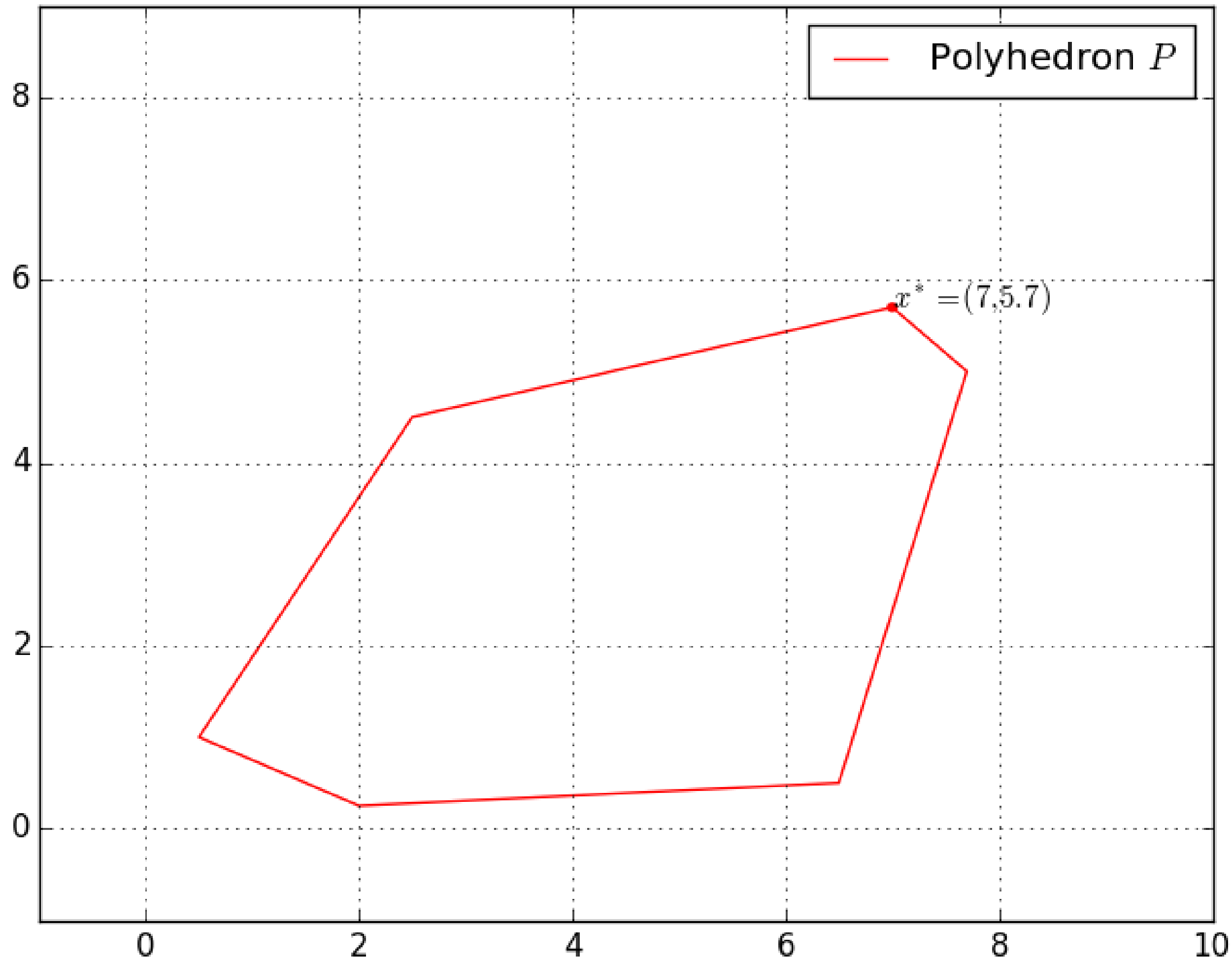


Figure 2: Basic solution when s_1 and s_6 are non-basic

Optimality of Extreme Points

Theorem 3. *Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a polyhedron and consider the problem $\min_{x \in \mathcal{P}} c^\top x$ for a given $c \in \mathbb{R}^n$. If $\mathcal{P}$ has at least one extreme point and there exists an optimal solution, then there exists an optimal solution that is an extreme point.*

- For linear optimization, a finite optimal cost is equivalent to the existence of an optimal solution.
- The previous result can be strengthened.
- Since any linear optimization problem can be written in standard form and all standard form polyhedra have an extreme point, we get the following:

Theorem 4. *Consider the linear optimization problem of minimizing $c^\top x$ over a nonempty polyhedron. Then, either the optimal cost is $-\infty$ or there exists an optimal solution.*

Iterative Search Algorithms

- Many optimization algorithms are *iterative* in nature.
- Geometrically, this means that they move from a given starting point to a new point in a specified *search direction*.
- This search direction is calculated to be both *feasible* and *improving*.
- The process stops when we can no longer find a feasible, improving direction.
- For linear optimization problems, it is always possible to find a feasible improving direction if we are not at an optimal point.
- This is essentially what makes linear optimization problems “easy” to solve.

Feasible and Improving Directions

Definition 6. *Let $\hat{x}$ be an element of a polyhedron $\mathcal{P}$. A vector $d \in \mathbb{R}^n$ is said to be a* feasible direction *if there exists $\theta \in \mathbb{R}_+$ such that $\hat{x} + \theta d \in \mathcal{P}$.*

Definition 7. *Consider a polyhedron $\mathcal{P}$ and the associated linear optimization problem $\min_{x \in \mathcal{P}} c^\top x$ for $c \in \mathbb{R}^n$. A vector $d \in \mathbb{R}^n$ is said to be an* improving direction *if $c^\top d < 0$.*

Notes:

- Once we find a feasible, improving direction, we want to move along that direction as far as possible.
- Recall that we are interested in extreme points.
- The simplex algorithm moves between adjacent extreme points using improving directions.

Constructing Feasible Search Directions in Linear Optimization

- Consider a BFS $\hat{x}$, so that $\hat{x}_N = 0$.
- Any feasible direction must increase the value of at least one of the nonbasic variables (why?).
- We will consider moving in *basic directions* that increase the value of exactly one of the nonbasic variables, say variable j. This means

$$\begin{aligned} d_j &= 1 \\ d_i &= 0 \text{ for every nonbasic index } i \neq j \end{aligned}$$

- In order to remain feasible, we must also have $\bar{A}d = 0$ (why?), which means

$$0 = \bar{A}d = \sum_{i=1}^{n} \bar{A}_i d_i = \sum_{i=1}^{m} \bar{A}_{B(i)} d_{B(i)} + \bar{A}_j = B d_B + \bar{A}_j \Rightarrow d_B = -B^{-1}\bar{A}_j$$

Constructing Improving Search Directions

- Now we know how to construct feasible search directions—how do we ensure they are improving?
- Recall that we must have $c^\top d < 0$.

Definition 8. *Let $\hat{x}$ be a basic solution, let B be an associated basis matrix, and let c_B be the vector of costs of the basic variables. For each j, we define the* reduced cost *$\bar{c}_j$ of variable j by*

$$\bar{c}_j = c_j - c_B^\top B^{-1} \bar{A}_j.$$

- The basic direction associated with variable j is improving if and only if $\bar{c}_j < 0$.
- Note that all basic variables have a reduced cost of 0 (why?).

Optimality Conditions

Theorem 5. *Consider a basic feasible solution $\hat{x}$ associated with a basis matrix B and let $\bar{c}$ be the corresponding vector of reduced costs.*

- *If $\bar{c} \geq 0$, then $\hat{x}$ is optimal.*
- *If $\hat{x}$ is optimal and nondegenerate, then $\bar{c} \geq 0$.*

Notes:

- The condition $\bar{c} \geq 0$ implies there are no feasible improving directions.
- However, $\bar{c}_j < 0$ does not ensure the existence of an improving, feasible direction unless the current BFS is nondegenerate

.

The Tableau

- The tableau looks like this

$-c_B^\top B^{-1}b$	$c^\top - c_B^\top B^{-1}\bar{A}$
$B^{-1}b$	$B^{-1}\bar{A}$

- In more detail, this is

$-c_B^\top x_B$	$\bar{c}_1$	$\cdots$	$\bar{c}_n$
$x_{B(1)}$ $\vdots$ $x_{B(m)}$	$B^{-1}\bar{A}_1$	$\cdots$	$B^{-1}\bar{A}_n$

Optimal Tableau in Example

Tableau and reduced costs when non-basic variables are s_1 and s_6:

```
[ 0.     0.    -1.22  0.     0.     0.     0.    -2.63]

[ 0.     1.     0.21  0.     0.     0.     0.     0.21] [   5.7  ]
[ 0.     0.     0.21  1.     0.     0.     0.     1.21] [  15.9  ]
[ 0.     0.    -4.63  0.     1.     0.     0.     9.37] [  53.4  ]
[ 0.     0.     4.    0.     0.     1.     0.     3.  ] [  93.1  ]
[ 0.     0.     1.    0.     0.     0.     1.    -2.75] [   3.33 ]
[ 1.     0.    -0.21  0.     0.     0.     0.     0.79] [   7.0  ]
```

Example

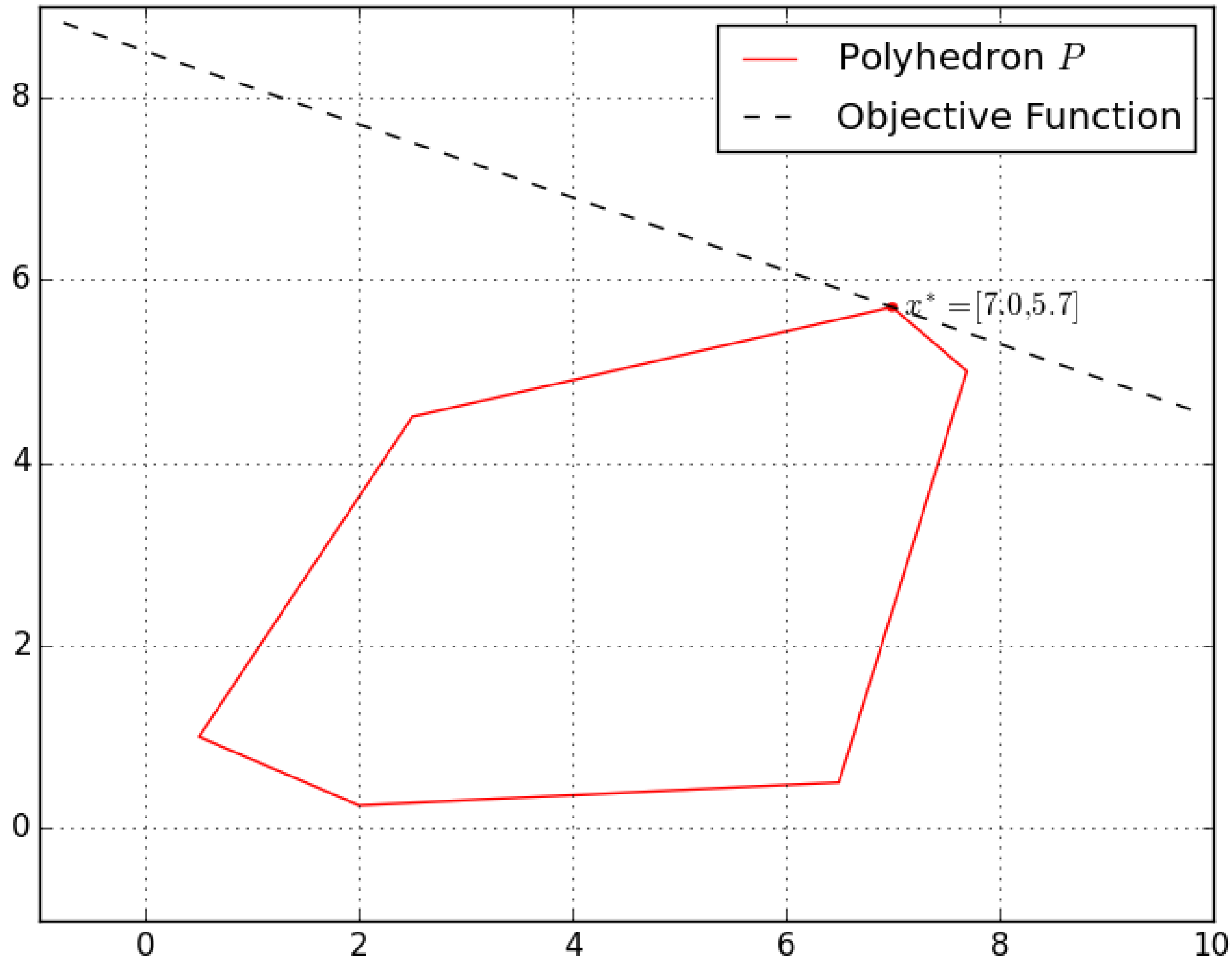


Figure 3: Optimal basic solution for example

The Revised Simplex Method

A typical iteration of the revised simplex method:

1. Start with a specified BFS $\hat{x}$ and the associated basis inverse B^{-1}.
2. Compute $p^\top = c_B^\top B^{-1}$ and the reduced costs $\bar{c}_j = c_j - p^\top \bar{A}_j$.
3. If $\bar{c} \geq 0$, then the current solution is optimal.
4. Select the entering variable j and compute $u = B^{-1}\bar{A}_j$.
5. If $u \leq 0$, then the LP is unbounded.
6. Determine the step size $\theta^* = \min_{\{i|u_i>0\}} \frac{\hat{x}_{B(i)}}{u_i}$.
7. Determine the new solution and the leaving variable i.
8. Update B^{-1}.
9. Go to Step 1.

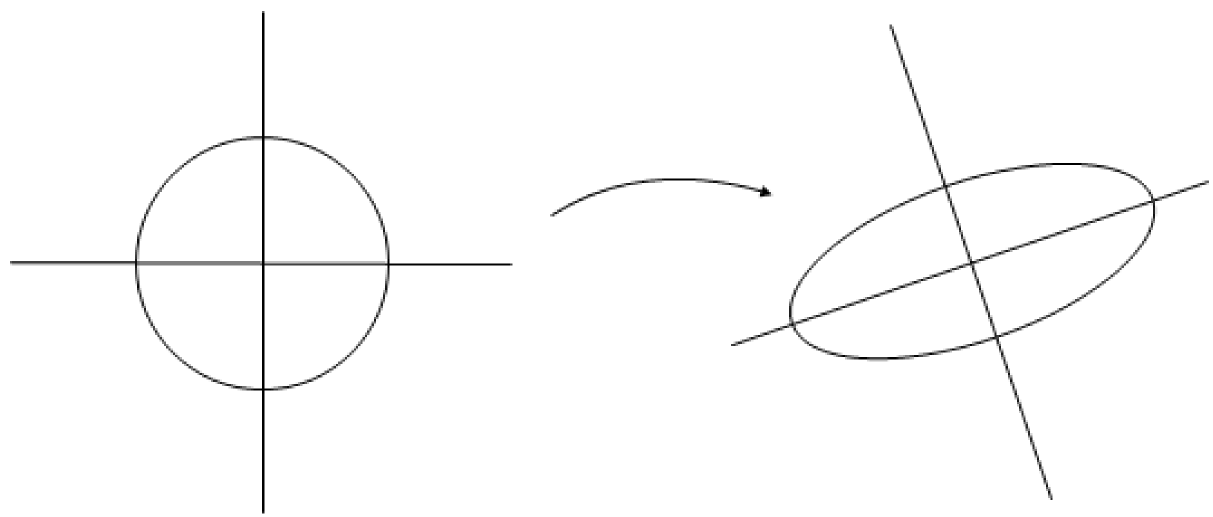
Numerical Considerations

- In the simplex algorithm, we are solving a sequence of closely related systems of equations.
- The factorization we are using to solve each of these systems is updated and round-off error accumulates.
- In practice, it is common to periodically discard the basis factorization and re-compute it from scratch to combat this problem.
- What factors affect the accuracy of solving just one of these systems from scratch?
- Naturally, the condition number of the current basis is important.
- Can we interpret the condition number of the basis in geometric terms?

The Geometry of Conditioning

- Consider again the geometric interperation of condition number of a matrix B.
- Roughly speaking, it is the ratio of the largest to smallest axes of the ellipsoid we get by pre-multiplying the points on a unit ball by B:

$$\{Bx \mid x \in \mathbb{R}, \|x\| = 1\}$$



- Question: What affects the geometry of this ellipsoid?

The Geometry of Conditioning

- Factors affecting the shape of the set $\{Bx \mid x \in \mathbb{R}, \|x\| = 1\}$.
  - The (relative) magnitude of the norms of the rows of B.
  - The "angles" between the rows.
- This is essentially because

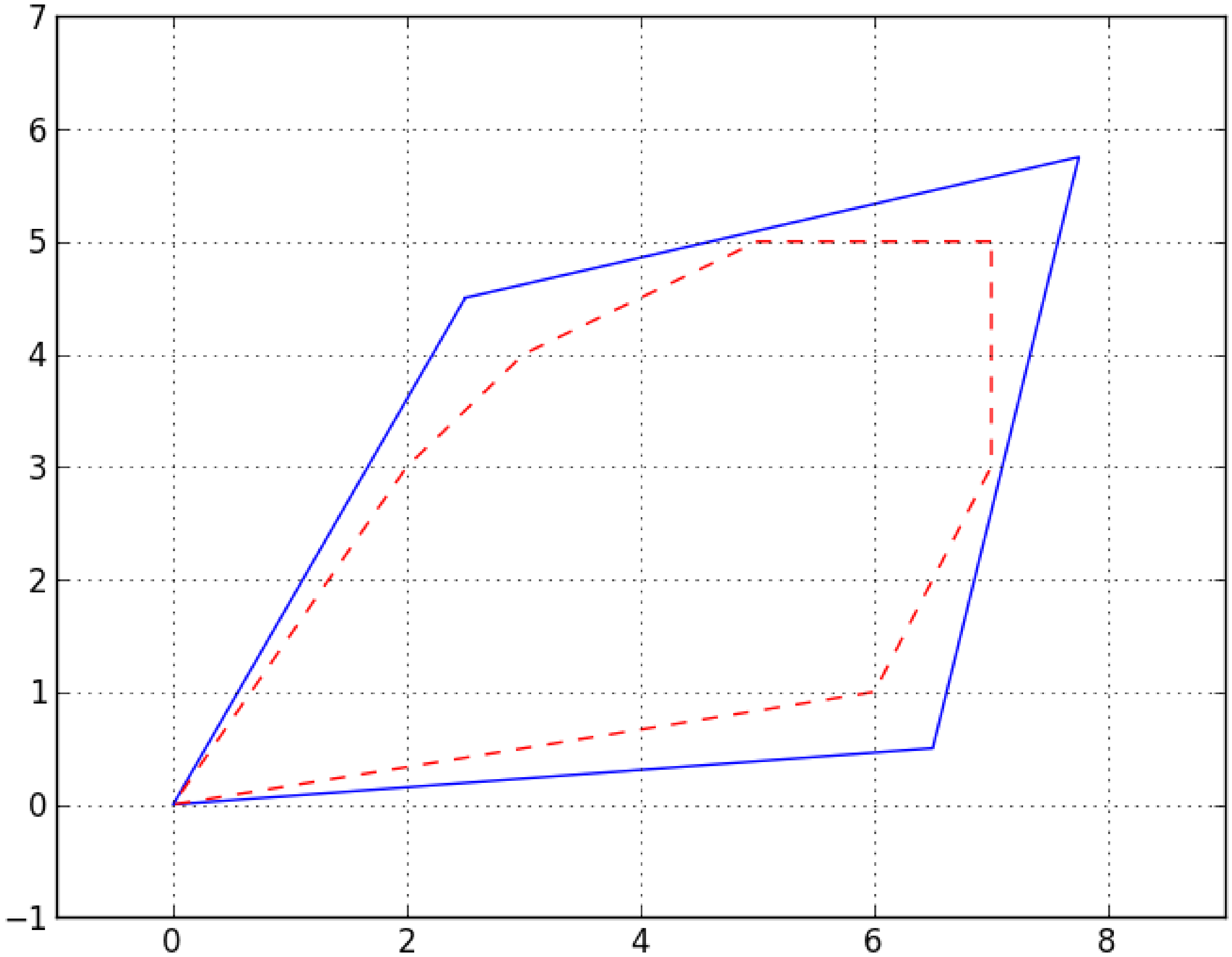
$$|x^\top y| = \|x\| \|y\| \cos \beta$$

  where β is the angle between x and y.
- Note that condition number is just the "worst case."

The Geometry of Conditioning

- Note that just because the matrix B is ill-conditioned does not mean that the problem of finding each individual component of the solution is ill-conditioned.
  - The condition number of the matrix is a worst-case measure over all the component-wise problems.
  - There is always one component that exhibits this worst-case behavior.
- Let r_i be the i^{th} row of B^{-1}.
- The relative condition of the problem for component i is affected by
  - the angle between r_i and f
  - the angle between r_i and b

The Geometry of Conditioning



The LP Dual Problem

- Consider a standard form LP $\min\{c^\top x : \bar{A}x = b, x \geq 0\}$.
- To derive the *dual problem*, we use Lagrangian relaxation and consider the function
$$g(p) = \min_{x \geq 0} \left[c^\top x + p^\top (b - \bar{A}x)\right]$$
in which infeasibility is penalized by a vector of *dual prices* p.
- For every vector p, $g(p)$ is a lower bound on the optimal value of the original LP.
- To achieve the best bound, we considered maximizing $g(p)$, which is equivalent to
$$\begin{aligned} \max \ & p^\top b \\ s.t. \ & p^\top \bar{A} \leq c \end{aligned}$$
- This LP is the dual to the original one.

Economic Interpretation of the Dual

- Recall that there always exists an optimal solution that is *basic*.
- We construct basic solutions by
  - Choosing a *basis* B of m linearly independent columns of $\bar{A}$.
  - Solving the system $Bx_B = b$ to obtain the values of the *basic variables*.
  - Setting remaining variables to value 0.
- If $x_B \geq 0$, then the associated basic solution is *feasible*.
- With respect to any basic feasible solution, it is easy to determine the impact of increasing a given activity.
- The *reduced cost*

$$\bar{c}_j = c_j - c_B^\top B^{-1} \bar{A}_j.$$

  of (nonbasic) variable j tells us how the objective function value changes if we increase the level of activity j by one unit.
- From the resource (dual) perspective, the quantity $u = c_B B^{-1}$ is a vector that tells us the marginal economic value of each resource.
- Thus, the vector u gives us a *price* for each resource.

Marginal Prices in AMPL

Again, recall the simple bond portfolio model from Lecture 3.

```
ampl: model bonds.mod;
ampl: solve;
...
ampl: display rating_limit, cash_limit;
rating_limit = 1
cash_limit = 2
```

- This tells us that the optimal marginal cost of the `rating_limit` constraint is 1.
- What does this tell us about the "cost" of improving the average rating?
- What is the return on an extra $\$1K$ of cash available to invest?

Another Interpretation of Marginal Prices

- Let's consider again the prices for the constraints in the simple bond portfolio model.
- By combining the two constraints with nonzero prices, we can get a third inequality that must be satisfied by any feasible solution:

$$\begin{aligned} 2\,[x_1 + x_2 \leq 100] \quad & + \\ 1\,[2x_1 + x_2 \leq 150] \quad & = \\ 4x_1 + 3x_2 \quad & \leq 350 \end{aligned}$$

- What does this tell us about the optimal solution value?

Economic Interpretation of Optimality

Example: A simple product mix problem.

```
ampl: var X1;
ampl: var X2;
ampl: maximize profit: 3*X1 + 3*X2;
ampl: subject to hours: 3*X1 + 4*X2 <= 120000;
ampl: subject to cash: 3*X1 + 2*X2 <= 90000;
ampl: subject to X1_limit: X1 >= 0;
ampl: subject to X2_limit: X2 >= 0;
ampl: solve;
...
ampl: display X1;
X1 = 20000
ampl: display X2;
X2 = 15000
```

Shadow Prices in Product Mix Model

```
ampl: model simple.mod
ampl: solve;
...
ampl: display hours, cash;
hours = 0.5
cash = 0.5
```

- This tells us that increasing the hours by 2000 will increase profit by $(2000)(0.5) = \$1000$.
- Hence, we should be willing to pay up to $\$.50/\text{hour}$ for additional labor hours (as long as the solution remains feasible).
- We can also see that the availability of cash and man hours are contributing equally to the cost of each product.

Economic Interpretation of Optimality

- In the preceding example, we can use the shadow prices to determine how much each product "costs" in terms of its constituent "resources."
- The reduced cost of a product is the difference between its selling price and the (implicit) cost of the constituent resources.
- If we discover a product whose "cost" is less than its selling price, we try to manufacture more of that product to increase profit.
- With the new product mix, the demand for various resources is changed and their prices are adjusted.
- We continue until there is no product with cost less than its selling price.
- This is the same as having the reduced costs nonpositive (recall this was a maximization problem).
- Complementary slackness says that we should only manufacture products for which cost and selling price are equal.
- This can be viewed as a sort of multi-round auction.

AMPL: Displaying Auxiliary Values with Suffixes

- In AMPL, it's possible to display much of the auxiliary information needed for sensitivity using suffixes.
- For example, to display the reduced cost of a variable, type the variable name with the suffix *.rc*.
- Recall again the short term financing example (`short_term_financing.mod`).

```
ampl: display credit.rc;
credit.rc [*] :=
 0  -0.003212
 1   0
 2  -0.0071195
 3  -0.00315
 4   0
 5   0
;
```

- How do we interpret this?

AMPL: Sensitivity Ranges

- AMPL does not have built-in sensitivity analysis commands.
- AMPL/CPLEX does provide such capability, however.
- To get sensitivity information, type the following

```
ampl: option cplex_options 'sensitivity';
```

- Solve the bond portfolio model:

```
ampl: solve;
...
suffix up OUT;
suffix down OUT;
suffix current OUT;
```

AMPL: Accessing Sensitivity Information

Access sensitivity information using the suffixes *.up* and *.down*. This is from the model `bonds.mod`.

```
ampl: display cash_limit.up, rating_limit.up, maturity_limit.up;
cash_limit.up = 102
rating_limit.up = 200
maturity_limit.up = 1e+20

ampl: display cash_limit.down, rating_limit.down, maturity_limit.down;
cash_limit.down = 75
rating_limit.down = 140
maturity_limit.down = 350

ampl: display buy.up, buy.down;
: buy.up buy.down    :=
A    6       3
B    4       2
;
```

AMPL: Sensitivity for the Short Term Financing Model

```
ampl: short_term_financing.mod;
ampl: short_term_financing.dat;
ampl: solve;
ampl: display credit, credit.rc, credit.up, credit.down;
:     credit      credit.rc     credit.up  credit.down    :=
0      0         -0.00321386   0.00321386    -1e+20
1     50.9804     0            0.00318204        0
2      0         -0.00711864   0.00711864    -1e+20
3      0         -0.00315085   0.00315085    -1e+20
4      0          0            0             -1e+20
;
```

AMPL: Sensitivity for the Short Term Financing Model (cont.)

```
ampl: display bonds, bonds.rc, bonds.up, bonds.down;
:      bonds       bonds.rc      bonds.up      bonds.down      :=
0     150           0           0.00399754   -0.00321386
1      49.0196      0           0            -0.00318204
2     203.434       0           0.00706931    0
3       0           0           0             0
4       0           0           0             0
;
```

AMPL: Sensitivity for the Short Term Financing Model (cont.)

```
ampl: display invest, invest.rc, invest.up, invest.down;
:      invest      invest.rc        invest.up        invest.down       :=
-1      0          0                 0                  0
0       0         -0.00399754        0.00399754    -1e+20
1       0         -0.00714           0.00714       -1e+20
2     351.944      0                 0.00393091         -0.0031603
3       0         -0.00391915        0.00391915    -1e+20
4       0         -0.007             0.007         -1e+20
5      92.4969     0             1e+20                  2.76446e-14
;
```

Sensitivity Analysis of the Dedication Model

Let's look at the sensitivity information in the dedication model

```
ampl: model dedication.mod;
ampl: data dedication.dat;
ampl: solve;
ampl: display cash_balance, cash_balance.up, cash_balance.down;
: cash_balance cash_balance.up cash_balance.down    :=
1    0.971429        1e+20          5475.71
2    0.915646       155010          4849.49
3    0.883046       222579          4319.22
4    0.835765       204347          3691.99
5    0.656395       105306          2584.27
6    0.619461       123507          1591.01
7    0.5327         117131           654.206
8    0.524289       154630             0
;
```

How can we interpret these?

Sensitivity Analysis of the Dedication Model

```
ampl: display buy, buy.rc, buy.up, buy.down;
:      buy          buy.rc          buy.up     buy.down     :=
A     62.1361   -1.42109e-14      105        96.4091
B      0          0.830612       1e+20       98.1694
C    125.243    -1.42109e-14      101.843    97.6889
D    151.505     1.42109e-14      101.374    93.2876
E    156.808    -1.42109e-14      102.917    80.7683
F    123.08      0                113.036   100.252
G      0          8.78684        1e+20       91.2132
H    124.157     0                104.989    92.3445
I    104.09      0                111.457   101.139
J     93.4579    0                 94.9      37.9011
;
```

Sensitivity Analysis of the Dedication Model

```
ampl: display cash, cash.rc, cash.up, cash.down;
: cash    cash.rc   cash.up   cash.down     :=
0   0    0.0285714   1e+20     0.971429
1   0    0.0557823   1e+20    -0.0557823
2   0    0.0326005   1e+20    -0.0326005
3   0    0.0472812   1e+20    -0.0472812
4   0    0.17937     1e+20    -0.17937
5   0    0.0369341   1e+20    -0.0369341
6   0    0.0867604   1e+20    -0.0867604
7   0    0.0084114   1e+20    -0.0084114
8   0    0.524289    1e+20    -0.524289
;
```

Sensitivity Analysis in PuLP and Pyomo

- Both PuLP and Pyomo also support sensitivity analysis through suffixes.
- Pyomo
  - The option `--solver-suffixes='.*'` should be used.
  - The supported suffixes are `.dual`, `.rc`, and `.slack`.
- PuLP
  - PuLP creates suffixes by default when supported by the solver.
  - The supported suffixed are `.pi` and `.rc`.

Sensitivity Analysis of the Dedication Model with PuLP

```
for t in Periods[1:]:
    prob += (cash[t-1] - cash[t]
             + lpSum(BondData[b, 'Coupon'] * buy[b]
               for b in Bonds if BondData[b, 'Maturity'] >= t)
             + lpSum(BondData[b, 'Principal'] * buy[b]
               for b in Bonds if BondData[b, 'Maturity'] == t)
             == Liabilities[t]), "cash_balance_%s"%t

status = prob.solve()

for t in Periods[1:]:
    print 'Present of $1 liability for period', t,
    print prob.constraints["cash_balance_%s"%t].pi
```