Review session: Exam question 1
The linear programming model and LP algorithms.

- Lecture 1
  - The linear programming problem.
  - Vanderbei, Chapter 1 and Chapter 2
- Lecture 2
  - Analysis of the simplex algorithm. The two-phase simplex algorithm.
  - Vanderbei, Chapter 2 and Chapter 3
- Lecture 3 (first half)
  - Analysis of the simplex algorithm.
  - Vanderbei, Chapter 4.
- Lecture 5
  - Implementation Issues and the revised simplex method.
  - Vanderbei, Chapter 6 (sections 1-5) and Chapter 8 (section 1-2)
- Lecture 13
  - The ellipsoid algorithm and interior point algorithms for linear programming.
  - Cursory: Papadimitriou and Steiglitz, pp 173-185. Vanderbei, Chapter 17, 18, and 22.

The LP model

- Optimizing (maximizing or minimizing) a **linear objective function** subject to **linear constraints** over real valued **decision variables**.

- Decision variables: $x_1, \dots, x_n \in \mathbb{R}$
- Linear objective function: $c_1 x_1 + \cdots + c_n x_n$
- Linear constraint: $a_1 x_1 + \cdots + a_n x_n \begin{Bmatrix} \leq \\ = \\ \geq \end{Bmatrix} b$

Solutions

- Solution:
  - Assignment of values to decision variables
- Feasible solution:
  - Solution that satisfies all constraints
  - LP is **infeasible** if there are no feasible solutions.
- Optimal solution:
  - A feasible solution that maximizes/minimizes the objective function among the set of feasible solutions.
  - LP is **unbounded** if there are arbitrarily good feasible solutions.

Standard Form

- **Maximization** problem
- Only $\leq$ **constraints** together with **nonnegativity constraints** for all decision variables.

- Concise matrix form: max $c^{\top}x$ s.t. $Ax \leq b$, $x \geq 0$.

- Every LP may be converted into ”equivalent” standard form LP.

- Addition of slack vars.: max $c^{\top}x$ s.t. $Ax + w = b$, $x, w \geq 0$

- Bundling x and w vars.: $\max \begin{bmatrix} c \\ 0 \end{bmatrix}^{\top} \begin{bmatrix} x \\ w \end{bmatrix}$ s.t. $\begin{bmatrix} A & I \end{bmatrix} \begin{bmatrix} x \\ w \end{bmatrix} = b$, $\begin{bmatrix} x \\ w \end{bmatrix} \geq 0$

- $\begin{bmatrix} c \\ 0 \end{bmatrix}$, $\begin{bmatrix} x \\ w \end{bmatrix}$, and $\begin{bmatrix} A & I \end{bmatrix}$ are ”renamed” as c, x and A in the matrix formulation of the Simplex algorithm.

Basis and Dictionaries

- Basis:
  - A selection of m **basic variables** from (x, w) such that their value are *uniquely determined* as (linear) functions of the remaining n **non-basic variables** from the linear system $Ax + w = b$.
  - Not every choice of m variables give a basis.
  - Choosing the m slack variables w *is* a basis.

- Basic solution:
  - Assigning 0 to non-basic variables and solving for the basic variables.

- Dictionary:
  - The system expressing the basic variables and the objective function as linear functions of the non-basic variables.
  - A dictionary is uniquely determined by the choice of basis.

Matrix Form

Maximize $c^\top x$

Subject to $Ax = b$

$x \geq 0$

$\mathcal{B}$ = indices of basic variables

$\mathcal{N}$ = indices of non-basic variables.

Partition of matrix: $A = [B \quad N]$

Partition of x and c: $x = \begin{bmatrix} x_{\mathcal{B}} \\ x_{\mathcal{N}} \end{bmatrix}$ and c $= \begin{bmatrix} c_{\mathcal{B}} \\ c_{\mathcal{N}} \end{bmatrix}$.

$$Ax = Bx_{\mathcal{B}} + Nx_{\mathcal{N}} = b$$

$$\zeta = c^\top x = c_{\mathcal{B}}^\top x_{\mathcal{B}} + c_{\mathcal{N}}^\top x_{\mathcal{N}}$$

Dictionary in matrix form

The matrix B must be invertible. Solving for $x_{\mathcal{B}}$ and substituting we get

$$\begin{aligned}\zeta &= c_{\mathcal{B}}^{\top} B^{-1} b - \left((B^{-1}N)^{\top} c_{\mathcal{B}} - c_{\mathcal{N}}\right)^{\top} x_{\mathcal{N}} \\ x_{\mathcal{B}} &= B^{-1} b - B^{-1} N x_{\mathcal{N}}\end{aligned}$$

Setting $x_{\mathcal{N}}$ to 0 we get the corresponding basic solution:

$$\begin{aligned}x_{\mathcal{N}}^{*} &= 0 \\ x_{\mathcal{B}}^{*} &= B^{-1} b\end{aligned}$$

With objective value:

$$\zeta^{*} = c_{\mathcal{B}}^{\top} B^{-1} b$$

Simplex Algorithm

- A **feasible** dictionary is maintained **explicitly** by the algorithm. Assume origo is feasible in the beginning.

- Pivoting:
  - Find **entering** variable: Pick variable with positive coefficient in objective function. Stop if no such variable.
  - Find **leaving** variable: Pick variable from basis to preserve non-negativity of basic variables.
  - (Special case: entering variable has non-negative coefficient in all equations – LP is unbounded!)
  - Rewrite equation of leaving variable in terms of entering variable.
  - Substitute for entering variable in dictionary.

- Partial correctness:
  - Objective function is expressed as a function of non-basic variables. These must be non-negative. Hence basic solution is optimal.

Degeneracy

- Dictionary is degenerate if some basic variable has value 0.
- A pivot operation might lead to different dictionary with same basic solution of current dictionary.
- Cycling is possible!
- Anti-cycling pivoting rules: Lexicographic method, Bland, …

- Lexicographic method: Replace b_i by $b_i + \epsilon_i$, where $0 < \epsilon_m \ll \epsilon_{m-1} \ll \cdots \ll \epsilon_1 \ll$ all other data.

- The matrix of coefficients of ϵ_i's start out as $m \times m$ identity matrix and its rank is maintained. In particular, no zero rows, which means we never encounter degenerate dictionaries.

- No degeneracy means termination is guaranteed.

Two-phase Simplex Algorithm

- Auxiliary problem
  - Max $-x_0$ s.t. $Ax - ex_0 \leq b$, $x_0, x \geq 0$, where $e = (1, \ldots, 1)^\mathsf{T}$.
  - Optimal value 0 if and only if $Ax \leq b$, $x \geq 0$ feasible.
  - In that case the final dictionary can be converted to feasible dictionary of original LP.

- Starting dictionary:
  - Make a pivot with x_0 and x_{m+i} where i is chosen with b_i minimal.

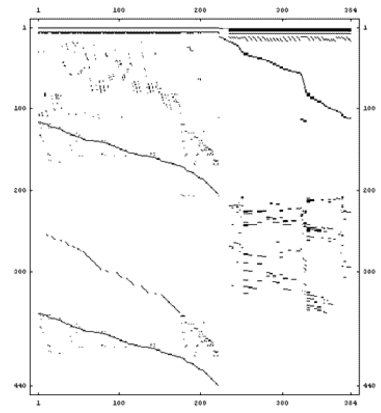
- Alternative two-phase simplex algorithm possible by *dual simplex algorithm*.

Fundamental theorem of linear programming

- For an arbitrary linear program in standard form:
  - If there is no optimal solution, then the problem is either infeasible or unbounded.
  - If a feasible solution exists, then a basic feasible solution exists.
  - If an optimal solution exists, then an optimal feasible solution exists.
- Proof: The two-phase simplex algorithm!

Revised Simplex Algorithm

- Very beneficial in practice where most coefficients are 0 (sparsity).



- Between iterations, only store:
  - Current basis, $\mathcal{B}$ (Set of indices).
  - Current basic solution, $x_{\mathcal{B}}^* = B^{-1}b - B^{-1}Nx_{\mathcal{N}}$.
  - Coefficients of objective function, $z_{\mathcal{N}}^* = (B^{-1}N)^\top c_{\mathcal{B}} - c_{\mathcal{N}}$ (= **dual** solution!).
- Main computational task:
  - Computing step directions:

$$\Delta x_{\mathcal{B}} = B^{-1}Ne_j$$
$$\Delta z_{\mathcal{N}} = -(B^{-1}N)^\top e_i$$

  - Amounts to solving linear systems.
  - LU-factorization (recording of steps of Gaussian elimination) , $B = LU$.
  - Exploit sparsity by choosing pivot elements heuristically.

Representation Issues

- Exact representation:
  - When using fractions we must repeatedly perform gcd operations to keep the size of numerator and denominator under control.
  - Can be avoided using integer pivoting: All equations are multiplied by $\det(B)$. This can be maintained using multiplication with current pivot coefficient and (exact integer) division with previous pivot coefficient.

- Floating point representation:
  - Must handle precision loss in a robust implementation.
  - Revised simplex algorithm is superior since errors are not accumulated.

Running time of Simplex Algorithm

- Pivot operations are efficiently computable.
- Worst-case exponential time! (Klee-Minty examples)
- For the Simplex algorithm the worst-case behavior is **not** typical.
- Number of iterations seems (almost) linear in the number of constraints in practice.
- Attempts at theoretical explanation of practical performance: Average-case analysis, smoothed analysis.

Worst-case efficient LP algorithms

- **Model 1:** Our computer holds exact real numbers. The size of the input is the number of real numbers in the input. The time complexity of an algorithm is the number of arithmetic operations performed.

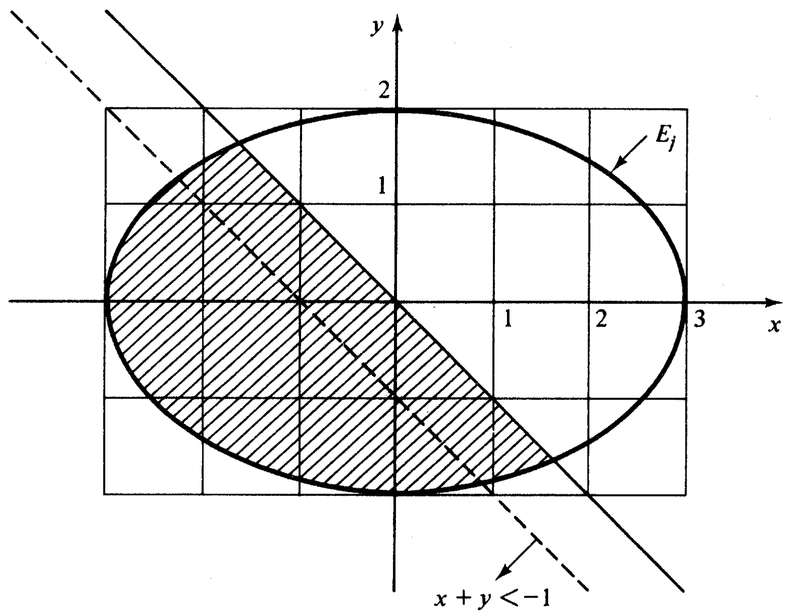
- **Model 2:** Our computer holds bits and bytes. The size of the input is the number of bits in the input. The time complexity of an algorithm is the number of bit-operations performed.

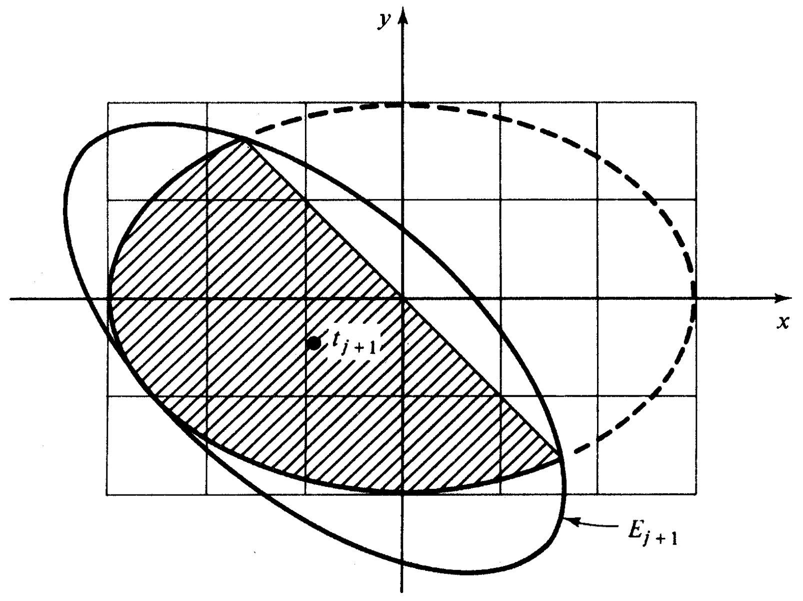
- We know an efficient algorithm for linear programming in Model 2 but not model 1:
  - **The Ellipsoid algorithm (Khachian, 1979).**
  - **Interior Point algorithms (Karmakar, 1984).**

Ellipsoid algorithm

Solves the equivalent (by duality theory) linear inequalities problem.

- Enclose (possibly a subset of) all possible feasible points within a large ball (i.e. ellipsoid).
- Test if center of ellipsoid is feasible.
- If not, the ellipsoid is partitioned into two parts by a hyperplane, with all possible feasible points in one part.
- Find a new smaller ellipsoid that encloses this half-ellipsoid (pick the smallest), and repeat.





Why this works

- Each iteration decreases volume by a factor $2^{-1/2(n+1)}$.
- First ellipsoid will have volume less than $(2n^2 2^{2L})^n$.
- If the system is feasible, the feasible region within the ball $B(0, n2^L)$ has volume at least $2^{-(n+2)L}$.
- Hence at most $K = 16n(n+1)L$ iterations are needed, since:

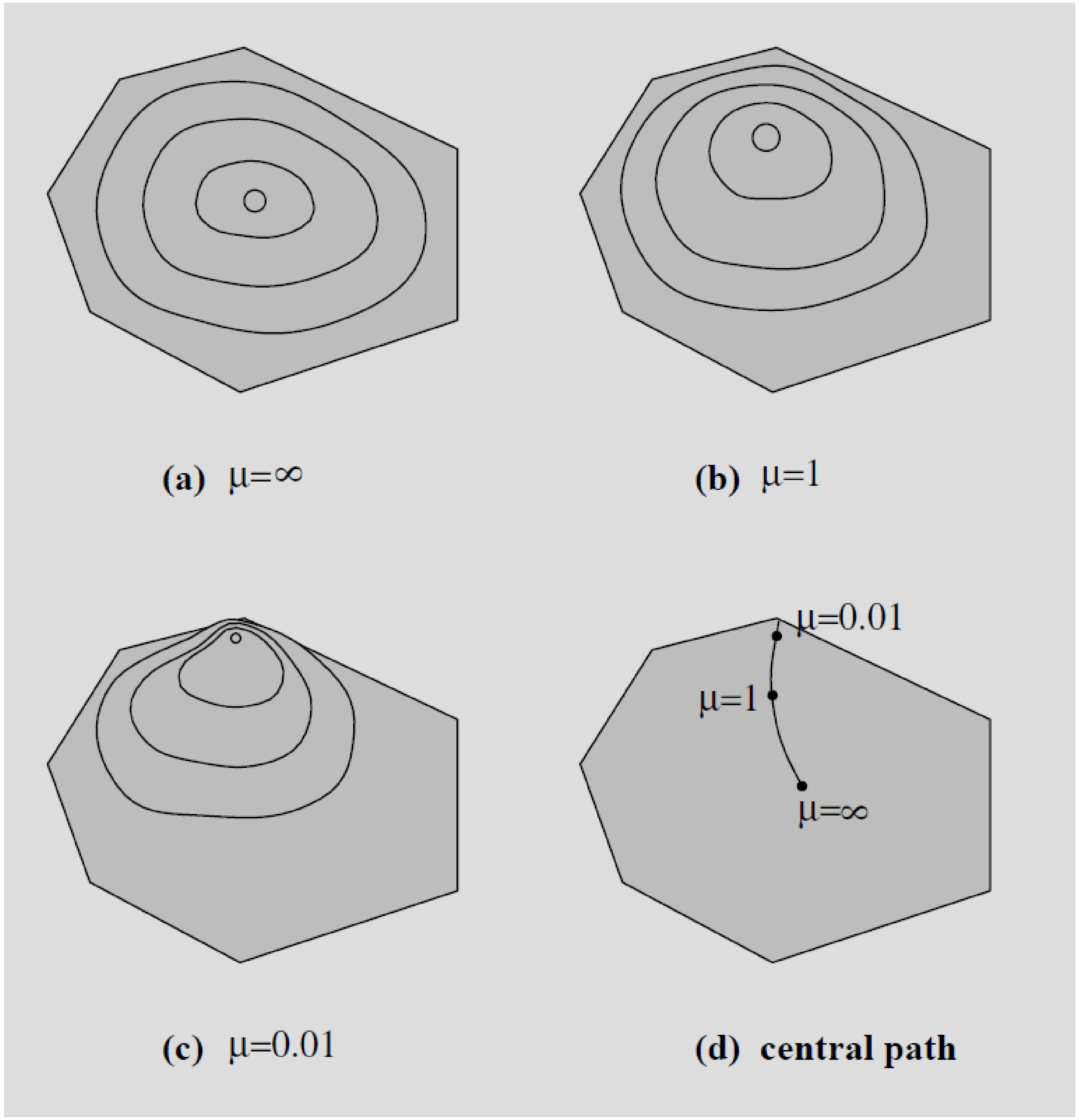
$$(2n^2 2^{2L})^n 2^{K/2(n+1)} = (2n^2 2^{2L})^n 2^{-8nL} < 2^{-(n+2)L}.$$

- Technical issues:
  - Must enlarge feasible region slight to ensure a positive volume
  - Must approximate a square-root.

Path-following method

- Given standard form LP:
  - max $c^\top x$ s.t. to $Ax \leq b, x \geq 0.$
- Add slack variables w:
  - max $c^\top x$ s.t. to $Ax + w = b, x, w \geq 0.$
- Convert to barrier problem:
  - max $c^\top x + \mu \sum_j \log x_j + \mu \sum_i \log w_i$ s.t. $Ax + w = b$
- Form Lagrangian:
  - $L(x, w, y) = c^\top x + \mu \sum_j \log x_j + \mu \sum_i \log w_i + y^\top (b - Ax - w)$
- Differentiate, equate to 0, and rearrange
  - $Ax + w = b, A^\top y - z = c, XZe = \mu e, YWe = \mu e$
- Introduce step directions, Δx, …, and substitute x by $x + \Delta x$, …
- Drop nonlinear terms, and solve resulting linear system.

The central path



Path-following algorithm

Given current point $(x, w, y, z) > 0$:

1. Estimate "value" for μ: $\mu = \delta \frac{z^\top x + y^\top w}{n+m}$
2. Compute step directions $(\Delta x, \Delta \mathrm{w}, \Delta y, \Delta z)$ pointing approximately at point $(x_\mu, w_\mu, y_\mu, z_\mu)$ on the central path.
3. Compute step length Θ such that $(\tilde{x}, \widetilde{w}, \tilde{y}, \tilde{z}) = (x, w, y, z) + \theta \cdot (\Delta x, \Delta \mathrm{w}, \Delta y, \Delta z) > 0$
4. Replace (x, w, y, z) by $(\tilde{x}, \widetilde{w}, \tilde{y}, \tilde{z})$
5. Repeat until (close to) optimal.

We also looked at the Homogeneous Self-Dual Method, and a predictor-corrector algorithm.

The path-following method extends to convex quadratic programming and convex programming in general.