

Lecture 1

Introduction and overview

- linear programming
- example from optimal control
- example from combinatorial optimization
- history
- course topics
- software

1-1

Linear program (LP)

$$\begin{aligned}
 &\text{minimize} && \sum_{j=1}^n c_j x_j \\
 &\text{subject to} && \sum_{j=1}^n a_{ij} x_j \leq b_i, \quad i = 1, \dots, m \\
 &&& \sum_{j=1}^n c_{ij} x_j = d_i, \quad i = 1, \dots, p
 \end{aligned}$$

variables: $x_j \in \mathbf{R}$

problem data: the coefficients $c_j, a_{ij}, b_i, c_{ij}, d_i$

- can be solved very efficiently (several 10,000 variables, constraints)
- widely available high-quality software
- extensive, useful theory (optimality conditions, sensitivity analysis, . . .)

Example: open-loop control problem

single-input/single-output system (with input u , output y)

$$y(t) = h_0u(t) + h_1u(t-1) + h_2u(t-2) + h_3u(t-3) + \dots$$

output tracking problem: minimize deviation from desired output $y_{\text{des}}(t)$

$$\max_{t=0, \dots, N} |y(t) - y_{\text{des}}(t)|$$

subject to input amplitude and slew rate constraints:

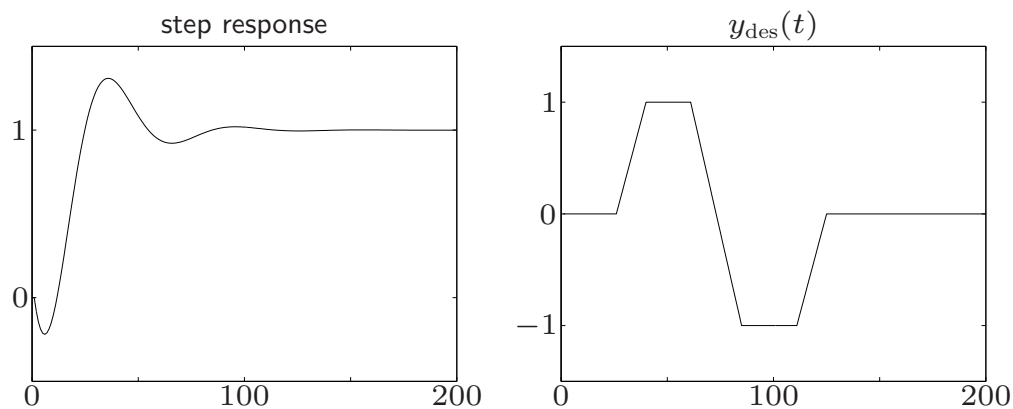
$$|u(t)| \leq U, \quad |u(t+1) - u(t)| \leq S$$

variables: $u(0), \dots, u(M)$ (with $u(t) = 0$ for $t < 0, t > M$)

solution: can be formulated as an LP, hence easily solved (more later)

example

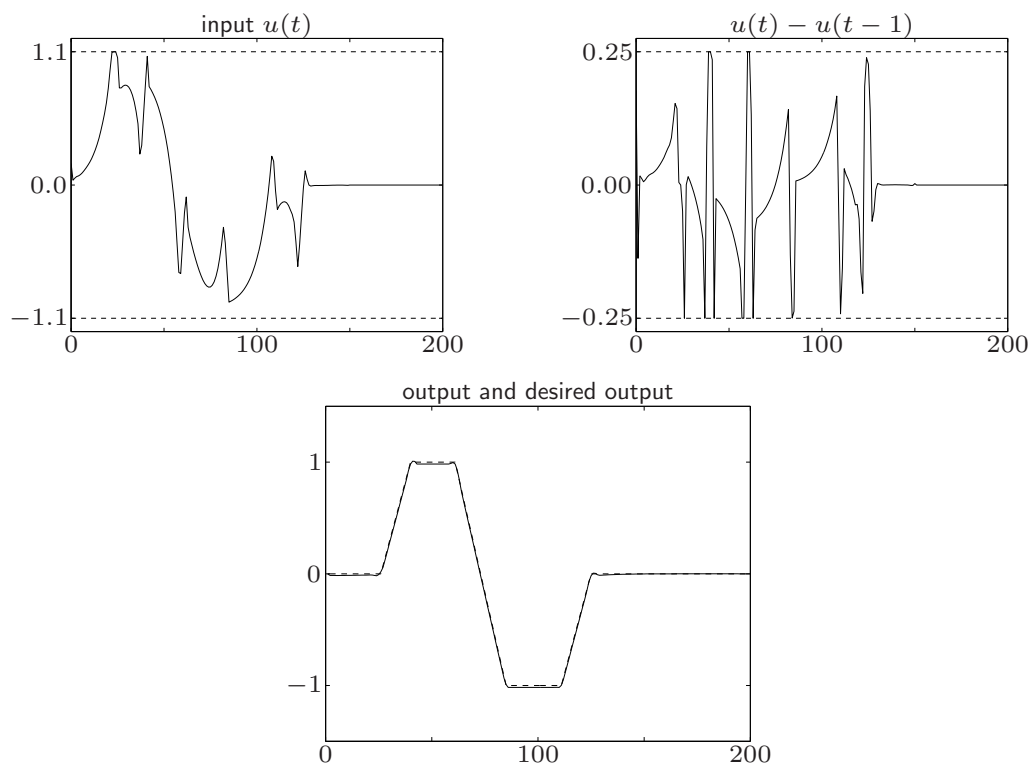
step response ($s(t) = h_t + \dots + h_0$) and desired output:



amplitude and slew rate constraint on u :

$$|u(t)| \leq 1.1, \quad |u(t) - u(t-1)| \leq 0.25$$

optimal solution (computed via linear programming)



Example: assignment problem

- match N people to N tasks
- each person is assigned one task; each task assigned to one person
- cost of assigning person i to task j is a_{ij}

combinatorial formulation

$$\begin{aligned} & \text{minimize} && \sum_{i,j=1}^N a_{ij}x_{ij} \\ & \text{subject to} && \sum_{i=1}^N x_{ij} = 1, \quad j = 1, \dots, N \\ & && \sum_{j=1}^N x_{ij} = 1, \quad i = 1, \dots, N \\ & && x_{ij} \in \{0, 1\}, \quad i, j = 1, \dots, N \end{aligned}$$

- variable $x_{ij} = 1$ if person i is assigned to task j ; $x_{ij} = 0$ otherwise
- $N!$ possible assignments, *i.e.*, too many to solve by enumeration

linear programming formulation

$$\begin{aligned} & \text{minimize} && \sum_{i,j=1}^N a_{ij}x_{ij} \\ & \text{subject to} && \sum_{i=1}^N x_{ij} = 1, \quad j = 1, \dots, N \\ & && \sum_{j=1}^N x_{ij} = 1, \quad i = 1, \dots, N \\ & && 0 \leq x_{ij} \leq 1, \quad i, j = 1, \dots, N \end{aligned}$$

- we have *relaxed* the constraints $x_{ij} \in \{0, 1\}$
- it can be shown that at the optimum $x_{ij} \in \{0, 1\}$ (see later)
- hence, can solve (this particular) combinatorial problem efficiently (via LP or specialized methods)

tractable combinatorial optimization problems usually have efficient equivalent LP formulations

Brief history

- **1940s** (Dantzig, Kantorovich, Koopmans, von Neumann, . . .): foundations of LP, motivated by economic and logistics problems
- **1947** (Dantzig): simplex algorithm
- **1950s–60s** applications in other fields (structural optimization, control theory, filter design, . . .)
- **1979** (Khachiyan) ellipsoid algorithm: more efficient (polynomial-time) than simplex in worst case, much slower in practice
- **1984** (Karmarkar): projective (interior-point) algorithm: polynomial-time worst-case complexity, and efficient in practice
- **1984–today**. variations of interior-point methods (improved complexity or efficiency in practice), software for large-scale problems

Course outline

the linear programming problem

linear inequalities, geometry of linear programming

applications

signal processing, control, structural optimization . . .

duality

algorithms

the simplex algorithm, interior-point algorithms

large-scale linear programming

techniques for LPs with special structure

integer linear programming

introduction, some basic techniques

Software

solvers: solve LPs described in some standard form

modeling tools: accept a problem in a simpler, more intuitive, notation and convert it to the standard form required by solvers

software for this course (see class website)

- platforms: Matlab, Octave, Python
- solvers: `linprog` (Matlab Optimization Toolbox), `lp236a.m` (Matlab/Octave), MOSEK (Matlab), CVXOPT (Python)
- modeling tools: CVX (Matlab), YALMIP (Matlab), CVXOPT (Python)