Recent Advances in Convex Optimization

Stephen Boyd

joint work with Michael Grant, Jacob Mattingley, Yang Wang

Electrical Engineering Department, Stanford University

Outline

- Convex optimization
- Some (simple) examples
- Parser/solvers for convex optimization
- Real-time embedded convex optimization

Optimization

- form mathematical model of real (design, analysis, synthesis, estimation, control, . . .) problem
- use computational algorithm to solve
- standard formulation:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

  x is the (decision) variable; f is the objective; $\mathcal{C}$ is the constraint set

- other formulations: multi-criterion/multi-level optimization, MDO, SAT problems, trade-off analysis, minimax, . . .

The good news

- **everything[1] is an optimization problem**

[1] *i.e.*, much of engineering design and analysis, data analysis

The bad news

- **you can't (really) solve most optimization problems**
- even simple looking problems are often intractable

Except for some special cases

- least-squares and variations (*e.g.*, optimal control, filtering)
- linear and quadratic programming
- **convex optimization**

well, OK, there are some other special cases

Convex optimization problem

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & x \in \mathcal{C} \end{array}$$

- $\mathcal{C}$ is convex (closed under averaging):

$$x, y \in \mathcal{C},\ \theta \in [0,1] \implies \theta x + (1-\theta) y \in C$$

- f is convex (graph of f curves upward):

$$\theta \in [0,1] \implies f(\theta x + (1-\theta) y) \leq \theta f(x) + (1-\theta) f(y)$$

- not always easy to recognize/validate convexity

Convex optimization

- (no analytical solutions, but) can solve convex optimization problems **extremely well** (in theory and practice)
  - get global solutions, with optimality certificate
  - problems with 10^3–10^5 variables, constraints solved by generic methods on generic processor
  - (much) larger problems solved by iterative methods and/or on multiple processors
  - differentiability plays a minor role
- beautiful (and fairly complete) theory

Applications of convex optimization

- convex problems come up much more often than was once thought
- many applications recently discovered in
  - control
  - combinatorial optimization
  - signal processing
  - image processing
  - communications, networking
  - analog and digital circuit design
  - statistics, machine learning
  - finance

Some recent (general) developments

- **robust optimization methods** that handle parameter variation, optimizing average or worst-case performance, quantiles, . . .
- ℓ_1-**based heuristics** for finding sparse solutions (compressed sensing, feature selection, . . .)
- **parser/solvers** make rapid prototyping easy
- **code generators** yield solvers that can be embedded in real-time systems

Outline

- Convex optimization
- Some (simple) examples
- Parser/solvers for convex optimization
- Real-time embedded convex optimization

Multi-period processor speed scheduling

- processor adjusts its speed $s_t \in [s^{\min}, s^{\max}]$ in each of T time periods
- energy consumed in period t is $\phi(s_t)$; total energy is $E = \sum_{t=1}^{T} \phi(s_t)$
- n jobs
  - job i available at $t = A_i$; must finish by deadline $t = D_i$
  - job i requires total work $W_i \geq 0$
- $S_{ti} \geq 0$ is effective processor speed allocated to job i in period t

$$s_t = \sum_{i=1}^{n} S_{ti}, \qquad \sum_{t=A_i}^{D_i} S_{ti} \geq W_i$$

Minimum energy processor speed scheduling

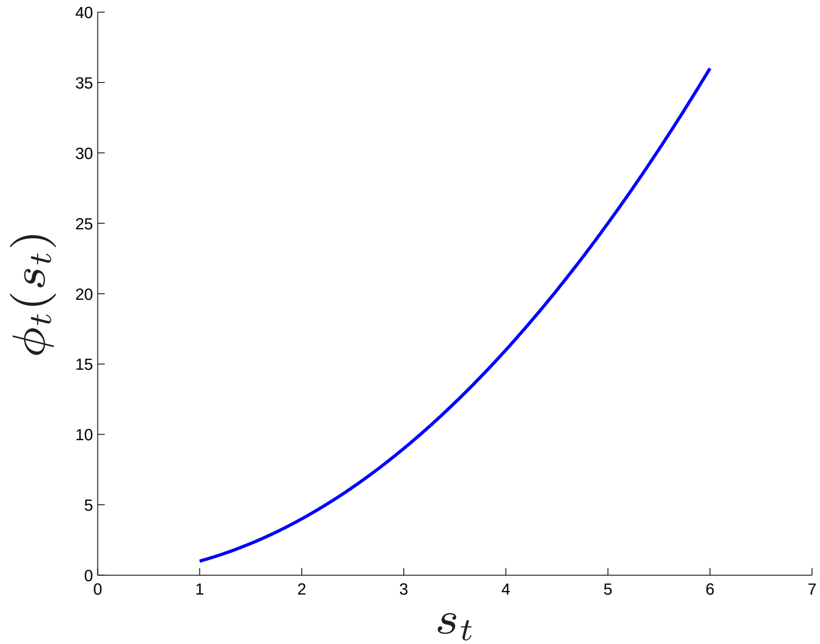
- choose speeds s_t and allocations S_{ti} to minimize total energy E

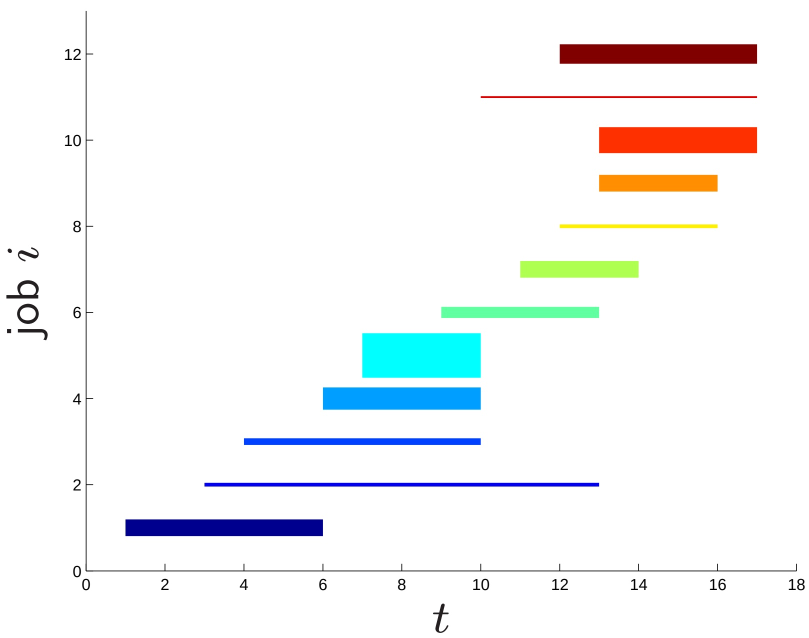
$$\begin{array}{ll} \text{minimize} & E = \sum_{t=1}^{T} \phi(s_t) \\ \text{subject to} & s^{\min} \leq s_t \leq s^{\max}, \quad t = 1, \ldots, T \\ & s_t = \sum_{i=1}^{n} S_{ti}, \quad t = 1, \ldots, T \\ & \sum_{t=A_i}^{D_i} S_{ti} \geq W_i, \quad i = 1, \ldots, n \end{array}$$

- a convex problem when ϕ is convex
- more sophisticated versions include
  - multiple processors
  - other constraints (thermal, speed slew rate, . . .)
  - stochastic models for (future) data

Example

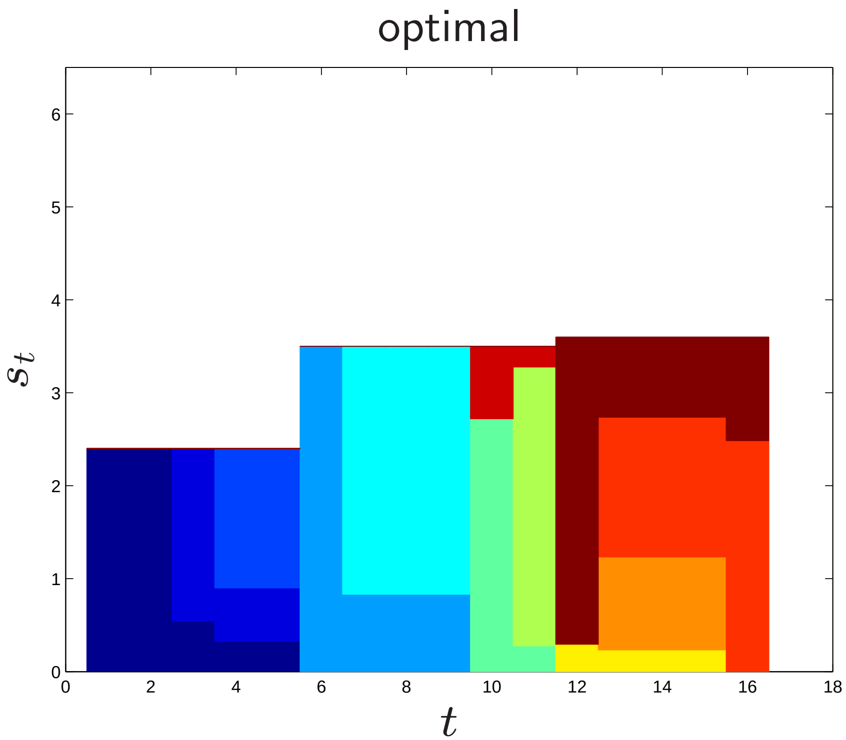
- $T = 16$ periods, $n = 12$ jobs
- $s^{\min} = 1$, $s^{\max} = 6$, $\phi(s_t) = s_t^2$
- jobs shown as bars over $[A_i, D_i]$ with area $\propto W_i$

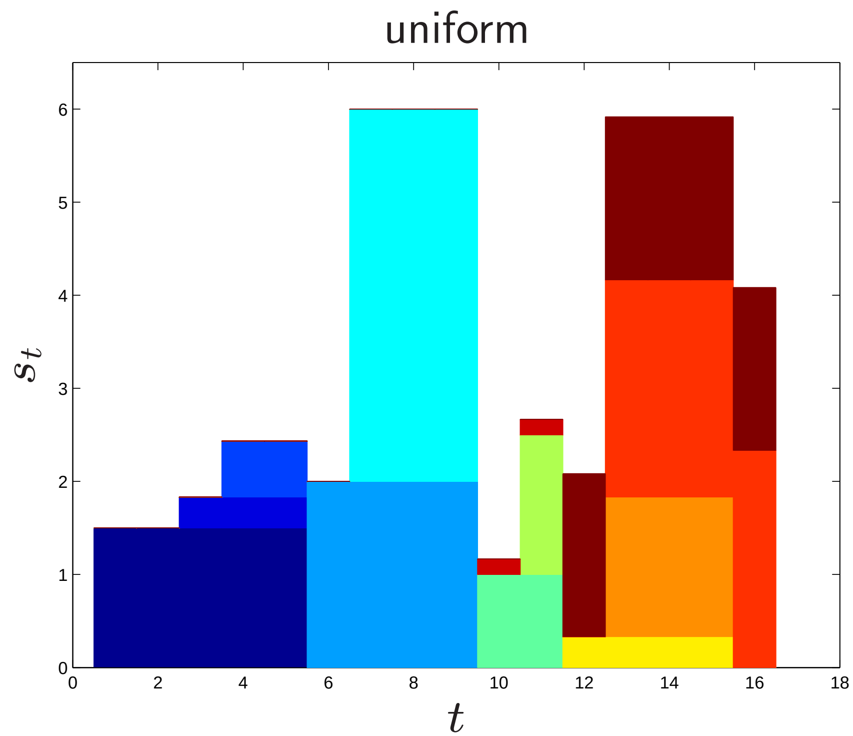




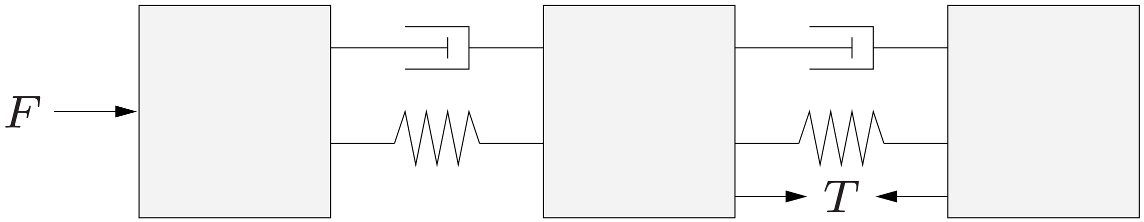
Optimal and uniform schedules

- uniform schedule: $S_{ti} = W_i/(D_i - A_i)$; gives $E^{\text{unif}} = 374.1$
- optimal schedule $S^\star_{ti}$ gives $E^\star = 167.1$





Minimum time control with active vibration supression

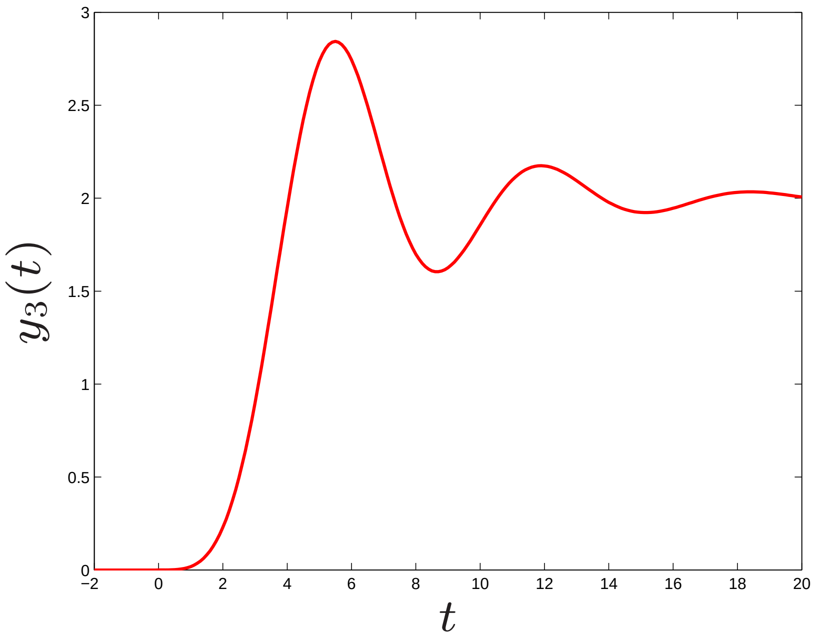


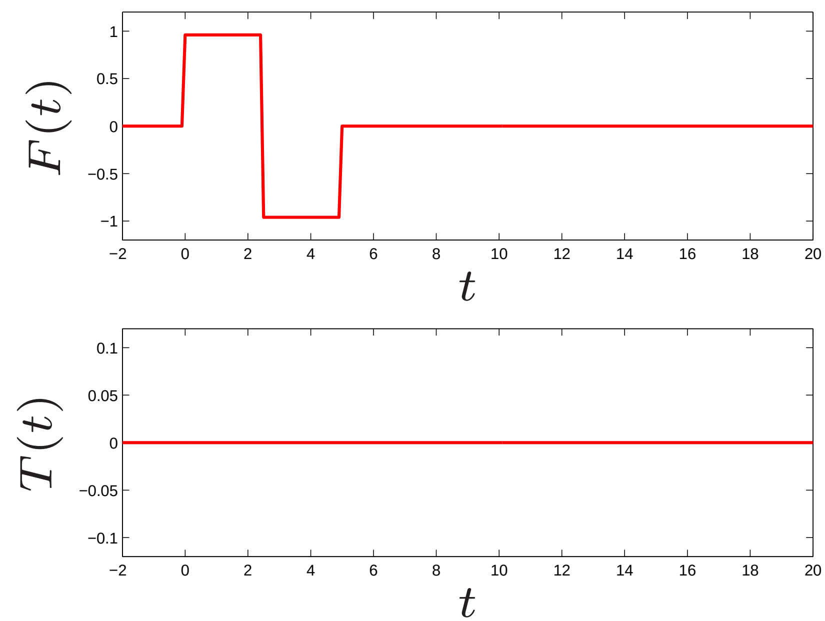
- force $F(t)$ moves object modeled as 3 masses (2 vibration modes)
- tension $T(t)$ used for active vibration supression
- goal: move object to commanded position as quickly as possible, with

$$|F(t)| \leq 1, \qquad |T(t)| \leq 0.1$$

Ignoring vibration modes

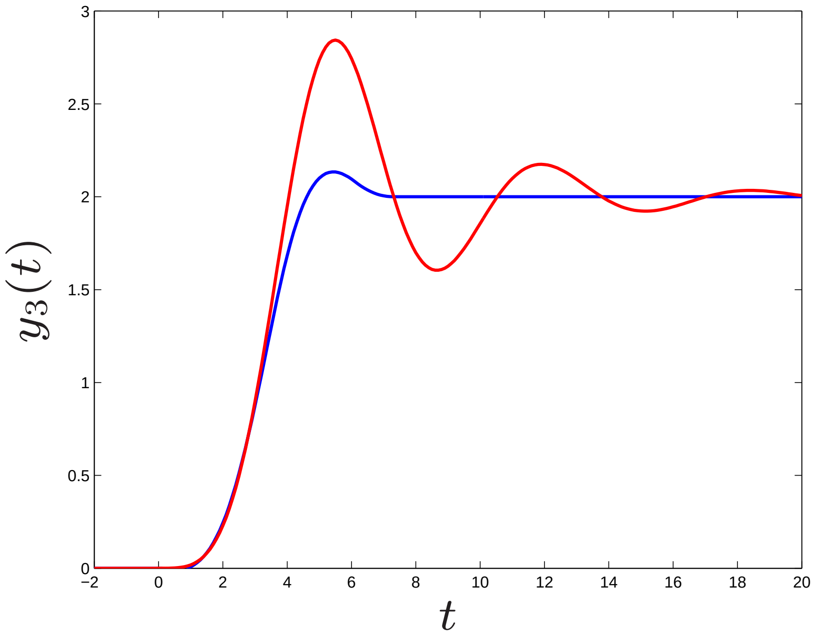
- treat object as single mass; apply only F
- analytical ('bang-bang') solution

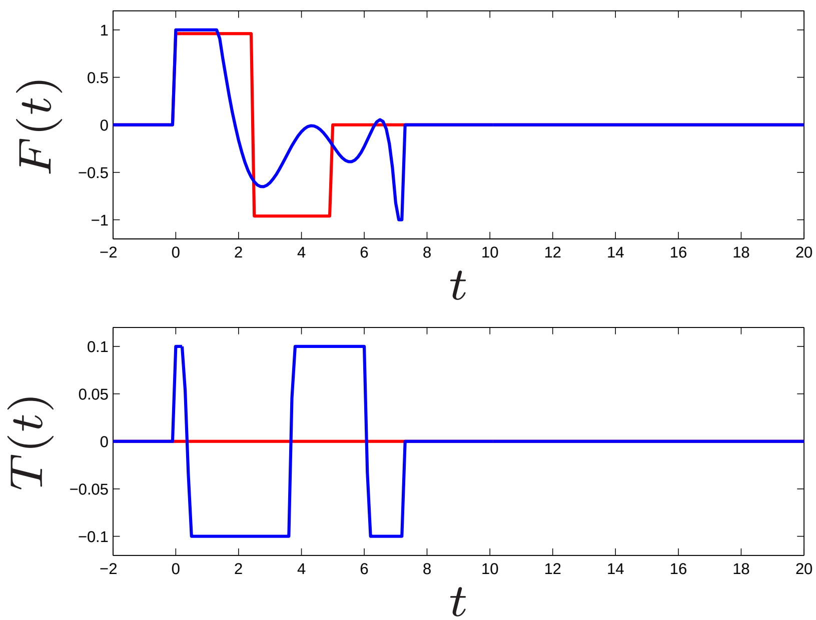




With vibration modes

- no analytical solution, but reduces to a quasiconvex problem
- can be solved by solving a small number of convex problems





Network utility maximization

- network with m links and n flows
- flow j has (nonnegative) flow rate f_j
- each flow passes over a fixed set of links (its route)
- total link traffic (sum of flows through it) cannot exceed capacity c_i
- choose flow rates to maximize utility $U(f) = \sum_{i=1}^{n} U_j(f_j)$
- U_j increasing and concave, *e.g.*,
  - $U_j(f_j) = \log f_j$ (log utility)
  - $U_j(f_j) = w_j \min\{f_j, s_j\}$ (linear with satiation)

Network utility maximization

- can express link capacity constraints as $Rf \leq c$, with

$$R_{ij} = \begin{cases} 1 & \text{flow } j \text{ passes through link } i \\ 0 & \text{otherwise} \end{cases}$$

- NUM problem is

$$\begin{array}{ll} \text{maximize} & U(f) \\ \text{subject to} & Rf \leq c, \quad f \geq 0 \end{array}$$

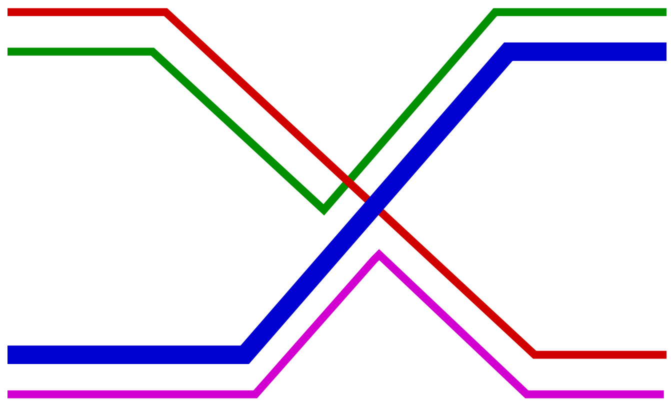
a convex optimization problem

- 'solved' (approximately) by distributed protocols

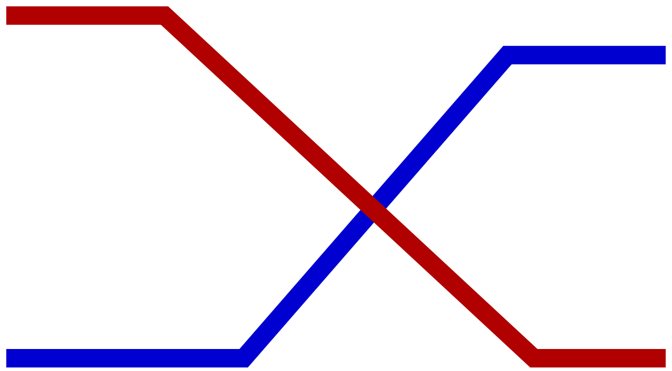
Example

- $U_j(f_j) = \min\{f_j, s_j\}$; $c = (2, 4, 4, 2)$, $s = (2, 1, 2, 3)$
- greedy flows: optimize over f_1, then f_2, . . .

optimal, $U^\star = 5$



greedy, $U = 4$



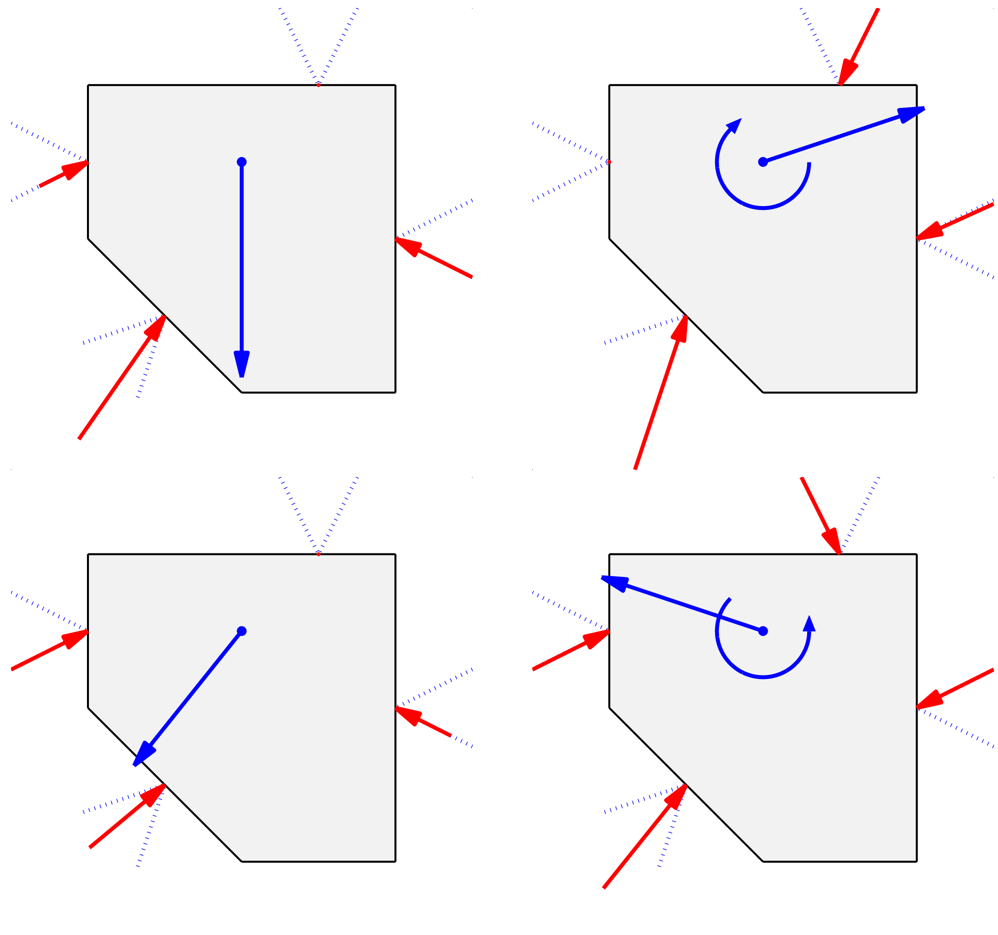
Grasp force optimization

- choose K grasping forces on object
  - resist external wrench
  - respect friction cone constraints
  - minimize maximum grasp force
- convex problem (second-order cone program):

minimize	$\max_i \|f^{(i)}\|_2$	*max contact force*
subject to	$\sum_i Q^{(i)} f^{(i)} = f^{\text{ext}}$	*force equillibrium*
	$\sum_i p^{(i)} \times (Q^{(i)} f^{(i)}) = \tau^{\text{ext}}$	*torque equillibrium*
	$\mu_i f_3^{(i)} \geq \left(f_1^{(i)2} + f_2^{(i)2}\right)^{1/2}$	*friction cone contraints*

variables $f^{(i)} \in \mathbf{R}^3$, $i = 1, \ldots, K$ (contact forces)

Example



Outline

- Convex optimization
- Some (simple) examples
- Parser/solvers for convex optimization
- Real-time embedded convex optimization

Parser/solvers for convex optimization

- specify convex problem in natural form
  - declare optimization variables
  - form convex objective and constraints using a specific set of atoms and calculus rules
- problem is convex-by-construction
- easy to parse, automatically transform to standard form, solve, and transform back
- implemented using object-oriented methods and/or compiler-compilers
- huge gain in productivity (rapid prototyping, teaching, research ideas)

Example (cvx)

convex problem, with variable $x \in \mathbf{R}^n$:

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|_2 + \lambda \|x\|_1 \\ \text{subject to} & Fx \leq g \end{array}$$

cvx specification:

```
cvx_begin
    variable x(n)     % declare vector variable
    minimize (norm(A*x-b,2) + lambda*norm(x,1))
    subject to  F*x <= g
cvx_end
```

when `cvx` processes this specification, it

- verifies convexity of problem
- generates equivalent IPM-compatible problem
- solves it using `SDPT3` or `SeDuMi`
- transforms solution back to original problem

the `cvx` code is easy to read, understand, modify

The same example, transformed by 'hand'

transform problem to SOCP, call `SeDuMi`, reconstruct solution:

```
% Set up big matrices.
[m,n] = size(A); [p,n] = size(F);
AA = [speye(n), -speye(n), speye(n), sparse(n,p+m+1); ...
      F, sparse(p,2*n), speye(p), sparse(p,m+1); ...
      A, sparse(m,2*n+p), speye(m), sparse(m,1)];
bb = [zeros(n,1); g; b];
cc = [zeros(n,1); gamma*ones(2*n,1); zeros(m+p,1); 1];
K.f = m; K.l = 2*n+p; K.q = m + 1;       % specify cone
xx = sedumi(AA, bb, cc, K);              % solve SOCP
x = x(1:n);                              % extract solution
```

Outline

- Convex optimization
- Some (simple) examples
- Parser/solvers for convex optimization
- Real-time embedded convex optimization

Real-time embedded optimization

- embed solvers in real-time applications (signal processing, control, . . .)
  $i.e.$, **solve an optimization problem at each time step**

- requires solvers that are fast, with known maximum execution time

- used now for applications with hour/minute time-scales
  (process control, supply chain and revenue 'management', trading . . .)

- **new methods allows millisecond/microsecond time-scales**

Solving specific problems

in developing a custom solver for a specific application, we can

- exploit structure very efficiently
- determine ordering, memory allocation beforehand
- cut corners in algorithm, $e.g.$, terminate early
- use warm start

to get **very fast** solver

Code generation

- describe optimization problem (family) in high level form
- automatically generate solver source code
- can do much at code generation time
- yields super fast solvers suitable for real-time embedded applications

Example: `cvxmod` specification

quadratic program, with variable $x \in \mathbf{R}^n$:

$$\begin{array}{ll} \text{minimize} & x^T P x + q^T x \\ \text{subject to} & Gx \leq h, \quad Ax = b \end{array}$$

`cvxmod` specification:

```
A = matrix(...); b = matrix(...)
P = param('P', n, n, psd=True); q = param('q', n)
G = param('G', m, n); h = param('h', m)
x = optvar('x', n)
qpfam = problem(minimize(quadform(x, P) + tp(q)*x),
                [G*x <= h, A*x == b])
```

Example: `cvxmod` code generation

- generate solver for problem family `qpfam` with

```
qpfam.codegen()
```

- output includes `qpfam/solver.c` and ancillary files
- solve instance with

```
status = solve(params, vars, work);
```

Sample solve times

problem family	vars	constrs	SDPT3 (ms)	cvxmod (ms)
control1	140	190	250	0.4
control2	360	1080	1400	2.0
control3	1110	3180	3400	13.2
order_exec	20	41	490	0.05
net_utility	50	150	130	0.23
actuator	50	106	300	0.17
grasp	30	66	300	0.05

Conclusions

- convex optimization problems come up in many application areas
- new tools make rapid prototyping easy
- new code generation methods yield solvers that can be embedded in real-time applications