4.13 Robust LP with interval coefficients

For the robust LP:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \preceq b \quad \text{for all } A \in \mathcal{A} \end{array}$$

where $\mathcal{A} = \{A \in \mathbf{R}^{m\times n} \mid \bar{A}_{ij} - V_{ij} \leq A_{ij} \leq \bar{A}_{ij} + V_{ij}\}$. So each coefficient of A (uncertainty set) lies in a box/interval.

Reformulation as an LP

For each constraint i, the robust requirement is

$$\sum_{j=1}^{n} A_{ij} x_j \leq b_i \quad \text{for all } A \in \mathcal{A}$$

This is equivalent to requiring that the worst-case value of the left-hand side satisfies the inequality:

$$\sup_{A\in\mathcal{A}} (Ax)_i \leq b_i$$

Since $\mathcal{A}$ is a compact set and $(Ax)_i$ is linear in A, the supremum is attained and equals a maximum:

$$\max_{A\in\mathcal{A}} \sum_{j=1}^{n} A_{ij} x_j \leq b_i$$

Because the uncertainty set is a product of independent intervals, the maximization separates across j, giving

$$\sum_{j=1}^{n} \max_{A_{ij}\in[\bar{A}_{ij}-V_{ij},\, \bar{A}_{ij}+V_{ij}]} A_{ij} x_j \leq b_i$$

For each term,

$$\max_{A_{ij}\in[\bar{A}_{ij}-V_{ij},\, \bar{A}_{ij}+V_{ij}]} A_{ij} x_j = \bar{A}_{ij} x_j + V_{ij} |x_j|$$

Therefore the robust constraints are equivalent to

$$\bar{A}x + V|x| \preceq b.$$

To express this as a linear program, introduce auxiliary variables $y \in \mathbb{R}^n$ satisfying

$$y_j \geq |x_j| \quad \text{for } j = 1, \ldots, n$$

which is equivalent to the linear inequalities

$$y \succeq x, \qquad y \succeq -x$$

The final LP formulation is

$$\begin{array}{ll} \text{minimize} & c^\top x \\ \text{subject to} & \bar{A}x + Vy \preceq b \\ & -y \preceq x \preceq y \end{array}$$

A4.25 Probability bounds

Given random variables $X_1, X_2, X_3, X_4 \in \{0, 1\}$ with:

$$\begin{aligned}
\textbf{prob}(X_1 = 1) &= 0.9 \\
\textbf{prob}(X_2 = 1) &= 0.9 \\
\textbf{prob}(X_3 = 1) &= 0.1 \\
\textbf{prob}(X_1 = 1, X_4 = 0 \mid X_3 = 1) &= 0.7 \\
\textbf{prob}(X_4 = 1 \mid X_2 = 1, X_3 = 0) &= 0.6
\end{aligned}$$

Solution

The objective quantity to bound (linear in p) is:

$$\textbf{prob}(X_4 = 1) = \sum_{i,j,k} p_{ijk1}$$

Then, identify the constraints as:

- The joint distribution as a vector in $\mathbb{R}^{16}$:

$$p_{ijkl} = \textbf{prob}(X_1 = i, X_2 = j, X_3 = k, X_4 = l), \quad i, j, k, l \in \{0, 1\}$$

- Probability simplex constraints:

$$p_{ijkl} \geq 0 \ \ \forall i, j, k, l, \qquad \sum_{i,j,k,l} p_{ijkl} = 1.$$

- Given marginals as linear constraints:
  - $\textbf{prob}(X_1 = 1) = 0.9 \ \rightarrow \ \sum_{j,k,l} p_{1jkl} = 0.9$
  - $\textbf{prob}(X_2 = 1) = 0.9 \ \rightarrow \ \sum_{i,k,l} p_{i1kl} = 0.9$
  - $\textbf{prob}(X_3 = 1) = 0.1 \ \rightarrow \ \sum_{i,j,l} p_{ij1l} = 0.1$
- Given conditionals converted to linear constraints:
  - The $\textbf{prob}(X_1 = 1, X_4 = 0 \mid X_3 = 1) = 0.7$ means

$$\frac{\textbf{prob}(X_1 = 1, X_4 = 0, X_3 = 1)}{\textbf{prob}(X_3 = 1)} = 0.7$$

hence

$$\sum_j p_{1j10} = 0.7 \sum_{i,j,l} p_{ij1l} = 0.07$$

– The $\mathbf{prob}(X_4 = 1 \mid X_2 = 1, X_3 = 0) = 0.6$ means

$$\frac{\mathbf{prob}(X_4 = 1, X_2 = 1, X_3 = 0)}{\mathbf{prob}(X_2 = 1, X_3 = 0)} = 0.6$$

hence

$$\sum_i p_{i101} = 0.6 \sum_{i,l} p_{i10l}.$$

So the solution involves solving two LPs (minimize and maximize) for the given objective, subject to the constraints above.

Code:

```
p = cp.Variable((2, 2, 2, 2), nonneg=True)  # so each p is >= 0
constraints = [
    cp.sum(p) == 1,
    cp.sum(p[1, :, :, :]) == 0.9,       # prob(X1=1)
    cp.sum(p[:, 1, :, :]) == 0.9,       # prob(X2=1)
    cp.sum(p[:, :, 1, :]) == 0.1,       # prob(X3=1)
    cp.sum(p[1, :, 1, 0]) == 0.07,      # prob(X1=1,X4=0,X3=1)
    cp.sum(p[:, 1, 0, 1]) == 0.6 * cp.sum(p[:, 1, 0, :])
]
prob_X4_1 = cp.sum(p[:, :, :, 1])

prob_min = cp.Problem(cp.Minimize(prob_X4_1), constraints)
prob_min.solve()
p_min = prob_min.value

prob_max = cp.Problem(cp.Maximize(prob_X4_1), constraints)
prob_max.solve()
p_max = prob_max.value
```

Results:

- Minimum $\mathbf{prob}(X_4 = 1) = \boxed{0.48}$
- Maximum $\mathbf{prob}(X_4 = 1) = \boxed{0.61}$

A6.13 Fitting with censored data

(a)

We have data points $(x^{(k)}, y^{(k)})$ for $k = 1, \ldots, K$, where:

- $y^{(1)}, \ldots, y^{(M)}$ are observed (uncensored)
- $y^{(M+1)}, \ldots, y^{(K)}$ are censored (known to be $> D$)

For censored data, $y^{(k)}$ for $k > M$ are optimization variables with constraints $y^{(k)} \geq D$.

So the problem becomes:

$$\begin{array}{ll} \text{minimize} & \sum_{k=1}^{M} (y^{(k)} - c^T x^{(k)})^2 + \sum_{k=M+1}^{K} (y^{(k)}_{\text{cens}} - c^T x^{(k)})^2 \\ \text{s.t.} & y^{(k)}_{\text{cens}} \geq D, \quad k = M+1, \ldots, K \end{array}$$

(b)

Censored fitting estimate $\hat{c}$:

```
[-0.406  0.408 -0.323 -0.649  0.340 -1.865 -0.918 -1.171 -0.345 -0.466
 -0.216  0.253  0.524  0.315 -0.505  0.584 -0.186  1.639  0.682  0.107]
```

Least-squares estimate $\hat{c}_{ls}$ (ignoring censored data):

```
[-0.869  0.388 -0.079 -0.527  0.448 -2.146 -0.792 -0.866 -0.183 -0.251
 -0.158  0.555  0.428  0.069 -0.434  0.357 -0.202  2.007  0.811  0.094]
```

Relative errors:

$$\frac{\|c_{\text{true}} - \hat{c}\|_2}{\|c_{\text{true}}\|_2} = \boxed{0.131} \quad \text{(censored fitting)}$$

$$\frac{\|c_{\text{true}} - \hat{c}_{ls}\|_2}{\|c_{\text{true}}\|_2} = \boxed{0.333} \quad \text{(ignoring censored data)}$$

The censored fitting method achieves significantly lower error than simply ignoring the censored data points, demonstrating the value of incorporating the constraint information.

Code:

```python
# Censored fitting
c_var = cp.Variable((n, 1))
y_cens = cp.Variable((K - M, 1))
residuals_obs = y - X_obs.T @ c_var
residuals_cens = y_cens - X_cens.T @ c_var
objective = cp.sum_squares(residuals_obs) + cp.sum_squares(residuals_cens)
constraints = [y_cens >= D]
prob = cp.Problem(cp.Minimize(objective), constraints)
prob.solve()
c_hat = c_var.value

# Least squares ignoring censored data
c_ls = np.linalg.lstsq(X_obs.T, y.flatten(), rcond=None)[0]

```

```
# Compute relative errors
rel_error_censored = np.linalg.norm(c_true.flatten() - c_hat.flatten()) / np.
    linalg.norm(c_true.flatten())
rel_error_ls = np.linalg.norm(c_true.flatten() - c_ls.flatten()) / np.linalg.norm(
    c_true.flatten())
```

A7.1 Maximum likelihood estimation of x and noise mean and covariance

The log-likelihood is:

$$\sum_{i=1}^{m} \log p(y_i - a_i^T x) = -m \log \sigma + \sum_{i=1}^{m} \log f\left(\frac{y_i - a_i^T x - \mu}{\sigma}\right)$$

Let $r_i = y_i - a_i^T x - \mu$ denote the residuals.

If f is log-concave, then $\log f$ is concave. It is required to show that the composition $\log f(r_i/\sigma)$ preserves concavity.

Consider $g(r, \sigma) = \log f(r/\sigma)$ for $\sigma > 0$. The function r/σ is quasilinear (linear in r for fixed σ, and the perspective operation preserves concavity).

Using the change of variables $\tau = 1/\sigma^2$ and noting that:

- $-m \log \sigma = \frac{m}{2} \log \tau$ is concave in $\tau > 0$
- For log-concave f, $\log f(r\sqrt{\tau})$ is jointly concave in (r, τ)

Therefore, maximizing the log-likelihood is equivalent to maximizing a concave function, a convex optimization problem.

A15.13 Bandlimited signal recovery from zero-crossings

(a)

Let $F \in \mathbf{R}^{n\times 2B}$ be the Fourier basis matrix. Then $y = F\theta$ where $\theta = [a; b]$.

Optimization problem:

$$\begin{array}{ll} \text{minimize} & \|F\theta\|_2 \\ \text{subject to} & s_t(F\theta)_t \geq 0, \quad t = 1, \ldots, n \\ & (F\theta)_{t_0} = 1 \quad \text{(normalization)} \end{array}$$

The sign constraints $s_t y_t \geq 0$ ensure consistency with observed signs. The normalization fixes the scale (since y and αy for $\alpha > 0$ have the same signs).

This is a convex SOCP (second-order cone program).

(b) Results

- Signal length: $n = 2048$
- Bandwidth: $B = 9$, $f_{\min} = 4$
- Relative recovery error: $\|\hat{y} - y\|_2 / \|y\|_2 = \boxed{0.146}$
- Sign matches: 2033/2048 (99.3%)

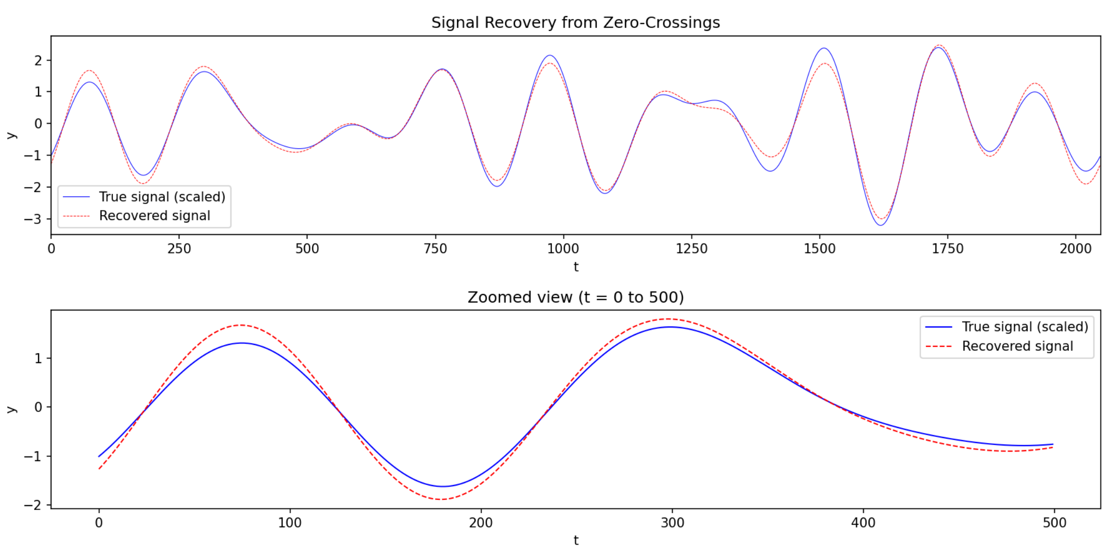

Figure 1: Recovered signal vs true signal. The recovery captures the main structure with moderate accuracy.

Code:

```
F = np.zeros((n, 2 * B))
for j in range(B):
    freq = f_min + j
    F[:, j] = np.cos(2 * np.pi * freq * t / n)
```

```
    F[:, B + j] = np.sin(2 * np.pi * freq * t / n)

coeff = cp.Variable(2 * B)
y_var = F @ coeff
constraints = [cp.multiply(s, y_var) >= 0, y_var[pos_idx] == 1]
prob = cp.Problem(cp.Minimize(cp.norm2(y_var)), constraints)
```

A17.30 Maximum Sharpe ratio portfolio

%textbfProblem: Maximize $S(x)$ subject to $\mathbf{1}^T x = 1$ and $\|x\|_1 \leq L^{\max}$.

(a)

The Sharpe ratio $S(x)$ is quasiconvex for $\mu^T x > 0$.

The sublevel set $\{x : S(x) \leq t\}$ for $t > 0$ is:

$$\{x : \mu^T x \leq t\|\Sigma^{1/2} x\|_2\}$$

This is equivalent to $\|\Sigma^{1/2} x\|_2 \geq (\mu^T x)/t$, which defines a convex set (second-order cone constraint). Hence $S(x)$ is quasiconvex.

(b) Transformation to convex problem

Let $y = x/(\mu^T x)$ (for $\mu^T x > 0$). Then:

- $\mu^T y = 1$
- $\mathbf{1}^T y = 1/(\mu^T x)$
- $S(x) = 1/\|\Sigma^{1/2} y\|_2$

Maximizing $S(x)$ is equivalent to minimizing $\|\Sigma^{1/2} y\|_2$ subject to $\mu^T y = 1$ and the transformed leverage constraint $\|y\|_1 \leq L^{\max} \cdot \mathbf{1}^T y$.

Results (for $n = 10$ assets, $L^{\max} = 1.5$):

- Expected return: 0.129
- Standard deviation: 0.061
- Sharpe ratio: $\boxed{2.13}$
- Leverage $\|x\|_1$: 1.32

Code:

```
y = cp.Variable(n)
constraints = [mu @ y == 1, cp.sum(y) >= 0,
               cp.norm1(y) <= L_max * cp.sum(y)]
prob = cp.Problem(cp.Minimize(cp.quad_form(y, Sigma)), constraints)
prob.solve()
x_opt = y.value / np.sum(y.value)
```

A19.10 Scheduling

We have n tasks with:

- Start time $s_i \geq 0$, finish time $f_i \geq s_i$
- Duration $d_i = f_i - s_i \geq m_i$ (minimum duration)
- Cost $\phi_i(d_i) = \alpha_i/(d_i - m_i)$ for $d_i > m_i$
- Precedence constraints: $(i, j) \in \mathcal{P}$ means task j starts after task i finishes
- Completion time $T = \max_i f_i$

(a) Pareto optimal trade-off

For fixed T, minimize total cost:

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{i=1}^{n} \frac{\alpha_i}{d_i - m_i} \\ \text{subject to} & s_i \geq 0 \\ & d_i \geq m_i + \epsilon \\ & s_i + d_i \leq T \\ & s_j \geq s_i + d_i \quad \forall (i, j) \in \mathcal{P} \end{array}$$

This is convex since $1/(d_i - m_i)$ is convex for $d_i > m_i$.
By solving for various $T \in [T_{\min}, T_{\max}]$, we trace out the Pareto frontier.

(b) Results

Code:

```
def solve_schedule(T_max):
    s = cp.Variable(n)
    d = cp.Variable(n)
    f = s + d
    cost = cp.sum(cp.multiply(alpha, cp.inv_pos(d - m)))
    constraints = [s >= 0, d >= m + 1e-4, f <= T_max]
    for i, j in P:
        constraints.append(s[j] >= f[i])
    prob = cp.Problem(cp.Minimize(cost), constraints)
    prob.solve()
    return prob.value, s.value, f.value
```

Pareto frontier: $T \in [10, 30]$, Cost $C \in [1.59, 8.26]$

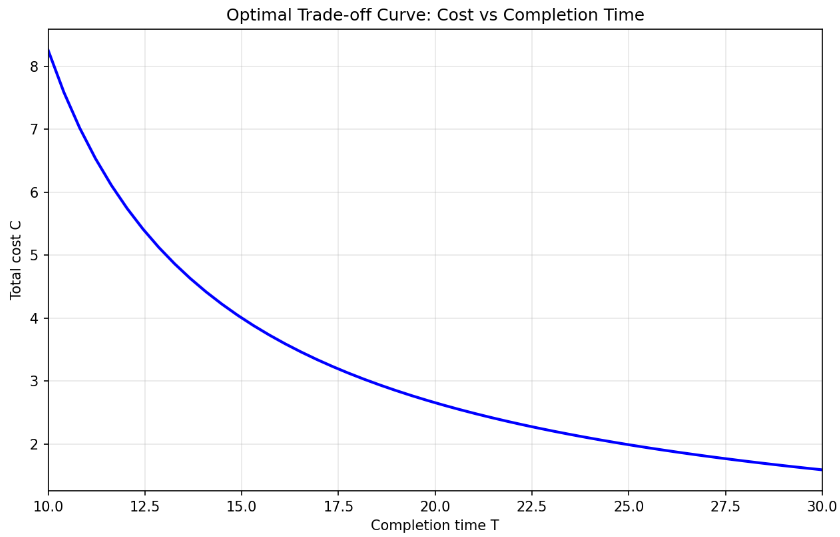

Figure 2: Optimal trade-off curve: Cost vs Completion time

Optimal schedule for $T = 20$:

Task	Start	Finish	Duration	Cost
1	0.00	8.03	8.03	0.21
2	0.00	5.19	5.19	0.49
3	5.19	11.97	6.78	0.18
4	5.19	10.64	5.45	0.36
5	5.19	20.00	14.81	0.08
6	0.00	15.49	15.49	0.08
7	8.03	15.49	7.46	0.20
8	11.97	20.00	8.03	0.21
9	10.64	15.49	4.84	0.38
10	15.49	20.00	4.51	0.46
			Total Cost:	$\boxed{2.66}$

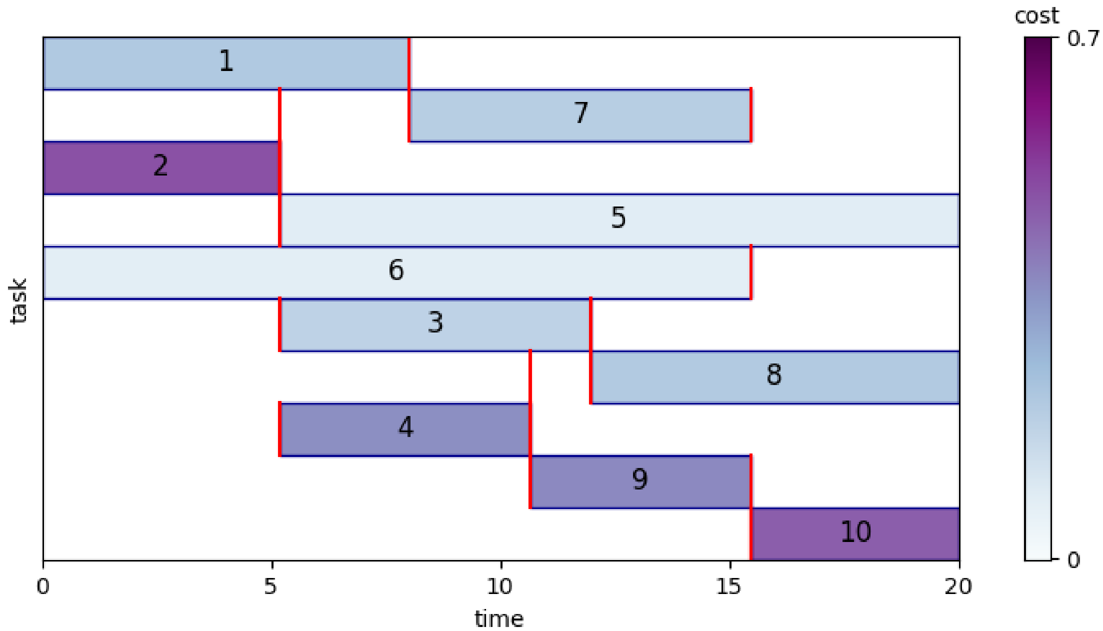

Figure 3: Gantt chart of optimal schedule for $T = 20$