5.1 A simple example

(a)

- Feasible set: from the constraint we obtain$\{x \mid 2 \leq x \leq 4\}$.
- Optimal point: since objective is a convex function, $x^\star = 2$ is the point that minimizes the objective
- Optimal value: $p^\star = 5$

(b)

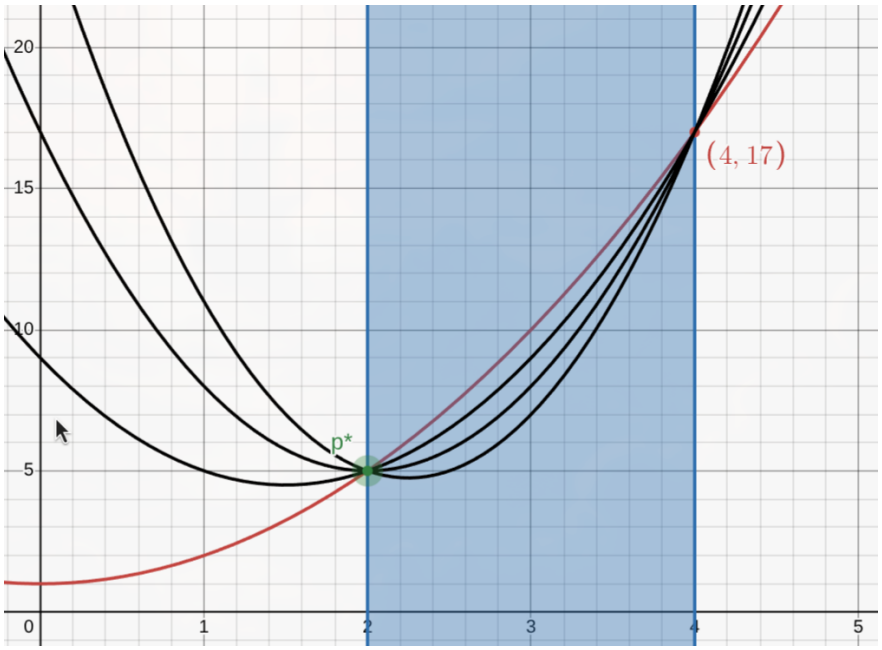

Figure 1: Plot of objective f_0 (red), showing feasible set (blue) and optimal point (green).

The constraint can be rewritten as $f_1(x) = (x-2)(x-4) = x^2 - 6x + 8 \leq 0$.
The Lagrangian is:

$$L(x, \lambda) = f_0(x) + \lambda f_1(x) = (x^2 + 1) + \lambda(x^2 - 6x + 8) = (1+\lambda)x^2 - 6\lambda x + 1 + 8\lambda$$

The dual function is $g(\lambda) = \inf_x L(x, \lambda)$, so we have to minimize the Lagrangian over x.

$$\frac{\partial L}{\partial x} = 2(1+\lambda)x - 6\lambda = 0 \implies x^\star = \frac{3\lambda}{1+\lambda}$$

Substituting back:

$$\begin{aligned} g(\lambda) = L(x^\star, \lambda) &= (1+\lambda)\left(\frac{3\lambda}{1+\lambda}\right)^2 - 6\lambda \cdot \frac{3\lambda}{1+\lambda} + 1 + 8\lambda \\ &= \frac{9\lambda^2}{1+\lambda} - \frac{18\lambda^2}{1+\lambda} + 1 + 8\lambda = \frac{-9\lambda^2}{1+\lambda} + 1 + 8\lambda \end{aligned}$$

For $\lambda \leq -1$, the Lagrangian is unbounded below.

Therefore:

$$g(\lambda) = \begin{cases} \dfrac{-9\lambda^2}{1+\lambda} + 1 + 8\lambda & \lambda > -1 \\ -\infty & \lambda \leq -1 \end{cases}$$

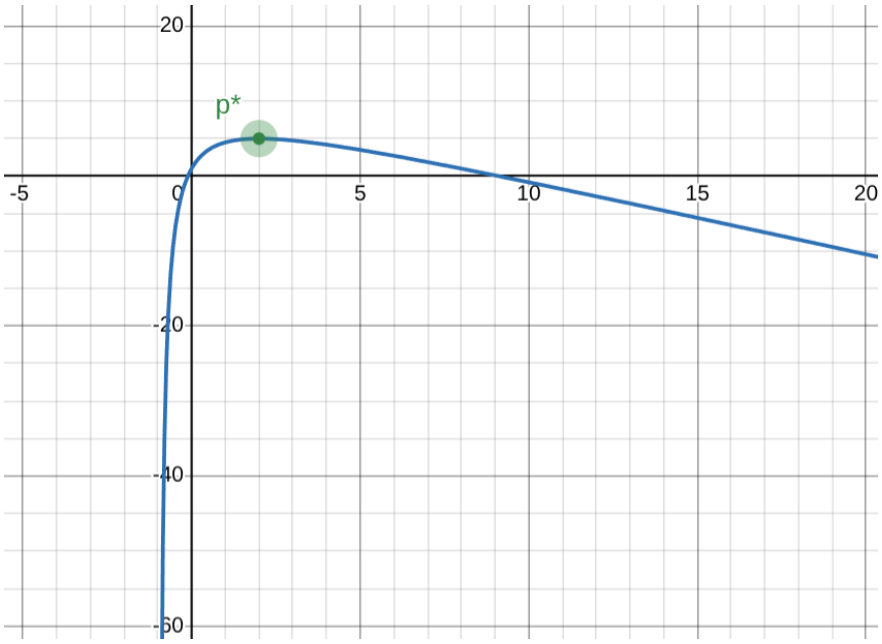

Figure 2: Plot of dual function $g(\lambda)$

We can see that $g(\lambda) \leq p^\star = 5$ for all $\lambda \geq 0$, verifying the lower bound property. Strong equality $g(\lambda) = p^\star$ holds at $\lambda = 2$.

(c)

$$\begin{aligned} \text{maximize} \quad & \frac{-9\lambda^2}{1+\lambda} + 1 + 8\lambda \\ \text{s.t.} \quad & \lambda \geq 0 \end{aligned}$$

It is a concave maximization problem for $\lambda > -1$, as seen in the graph.
To find the optimal $\lambda^\star$, we take the derivative and set it to zero:

$$g'(\lambda) = \frac{-18\lambda(1+\lambda) + 9\lambda^2}{(1+\lambda)^2} + 8 = \frac{-9\lambda^2 - 18\lambda}{(1+\lambda)^2} + 8 = 0$$

$$\frac{-9\lambda(\lambda+2)}{(1+\lambda)^2} + 8 = 0 \implies 8(1+\lambda)^2 = 9\lambda(\lambda+2)$$

$$\lambda^2 + 2\lambda - 8 = 0 \implies (\lambda+4)(\lambda-2) = 0$$

Since $\lambda \geq 0$, the dual optimal is $\lambda^\star = 2$.
So the dual optimal value is:

$$d^\star = g(2) = 5$$

Since $d^\star = p^\star = 5$, **strong duality holds**.

(d) Sensitivity analysis.
The constraint can be rewritten as

$$(x-2)(x-4) = x^2 - 6x + 8 \le u \quad \therefore \quad x^2 - 6x + (8-u) \le 0.$$

The quadratic in x attains nonpositive values if its discriminant is nonnegative, which is:

$$(-6)^2 - 4(1)(8-u) = 36 - 32 + 4u = 4(1+u).$$

Problem is feasible if $u \ge -1$, and the roots are $x = 3 \pm \sqrt{1+u}$, so the feasible set is $\{x \mid 3 - \sqrt{1+u} \le x \le 3 + \sqrt{1+u}\}$. For $-1 \le u \le 8$: the unconstrained minimizer $x = 0$ is not feasible, so the optimal point is $x^\star(u) = 3 - \sqrt{1+u}$, with optimal value $p^\star(u) = 1 + (3 - \sqrt{1+u})^2 = 11 + u - 6\sqrt{1+u}$. For $u \ge 8$ the constraint is inactive, $x^\star(u) = 0$, and $p^\star(u) = 1$.

$$p^\star(u) = \begin{cases} \infty, & u < -1, \\ 11 + u - 6\sqrt{1+u}, & -1 \le u \le 8, \\ 1, & u \ge 8. \end{cases}$$

At $u = 0$,

$$\frac{dp^\star}{du}\Big|_{u=0} = -2 = -\lambda^\star,$$

5.3 Problems with one inequality constraint

Lagrangian is $L(x, \lambda) = c^\top x + \lambda f(x)$.
Dual function is, for , $x \in \mathrm{dom} f_0(x)$

$$\begin{aligned} g(\lambda) &= \inf_x L(x, \lambda) = \inf_x (c^\top x + \lambda f(x)) \\ &= \lambda \inf_x \Big((c/\lambda)^\top x + f(x) \Big) \\ &= -\lambda \sup_x \Big((-c/\lambda)^\top x - f(x) \Big) \\ &= -\lambda f^*(-c/\lambda) \end{aligned}$$

where $f^*(y) = \sup_x (y^\top x - f(x))$ is the conjugate of f.
Dual Problem.

$$\begin{aligned} \text{maximize} \quad & -\lambda f^*(-c/\lambda) \\ \text{s.t.} \quad & \lambda \geq 0 \end{aligned}$$

By definition, the conjugate f^* is always convex. Then, the perspective function is defined as $g(x, t) = tf(x/t)$, with $t = -\eta, x = c$, and $f(\cdot) = f^*(\cdot)$, so the dual function is the perspective of f^*, and the perspective of a convex function is convex as well.

A4.17 Heuristic suboptimal solution for Boolean LP

The LP relaxation gives lower bound $L = -34.42$. Threshold rounding with $\hat{x}_i = \mathbf{1}(x_i^{\mathrm{rlx}} \geq t)$ yields feasible Boolean solutions for $t \geq 0.556$. The best upper bound $U = -33.58$ occurs at $t^* = 0.556$, with optimality gap $U - L = 0.84$ (only 2.4% of $|L|$). As t increases, fewer components are set to 1, reducing constraint violations but worsening the objective (since $c < 0$). The small gap indicates the heuristic finds a near-optimal solution.

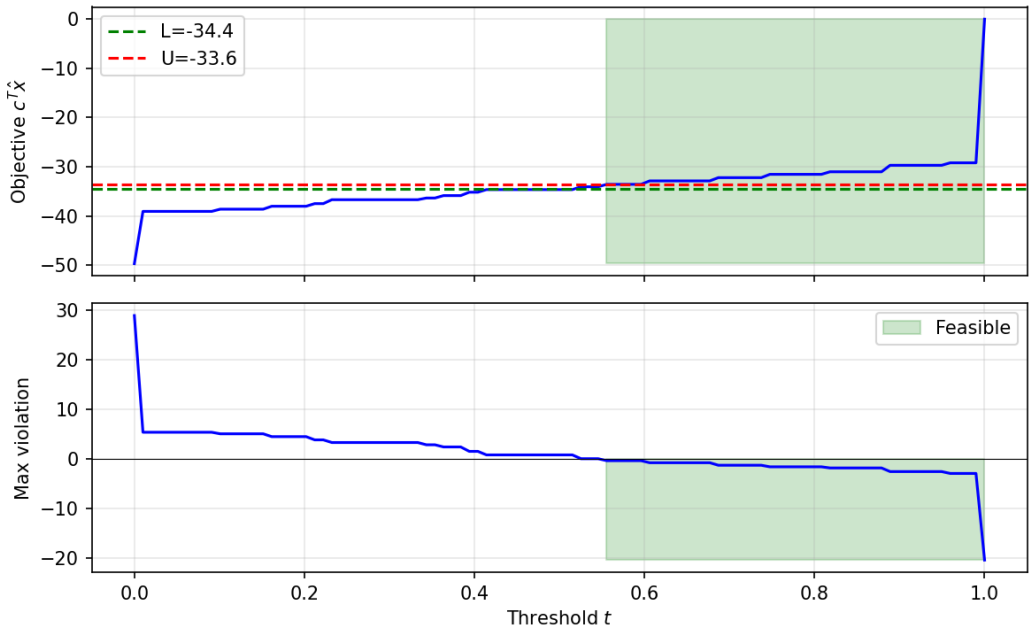

Figure 3: Objective value and max constraint violation

Code

```
# LP relaxation
x = cp.Variable(n)
prob = cp.Problem(cp.Minimize(c @ x), [A @ x <= b, 0 <= x, x <= 1])
prob.solve()
x_rlx, L = x.value, prob.value

# Threshold rounding
t_vals = np.linspace(0, 1, 100)
objs = np.array([c @ (x_rlx >= t) for t in t_vals])
viols = np.array([np.max(A @ (x_rlx >= t) - b) for t in t_vals])
feas = viols <= 1e-6

# Best feasible
best_idx = np.where(feas)[0][np.argmin(objs[feas])]
U, t_best = objs[best_idx], t_vals[best_idx]
print(f"L = {L:.2f}, U = {U:.2f}, gap = {U-L:.2f}, t* = {t_best:.2f}")
```

A5.1 Numerical perturbation analysis example

(a)

Solving QP

- Optimal primal variables $x_1^* = -2.333333$ $x_2^* = 0.166667$
- Optimal objectiva value $\lambda_1^* = 2.747741$ $\lambda_2^* = 2.885233$ 0.040072

- Optimal objective value $p^* = 8.222222$

Code:

```
x = cp.Variable(2)
x1, x2 = x[0], x[1]

# Objective rewriten as (1/2)*xˆT*P*x + qˆT*x:
P = np.array([[2, -1], [-1, 4]])
q = np.array([-1, 0])
objective = cp.Minimize(0.5 * cp.quad_form(x, P) + q @ x)

constraints = [
    x1 + 2*x2 <= u1,
    x1 - 4*x2 <= u2,
    5*x1 + 76*x2 <= 1
]

problem = cp.Problem(objective, constraints)
p_star = problem.solve()

lambda1 = constraints[0].dual_value
lambda2 = constraints[1].dual_value
lambda3 = constraints[2].dual_value
```

KKT verification:
KKT conditions hold, as seen in code output below:

```
1. Primal constraints (f_i <= 0):
   f1 = x1 + 2*x2 - u1 = 4.440892e-16
   f2 = x1 - 4*x2 - u2 = 0.000000e+00
   f3 = 5*x1 + 76*x2 - 1 = 5.329071e-15
2. Dual feasibility (lambda >= 0):
   lambda1 = 2.747741
   lambda2 = 2.885233
   lambda3 = 0.040072
3. Complementary slackness (lambda_i * f_i ˜0):
   lambda1 * f1 = 1.220242e-15
   lambda2 * f2 = 0.000000e+00
   lambda3 * f3 = 2.135451e-16
4. Gradient ˜0:
   dL/dx1 = 1.887379e-15
   dL/dx2 = 0.000000e+00
```

Code

```
f1 = x1 + 2*x2 - u1
f2 = x1 - 4*x2 - u2
f3 = 5*x1 + 76*x2 - 1
```

```
print(f"1. Primal constraints (f_i <= 0):")
print(f"   f1 = x1 + 2*x2 - u1 = {f1:.6e}")
print(f"   f2 = x1 - 4*x2 - u2 = {f2:.6e}")
print(f"   f3 = 5*x1 + 76*x2 - 1 = {f3:.6e}")

# 2. Dual constraints: lambda >= 0
print(f"2. Dual feasibility (lambda >= 0):")
print(f"   lambda1 = {l1:.6f}")
print(f"   lambda2 = {l2:.6f}")
print(f"   lambda3 = {l3:.6f}")

# 3. Complementary slackness: lambda_i * f_i = 0
cs1 = l1 * f1
cs2 = l2 * f2
cs3 = l3 * f3
print(f"3. Complementary slackness (lambda_i * f_i ~0):")
print(f"   lambda1 * f1 = {cs1:.6e}")
print(f"   lambda2 * f2 = {cs2:.6e}")
print(f"   lambda3 * f3 = {cs3:.6e}")

# 4. Stationarity: gradient of Lagrangian w.r.t x = 0
grad_x1 = 2*x1 - x2 - 1 + l1 + l2 + 5*l3
grad_x2 = 4*x2 - x1 + 2*l1 - 4*l2 + 76*l3
print(f"4. Gradient ~0:")
print(f"   dL/dx1 = {grad_x1:.6e}")
print(f"   dL/dx2 = {grad_x2:.6e}")
```

(b) Sensitivity analysis

Sensitivity analysis using $p^\star_{\text{pred}} = p^\star - \lambda_1^\star \delta_1 - \lambda_2^\star \delta_2$:

δ_1	δ_2	$p^\star_{\text{pred}}$	$p^\star_{\text{exact}}$
0.0	0.0	8.2222	8.2222
0.0	−0.1	8.5107	8.7064
0.0	0.1	7.9337	7.9800
−0.1	0.0	8.4970	8.5650
−0.1	−0.1	8.7855	8.8156
−0.1	0.1	8.2085	8.3189
0.1	0.0	7.9474	8.2222
0.1	−0.1	8.2360	8.7064
0.1	0.1	7.6589	7.7515

$p^*_{\text{pred}} \leq p^*_{\text{exact}}$ holds in every case.

Code

```
deltas = [-0.1, 0, 0.1]
for d1 in deltas:
    for d2 in deltas:
        u1_pert = u1_base + d1
```

```
u2_pert = u2_base + d2
sol_pert = solve_qp(u1_pert, u2_pert)
p_exact = sol_pert['p_star']
p_pred = sol_base['p_star'] - sol_base['lambda1']*d1 - sol_base['lambda2']*d2
inequality_holds = p_pred <= p_exact + 1e-6  # small tolerance
```

A5.3 Relative entropy

The Lagrangian is

$$L(x, \nu_1, \nu_2) = \sum_{k=1}^{n} x_k \log\left(\frac{x_k}{y_k}\right) + \nu_1^T(b - Ax) + \nu_2(1 - \mathbf{1}^T x).$$

Differentiating with respect to x_k gives

$$\frac{\partial L}{\partial x_k} = \log\left(\frac{x_k}{y_k}\right) + 1 - a_k^T\nu_1 - \nu_2 = 0,$$

so

$$x_k = y_k \exp(a_k^T\nu_1 + \nu_2 - 1).$$

Using $\sum_{k=1}^{n} x_k = 1$,

$$e^{\nu_2 - 1} = \frac{1}{\sum_{k=1}^{n} y_k e^{a_k^T\nu_1}}.$$

Substituting back yields the dual function

$$g(\nu_1) = b^T\nu_1 - \log\left(\sum_{k=1}^{n} y_k e^{a_k^T\nu_1}\right),$$

and the dual problem

$$\text{maximize} \quad b^T\nu_1 - \log\left(\sum_{k=1}^{n} y_k e^{a_k^T\nu_1}\right).$$

A5.36

1. If $\lambda^\star$ is large, then tightening the constraint (decreasing s) causes the optimal value to increase a lot.
2. If $\lambda^\star$ is large, then loosening the constraint (increasing s) causes the optimal value to decrease a lot.
3. If $\lambda^\star = 0$, then optimal value is unchanged.

A17.5 Simple portfolio optimization

(a) Minimum-risk portfolios with same expected return as uniform portfolio ($= 0.0729$):

Portfolio	Std Dev	Risk Reduction
Uniform ($x = \mathbf{1}/n$)	0.0666	–
No constraints	0.0090	86.4%
Long-only ($x \geq 0$)	0.0395	40.7%
Short limit ($\mathbf{1}^T x_- \leq 0.5$)	0.0150	77.5%

The unconstrained portfolio achieves the lowest risk by taking short positions. The long-only constraint is most restrictive. Allowing limited shorting (0.5) significantly reduces risk compared to long-only.

(b) Efficient frontier:

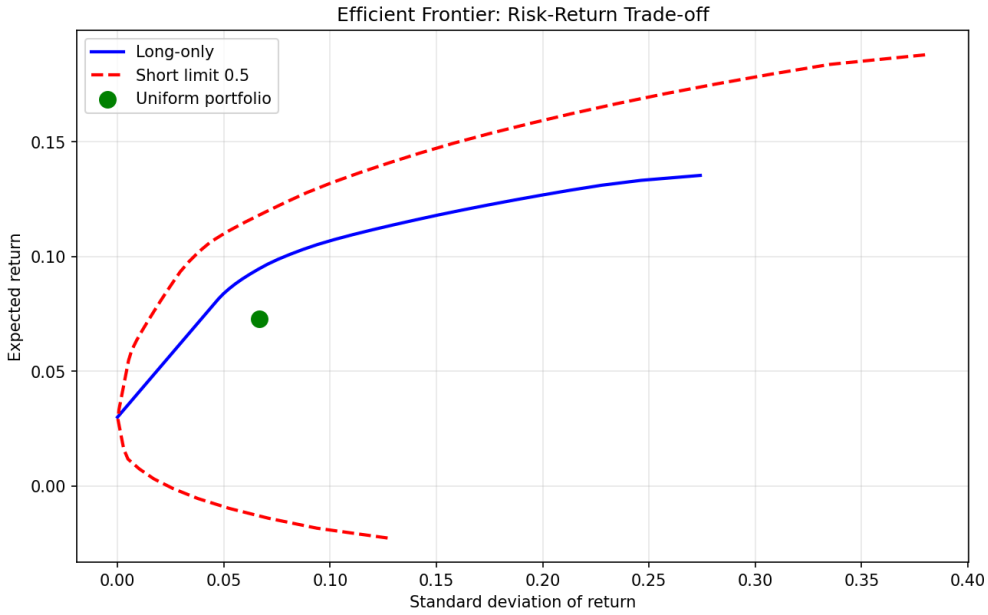

Figure 4: Risk-return trade-off curves for long-only and short-limited portfolios.

The short-limited portfolio dominates the long-only portfolio (lower risk for same return). Both curves pass through lower-risk regions than the uniform portfolio.

Code

```
# (a) Minimum-variance portfolio with target return
def min_var(r_target, long_only=False, short_limit=None):
    x = cp.Variable(n)
    cons = [cp.sum(x) == 1, pbar.flatten() @ x == r_target]
    if long_only: cons.append(x >= 0)
    if short_limit: cons.append(cp.sum(cp.neg(x)) <= short_limit)
    cp.Problem(cp.Minimize(cp.quad_form(x, S)), cons).solve()
    return x.value, np.sqrt(x.value @ S @ x.value)

_, std_long = min_var(r_unif, long_only=True)
_, std_short = min_var(r_unif, short_limit=0.5)
```

```
# (b) Efficient frontier
r_min, r_max = min(pbar), max(pbar)  # return range for long-only
returns = np.linspace(r_min, r_max, 50)
stds = [min_var(r, long_only=True)[1] for r in returns]
plt.plot(stds, returns)
```

A20.10 Energy storage trade-offs

(a) The optimal charging profile can be found by solving a linear program. The objective is to minimize total electricity cost $p^T(u+c)$, where $u+c$ is the net consumption from the grid. The constraints encode:

- Battery dynamics: $q_{t+1} = q_t + c_t$ (energy stored increases by charge amount)
- Cyclic operation: $q_1 = q_T + c_T$ (end with same charge as start)
- Capacity limits: $0 \le q_t \le Q$
- Rate limits: $-D \le c_t \le C$ (discharge/charge rates)
- No grid export: $u_t + c_t \ge 0$ (can't pump power back)

This is an LP in variables $c, q \in \mathbb{R}^T$:

$$\begin{array}{rl} \text{minimize} & p^T(u+c) \\ \text{subject to} & q_{t+1} = q_t + c_t,\ t = 1,\dots,T-1, \quad q_1 = q_T + c_T \\ & 0 \le q_t \le Q, \quad -D \le c_t \le C, \quad u_t + c_t \ge 0 \end{array}$$

(b) Results with $Q = 35$, $C = D = 3$:

Cost without battery	459.69
Cost with battery	379.41
Savings	80.28 (17.5%)

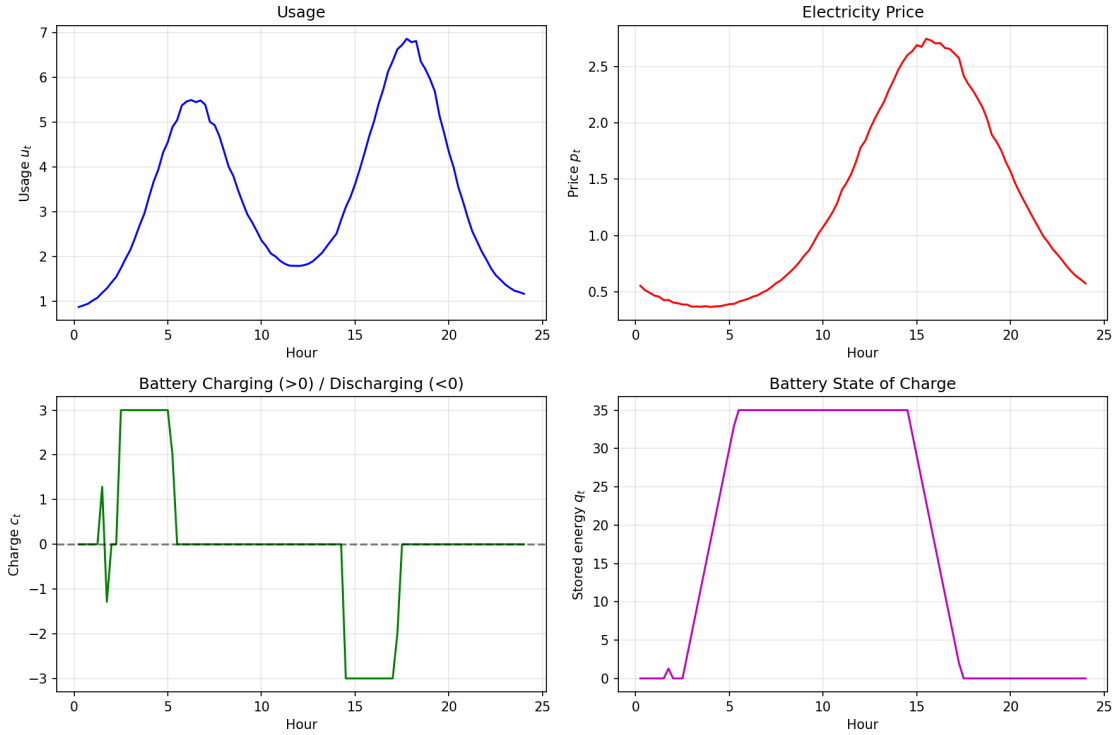

Figure 5: Usage, price, charging profile, and battery state of charge.

The battery charges when price is low and discharges when price is high.

(c) Trade-off curves:

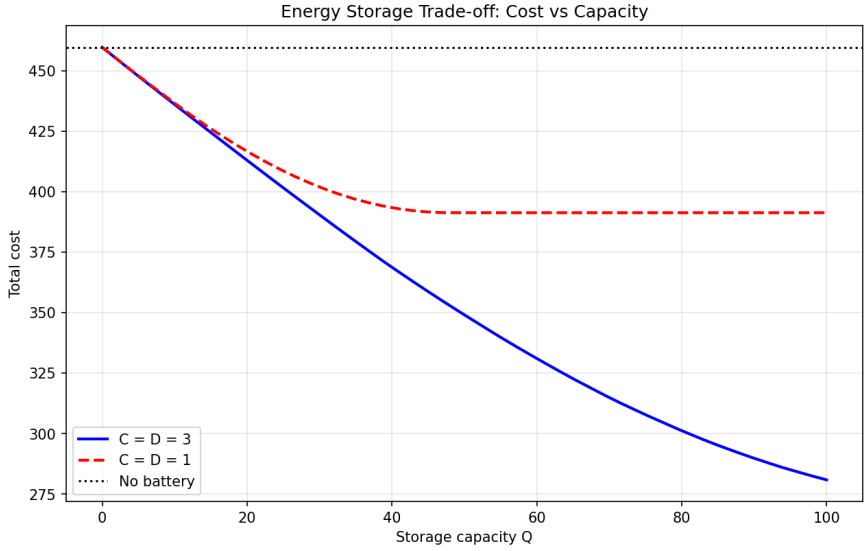

Figure 6: Cost vs storage capacity for different charge/discharge limits.

Interpretation:

- At $Q = 0$: No benefit, cost equals baseline (459.69)

- Higher C, D allows faster charge/discharge, achieving greater savings
- Diminishing returns as Q increases (curve flattens)

Code

```
def solve_storage(Q, C, D):
    c, q = cp.Variable(T), cp.Variable(T)
    cons = [
        q[1:] == q[:-1] + c[:-1],   # dynamics
        q[0] == q[-1] + c[-1],       # cyclic
        q >= 0, q <= Q,              # capacity
        c >= -D, c <= C,             # charge/discharge rate
        u.flatten() + c >= 0         # net consumption >= 0
    ]
    prob = cp.Problem(cp.Minimize(p.flatten() @ (u.flatten() + c)), cons)
    prob.solve()
    return prob.value

# (b) With Q=35, C=D=3
cost_with = solve_storage(35, 3, 3)
cost_without = float(p.T @ u)

# (c) Trade-off: vary Q
costs = [solve_storage(Q, 3, 3) for Q in np.linspace(0, 100, 50)]
```