Buying Gas Under Uncertainty

Kjell Raaijmakers (1244095), Jeroen van Riel (1236068)

July 20, 2022

Question a

Let D denote the demand, which is uniformly distributed, and let $F(x) = \mathbb{P}(D \leq x)$. Furthermore, we let $c = 0.5$ be the costs of the gas, we let $b = 1$ be the cost of the gas when purchased at a later date, and let $h = 0.1$ be the holding cost of the gas that is not used. Then this can be written as an inventory version of the newsvendor problem

$$\min_{x \geq 0} \mathbb{E}[G(x, D)],$$

with

$$G(x, d) := cx + b(d - x)^+ + h(x - d)^+.$$

From the lecture, we know that the solution for this problem is given by

$$x^* = F^{-1}\left(\frac{b - c}{b + h}\right) = F^{-1}\left(\frac{5}{11}\right)$$

Because we assume $D \sim U(38, 44)$, we have that

$$F(x) = \frac{x - 38}{44 - 38} = \frac{x - 38}{6} \qquad \text{for } x \in (38, 44),$$

which means that

$$F^{-1}(x) = 6x + 38,$$

so we get

$$x^* = F^{-1}\left(\frac{5}{11}\right) = 6 \cdot \frac{5}{11} + 38 \approx 40.7273.$$

Question b

The moments can simply be calculated from the distribution that we considered first. We find

$$\mu = \frac{38 + 44}{2} = 41$$

and

$$\sigma^2 = \frac{(44 - 38)^2}{12} = 3.$$

We now consider all probability distributions with $\mu = 41$ and $\sigma^2 = 3$. Given that we buy x, we define the profit as

$$\pi(x) = (p - c)\mu - (c - s)\mathbb{E}[(x - D)^+] - (p - c)\mathbb{E}[(D - x)^+].$$

The costs $b := p - c$ and $h := c - s$ may be interpreted as the cost of overage and shortage, respectively.

Now using that $(x-y)^+ - (y-x)^+ = x - y$ and $x - (x-y)^+ = min(x,y)$, we derive

$$\begin{aligned}\pi(x) &= (p-c)\mu - p\mathbb{E}[(D-x)^+)] + s\mathbb{E}[(x-D)^+] + c\mathbb{E}[(D-x)^+ - (x-D)^+]\\ &= p\mathbb{E}[D - (D-x)^+] + s\mathbb{E}[(x-D)^+] - cx\\ &= p\mathbb{E}[min(x,D)] + s\mathbb{E}[(x-D)^+] - cx.\end{aligned}$$

The objective given by

$$\max_x \pi(x)$$

subject to $x \geq 0$ and $\mathbb{E}[D] = \mu, Var(D) = \sigma^2$ has been solved in [1]. Note that the definitions in that paper ensure $m/d = (p-c)/(c-s) = b/h$, so we can use their equation (5) to obtain

$$\begin{aligned}x^* &= \mu + \frac{\sigma}{2}\left(\sqrt{\frac{b}{h}} - \sqrt{\frac{h}{b}}\right)\\ &= 41 + \frac{\sqrt{3}}{2}\left(\sqrt{\frac{1}{0.1}} - \sqrt{\frac{0.1}{1}}\right) \approx 43.4648.\end{aligned}$$

Question c

Let $h = 0.1$ be the holding (overage) cost, $b = 1$ the back-order (shortage) cost, let $c_1 = 0.5$ be the current price and let $c_2 = 0.75$ be the price in September, when we know that the demand is uniformly distributed on the interval $[A, B]$. Given x and bounds $\xi = (A, B)$, the second-stage problem may be formulated as

$$\min_{y \geq 0} c_2 y + \mathbb{E}_D[G(x,y,D)]$$

where the expectation is taken over $D \sim U(A,B)$ and

$$G(x,y,D) = b(D - (x+y))^+ + h((x+y) - D)^+.$$

The first-stage problem is

$$\min_{x \geq 0} c_1 x + \mathbb{E}_\xi[Q(x,\xi)],$$

where $Q(x,\xi)$ denotes the optimal value of the second-stage problem and the expectation is taken over $\xi = (A,B) = (\min(U_1,U_2), \max(U_1,U_2))$ with $U_i \sim U(38,44)$.

We now solve the second-stage optimization problem analytically (in terms of x and ξ) similarly as done in [2]. Start by rewriting $g(y) := \mathbb{E}[G(x,y,D)]$ as

$$g(y) = g(0) + \int_0^y g'(z)dz,$$

for $y \geq 0$. We have $g(0) = \mathbb{E}[b(D-x)^+ + h(x-D)^+]$. Furthermore, using

$$\frac{d}{dx}\mathbb{E}[(D-x)^+] = -\mathbb{P}(D \geq x)$$

and

$$\frac{d}{dx}\mathbb{E}[(x-D)^+] = \mathbb{P}(D \leq x),$$

we derive

$$\begin{aligned} g'(z) &= \frac{d}{dz}\mathbb{E}[b(D-(x+z))^+ + h((x+z)-D)^+] \\ &= -b\mathbb{P}(D \geq x+z) + h\mathbb{P}(D \leq x+z) \\ &= -b(1-\mathbb{P}(D \leq x+z)) + h\mathbb{P}(D \leq x+z) \\ &= -b + (b+h)F(x+z). \end{aligned}$$

Note that $\mathbb{E}[b(D-x)^+ + h(x-D)^+] = b\mathbb{E}D - bx + (b+h)\int_0^x F(z)dz$ using the same steps. Therefore, we end up with the objective

$$c_2 y + \mathbb{E}[G(x,y,D)] = b\mathbb{E}D - bx + (c_2-b)y + (b+h)\left(\int_0^x F(z)dz + \int_0^y F(x+z)dz\right).$$

Now setting the derivative with respect to y to zero yields $c_2 - b + (b+h)F(x+y) = 0$, hence the minimum is attained by

$$\bar{y} = F^{-1}(\kappa) - x,$$

where $\kappa := \frac{b-c_2}{b+h}$. From the cumulative distribution function

$$F(z) = \begin{cases} 0 \text{ for } z < A, \\ \frac{z-A}{B-A} \text{ for } A \leq z \leq B, \\ 1 \text{ for } z > B, \end{cases}$$

we find $\bar{y} = \lambda - x$, where $\lambda := A + \kappa(B-A)$. Furthermore, we calculate the integrals

$$\int_0^x F(z)dz = \frac{(x-A)^2}{2(B-A)}\mathbb{1}\{A \leq x \leq B\} + \left(x - \frac{A+B}{2}\right)\mathbb{1}\{x > B\}$$

and

$$\int_0^{\bar{y}} F(z+x)dz = \frac{(\lambda-A)^2}{2(B-A)}\mathbb{1}\{A \leq \lambda \leq B\} + \left(\lambda - \frac{A+B}{2}\right)\mathbb{1}\{\lambda > B\}.$$

Now assuming that the future back-order costs are higher ($b > c_2$), we have that $0 < \kappa < 1$ and hence $A \leq \lambda \leq B$, so this last expression further simplifies to

$$\int_0^{\bar{y}} F(z+x)dz = (B-A)\kappa^2/2$$

We are now ready to express $Q(x,\xi)$ in terms of x, A and B:

$$\begin{aligned} Q(x,\xi) &= c_2\bar{y} + \mathbb{E}[G(x,\bar{y},D)] \\ &= C - c_2 x + (b+h)\left(\frac{(x-A)^2}{2(B-A)}\mathbb{1}\{A \leq x \leq B\} + \left(x - \frac{A+B}{2}\right)\mathbb{1}\{x > B\}\right) \end{aligned}$$

where $C := b\left(\frac{A+B}{2}\right) + (c_2-b)\lambda + (b+h)(B-A)\kappa^2/2$. For concrete values $A = 38$ and $B = 44$, Figure 1 shows the shape of the cost function.

The first-stage problem can be solved by minimizing corresponding *sample average approximation* objective

$$\min_{x \geq 0} c_1 x + \frac{1}{N}\sum_{j=1}^{N} Q(x,\xi_j),$$

where ξ is a random sample of size N. We can use a simple grid search on x to find a value close the the optimum. Figure 2 shows the approximate objectives for samples with $N \in \{100, 200, 300, 1000, 10000\}$. The corresponding optimal values of x and the actual expected costs can be found in Table 1. As the figure shows, the approximated cost functions are rather close to each other.

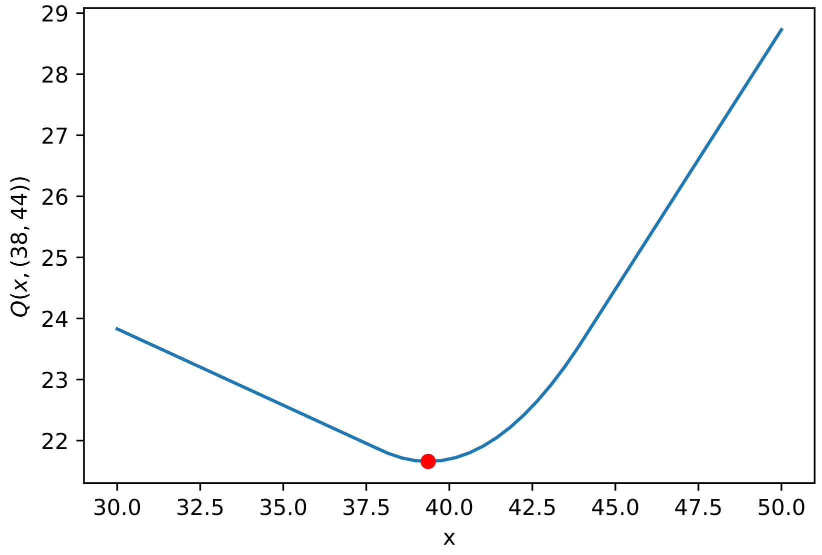


Figure 1: Shape of the cost function $Q(x, \xi)$ for specific values $A = 38$ and $B = 44$ with the minimum attained by $x \approx 39.3636$.

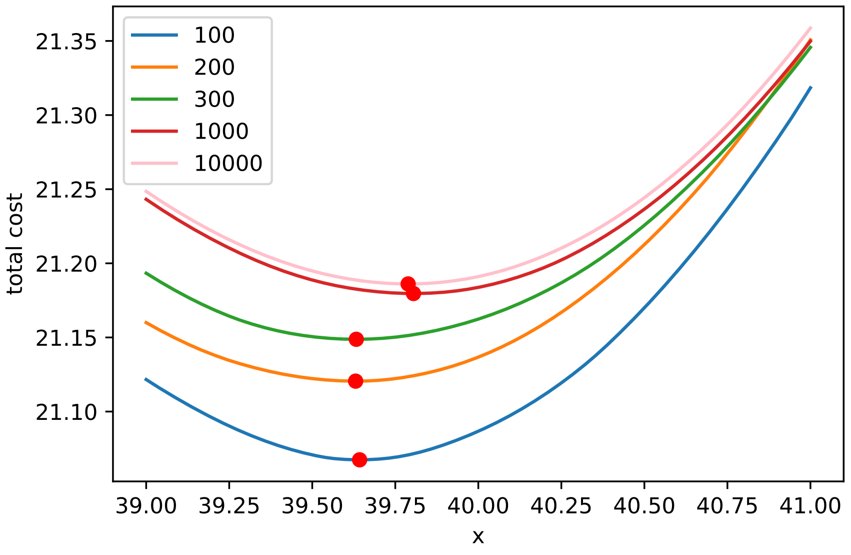


Figure 2: Approximate cost functions for samples with $N \in \{100, 200, 300, 1000, 10000\}$ with the minima found using 1000 values of x between between 38 and 44.

Table 1: Optimal value x and costs for various number of samples N.

N	x^*	cost
100	39.6426	21.0675
200	39.6306	21.1205
300	39.6326	21.1488
1000	39.8048	21.1796
10000	39.7888	21.1861

Analytical solution

We note that it would perhaps be possible to find a closed-form expression for $\mathbb{E}[Q(x,\xi)]$ by first determining the distribution of $\xi = (A, B)$ analytically. We already calculated that for $U_i \sim U(L, H)$

$$\mathbb{P}(A \leq z) = 1 - \left(\frac{H-z}{H-L}\right)^2$$

and

$$\mathbb{P}(B \leq z) = \left(\frac{z-L}{H-L}\right)^2,$$

with expected values $\mathbb{E}A = 2L/3 + H/3$ and $\mathbb{E}B = L/3 + 2H/3$.

Using

$$\frac{(x-A)^2}{2(B-A)} = \left(\frac{1}{2(B-A)}\right)x^2 + \left(\frac{A}{A-B}\right)x + \frac{A^2}{2(B-A)},$$

we can collect the coefficients of x such that

$$Q(x,\xi) = -c_2 x + C$$

for $x < A$,

$$Q(x,\xi) = \left(\frac{b+h}{2(B-A)}\right)x^2 + \left(\frac{A(b+h)}{A-B} - c_2\right)x + \frac{A^2(b+h)}{2(B-A)} + C$$

for $x \in [A, B]$ and

$$Q(x,\xi) = (b+h-c_2)\,x - (b+h)(A+B)/2 + C$$

for $x > B$.

It is possible to calculate the distribution of these coefficients. However, complications may arise from the fact that the function is defined piece-wise, because the inflection points are A and B and thus random. We did not verify this, but it seems to us that we may express the expectation as

$$\mathbb{E}[Q(x,\xi)] = f_l(x)\mathbb{P}(x < A) + f_m(x)\mathbb{P}(A \leq x \leq B) + f_r(x)\mathbb{P}(x > B),$$

where f_l, f_m, f_r are the expectation versions of the three pieces, so for example

$$f_m(x) = \mathbb{E}\left[\frac{1}{2(B-A)}\right]x^2 + \mathbb{E}\left[\frac{A}{A-B}\right]x + \frac{A^2}{2(B-A)}.$$

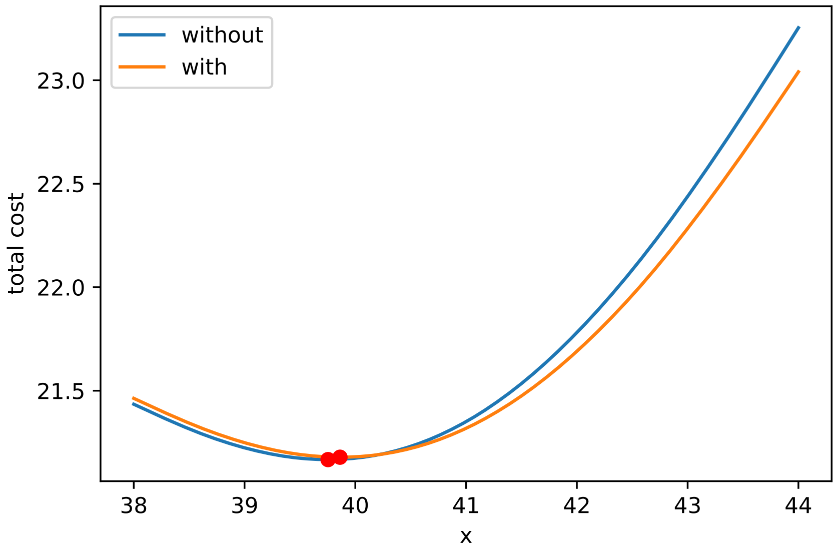


Figure 3: The cost curves for the problem with and without a storage unit of size 3.

Question d

We can use the storage unit to reduce possible storage costs in case over overage, so when the demand is less than expected. The second-stage problem changes because we now have

$$G(x,y,D) = b(D-(x+y))^+ + h((x+y)-D-3)^+.$$

Going through the same steps as in (c), we obtain the objective

$$c_2y + \mathbb{E}[G(x,y,D)] = b\mathbb{E}D - bx + (c_2-b)y + b\left(\int_0^x F(z)dz + \int_0^y F(x+z)dz\right) + h\left(\int_0^x F(z-3)dz + \int_0^y F(x+z-3)dz\right).$$

Similarly, we can find the value of y for which the minimum is attained by computing the derivative with respect to y and setting it to zero. This yields $c_2 - b + bF(x+y) + h(F(x+y-3) = 0$, from which we find that

$$\bar{y} = \frac{3h}{b+h} + A + \kappa(B-A) - x.$$

Now this determines $Q(x,\xi)$, so we can again use the sample average approximation method to solve the problem and to find the expected cost in the new problem setting to see what the expected advantage is.

We implemented this method, but we found that the cost was actually higher, see Figure 3. We probably made a mistake in the calculations, because this is certainly not what we would expect.

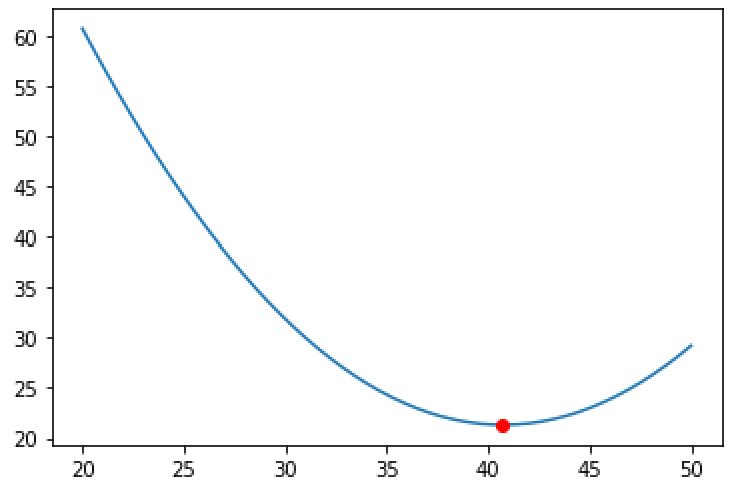
In [ ]:
```
import cvxpy as cp
import numpy as np
import matplotlib.pyplot as plt
```

In [ ]:
```
c1 = 0.5 # now
c2 = 0.75 # in september
b = 1 # after september
h = 0.1
kappa = (b-c2)/(b+h)
```

Solving a single-stage problem (exercise a)

In [ ]:
```
c = c1 # let's consider the situation in exercises (a) and (b)

def cost(y, A, B):
    return b * (A+B) / 2 + (c-b) * y - (b+h) * (A-y)*(A-y) / (2*(A-B))

# formula (1.6) in Shapiro's tutorial
def solve_second_exact(A, B):
    k = (b - c) / (b + h)
    y = A * (1-k) + B * k
    return y, cost(y, A, B)

# problem (2.5) in Shapiro's tutorial
def solve_second_MC(A, B, K):
    scenarios = np.random.uniform(low=A, high=B, size=K)

    y = cp.Variable(1)
    t = cp.Variable(K)
    objective = cp.Minimize(cp.sum(t) / K)
    constraints = [y >= 0]

    for i, dk in enumerate(scenarios):
        constraints.append((c-b) * y - t[i] <= -b * dk)
        constraints.append((c+h) * y - t[i] <= h * dk)

    prob = cp.Problem(objective, constraints)
    prob.solve()

    candidate = y.value[0]
    return candidate, prob.value
```

In [ ]:
```
x = np.linspace(20, 50)
plt.plot(x, cost(x, 38, 44))
y, cst = solve_second_exact(38, 44)
plt.plot(y, cst, 'ro');
```



In [ ]:
```
y1, cost1 = solve_second_exact(38, 44)
y2, cost2 = solve_second_MC(38, 44, 1000)
print(f'exact solution: {y1}, with cost: {cost1}')
print(f'MC solution: {y2}, with cost: {cost2}')
```

```
exact solution: 40.72727272727273, with cost: 21.318181818181817
MC solution: 40.71708562750193, with cost: 21.30594208765749
```

Solving the two-stage version (exercise c)

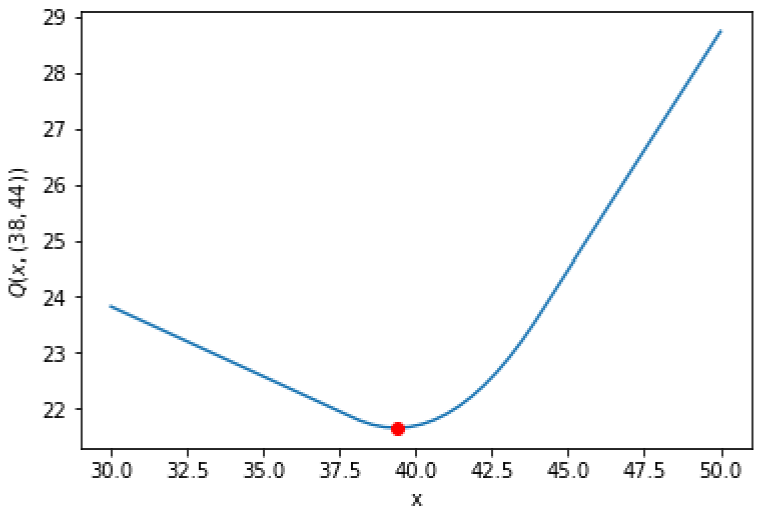
Monte Carlo for first stage only

We first express $Q(x, \xi)$ in terms of x, A, B.

In [ ]:
```
# for A <= x <= B
def d2(A, B):
    return (b+h) / (2*(B-A))
def d1(A, B):
    return (A/(A-B)) * (b+h) -c2 + c1
def d0(A, B):
    return ((b+h)*A*A) / (2 * (B-A))

# for x > B
Bd1 = b + h -c2 + c1
def Bd0(A, B):
    return (b+h) * ((B-A) / 2 - B)

# constant term in Q
def C(A, B):
    k = (b-c2)/(b+h)
    l = A + k*(B-A)
    return b*(A+B)/2 + (c2 - b) * l + (b+h)*(B - A)*k*k/2

# we define the cost function piecewise
def Q(x, A, B):
    if A <= x <= B:
        return d2(A, B) * x*x + d1(A, B) * x + d0(A, B) + C(A,B)
    elif x > B:
        return Bd1 * x + Bd0(A, B) + C(A,B)
    else:
        return (-c2 + c1) *x + C(A,B)
```

In [ ]:
```
# solve an example for given A, B
A = 38
B = 44
xs = np.linspace(30, 50)
y = [Q(x, A, B) for x in xs]

def x_opt(d1, d2):
    return -d1/(2*d2) # extreme point of parabola

s = x_opt(d1(A, B), d2(A, B))

plt.plot(xs, y)
plt.plot(s, Q(s, A, B), 'ro')
plt.xlabel('x'); plt.ylabel(r'$Q(x,(38,44))$')
plt.savefig('example_Q.pdf')
print(f'optimal x: {s}')
```

```
optimal x: 39.36363636363636
```

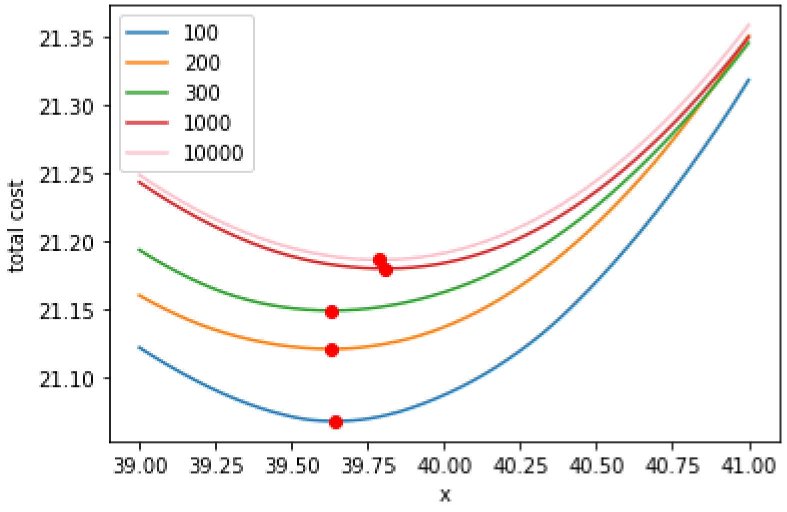


We can now perform a straightforward grid search on x to find a value close to the optimum.

In [ ]:
```
def get_bounds(N, low=38, high=44):
    # samples ksi_i = (A_i, B_i) = (min(U_1, U_2), max(U_1, U_2)), where U_i ~
    res = np.sort(np.random.uniform(low=low, high=high, size=(N,2)), axis=1)
    return res

def get_Q(xs, Q, bounds):
    As, Bs = bounds[:,0], bounds[:,1]
    def objective(x):
        res = 0
        for A, B in zip(As, Bs):
            res += Q(x, A, B)
        return res / len(bounds)

    return [objective(x) for x in xs]
```

In [ ]:
```
def find_solution(N):
    xs =  np.linspace(39, 41, num=1000)
    ys = get_Q(xs, Q, get_bounds(N))
    args = {'color': 'pink'} if N > 1000 else {} # default color is hard to di
    plt.plot(xs, ys, label=N, **args)
    s = xs[np.argmin(ys)]
    cst = np.min(ys)
    plt.plot(s, cst, 'ro')
    plt.xlabel('x')
    plt.ylabel('total cost')
    print(f'solution x={s:.4f}, with cost={cst:.4f}')

find_solution(100)
find_solution(200)
find_solution(300)
find_solution(1000)
find_solution(10000)
plt.legend()
plt.savefig('MC_grid.pdf')
```

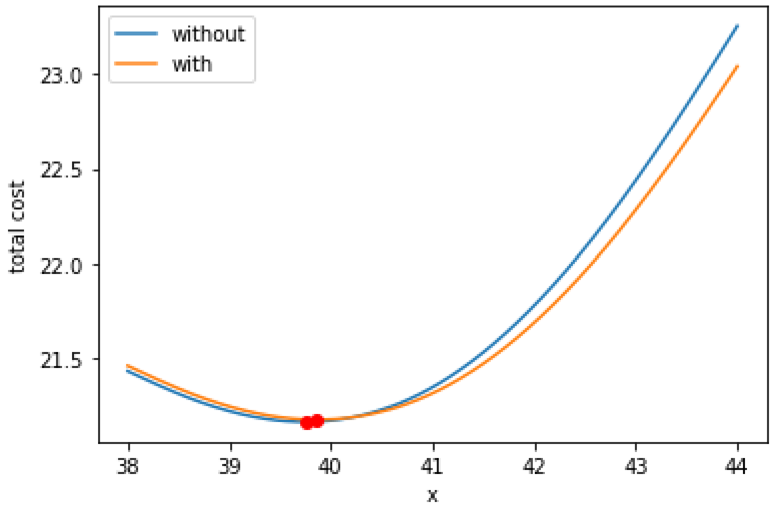
```
solution x=39.6426, with cost=21.0675
solution x=39.6306, with cost=21.1205
solution x=39.6326, with cost=21.1488
solution x=39.8048, with cost=21.1796
solution x=39.7888, with cost=21.1861
```



Value of storage unit (exercise d)

In [ ]:
```
def integral(y, x, A, B):
    if A <= x+y <= B:
        return (x+y-A)*(x+y-A) / (2*(B-A))
    elif x+y > B:
        return x+y - (A+B)/2
    else:
        return 0

def Q1(x, A, B):
    y_hat = A + kappa*(B-A) - x

    return c1*x + b*(A+B)/2 - b*x + (c2 - b) * y_hat \
        + (b+h)*(integral(x,0,A,B) + integral(y_hat,x,A,B))

def Q2(x, A, B, s=3):
    y_hat = (s*h)/(b+h) + A + kappa*(B-A) - x
    return c1*x + b*(A+B)/2 - b*x + (c2 - b) * y_hat \
        + b*(integral(x,0,A,B) + integral(y_hat,x,A,B)) \
        + h*(integral(x,-s,A,B) + integral(y_hat,x-s,A,B))

N = 1000
xs =  np.linspace(38, 44, num=1000)
bounds = get_bounds(N)
y1 = get_Q(xs, Q1, bounds)
y2 = get_Q(xs, Q2, bounds)
plt.plot(xs, y1, label='without')
plt.plot(xs, y2, label='with')
s1 = xs[np.argmin(y1)]
cst1 = np.min(y1)
s2 = xs[np.argmin(y2)]
cst2 = np.min(y2)
plt.plot(s1, cst1, 'ro')
plt.plot(s2, cst2, 'ro')
plt.xlabel('x')
plt.ylabel('total cost')
plt.legend()
plt.savefig('storage.pdf')

print(s1, s2)
print(f'cost difference = {cst1 - cst2}')
```

```
39.753753753753756 39.86186186186186
cost difference = -0.011519352352703294
```



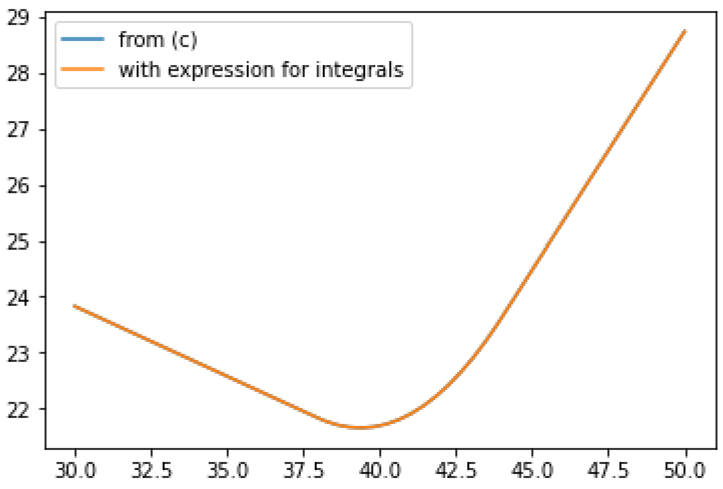
There is probably something wrong with our calculations, because we expect the cost with the storage unit to be lower.

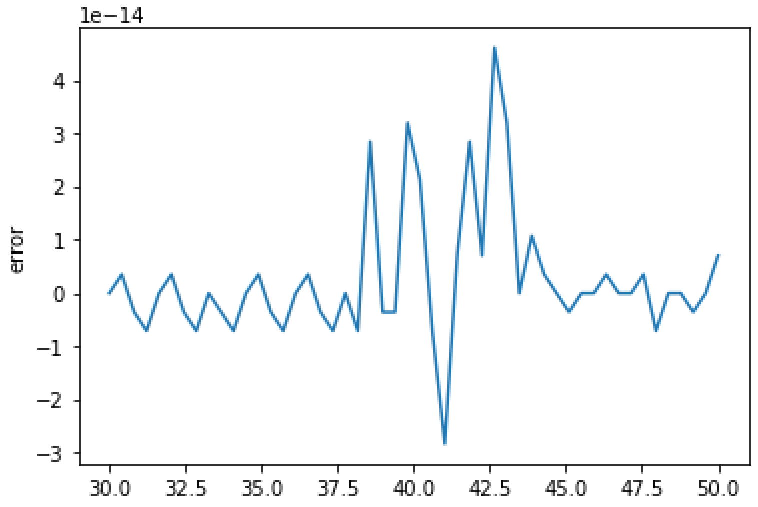
Sanity check

To check whether both methods actually compute the same values. Initially, we did not use the "integral" utility function and we wanted to split all the coefficients, because we anticipated on analytically deriving their expected value. However, we can greatly reduce the complexity of the expressions by simply using this "integral" utility function as shown below.

```
In [ ]: A=38
        B=44
```

```python
xs = np.linspace(30, 50)

# cost
cst1 = [Q(x, A, B) for x in xs]
cst2 = [Q1(x, A, B) for x in xs]
plt.plot(xs, cst1, label='from (c)')
plt.plot(xs, cst2, label='with expression for integrals')
plt.legend()
plt.show()
plt.plot(xs, np.array(cst1) - np.array(cst2))
plt.ylabel('error');
```


from (c)
with expression for integrals


1e−14
error