

Robust

Aayush Bajaj

July 10, 2026

Contents

1	the stance	1
2	uncertainty sets	2
3	robust counterparts	3
3.1	box: soyster's counterpart	3
3.2	ellipsoids and second-order cones	3
4	the budget of uncertainty	4
4.1	the protection function	4
4.2	dualising the inner problem	4
4.3	the price of robustness	5
5	robust vs stochastic	5
6	code	6
7	see also	7

every **linear program** you have ever written down was a lie: the coefficients came from measurements, forecasts and vendor spreadsheets, and the optimal vertex — sitting, by design, on the boundary of the feasible region — shatters the moment any of them wobbles. robust optimisation is the pessimist's response: declare a set $\mathcal{U}$ of realisations you refuse to be hurt by, and demand feasibility for *every* member of it. no distributions, no expectations, no scenarios — just a set and a worst case. the surprise, and the reason the field exists, is that this worst case can usually be folded back into a deterministic problem of the same (or nearly the same) complexity class.¹

an optimal basic solution binds n constraints with zero slack; it is maximally exposed to data error by construction

1 the stance

an uncertain linear constraint is a semi-infinite one:

$$a^\top x \leq b \quad \forall a \in \mathcal{U} \quad \Longleftrightarrow \quad \sup_{a \in \mathcal{U}} a^\top x \leq b. \quad (1)$$

¹ben-tal and nemirovski (1998), *robust convex optimization*, mathematics of operations research 23(4); the book-length treatment — including adjustable RO and the tractability map from uncertainty-set geometry to counterpart class — is ben-tal, el ghaoui and nemirovski (2009), *robust optimization*, princeton university press.

read it as a game: you commit to x first, then an adversary picks the most damaging $a \in \mathcal{U}$. a robust-feasible x survives the adversary's best move; the **robust counterpart** (RC) of the problem is what you get after evaluating the inner sup in closed form. three structural facts keep the machinery manageable:

- **constraint-wise decomposition.** the adversary attacks each constraint independently — coupling uncertainty across rows never helps it, so it suffices to robustify one row at a time against the projection of $\mathcal{U}$ onto that row's coefficients.¹
- **uncertain objective.** fold $\max_x \min_{c \in \mathcal{U}_c} c^\top x$ into the constraints via the epigraph: maximise t subject to $t \leq c^\top x$ for all $c \in \mathcal{U}_c$ — an uncertain constraint like any other.
- **uncertain right-hand side.** treat b as the coefficient of a phantom variable pinned at -1 ; nothing new is needed. equality constraints, by contrast, are essentially unrobustifiable — demanding $a^\top x = b$ for a continuum of a forces x toward the trivial kernel, which is why robust models are written with inequalities.

the entire character of the resulting problem — LP, SOCP, SDP, or intractable — is decided by the geometry of $\mathcal{U}$. choosing the set *is* the modelling act.

2 uncertainty sets

write the uncertain row as $a = \bar{a} + \delta$, with $\bar{a}$ the nominal value and δ the deviation. the three workhorse sets, in deviation coordinates scaled so that $\hat{a}_j$ is the largest credible deviation of coefficient j :

- **box** (interval): $|\delta_j| \leq \hat{a}_j$ for every j . buys you: an LP counterpart and an iron guarantee. costs you: protection against the corner realisation where all n coefficients conspire to their extremes simultaneously — an event so improbable under any reasonable noise model that hedging it is pure waste.
- **ellipsoidal:** $\delta = Pu$, $\|u\|_2 \leq \Omega$. buys you: corners rounded off, a clean chance-constraint interpretation under gaussian noise, and control of correlated deviations through P . costs you: the counterpart is a second-order cone program, not an LP.¹
- **budget** (polyhedral, bertsimas-sim): the box, intersected with $\sum_j |\delta_j|/\hat{a}_j \leq \Gamma$ — at most Γ coefficients may deviate fully.² buys you: the corner-shaving of the ellipsoid *while staying polyhedral*, so the counterpart is again an LP, and the single dial Γ sweeps from nominal recklessness to box paranoia.

²bertsimas and sim (2004), *the price of robustness*, operations research 52(1), which also covers the discrete case and the probability-of-violation bounds.

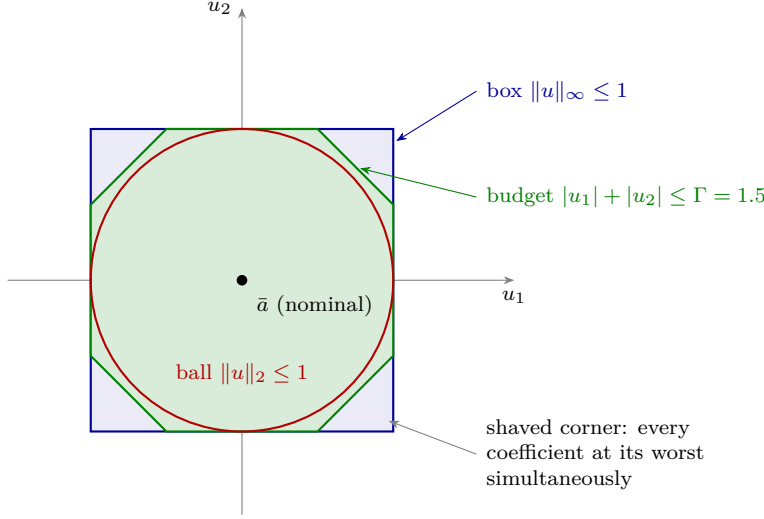


Figure 1: the three workhorse uncertainty sets in deviation space, $n = 2$, deviations scaled to $[-1, 1]$: the box, the inscribed ball, and the budget polytope with $\Gamma = 1.5$, which keeps the box's faces but shaves off exactly the conspiratorial corners.

3 robust counterparts

3.1 box: soyster's counterpart

take one constraint $\sum_j a_j x_j \leq b$ with each $a_j \in [\bar{a}_j - \hat{a}_j, \bar{a}_j + \hat{a}_j]$ independently. the adversary maximises term by term — deviation j simply matches the sign of x_j :

$$\sup_{a \in \mathcal{U}_{\text{box}}} \sum_j a_j x_j = \sum_j \bar{a}_j x_j + \sum_j \max_{|\delta_j| \leq \hat{a}_j} \delta_j x_j = \sum_j \bar{a}_j x_j + \sum_j \hat{a}_j |x_j|, \quad (2)$$

so the robust constraint is $\sum_j \bar{a}_j x_j + \sum_j \hat{a}_j |x_j| \leq b$. the absolute values break linearity; restore it with auxiliary variables $u_j \geq |x_j|$:

$$\begin{aligned} \sum_j \bar{a}_j x_j + \sum_j \hat{a}_j u_j &\leq b \\ -u_j &\leq x_j \leq u_j \quad \forall j. \end{aligned}$$

the two forms are equivalent: given robust-feasible x , set $u_j = |x_j|$; conversely any feasible (x, u) has $u_j \geq |x_j|$, so its left-hand side only overestimates the true worst case. this is soyster's 1973 model³ — the earliest robust counterpart, an LP mapped to an LP, and for two decades also the argument *against* robustness: hedging the full box is so conservative that the objective often degrades more than anyone will accept.

when the model already forces x nonnegative the machinery collapses: the counterpart is just the pessimistic LP with coefficients $\bar{a}_j$ plus $\hat{a}_j$

3.2 ellipsoids and second-order cones

for $\mathcal{U}_{\text{ell}} = \{\bar{a} + Pu : \|u\|_2 \leq \Omega\}$, cauchy–schwarz evaluates the inner supremum exactly — the adversary aligns u with $P^\top x$ — and the robust constraint becomes

³soyster (1973), *convex programming with set-inclusive constraints and applications to inexact linear programming*, operations research 21(5).

$$\bar{a}^\top x + \Omega \|P^\top x\|_2 \leq b, \quad (3)$$

a second-order cone constraint: the RC of an LP with ellipsoidal uncertainty is an SOCP, still convex and solved by interior-point methods in polynomial time.¹ the parameter Ω is a safety dial with a probabilistic reading: if a is gaussian with mean $\bar{a}$ and covariance $\Sigma = PP^\top$, the constraint above with $\Omega = \Phi^{-1}(1 - \epsilon)$ is *exactly* the chance constraint $\Pr(a^\top x \leq b) \geq 1 - \epsilon$ — robust optimisation quietly solving a stochastic problem without ever sampling.⁴

the same mechanism runs backwards too: robustifying least squares against norm-bounded matrix error recovers ridge regularisation — robustness and regularisation are the same trick

4 the budget of uncertainty

bertsimas and sim’s observation:² the box is conservative precisely because it lets *all* n coefficients deviate at once, yet in any plausible world only a few things go wrong together. so budget the adversary. with scaled deviations $z_j = |\delta_j|/\hat{a}_j \in [0, 1]$, allow

$$\mathcal{U}_\Gamma = \left\{ \bar{a} + \delta : |\delta_j| \leq \hat{a}_j, \sum_j |\delta_j|/\hat{a}_j \leq \Gamma \right\}, \quad (4)$$

i.e. at most Γ coefficients deviate fully ($\lfloor \Gamma \rfloor$ of them all the way, one more by the fraction $\Gamma - \lfloor \Gamma \rfloor$). $\Gamma = 0$ is the nominal problem; $\Gamma = n$ recovers the full box.

4.1 the protection function

the worst case splits into the nominal part plus a **protection function** — the adversary’s inner maximisation:

$$\beta(x, \Gamma) = \max \left\{ \sum_j \hat{a}_j |x_j| z_j : \sum_j z_j \leq \Gamma, 0 \leq z_j \leq 1 \right\}, \quad (5)$$

and the robust constraint is $\sum_j \bar{a}_j x_j + \beta(x, \Gamma) \leq b$. the continuous relaxation over z is legitimate: the polytope $\{0 \leq z \leq 1, \sum_j z_j \leq \Gamma\}$ has vertices with at most one fractional coordinate, so the LP optimum coincides with the combinatorial “pick the $\lfloor \Gamma \rfloor$ worst coefficients, plus a fraction of the next” adversary.

4.2 dualising the inner problem

$\beta(x, \Gamma)$ is itself a maximisation sitting inside a $\leq$ constraint — unusable as written. but for fixed x it is an LP, feasible ($z = 0$) and bounded, so strong LP duality applies. attach dual variable $z \geq 0$ to the budget row and $p_j \geq 0$ to each cap $z_j \leq 1$:

$$\beta(x, \Gamma) = \min \left\{ \Gamma z + \sum_j p_j : z + p_j \geq \hat{a}_j |x_j| \ \forall j, \ z \geq 0, \ p_j \geq 0 \right\}. \quad (6)$$

the general principle: an inner max inside a less-than constraint becomes an inner min by duality, and a min may be replaced by any feasible dual point — existential, so the dual variables simply join the outer problem

substituting the dual and linearising $|x_j|$ with y_j gives the full bertsimas–sim counterpart of $\max c^\top x$ subject to uncertain rows i (uncertain coefficients $j \in J_i$, budget Γ_i per row):

⁴el ghaoui and lebet (1997), *robust solutions to least-squares problems with uncertain data*, siam journal on matrix analysis and applications 18(4).

$$\begin{aligned}
& \text{maximise} && c^\top x \\
& \text{subject to} && \sum_j \bar{a}_{ij} x_j + \Gamma_i z_i + \sum_{j \in J_i} p_{ij} \leq b_i \quad \forall i \\
& && z_i + p_{ij} \geq \hat{a}_{ij} y_j \quad \forall i, j \in J_i \\
& && -y_j \leq x_j \leq y_j \quad \forall j \\
& && z_i \geq 0, \quad p_{ij} \geq 0, \quad y_j \geq 0.
\end{aligned}$$

still an LP — $m + n$ extra variable groups and $O(mn)$ extra rows, nothing a modern solver notices. the same dualisation template works for any polyhedral $\mathcal{U}$, and the trick extends unchanged to integer programs: robustifying costs only continuous variables, so a MIP stays a MIP with the same discrete difficulty.²

4.3 the price of robustness

Γ is a conservatism dial, and turning it has two monotone effects: the objective degrades (the **price of robustness**) and the violation probability falls. the exchange rate is favourably nonlinear — the first units of Γ buy the most protection, because deflecting the one or two most damaging deviations is where the action is; the last units insure the corner conspiracies that essentially never happen. bertsimas and sim make the second effect precise: if the true deviations are independent and symmetrically distributed on $[-\hat{a}_j, \hat{a}_j]$, the robust solution’s violation probability admits bounds of the form $\exp(-\Gamma^2/2n)$, with a sharper normal-tail approximation driven by $(\Gamma - 1)/\sqrt{n}$.² the reading that matters: a budget of order $\sqrt{n}$ — not n — already drives violation probability toward zero, so full-box protection overpays by roughly a square root.

5 robust vs stochastic

stochastic programming is the other mature answer to data uncertainty, and the contrast is philosophical before it is technical:

axis	robust optimisation	stochastic programming
uncertainty	a set $\mathcal{U}$; no distribution	a probability distribution (known or estimated)
guarantee	feasible for <i>every</i> realisation in $\mathcal{U}$	good <i>expected</i> cost; feasibility often only probabilistic
data appetite	support / bounds only	the whole distribution, or enough samples to fake it
tractability	RC stays LP or SOCP for the standard sets	scenario trees grow exponentially in stages
recourse	classic RO decides everything here-and-now	two-stage recourse is native
failure mode	conservatism — paying for phantom worst cases	misspecification — optimising against the wrong law

- when constraint violation is catastrophic (structural engineering, aircraft scheduling, chemotherapy dosing) and distributions are guesswork, the robust guarantee is the honest one: it holds for realisations, not on average.
- when violations are merely costly and data are plentiful, the stochastic answer is less conservative — robust solutions deliberately sacrifice expected performance for a guarantee the application may not need.
- classic RO’s weakness is recourse: real decisions unfold in stages, and pretending everything is decided before uncertainty resolves overprices flexibility. *adjustable* robust optimisation patches this with decision rules (typically affine in the observed data), trading exactness for tractability.¹
- the modern midpoint is **distributionally robust optimisation**: worst case over a *set of distributions* (matching known moments, or within a wasserstein ball of the empirical law) — robust in shape, stochastic in substance, and the counterpart machinery is the same dualisation game played one level up.

6 code

the production LP from the experiment: two products, profits (25, 20); labour $4x_1 + 3x_2 \leq 480$ and material $2x_1 + 3x_2 \leq 360$, every technology coefficient trusted only to $\pm 10\%$. nominal optimum at the intersection of both constraints — zero slack anywhere, maximal fragility. the code builds the bertsimas-sim counterpart exactly as derived (variables x, y, z_i, p_{ij}), sweeps Γ , and monte-carlos the violation rate of each solution under uniform coefficient noise:

```

1 import numpy as np
2 from scipy.optimize import linprog
3
4 # production LP: two products, two resources, uncertain usage coefficients
5 profit = np.array([25.0, 20.0])           # dollars per unit
6 Abar = np.array([[4.0, 3.0],             # labour hours per unit
7                  [2.0, 3.0]])            # material kg per unit
8 Ahat = 0.10 * Abar                       # +-10% coefficient uncertainty
9 b = np.array([480.0, 360.0])
10 m, n = Abar.shape
11
12 def nominal():
13     r = linprog(-profit, A_ub=Abar, b_ub=b, bounds=[(0, None)] * n,
14                 method="highs")
15     return r.x, -r.fun
16
17 def bertsimas_sim(gamma):
18     """budget-of-uncertainty counterpart, one budget per constraint row.
19     variables: x (n), y (n) with y_j >= |x_j|, z (m), p (m*n)."""
20     nv = n + n + m + m * n
21     ix = lambda j: j
22     iy = lambda j: n + j
23     iz = lambda i: 2 * n + i
24     ip = lambda i, j: 2 * n + m + i * n + j
25     c = np.zeros(nv); c[:n] = -profit
26     A_ub, b_ub = [], []
27     for i in range(m):                     # abar_i x + Gamma z_i + sum_j p_ij
28         row = np.zeros(nv)
29         row[:n] = Abar[i]; row[iz(i)] = gamma
30         for j in range(n): row[ip(i, j)] = 1.0
31         A_ub.append(row); b_ub.append(b[i])
32         for j in range(n):                 # ahat_ij y_j - z_i - p_ij <= 0
33             row = np.zeros(nv)
34             row[iy(j)] = Ahat[i, j]; row[iz(i)] = -1.0; row[ip(i, j)] = -1.0
35             A_ub.append(row); b_ub.append(0.0)
36         for j in range(n):                 # x_j - y_j <= 0 and -x_j - y_j <= 0
37             row = np.zeros(nv); row[ix(j)] = 1.0; row[iy(j)] = -1.0
38             A_ub.append(row); b_ub.append(0.0)
39             row = np.zeros(nv); row[ix(j)] = -1.0; row[iy(j)] = -1.0
40             A_ub.append(row); b_ub.append(0.0)
41     r = linprog(c, A_ub=np.array(A_ub), b_ub=np.array(b_ub),
42                 bounds=[(0, None)] * nv, method="highs")
43     return r.x[:n], -r.fun
44
45 def violation_rate(x, trials=200_000, seed=0):
46     """monte carlo: every a_ij drawn uniform in [abar-ahat, abar+ahat]."""
47     rng = np.random.default_rng(seed)
48     A = Abar + Ahat * rng.uniform(-1, 1, size=(trials, m, n))
49     lhs = A @ x                                     # (trials, m)
50     return np.mean(np.any(lhs > b + 1e-9, axis=1))
51
52 x_nom, z_nom = nominal()
53 print(f"nominal optimum: x = ({x_nom[0]:.3f}, {x_nom[1]:.3f}), "
54       f"profit = {z_nom:.2f}")

```

```

55 print()
56 print(f"{'Gamma':>5} {'x1':>8} {'x2':>8} {'profit':>9} "
57       f"{'cost of robustness':>19} {'P(violate)':>11}")
58 for g in (0.0, 0.5, 1.0, 1.5, 2.0):
59     x, z = bertsimas_sim(g)
60     print(f"{g:>5.1f} {x[0]:>8.3f} {x[1]:>8.3f} {z:>9.2f} "
61           f"{100 * (z_nom - z) / z_nom:>18.2f}% {violation_rate(x):>10.4f}")
62
63 # sanity check: Gamma = n must equal the soyster / full-box counterpart
64 r = linprog(-profit, A_ub=Abar + Ahat, b_ub=b, bounds=[(0, None)] * n,
65             method="highs")
66 x_box, z_box = r.x, -r.fun
67 xg, zg = bertsimas_sim(2.0)
68 print()
69 print(f"box counterpart directly: x = ({x_box[0]:.3f}, {x_box[1]:.3f}), "
70       f"profit = {z_box:.2f} (matches Gamma = 2: {np.allclose(zg, z_box)})")

```

nominal optimum: $x = (60.000, 80.000)$, profit = 3100.00

Gamma	x1	x2	profit	cost of robustness	P(violate)
0.0	60.000	80.000	3100.00	0.00%	75.0665%
0.5	59.751	76.349	3020.75	2.56%	45.8090%
1.0	59.155	73.239	2943.66	5.04%	17.6200%
1.5	56.926	72.865	2880.46	7.08%	4.5460%
2.0	54.545	72.727	2818.18	9.09%	0.0000%

box counterpart directly: $x = (54.545, 72.727)$, profit = 2818.18 (matches Gamma = 2: True)

read the table as the price-of-robustness curve made flesh:

- the **nominal** plan is violated in 75% of sampled worlds — both constraints bind at the optimum, so any adverse wobble in either row breaks the plan. this is not bad luck; it is what “optimal vertex” means.
- the **full box** ($\Gamma = 2$, identical to the soyster counterpart, as the sanity check confirms) costs 9.09% of profit for a 0% violation rate — with $\pm 10\%$ coefficient noise the guarantee price is almost exactly the noise level, since inflating every coefficient by 10% deflates the feasible polytope by roughly the same factor.
- the dial between them is **concave in protection**: the first half-unit of Γ buys 29 points of violation probability for 2.6% of profit; the last half-unit buys the final 4.5 points for another 2%. a decision-maker who tolerates a 5% failure rate stops at $\Gamma \approx 1.5$ and pockets two points of profit relative to the paranoid plan — precisely the negotiation bertsimas and sim built the formulation to enable.

7 see also

- constraint programming** — branch-and-infer, the other answer to hard feasibility
- linear programming** — the nominal problem, and the duality machinery the counterparts run on
- stochastic programming** — the distributional rival: expectations and scenarios instead of sets
- quadratic programming** — next door to the SOCPs that ellipsoidal counterparts produce

- **integer programming** — the budget trick robustifies MIPs without adding discrete difficulty
- **multi-objective programming** — the price of robustness is a two-objective trade-off wearing a single dial