

Linear Programming

Aayush Bajaj

July 10, 2026

Contents

1	standard form	1
2	geometry	2
2.1	a worked example	2
3	duality	3
3.1	the dual of the worked example	4
4	the simplex method	4
4.1	tableau walkthrough	4
4.2	degeneracy	5
5	interior-point methods	5
6	applications	5
6.1	the diet problem	5
6.2	max-flow as LP	5
7	code	6
8	see also	7

“programming” here means *planning*, not coding — the word predates the software sense. a linear program optimises a linear objective over a region carved out by linear inequalities. it is the base camp of mathematical programming: **quadratic**, **integer** and **non-linear** programming all generalise it in one direction or another, and all of them lean on LP machinery (relaxations, duals, warm starts) to get anything done. clrs devotes chapter 29 to it (Cormen, Thomas H. and Leiserson, Charles E. and Rivest, Ronald L. and Stein, Clifford, 2009).

dantzig invented the simplex method in 1947 for air-force logistics planning; kantorovich had the theory in 1939 but the soviets sat on it

1 standard form

a linear program (LP) in *standard form*:

$$\begin{array}{ll}\text{maximise} & c^\top x \\ \text{subject to} & Ax \leq b \\ & x \geq 0,\end{array}$$

with $c, x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, $A \in \mathbb{R}^{m \times n}$. every LP converts to this shape by mechanical

rewrites:

- **minimise** $c^\top x$: maximise $-c^\top x$.
- **equality** $a^\top x = b_i$: the pair $a^\top x \leq b_i$ and $-a^\top x \leq -b_i$.
- **free variable** x_j : substitute $x_j = x_j^+ - x_j^-$ with $x_j^+, x_j^- \geq 0$.
- **$\geq$ constraint**: multiply through by -1 .

for the simplex method one more step gives *slack form*: each inequality $a_i^\top x \leq b_i$ becomes an equality $a_i^\top x + s_i = b_i$ with a slack variable $s_i \geq 0$. geometry unchanged, algebra friendlier.

an LP is exactly one of three things: **infeasible** (no x satisfies the constraints), **unbounded** (feasible points achieve arbitrarily large objective), or it has a finite **optimum**.

2 geometry

the feasible region $\{x : Ax \leq b, x \geq 0\}$ is an intersection of half-spaces — a convex **polyhedron** (a **polytope** when bounded). the objective $c^\top x$ is linear, so its level sets are parallel hyperplanes; optimising means sliding a hyperplane as far as it will go while still touching the region.

vertex optimality theorem. if an LP has an optimal solution and its feasible region has at least one vertex, then some vertex is optimal. sketch: take any optimal x^* . if x^* is not a vertex, some direction $d \neq 0$ keeps both $x^* \pm \epsilon d$ feasible; optimality forces $c^\top d = 0$ (otherwise one of the two directions improves), so we may walk along $\pm d$ at constant objective until an extra constraint becomes tight. repeat: each step raises the count of tight, linearly independent constraints, and after at most n steps we stand on a vertex with the same objective value.

a vertex — equivalently a basic feasible solution — is a feasible point where n linearly independent constraints hold with equality

the consequence is enormous: a continuous optimisation problem over uncountably many points reduces to a search over finitely many vertices — at most $\binom{n+m}{n}$ of them. the catch is that “finitely many” can be astronomically many, which is the whole story of LP algorithms.

2.1 a worked example

the LP used throughout this page:

this is the classic wyndor-glass-style two-variable factory problem: two products, three shared resources

$$\begin{array}{ll}\text{maximise} & 3x + 5y \\ \text{subject to} & x \leq 4 \\ & 2y \leq 12 \\ & 3x + 2y \leq 18 \\ & x, y \geq 0.\end{array}$$

the feasible region is a pentagon with vertices $(0, 0), (4, 0), (4, 3), (2, 6), (0, 6)$. evaluating $3x + 5y$ at each: 0, 12, 27, 36, 30. the optimum is $z^* = 36$ at $(2, 6)$.

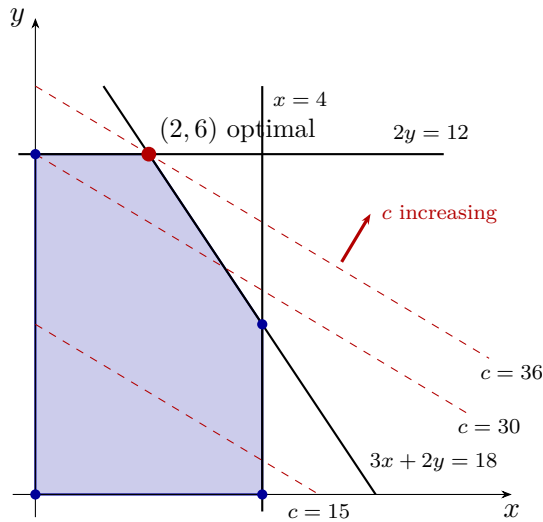


Figure 1: the feasible pentagon for the worked LP. dashed lines are objective isolines $3x+5y = c$; the objective improves toward the upper right and last touches the region at the vertex $(2, 6)$.

3 duality

every LP has a shadow. attach a multiplier $u_i \geq 0$ to each constraint and combine: any feasible x satisfies $u^\top Ax \leq u^\top b$. if the multipliers are chosen so that $A^\top u \geq c$ componentwise, then (using $x \geq 0$)

$$c^\top x \leq (A^\top u)^\top x = u^\top Ax \leq u^\top b = b^\top u. \quad (1)$$

so *every* such u certifies an upper bound on the primal optimum. finding the tightest certificate is itself an LP — the **dual**:

$$\begin{aligned} & \text{minimise} && b^\top u \\ & \text{subject to} && A^\top u \geq c \\ & && u \geq 0. \end{aligned}$$

- **weak duality** (just proved): primal feasible x , dual feasible u give $c^\top x \leq b^\top u$. corollary: if you exhibit a primal and dual pair with equal objectives, both are optimal.
- **strong duality**: if the primal has a finite optimum, so does the dual, and the optima are *equal*. this is the deep theorem (it needs farkas' lemma or the correctness of simplex; clrs proves it via simplex termination (Cormen, Thomas H. and Leiserson, Charles E. and Rivest, Ronald L. and Stein, Clifford, 2009)).
- **complementary slackness**: x^*, u^* are simultaneously optimal iff $u_i^* > 0 \Rightarrow a_i^\top x^* = b_i$ and $x_j^* > 0 \Rightarrow (A^\top u^*)_j = c_j$. a positive dual price forces its constraint to bind; a used primal variable forces its dual constraint to bind.

3.1 the dual of the worked example

$$\begin{array}{ll}\text{minimise} & 4u_1 + 12u_2 + 18u_3 \\ \text{subject to} & u_1 + 3u_3 \geq 3 \\ & 2u_2 + 2u_3 \geq 5 \\ & u_1, u_2, u_3 \geq 0.\end{array}$$

solve it by complementary slackness from the known primal optimum (2, 6):

- constraint 1 is slack ($2 < 4$) so $u_1^* = 0$.
- $x^* = 2 > 0$ forces $u_1 + 3u_3 = 3$, so $u_3^* = 1$.
- $y^* = 6 > 0$ forces $2u_2 + 2u_3 = 5$, so $u_2^* = 3/2$.

dual objective: $4(0) + 12(3/2) + 18(1) = 36 = z^*$. strong duality checks out.

economically the duals are shadow prices: one extra unit of resource 3 is worth exactly 1 unit of objective, and resource 1 is worthless at the margin because we are not even using what we have

4 the simplex method

walk from vertex to adjacent vertex, always improving the objective; stop when no neighbour improves — by convexity that vertex is globally optimal. algebraically each vertex is a *basic feasible solution* (a choice of basic variables solved against the constraints, the rest pinned at zero), and the walk is bookkept in a tableau.

4.1 tableau walkthrough

slack form of the worked example: $x + s_1 = 4$, $2y + s_2 = 12$, $3x + 2y + s_3 = 18$, objective row $z - 3x - 5y = 0$. start at the vertex (0, 0) with basis (s_1, s_2, s_3):

	x	y	s1	s2	s3	rhs
s1	1	0	1	0	0	4
s2	0	2	0	1	0	12
s3	3	2	0	0	1	18
z	-3	-5	0	0	0	0

entering variable — most negative objective coefficient: y (coefficient -5 ; each unit of y buys 5 units of z). **leaving variable** — ratio test $\min\{12/2, 18/2\} = 6$ picks s_2 (row 1 has no y to run out of). pivot on the y -entry of the s_2 row:

	x	y	s1	s2	s3	rhs
s1	1	0	1	0	0	4
y	0	1	0	1/2	0	6
s3	3	0	0	-1	1	6
z	-3	0	0	5/2	0	30

we are now at vertex (0, 6) with $z = 30$. still a -3 under x , so **enter** x ; ratio test $\min\{4/1, 6/3\} = 2$ makes s_3 **leave**:

	x	y	s1	s2	s3	rhs
s1	0	0	1	1/3	-1/3	2
y	0	1	0	1/2	0	6
x	1	0	0	-1/3	1/3	2
z	0	0	0	3/2	1	36

no negative coefficients remain: **optimal**. read off $x = 2$, $y = 6$, $z = 36$. the

path taken was $(0, 0) \rightarrow (0, 6) \rightarrow (2, 6)$ — two pivots, hugging the boundary of the pentagon. bonus: the final objective row’s slack coefficients $(0, 3/2, 1)$ are exactly the optimal dual variables — simplex solves both problems at once.

worst case, simplex visits exponentially many vertices (the klee-minty cube drives dantzig’s pivot rule through all 2^n corners), yet on real instances it is routinely near-linear in the number of constraints — one of the great embarrassments of complexity theory.¹

4.2 degeneracy

if a basic variable sits at zero, a pivot may not move the vertex at all, and careless pivot rules can cycle forever. bland’s rule (always pick the lowest-index eligible variable) or lexicographic tie-breaking guarantees termination (Cormen, Thomas H. and Leiserson, Charles E. and Rivest, Ronald L. and Stein, Clifford, 2009).

5 interior-point methods

instead of crawling the boundary, punch through the middle: replace the constraints with a logarithmic barrier $-\mu \sum_i \log(b_i - a_i^\top x)$, solve the smooth unconstrained problem by newton steps, and shrink $\mu \rightarrow 0$ so the iterates follow the *central path* to the optimal face. karmarkar’s 1984 algorithm made this approach both polynomial-time and practical; modern primal-dual interior-point solvers need only a few dozen newton iterations almost independently of problem size. today’s rule of thumb: simplex wins on warm starts and repeated re-solves (its basis is a resumable certificate), barrier wins on huge sparse one-shot solves. the ellipsoid method settled polynomiality first (khachiyan, 1979) but is hopeless in practice — its enduring role is theoretical: it solves LPs with exponentially many constraints in polynomial time, given a separation oracle.

6 applications

6.1 the diet problem

the original LP (stigler, 1945): meet nutritional floors at minimum cost. variables = quantity of each food, constraints = one per nutrient, objective = total cost. solved honestly below — and, true to form, the solution is depressingly monotonous, because optimal vertices lie on few constraints and therefore use few foods.

stigler’s 1939 diet came out as wheat flour, evaporated milk, cabbage, spinach and dried navy beans — about 39 dollars a year

6.2 max-flow as LP

network optimisation is secretly LP. for a flow network $G = (V, E)$ with capacities c_e , source s , sink t :

¹spielman and teng’s *smoothed analysis* (2004) explains it: perturb any LP instance slightly at random and the expected number of simplex pivots becomes polynomial. adversarial instances are brittle.

references

Cormen, Thomas H. and Leiserson, Charles E. and Rivest, Ronald L. and Stein, Clifford (2009). *Introduction to Algorithms*, MIT Press.

$$\begin{aligned}
& \text{maximise} && \sum_{(s,v) \in E} f_{sv} - \sum_{(v,s) \in E} f_{vs} \\
& \text{subject to} && 0 \leq f_e \leq c_e \quad \forall e \in E \\
& && \sum_{(u,v) \in E} f_{uv} = \sum_{(v,w) \in E} f_{vw} \quad \forall v \in V \setminus \{s, t\}.
\end{aligned}$$

the LP dual of max-flow is min-cut — max-flow/min-cut *is* strong duality wearing a graph costume. shortest paths and min-cost flow have equally direct formulations (Cormen, Thomas H. and Leiserson, Charles E. and Rivest, Ronald L. and Stein, Clifford, 2009). the constraint matrices here are *totally unimodular*, so the LP optimum lands on integer vertices for free — the full story is on the [integer programming](#) page, and the combinatorial algorithms that beat general LP solvers on these problems live under [classical algorithms](#).

7 code

`scipy.optimize.linprog` (the highs solver) on both examples:

```

1 import numpy as np
2 from scipy.optimize import linprog
3
4 # the worked example: max 3x + 5y s.t. x <= 4, 2y <= 12, 3x + 2y <= 18
5 c = [-3, -5] # linprog minimises, so negate
6 A_ub = [[1, 0], [0, 2], [3, 2]]
7 b_ub = [4, 12, 18]
8 res = linprog(c, A_ub=A_ub, b_ub=b_ub, bounds=[(0, None)] * 2, method="highs")
9 print("primal: x* =", res.x, " z* =", -res.fun)
10 print("duals: y* =", res.ineqlin.marginals) # sign flipped by the min
11 # convention
12
13 # a small diet problem: minimise cost, meet nutrient floors
14 # foods: oats, chicken, spinach (per 100 g)
15 cost = [0.55, 1.40, 0.80] # dollars
16 protein = [16.9, 31.0, 2.9] # grams
17 energy = [389, 165, 23] # kcal
18 iron = [4.7, 1.0, 2.7] # mg
19 A = -np.array([protein, energy, iron]) # >= floors -> -A x <= -b
20 b = -np.array([60, 1800, 12])
21 res2 = linprog(cost, A_ub=A, b_ub=b, bounds=[(0, 8)] * 3, method="highs")
22 x = res2.x
23 print("\ndiet (units of 100 g): oats, chicken, spinach =", np.round(x, 3))
24 print("cost = $%.2f" % res2.fun)
25 print("protein %.0f g, energy %.0f kcal, iron %.1f mg"
26 % (protein @ x, energy @ x, iron @ x))

```

```

primal: x* = [2. 6.] z* = 36.0
duals: y* = [-0. -1.5 -1. ]

```

```

diet (units of 100 g): oats, chicken, spinach = [4.627 0. 0. ]
cost = $2.54
protein 78 g, energy 1800 kcal, iron 21.7 mg

```

the reported marginals $(0, -1.5, -1)$ are the duals $(0, 3/2, 1)$ with sign flipped by the minimisation encoding — matching the tableau’s final objective row and the complementary-slackness calculation exactly. and the diet LP does what diet LPs always do: 463 grams of oats a day, nothing else. the energy floor binds; protein and iron overshoot for free. vertex solutions are extreme by nature — if you want variety you must ask for it in the constraints.

8 see also

- **quadratic programming** — the objective goes quadratic, KKT replaces the tableau
- **integer programming** — demand integrality and lose polynomial time
- **non-linear programming** — the general constrained-optimisation story
- **dynamic programming** — the other “programming” that is really planning
- **classical algorithms** — max-flow and friends, solved combinatorially