Deck 05 · Optimization II: constraints and multipliers

4.1 Understanding constrained solutions

4.2 Solving constrained inverse kinematics

Özgür S. Öğüz · Bilkent University · Fall 2026

Improve the task while respecting joint limits, keeping the required distance from obstacles, and meeting exact goals.

Clipping the elbow leaves the shoulder unfinished

Same arm and target as Deck 04:

$$f(\mathbf{q}) = \tfrac{1}{2}\,\|\varphi(\mathbf{q}) - \mathbf{y}^*\|^2, \qquad \mathbf{y}^* = (0.5,\ 0.2)\ \mathrm{m}$$

result	elbow	tip error
unconstrained solution	$\pm 80.4^\circ$	0
clip each update to the $\pm 60^\circ$ range	60°	14.6 cm

The elbow cannot bend further, but the shoulder can still reduce the error.

We need the best allowed configuration.

modules/opt-explorer · Trajectory tab, the lecture arm with elbow clipping (the same preset as Deck 04 slide 37) ▶ run it live

Separate the cost from the requirements

$$\min_{\mathbf{z}\in\mathbb{R}^d} f(\mathbf{z}) \quad \text{subject to} \quad g_i(\mathbf{z}) \le 0, \qquad h_j(\mathbf{z}) = 0$$

robotics requirement	constraint
joint within its range	$q_j - q_j^{\max} \le 0, \quad q_j^{\min} - q_j \le 0$
selected point on the robot stays outside an obstacle's clearance region	$r_o - \lvert\mathbf{p}(\mathbf{q}) - \mathbf{c}\rvert \le 0$
tip exactly at the goal	$\varphi(\mathbf{q}) - \mathbf{y}^* = \mathbf{0}$

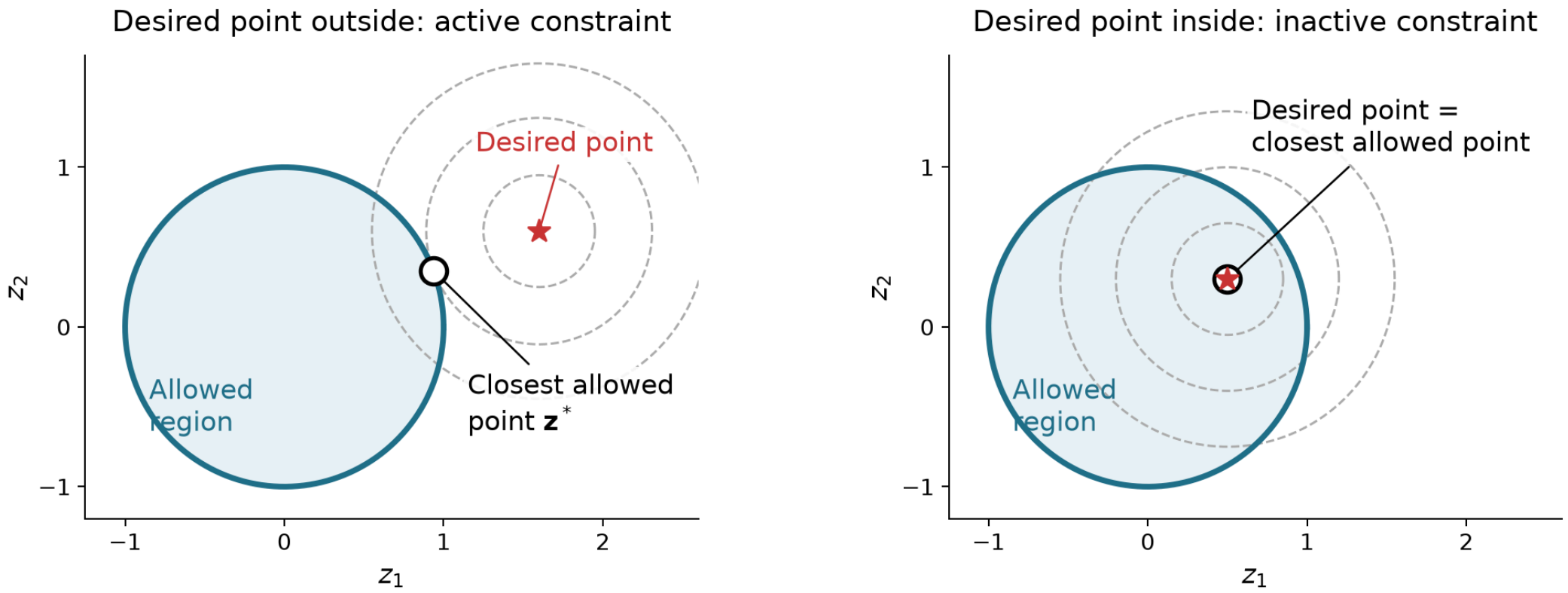
$\mathbf{z}$: decision variables; f: cost; g_i: inequality; h_j: scalar equality.

$\mathbf{p}(\mathbf{q})$: position of the selected point on a robot link.

$\mathbf{c}$: obstacle center; r_o: obstacle radius plus the required safety margin.

Feasible means every requirement holds. Positive g_i means a violation.

A constraint can be active or inactive



Dashed circles: equal cost, equal distance from the desired point

Choose $\mathbf{z}$ closest to the fixed desired point $\mathbf{z}_{\text{target}}$: $\min_{\mathbf{z}} \frac{1}{2}\|\mathbf{z} - \mathbf{z}_{\text{target}}\|^2$ with $g(\mathbf{z}) = \|\mathbf{z}\| - 1 \leq 0$.

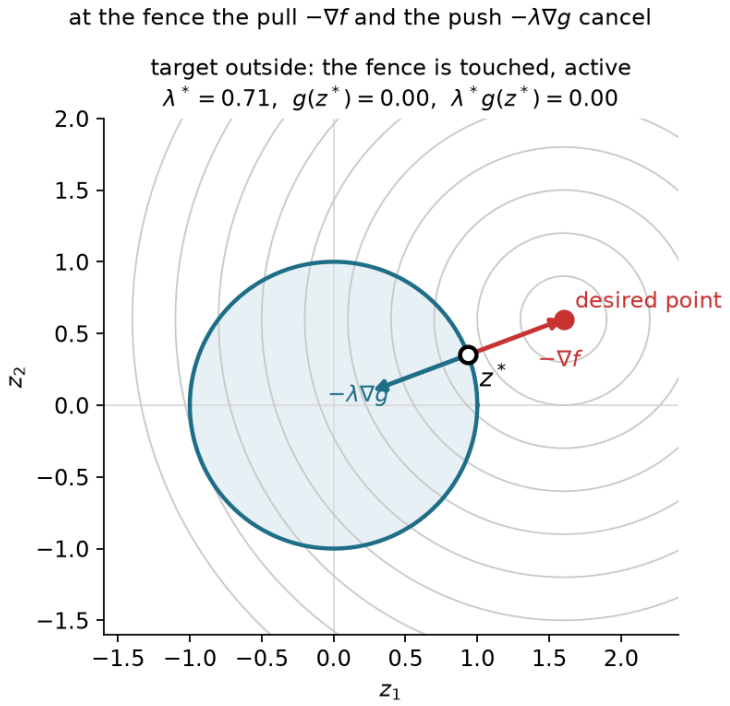
Active: on the boundary. **Inactive:** strictly inside the allowed region.

At a fence, the gradients balance

The cost favors a move through the fence. The best allowed point has no remaining downhill motion along it.

$$\underbrace{-\nabla f(\mathbf{z}^*)}_{\text{downhill direction}} + \underbrace{(-\lambda \nabla g(\mathbf{z}^*))}_{\text{inward constraint correction}} = \mathbf{0}$$

- $\mathbf{z}^*$: the candidate solution
- ∇g: toward the forbidden side
- $\lambda \geq 0$: the **Lagrange multiplier**



In the force picture, the fence pushes inward and never pulls outward. An inactive fence has $\lambda = 0$.

Compute one multiplier from the balance

$$f(\mathbf{z}) = \tfrac{1}{2}\|\mathbf{z} - \mathbf{z}_{\text{target}}\|^2, \qquad g(\mathbf{z}) = \|\mathbf{z}\| - 1 \le 0$$

Target and closest allowed point:

$$\mathbf{z}_{\text{target}} = (2, 0), \qquad \mathbf{z}^* = (1, 0)$$

$$\underbrace{\begin{bmatrix} -1 \\ 0 \end{bmatrix}}_{\nabla f(\mathbf{z}^*)} + \lambda \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\nabla g(\mathbf{z}^*)} = \mathbf{0} \quad \Longrightarrow \quad \lambda = 1$$

Inside target: no restriction, $\lambda = 0$.

Here the multiplier equals the distance beyond the fence because of this cost and the unit constraint gradient.

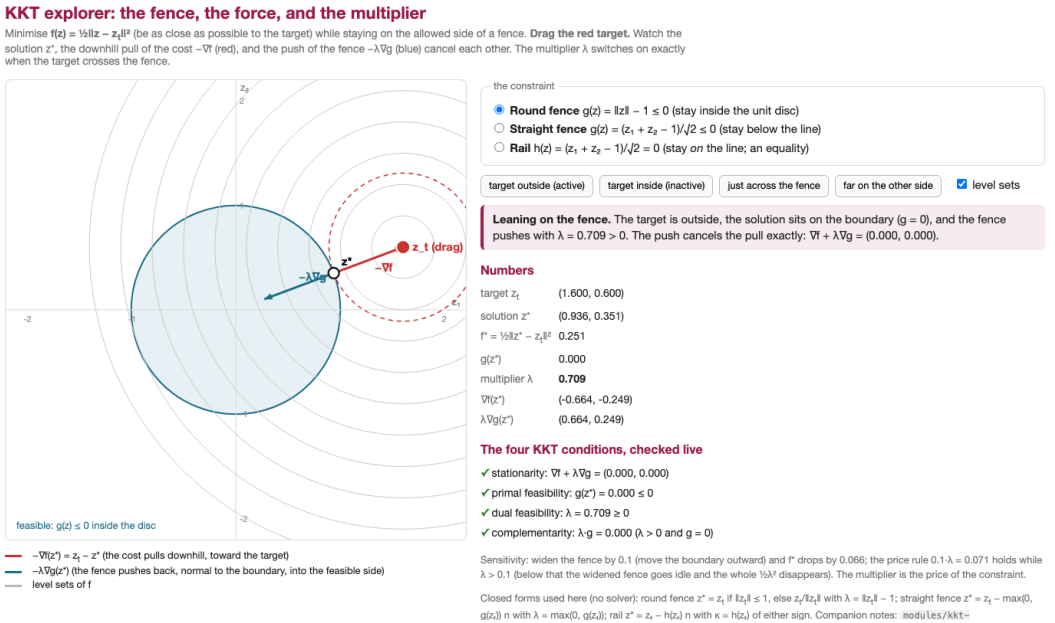
Predict the effect of crossing the fence

The demo opens with the target inside.

Predict: move the target to $(1.02, 0)$, just beyond the unit disc. What multiplier will balance the gradients?

Then compare a target farther outside.

Watch the allowed solution and the two balancing arrows.



modules/kkt-explorer/index.html · round fence ▶ run it live

An exact requirement acts like a rail

- An exact goal must hold from either side: a **rail**
- A rail can oppose a pull in either direction: its multiplier κ can have either sign
- An equality must hold even when its multiplier is zero

$$h(\mathbf{z}) = 0, \qquad \nabla f(\mathbf{z}^*) + \kappa \nabla h(\mathbf{z}^*) = \mathbf{0}$$

For a robot's exact position goal:

$$\mathbf{h}(\mathbf{q}) = \varphi(\mathbf{q}) - \mathbf{y}^* = \mathbf{0}$$

Each scalar equality component has its own multiplier.

modules/kkt-explorer · straight fence, then rail ▶ run it live

Four checks on a constrained answer

$$\nabla f + \sum_i \lambda_i \nabla g_i + \sum_j \kappa_j \nabla h_j = \mathbf{0} \quad \text{gradients balance}$$

$$g_i \le 0, \qquad h_j = 0 \quad \text{all requirements hold}$$

$$\lambda_i \ge 0 \quad \text{fences push inward}$$

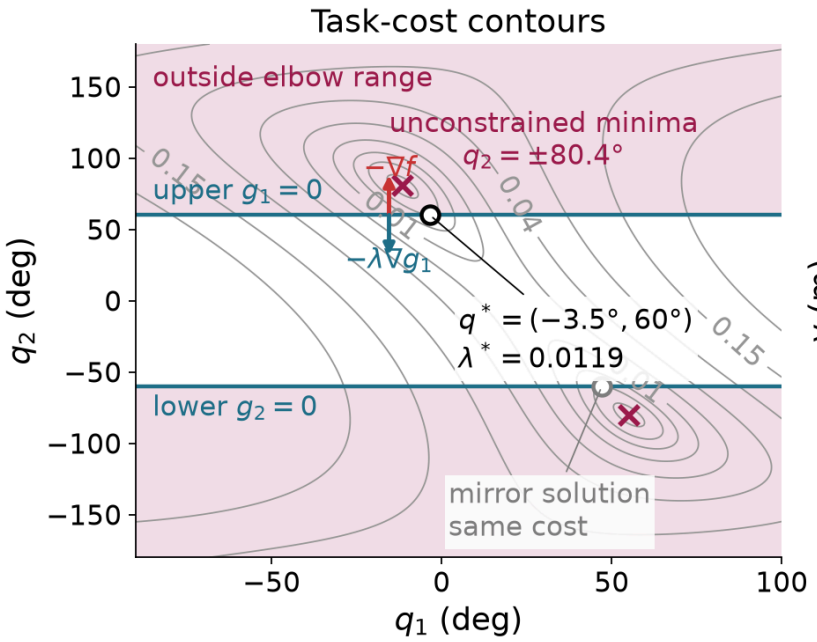
$$\lambda_i g_i = 0 \quad \text{inactive fences have no push}$$

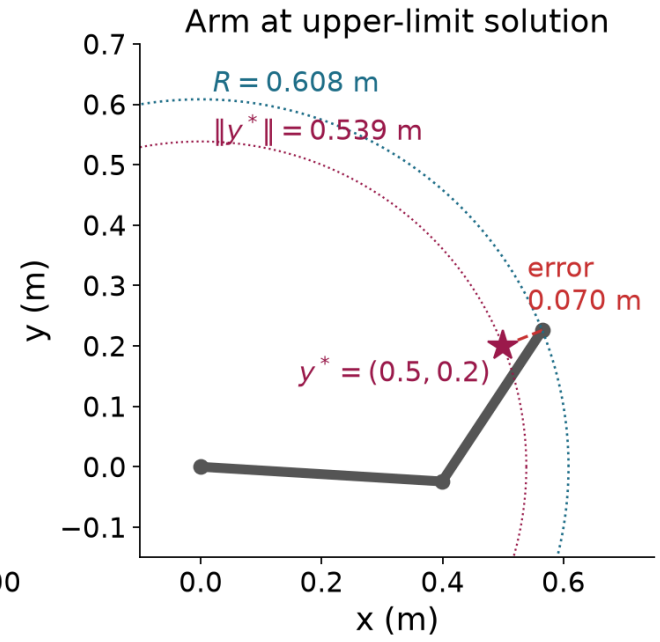
At the candidate solution; equality multipliers may have either sign.

These are the **Karush–Kuhn–Tucker (KKT) conditions**.

Necessary at a local minimum under regularity assumptions. Passing these checks does not prove a minimum.

The constrained arm finishes the shoulder motion





result	tip error
unconstrained step; clamp elbow to ±60°	14.6 cm
constrained solution	7.0 cm

Positive-elbow branch:

$$\mathbf{q}^* = (-3.5^\circ,\ 60^\circ)$$

The shoulder is optimized along the elbow limit.

Upper limit: active, $\lambda_1 = 0.0119\ \mathrm{m}^2/\mathrm{rad}$. Lower limit: inactive, $\lambda_2 = 0$.

code/constrained_ik.py (b) · modules/opt-explorer, Constrained tab, the lecture arm in its answer view ▶ run it live

A multiplier predicts the value of relaxing a limit

Relax one fence to $g(\mathbf{z}) \le b$; let $f^*(b)$ be the optimized cost.

$$\frac{df^*}{db} = \nabla f^\top \frac{d\mathbf{z}^*}{db}$$

$$= -\lambda \underbrace{\nabla g^\top \frac{d\mathbf{z}^*}{db}}_{=1} = -\lambda$$

Line 2 uses slide 5's balance; the brace is 1 while the fence stays active (backup 34).

$$\Delta f^* \approx -\lambda \, \Delta b$$

Upper elbow limit relaxed from 60° to 61°:

quantity	value
multiplier	$0.01192\ \mathrm{m^2/rad}$
predicted cost change	$-2.08 \times 10^{-4}\ \mathrm{m^2}$
computed cost change	$-2.05 \times 10^{-4}\ \mathrm{m^2}$

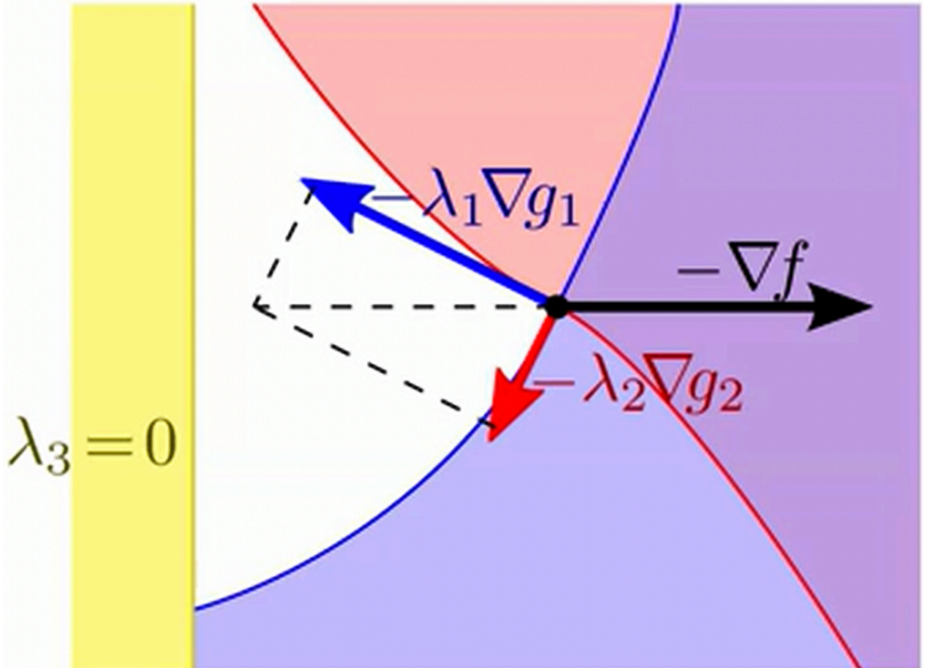
Locally, the multiplier is cost saved per unit of relaxation.

code/constrained_ik.py (c) · modules/opt-explorer, the lecture arm with the limit relaxed to 61° ▶ run it live · alternate: modules/kkt-explorer, the disc sensitivity readout ▶ run it live

Which fences contribute to the balance?

- Several touched fences can contribute to one balance
- An inactive fence has a zero multiplier
- A touched fence may also have a zero multiplier (e.g., when the unconstrained minimum lies exactly on its boundary)
- An equality constraint may have a positive, negative, or zero multiplier

We generally do not know the active constraints before solving.



One function collects the gradient balance

$$L(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\kappa}) = f(\mathbf{z}) + \boldsymbol{\lambda}^\top \mathbf{g}(\mathbf{z}) + \boldsymbol{\kappa}^\top \mathbf{h}(\mathbf{z})$$

This is the **Lagrangian**.

Each constraint has its own multiplier. The vectors collect the inequality values and their multipliers, and the equality values and their multipliers.

$$\nabla_{\mathbf{z}} L = \nabla f + \sum_i \lambda_i \nabla g_i + \sum_j \kappa_j \nabla h_j$$

- At a KKT point, this gradient is zero
- Feasibility and the other KKT checks still matter
- The second half shows how solvers handle constraints and obtain multiplier estimates

The same requirements appear throughout robotics

robotics task	constrained formulation
choose a configuration	inverse kinematics with joint limits, clearance, and an exact goal
choose a motion	optimize a trajectory, with requirements at its sampled configurations
update a controller	choose a local motion while respecting limits
refine a learned proposal	check or improve the proposed configuration or motion with a constrained solver

A solver result still needs feasibility checks. Local convergence does not establish a globally best answer or physical safety.

Summary of 4.1

- **Feasibility:** obey every fence and rail
- **Balance:** $\nabla f(\mathbf{z}^*)$ need not be $\mathbf{0}$
  - $\nabla f(\mathbf{z}^*) = -\sum_i \lambda_i \nabla g_i(\mathbf{z}^*) - \sum_j \kappa_j \nabla h_j(\mathbf{z}^*)$
- **Activity:**
  - inactive fence, $g_i(\mathbf{z}^*) < 0$: $\lambda_i = 0$
  - active fence, $g_i(\mathbf{z}^*) = 0$: $\lambda_i \geq 0$
- **Rails:** always active; κ_j of either sign
- **Sensitivity:** a multiplier predicts the local benefit of relaxing a requirement

Next: how does a solver find a constrained answer?

Break

Second half: constrained inverse kinematics, penalty and barrier methods, and the augmented Lagrangian method.

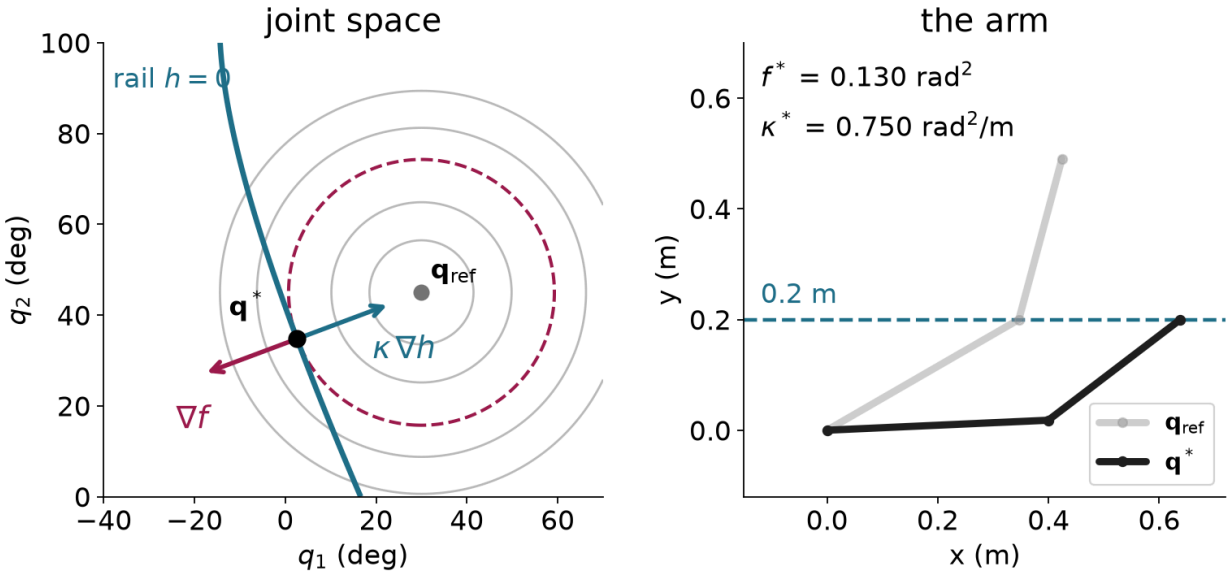
Inverse kinematics as a nonlinear program

$$\min_{\mathbf{q}} \ \tfrac{1}{2}\|\mathbf{q}-\mathbf{q}_{\text{ref}}\|_{\mathbf{D}}^2 \quad \text{s.t.} \quad \varphi(\mathbf{q})-\mathbf{y}^{*}=\mathbf{0}, \quad \mathbf{q}^{\min}\le\mathbf{q}\le\mathbf{q}^{\max}, \quad r_o+r_p-\|\mathbf{p}_p(\mathbf{q})-\mathbf{c}\|\le 0$$

- Changes from Deck 04:
  - target: soft → **hard equality**
  - cost: a **fixed** posture $\mathbf{q}_{\text{ref}}$, no step penalty
- $\mathbf{q}_{\text{ref}}$: $\mathbf{q}_0$ (essentials §5), $\mathbf{q}_{\text{ready}}$ (Panda demo)
- Same slot, more features: gaze, balance, contacts (Deck 06, A2)

Lecture arm, $\mathbf{D}=\mathbf{I}$, only the tip height fixed:

$$h(\mathbf{q})=\varphi_2(\mathbf{q})-0.2\,\text{m}=0$$



A curve of feasible answers; the posture cost picks one.

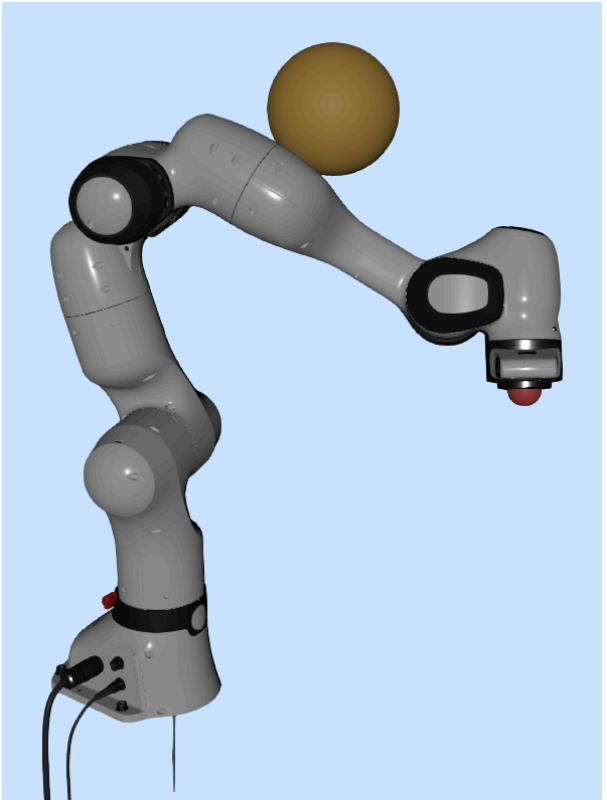
code/constrained_ik.py (n) · modules/opt-explorer, Constrained tab, the lecture arm with a tip-height rail ▶ run it live

Live demo: IK as a nonlinear program on the Panda

Slide 17's program on a 7-joint arm; one solver, SNOPT, through Drake's `InverseKinematics`

term	on the Panda
cost	$\frac{1}{2}\|\mathbf{q} - \mathbf{q}_{\text{ready}}\|^2$
rail	flange on the target: $\varphi(\mathbf{q}) - \mathbf{y}^* = \mathbf{0}$
fences	URDF joint limits; all 44 collision spheres of links 1–8 clear of the obstacle

live start	value
target $\mathbf{y}^*$	$(0.45, 0.10, 0.45)$ m
obstacle center $\mathbf{c}$	$(0.175, 0.15, 0.75)$ m, above the elbow
radius r_o	0.08 m



- **Predict:** which link touches; the active set 10 and 15 cm lower; a target inside the obstacle
- Live page: target sliders and presets; cost, active rows with λ, $\boldsymbol{\kappa}$, each link's clearance

drake/03-ik-as-nlp/ik_nlp_panda.py --app ▶ run it live

Panda results: active sets, multipliers and prices

page state (slide 18)	active inequalities	λ	$\boldsymbol{\kappa}$	cost
15 cm lower	none	–	$(-2.23, -0.74, 2.01)$	0.408
near the base	joint 4, lower limit	1.153	$(4.42, 0.00, 0.48)$	0.412
the start	2 spheres of link 5	$0.79, 7.93$	$(-2.96, -2.31, -1.67)$	0.364

the start, radius r_o: $0.080 \to 0.081$ m	**cost change**
predicted: $(\lambda_1 + \lambda_2) \times 0.001$	$+0.00873$
re-solved	$+0.00884$

- Rail (equality): always active; $\boldsymbol{\kappa} \neq \mathbf{0}$ in all three states
- Fence (inequality): $\lambda = 0$ when idle (15 cm lower), $\lambda > 0$ when it pushes

the live page of slide 18 (drake/03-ik-as-nlp/ik_nlp_panda.py --app) · the print mode gives the same numbers

How does a solver handle the fences? Three ways to build one out of costs

construction	augmented objective	outer loop	character
squared penalty	$S = f + \mu \sum_i [g_i > 0]\, g_i^2 + \mu \sum_j h_j^2$	$\mu \to \infty$	cost grows quadratically past the fence; trespasses wherever a fence must push ($\lambda^* > 0$)
log barrier	$B = f - \mu \sum_i \log(-g_i)$	$\mu \to 0$	cost grows without bound as the fence nears from inside; never touches; needs a feasible start; fences only
augmented Lagrangian	$A = f + \sum_i [g_i > 0 \vee \lambda_i > 0](\lambda_i g_i + \mu g_i^2) + \sum_j (\kappa_j h_j + \mu h_j^2)$	$\boldsymbol{\lambda}, \boldsymbol{\kappa}$ updated between inner solves; μ can stay fixed (slide 24)	the penalty plus a linear term $\lambda_i g_i$; λ corrected from the violation

- **Shared idea:** in the balance $\nabla f + \lambda \nabla g = \mathbf{0}$, replace the unknown λ by a formula we can compute: $2\mu g$, $-\mu/g$, $\lambda + 2\mu g$
  - unconstrained inner solves (Deck 04); one KKT condition relaxed, by a controlled amount

So far: closed forms (2-D arm) and SNOPT, a sequential quadratic programming solver (Panda; backup 39). Slide 27 runs one of these three on the Panda and matches SNOPT.

Squared penalty: a quadratic cost past the fence

$$S(\mathbf{z};\mu) = f(\mathbf{z}) + \mu\,[g > 0]\,g(\mathbf{z})^2$$

At the minimum of S, past the fence:

$$\nabla f + \underbrace{2\mu g}_{\text{acts as } \lambda}\,\nabla g = \mathbf{0} \quad \Rightarrow \quad g \approx \frac{\lambda^*}{2\mu}$$

- **Trespass** $\propto 1/\mu$: exact only as $\mu \to \infty$
- **Large μ:** ill-conditioned inner solves

Elbow limit, from $(30^\circ, 45^\circ)$:

μ	q_2 (deg)	violation	$2\mu g$
1	60.338	0.338°	0.01181
10	60.034	0.034°	0.01191
100	60.003	0.003°	0.01192
1000	60.000	0.0003°	0.01192

$$\lambda^* = 0.01192$$

code/constrained_ik.py (d) · modules/opt-explorer, Constrained tab, the lecture arm with the squared penalty ▶ run it live

Log barrier: a cost that grows without bound at the fence

$$B(\mathbf{z};\mu) = f(\mathbf{z}) - \mu \log(-g(\mathbf{z}))$$

$$(1)\ \text{stationarity of } B: \quad \nabla f - \tfrac{\mu}{g}\nabla g = \mathbf{0}$$

$$(2)\ \text{name the push:} \quad \lambda := -\mu/g > 0$$

$$(3)\ \text{substitute:} \quad \nabla f + \lambda \nabla g = \mathbf{0}, \qquad \lambda\, g = -\mu$$

- **Start and stay strictly inside** the inequality boundaries
- **Decrease** μ to approach the boundary more closely

Equality constraints need separate handling.

Elbow limit, from $(30^\circ, 45^\circ)$:

μ	elbow angle
10^{-2}	28.833°
10^{-3}	55.786°
10^{-4}	59.527°
10^{-6}	59.995°

Approaches the 60° limit from below.

code/constrained_ik.py (d) · modules/opt-explorer, the lecture arm with the log barrier ▶ run it live

Augmented Lagrangian: the penalty plus a learned multiplier

$$A = f + \lambda g + \mu g^2 \quad (\text{while } g > 0 \text{ or } \lambda > 0)$$

1. **Inner solve:** hold multipliers fixed and optimize the configuration.

At an inner stationary point:

$$0 = \nabla f + (\lambda + 2\mu g)\, \nabla g$$

The coefficient $\lambda + 2\mu g$ gives an updated multiplier estimate.

2. **Outer update:** hold the configuration fixed and update the multipliers

$$\lambda \leftarrow \max(\lambda + 2\mu g(\mathbf{z}'),\, 0), \qquad \kappa \leftarrow \kappa + 2\mu h(\mathbf{z}')$$

Why update the multiplier?

- At the boundary, the violation is zero, so the squared penalty contributes no gradient
- The multiplier term can still balance the task gradient
- This arm example reduces violation with μ **fixed**

For squared-error tasks, the inner solve can reuse damped Gauss–Newton.
The residual construction is in the detailed notes.

Toussaint 2014, arXiv:1412.4329 · modules/opt-explorer, the lecture arm with the augmented Lagrangian at μ = 1 ▶ run it live

The augmented Lagrangian learns the multiplier

A fixed penalty weight can be sufficient

- Near a suitable local minimum, with sufficiently large μ
- No guarantee from an arbitrary starting configuration

Implementations can choose different weight policies

implementation	penalty weight μ
this elbow example	held fixed at 1
RAI's documented default	multiplied by 5 per outer update

A fixed weight can work; increasing it remains an implementation choice.
Stopping checks are explained on slide 25.

Elbow limit, $\mu = 1$ fixed:

outer	λ before	q_2 (deg)	g	λ after
0	0	60.338	$+0.338^\circ$	0.011805
1	0.011805	60.003	$+0.003^\circ$	0.011917
2	0.011917	60.000	$+0.00003^\circ$	0.011918
3	0.011918	60.000	0	0.011918

lower limit: $\lambda_2 = 0$ throughout

Toussaint 2014, arXiv:1412.4329 · drake/04-constrained-ik-al/constrained_ik_panda.py (the same loop on the Panda; fallback fig/drake-constrained-ik-al.png) ▶ run it live · alternate: modules/opt-explorer, Constrained tab ▶ run it live

Stopping with fences: three residuals, not one gradient

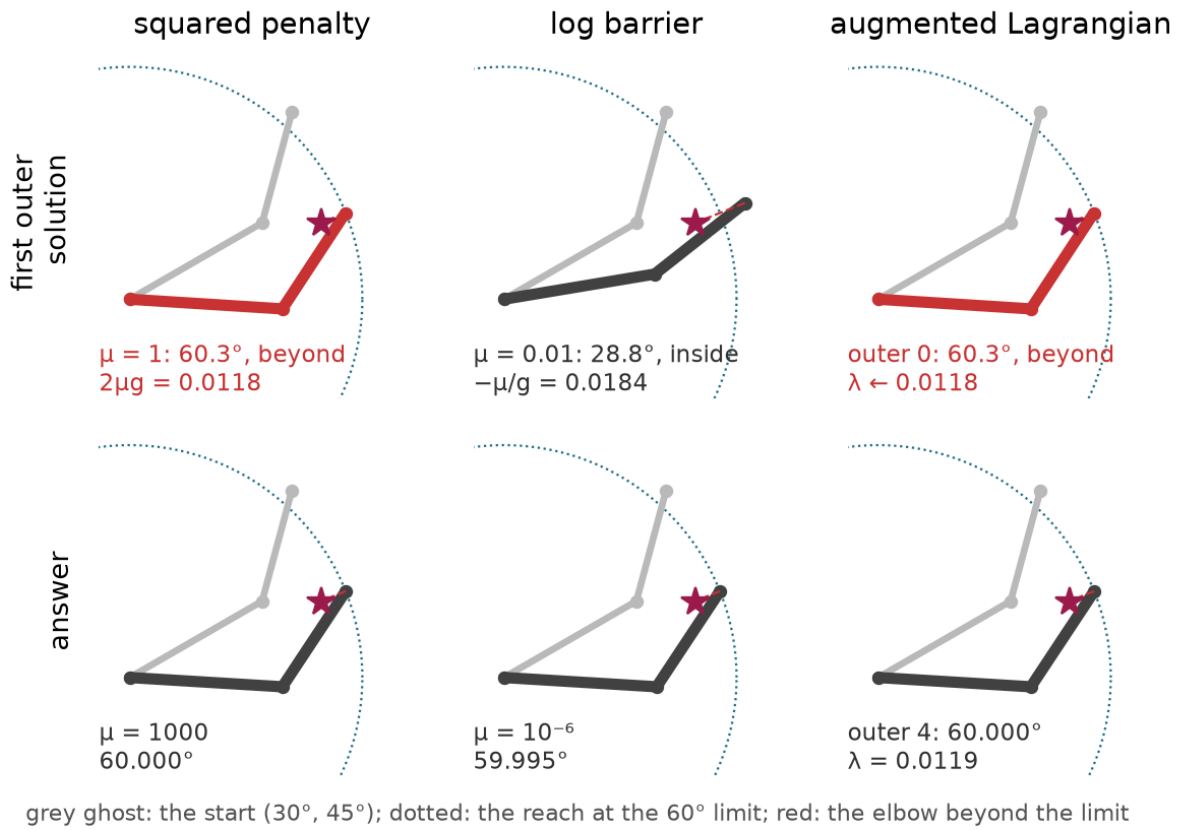
residual (updated multipliers)	meaning	elbow limit, AL $\mu = 1$, outer 3	Three-point trajectory, RAI result
feasibility: $\max g^{+}$	on the allowed side	4.9×10^{-9}	1.2×10^{-11}
feasibility: $\max \lvert h_j \rvert$	on every rail	— (no rail)	4.5×10^{-13}
Lagrangian stationarity: $\lVert \nabla f + \sum_i \lambda_i \nabla g_i + \sum_j \kappa_j \nabla h_j \rVert$	forces balance	1.3×10^{-11}	5.4×10^{-6}
complementarity: $\max_i \lvert \lambda_i g_i \rvert$	idle fences push nothing	5.9×10^{-11}	5.2×10^{-9}

Recorded replay: the tip error settles while the violation keeps falling.

solved	stationary	stalled	limit
all three residuals small	force balance holds; another residual not settled	an inner solve rejects its direction	outer iteration cap reached

code/constrained_ik.py (d) · code/al_gauss_newton.py · Deck 06 code/check_rai_nlp.py · code/check_solvers.py

Comparison



grey ghost: the start (30°, 45°); dotted: the reach at the 60° limit; red: the elbow beyond the limit

	squared penalty	log barrier	augmented Lagrangian
rails	yes	no	yes
start anywhere	yes	no, strictly inside	yes
iterates feasible	no, trespass $\propto 1/\mu$	yes, interior	no, until λ settles
exact in the limit	$\mu \to \infty$	$\mu \to 0$	fixed μ, λ converges
conditioning	worsens as $\mu \to \infty$	worsens as $\mu \to 0$	μ need not grow
where you meet it	soft costs	interior-point (IPOPT)	KOMO, RAI's `NLP_Solver`

The target is a constraint; posture is the cost

Slide 18's program without the sphere (the print mode's runs 1 and 2), solved by slide 24's loop with both weights fixed:

$$\min_{\mathbf{q}} \ \tfrac{1}{2}\|\mathbf{q}-\mathbf{q}_{\text{ready}}\|^2 \quad \text{s.t.} \quad \underbrace{\varphi(\mathbf{q})-\mathbf{y}^*=\mathbf{0}}_{\text{rail } \mathbf{h},\ \boldsymbol{\kappa},\ \mu_h=1000\ \text{rad}^2/\text{m}^2} \quad , \quad \underbrace{\mathbf{q}^{\min}\le\mathbf{q}\le\mathbf{q}^{\max}}_{\text{14 fences},\ \boldsymbol{\lambda},\ \mu_g=100}$$

run	target $\mathbf{y}^*$ (m)	joint bounds at the answer	λ
A (the start's target)	$(0.45, 0.10, 0.45)$	all 14 **inactive**	$\mathbf{0}$
B (behind the robot)	$(-0.20, 0, 0.60)$	joint 2 **active** at its lower bound -1.7628 rad	1.053 (joint 2 only)

Both answers are local solutions from the demo's stated initial postures.

- Slide 24: the squared distance was the **cost**. Here the distance is the rail's **violation** $\|\mathbf{h}\|$, driven to 0 within tolerance
- Inner step, a joint update: the arm moves, $\boldsymbol{\kappa}$ and $\boldsymbol{\lambda}$ frozen
- Outer step, a multiplier update, the arm fixed: $\boldsymbol{\kappa} \leftarrow \boldsymbol{\kappa} + 2\mu_h\mathbf{h}$ and $\lambda_i \leftarrow \max(0, \lambda_i + 2\mu_g g_i)$

drake/05-ik-nlp-al/ik_nlp_al_panda.py --replay --app (fallback fig/drake-ik-nlp-al.png) ▶ run it live

Failure modes (1): posing and stopping

- **Sign of g:** "distance at least r_o" is $r_o - \|\mathbf{p}_p - \mathbf{c}\| \leq 0$; reversed, the obstacle's inside is feasible
- **Negative λ:** fails dual feasibility, so not a KKT point as posed; a negative κ is fine
- **Penalty "converged" but violated:** a finite μ trespasses wherever the fence must push; check g, not only the step norm
- **Stopping on $\|\nabla f\|$ with fences:** 0.012 at the elbow optimum; use slide 25's three residuals

Failure modes (2): interpreting a solver result

- **Log barrier:** accepted iterates need strictly satisfied inequalities; equalities need separate treatment
- **KKT residuals:** a stationary point can still be a saddle
- **Local solve:** the initial guess can change the returned answer
- **Failure to find a feasible point:** a solver report alone does not prove infeasibility

State the model, initial guess, and measured residuals.

Constrained optimization supplies an allowed endpoint

- **Formulate:** a cost ranks answers; constraints state requirements
- **Solve:** penalty, barrier, or augmented-Lagrangian constructions
- **Check:** feasibility, Lagrangian stationarity, and complementarity
- **Interpret:** local answers and multipliers depend on the stated problem

Next: connect the current configuration to the chosen goal.

From an allowed goal to a safe motion

We can find an allowed goal configuration.

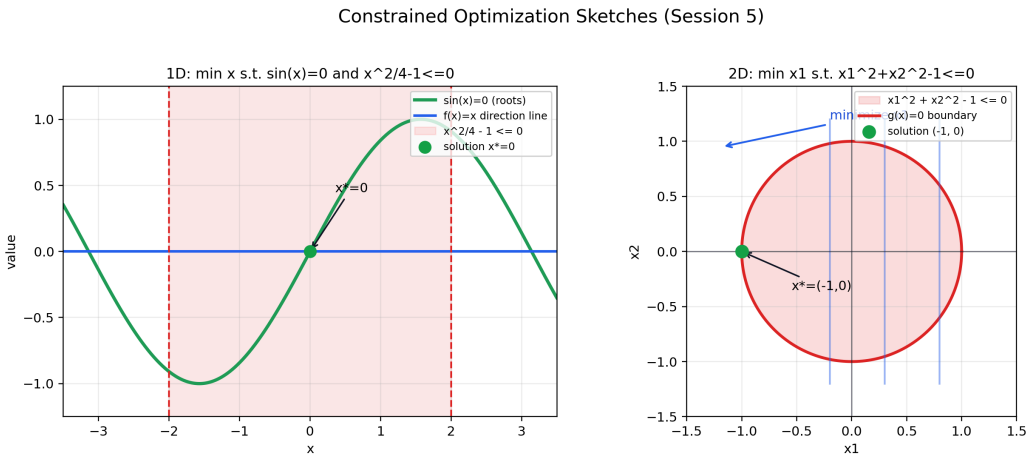
We still need a motion that reaches it safely.

Deck 06 · Motion planning: optimization and sampling

Backup slides

Backup · Solve by sketching

1. $\min x$ s.t. $\sin x = 0$, $x^2/4 - 1 \leq 0$: which constraints are active, which multipliers nonzero?
2. $\min x_1$ s.t. $x_1^2 + x_2^2 - 1 \leq 0$: force balance at the solution?
3. Wedge: $\min x + y$ s.t. $0.5x - y - 0.2 \leq 0$, $-x + 0.5y - 0.2 \leq 0$
   - solution: $(x, y) = (-0.4, -0.4)$, both active
   - multipliers: $\lambda_0 = \lambda_1 = 2$



Backup · The price of a constraint, derived

Setup, perturbed problem: $\min f(\mathbf{z})$ s.t. $g(\mathbf{z}) \le b$, active near $b = 0$

- solution $\mathbf{z}^*(b)$, optimal cost $f^*(b) = f(\mathbf{z}^*(b))$
- assumption: the same fences stay active for small b, so $\mathbf{z}^*(b)$ moves smoothly

1. Differentiate the identity $g(\mathbf{z}^*(b)) = b$: $\quad \nabla g(\mathbf{z}^*)^\top \dfrac{d\mathbf{z}^*}{db} = 1$
2. Differentiate the cost; use stationarity $\nabla f = -\lambda \nabla g$:

$$\frac{df^*}{db} = \nabla f(\mathbf{z}^*)^\top \frac{d\mathbf{z}^*}{db} = -\lambda \, \nabla g(\mathbf{z}^*)^\top \frac{d\mathbf{z}^*}{db} = -\lambda$$

3. Several constraints: $\partial f^* / \partial b_i = -\lambda_i$; inactive ones have $\lambda_i = 0$ and no price

Backup · The central path, and the barrier's relaxed KKT

At the inner optimum of $B(\mathbf{z};\mu)$, with

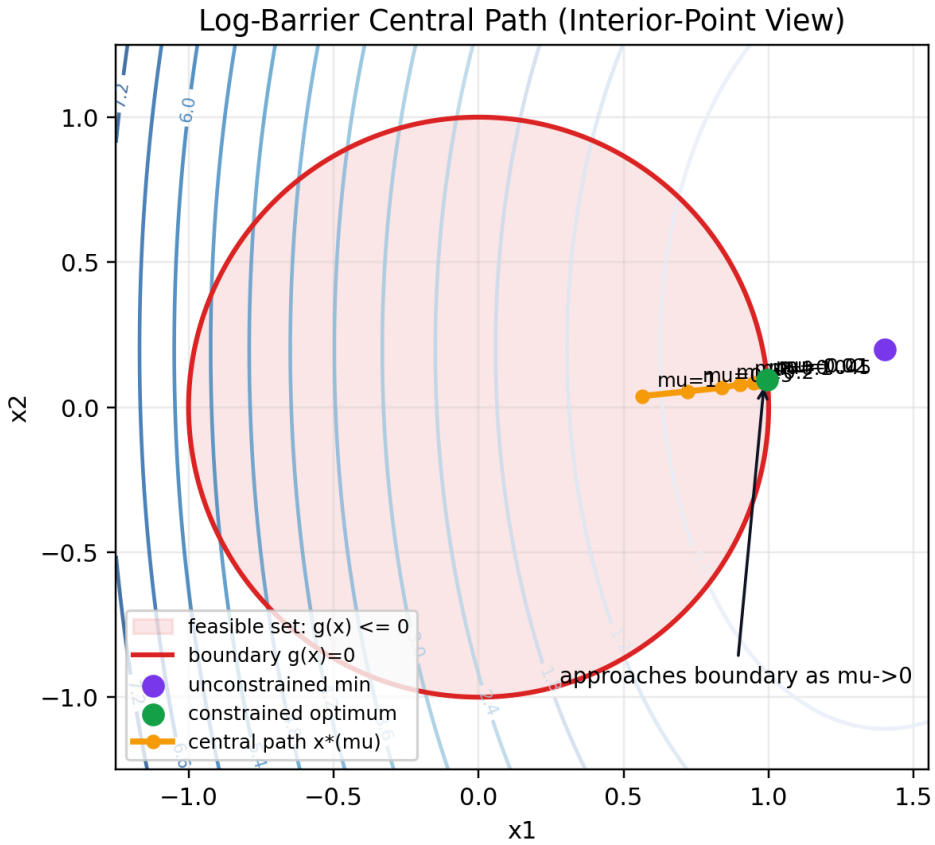
$$\lambda_i := -\mu/g_i > 0$$

condition	exact KKT	barrier optimum
stationarity	$\nabla f + \sum \lambda_i \nabla g_i = \mathbf{0}$	same
dual feasibility	$\lambda_i \geq 0$	automatic
primal feasibility	$g_i \leq 0$	strict, $g_i < 0$
complementarity	$\lambda_i g_i = 0$	$\lambda_i g_i = -\mu$

Duality gap bound (**convex** only, m fences):

$$f(\mathbf{z}'(\mu)) - f^* \leq m\mu$$

- arm's saddle run (notes §7 (5)): fails by four orders of magnitude



Backup · Gradient and Hessian of the barrier

$$\nabla_{\mathbf{z}} B = \nabla f - \mu \sum_i \frac{1}{g_i} \nabla g_i, \qquad \nabla^2_{\mathbf{z}} B \approx \nabla^2 f + \mu \sum_i \frac{1}{g_i^2} \nabla g_i \nabla g_i^\top$$

- $-\mu / g_i > 0$: a repulsive force, growing as the fence is approached
- Hessian model drops $\nabla^2 g_i$, as Gauss-Newton drops the residuals' second derivatives
- Added term: positive semidefinite, size λ_i^2 / μ
  - active fence: unbounded as $\mu \to 0$; conditioning worsens, like the penalty's
- Inner solver: Newton on B with this model, Armijo backtracking, warm start
  - stop when $m\mu$ is below the tolerance: a valid test for convex problems only

Backup · The augmented Lagrangian, derived

1. One rail: $A = f + \kappa h + \mu h^2$; stationarity at the inner minimum z':

$$0 = \nabla f + \kappa \nabla h + 2\mu h \nabla h = \nabla f + (\kappa + 2\mu h)\nabla h$$

2. Match with KKT, $\nabla f + \kappa^* \nabla h = 0$: the parenthesis is the new κ
3. Linear f and h: exact after one update

One-dimensional example (table at right):

- problem: $\min \frac{1}{2} z^2$ s.t. $z - 1 = 0$
- penalty weight: $\mu = 1$
- inner minimum: $z' = (2 - \kappa)/3$

outer	κ	z'	$h(z')$	κ new
0	0	0.6667	-0.3333	-0.6667
1	-0.6667	0.8889	-0.1111	-0.8889
2	-0.8889	0.9630	-0.0370	-0.9630
3	-0.9630	0.9877	-0.0123	-0.9877
∞	-1	1	0	-1

h shrinks by $1/(1 + 2\mu) = 1/3$ per outer iteration at **fixed** μ.

Backup · The full algorithm with inequalities

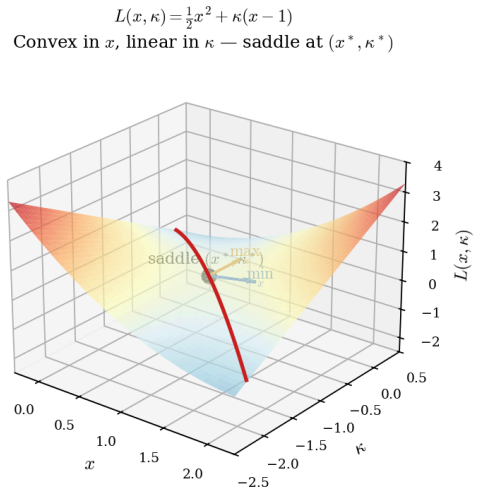
```
μ ← μ0;  λ ← 0;  κ ← 0
repeat
    z ← argmin_z  f(z) + Σ_i [g_i(z) > 0 ∨ λ_i > 0] (λ_i g_i(z) + μ g_i(z)²) + Σ_j (κ_j h_j(z) + μ h_j(z)²)     # inner: Newton / Gauss-Newton / BFGS, warm-started
    λ_i ← max(λ_i + 2μ g_i(z), 0)                                                                               # fences: clamp at zero
    κ_j ← κ_j + 2μ h_j(z)                                                                                       # rails
    optionally μ ← c μ                      # RAI's documented default: c = 5
until  max_i g_i(z)⁺, max_j |h_j(z)| ≤ 10⁻⁹              # feasibility          (slide 25's three residuals,
  and  ‖∇f + Σ_i λ_i ∇g_i + Σ_j κ_j ∇h_j‖ ≤ 10⁻⁸           # Lagrangian stationarity   with the updated multipliers;
  and  max_i |λ_i g_i(z)| ≤ 10⁻¹¹                        # complementarity       stop also if an inner solve stalls)
```

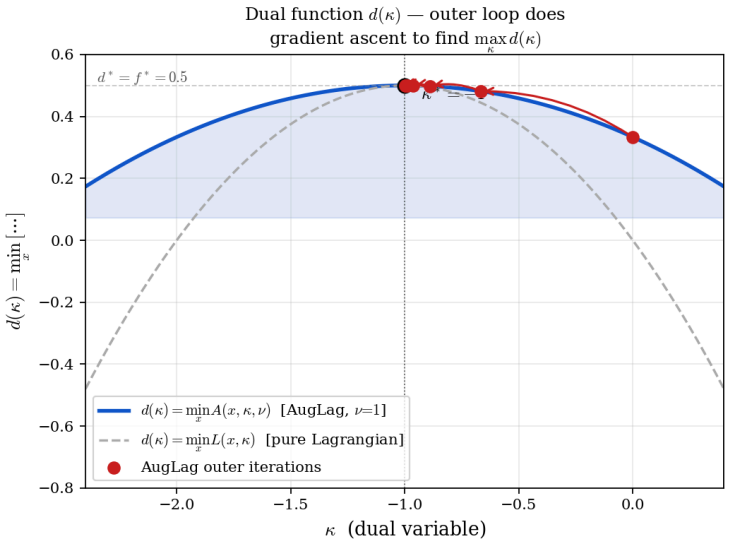
- The bracket $[g_i > 0 \vee \lambda_i > 0]$ decides activity **pointwise**, at every evaluation: no enumeration of active sets
- $\lambda_i > 0$ with $g_i < 0$: a transient after a constraint stops being active; the clamp drives it to zero
- Theory: a fixed μ above a problem-dependent threshold suffices
- Practice: growth speeds convergence (RAI's documented default: slide 24)

Toussaint 2014, arXiv:1412.4329

Backup · The solver landscape, and the saddle-point view

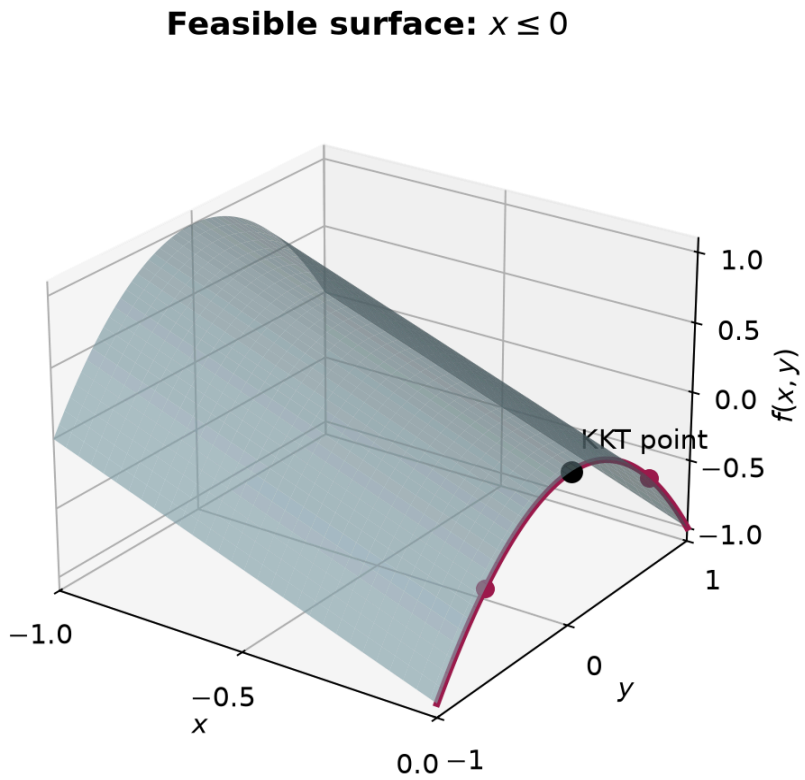
method	idea
augmented Lagrangian	guess the multipliers, minimize, correct them from the residual
primal-dual Newton	Newton on the KKT system in $(\mathbf{z}, \boldsymbol{\lambda}, \boldsymbol{\kappa})$ jointly; interior-point solvers (IPOPT)
sequential quadratic programming	a quadratic program (QP) with the linearized constraints per step; multipliers inherited from the QP (SNOPT)
active-set	guess the touched fences, solve the equality problem, revise

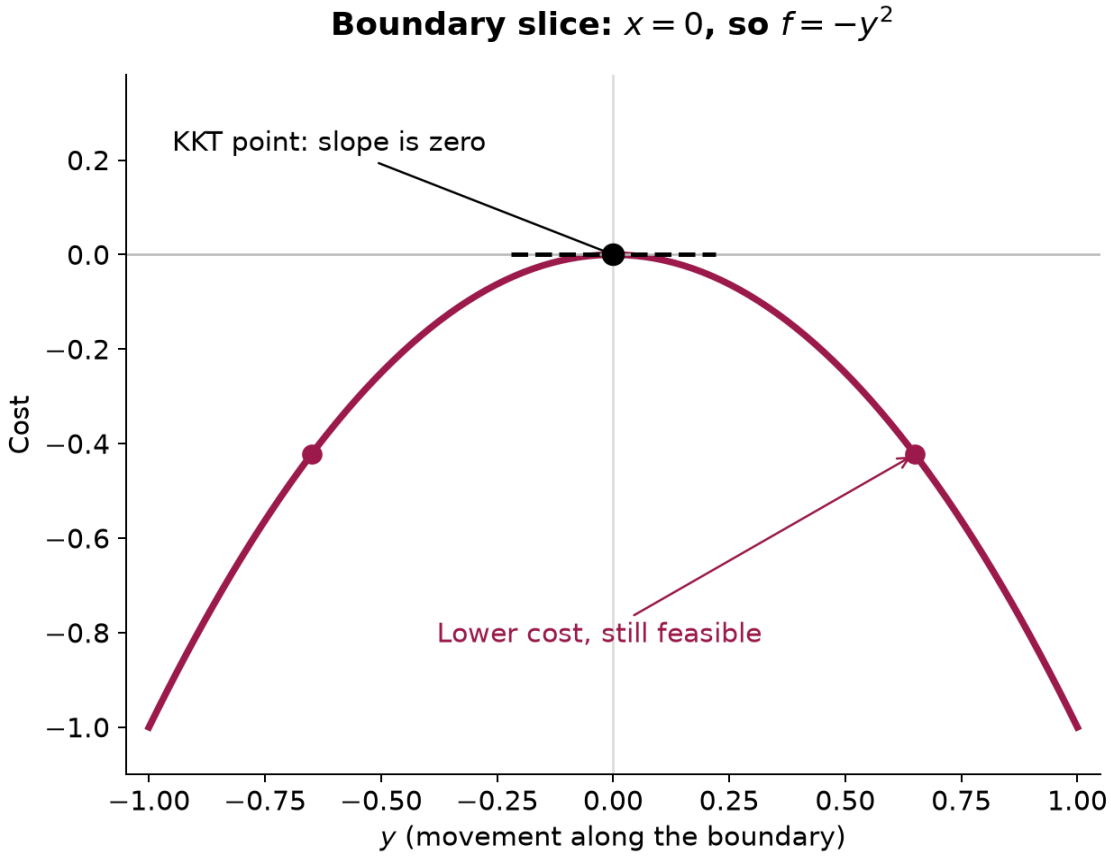




Backup · KKT balance does not guarantee a minimum

Minimize $f(x, y) = -x - y^2$ **subject to** $x \leq 0$





At the origin, all KKT checks pass with $\lambda = 1$.

Moving along the boundary still lowers the cost.