ORF307 – Optimization

17. Interior-point methods

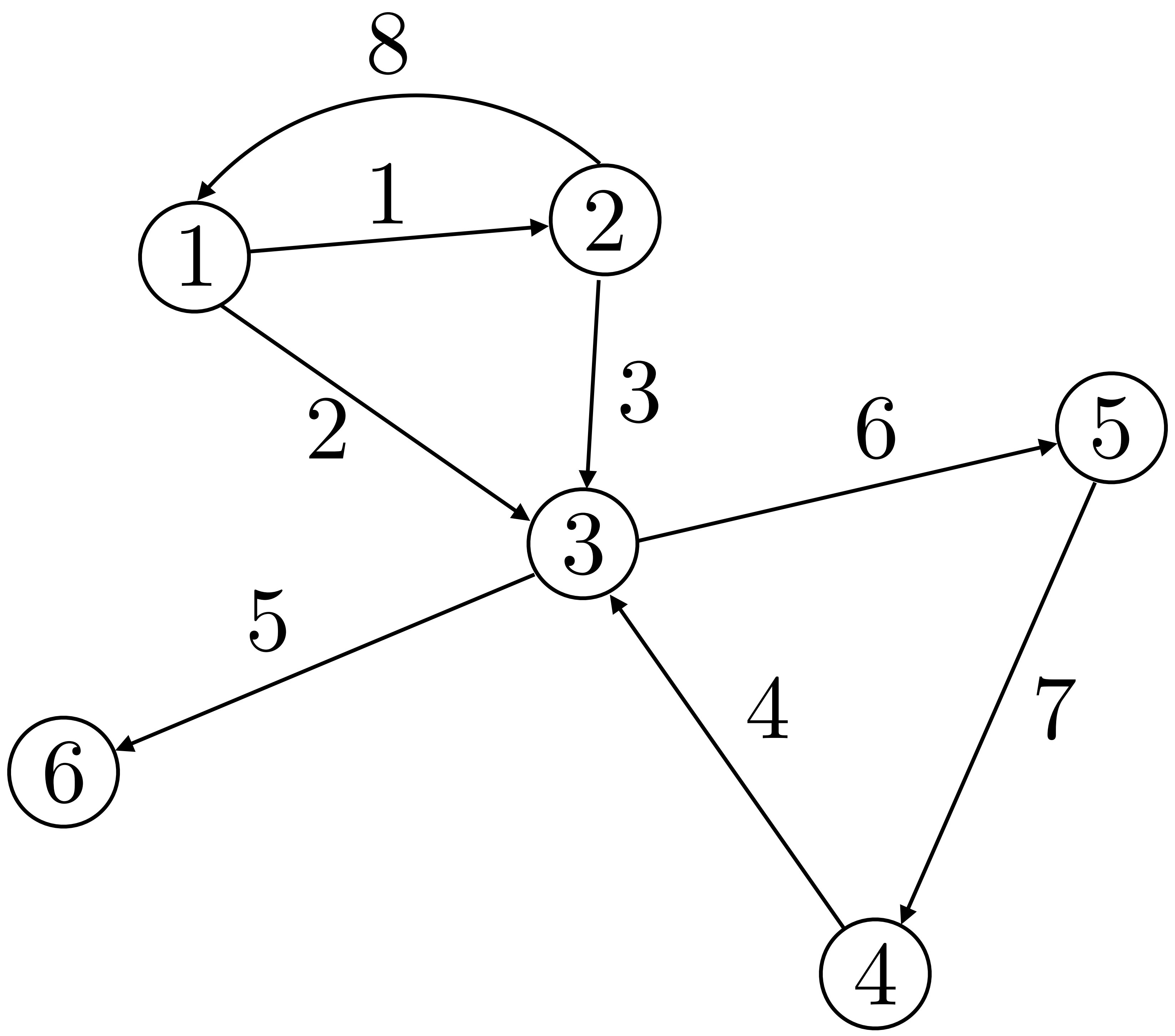
Bartolomeo Stellato — Spring 2022

Ed Forum

- We learned about networks this lecture and in another class, I learned about neural networks. I get that the two things are similar, in the sense that there exists weights to the edges. In neural networks, there also the bias on a node. However in our examples in lecture, the b is referenced as a supply vector. I was wondering if the two were the same.
- In our example, we only looked at the case where the external supply came into certain nodes while the external demand came into others, but we never looked at nodes that had both supply and demand coming in. Can this ever happen and if so how would this affect b since it appears the values of b are usually broken up into positive for supply and negative for demand?

Recap

Arc-node incidence matrix



$m \times n$ matrix A with entries

$$A_{ij} = \begin{cases} 1 & \text{if arc } j \text{ starts at node } i \\ -1 & \text{if arc } j \text{ ends at node } i \\ 0 & \text{otherwise} \end{cases}$$

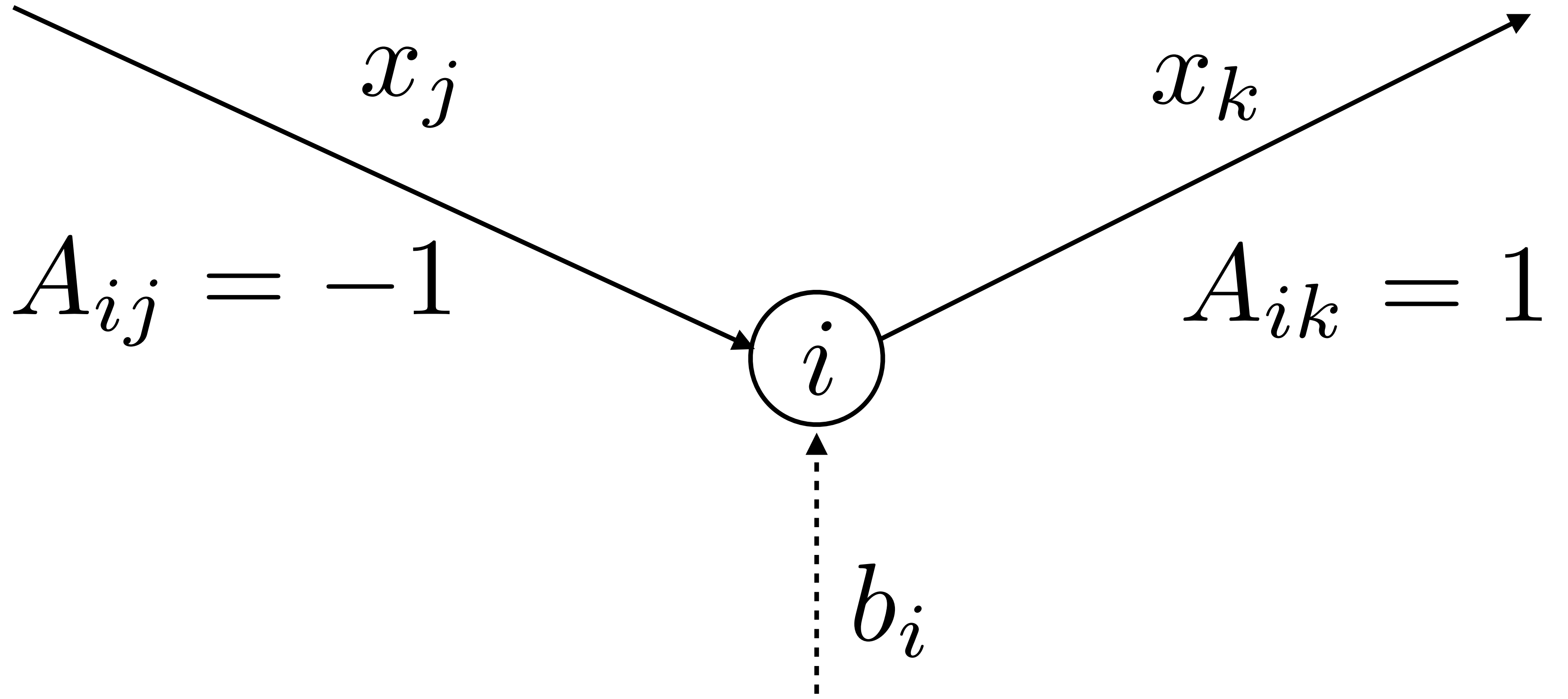
Note Each column has one -1 and one 1

$$A = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & -1 & -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix}$$

External supply

supply vector $b \in \mathbf{R}^m$

- b_i is the external supply at node i (if $b_i < 0$, it represents demand)
- We must have $\mathbf{1}^T b = 0$ (total supply = total demand)



Balance equations

$$\sum_{j=1}^{n} A_{ij} x_j = (Ax)_i = b_i, \quad \text{for all } i \quad \longrightarrow \quad Ax = b$$

$(Ax)_i$: Total leaving flow

b_i: Supply

Minimum cost network flow problem

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & 0 \leq x \leq u \end{array}$$

- c_i is unit cost of flow through arc i
- Flow x_i must be nonnegative
- u_i is the maximum flow capacity of arc i
- Many network optimization problems are just special cases

Today's lecture
Interior point methods

- History
- Newton's method
- Central path
- Primal-dual path-following algorithm
- Logarithmic barrier functions

History

A brief history of linear optimization

1940s :

- Foundations and applications in economics and logistics (Kantorovich, Koopmans)
- **1947** : Development of the **simplex method** by Dantzig

1950s—70s:

- Applications expand to engineering, OR, computer science…
- **1975** : Nobel prize in economics for Kantorovich and Koopmans

1980s:

- Development of polynomial time algorithms for LPs
- **1984** : Development of the **interior point method** by Karmarkar

—Today:

- Continued algorithm development. Expansion to very large problems.

Ellipsoid method

Khachian (1979)

Answer to major question

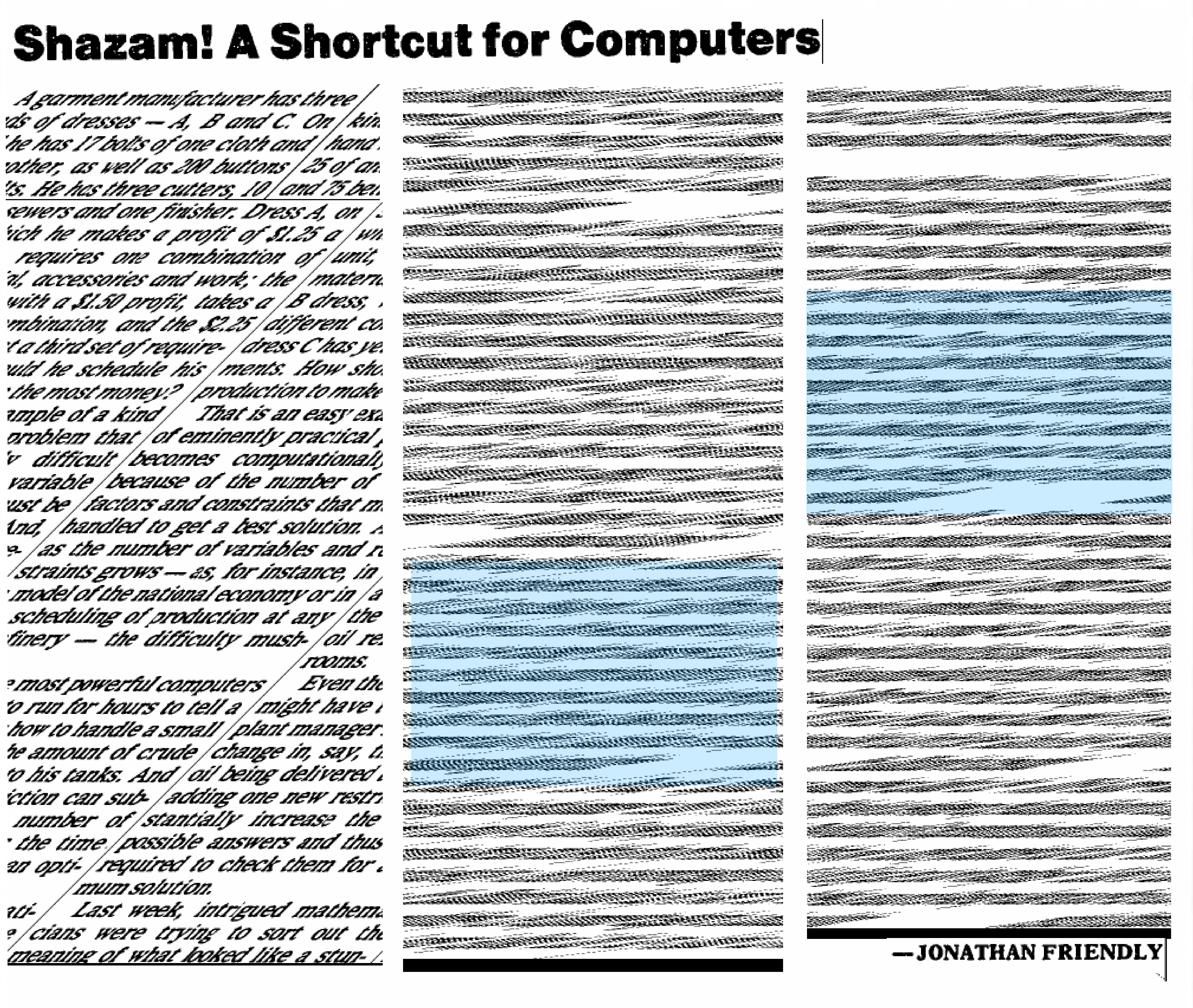
Is worst-case LP complexity polynomial? **Yes!**

Drawbacks

Very inefficient. Much slower than simplex!

Benefits

Motivated new research directions


Shazam! A Shortcut for Computers

—JONATHAN FRIENDLY

The New York Times
Published: November 11, 1979

Interior-point methods

1980s-1990s: interior point methods

- Karmarkar's algorithm (1984)
- Competitive with simplex, often faster for larger problems
- Began huge effort in algorithm development for convex optimization

Breakthrough in Problem Solving

By JAMES GLEICK

A 28-year-old mathematician at A.T.&T. Bell Laboratories has made a startling theoretical breakthrough in the solving of systems of equations that often grow too vast and complex for the most powerful computers.

The discovery, which is to be formally published next month, is already circulating rapidly through the mathematical world. It has also set off a deluge of inquiries from brokerage houses, oil companies and airlines, industries with millions of dollars at stake in problems known as linear programming.

These problems are fiendishly complicated systems, often with thousands of variables. They arise in a variety of commercial and government applications, ranging from allocating time on a communications satellite to routing millions of telephone calls over long distances. Linear programming is particularly useful whenever a limited, expensive resource must be spread most efficiently among competing users. And investment companies use the approach in creating portfolios with the best mix of stocks and bonds.

Faster Solutions Seen

The Bell Labs mathematician, Dr. Narendra Karmarkar, has devised a radically new procedure that may speed the routine handling of such problems by businesses and Government agencies and also make it possible to tackle problems that are now far out of reach.

"This is a path-breaking result," said Dr. Ronald L. Graham, director of mathematical sciences for Bell Labs in Murray Hill, N.J. "Science has its moments of great progress, and this may well be one of them."

Because problems in linear programming can have billions or more possible answers, even high-speed computers cannot check every one. So computers must use a special procedure, an algorithm, to examine as few answers as possible before finding the best one — typically the one that minimizes cost or maximizes efficiency.

A procedure devised in 1947, the simplex method, is now used for such prob-

Continued on Page A19, Column 1

Newton’s method

Newton's root finding method

Goal: solve

$$h(x) = 0$$

Method

1. Make a guess x^k and a linear approximation

$$h(x) \approx h(x^k) + \frac{\partial h}{\partial x^k}(x^k)(x - x^k)$$

2. Iteratively set $h(x^k)$ to 0

$$h(x^k) + \frac{\partial h}{\partial x^k}(x^k)(x^{k+1} - x^k) = 0$$

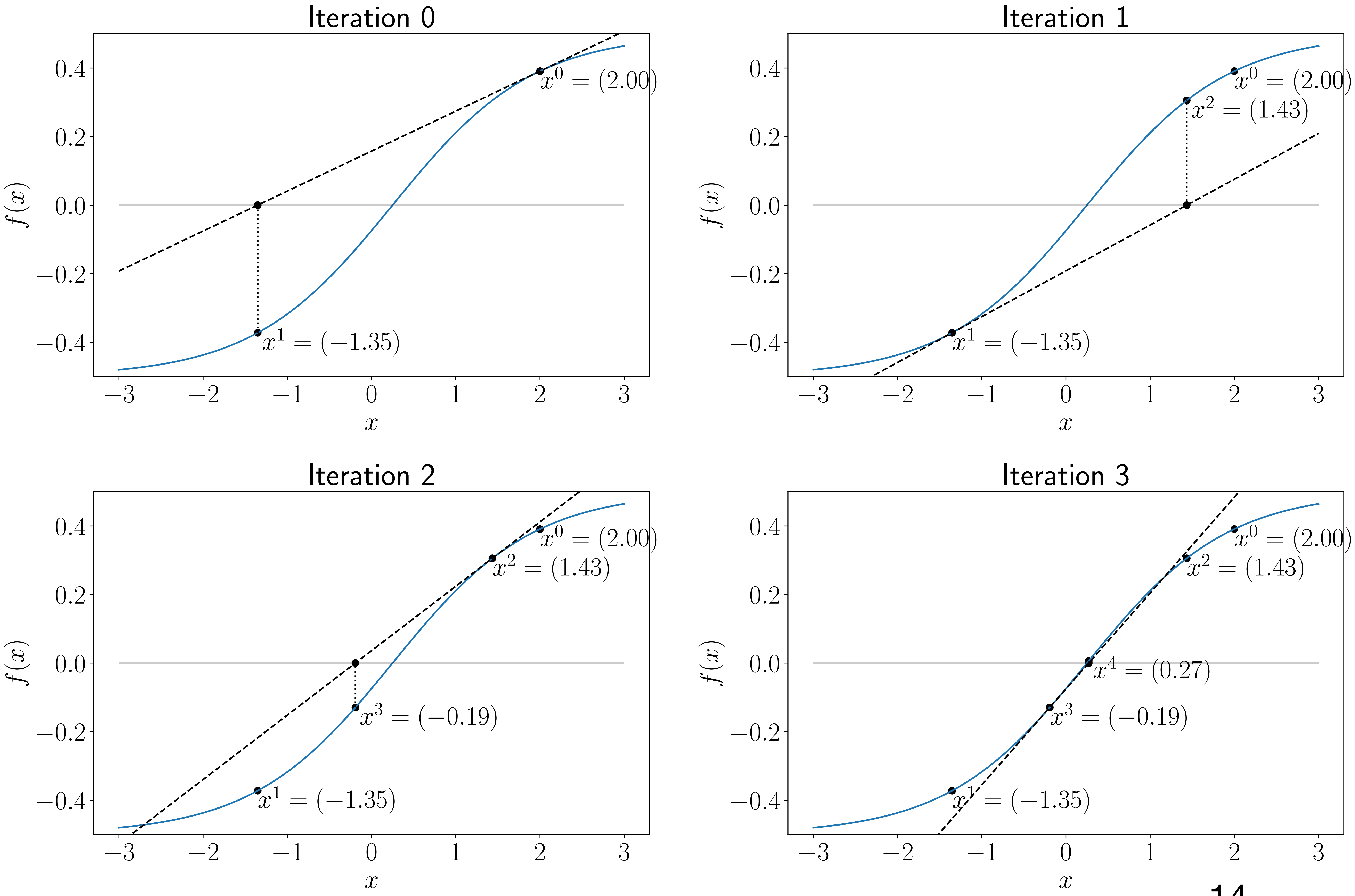
Newton's method example

$$f(x) = \frac{1}{1 + e^{-1.2x+0.3}} - 0.5$$

$$f(x) = 0$$

$$\downarrow$$

$$x^\star = 0.3$$



Newton’s root finding method (multivariable)

Goal: solve
$$h(x) = 0$$

Method

1. Make a guess x^k and a linear approximation

$$h(x) \approx h(x^k) + Dh(x^k)(x - x^k)$$

2. Iteratively set $h(x^k)$ to 0

$$h(x^k) + Dh(x^k)(x^{k+1} - x^k) = 0$$

Derivative

$$Dh = \begin{bmatrix} \frac{\partial h_1}{\partial x_1} & \cdots & \frac{\partial h_1}{\partial x_n} \\ \vdots & \vdots & \vdots \\ \frac{\partial h_m}{\partial x_1} & \cdots & \frac{\partial h_m}{\partial x_n} \end{bmatrix}$$

Newton method iterations

$$h(x^k) + Dh(x^k)\underbrace{(x^{k+1} - x^k)}_{\Delta x} = 0$$

Iterations

- Solve $Dh(x^k)\Delta x = -h(x^k)$
- $x^{k+1} \leftarrow x^k + \Delta x$

Remarks

- Iterations can be **expensive** (linear system solution)
- **Fast convergence** close to the solution $x^\star$

Linear optimization as a root finding problem

Optimality conditions

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax \leq b \end{array} \longrightarrow$$

Primal

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax + s = b \\ & s \geq 0 \end{array}$$

Dual

$$\begin{array}{ll} \text{maximize} & -b^T y \\ \text{subject to} & A^T y + c = 0 \\ & y \geq 0 \end{array}$$

KKT conditions

$$\begin{aligned} Ax + s - b &= 0 \\ A^T y + c &= 0 \\ s_i y_i &= 0, \quad i = 1, \dots, m \\ s, y &\geq 0 \end{aligned}$$

Linear optimization as a root finding problem

$$
\begin{aligned}
Ax + s - b &= 0 \\
A^T y + c &= 0 \\
s_i y_i &= 0, \quad i = 1, \dots, m \\
s, y &\geq 0
\end{aligned}
$$

Diagonalize complementary slackness

$$
S = \mathbf{diag}(s) = \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & \ddots & \\ & & & s_m \end{bmatrix}
$$

$$
Y = \mathbf{diag}(y) = \begin{bmatrix} y_1 & & & \\ & y_2 & & \\ & & \ddots & \\ & & & y_m \end{bmatrix}
$$

$$
SY\mathbf{1} = \mathbf{diag}(s)\mathbf{diag}(y)\mathbf{1} = \begin{bmatrix} s_1 y_1 & & & \\ & s_2 y_2 & & \\ & & \ddots & \\ & & & s_m y_m \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} = \begin{bmatrix} s_1 y_1 \\ s_2 y_2 \\ \vdots \\ s_m y_m \end{bmatrix}
$$

$$
\downarrow
$$

$$
s_i y_i = 0, \quad i = 1, \dots, m \quad \Longleftrightarrow \quad SY\mathbf{1} = 0
$$

Main idea

Optimality conditions

$$h(y, x, s) = \begin{bmatrix} Ax + s - b \\ A^T y + c \\ SY\mathbf{1} \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ SY\mathbf{1} \end{bmatrix} = 0$$

$$s, y \geq 0$$

$$S = \mathbf{diag}(s)$$
$$Y = \mathbf{diag}(y)$$

- Apply variants of Newton's method to solve $h(x, s, y) = 0$
- Enforce $s, y > 0$ (strictly) at every iteration
- **Motivation** avoid getting stuck in "corners"

Newton's method for optimality conditions

Optimality conditions

$$h(y, x, s) = \begin{bmatrix} Ax + s - b \\ A^T y + c \\ SY\mathbf{1} \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ SY\mathbf{1} \end{bmatrix} = 0$$

$$s, y \geq 0$$

Derivative

$$Dh(y, x, s) = \begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix}$$

Iterations

- Solve $Dh(y^k, x^k, s^k)\Delta(y^k, x^k, s^k) = -h(y^k, x^k, s^k)$
- $\begin{bmatrix} y^{k+1} \\ x^{k+1} \\ s^{k+1} \end{bmatrix} \leftarrow \begin{bmatrix} y^k \\ x^k \\ s^k \end{bmatrix} + \Delta(y^k, x^k, s^k)$

Caution!

It might make (s, y) negative!

Central path

Line search to stay feasible

Root-finding equation

$$h(y, x, s) = \begin{bmatrix} Ax + s - b \\ A^T y + c \\ SY\mathbf{1} \end{bmatrix} = \begin{bmatrix} r_p \\ r_d \\ SY\mathbf{1} \end{bmatrix} = 0$$

Linear system

$$\overbrace{\begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix}}^{Dh} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta s \end{bmatrix} = \overbrace{\begin{bmatrix} -r_p \\ -r_d \\ -SY\mathbf{1} \end{bmatrix}}^{-h}$$

Residuals

$$r_p = Ax + s - b$$

$$r_d = A^T y + c$$

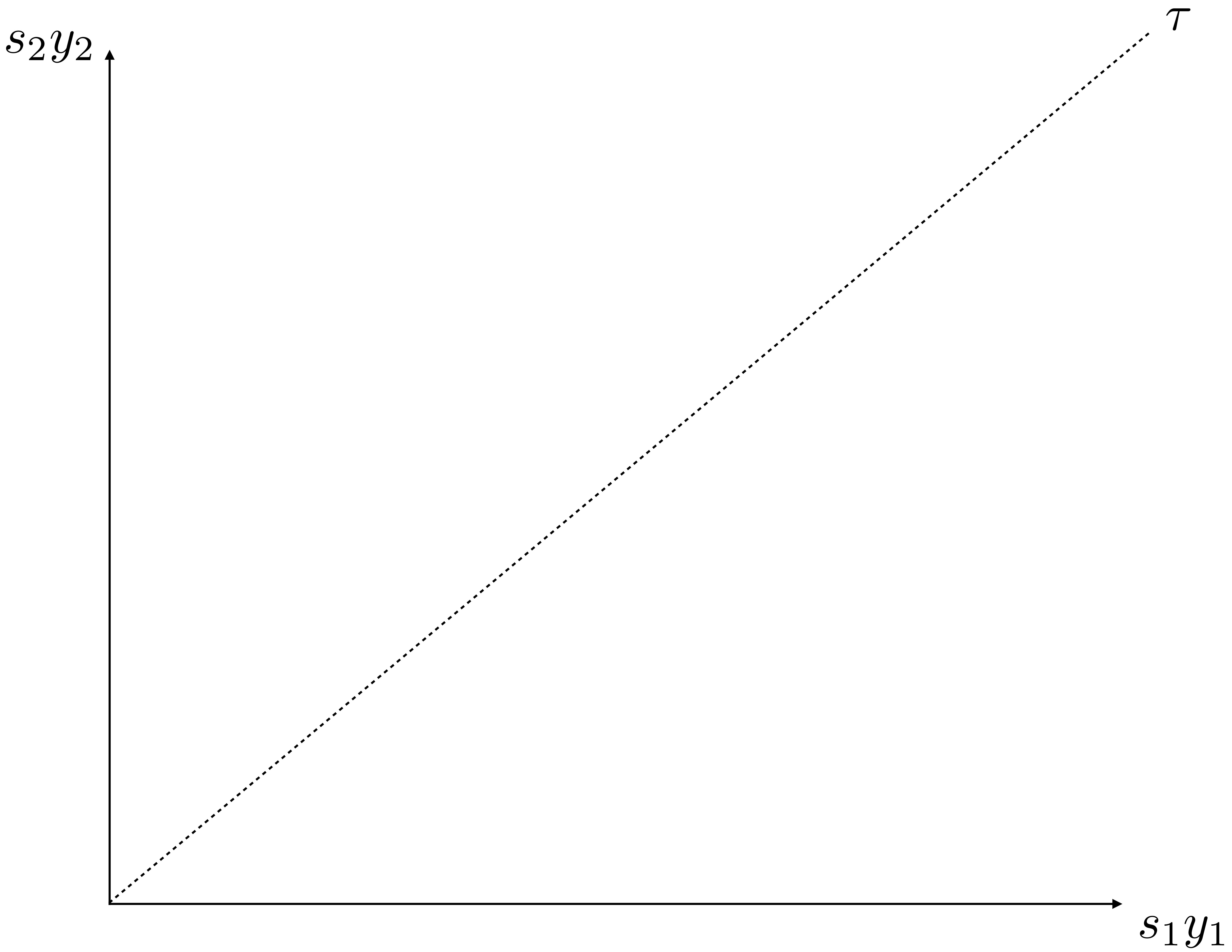
Issue

Pure **Newton's step** does not allow significant progress towards $h(y, x, s) = 0$ **and** $s, y \geq 0$.

Line search to enforce $s, y > 0$

$$(y, x, s) \leftarrow (y, x, s) + \alpha(\Delta y, \Delta x, \Delta s)$$

The central path



Smoothed optimality conditions

Optimality conditions

$$Ax + s - b = 0$$

$$A^T y + c = 0$$

$$s_i y_i = \tau$$

$$s, y \geq 0$$

← Same τ for every pair

Same optimality conditions for a "smoothed" version of our problem

Duality gap

$$s^T y = (b - Ax)^T y = b^T x - x^T A^T y = b^T y + c^T x$$

Newton's method for smoothed optimality conditions

Smoothed optimality conditions

$$h_\tau(y, x, s) = \begin{bmatrix} Ax + s - b \\ A^T y + c \\ SY\mathbf{1} - \tau\mathbf{1} \end{bmatrix} = 0$$

$$s, y \geq 0$$

Linear system

$$\begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_p \\ -r_d \\ -SY + \tau\mathbf{1} \end{bmatrix}$$

Line search to enforce $s, y > 0$

$$(y, x, s) \leftarrow (y, x, s) + \alpha(\Delta y, \Delta x, \Delta s)$$

The path parameter

Linear system

$$\begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_p \\ -r_d \\ -SY\mathbf{1} + \sigma\mu\mathbf{1} \end{bmatrix}$$

Duality measure

$$\mu = \frac{s^T y}{m}$$

(average value of the pairs $s_i y_i$)

Centering parameter

$$\sigma \in [0, 1]$$

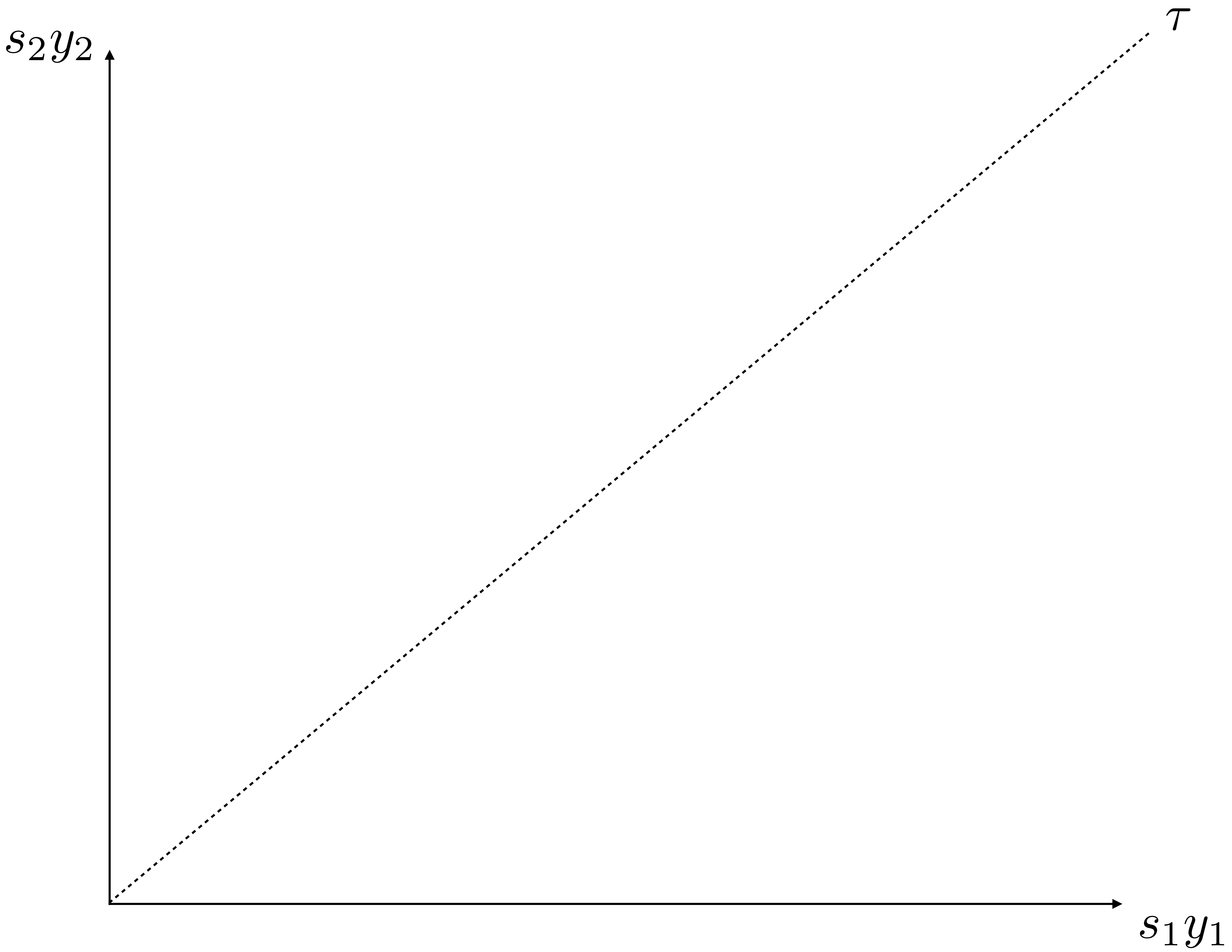
$\sigma = 0 \quad \Rightarrow$ Newton step

$\sigma = 1 \quad \Rightarrow$ Centering step towards $(y^\star(\mu), x^\star(\mu), s^\star(\mu))$

Line search to enforce $s, y > 0$

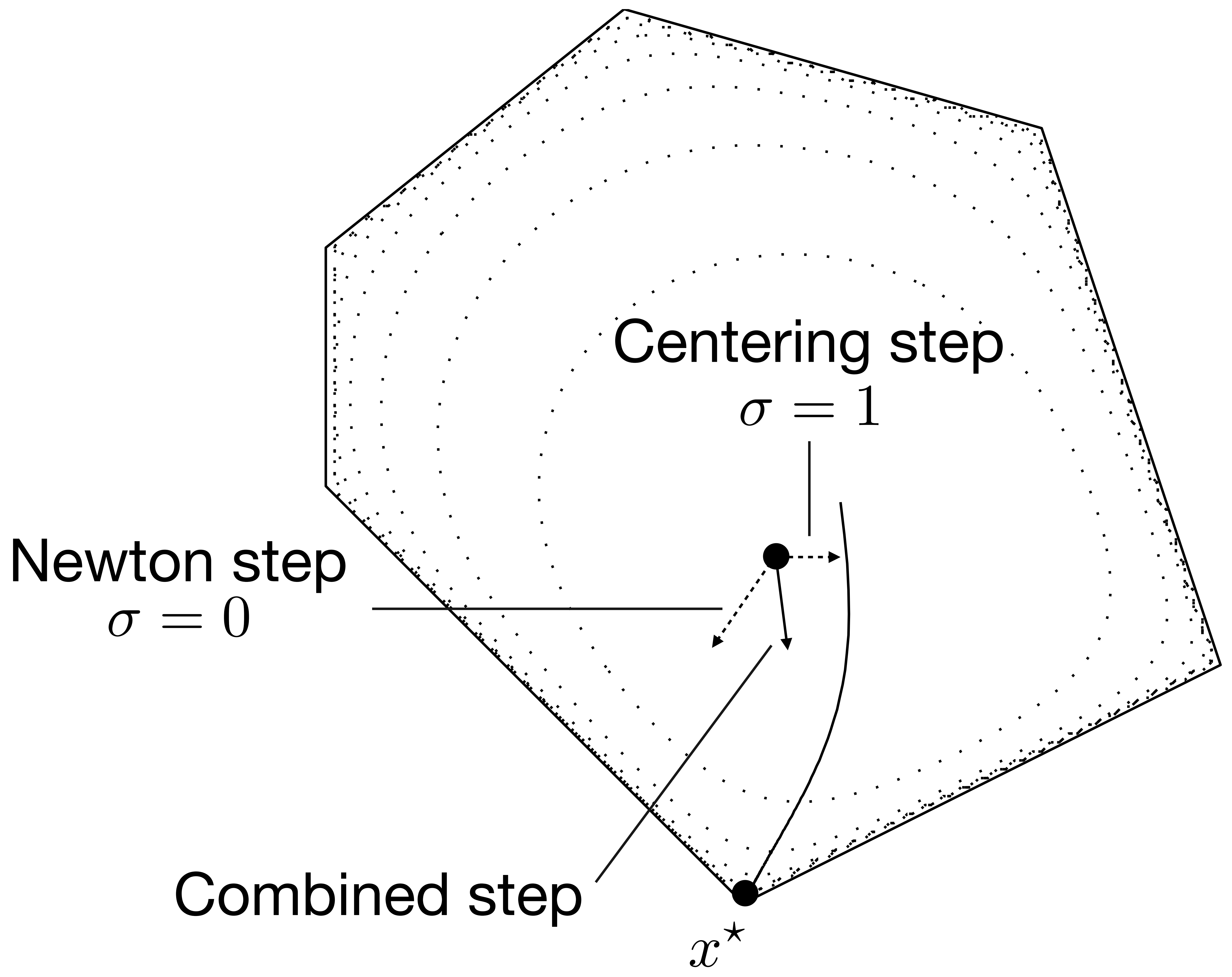
$$(y, x, s) \leftarrow (y, x, s) + \alpha(\Delta y, \Delta x, \Delta s)$$

The central path



Primal-dual path-following method

Path-following algorithm idea



Centering step
It brings towards the **central path** and is usually biased towards $s, y > 0$.
No progress on duality measure μ

Newton step
It brings towards the **zero duality measure** μ. Quickly violates $s, y > 0$.

Combined step
Best of both worlds with longer steps

Primal-dual path-following algorithm

Initialization

1. Given (x_0, s_0, y_0) such that $s_0, y_0 > 0$

Iterations

1. Choose $\sigma \in [0, 1]$

2. Solve $\begin{bmatrix} 0 & A & I \\ A^T & 0 & 0 \\ S & 0 & Y \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta x \\ \Delta s \end{bmatrix} = \begin{bmatrix} -r_p \\ -r_d \\ -SY\mathbf{1} + \sigma\mu\mathbf{1} \end{bmatrix}$ where $\mu = s^T y / m$

3. Find maximum α such that $y + \alpha\Delta y > 0$ and $s + \alpha\Delta s > 0$

4. Update $(y, x, s) \leftarrow (y, x, s) + \alpha(\Delta y, \Delta x, \Delta s)$

Working towards optimality conditions

Optimality conditions satisfied **only at convergence**

Primal residual

$$r_p = Ax + s - b \to 0$$

Dual residual

$$r_d = A^T y + c \to 0$$

Complementary slackness

$$s^T y \to 0$$

Stopping criteria

$$\|r_p\| \leq \epsilon_{\text{pri}}$$

$$\|r_d\| \leq \epsilon_{\text{dua}}$$

$$s^T y \leq \epsilon_{\text{gap}}$$

Logarithmic barrier functions

Smoothed optimality conditions

Optimality conditions

$$
\begin{aligned}
Ax + s - b &= 0\\
A^T y + c &= 0\\
s_i y_i &= \tau\\
s, y &\geq 0
\end{aligned}
$$

Same τ for every pair

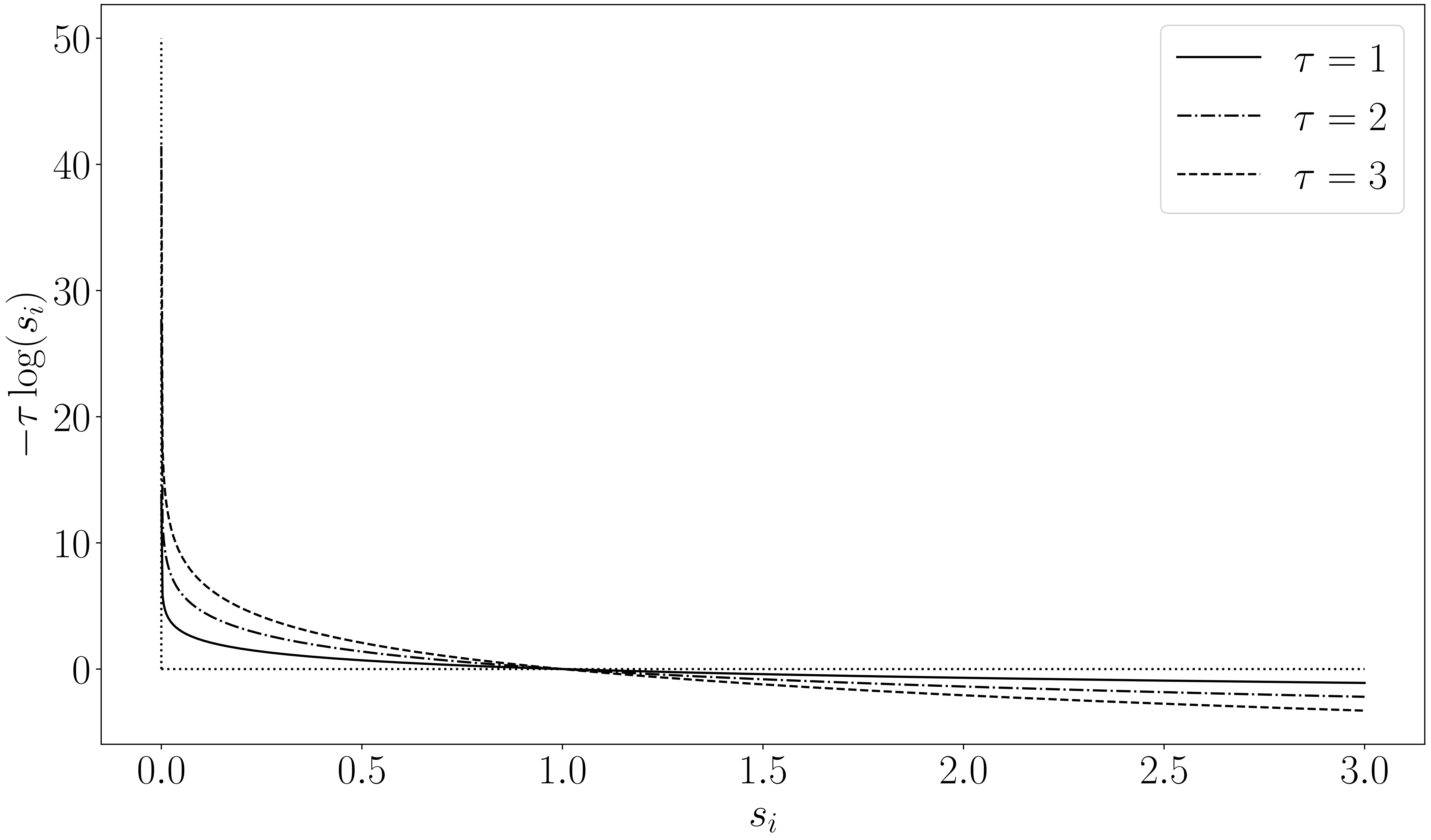
Same optimality conditions for a "smoothed" version of our problem

Do solutions actually exist?

What do they represent?

Logarithmic barrier

$\phi(s) = -\tau \sum_{i=1}^{m} \log(s_i)$ on domain $s_i > 0$



As $\tau \to 0$ it approximates

$$\mathcal{I}_{s_i \geq 0} = \begin{cases} 0 & \text{if } s_i \geq 0 \\ \infty & \text{otherwise} \end{cases}$$

Smoothed problem

minimize $c^T x$
subject to $Ax + s = b$
$s \geq 0$

$\longrightarrow$

minimize $c^T x + \phi(s) = c^T x - \tau \sum_{i=1}^m \log(s_i)$
subject to $Ax + s = b$

Lagrangian function

$$L(x, s, y) = c^T x - \tau \sum_{i=1}^{m} \log(s_i) + y^T (Ax + s - b)$$

$$\frac{\partial L}{\partial x} = A^T y + c = 0$$

$$\frac{\partial L}{\partial s_i} = -\tau \frac{1}{s_i} + y_i = 0 \quad \Longrightarrow s_i y_i = \tau$$

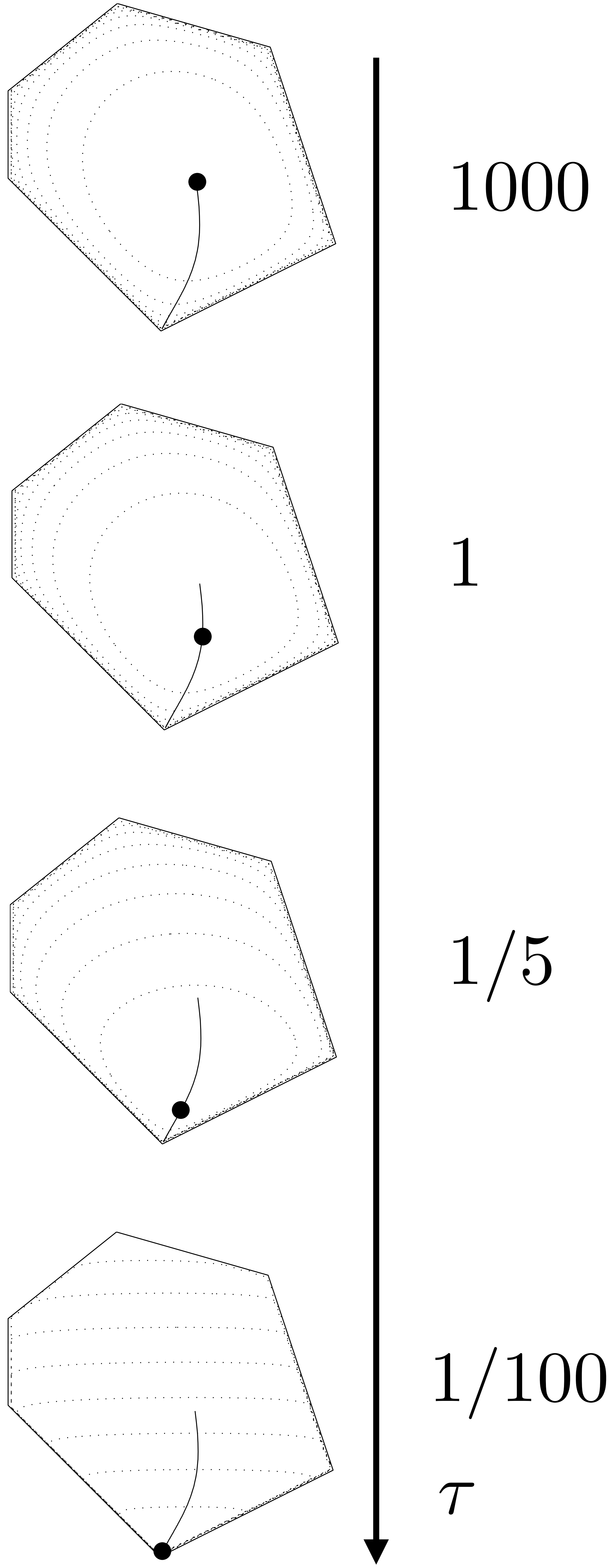
Central path

Analytic Center
$\tau \to \infty$

minimize $\quad c^T x - \tau \sum_{i=1}^m \log(s_i)$
subject to $\quad Ax + s = b$

Set of points $(x^\star(\tau), s^\star(\tau), y^\star(\tau))$
with $\tau > 0$ such that

$$
\begin{aligned}
Ax + s - b &= 0\\
A^T y + c &= 0\\
s_i y_i &= \tau\\
s, y &\geq 0
\end{aligned}
$$



Main idea

Follow central path as $\tau \to 0$

Interior-point methods for linear optimization

Today, we learned to:

- **Apply** Newton’s method to solve optimality conditions
- **Follow** the central path and the smoothed optimality conditions
- **Use logarithmic barrier functions** to interpret central path steps

References

- D. Bertsimas and J. Tsitsiklis: Introduction to Linear Optimization
    - Chapter 9.4 — 9.6: Interior point methods

- R. Vanderbei: Linear Programming
    - Chapter 17: The Central Path
    - Chapter 15: A Path-Following Method

Next lecture

- Practical interior-point method (Mehrotra predictor-corrector algorithm)
- Implementation details
- Interior-point vs simples