Efficient First-Order Methods for Linear (and Semidefinite and ~~Hyperbolic~~) Programming

not enough time!

Jim Renegar

(paper recently appeared on ArXiv)

Simons Institute
Workshop on Semidefinite Optimization,
Approximation and Applications

Background –

An "Optimal" Subgradient Method for Unconstrained Optimization:

$\max\ f(x)$ where $f : \mathbb{R}^n \to \mathbb{R}$ is concave and Lipschitz continuous:

$$|f(x) - f(y)| \leq \underset{\text{Lipschitz constant}}{\kappa_0} \|x - y\|$$

Inputs:

- Initial iterate: x_0
- Number of iterations: N
- Distance upper bound: R, a value for which there exists optimal x^* satisfying $\|x_0 - x^*\| \leq R$.

Iteration: For current iterate x_k, compute a subgradient $\nabla f(x_k)$, then let

$$x_{k+1} = x_k + \frac{R}{\sqrt{N}\,\|\nabla f(x_k)\|}\nabla f(x_k)$$

Theorem (from Nesterov's book): $f(x^*) - f(x_N) \leq \dfrac{\kappa_0 R}{\sqrt{N}}$

Corollary: $N \geq \left(\dfrac{\kappa_0 R}{\epsilon}\right)^2 \Rightarrow f(x^*) - f(x_N) \leq \epsilon$

$\max\ f(x)$

Theorem (from Nesterov's book): $f(x^*) - f(x_N) \le \dfrac{\kappa_0 R}{\sqrt{N}}$

Corollary: $N \ge \left(\dfrac{\kappa_0 R}{\epsilon}\right)^2 \quad \Rightarrow \quad f(x^*) - f(x_N) \le \epsilon$

Subgradient Method for Constrained Optimization:

$$\begin{array}{rl} \max & f(x) \\ \text{s.t.} & x \in Q \end{array}$$

closed, convex set

Iteration: For current iterate x_k, compute x_{k+1} as above,
but if $x_{k+1} \notin Q$,
replace x_{k+1} by its orthogonal projection onto Q.

Clearly, this can be done efficiently only if Q is a simple set, such as an affine space or, say, an ℓ_∞ unit ball.

$\max\ f(x)$

Theorem (from Nesterov's book): $f(x^*) - f(x_N) \leq \dfrac{\kappa_0\, R}{\sqrt{N}}$

Corollary: $N \geq \left(\dfrac{\kappa_0\, R}{\epsilon}\right)^2 \quad \Rightarrow \quad f(x^*) - f(x_N) \leq \epsilon$

Subgradient Method for Linearly Constrained Optimization:

$$\begin{array}{rl} \max & f(x) \\ \text{s.t.} & Ax = b \end{array}$$

Iteration: For current iterate x_k, compute a subgradient $\nabla f(x_k)$ and its orthogonal projection G_k onto the nullspace of A, then let

$$x_{k+1} = x_k + \frac{R}{\sqrt{N}\,\|G_k\|}\, G_k$$

Recurring theme in literature on first-order methods:

*Pack as much as possible** *into the objective function,*
leaving only simple constraints
– *ideally leaving no constraints or only linear equations.*

*— subject to keeping a "nice" objective function

Example that is not "nice":

$$i(x) := \begin{cases} 0 & \text{if } x \geq 0 \\ \infty & \text{otherwise} \end{cases}$$

$$\begin{array}{rl} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array} \quad \longrightarrow \quad \begin{array}{rl} \min & c^T x + i(x) \\ \text{s.t.} & Ax = b \end{array}$$

The objective is convex,
but not Lipschitz continuous
on a neighborhood of $\{x : Ax = b\}$.

Background –

Regarding Nesterov's Optimal Method for Smooth Functions:

$\max f(x)$ where $f : \mathbb{R}^n \to \mathbb{R}$ is concave and $\nabla f(x)$ is Lipschitz continuous:

$$\| \nabla f(x) - \nabla f(y) \| \leq \underset{\text{Lipschitz constant}}{\underset{\uparrow}{\kappa_1}} \, \| x - y \|$$

Inputs:

- Initial iterate: x_0
- Lipschitz constant (or upper bound): κ_1

Theorem (Nesterov): The sequence generated by the algorithm satisfies

$$f(x^*) - f(x_k) \leq \kappa_1 \left(\frac{2\,R}{k+2} \right)^2 \qquad \text{where } R = \| x_0 - x^* \|$$

Corollary: $$k \geq 2\,R \sqrt{\frac{\kappa_1}{\epsilon}} \quad \Rightarrow \quad f(x^*) - f(x_k) \leq \epsilon$$

Compare with subgradient method result:

$$N \geq \left(\frac{\kappa_0\, R}{\epsilon} \right)^2 \quad \Rightarrow \quad f(x^*) - f(x_N) \leq \epsilon$$

Background –

Regarding Nesterov's Optimal Method for Smooth Functions:

$\max\ f(x)$

where $f : \mathbb{R}^n \to \mathbb{R}$ is concave
and $\nabla f(x)$ is Lipschitz continuous:

$$\|\nabla f(x) - \nabla f(y)\| \leq \kappa_1 \|x - y\|$$

Lipschitz constant

Inputs:

- Initial iterate: x_0
- Lipschitz constant (or upper bound): κ_1

Theorem (Nesterov): The sequence generated by the algorithm satisfies

$$f(x^*) - f(x_k) \leq \kappa_1 \left(\frac{2\,R}{k+2} \right)^2 \qquad \text{where } R = \|x_0 - x^*\|$$

Corollary: $$k \geq 2\,R \sqrt{\frac{\kappa_1}{\epsilon}} \quad \Rightarrow \quad f(x^*) - f(x_k) \leq \epsilon$$

Result can be extended to <u>very special</u> constrained optimization problems,
e.g., ones with tractable "prox" functions:

$$\begin{array}{rl} \max & f(x) \\ \text{s.t.} & \sum_j x_j = 1 \\ & x \geq 0 \end{array} \qquad\qquad \begin{array}{rl} \max & f(X) \\ \text{s.t.} & \mathbf{tr}(X) = 1 \\ & x \succeq 0 \end{array}$$

The great advances in algorithms for solving huge optimization problems
in some high-profile areas (e.g., compressed sensing)
are based on approaches related to this.

Regarding Nesterov's Optimal Method for Smooth Functions:

$\max\ f(x)$

where $f : \mathbb{R}^n \to \mathbb{R}$ is concave and $\nabla f(x)$ is Lipschitz continuous:

$$\|\nabla f(x) - \nabla f(y)\| \leq \underset{\text{Lipschitz constant}}{\underset{\uparrow}{\kappa_1}} \|x - y\|$$

Inputs:

- Initial iterate: x_0
- Lipschitz constant (or upper bound): κ_1

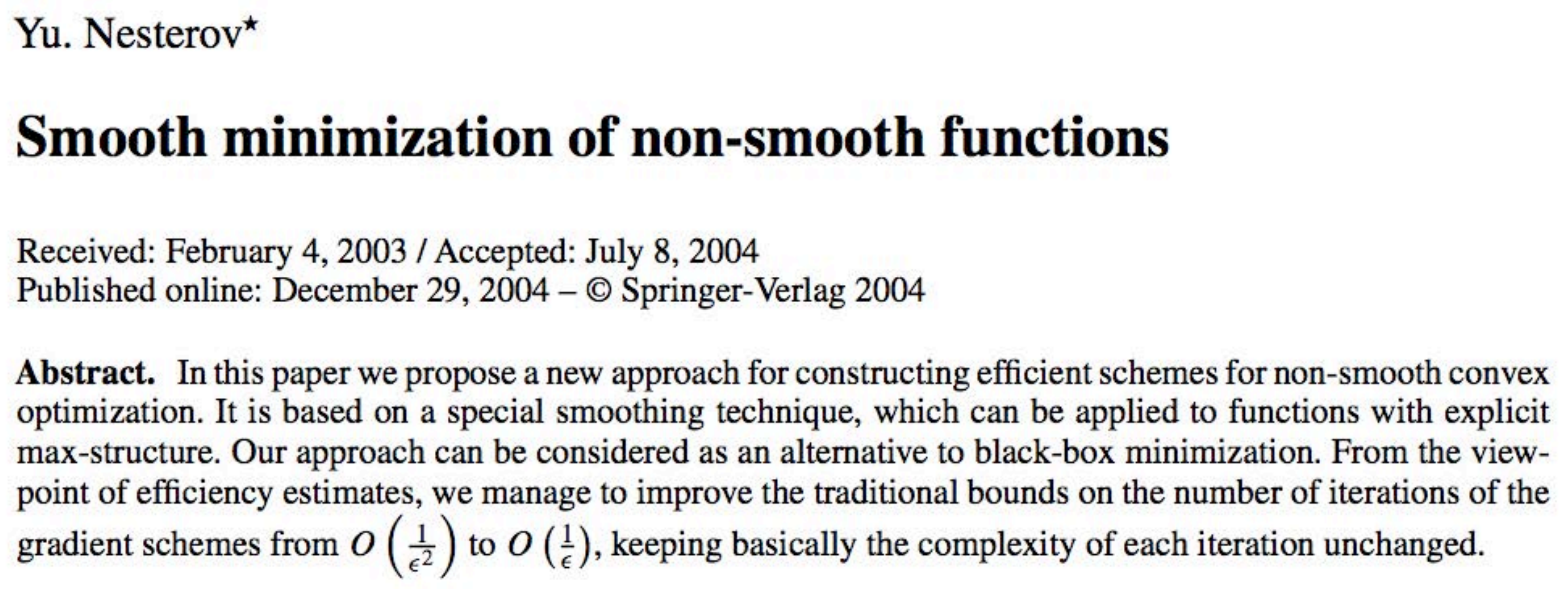
Theorem (Nesterov): The sequence generated by the algorithm satisfies

$$f(x^*) - f(x_k) \leq \kappa_1 \left(\frac{2\,R}{k+2} \right)^2 \qquad \text{where } R = \|x_0 - x^*\|$$

Corollary: $k \geq 2\,R\sqrt{\dfrac{\kappa_1}{\epsilon}} \quad \Rightarrow \quad f(x^*) - f(x_k) \leq \epsilon$

But cannot be extended even to $\begin{array}{ll} \max & f(x) \\ \text{s.t.} & -1 \leq x_j \leq 1 \quad (j = 1, \ldots, n) \end{array}$

Guzmán and Nemirovski (2013): $\Omega\left(\dfrac{\kappa_1}{\epsilon \ln n} \right)$



Yu. Nesterov*

Smooth minimization of non-smooth functions

Received: February 4, 2003 / Accepted: July 8, 2004
Published online: December 29, 2004 – © Springer-Verlag 2004

Abstract. In this paper we propose a new approach for constructing efficient schemes for non-smooth convex optimization. It is based on a special smoothing technique, which can be applied to functions with explicit max-structure. Our approach can be considered as an alternative to black-box minimization. From the viewpoint of efficiency estimates, we manage to improve the traditional bounds on the number of iterations of the gradient schemes from $O\left(\frac{1}{\epsilon^2}\right)$ to $O\left(\frac{1}{\epsilon}\right)$, keeping basically the complexity of each iteration unchanged.

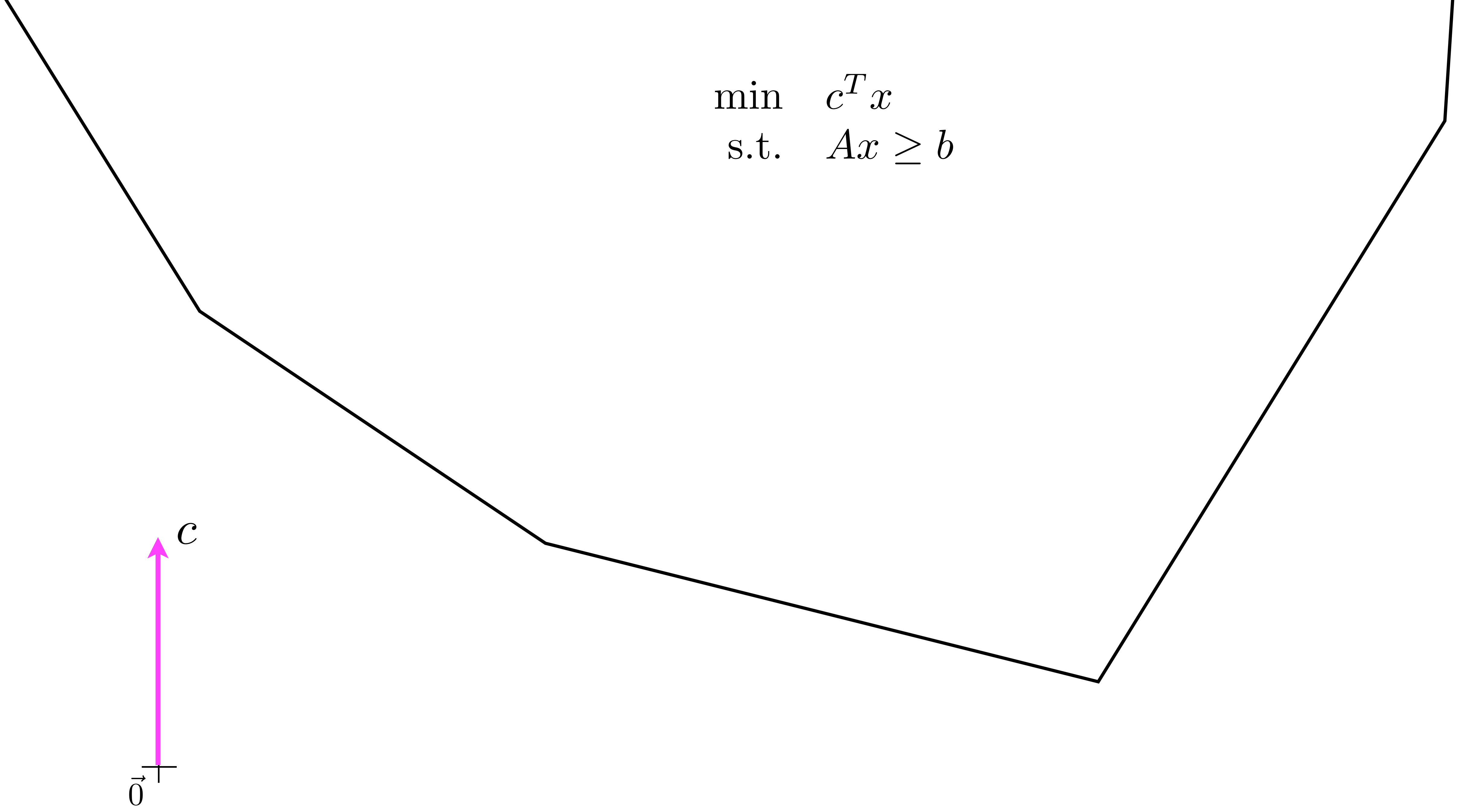
Example of a problem that is "smoothed" in the paper:

$$\min_{x} \max_{1 \le i \le m} |\alpha_i^T x + b_i| \quad \text{s.t. } x \in Q$$

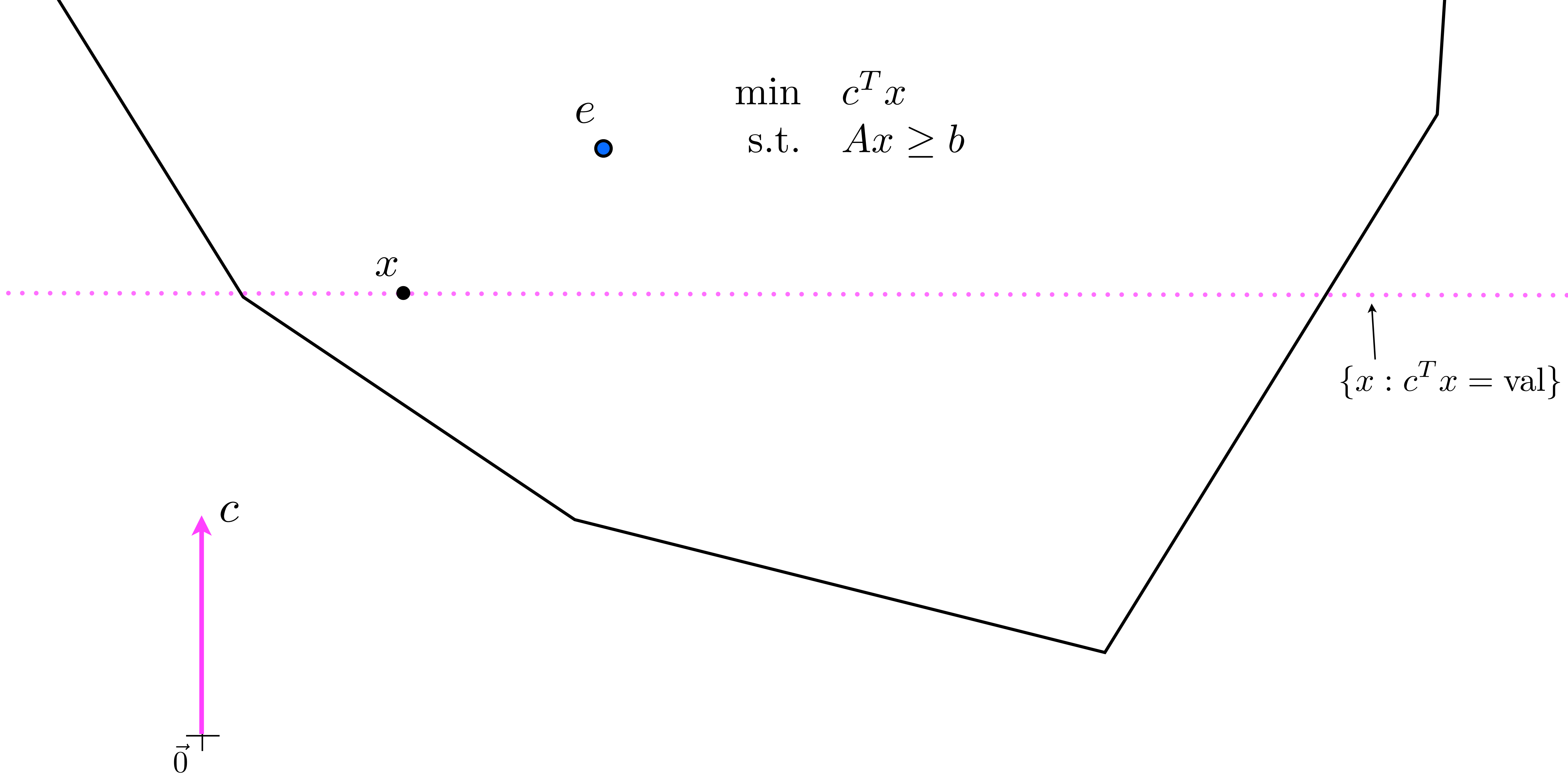
where Q is a closed, bounded, convex set
over which it is easy to optimize "nice" functions (e.g., a unit ball).

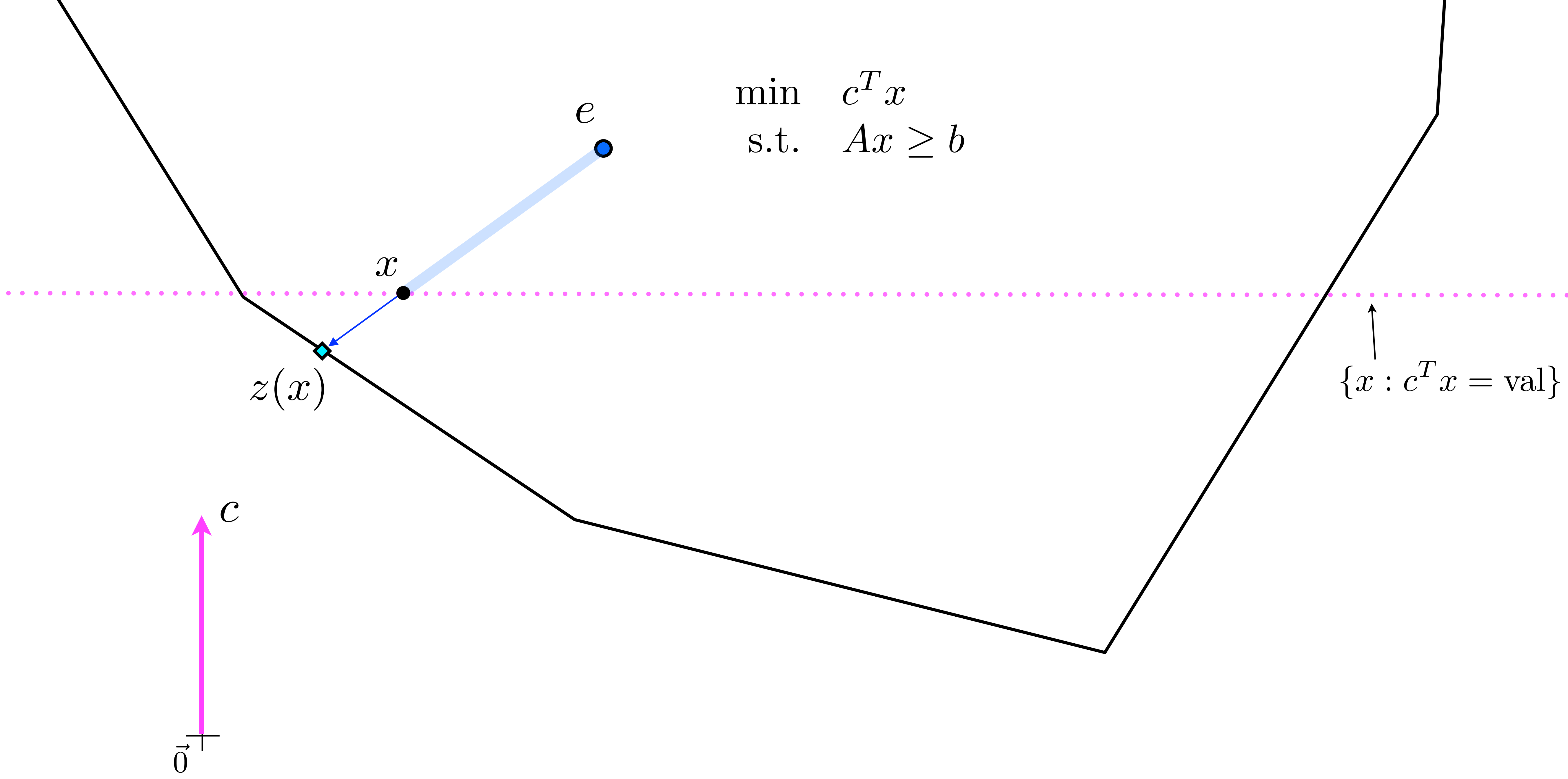
The approach is ingenious,
but one is left wondering,
isn't there a simpler and more general way
of obtaining the desired $O(1/\epsilon)$ iteration bound?

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \end{aligned}$$



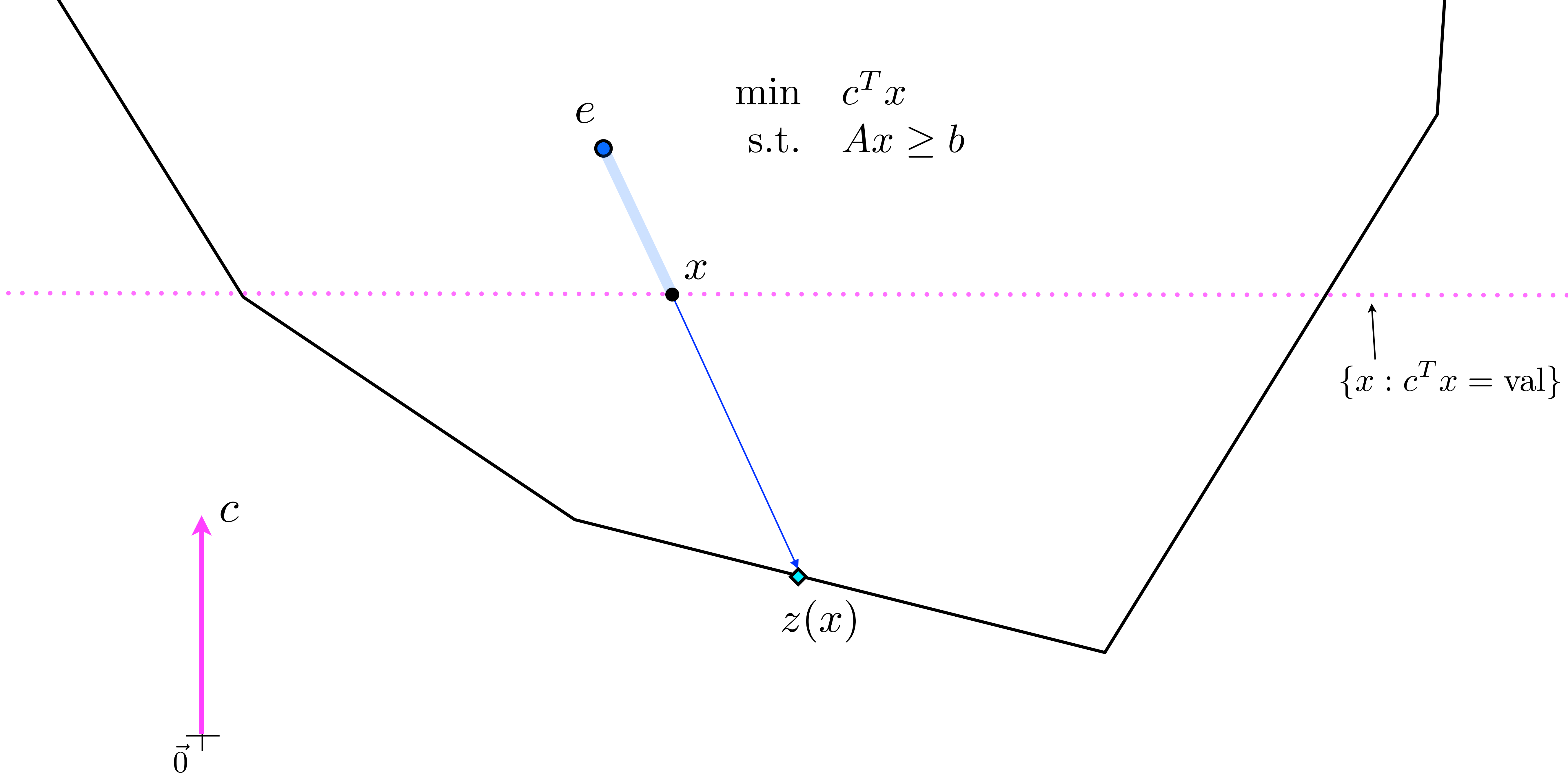
$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \geq b \end{array}$$

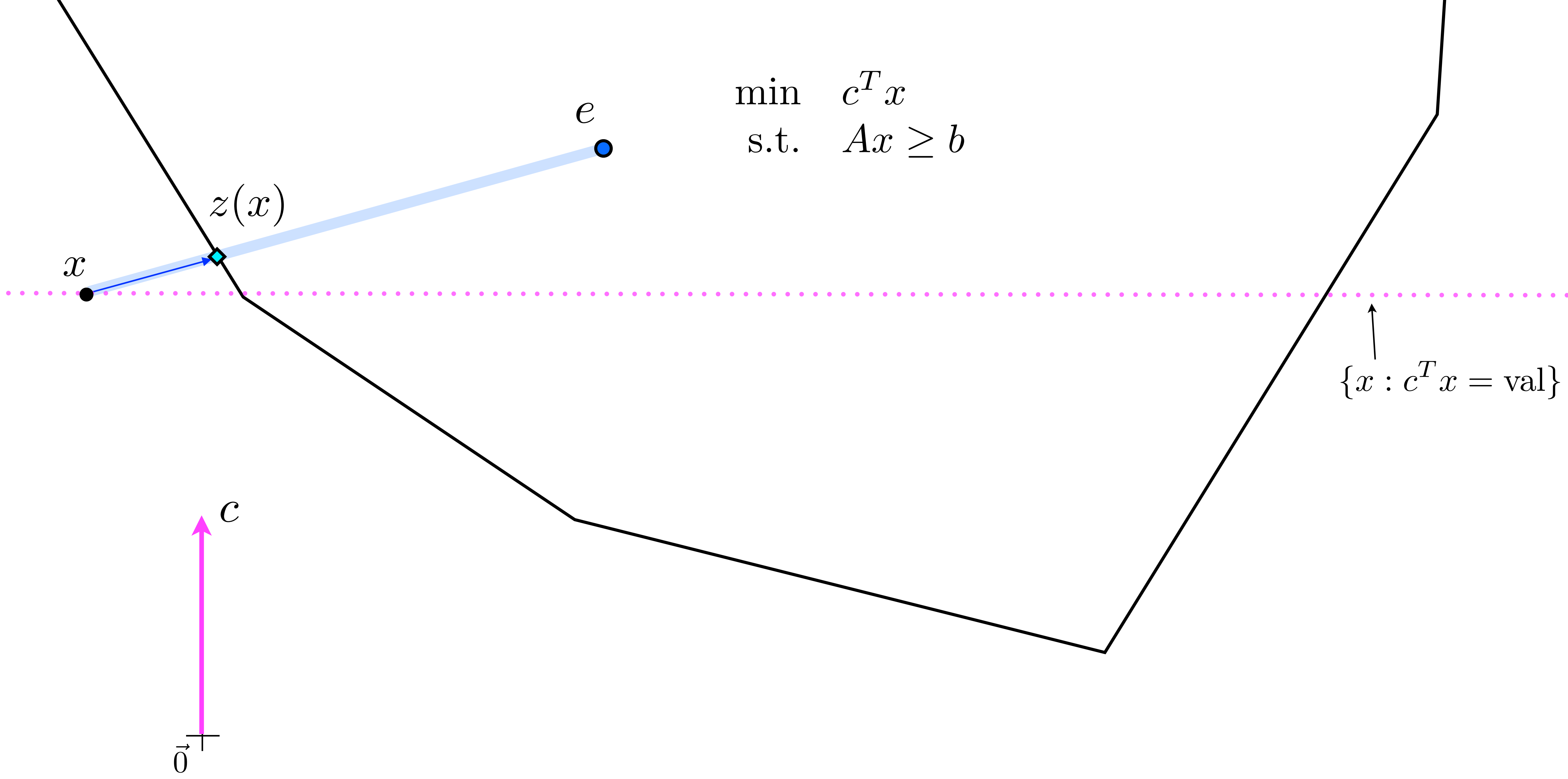




$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \geq b \end{array}$$

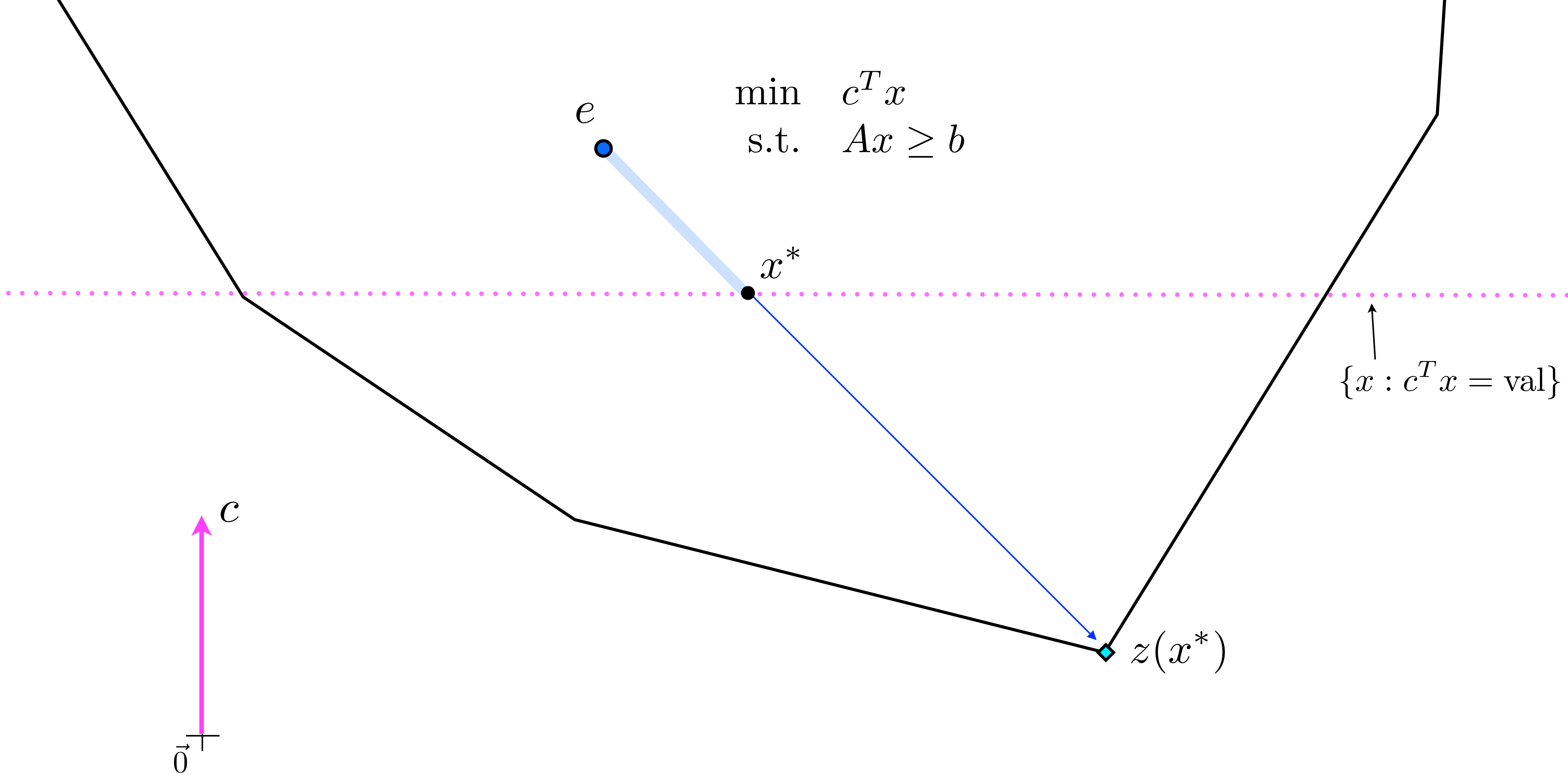
$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \geq b \end{array}$$

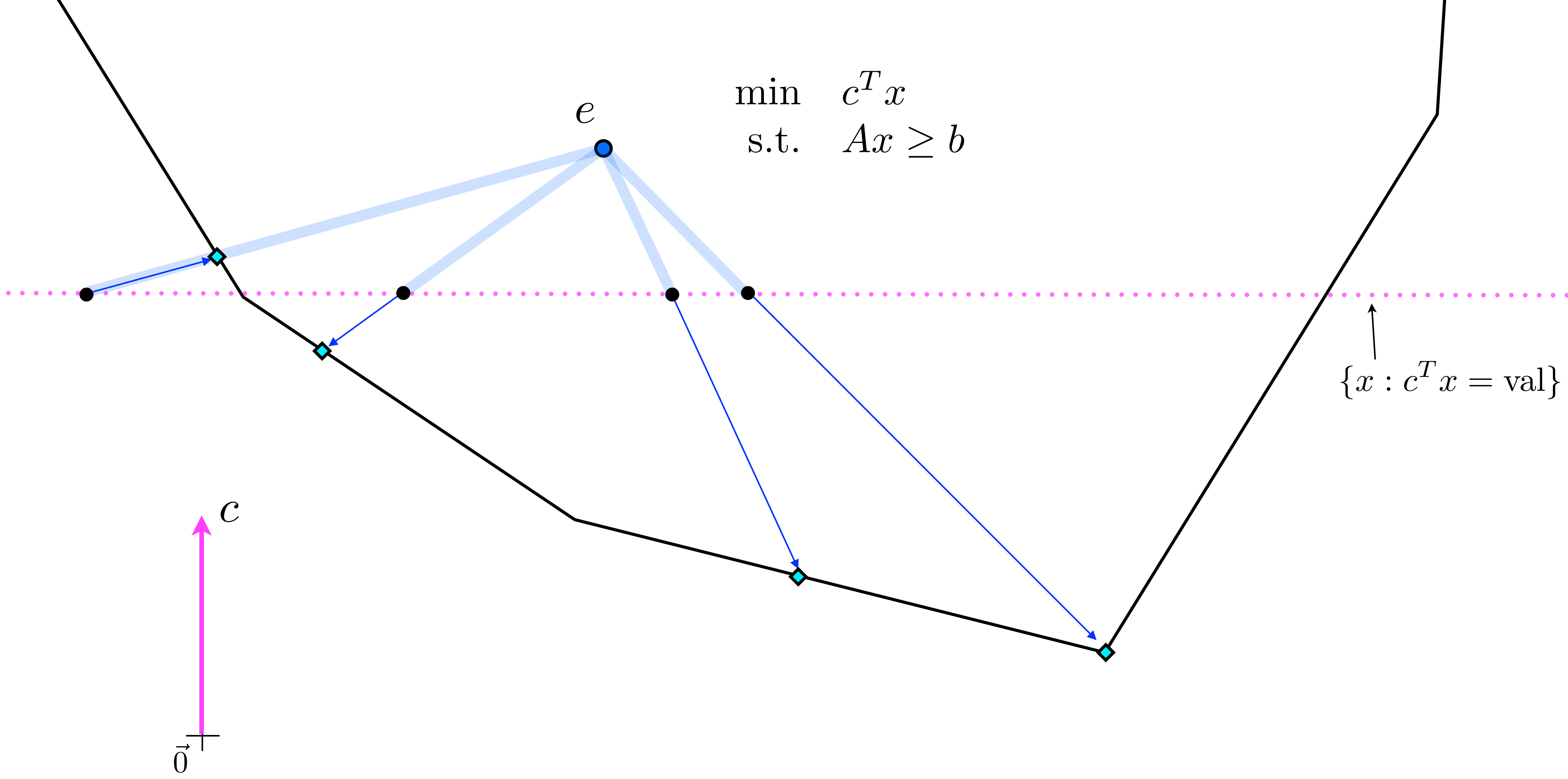


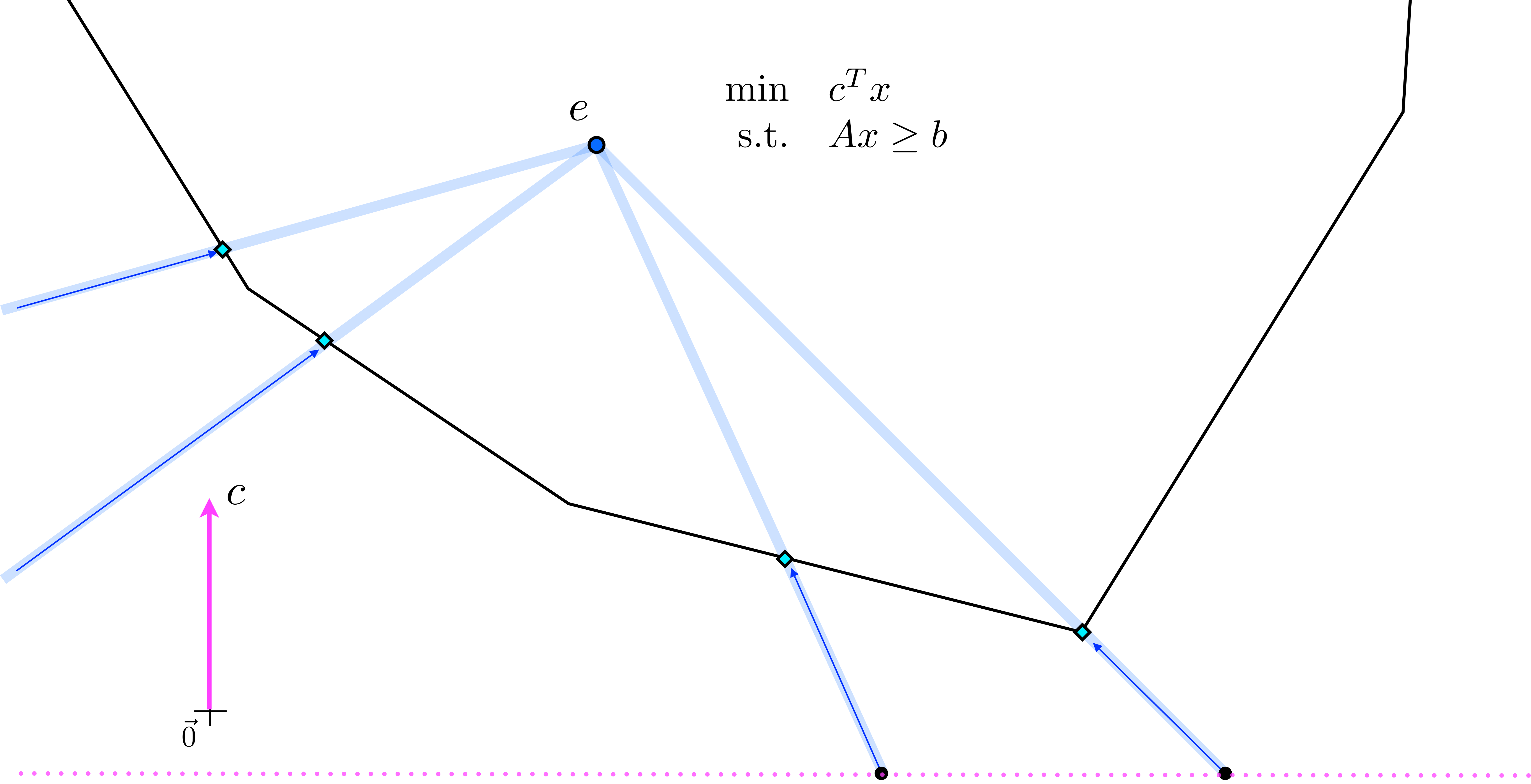


$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \geq b \end{array}$$

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \end{aligned}$$




e
$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \geq b \end{array}$
$\{x : c^T x = \text{val}\}$
c
$\vec{0}$



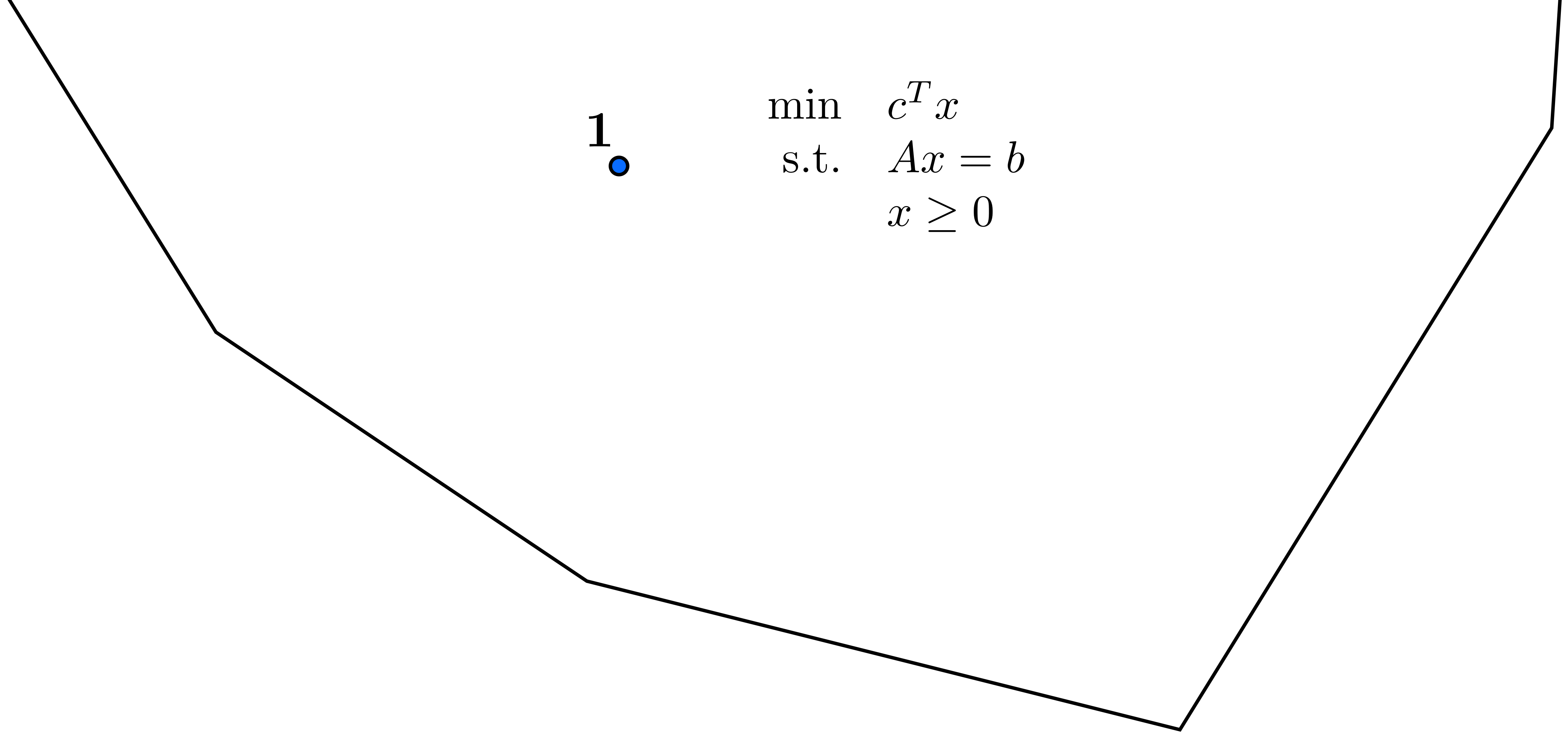
$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax \geq b \end{aligned}$$

$$\left.\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax \bigcirc\!\!\!\!\!= b \\ & x \geq 0 \end{array}\right\} \text{LP}$$

Assume optimal value – denoted opt_val – is finite.

Assume $\mathbf{1}$ is feasible.

vector of all one's



$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \end{aligned}$$

Think of this 2-dimensional plane as being the slice of $\mathbb{R}^n$ *cut out by* $\{x : Ax = b\}$.

$$\left.\begin{array}{rl} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array}\right\} \text{LP}$$

Assume optimal value – denoted opt_val – is finite.

Assume $\mathbf{1}$ is feasible.

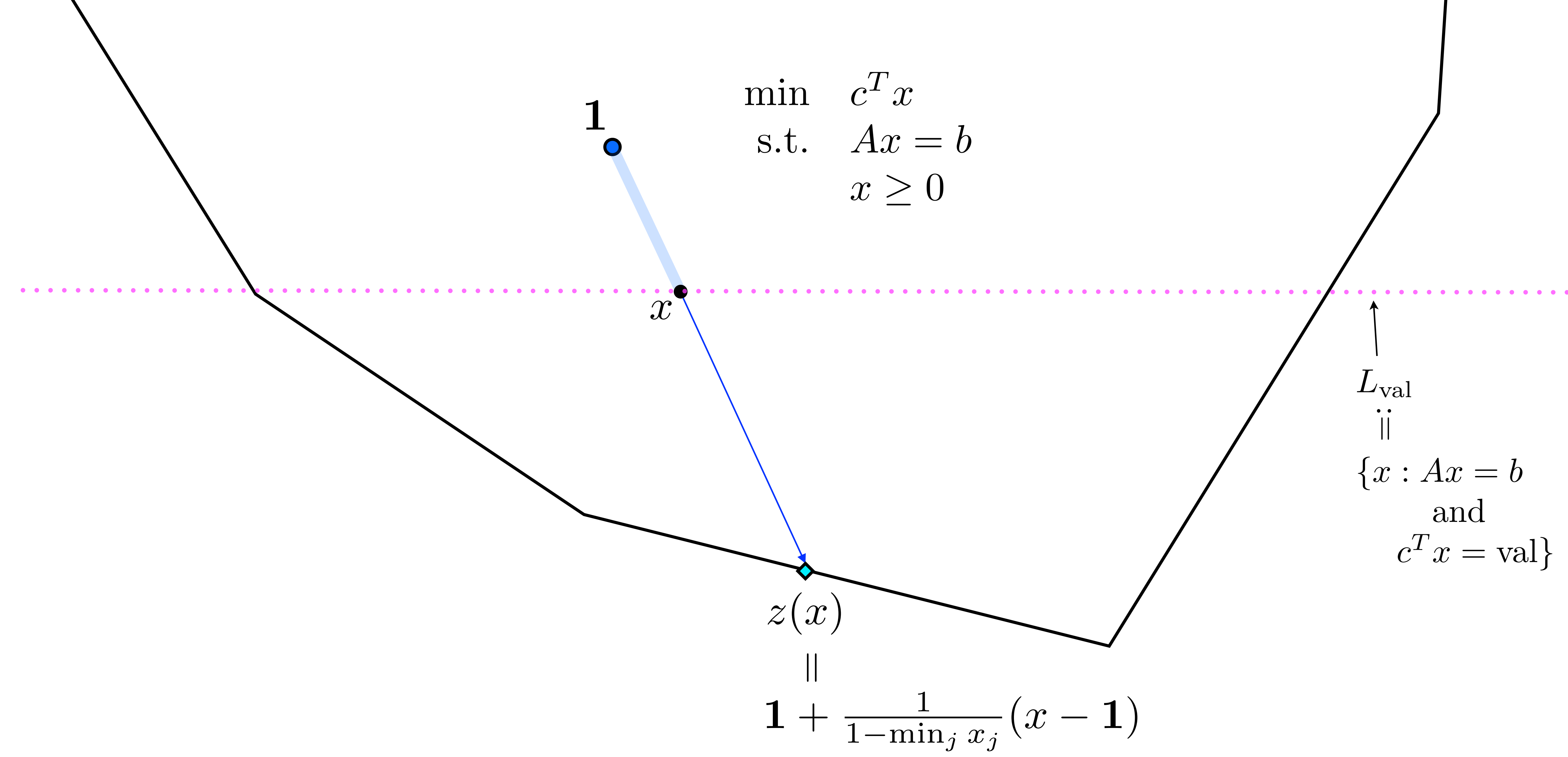
vector of all one's

Lemma: If x satisfies $Ax = b$ and $c^T x < c^T \mathbf{1}$, then $\min_j x_j < 1$

Proof: Otherwise $x(t) := \mathbf{1} + t\,(x - \mathbf{1})$ feasible for all $t \geq 0$, and $c^T x(t) \to -\infty$, contradicting opt_val finite.

Projection (from $\mathbf{1}$) of x to boundary of feasible region:

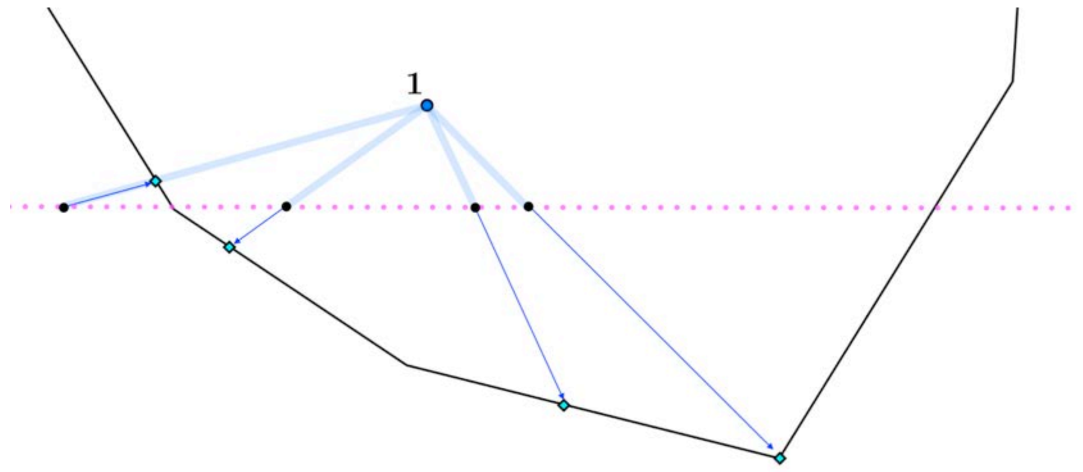
$$z(x) \; = \; \mathbf{1} + \tfrac{1}{1 - \min_j x_j} \, (x - \mathbf{1})$$


$\min \quad c^T x$
$\text{s.t.} \quad Ax = b$
$x \geq 0$
$\mathbf{1}$
x
L_{val}
$::$
$||$
$\{x : Ax = b$ and $c^T x = \text{val}\}$
$z(x)$
$||$
$\mathbf{1} + \frac{1}{1 - \min_j x_j}(x - \mathbf{1})$

$$\left.\begin{array}{rl} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array}\right\} \text{LP}$$

opt_val	optimal value, assumed finite
val	a fixed value satisfying $\text{val} < c^T \mathbf{1}$
L_{val}	$\{x : Ax = b \text{ and } c^T x = \text{val}\}$

$$x \mapsto z(x) := \mathbf{1} + \tfrac{1}{1 - \min_j x_j}\,(x - \mathbf{1})$$



$$\begin{aligned} c^T z(x) &= c^T \mathbf{1} + \tfrac{1}{1-\min_j x_j}\,(c^T x - c^T \mathbf{1}) \\ &= c^T \mathbf{1} + \tfrac{1}{1-\min_j x_j}\,\underbrace{(\,\text{val} - c^T \mathbf{1}\,)}_{\text{a negative constant}} \end{aligned}$$

Thus, for $x, y \in L_{\text{val}}$, $\quad c^T z(x) < c^T z(y) \quad \Leftrightarrow \quad \min_j x_j > \min_j y_j$

Theorem: LP is equivalent to
$$\begin{array}{rl} \max_x & \min_j x_j \\ \text{s.t.} & Ax = b \\ & c^T x = \text{val} \end{array}$$

$$\left.\begin{array}{rl} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array}\right\} \text{LP} \quad \longleftrightarrow \quad \begin{array}{rl} \max_x & \min_j x_j \\ \text{s.t.} & Ax = b \\ & c^T x = \text{val} \end{array}$$

$x \mapsto \min_j x_j$ is <u>the</u> exemplary nonsmooth concave function

- Lipschitz continuous with constant $\kappa_0 = 1$
- Subgradients at x are the convex combinations
  of the standard basis vectors $e(k)$ for which $x_k = \min_j x_j$

Thus, projected subgradients at x are the convex combinations
of the corresponding columns of the projection matrix

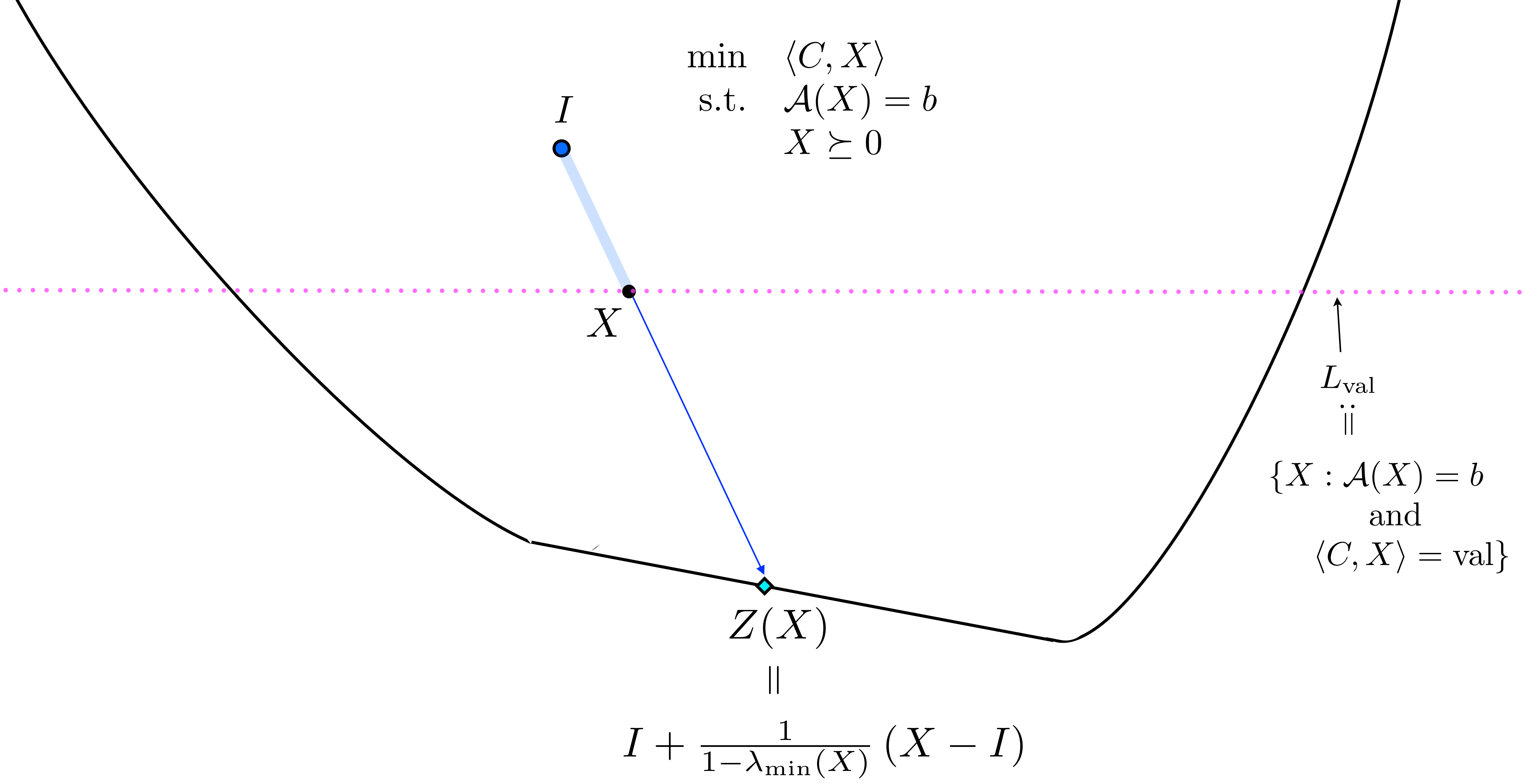
$$\bar{P} := I - \bar{A}^T(\bar{A}\,\bar{A}^T)^{-1}\bar{A} \quad \text{where } \bar{A} = \left[\begin{smallmatrix} A \\ c^T \end{smallmatrix}\right]$$

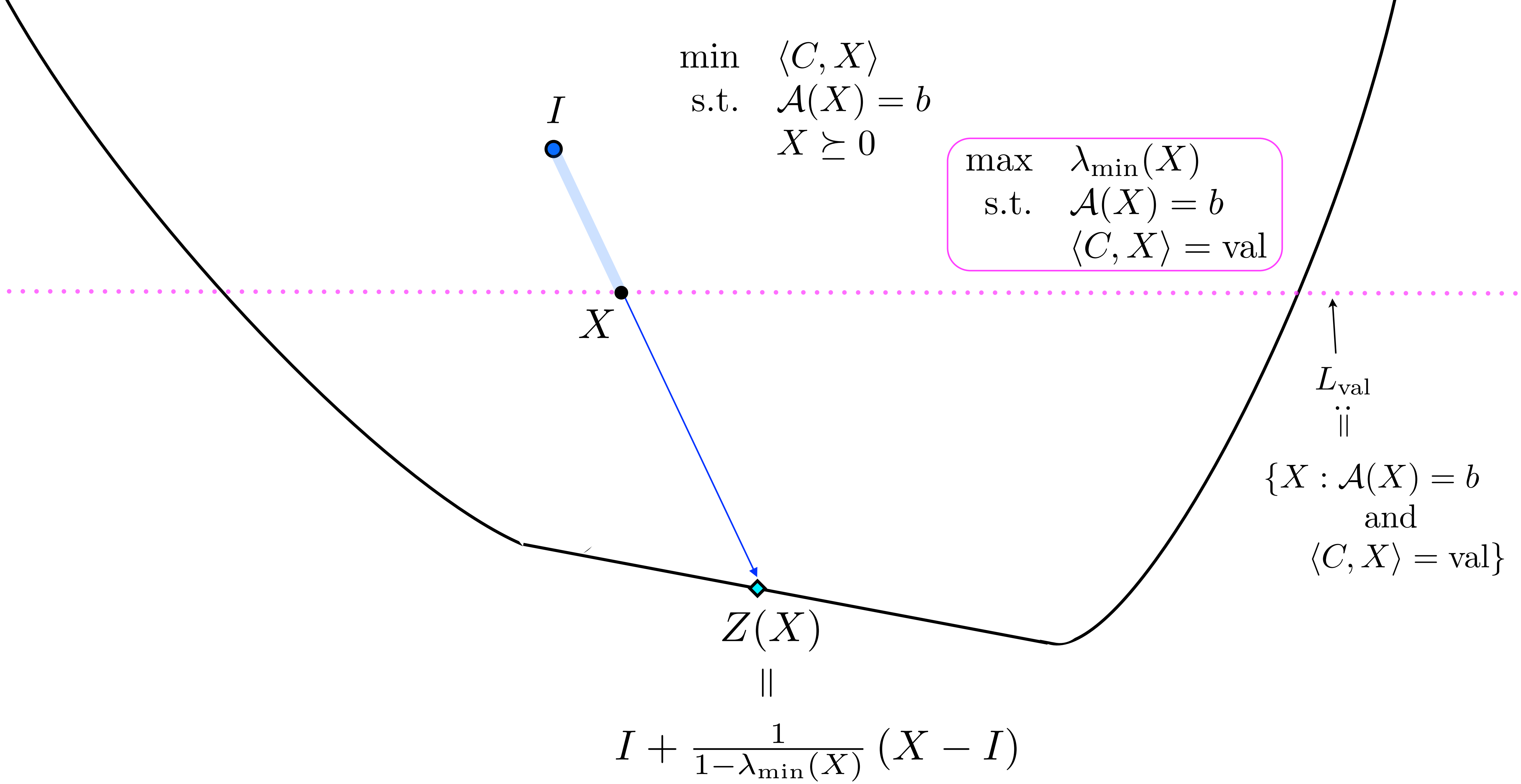
Hence, in implementing a subgradient method,
one option in choosing a subgradient at x
is simply to compute any column $\bar{P}_k$ for which $x_k = \min_j x_j$.

If this is done for all iterates x, the algorithm assumes a combinatorial flavor.

If, in addition, line searches are done exactly,
the algorithm becomes distinctly combinatorial. (Cost of exact line search is $O(n \log n)$)

Moreover, with a modest amount of preprocessing work,
the cost of each iterate
is proportional to the number of nonzero entries in A.


$\min \quad \langle C, X \rangle$
s.t. $\quad \mathcal{A}(X) = b$
$X \succeq 0$
I
X
L_{val}
$::$
$\|$
$\{X : \mathcal{A}(X) = b$
and
$\langle C, X \rangle = \text{val}\}$
$Z(X)$
$\|$
$I + \frac{1}{1-\lambda_{\min}(X)}(X - I)$



$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \end{array}$$

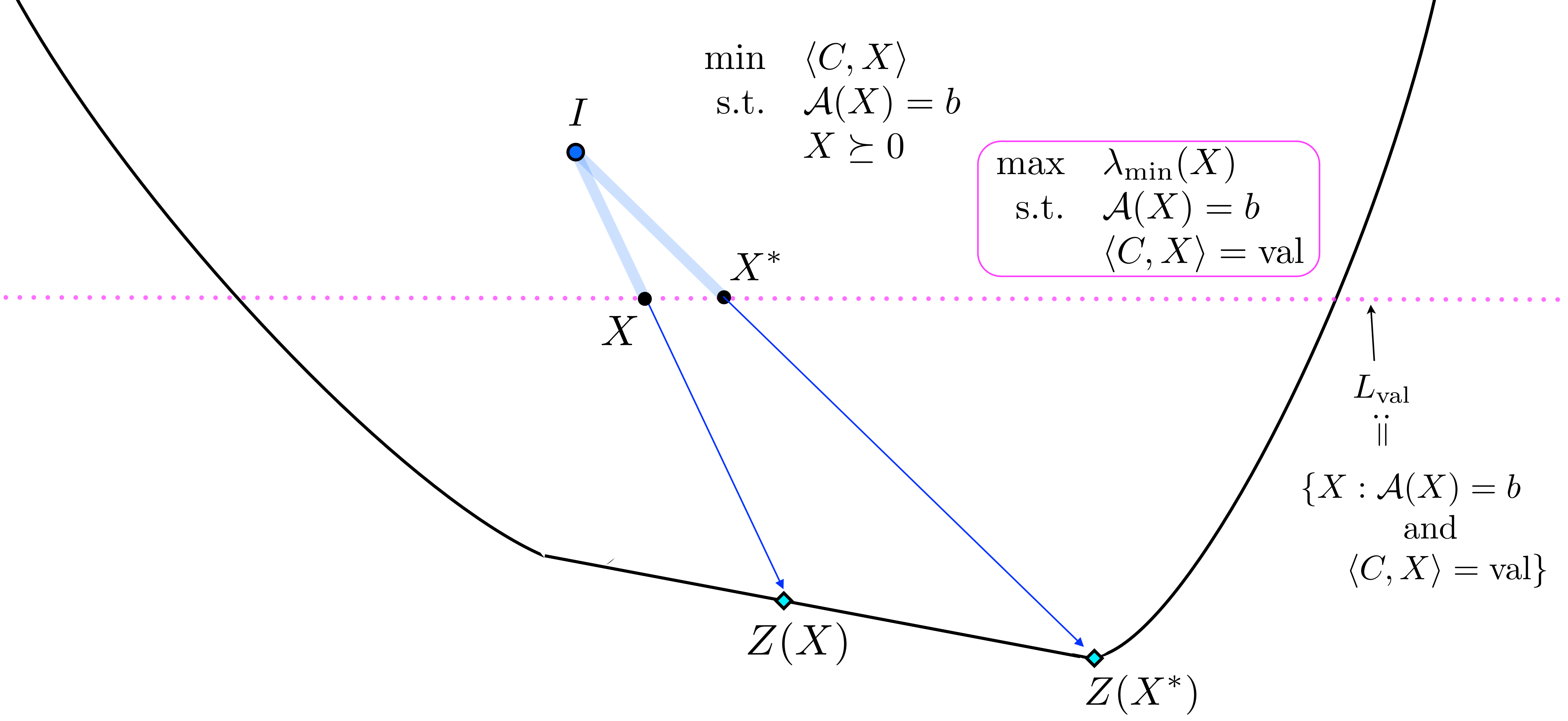
$$\boxed{\begin{array}{ll} \max & \lambda_{\min}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X\rangle = \text{val} \end{array}}$$

$$L_{\text{val}} \mathrel{\mathop:}= \{X : \mathcal{A}(X) = b \text{ and } \langle C, X\rangle = \text{val}\}$$

$$Z(X) = I + \frac{1}{1-\lambda_{\min}(X)}(X - I)$$

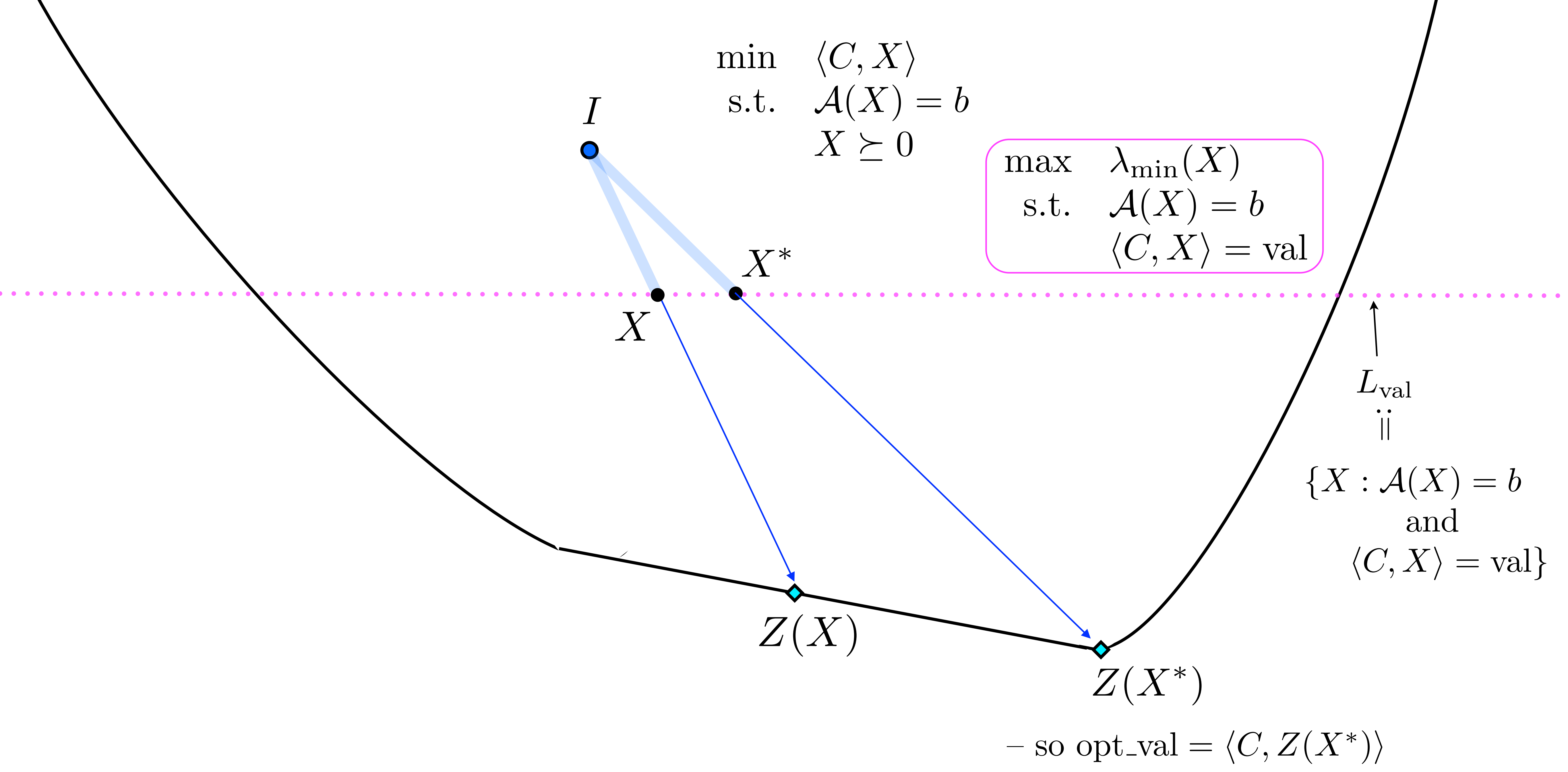
$$\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \end{array}$$

$$\boxed{\begin{array}{ll} \max & \lambda_{\min}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X\rangle = \text{val} \end{array}}$$



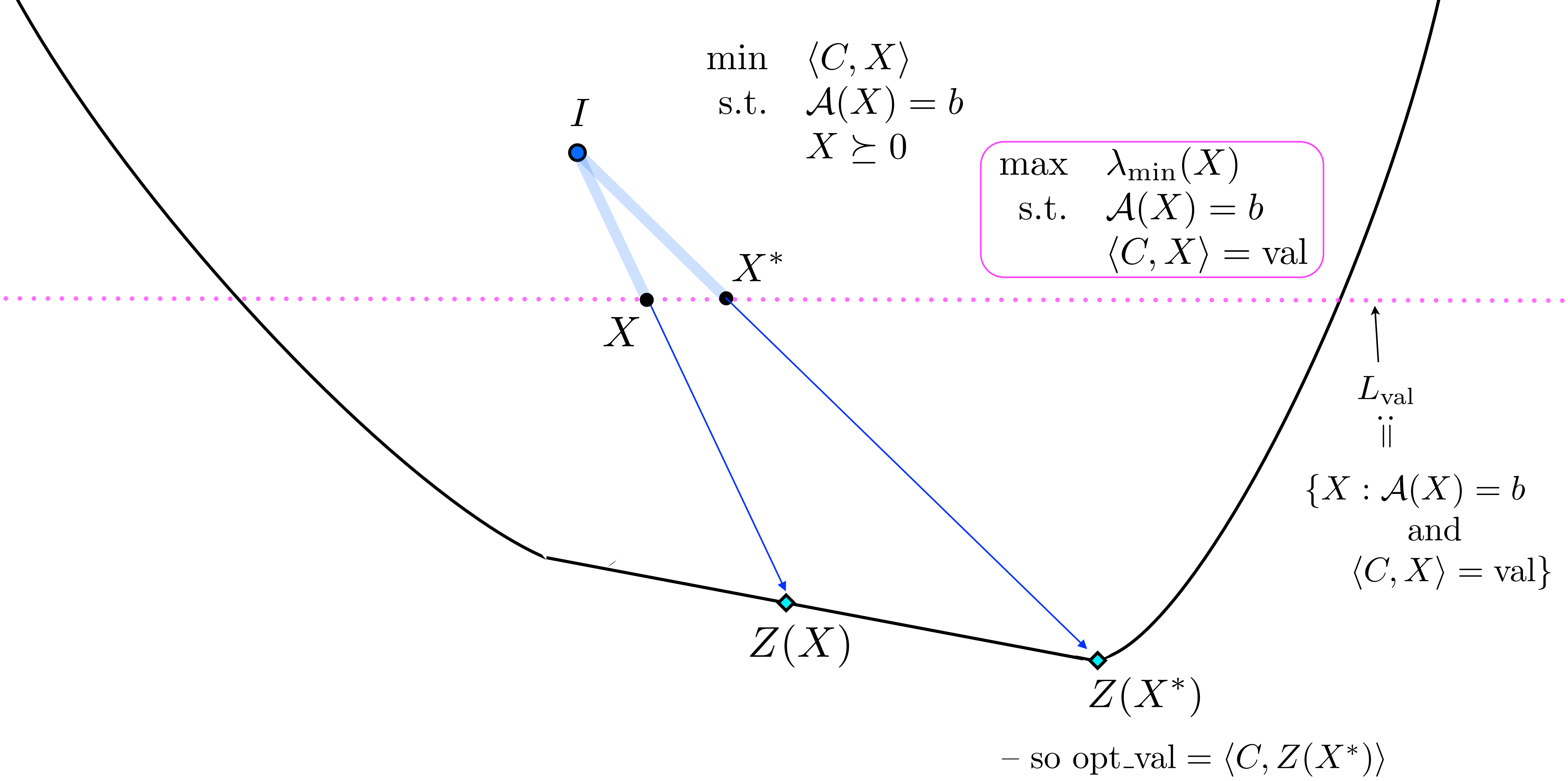
$$L_{\text{val}} \mathrel{\rotatebox{90}{$:=$}} \{X : \mathcal{A}(X) = b \text{ and } \langle C, X\rangle = \text{val}\}$$

– so opt_val $= \langle C, Z(X^*)\rangle$


$\min \ \langle C, X\rangle$ s.t. $\mathcal{A}(X) = b$, $X \succeq 0$
$\max \ \lambda_{\min}(X)$ s.t. $\mathcal{A}(X) = b$, $\langle C, X\rangle = \text{val}$
I
X^*
X
$L_{\text{val}} \ \vdots\vdots \ \{X : \mathcal{A}(X) = b$ and $\langle C, X\rangle = \text{val}\}$
$Z(X)$
$Z(X^*)$
– so opt_val $= \langle C, Z(X^*)\rangle$

Goal: Compute X satisfying $\dfrac{\langle C, Z(X)\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \leq \epsilon$

To accomplish this, how accurately does $\lambda_{\min}(X)$ *need to approximate* $\lambda_{\min}(X^*)$*?*



$\begin{array}{rl} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \end{array}$
$\begin{array}{rl} \max & \lambda_{\min}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X\rangle = \text{val} \end{array}$
I
X^*
X
L_{val}
$\vdots$
$\{X : \mathcal{A}(X) = b$
and
$\langle C, X\rangle = \text{val}\}$
$Z(X)$
$Z(X^*)$
– so opt_val $= \langle C, Z(X^*)\rangle$

Proposition:

$$\frac{\langle C, Z(X)\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \le \epsilon$$

$$\Leftrightarrow$$

$$\lambda_{\min}(X^*) - \lambda_{\min}(X) \le \frac{\epsilon}{1-\epsilon} \, \frac{\langle C, I\rangle - \text{val}}{\langle C, I\rangle - \text{opt\_val}}$$

$\left.\begin{array}{ll} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \end{array}\right\}$ SDP	$\begin{array}{ll} \max & \lambda_{\min}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X\rangle = \text{val} \end{array}$
$\dfrac{\langle C, Z(X)\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \leq \epsilon$	$\lambda_{\min}(X^*) - \lambda_{\min}(X) \leq \dfrac{\epsilon}{1-\epsilon}\, \dfrac{\langle C, I\rangle - \text{val}}{\langle C, I\rangle - \text{opt\_val}}$

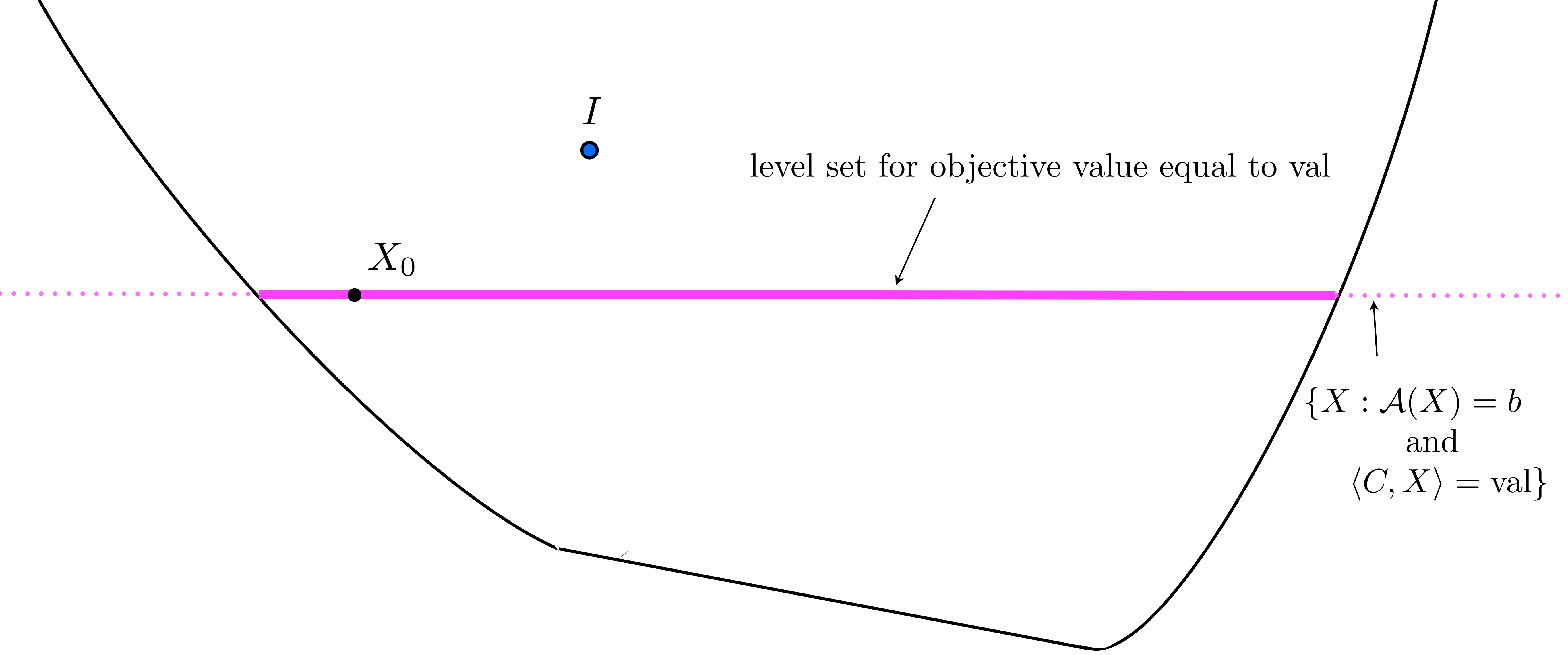
Corollary for subgradient method applied to SDP equivalent problem:

$$N \geq \Big(\frac{\overset{\text{an upper bound on } \|X_0 - X^*\|}{R}}{\epsilon} \; \frac{\langle C, I\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{val}} \Big)^2 \qquad \Rightarrow \qquad \frac{\langle C, Z(X_N)\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \leq \epsilon$$

The upper bound R has no direct meaning to SDP.

To replace R with a meaningful quantity,
we require the user to provide SDP-feasible X_0 satisfying $\langle C, X_0\rangle < \langle C, I\rangle$.

We then let $\text{val} := \langle C, X_0\rangle$.


I
level set for objective value equal to val
X_0
$\{X : \mathcal{A}(X) = b$
and
$\langle C, X \rangle = \text{val}\}$

Let diam denote an upper bound
on *all* SDP level sets for objective values val $< \langle C, I \rangle$

$\left.\begin{array}{rl} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \end{array}\right\}$ SDP	$\begin{array}{rl} \max & \lambda_{\min}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X\rangle = \text{val} \end{array}$
$\dfrac{\langle C, Z(X)\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \leq \epsilon$	$\lambda_{\min}(X^*) - \lambda_{\min}(X) \leq \dfrac{\epsilon}{1-\epsilon} \, \dfrac{\langle C, I\rangle - \text{val}}{\langle C, I\rangle - \text{opt\_val}}$

Corollary for subgradient method applied to SDP equivalent problem:

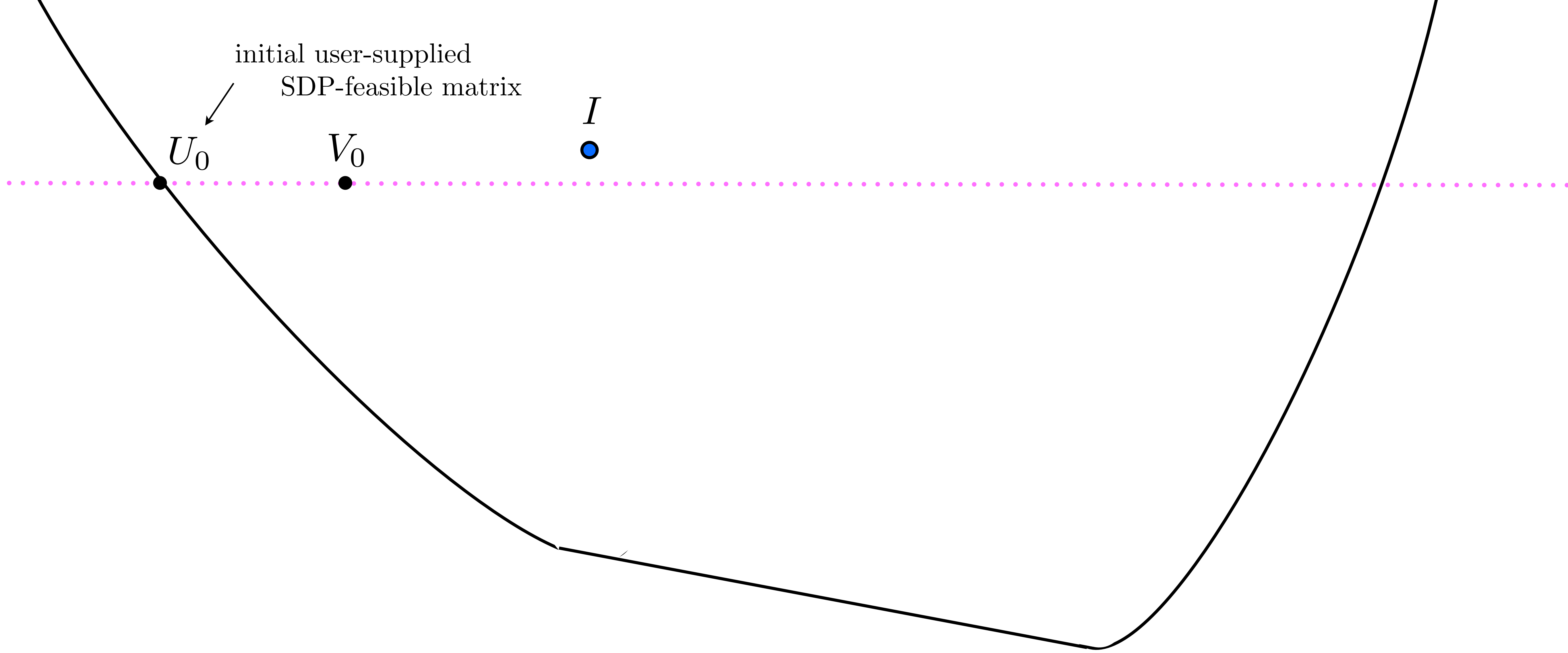
an upper bound on $\|X_0 - X^*\|$

$$N \geq \left(\frac{R}{\epsilon} \, \frac{\langle C, I\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{val}} \right)^2 \qquad \Rightarrow \qquad \frac{\langle C, Z(X_N)\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \leq \epsilon$$

$$N \geq \left(\frac{\boxed{\text{diam}}}{\epsilon} \, \frac{\langle C, I\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{val}} \right)^2 \qquad \Rightarrow \qquad \frac{\langle C, Z(X_N)\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \leq \epsilon$$

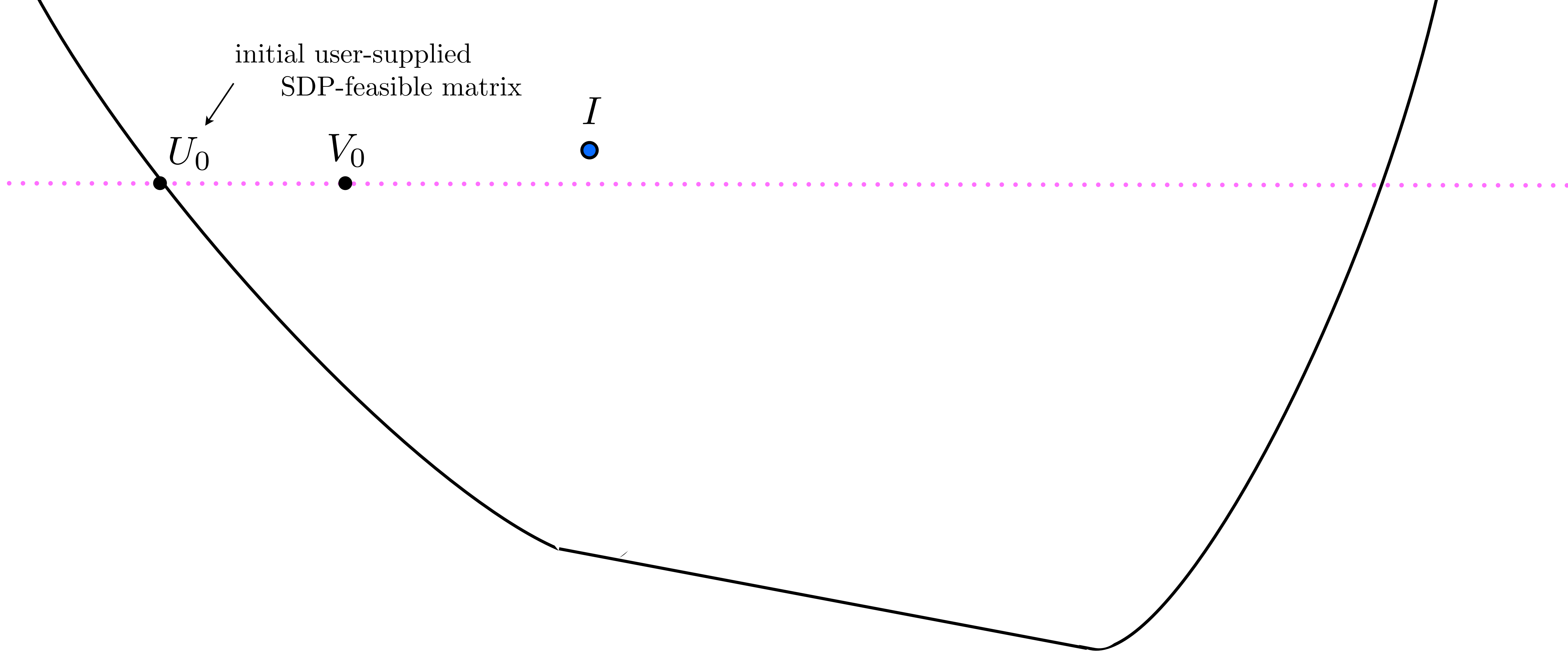
The ratio $\frac{\langle C,I\rangle - \text{opt\_val}}{\langle C,I\rangle - \text{val}}$ is problematic
in that we want not to assume opt_val is known,
and in that even in nice situations, the ratio can be large,
making the magnitude of N be uninteresting in most applications.

We thus are lead to create a two-phase computational procedure ...



Phase I:

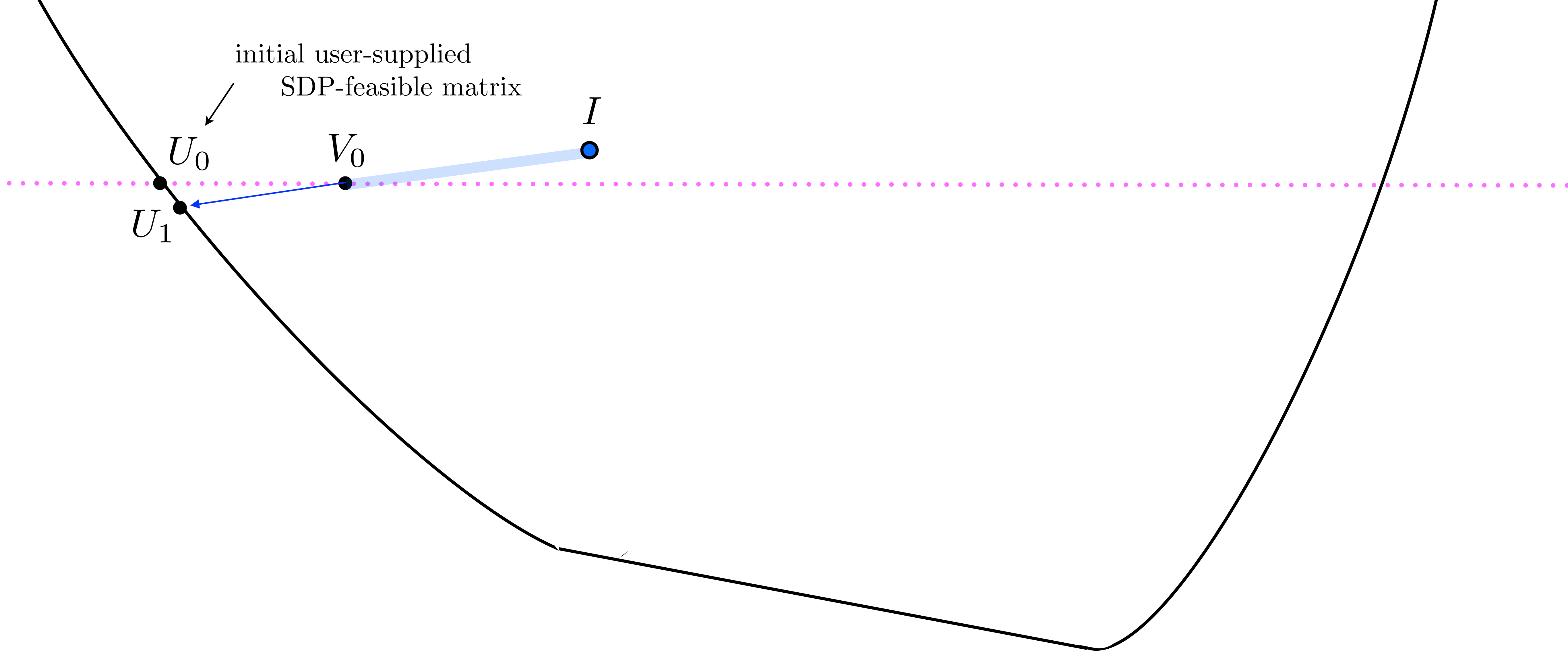
1) Beginning at U_k, $\overbrace{\text{apply } \lceil 9\,\mathrm{diam}^2 \rceil}^{\text{For this we are having to make the strong assumption that we know and upper bound diam on the diameter of the level sets.}}$ iterations of subgradient method, resulting in a matrix V_k



Phase I:

1) Beginning at U_k, $\overbrace{\text{apply } \lceil 9\,\mathrm{diam}^2 \rceil}^{\substack{\text{For this we are having to make the strong assumption} \\ \text{that we know and upper bound diam} \\ \text{on the diameter of the level sets.}}}$ iterations of subgradient method,
resulting in a matrix V_k

2) If $\lambda_{\min}(V_k) \leq 1/3$, then terminate with output $X_0 = U_k$.



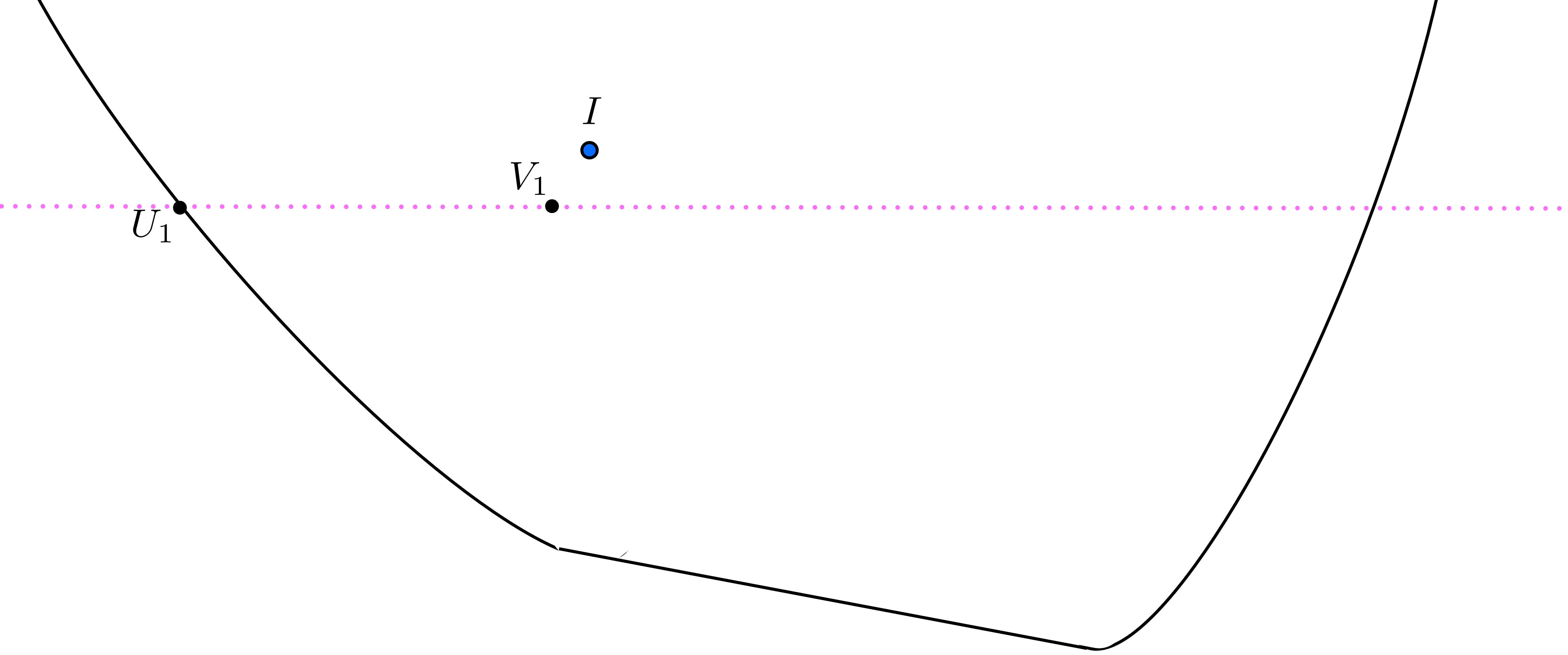
Phase I:

For this we are having to make the strong assumption that we know and upper bound diam on the diameter of the levels sets.

1) Beginning at U_k, $\overbrace{\text{apply } \lceil 9\,\mathrm{diam}^2 \rceil}$ iterations of subgradient method, resulting in a matrix V_k

2) If $\lambda_{\min}(V_k) \leq 1/3$, then terminate with output $X_0 = U_k$.

3) Compute $U_{k+1} = Z(V_{k+1})$, let $k + 1 \to k$, and go to step 1.

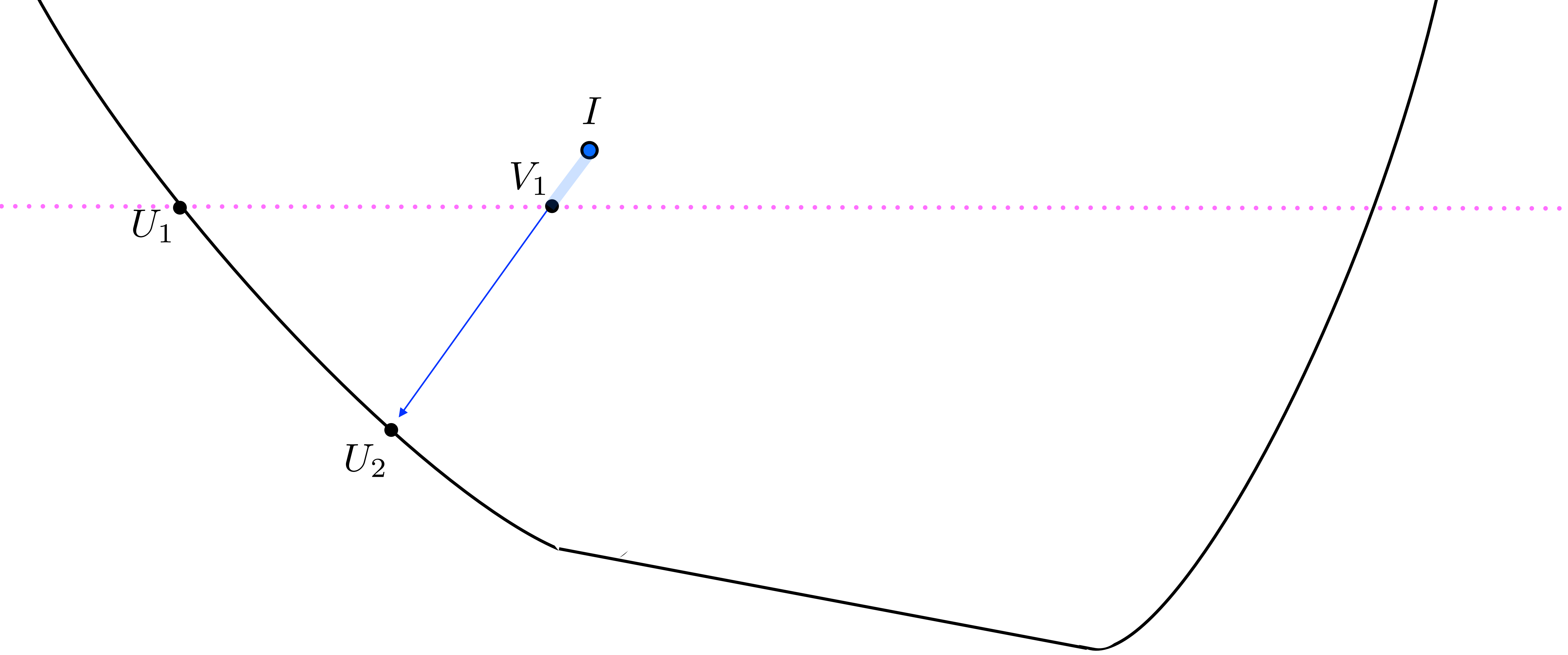


Phase I:

1) Beginning at U_k, apply $\lceil 9\,\mathrm{diam}^2 \rceil$ iterations of subgradient method, resulting in a matrix V_k

2) If $\lambda_{\min}(V_k) \leq 1/3$, then terminate with output $X_0 = U_k$.

3) Compute $U_{k+1} = Z(V_{k+1})$, let $k+1 \to k$, and go to step 1.

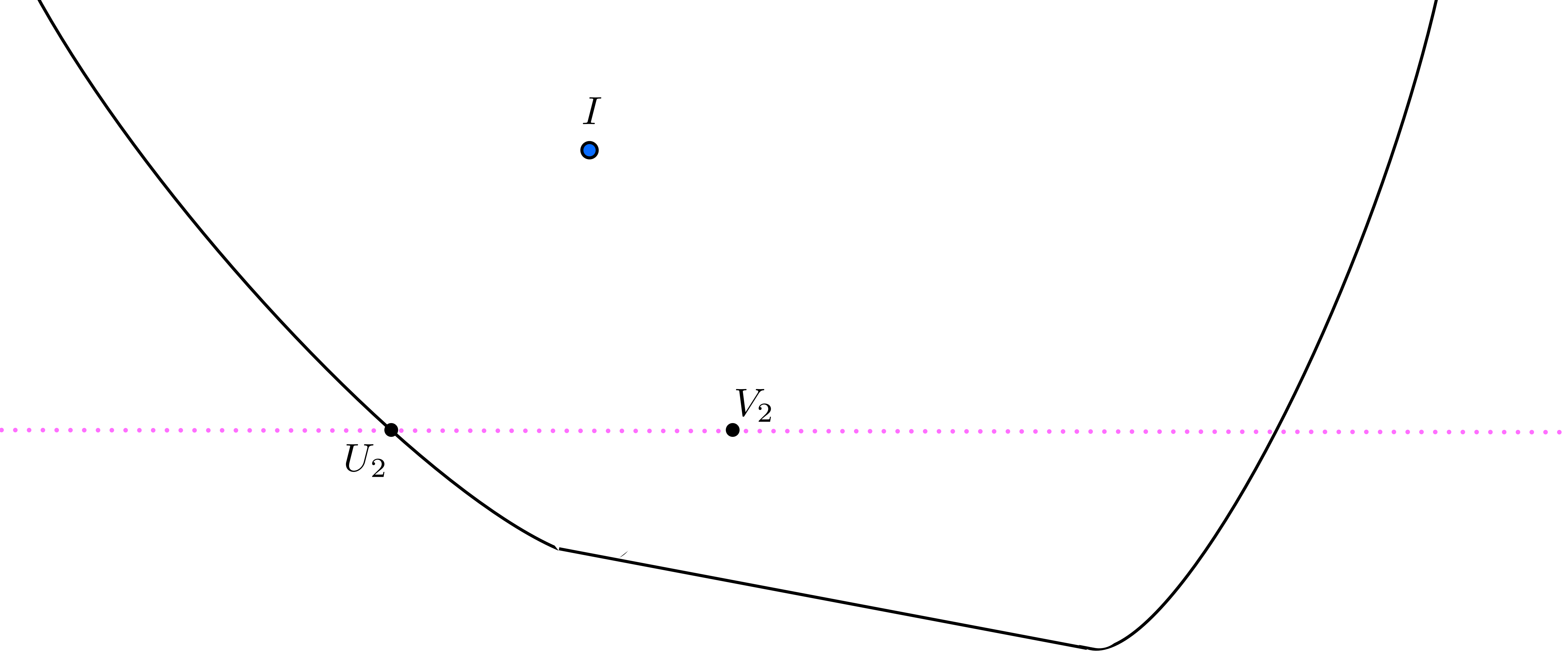


Phase I:

1) Beginning at U_k, apply $\lceil 9\,\mathrm{diam}^2 \rceil$ iterations of subgradient method, resulting in a matrix V_k

2) If $\lambda_{\min}(V_k) \leq 1/3$, then terminate with output $X_0 = U_k$.

3) Compute $U_{k+1} = Z(V_{k+1})$, let $k+1 \rightarrow k$, and go to step 1.

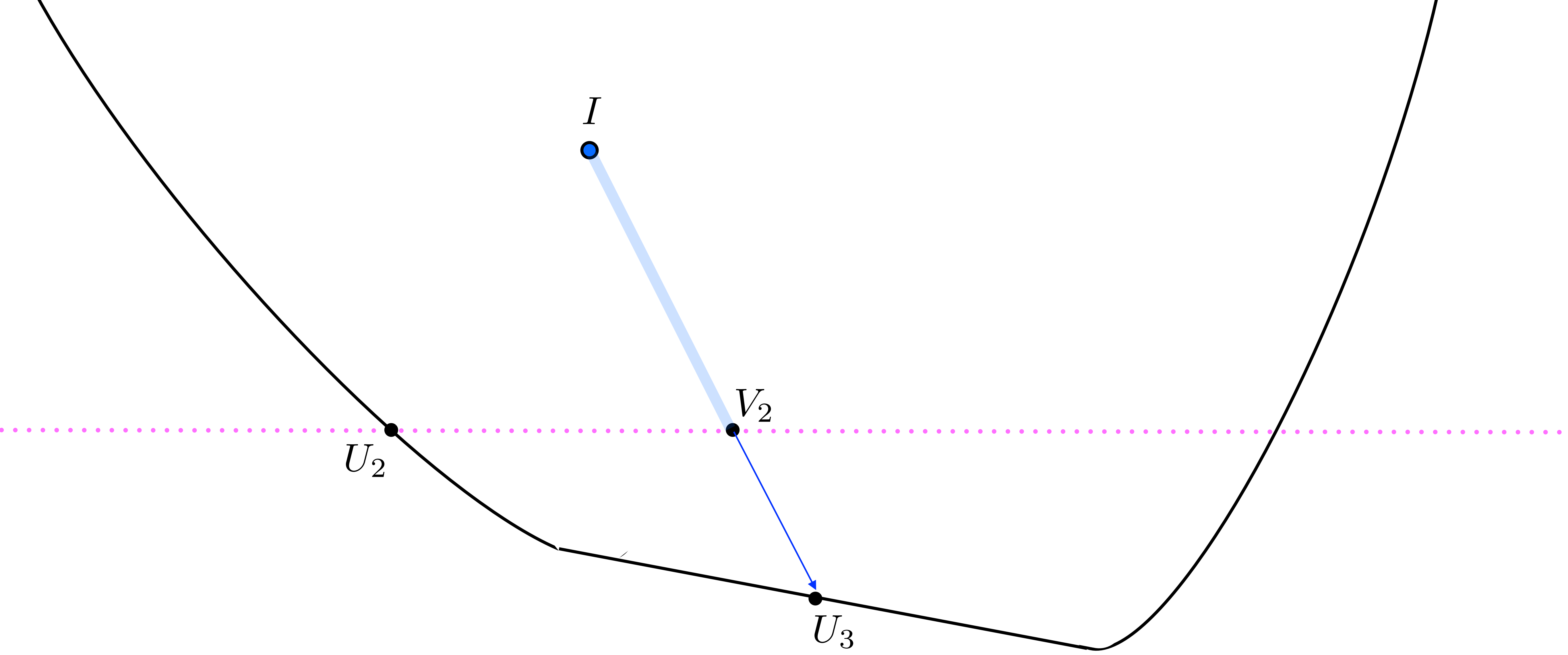


Phase I:

1) Beginning at U_k, apply $\lceil 9\,\mathrm{diam}^2 \rceil$ iterations of subgradient method, resulting in a matrix V_k

2) If $\lambda_{\min}(V_k) \leq 1/3$, then terminate with output $X_0 = U_k$.

3) Compute $U_{k+1} = Z(V_{k+1})$, let $k+1 \rightarrow k$, and go to step 1.

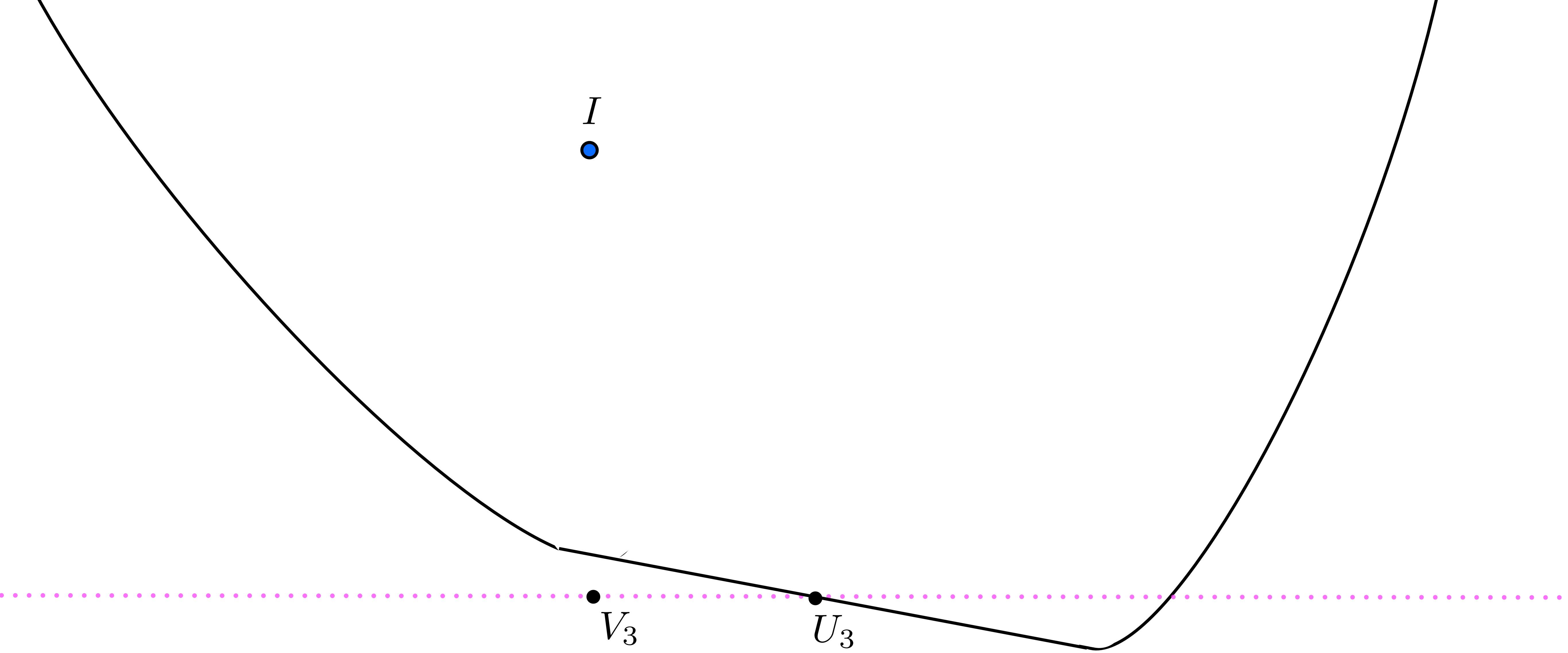


Phase I:

1) Beginning at U_k, apply $\lceil 9\,\mathrm{diam}^2 \rceil$ iterations of subgradient method, resulting in a matrix V_k

2) If $\lambda_{\min}(V_k) \leq 1/3$, then terminate with output $X_0 = U_k$.

3) Compute $U_{k+1} = Z(V_{k+1})$, let $k + 1 \to k$, and go to step 1.

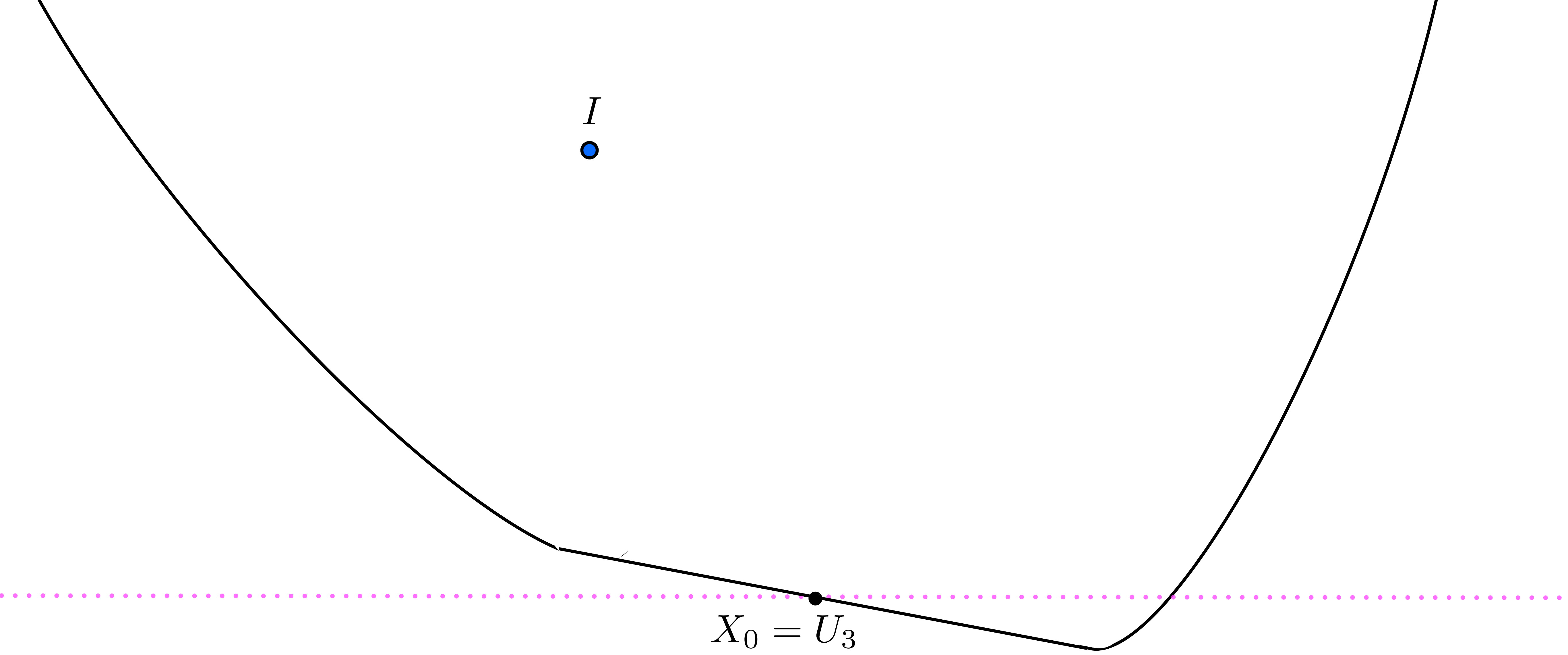


Phase I:

1) Beginning at U_k, apply $\lceil 9\,\mathrm{diam}^2 \rceil$ iterations of subgradient method, resulting in a matrix V_k

2) If $\lambda_{\min}(V_k) \leq 1/3$, then terminate with output $X_0 = U_k$.

3) Compute $U_{k+1} = Z(V_{k+1})$, let $k+1 \to k$, and go to step 1.

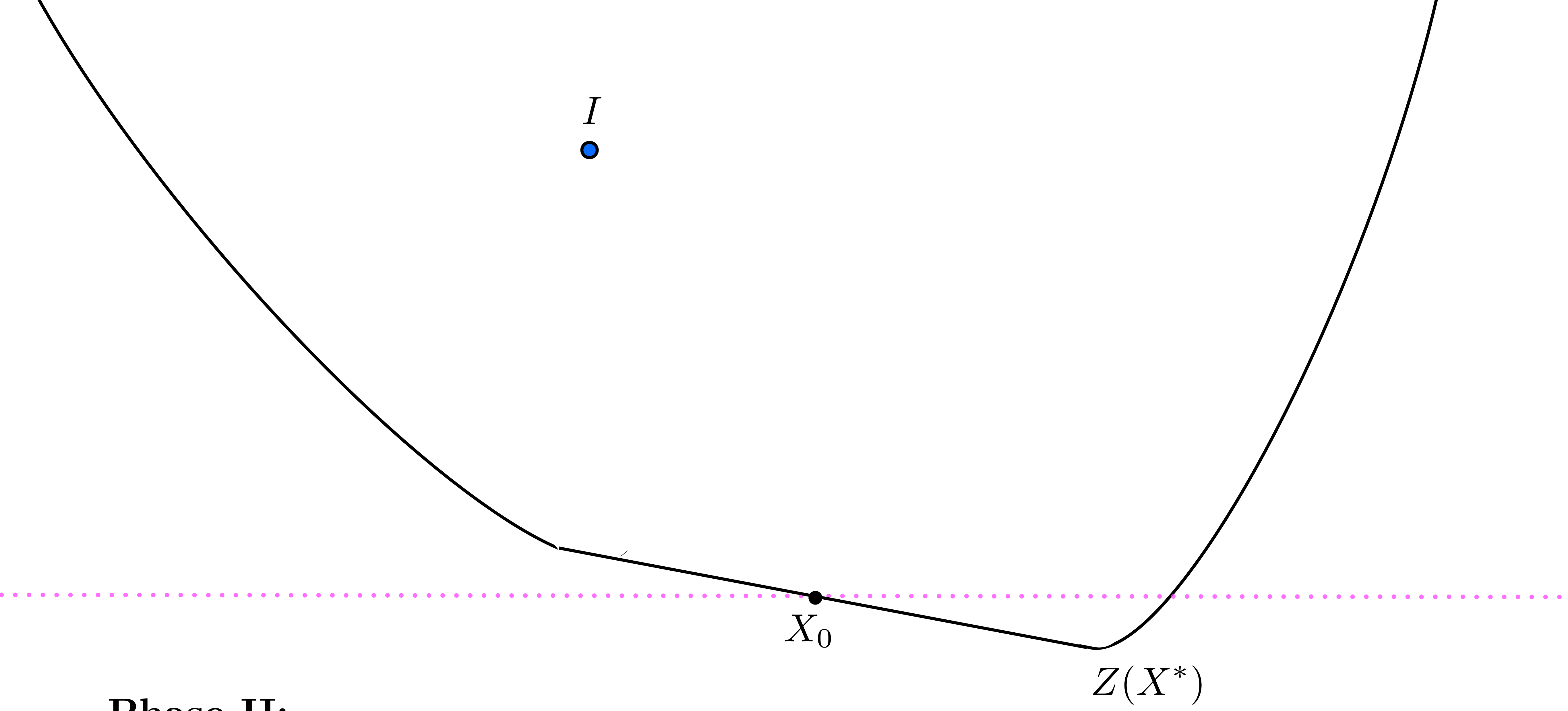


Phase I:

1) Beginning at U_k, apply $\lceil 9\,\mathrm{diam}^2 \rceil$ iterations of subgradient method, resulting in a matrix V_k

2) If $\lambda_{\min}(V_k) \leq 1/3$, then terminate with output $X_0 = U_k$.

3) Compute $U_{k+1} = Z(V_{k+1})$, let $k+1 \rightarrow k$, and go to step 1.



Phase II:

Beginning at X_0, apply $\lceil (3\,\mathrm{diam}/\epsilon)^2 \rceil$ iterations of subgradient method.

$$\begin{array}{rl} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \end{array}$$

Phase I + Phase II gives an algorithm to compute X for which the projection $Z = Z(X)$ satisfies $\dfrac{\langle C, Z\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \leq \epsilon$

Theorem: The total number of subgradient iterations does not exceed

$$(9\ \text{diam}^2 + 1) \cdot \left(\frac{1}{\epsilon^2} + \log_{3/2}\left(\frac{\langle C, I\rangle - \text{opt\_val}}{\langle C, I\rangle - \underbrace{\text{val}_0}_{\text{objective value of input matrix } U_0}} \right) \right)$$

Corollary: Assume I is on the central path.

If the initial matrix is chosen as $U_0 = I - \frac{1}{\lambda_{\max}(\pi(C))}\, \underbrace{\pi(C)}_{\text{projection of } C \text{ onto nullspace of } \mathcal{A}}$,

then the total number of subgradient iterations does not exceed

$$(9\ \text{diam}^2 + 1) \cdot \left(\frac{1}{\epsilon^2} + \log_{3/2}(n) \right)$$

Compare with interior-point methods: $O(\sqrt{n}\, \log(1/\epsilon))$ iterations

Smoothing

Following Nesterov, rely on the smooth concave function

$$f_\mu(X) := -\mu \ln \sum_j e^{-\lambda_j(X)/\mu} \qquad \text{(for fixed } \mu > 0)$$

Easy to see: $\lambda_{\min}(X) - \mu \ln n \;\le\; f_\mu(X) \;\le\; \lambda_{\min}(X)$

Not so obvious, but which Nesterov showed:

$$\|\nabla f_\mu(X) - \nabla f_\mu(Y)\| \;\le\; \tfrac{1}{\mu}\,\|X - Y\|$$

that is, $X \mapsto \nabla f_\mu(X)$ has Lipschitz constant $\kappa_1 = 1/\mu$

$$\nabla f_\mu(X) = \frac{1}{\sum_j e^{-\lambda_j(X)/\mu}}\, Q \begin{bmatrix} e^{-\lambda_1(X)/\mu} & & \\ & \ddots & \\ & & e^{-\lambda_n(X)/\mu} \end{bmatrix} Q^T$$

where $X = Q \begin{bmatrix} \lambda_1(X) & & \\ & \ddots & \\ & & \lambda_n(X) \end{bmatrix} Q^T$ is an eigendecomposition of X

For linear programming: $\nabla f_\mu(x) = \dfrac{1}{\sum_j e^{-x_j/\mu}} \begin{bmatrix} e^{-x_1/\mu} \\ \vdots \\ e^{-x_n/\mu} \end{bmatrix}$

$$\begin{array}{rl} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \end{array} \quad \equiv \quad \begin{array}{rl} \max & \lambda_{\min}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X\rangle = \text{val} \end{array} \quad \approx \quad \begin{array}{rl} \max & f_\mu(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X\rangle = \text{val} \end{array}$$

Same goal as before:

Compute X for the the projection $Z = Z(X)$ satisfies $\dfrac{\langle C, Z\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \le \epsilon$

Now we rely on Nesterov's optimal method for smooth functions rather than on the subgradient method.

Recall: $k \ge 2\,R\sqrt{\dfrac{\kappa_1}{\epsilon}} \quad \Rightarrow \quad f(x^*) - f(x_k) \le \epsilon$

As before, the algorithm has two phases.

In Phase I, we choose $\mu = \frac{1}{6\,\ln n}$, and in Phase II, $\mu = \frac{\epsilon}{6\,\ln n}$

– hence, in Phase I, $\kappa_1 = 6\,\ln n$, and in Phase II, $\kappa_1 = \frac{6\,\ln n}{\epsilon}$.

Except for being slightly more technical, the analysis proceeds exactly as before.

$$\begin{array}{rl} \min & \langle C, X\rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & X \succeq 0 \end{array} \quad \equiv \quad \begin{array}{rl} \max & \lambda_{\min}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X\rangle = \text{val} \end{array} \quad \approx \quad \begin{array}{rl} \max & f_\mu(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X\rangle = \text{val} \end{array}$$

Same goal as before:

Compute X for the the projection $Z = Z(X)$ satisfies $\dfrac{\langle C, Z\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{opt\_val}} \leq \epsilon$

Theorem: The total number of first-order iterations does not exceed

$$12\sqrt{\ln n} \cdot \text{diam} \cdot \left(\frac{1}{\epsilon} + \log_{5/4}\left(\frac{\langle C, I\rangle - \text{opt\_val}}{\langle C, I\rangle - \text{val}_0} \right) \right)$$

objective value of input matrix U_0

Corollary: Assume I is on the central path.

If the initial matrix is chosen as $U_0 = I - \frac{5}{6}\frac{1}{\lambda_{\max}(\pi(C))}\pi(C)$,

projection of C onto nullspace of $\mathcal{A}$

then the total number of first-order iterations does not exceed

$$12\sqrt{\ln n} \cdot \text{diam} \cdot \left(\frac{1}{\epsilon} + \log_{5/4}(n) + 1 \right)$$

Compare with interior-point methods: $O(\sqrt{n}\,\log(1/\epsilon))$ iterations

$$\left.\begin{array}{rl} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array}\right\} \text{LP} \quad \leftrightarrow \quad \begin{array}{rl} \max & \min_j x_j \\ \text{s.t.} & Ax = b \\ & c^T x = \text{val} \end{array}$$

$$\left.\begin{array}{rl} \min & \langle C, X \rangle \\ \text{s.t.} & \mathcal{A}(X) = b \\ & x \succeq 0 \end{array}\right\} \text{SDP} \quad \leftrightarrow \quad \begin{array}{rl} \max & \lambda_{\min}(X) \\ \text{s.t.} & \mathcal{A}(X) = b \\ & \langle C, X \rangle = \text{val} \end{array}$$

Disclaimers:

- No claim is made for the approach being an algorithmic advance
  for problems where $O(1/\sqrt{\epsilon})$ algorithms have been devised.

However, there are important closely-related problems
for which it is an advance ...

- No claim is even made for the approach being an algorithmic advance
  for some problems where $O(1/\epsilon)$ algorithms already have been devised.

Indeed, Nesterov's 2004 approach to smoothing
definitely is computationally superior for some problems,
even if it is far more difficult to understand
and is far less general.

- Definitely no claim is made that the specific algorithms developed herein
  are the best approaches for utilizing the framework.

The point was just to show that well-known first-order methods
can be straightforwardly utilized
to obtain complexity results of the desired types.

Claim:

- The framework is <u>extremely</u> interesting in that it fits so well
  with first-order methods but has been overlooked until now.
  (This claim, however, is completely obvious.)

Thanks for listening!