CSE 421

Introduction to Algorithms

Lecture 26: Linear Programming Algorithms

Standard Form LP

Maximize $\boldsymbol{c}^\top \boldsymbol{x}$
subject to

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$
$$\boldsymbol{x} \geq \boldsymbol{0}$$

Algorithms for Linear Programs

Simplex Algorithm (1947) "local search"

- Simple
- Often fast in practice
- Not polynomial time (on pathological counterexamples)

Math 407

Ellipsoid Algorithm Khachiyan 1979 "Divide and Conquer"

- More complicated
- First polynomial time algorithm, but not always fast

Interior Point Methods Karmarkar 1983

- Even more complicated based on differential equation ideas
- Polynomial time, fast in practice; simplex better for small input size

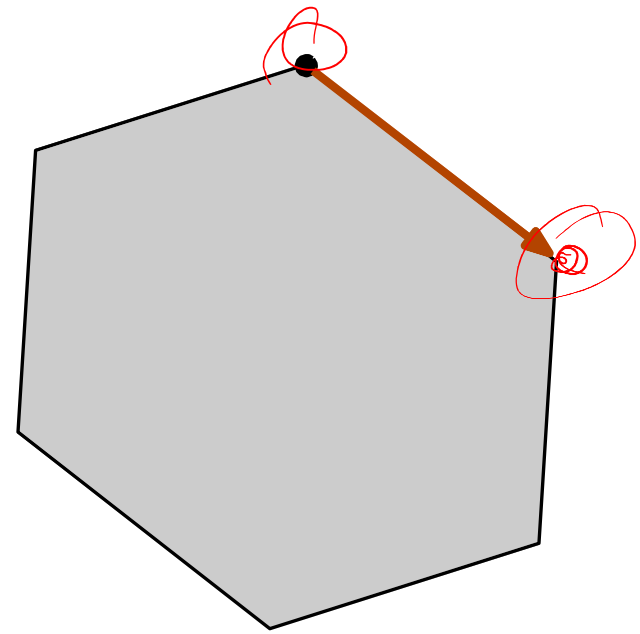
The Simplex Algorithm

Simplex Algorithm:

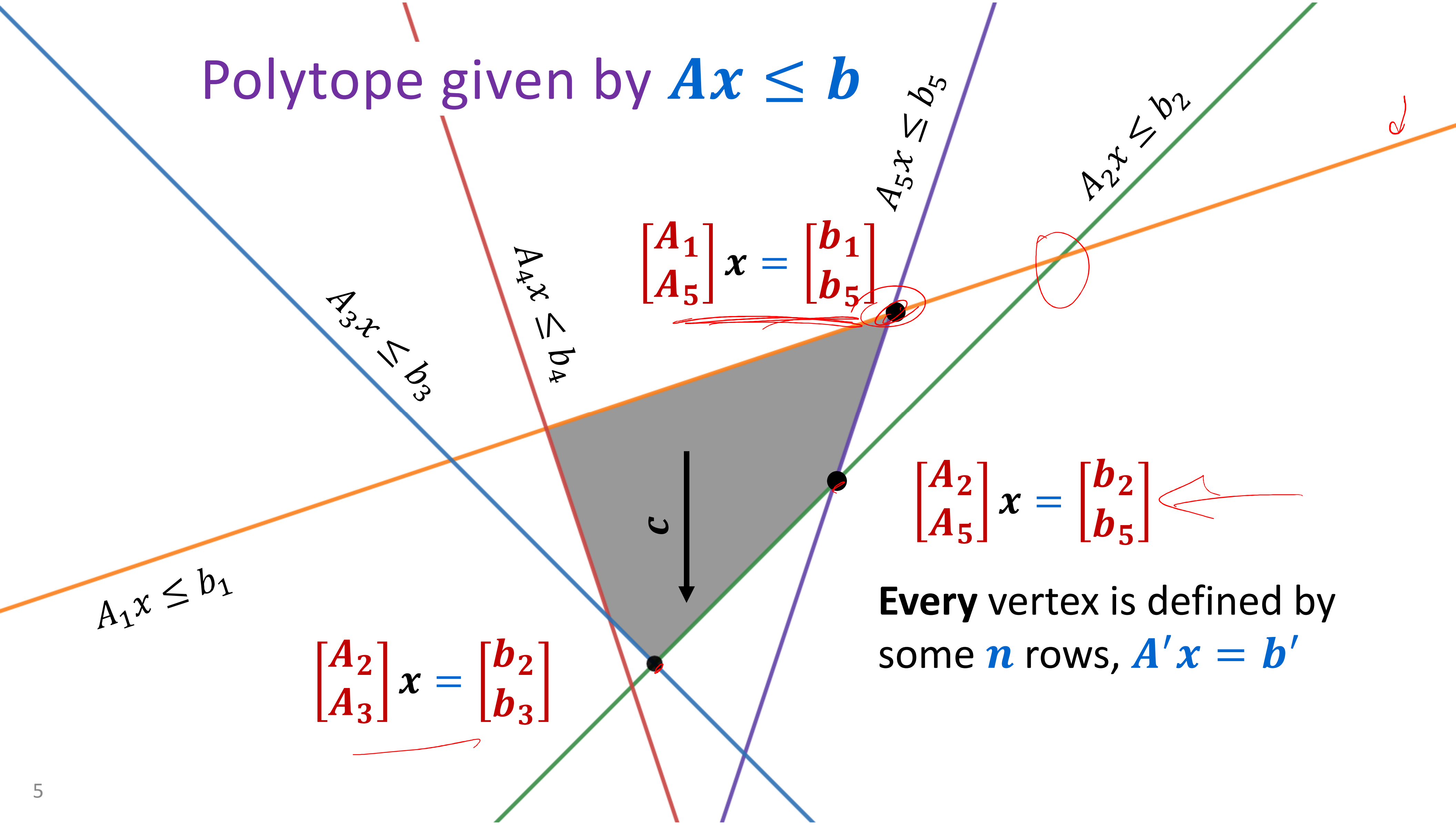
- Start with a vertex of the polytope
- In each step move to a neighboring vertex that is *lower* (larger $\boldsymbol{c}^\top \boldsymbol{x}$).

Creates a path running along the edges and vertices on the outside of the polytope

- Since the polytope is convex, this will never get stuck before reaching the lowest point.



Polytope given by $\boldsymbol{Ax \leq b}$



Every vertex is defined by some $\boldsymbol{n}$ rows, $\boldsymbol{A'x = b'}$

Simplex: How to find the start vertex

$b = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$

We can't just choose any subset of $\boldsymbol{n}$ equations since their solution might not be in the polytope $\boldsymbol{P}$...

Given

Maximize $\boldsymbol{c}^\top \boldsymbol{x}$
subject to
$$\boldsymbol{A x} \leq \boldsymbol{b}$$
$$\boldsymbol{x} \geq \mathbf{0}$$

$\Rightarrow$

Solve **P**

Find point s.t.
$$\boldsymbol{A x} \leq \boldsymbol{b}$$
$$\boldsymbol{x} \geq \mathbf{0}$$

$\Leftrightarrow$

Solve **Q**

Minimize $\underset{=0}{\boldsymbol{z}_1} + \cdots + \underset{=0}{\boldsymbol{z}_m}$
subject to
$$\boldsymbol{A x} - \boldsymbol{z} \leq \boldsymbol{b}$$
$$\boldsymbol{x}, \boldsymbol{z} \geq \mathbf{0}$$

and $z = 0$

Polytope $\boldsymbol{P}$ is not empty iff minimum for $\boldsymbol{Q}$ has $\boldsymbol{z} = \mathbf{0}$ and $\boldsymbol{x} \in \boldsymbol{P}$

Q is just another LP, but we set it up so we know a start vertex:
$\boldsymbol{x} = \mathbf{0}$ and $\boldsymbol{z} = \max(\mathbf{0}, -\boldsymbol{b})$ so we can use Simplex on **Q** to find the start vertex for the given LP and then run Simplex again!

$-z \leq b \Rightarrow z \geq -b$

$z \geq 0$

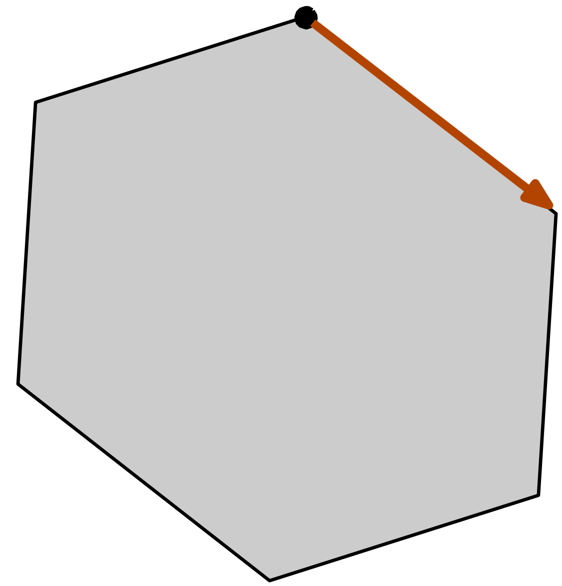
The Simplex Algorithm

Simplex Algorithm:

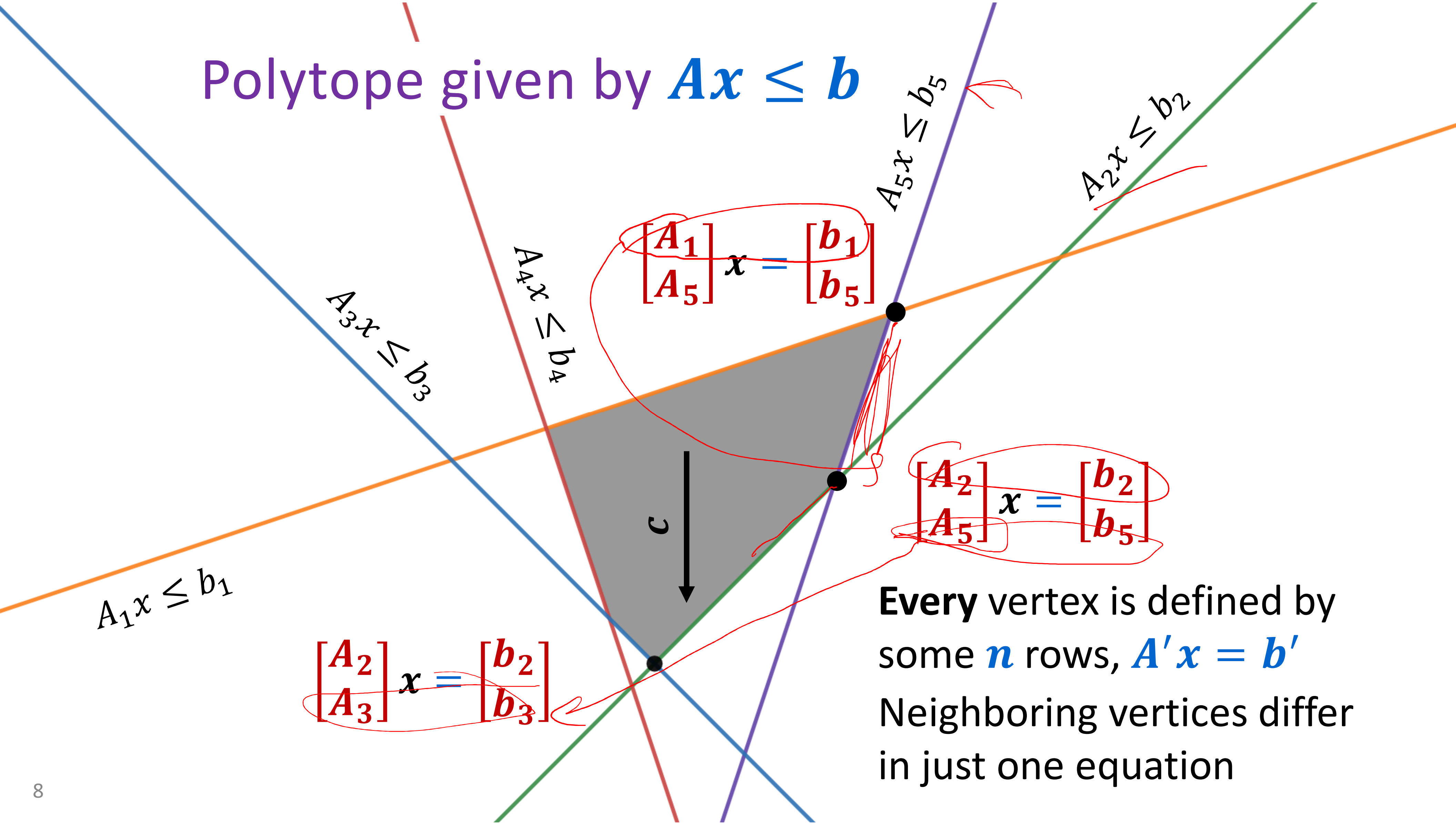
- Start with a vertex of the polytope
- In each step move to a neighboring vertex that is *lower* (larger $\boldsymbol{c}^\top \boldsymbol{x}$).

Creates a path running along the edges and vertices on the outside of the polytope

- Since the polytope is convex, this will never get stuck before reaching the lowest point.



Polytope given by $\boldsymbol{Ax} \leq \boldsymbol{b}$



Every vertex is defined by some $\boldsymbol{n}$ rows, $\boldsymbol{A'x = b'}$

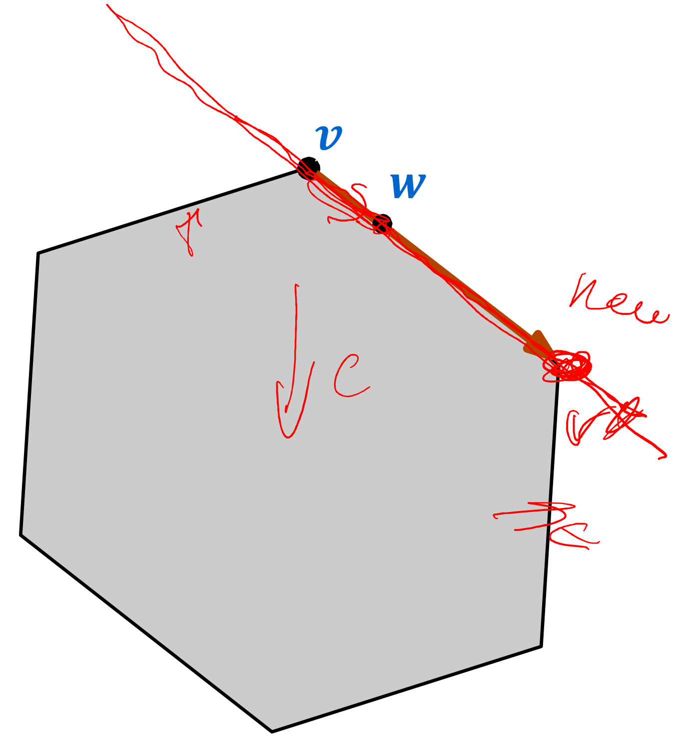
Neighboring vertices differ in just one equation

Simplex: Moving to a better vertex

Maximize $\boldsymbol{c}^\top \boldsymbol{x}$
subject to

$$\boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}$$
$$\boldsymbol{x} \geq \boldsymbol{0}$$

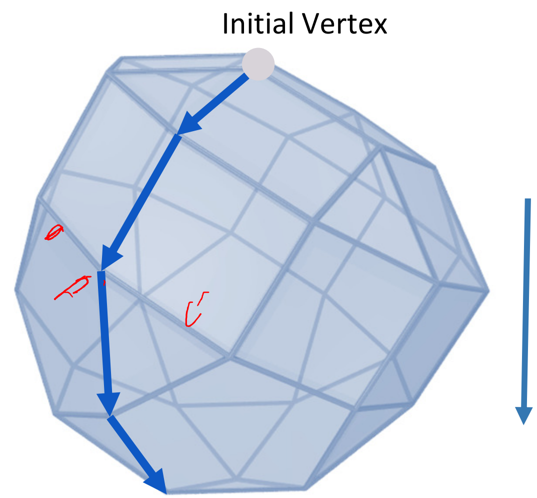
1. At current vertex have $\boldsymbol{n}$ tight equations $\boldsymbol{A}'\boldsymbol{v} = \boldsymbol{b}'$
2. Can find $\boldsymbol{1}$ equation to replace and a point $\boldsymbol{w}$
   - satisfying the other $\boldsymbol{n-1}$
   - with $\boldsymbol{c}^\top(\boldsymbol{w}-\boldsymbol{v}) > \boldsymbol{0}$.
3. Move to new vertex of form $\boldsymbol{v}' = \boldsymbol{v} + \boldsymbol{\delta w} \in \boldsymbol{P}$ for $\boldsymbol{\delta} > \boldsymbol{0}$
   - Increase $\boldsymbol{\delta}$ until some new constraint becomes tight.



The Simplex Algorithm: The downside

Simplex Algorithm:
- Start with a vertex of the polytope
- In each step move to a neighboring vertex that is *lower* (larger $\boldsymbol{c}^\top \boldsymbol{x}$).

Creates a path running along the edges and vertices on the outside of the polytope
- Since the polytope is convex, this will never get stuck before reaching the lowest point.



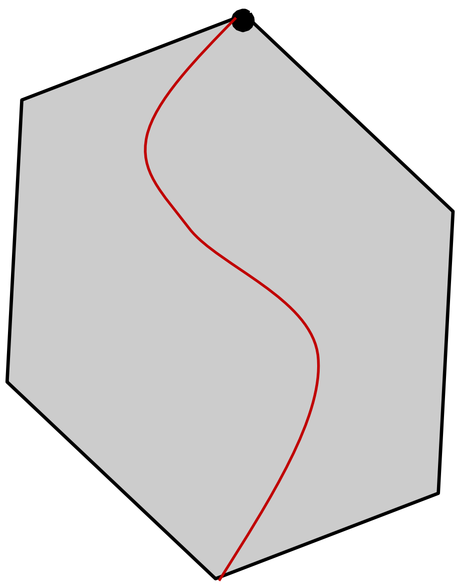
Problem: Many paths to choose from; # of vertices on path can be exponential!

Interior Point Algorithms

Interior Point Idea:

- Start with a point in the polytope, either a vertex or in the interior
- Follow approximations to a curving "central path" that
  - tunnels through the polytope
  - avoids the boundary using loss functions

  and eventually gets to the optimum



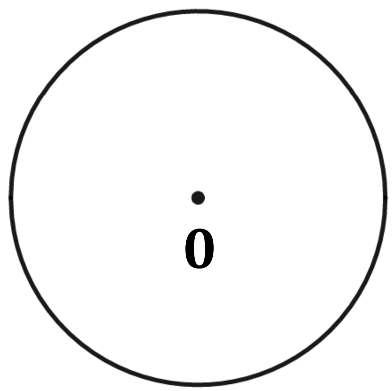
Can be implemented efficiently using data structure tricks. Also leads to best randomized algorithms for network flow. Too complicated for us.

Ellipsoid Method

Ellipsoid:

- A squished ball

$$x^2 + y^2 \leq 1$$



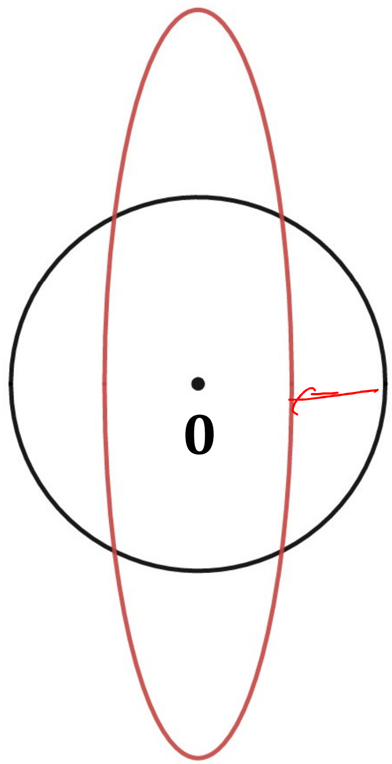
Rati

Ellipsoid Method

Ellipsoid:

- A squished ball

$$x^2 + y^2 \leq 1$$



$$(2x)^2 + (y/2)^2 \leq 1$$

Ratio of ~~area~~ volume of ellipsoid to the sphere

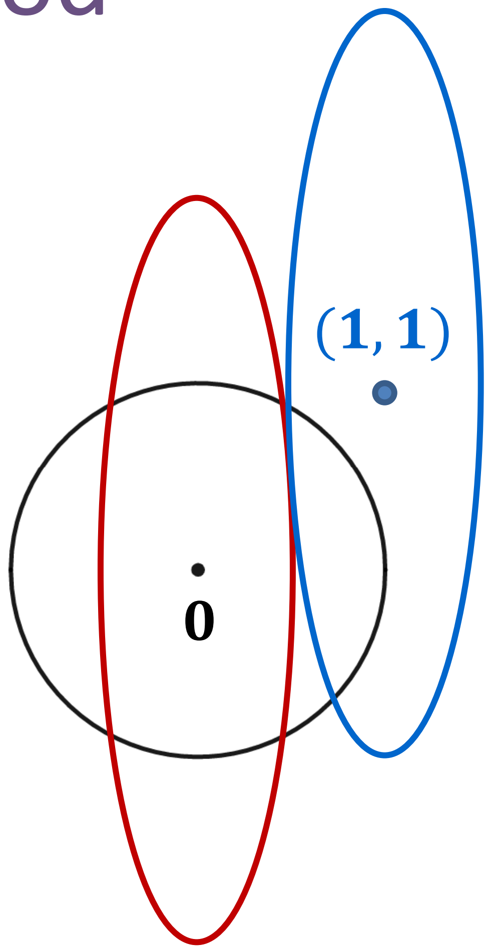
$$\frac{2}{1} \cdot \frac{1}{2} = 1$$

Ellipsoid Method

Ellipsoid:

- A squished ball

$$x^2 + y^2 \leq 1$$



$$(2x)^2 + (y/2)^2 \leq 1$$

Can shift the center

$$2(x-1)^2 + ((y-1)/2)^2 \leq 1$$

Ratio of area of ellipsoid to the sphere

$$\frac{2}{1} \cdot \frac{1}{2} = 1$$

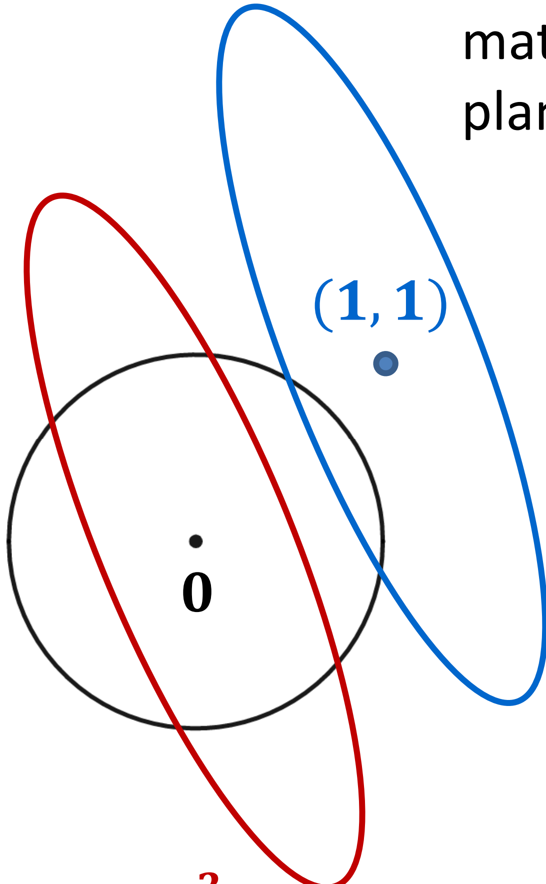
Ellipsoid Method

Let $\boldsymbol{U} = \begin{bmatrix} \boldsymbol{U_1} \\ \boldsymbol{U_2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{u_{11}} & \boldsymbol{u_{12}} \\ \boldsymbol{u_{21}} & \boldsymbol{u_{22}} \end{bmatrix}$ be a matrix that rotates space (here the plane) by some angle $\boldsymbol{\theta}$.

Ellipsoid:

- A squished ball

$$\boldsymbol{x^2 + y^2 \le 1}$$



$$\left(\boldsymbol{2U_1} \begin{bmatrix} \boldsymbol{x - 1} \\ \boldsymbol{y - 1} \end{bmatrix}\right)^{\mathbf{2}} + \left(\boldsymbol{U_2} \begin{bmatrix} \boldsymbol{x - 1} \\ \boldsymbol{y - 1} \end{bmatrix} / \mathbf{2}\right)^{\mathbf{2}} \le \mathbf{1}$$

Rotates the ellipsoid by angle $\boldsymbol{-\theta}$.

$$\left(\boldsymbol{2U_1} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix}\right)^{\mathbf{2}} + \left(\boldsymbol{U_2} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} / \mathbf{2}\right)^{\mathbf{2}} \le \mathbf{1}$$

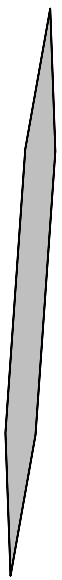
The desired solution is bounded

Theorem: If the LP solution is finite then its magnitude is at most $\mathbf{2}^{\boldsymbol{poly}(\text{input length})}$.

Proof: If the optimum is finite then the solution occurs at a vertex which is the solution of some $\boldsymbol{A'x = b'}$, equivalently $\boldsymbol{x = (A')^{-1}b'}$. The matrix inverse has coordinates with at most # of input bits.

Theorem: If the LP optimum is finite then the volume of the polytope is at least $\mathbf{2}^{-\boldsymbol{poly}(\text{input length})}$.

Proof: General idea: The smallest angle is at least $\mathbf{2}^{-\boldsymbol{poly}(\text{input length})}$.



Ellipsoid Method

Reduce solving:

Maximize $c^\top x$
subject to

$$Ax \le b$$
$$x \ge 0$$

To solving:

Is there an x s.t.

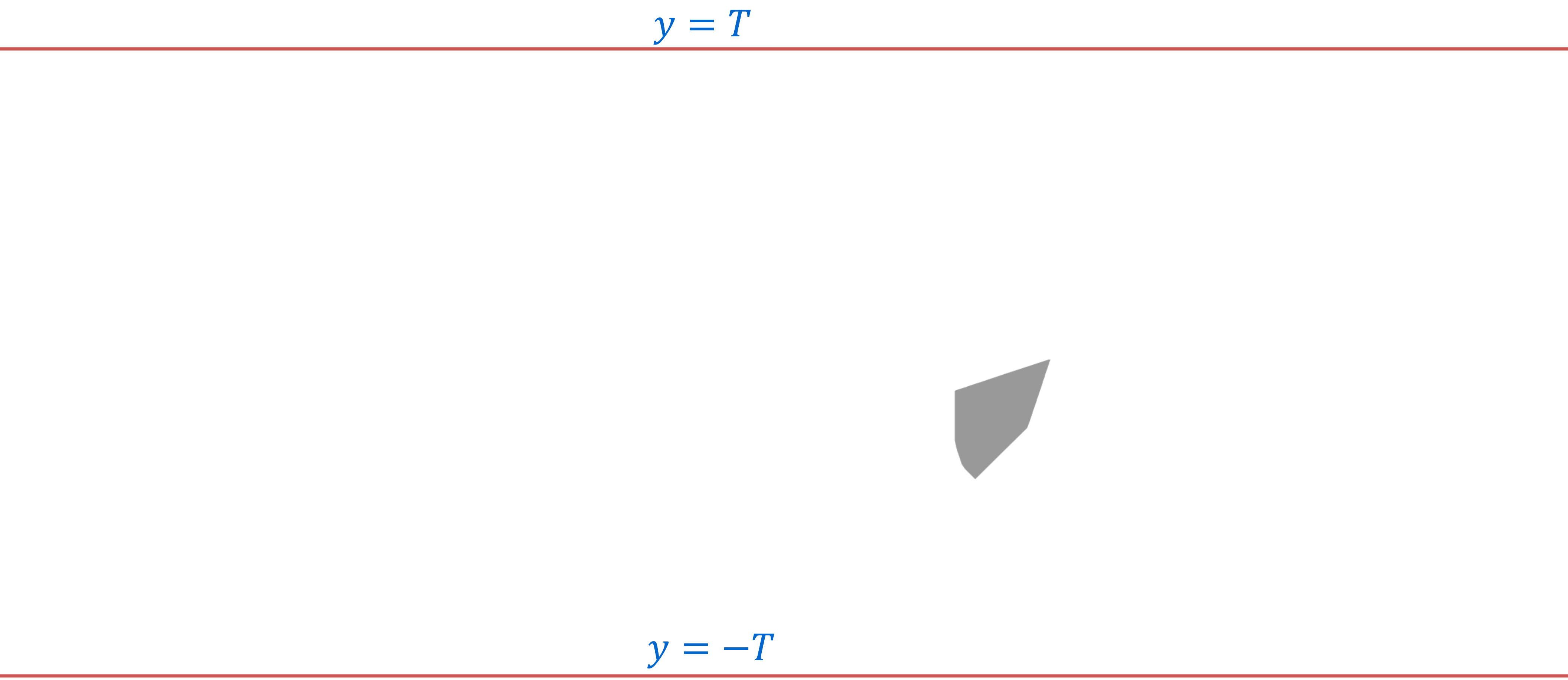
$$c^\top x \ge d$$
$$Ax \le b$$
$$x \ge 0 \ ?$$

Theorem: If the LP solution is finite then its magnitude is at most 2^{poly}(input length).

Corollary: In polytime we can compute $T \in 2^{poly}$(input length) such that if the LP optimum x is finite then $-T \le c^\top x \le T$.

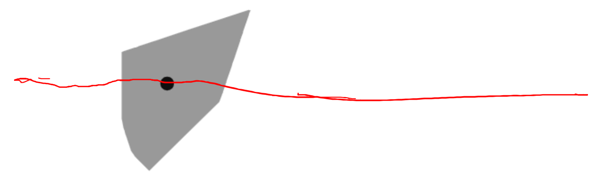
Claim: If we have a polynomial time algorithm ~~FindPoint to~~ find ~~some~~ point x inside any given polytope then we can solve LPs in polynomial time using binary search with different values of d as above.
(Only $poly$(input length) calls.)

Using binary search



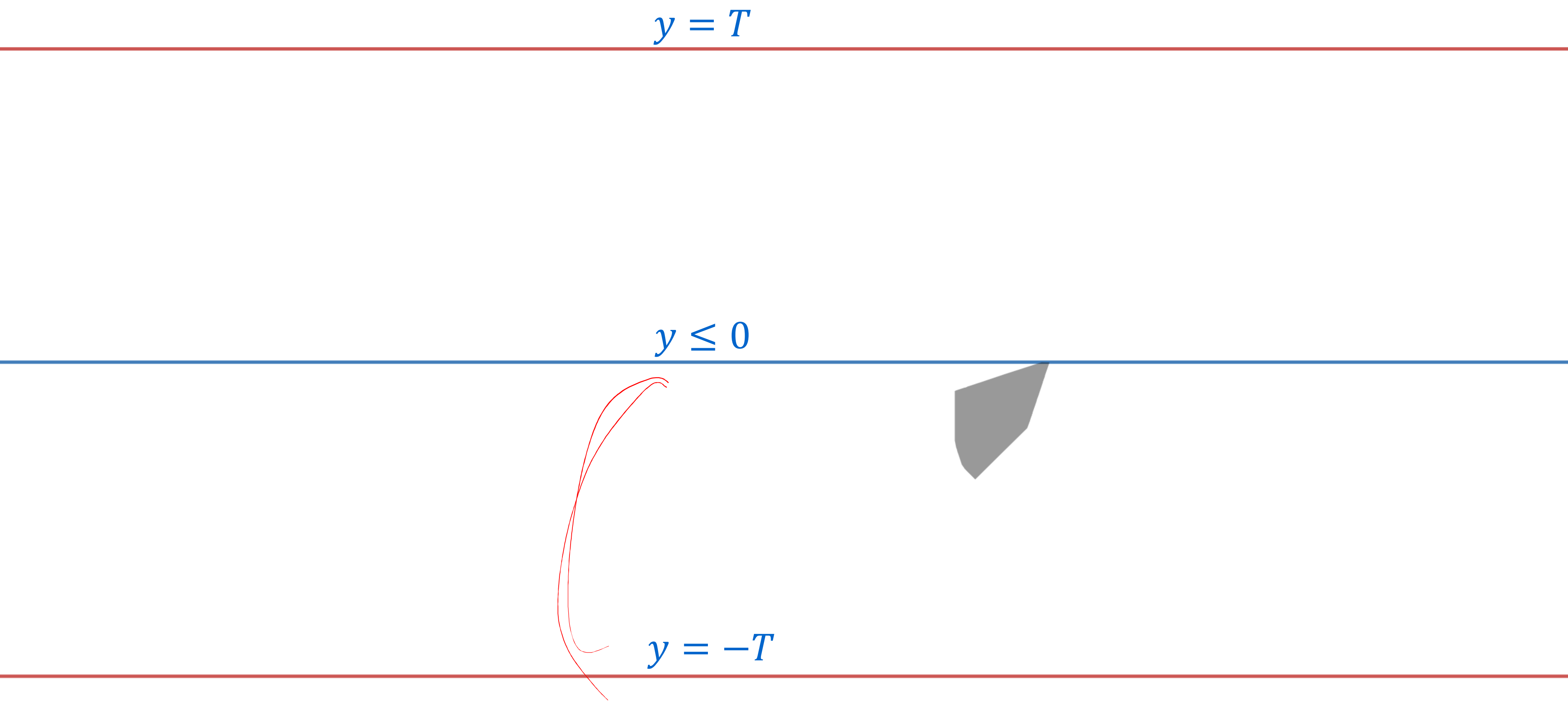
Check if polytope is empty using ~~FindPoint~~

$y = T$

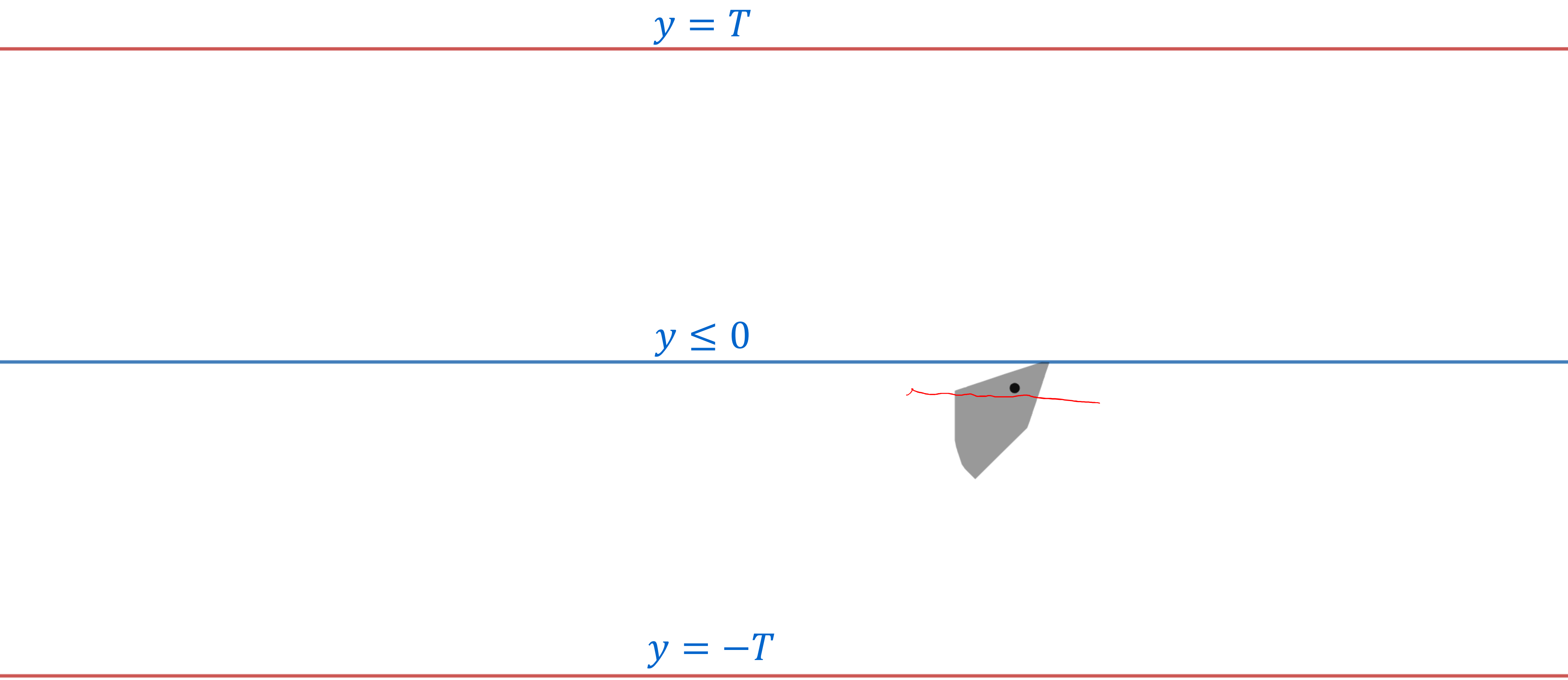


$y = -T$

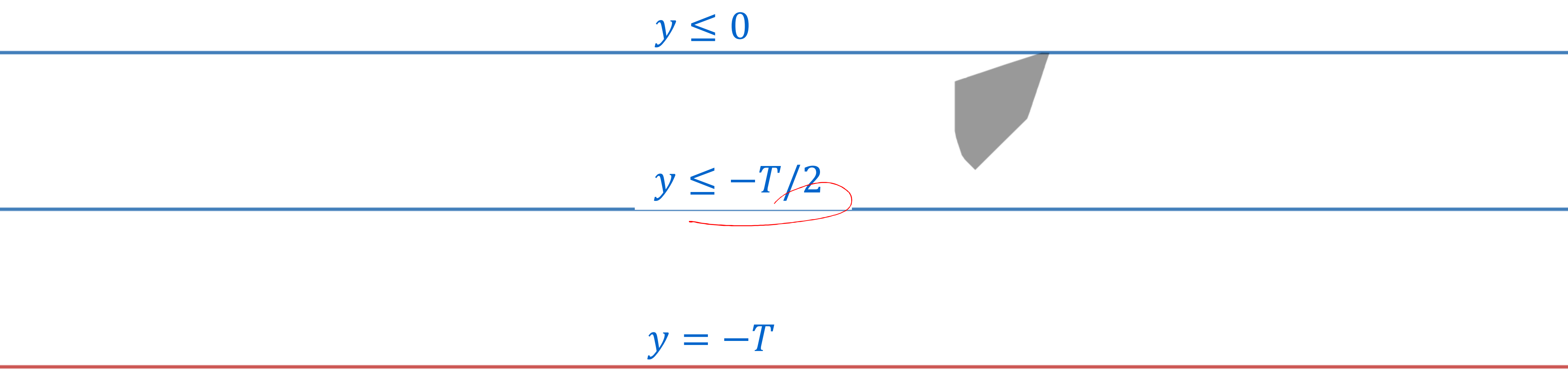
Add new constraint



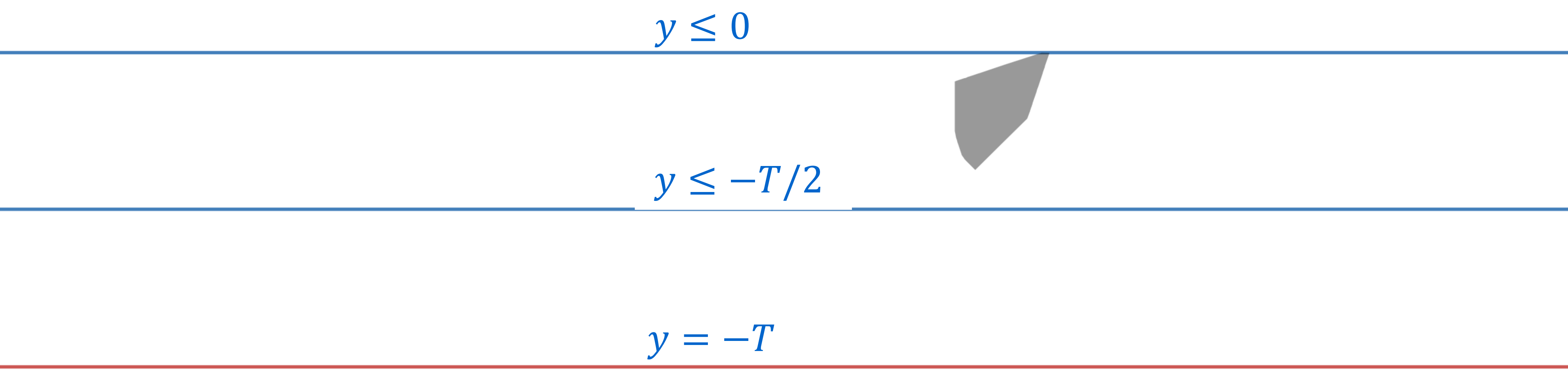
Call FindPoint



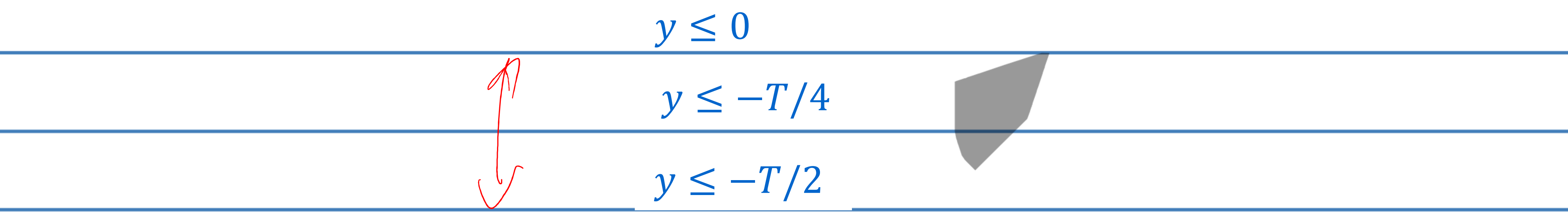
Add new constraint



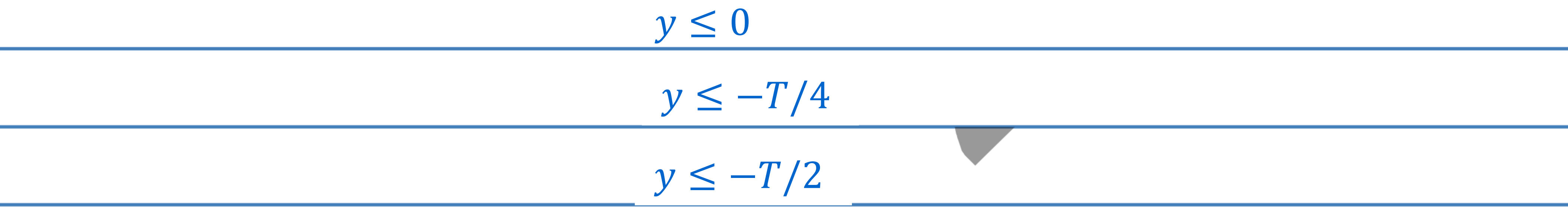
FindPoint: Polytope is empty!



Add new constraint



Add new constraint



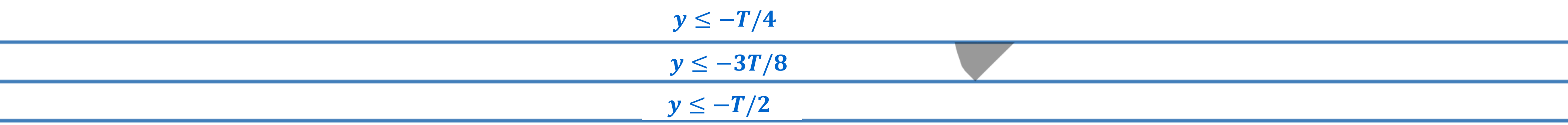
Find point

$y \leq 0$

$y \leq -T/4$

$y \leq -T/2$

Add new constraint



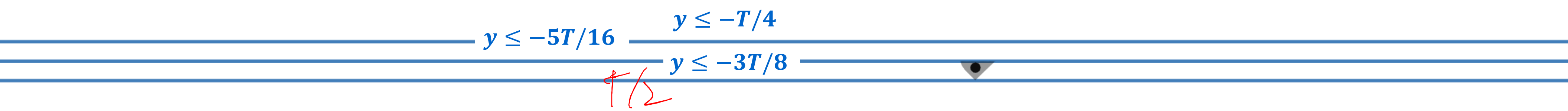
Find point: Polytope is empty!

$y \leq -T/4$

$y \leq -3T/8$

$y \leq -T/2$

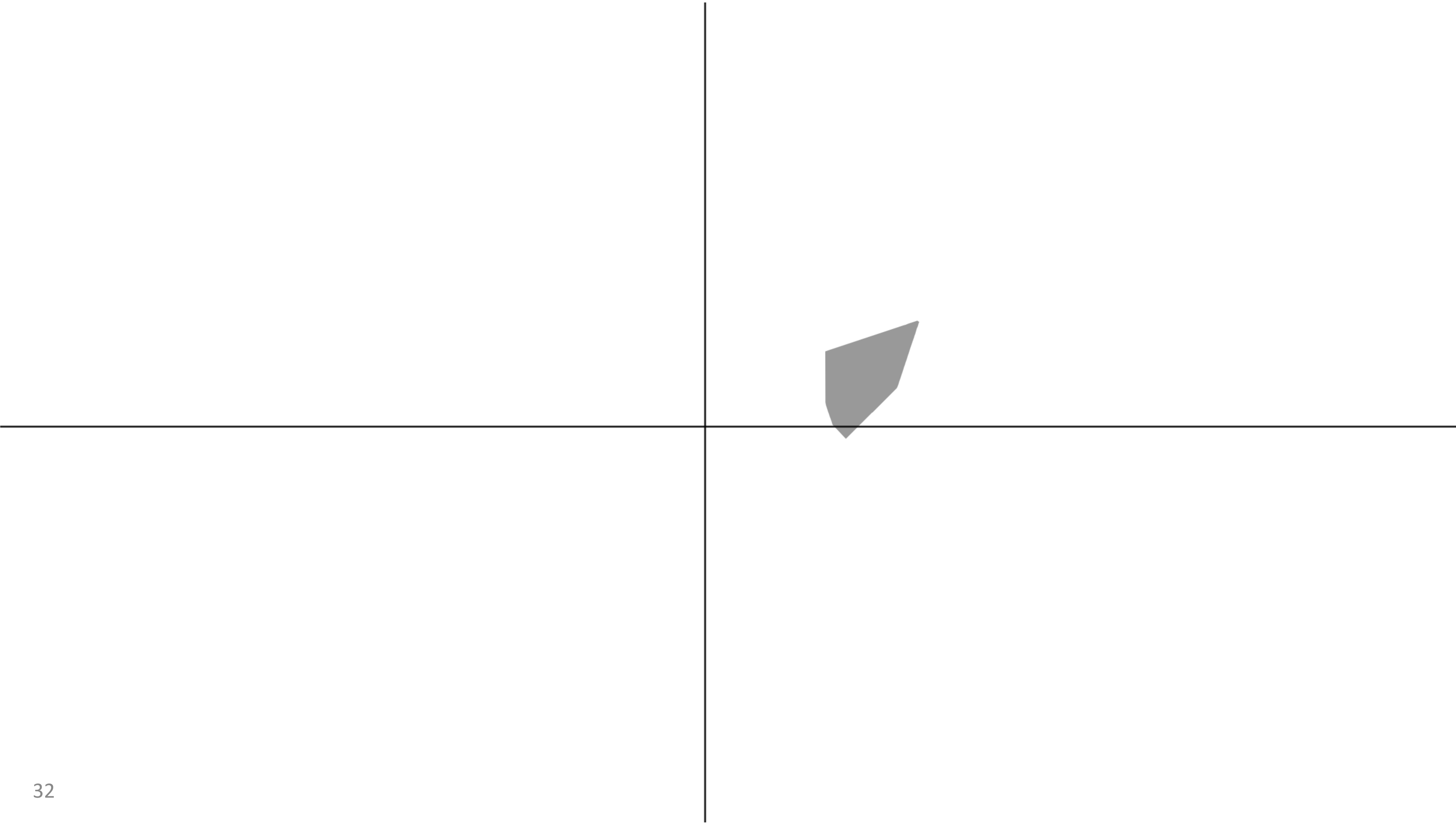
Add new constraint... Find point ...

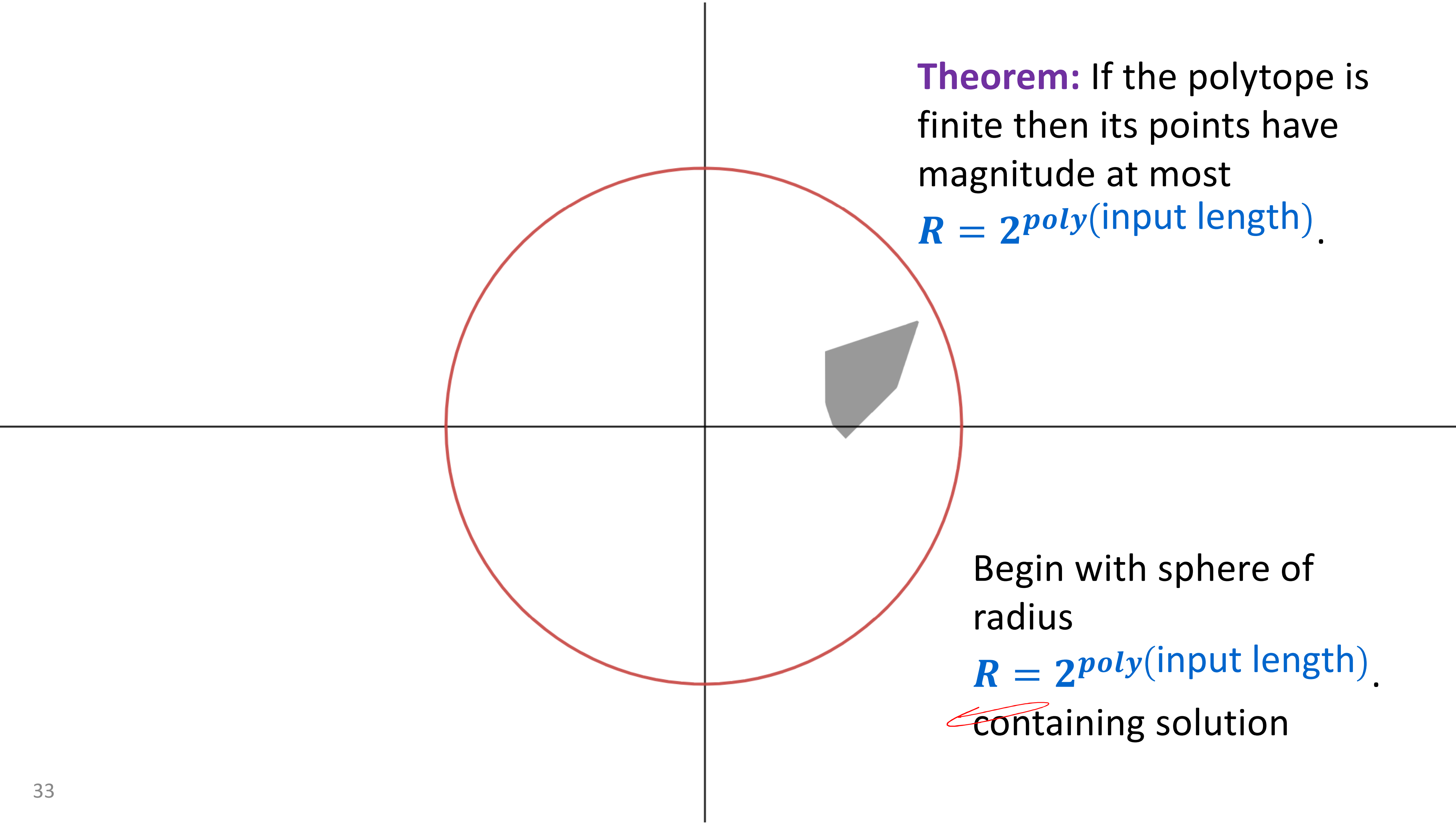


Conclusion: It is enough to give an algorithm to find a point in a polytope.

Ellipsoid algorithm for finding points in polytopes

Idea: Iteratively find ellipsoids where the density of the polytope within each ellipsoid is larger and larger, until a point is found

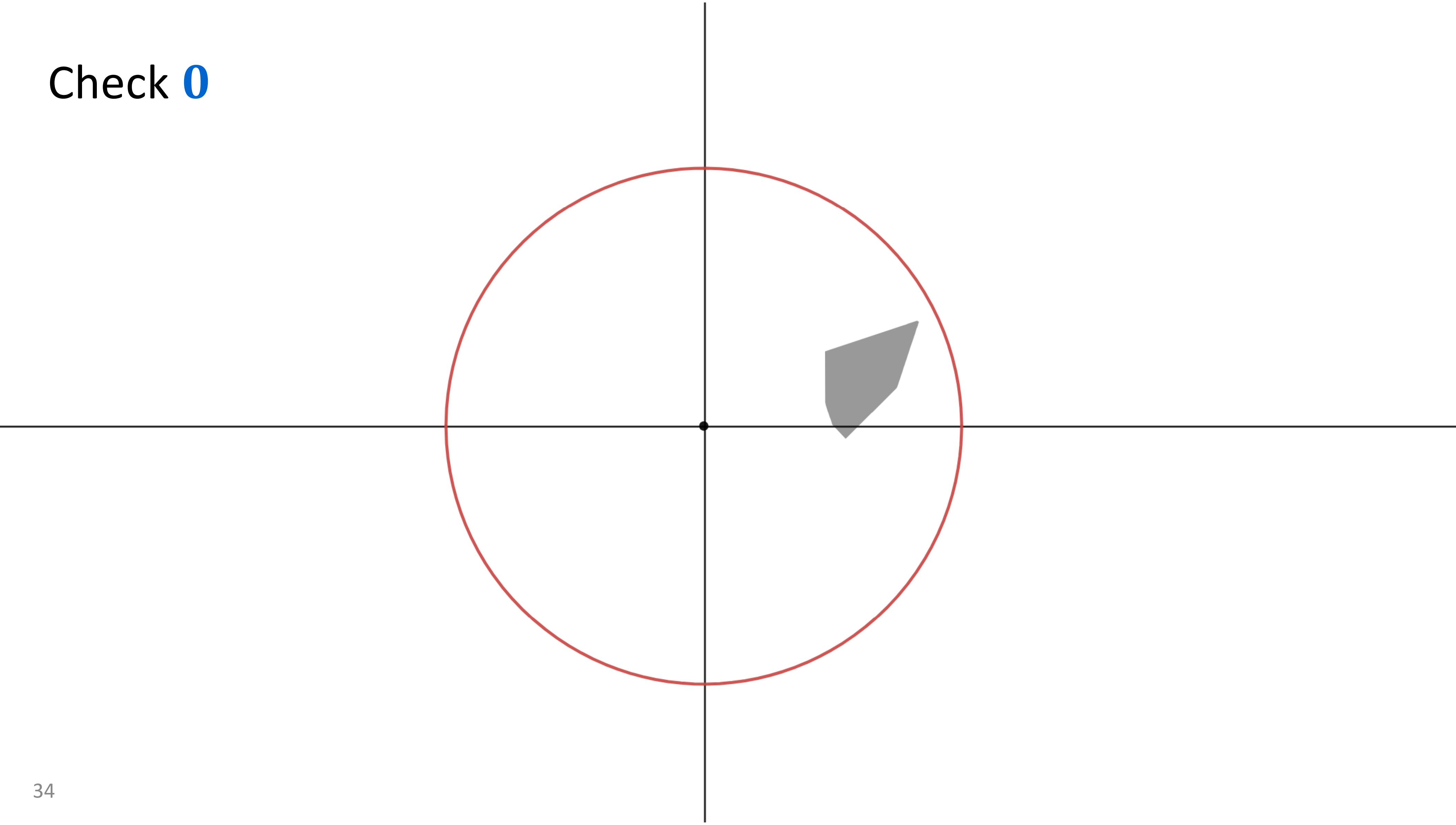




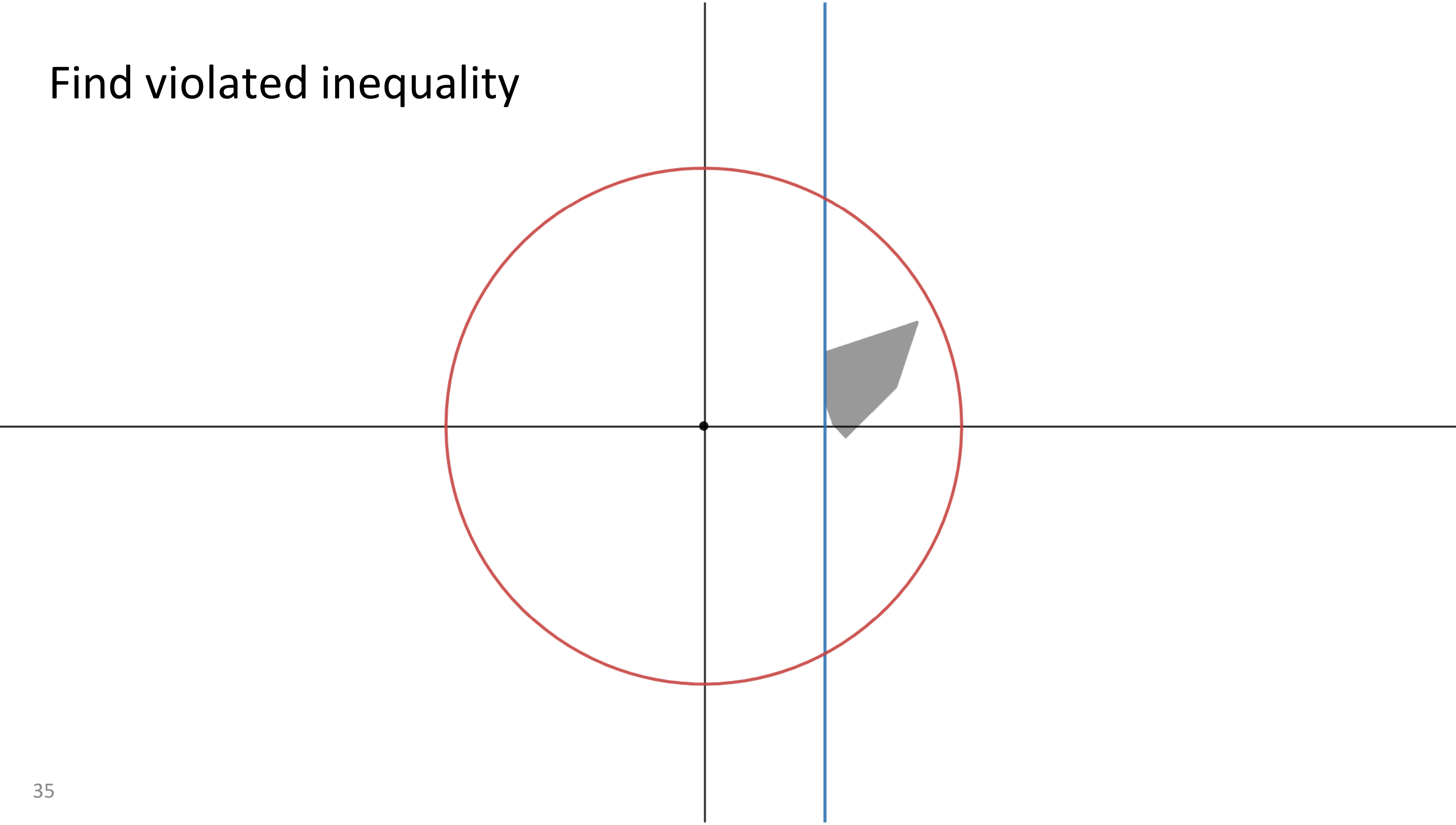
Theorem: If the polytope is finite then its points have magnitude at most $\boldsymbol{R = 2^{poly}}\text{(input length)}$.

Begin with sphere of radius $\boldsymbol{R = 2^{poly}}\text{(input length)}$.
containing solution

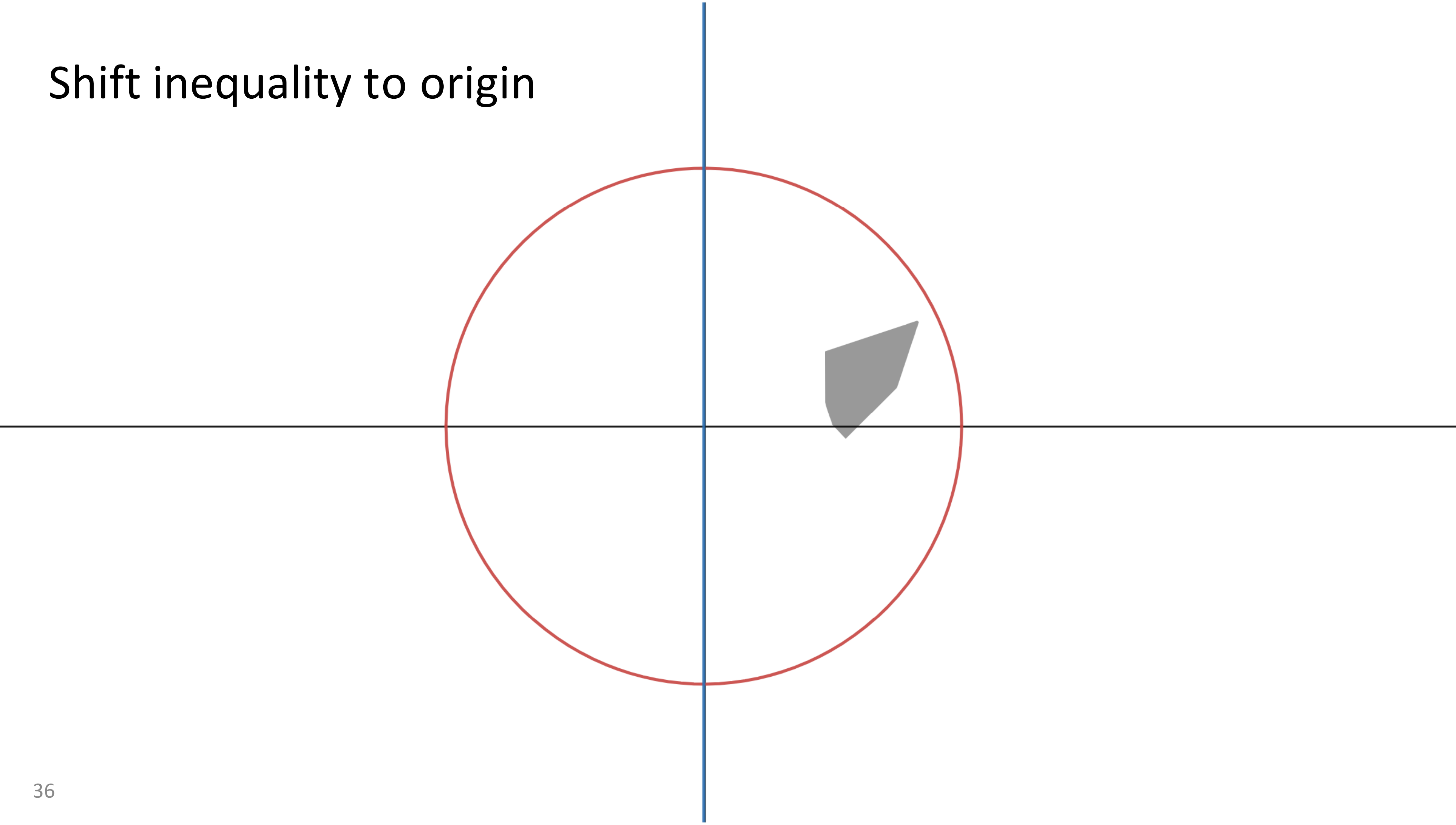
Check **0**



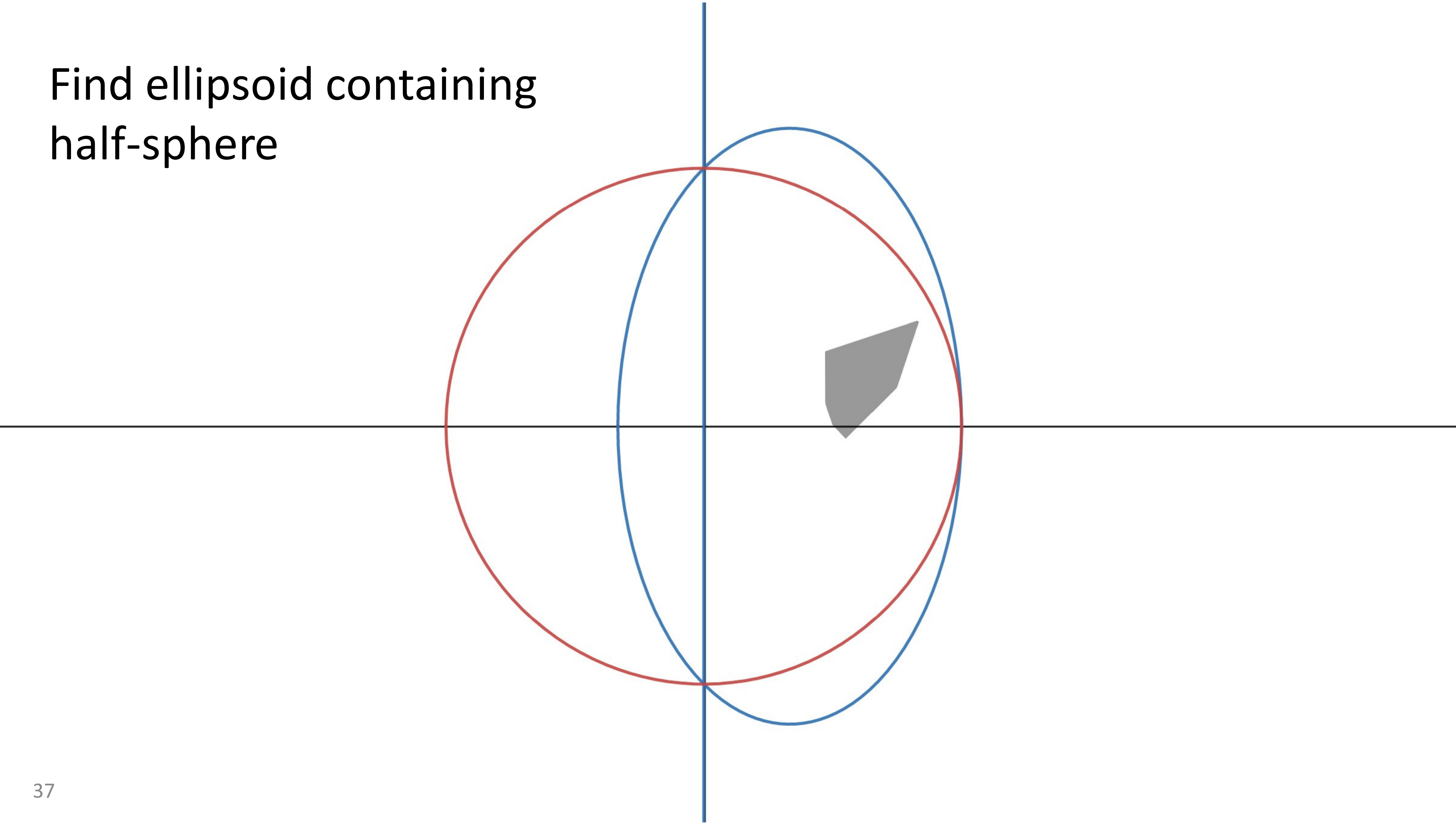
Find violated inequality



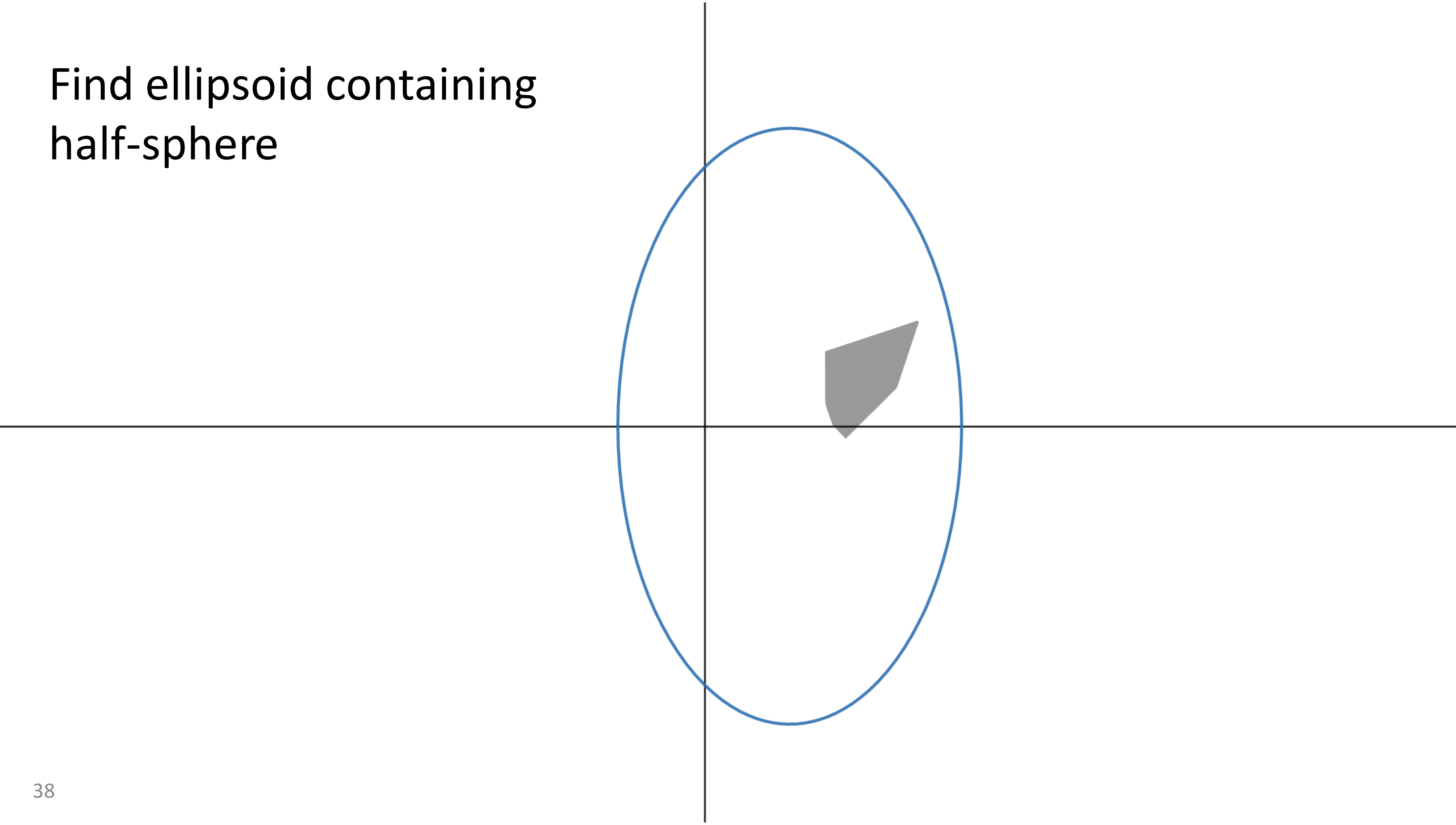
Shift inequality to origin



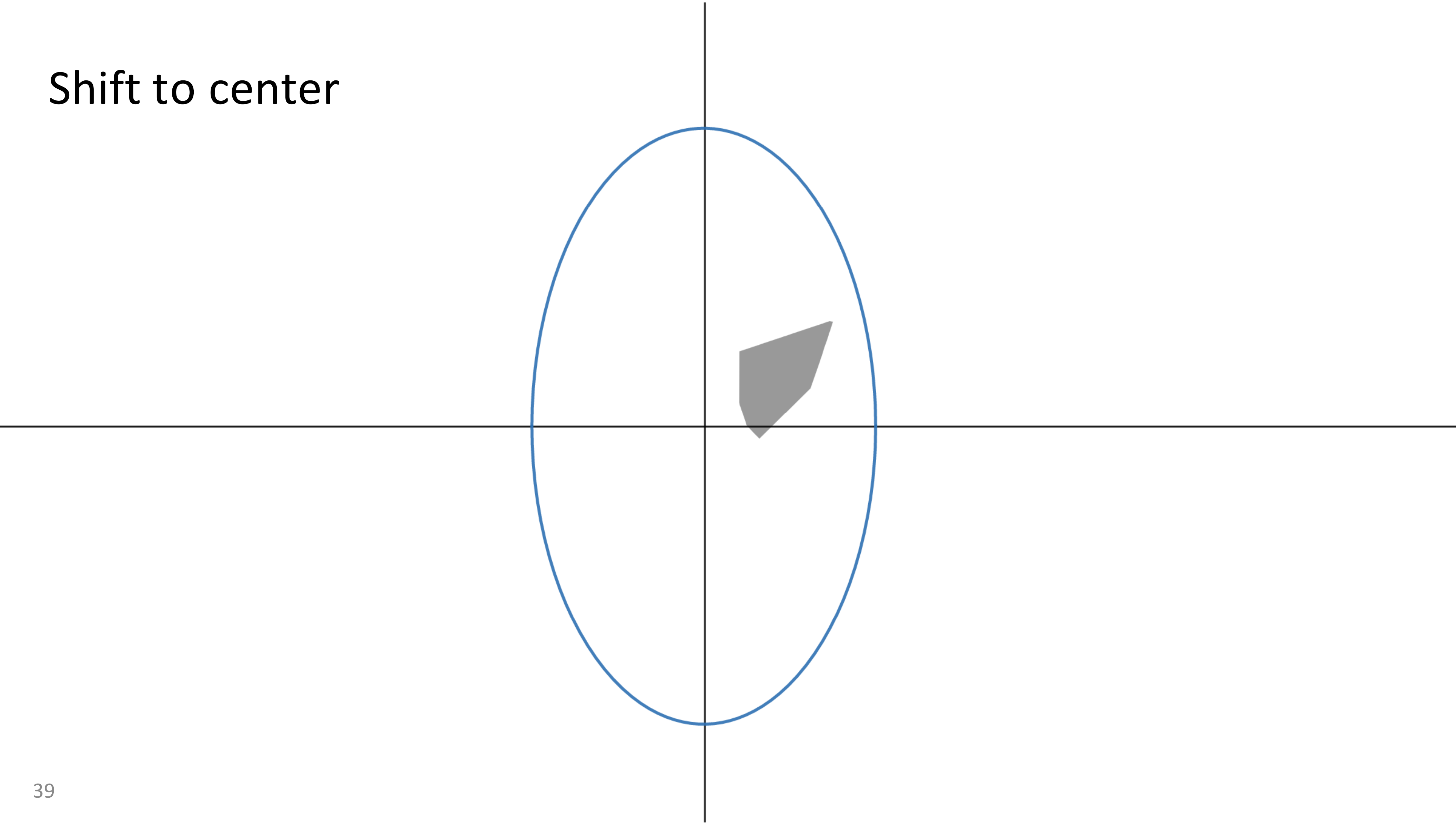
Find ellipsoid containing half-sphere



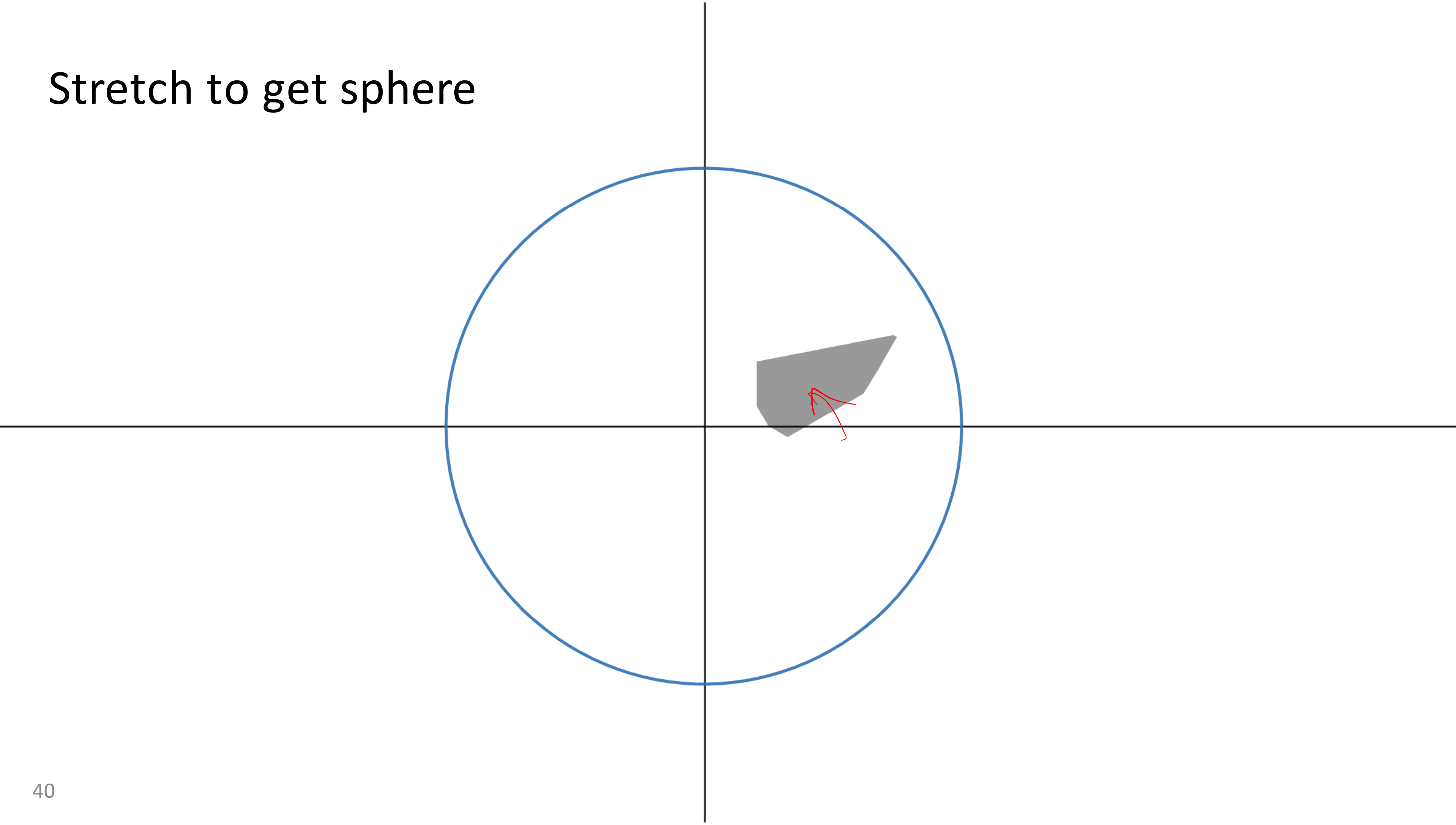
Find ellipsoid containing half-sphere



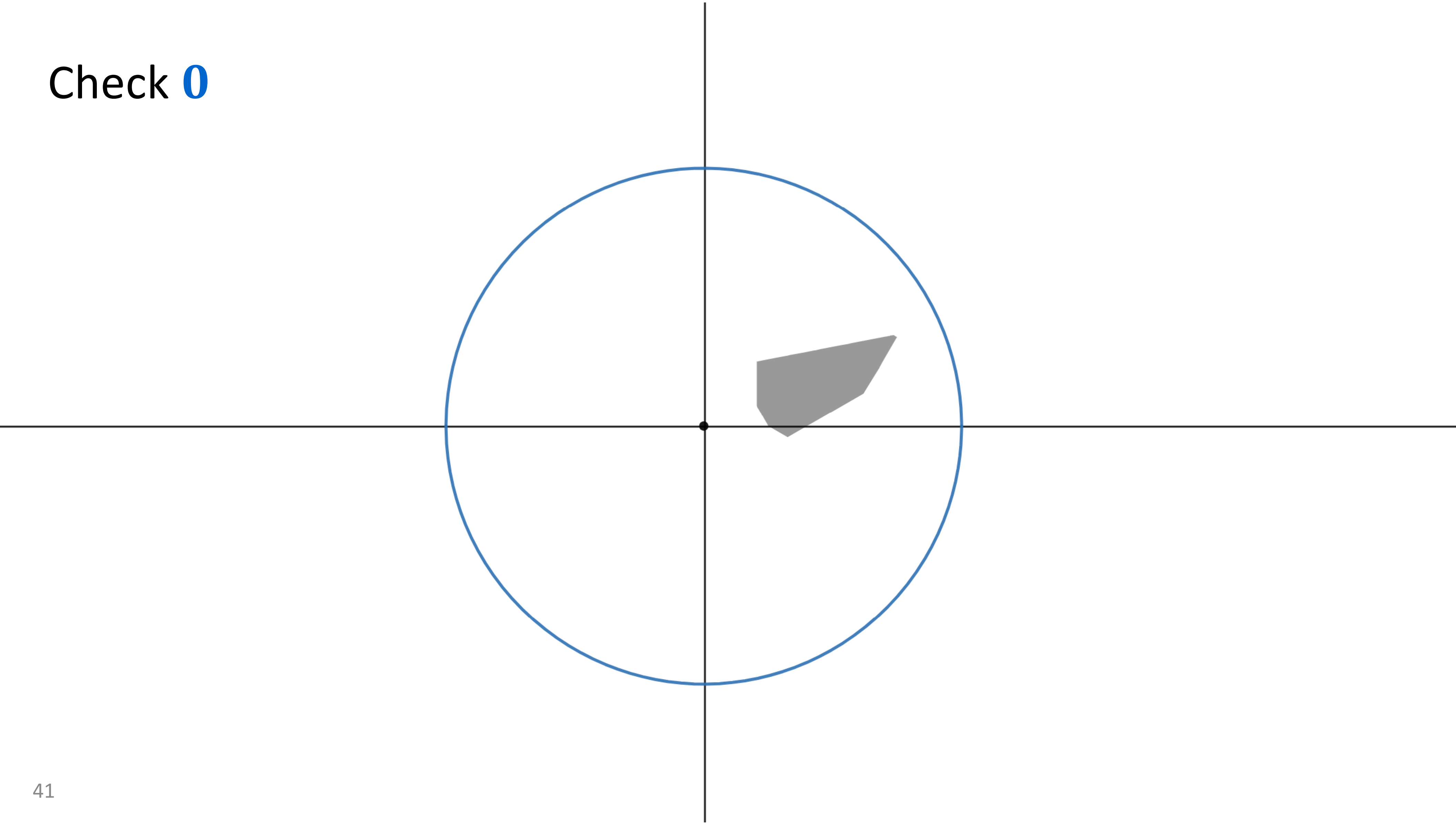
Shift to center



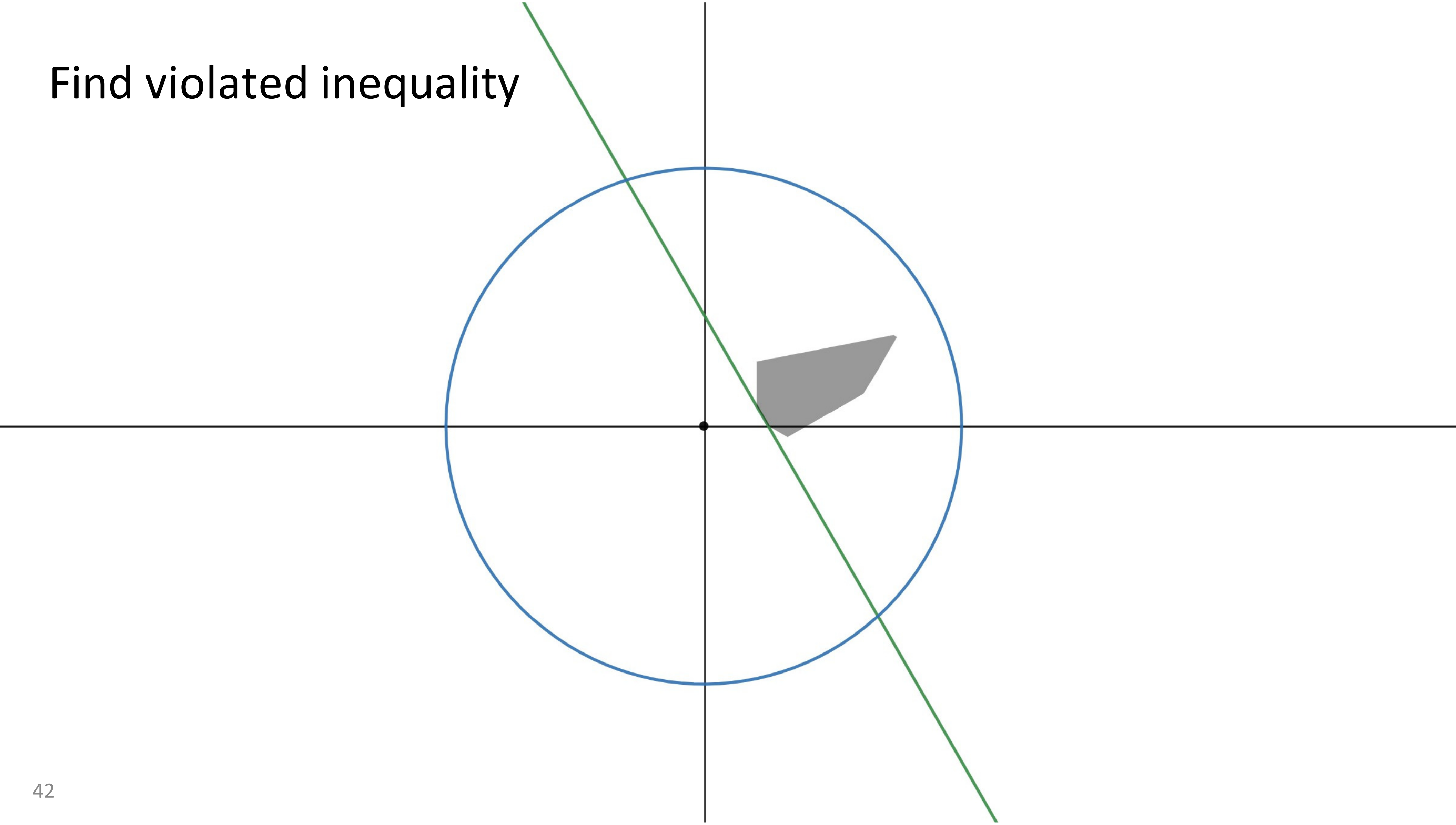
Stretch to get sphere



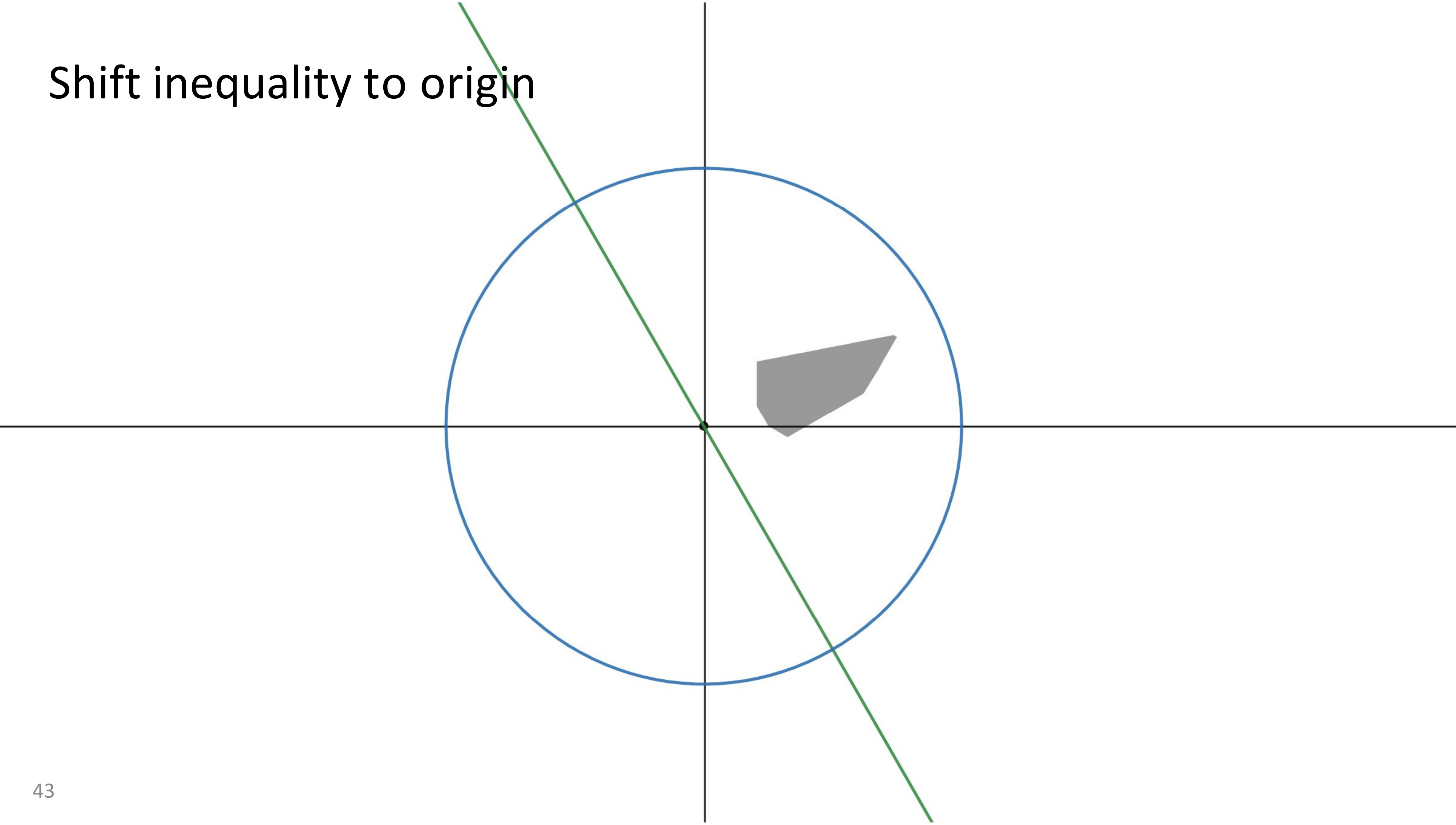
Check **0**



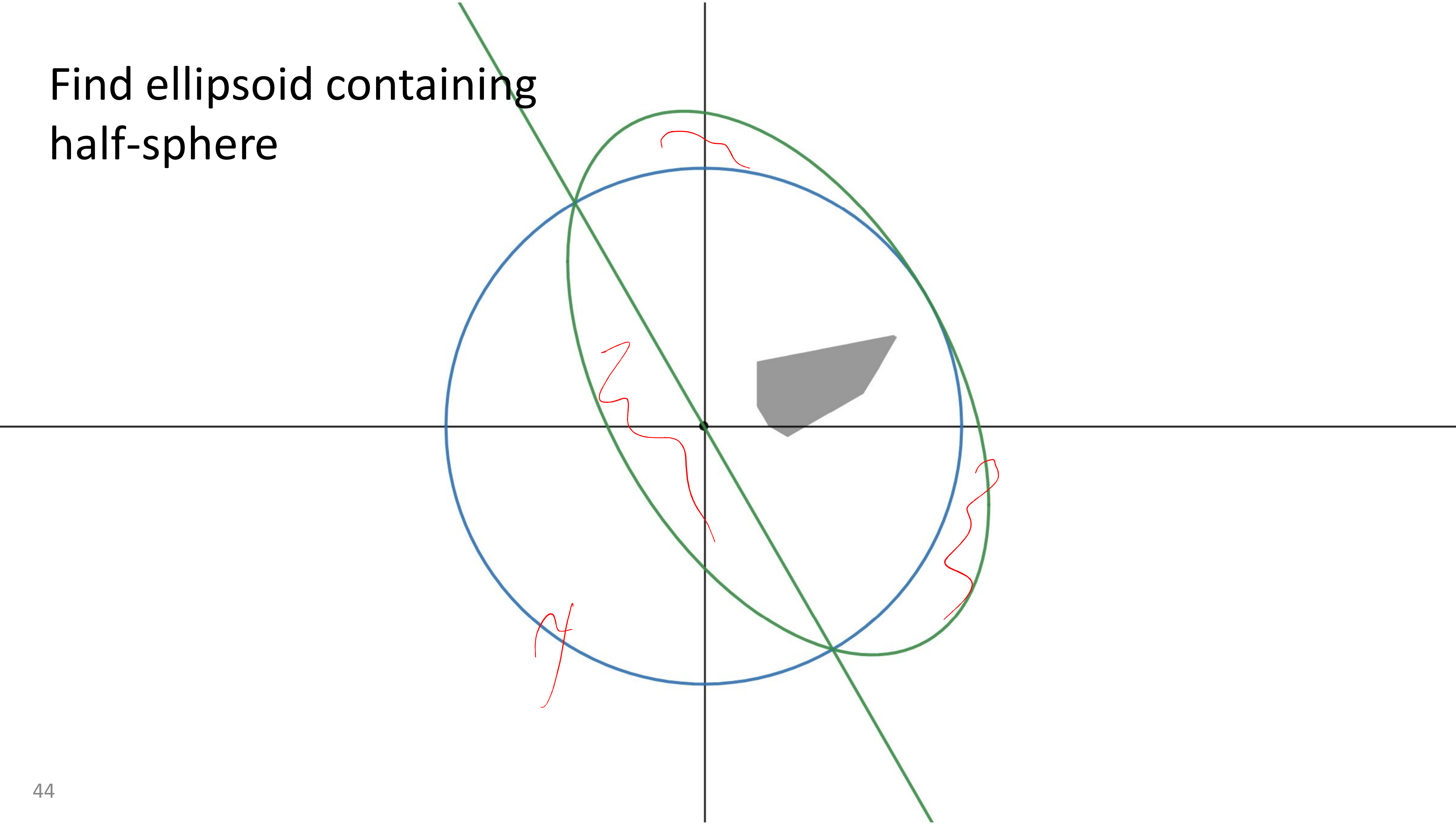
Find violated inequality



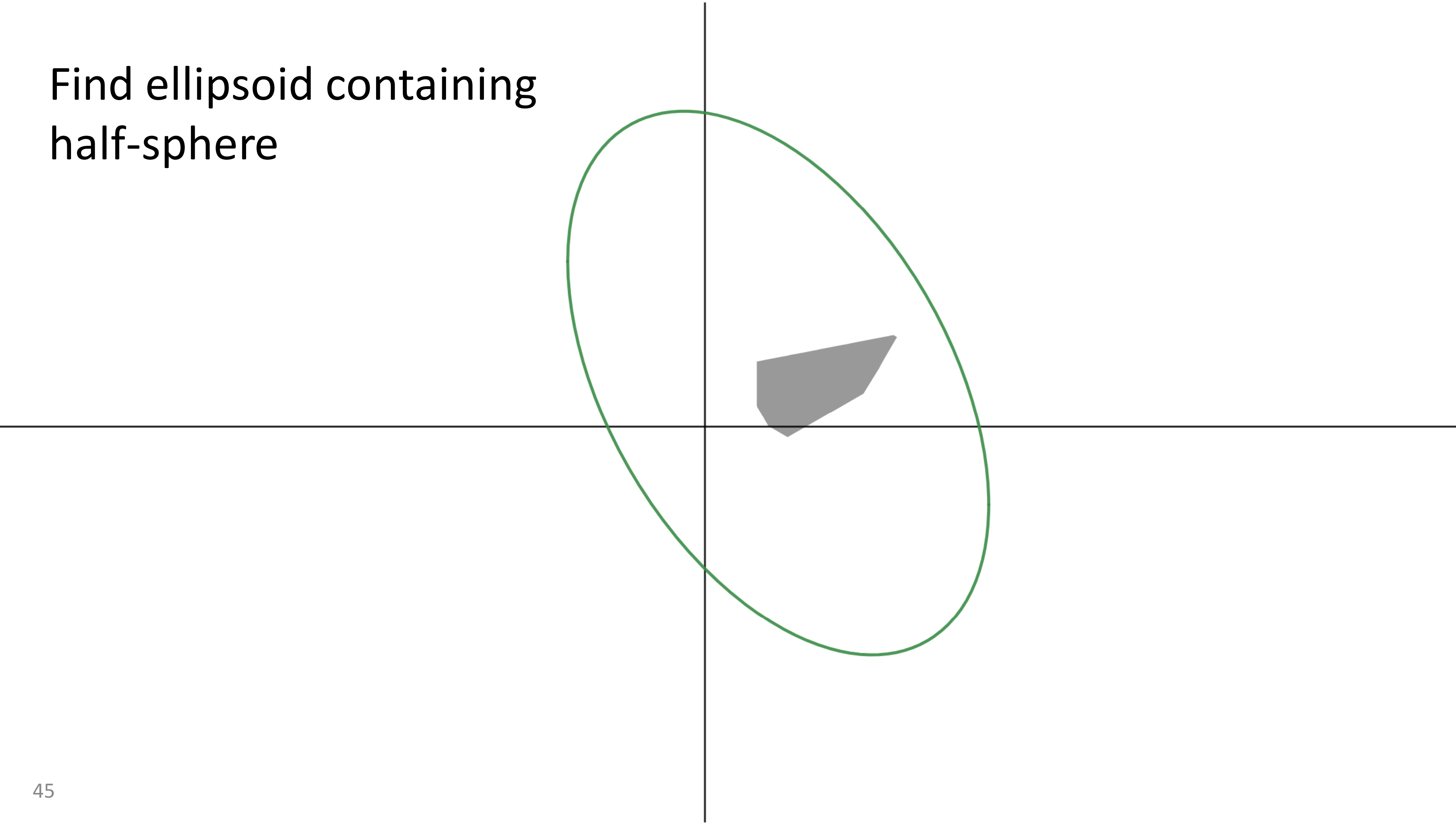
Shift inequality to origin



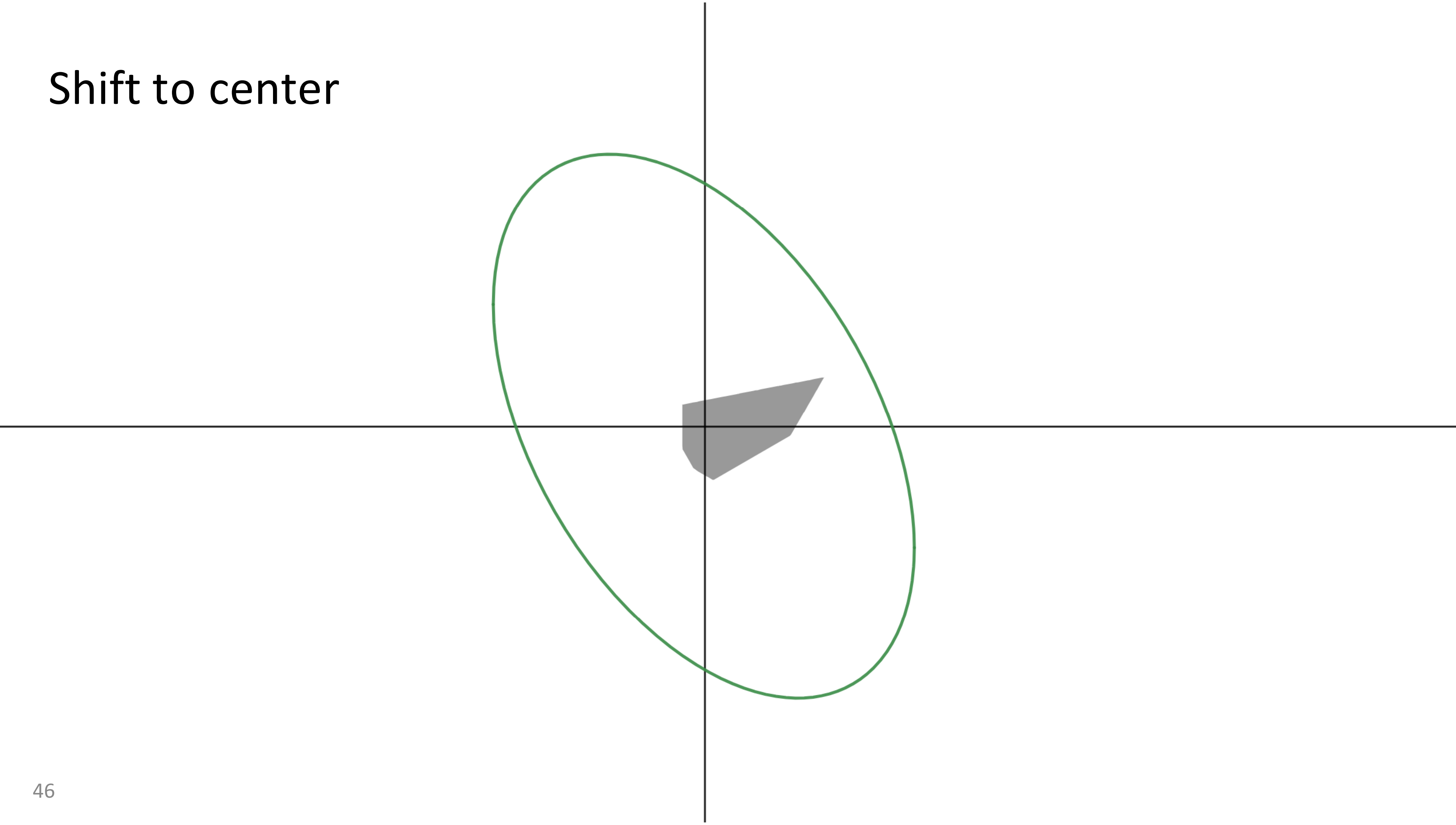
Find ellipsoid containing half-sphere



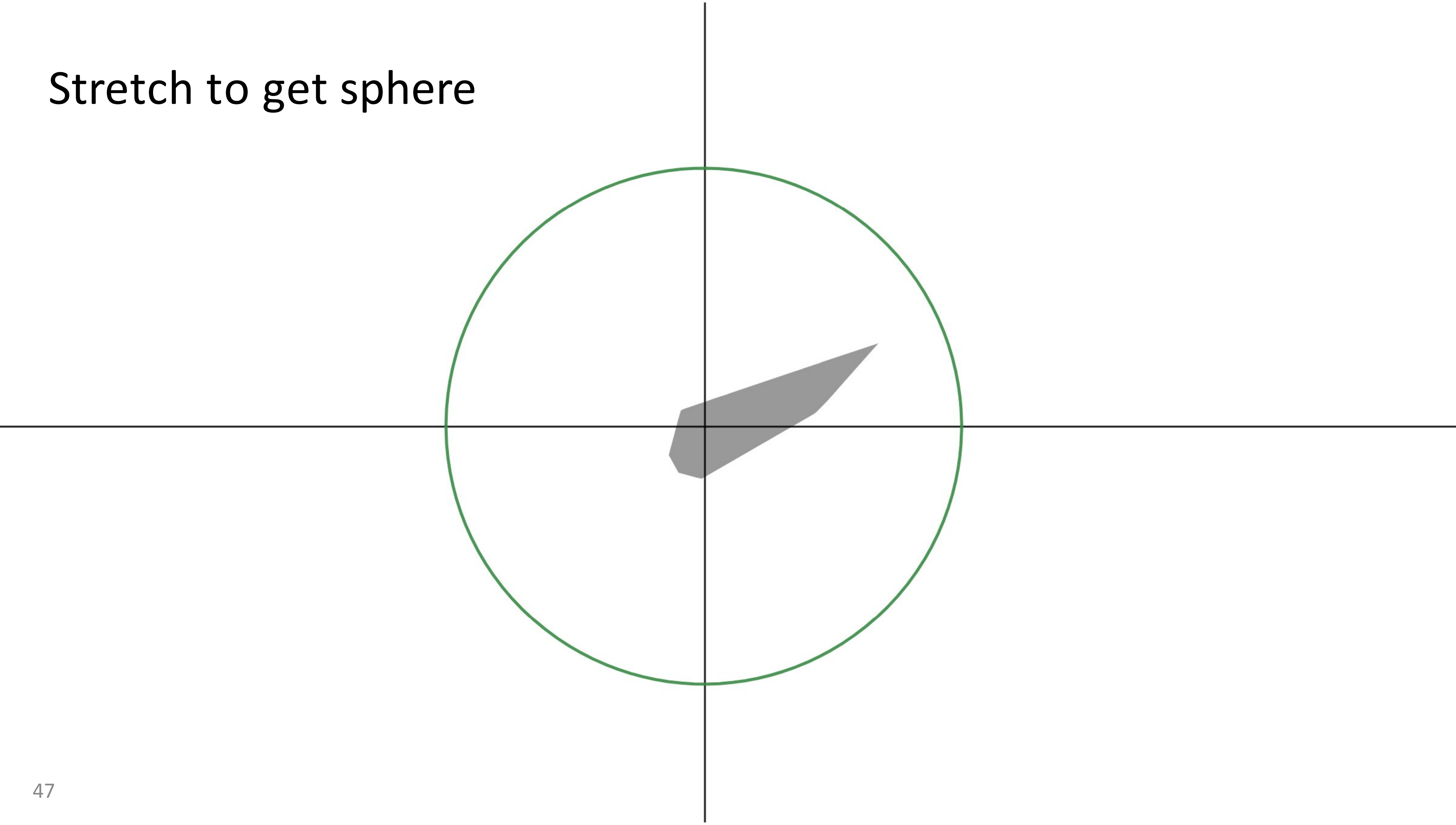
Find ellipsoid containing half-sphere



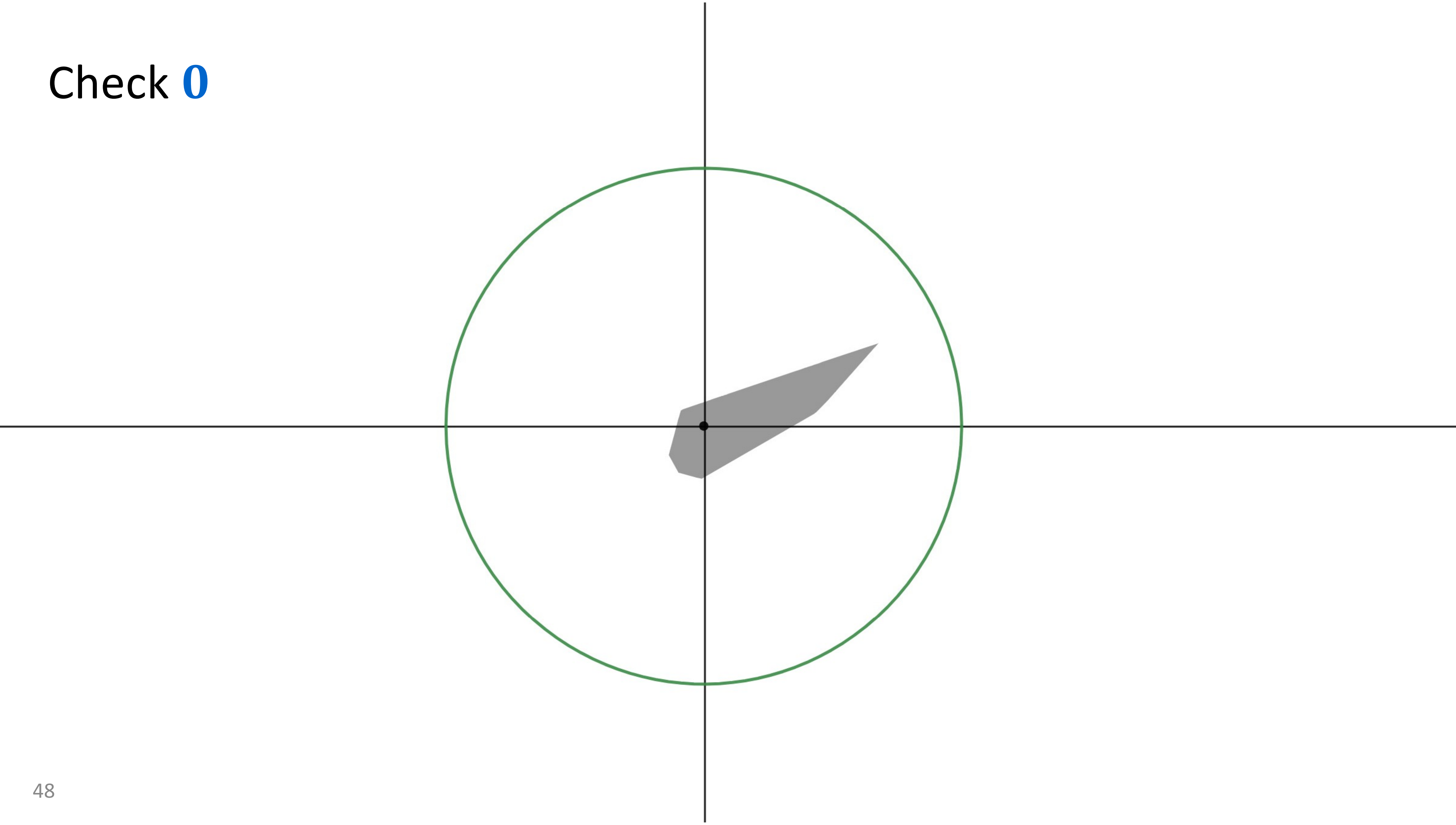
Shift to center



Stretch to get sphere



Check $\mathbf{0}$



Ellipsoid Method

Is there an x s.t.

$$c^{\top}x \geq d$$
$$Ax \leq b$$
$$x \geq 0\ ?$$

Algorithm to find a point in a non-empty bounded polytope P:

1. Compute ellipsoid E as a radius R sphere containing polytope P
2. If $\mathbf{0} \in P$, output original equivalent point
   a. Otherwise identify violated constraint for $\mathbf{0}$, shift to origin to identify half-sphere inside E
   b. Let E' be ellipsoid containing half-sphere
   c. Shift and rescale E' to sphere E, applying the same to P and begin step 2.

Key Lemma:
$$vol(E')/vol(E) \leq e^{-1/(2n+1)}$$

Corollary:
$$\frac{vol(P)}{vol(E')} \geq e^{1/(2n+1)} \frac{vol(P)}{vol(E)}$$

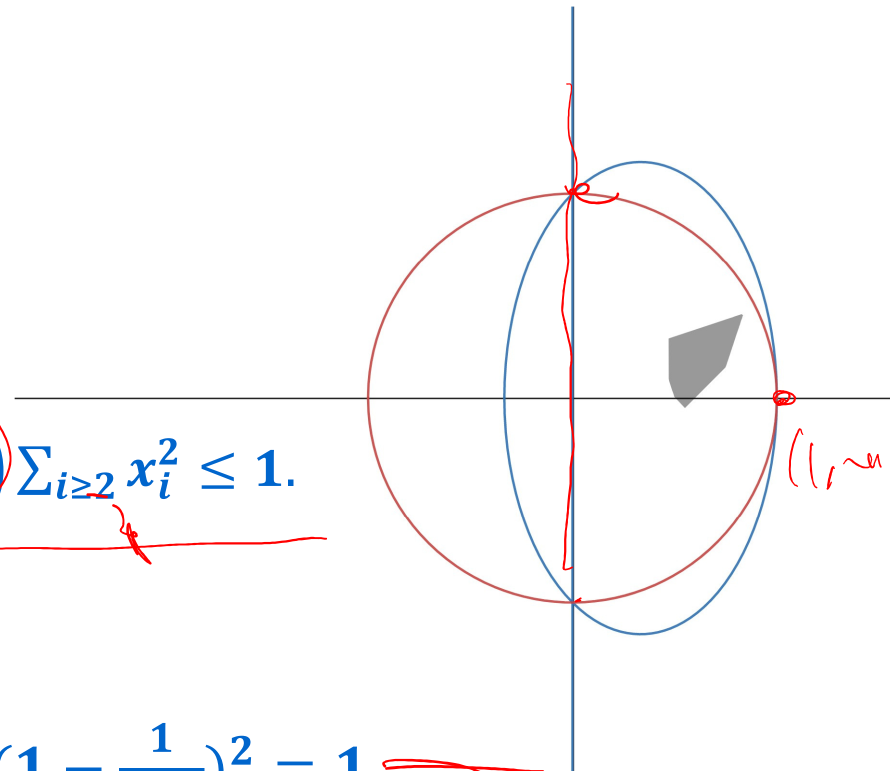
Thm: After t rounds
$$\frac{vol(P)}{vol(E_t)} \geq e^{t/(2n+1)} \frac{vol(P)}{vol(E)}$$

Cor: Ellipsoid algorithm halts after $poly$(input length) steps

Key Lemma part 1

Define $\boldsymbol{E}$ by $\sum_i x_i^2 \le 1$

Let $\boldsymbol{E'}$ be given by $\left(\frac{n+1}{n}\right)^2 \left(x_1 - \frac{1}{n+1}\right)^2 + \left(1 - \frac{1}{n^2}\right) \sum_{i \ge 2} x_i^2 \le 1.$



Claim: $\boldsymbol{E'}$ contains the positive half-sphere.

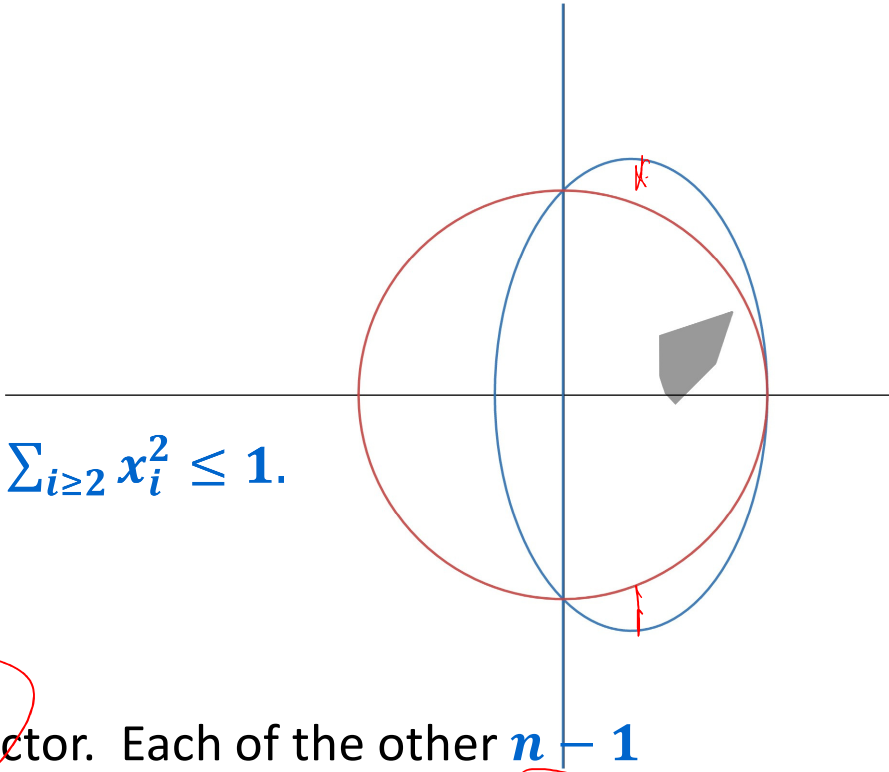
Proof: Note that $(1, 0, \dots, 0) \in E'$ since $\left(\frac{n+1}{n}\right)^2 \left(1 - \frac{1}{n+1}\right)^2 = 1.$

Consider intersection of $\boldsymbol{E}$ with $x_1 = 0$: $(0, x_2, \dots, x_n)$ with $\sum_{i \ge 2} x_i^2 \le 1$

LHS for $\boldsymbol{E'}$ for these points: $\le \left(\frac{n+1}{n}\right)^2 \left(-\frac{1}{n+1}\right)^2 + \left(1 - \frac{1}{n^2}\right) = \frac{1}{n^2} + 1 - \frac{1}{n^2} = 1$

so these points are all in $\boldsymbol{E'}$.

Key Lemma part 2



Define $\boldsymbol{E}$ by $\sum_i x_i^2 \leq 1$

Let $\boldsymbol{E'}$ be given by $\left(\frac{n+1}{n}\right)^2 \left(x_1 - \frac{1}{n+1}\right)^2 + \left(1 - \frac{1}{n^2}\right) \sum_{i \geq 2} x_i^2 \leq 1.$

Claim: $vol(E')/vol(E) \leq e^{-1/(2n+1)}$

Proof: First coordinate scales down by an $\left(\frac{n}{n+1}\right)$ factor. Each of the other $n-1$ coordinates scales up by a $\left(\frac{n^2}{n^2-1}\right)^{1/2}$ factor. Volume scales by $\left(\frac{n}{n+1}\right)\left(\frac{n^2}{n^2-1}\right)^{(n-1)/2}$

This is $\left(1 - \frac{1}{n+1}\right)\left(1 + \frac{1}{n^2-1}\right)^{(n-1)/2} \leq e^{-\frac{1}{n+1}} e^{\frac{1}{2(n+1)}} = e^{\frac{-1}{2(n+1)}}$ using $1 + z \leq e^z$.