18 Simplex Alg. Part II

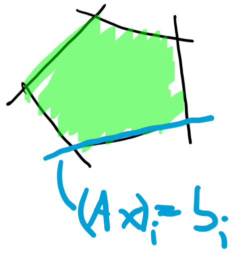
Thursday, March 13, 2025 13:52

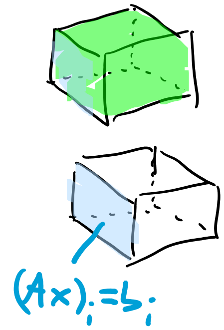
$A \in \mathbb{R}^{n\times d}$ $b \in \mathbb{R}^n$ $n \geq d$

$\{x \in \mathbb{R}^d \mid Ax \leq b\}$

- feasible x form a polytope
- each facet of polytope is a constraint $(Ax)_i \leq b_i$

with $(Ax)_i = b_i$ for points on the facet.



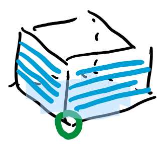


---

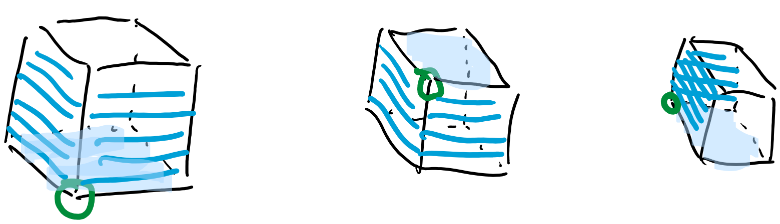
vertex $x \in \mathbb{R}^d$ of polytope is described by set $B \subseteq \{1..n\}$

- $Ax \leq b$
- $A_B x = b_B$ where A_B, b_B subset of rows of A and b as specified by index set B
- A_B full rank, A_B must be invertible

$x = (A_B)^{-1} b_B$ (by $A_B x = b_B$)



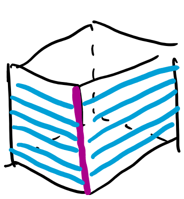
$(Ax)_i = b_i$

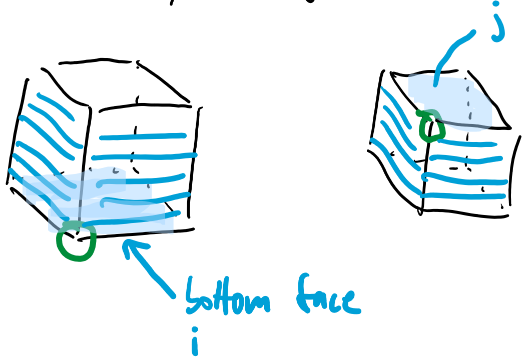


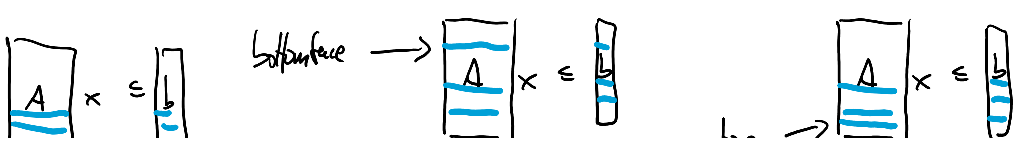
2 vertices $x, x' \in \mathbb{R}^d$ are connected by an edge

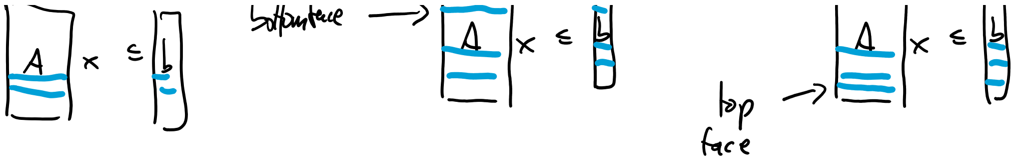
if $B' = (B \setminus \{i\}) \cup \{j\}$

$B, B' \subseteq \{1...n\}$



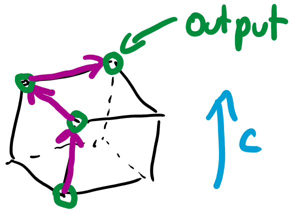


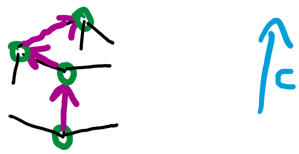


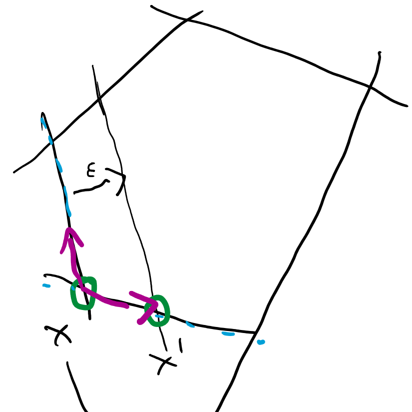


Simplex Algorithm: solve $\max c^T x$, $Ax \le b$

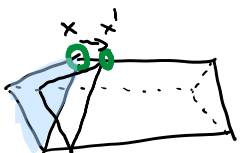
- start at some vertex of polytope $\{x \mid Ax \le b\}$
- Go along edges, from vertex to vertex, always increasing $c^T x$



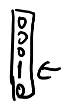




$$A_B x = b_B$$



$(Ax)_i \le b_i - \varepsilon$



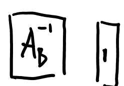
edge direction

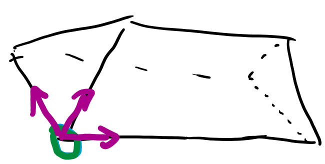
$$x' - x = \underbrace{A_B^{-1}(b_B - \varepsilon \cdot e_i)}_{x'} - \underbrace{A_B^{-1} b_B}_{x}$$

$$= -\varepsilon \cdot A_B^{-1} e_i = -\varepsilon \cdot \text{ ith column of } A_B^{-1}$$

Given a vertex

the incident edge directions are the columns of $-A_B^{-1}$

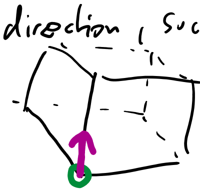




$-A_B^{-1}$

We want to move in direction, such that $c^T x$ increases

$$x^{new} \leftarrow x - \underset{\substack{\uparrow \\ \ge 0,\ \in \mathbb{R}}}{\lambda} \cdot \underline{A_B^{-1} e_i}$$



We want that $c^T x^{new} > c^T x$

$$c^T(x - \lambda \cdot A_B^{-1} e_i) = c^T x - \lambda \cdot c^T A_B^{-1} e_i$$

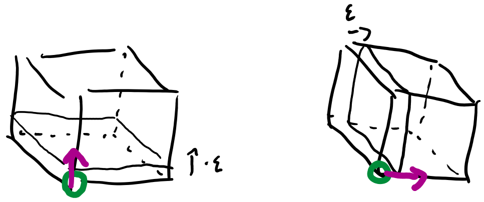
$$\Leftrightarrow -\lambda\, c^T A_B^{-1} e_i > 0 \quad \Leftrightarrow \quad c^T A_B^{-1} e_i < 0$$

$\iff -\lambda c^T A_B e_i > 0 \iff \dots$

i-th entry of $c^T A_B^{-1}$

$(c^T A_B^{-1}) e_i = \underbrace{\boxed{\quad}}_{c^T A_B^{-1}} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

To pick the correct direction, we compute $c^T A_B^{-1}$ and then pick index i where $(c^T A_B)_i < 0$

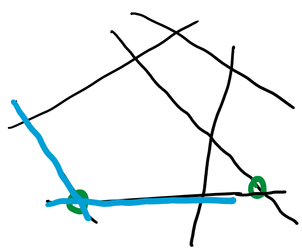


This i corresponds to the index removed from B when we move along the edge.

Q: How far can we move in that direction?
How big is λ?

Find largest λ such that

$\left(A\left(x - \lambda A_B^{-1} e_i\right)\right)_j \le b_j \quad \forall j$



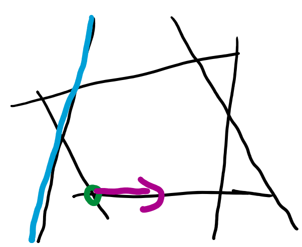
$\iff -\left(A\left(A_B^{-1} e_i\right) \cdot \lambda\right)_j \le b_j - (Ax)_j$

$(Ax)_j \le b_j$

$$\lambda = \min_{\substack{j \notin B \\ (A(A_B^{-1} e_i))_j < 0}} \frac{(b - Ax)_j}{-(A(A_B^{-1} e_i))_j}$$

$\leftarrow$ distance to jth facet

$\leftarrow$ velocity, how fast we move towards jth facet

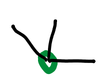


Simplex Algorithm

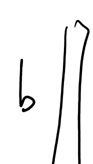
Assume x is a vertex, $B \subseteq \{1 \dots n\}$

Repeat:

Find i where $(c^T A_B^{-1})_i < 0$ // Find edge that increases $c^T x$



if such i exists . . . .



increases $c'x$

if such i exists

$$\lambda = \min_{\substack{j \notin B \\ (A(A_B^{-1}e_i))_j < 0}} \frac{(b - Ax)_j}{-(A A_B^{-1} e_i)_j}$$

// compute how far we can move

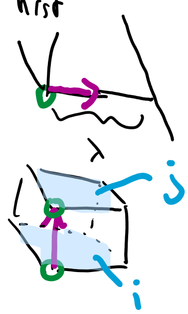
j specifies constraint/facet we hit first

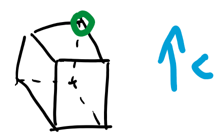
$B = (B \setminus \{i\}) \cup \{j\}$

$x \leftarrow x - \lambda A_B^{-1} e_i$

else

return x





Correctness Input was $\max c^T x$

$$Ax \le b$$

dual is $\min b^T y$

$$A^T y = c$$
$$y \ge 0$$

Algo returns some x, but is it optimal?

If there exists y with $A^T y = c$, $y \ge 0$ and $b^T y = c^T x$ then x (and y) are optimal solutions.

$y \in \mathbb{R}^n$ $\quad y_B = (A_B^{-1})^T c \ge 0$

$y_{rest} = 0$

$$(M^T)^{-1} = (M^{-1})^T$$

$$A^T y = \begin{bmatrix} \boxed{A_B^T} & A^T_{rest} \end{bmatrix} \begin{bmatrix} \boxed{y_B} \\ y_{rest} \end{bmatrix} = A_B^T y_B = A_B^T (A_B^T)^{-1} c = c$$

$$A'y = \begin{bmatrix} A_B^T & A_{rest}^T \end{bmatrix} \begin{bmatrix} \\ y_{rest} \end{bmatrix} = A_B^T y_B = A_B' (A_B') c = c$$

$$b^T y = b_B^T y_B + b_{rest}^T y_{rest} = b_B^T y_B = b^T (A_B^{-1})^T c = (A_B^{-1} b_B)^T c$$

$$= x^T c = c^T x$$

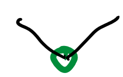
$$\begin{bmatrix} b_B & b_{rest} \end{bmatrix} \begin{bmatrix} y_B \\ y_{rest} \end{bmatrix}$$

$\Rightarrow$ x and y are both optimal solutions

Time complexity: A_B^{-1} $d \times d$ $O(d^3)$ // HW $O(d^2)$ per iteration

$A \cdot$ vector $A, n \times d$ $O(nd)$

$\Rightarrow O((nd + d^3) \cdot \#\text{ iterations})$



In practice $O(n)$ iterations typically suffice

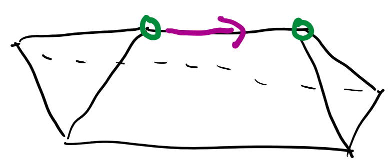
There are worst-case examples where we have $> 2^d$ iterations.

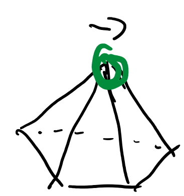
$$\boxed{A} x \le b$$

poly time is proven/known if tiny noise is added to b

Interior Point Methods n^3 Cohen Lee Song 19

Barrier Method





$\lambda = 0$

Bland's rule