Lecture 6: Quadratic programming

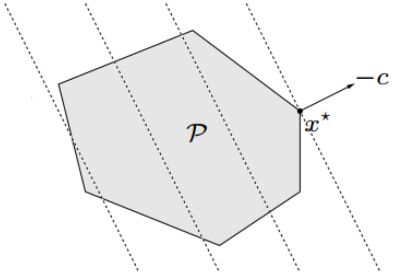
- Quadratic programming; convex and non-convex QPs
- Equality constrained QPs
  - Building block of general QP solvers (next time)

Reference: N&W Ch.15.3-15.5, 16.1-2,4-5

Types of constrained optimization problems

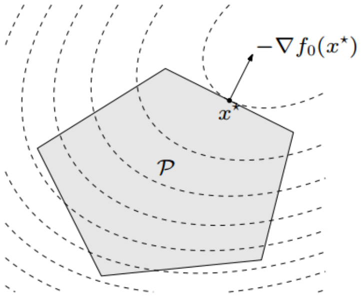
- Linear programming
  - Convex problem
  - Feasible set polyhedron

$$
\begin{aligned}
\text{minimize} \quad & c^\mathsf{T} x \\
\text{subject to} \quad & Ax \le b \\
& Cx = d
\end{aligned}
$$



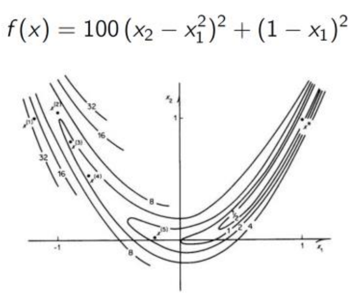
- Quadratic programming
  - Convex problem if $P \geq 0$
  - Feasible set polyhedron

$$
\begin{aligned}
\text{minimize} \quad & \frac{1}{2} x^\mathsf{T} P x + q^\mathsf{T} x \\
\text{subject to} \quad & Ax \le b \\
& Cx = d
\end{aligned}
$$



- Nonlinear programming
  - In general non-convex!

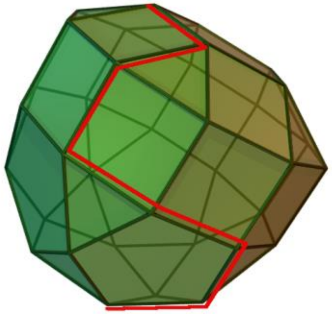
$$
\begin{aligned}
\text{minimize} \quad & f(x) \\
\text{subject to} \quad & g(x) = 0 \\
& h(x) \geq 0
\end{aligned}
$$



$$
\min_{x \in \mathbb{R}^n} f(x) \qquad \text{subject to} \quad
\begin{cases}
c_i(x) = 0, & i \in \mathcal{E}, \\
c_i(x) \geq 0, & i \in \mathcal{I}.
\end{cases}
$$

Last time: The simplex method for LP

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



- The Simplex algorithm
  - The feasible set of LPs are (convex) polytopes
  - LP solution is a vertex ("corner") of the feasible set
  - Simplex works by going from vertex to neighbouring vertex (which are all "basic feasible points", BFP) in such a manner that the objective decreases in each iteration.
  - In each iteration, we solve a linear system to find which component in the "basis" (set of "not active constraints") we should change
  - Almost guaranteed convergence (if LP not unbounded or infeasible)
- Complexity:
  - Typically, at most *2m* to *3m* iterations
  - Worst case: All vertices must be visited (exponential complexity in *n*)
- Active set methods (such as simplex method):
  - Maintains explicitly an estimate of the set of inequality constraints that are active at the solution (the set $\mathcal{N}$ for the simplex method)
  - Makes small changes to the set in each iteration (a single index in simplex)

- Today, and next lecture: Active set method for QP

Why are we interested in QPs?

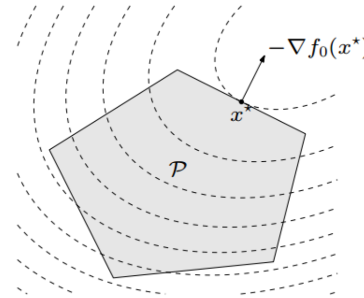
Three (main) reasons:

- It is the most “easy” nonlinear programming problem (so special that it is given a separate name; quadratic programming)
  - “easy”: efficient algorithms exists, especially for convex QPs

- The QP is the basic building block of SQP (“sequential quadratic programming”), a common method for solving general nonlinear programs
  - Topic in end of course (N&W Ch. 18)

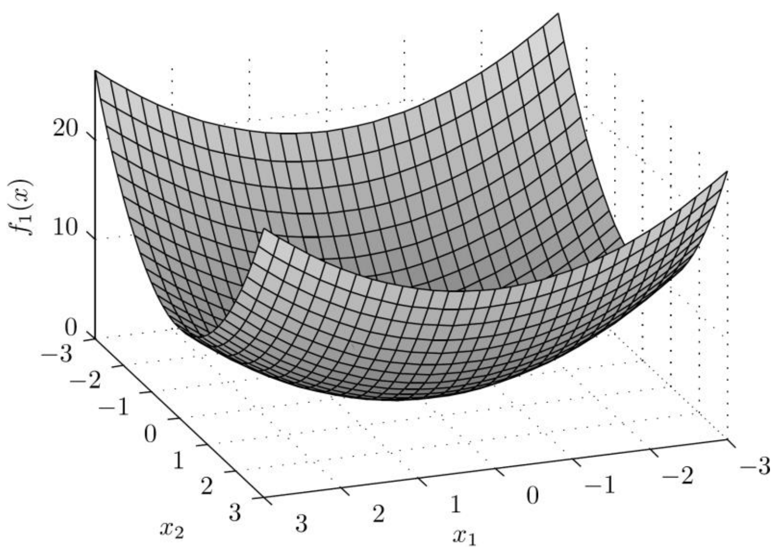
- QPs are very much used in control, especially as solvers in what is called MPC (“Model Predictive Control”)
  - Topic in a few weeks
  - Also used in finance (“Portifolio optimization”), some types of Machine Learning/regression problems, control allocation, economics, …

Convex QP

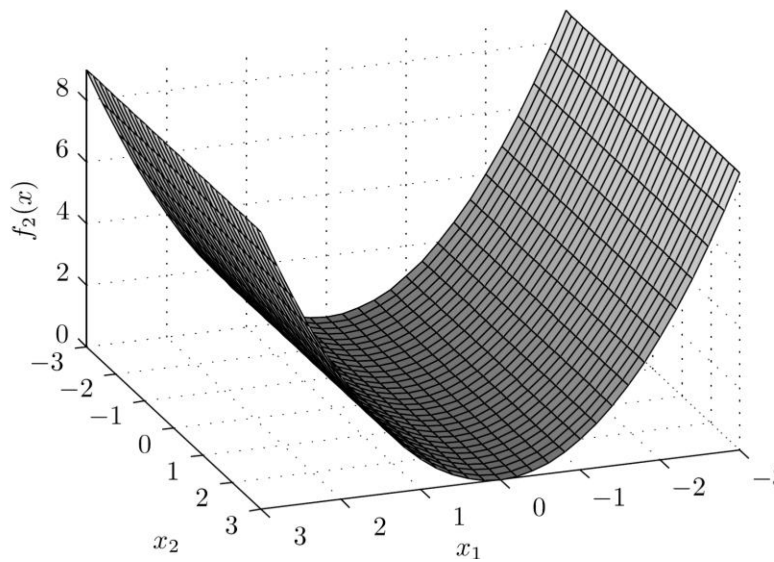


- Feasible set is (convex) polytope

- Objective is quadratic function, which can be non-convex (concave or indefinite), convex or strictly convex



$G > 0$, strictly convex



$G \geq 0$, convex

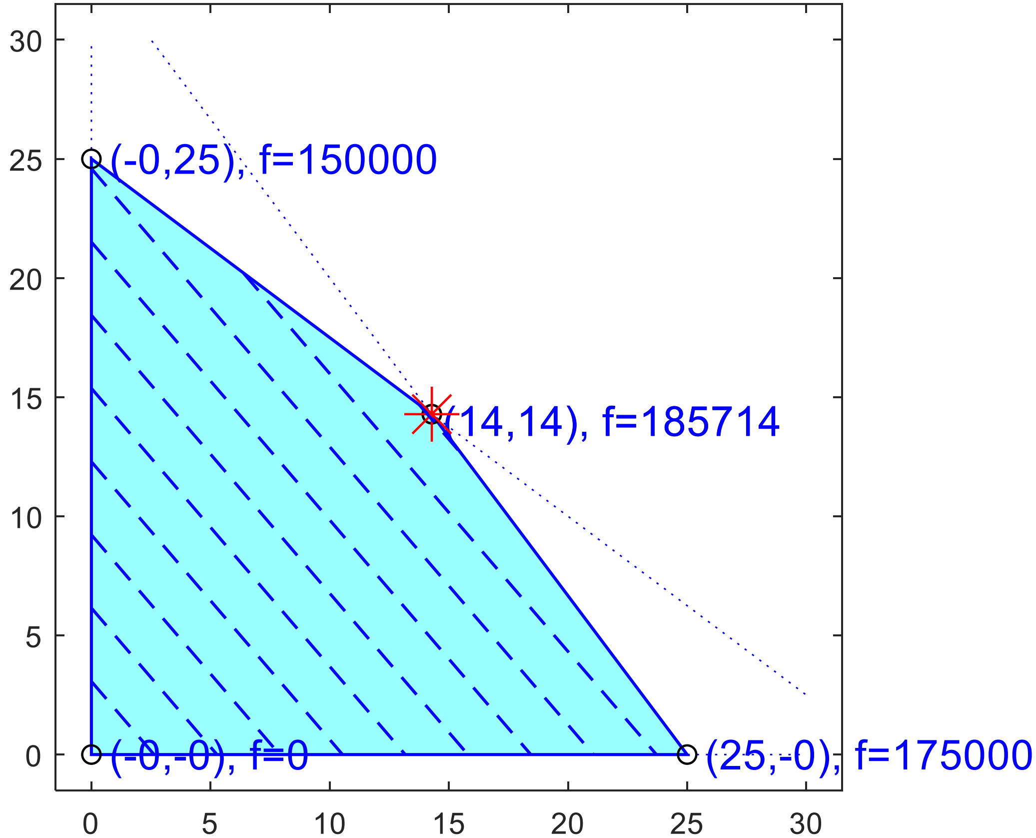
QP example: Farming example with changing prices

- A farmer wants to grow apples (A) and bananas (B)
- He has a field of size 100 000 m^2
- Growing 1 tonne of A requires an area of 4 000 m^2, growing 1 tonne of B requires an area of 3 000 m^2
- A requires 60 kg fertilizer per tonne grown, B requires 80 kg fertilizer per tonne grown
- The profit for A is (7000 – 200 x_1) per tonne (including fertilizer cost), the profit for B is (6000 – 140 x_2) per tonne (including fertilizer cost)
- The farmer can legally use up to 2000 kg of fertilizer
- He wants to maximize his profits

LP farming example: Geometric interpretation and solution

$$\max_{x_1,\, x_2} \quad 7000x_1 + 6000x_2$$

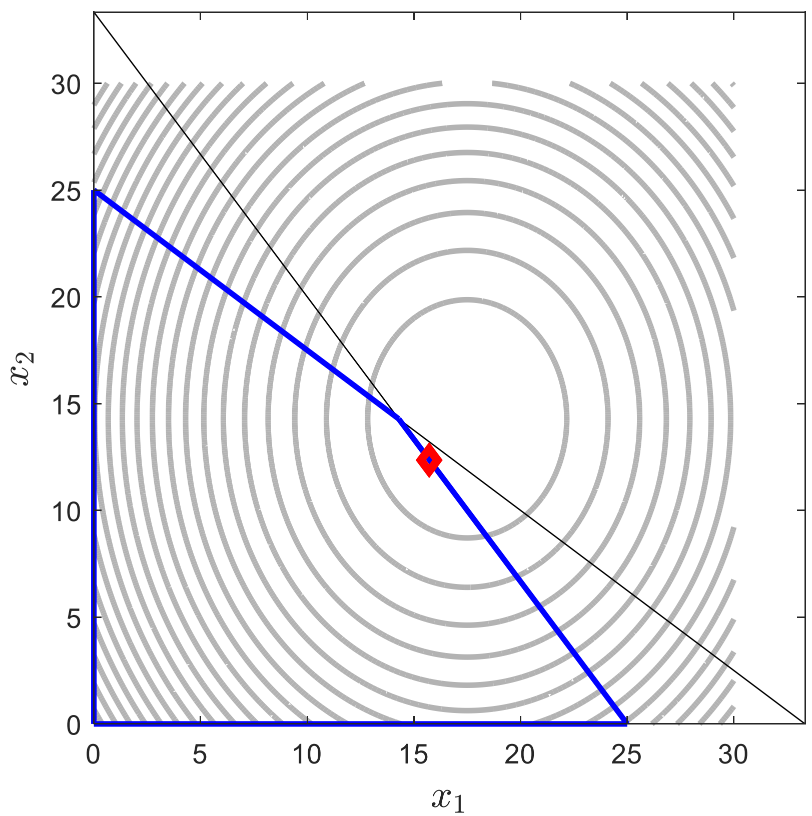
$$\text{subject to:} \quad 4000x_1 + 3000x_2 \leq 100000$$
$$60x_1 + 80x_2 \leq 2000$$
$$x_1 \geq 0$$
$$x_2 \geq 0$$



QP farming example: Geometric interpretation and solution

$$\max_{x_1,\, x_2} \quad (7000 - 200x_1)x_1 + (6000 - 140x_2)x_2$$

$$\begin{aligned}
\text{subject to:} \quad 4000x_1 + 3000x_2 &\leq 100000 \\
60x_1 + 80x_2 &\leq 2000 \\
x_1 &\geq 0 \\
x_2 &\geq 0
\end{aligned}$$



KKT conditions (Theorem 12.1)

Lagrangian: $$\mathcal{L}(x,\lambda) = f(x) - \sum_{i\in\mathcal{E}\cup\mathcal{I}} \lambda_i c_i(x)$$

KKT-conditions (First-order necessary conditions): If x^* is a local solution and LICQ holds, then there exist λ^* such that

$$
\begin{aligned}
\nabla_x \mathcal{L}(x^*, \lambda^*) &= 0, && && \text{(stationarity)} \\
c_i(x^*) &= 0, && \forall i \in \mathcal{E}, && \\
c_i(x^*) &\geq 0, && \forall i \in \mathcal{I}, && \text{(primal feasibility)} \\
\lambda_i^* &\geq 0, && \forall i \in \mathcal{I}, && \text{(dual feasibility)} \\
\lambda_i^* c_i(x^*) &= 0, && \forall i \in \mathcal{E} \cup \mathcal{I}. && \text{(complementarity condition/ complementary slackness)}
\end{aligned}
$$

Either $\lambda_i^* = 0$ *or* $c_i(x^*) = 0$

(*strict* complimentarity: Only one of them is zero)

Example 16.2

$$\min_x \quad \frac{1}{2}x^\top G x + c^\top x$$
$$\text{subject to} \quad Ax = b$$

$$\min_{x_1,x_2,x_3} 3x_1^2 + 2x_1x_2 + x_1x_3 + 2.5x_2^2 + 2x_2x_3 + 2x_3^2 - 8x_1 - 3x_2 - 3x_3$$
$$\text{subject to} \quad x_1 + x_3 = 3, \quad x_2 + x_3 = 0$$

$$\text{Matrices:} \quad G = \begin{pmatrix} 6 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 4 \end{pmatrix}, \quad c = \begin{pmatrix} -8 \\ -3 \\ -3 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$

Note symmetry of G.
Always possible!

```
>> G = [6 2 1; 2 5 2; 1 2 4]; c = [-8; -3; -3]; A = [1 0 1;0 1 1]; b = [3;0];
>> K = [G, -A'; A, zeros(2,2)];
>> K\[-c;b]            % X = A\B is the solution to the equation A*X = B

ans =

   2.0000
  -1.0000
   1.0000
   3.0000
  -2.0000
```

x^*

λ^*

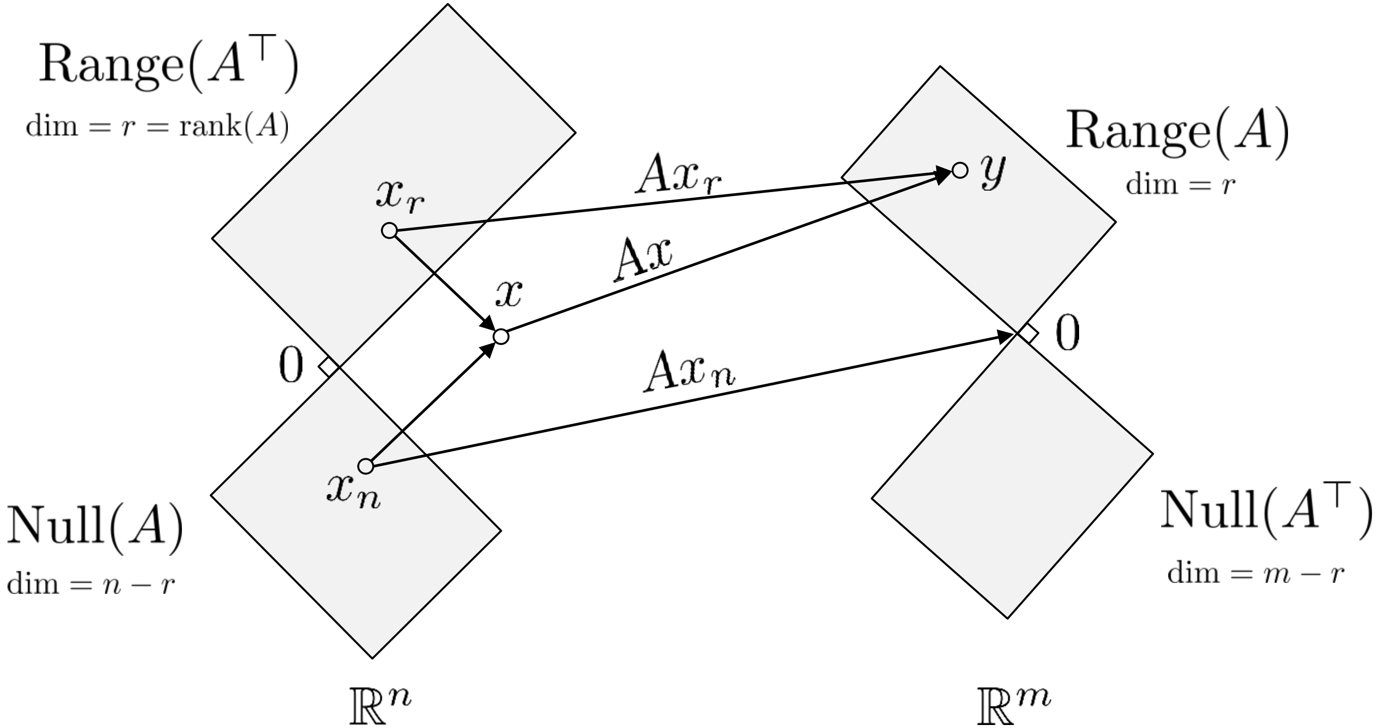
Fundamental theorem of linear algebra

A matrix $A \in \mathbb{R}^{m\times n}$ maps a vector $x \in \mathbb{R}^n$ into a vector $y \in \mathbb{R}^m$, $y = Ax$.

Nullspace of A: $\mathrm{Null}(A) = \{w \mid Aw = 0\}$

Rangespace (columnspace) of A: $\mathrm{Range}(A) = \{w \mid w = Av, \text{for some } v\}$

Fundamental theorem of linear algebra: $\mathrm{Null}(A) \oplus \mathrm{Range}(A^\top) = \mathbb{R}^n$



Example 16.2

$$\min_{x} \quad \frac{1}{2} x^\top G x + c^\top x$$
$$\text{subject to} \quad Ax = b$$

$$\min_{x_1, x_2, x_3} 3x_1^2 + 2x_1x_2 + x_1x_3 + 2.5x_2^2 + 2x_2x_3 + 2x_3^2 - 8x_1 - 3x_2 - 3x_3$$
$$\text{subject to} \quad x_1 + x_3 = 3, \quad x_2 + x_3 = 0$$

$$\text{Matrices:} \quad G = \begin{pmatrix} 6 & 2 & 1 \\ 2 & 5 & 2 \\ 1 & 2 & 4 \end{pmatrix}, \quad c = \begin{pmatrix} -8 \\ -3 \\ -3 \end{pmatrix}, \quad A = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$

Note symmetry of G.
Always possible!

```
>> G = [6 2 1; 2 5 2; 1 2 4]; c = [-8; -3; -3]; A = [1 0 1;0 1 1]; b = [3;0];
>> K = [G, -A'; A, zeros(2,2)];
>> K\[-c;b]          % X = A\B is the solution to the equation A*X = B

ans =

    2.0000
   -1.0000
    1.0000
    3.0000
   -2.0000
```

x^*

λ^*

```
>> [Q,R,P] = qr(A')

Q =

   -0.7071    0.4082   -0.5774
         0   -0.8165   -0.5774
   -0.7071   -0.4082    0.5774

R =

   -1.4142   -0.7071
         0   -1.2247
         0         0

P =

     1     0
     0     1
```

Y

Z

Direct solutions of KKT system (16.2)

- Full space:

$$\begin{pmatrix} G & A^\top \\ A & 0 \end{pmatrix} \begin{pmatrix} -p \\ \lambda^* \end{pmatrix} = \begin{pmatrix} c + Gx \\ Ax - b \end{pmatrix}$$

  - Use LU
  - Or better: Since KKT-matrix is symmetric, use LDL-method
    - Cholesky cannot be used, since KKT-matrix is indefinite for $m \geq 1$
- Reduced space, efficient if $n\text{-}m \ll n$:

$$(AY)p_Y = b - Ax$$
$$(Z^\top GZ)p_Z = -Z^\top GYp_y + Z^\top(c + Gx)$$
$$p = Yp_y + Zp_z$$

  - Solve two much smaller systems using LU and Cholesky
    - both with complexity that scales with n^3
  - Main complexity is calculating basis for nullspace. Usual method is using QR.

- Alternative to direct methods: Iterative methods (16.3)
  - For very large systems, can be parallelized