Advanced Optimization
Lecture 4: Gradient Descent, Constrained and Linear Optimization

October 27, 2021

CentraleSupélec / ESSEC Business School

dimo.brockhoff@inria.fr

Dimo Brockhoff

Inria Saclay – Ile-de-France

Course Overview

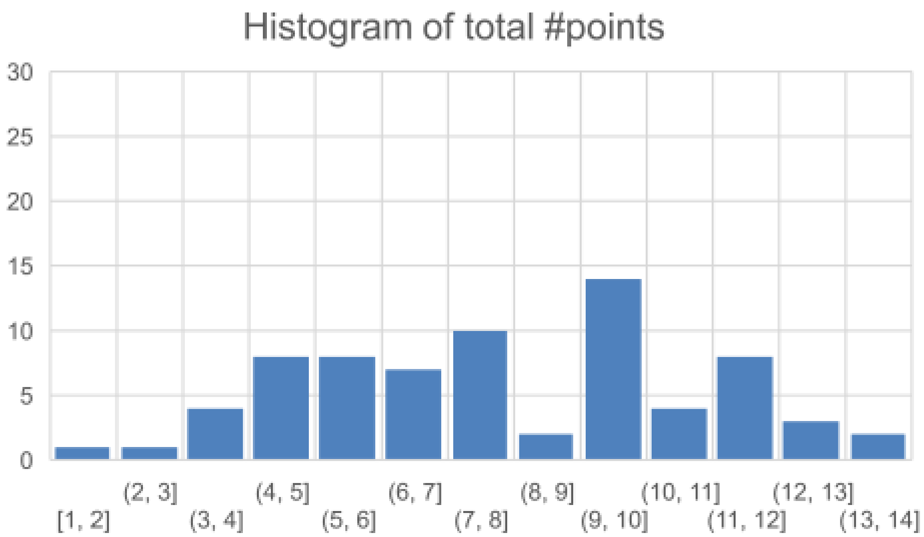
		Topic
Wed, 13.10.2021	PM	Introduction, examples of problems, problem types
Wed, 20.10.2021	PM	Continuous (unconstrained) optimization: convexity, gradients, Hessian, … [technical test Evalmee]
Wed, 27.10.2021	PM	Continous optimization II: [1st mini-exam] Constrained optimization: Lagrangian, optimality conditions
Wed, 03.11.2021	PM	gradient descent, Newton direction, quasi-Newton (BFGS) Linear programming: duality, maxflow/mincut, simplex algo
Wed, 10.11.2021	PM	Gradient-based and derivative-free stochastic algorithms: SGD and CMA-ES
Wed, 17.11.2021	PM	Other blackbox optimizers: Nelder-Mead, Bayesian optimization *[2nd mini-exam]*
Wed, 24.11.2021	PM	Benchmarking solvers: runtime distributions, performance profiles
Tue, 30.11.2021	23:59	Deadline open source project (PDF sent by email)
Wed, 01.12.2021	PM	Discrete optimization: branch and bound, branch and cut, k-means clustering
Wed, 15.12.2021	PM	Exam

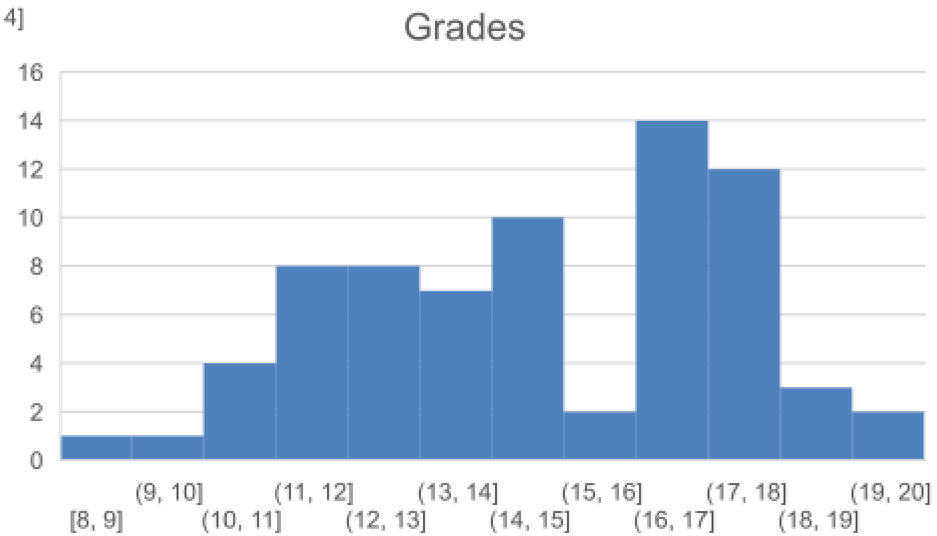
Course Overview

		Topic
Wed, 13.10.2021	PM	Introduction, examples of problems, problem types
Wed, 20.10.2021	PM	Continuous (unconstrained) optimization: convexity, gradients, Hessian, … [technical test Evalmee]
Wed, 27.10.2021	PM	Continous optimization II: [1st mini-exam] Constrained optimization: Lagrangian, optimality conditions
Wed, 03.11.2021	PM	gradient descent, Newton direction, quasi-Newton (BFGS) Linear programming: duality, maxflow/mincut, simplex algo
Wed, 10.11.2021	PM	Gradient-based and derivative-free stochastic algorithms: SGD and CMA-ES
Wed, 17.11.2021	PM	Other blackbox optimizers: Nelder-Mead, Bayesian optimization *Okay with everybody!*
Wed, 24.11.2021	PM	Benchmarking solvers: runtime distributions, performance profiles [2nd mini-exam]
Tue, 30.11.2021	23:59	Deadline open source project (PDF sent by email)
Wed, 01.12.2021	PM	Discrete optimization: branch and bound, branch and cut, k-means clustering
Wed, 15.12.2021	PM	Exam

speaking of the mini-exam…

Grading Almost Finalized…





Details on Continuous Optimization Lectures

Introduction to Continuous Optimization

- examples and typical difficulties in optimization

Mathematical Tools to Characterize Optima

- reminders about differentiability, gradient, Hessian matrix
- unconstraint optimization
  - first and second order conditions
  - convexity
- constraint optimization
  - Lagrangian, optimality conditions

Gradient-based Algorithms

- gradient descent
- quasi-Newton method (BFGS) and invariances

Linear programming, duality

Learning in Optimization / Optimization in Machine Learning

- Stochastic gradient descent (SGD) + Adam
- CMA-ES (adaptive algorithms / Information Geometry)
- Other derivative-free algorithms: Nelder-Mead, Bayesian opt.

back to
optimality conditions
in constrained optimization

Reminder: Euler-Lagrange Equation

Theorem:

Be U an open set of $(E, ||\,.\,||)$, and $f: U \to \mathbb{R}$, $g: U \to \mathbb{R}$ in $\mathcal{C}^1$.

Let $a \in E$ satisfy

$$\begin{cases} f(a) = \inf \{f(x) \mid x \in \mathbb{R}^n, g(x) = 0\} \\ g(a) = 0 \end{cases}$$

i.e. a is optimum of the problem

If $\nabla g(a) \neq 0$, then there exists a constant $\lambda \in \mathbb{R}$ called *Lagrange multiplier*, such that

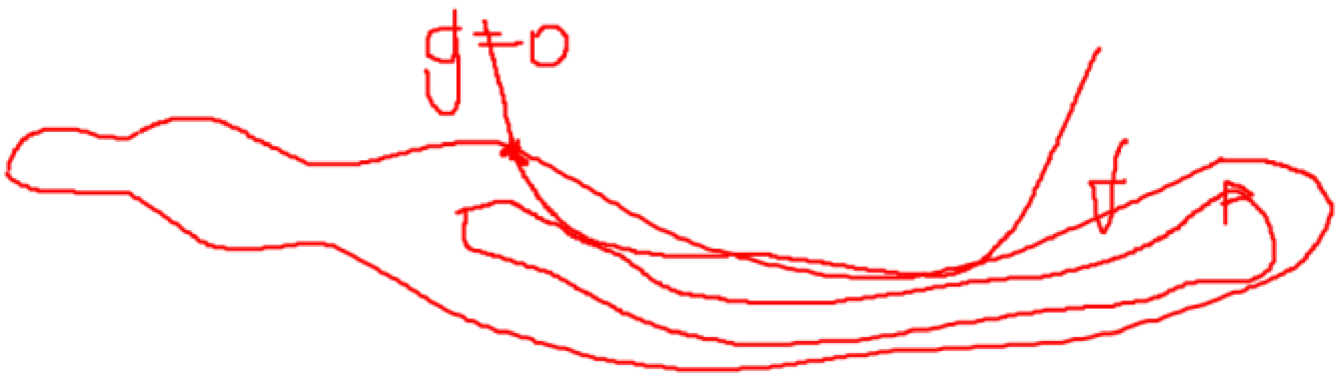
$$\underbrace{\nabla f(a) + \lambda \nabla g(a) = 0}_{} \quad \text{Euler} - \text{Lagrange equation}$$

i.e. gradients of f and g in a are colinear

Interpretation of Euler-Lagrange Equation

Intuitive way to retrieve the Euler-Lagrange equation:

- In a local minimum a of a constrained problem, the hypersurfaces (or level sets) $f = f(a)$ and $g = 0$ are necessarily tangent (otherwise we could decrease f by moving along $g = 0$).



- Since the gradients $\nabla f(a)$ and $\nabla g(a)$ are orthogonal to the level sets $f = f(a)$ and $g = 0$, it follows that $\nabla f(a)$ and $\nabla g(a)$ are colinear.

Generalization to More than One Constraint

Theorem

- Assume $f: U \to \mathbb{R}$ and $g_k: U \to \mathbb{R}$ $(1 \leq k \leq p)$ are $\mathcal{C}^1$.
- Let a be such that

$$\begin{cases} f(a) = \inf \{f(x) \mid x \in \mathbb{R}^n, \quad g_k(x) = 0, \quad 1 \leq k \leq p\} \\ \quad g_k(a) = 0 \ \text{ for all } 1 \leq k \leq p \end{cases}$$

- If $\left(\nabla g_k(a)\right)_{1 \leq k \leq p}$ are linearly independent, then there exist p real constants $(\lambda_k)_{1 \leq k \leq p}$ such that

$$\nabla f(a) + \sum_{k=1}^{p} \lambda_k \nabla g_k(a) = 0$$

Lagrange multiplier

again: a does not need to be global but local minimum

The Lagrangian

- Define the Lagrangian on $\mathbb{R}^n \times \mathbb{R}^p$ as

$$\mathcal{L}(x, \{\lambda_k\}) = f(x) + \sum_{k=1}^{p} \lambda_k g_k(x)$$

- To find optimal solutions, we can solve the optimality system

$$\begin{cases} \text{Find } (x, \{\lambda_k\}) \in \mathbb{R}^n \times \mathbb{R}^p \text{ such that } \nabla f(x) + \sum_{k=1}^{p} \lambda_k \nabla g_k(x) = 0 \\ g_k(x) = 0 \;\text{ for all } 1 \le k \le p \end{cases}$$

$$\Leftrightarrow \begin{cases} \text{Find } (x, \{\lambda_k\}) \in \mathbb{R}^n \times \mathbb{R}^p \text{ such that } \nabla_x \mathcal{L}(x, \{\lambda_k\}) = 0 \\ \nabla_{\lambda_k} \mathcal{L}(x, \{\lambda_k\})(x) = 0 \;\text{ for all } 1 \le k \le p \end{cases}$$

Inequality Constraint: Definitions

Let $\mathcal{U} = \{x \in \mathbb{R}^n \mid g_k(x) = 0 \text{ (for } k \in E),\ g_k(x) \leq 0 \text{ (for } k \in I)\}$.

Definition:

The points in $\mathbb{R}^n$ that satisfy the constraints are also called *feasible* points.

Definition:

Let $a \in \mathcal{U}$, we say that the constraint $g_k(x) \leq 0$ (for $k \in I$) is *active* in a if $g_k(a) = 0$.

The definition of active constraints
was correct in the slides during the
lecture, but the examples were probably not all.
I apologize and will give some more examples next time.

Theorem (Karush-Kuhn-Tucker, KKT):

Let U be an open set of $(\mathbb{R}^n, || \, ||)$ and $f: U \to \mathbb{R}$, $g_k: U \to \mathbb{R}$, all $\mathcal{C}^1$

Furthermore, let $a \in U$ satisfy

$$\begin{cases} f(a) = \inf(f(x) \mid x \in \mathbb{R}^n, g_k(x) = 0 \ (\text{for } k \in E), g_k(x) \leq 0 \ (\text{for } k \in \mathrm{I}) \\ g_k(a) = 0 \ (\text{for } k \in E) \\ g_k(a) \leq 0 \ (\text{for } k \in I) \end{cases}$$

also works again for a being a local minimum

Let I_a^0 be the set of constraints that are active in a. Assume that $\left(\nabla g_k(a)\right)_{k \in E \cup I_a^0}$ are linearly independent.

Then there exist $(\lambda_k)_{1 \leq k \leq p}$ that satisfy

$$\begin{cases} \nabla f(a) + \sum_{k=1}^{p} \lambda_k \nabla g_k(a) = 0 \\ g_k(a) = 0 \ (\text{for } k \in E) \\ g_k(a) \leq 0 \ (\text{for } k \in I) \\ \lambda_k \geq 0 \ (\text{for } k \in I_a^0) \\ \lambda_k g_k(a) = 0 \ (\text{for } k \in E \cup I) \end{cases}$$

Inequality Constraint: Karush-Kuhn-Tucker Theorem

Theorem (Karush-Kuhn-Tucker, KKT):

Let U be an open set of $(\mathbb{R}^n, ||\ ||)$ and $f: U \to \mathbb{R}$, $g_k: U \to \mathbb{R}$, all $\mathcal{C}^1$

Furthermore, let $a \in U$ satisfy

$$\begin{cases} f(a) = \inf(f(x) \mid x \in \mathbb{R}^n, g_k(x) = 0 \text{ (for } k \in E), g_k(x) \leq 0 \text{ (for } k \in \mathrm{I}) \\ g_k(a) = 0 \text{ (for } k \in E) \\ g_k(a) \leq 0 \text{ (for } k \in I) \end{cases}$$

Let I_a^0 be the set of constraints that are active in a. Assume that $\left(\nabla g_k(a)\right)_{k \in E \cup I_a^0}$ are linearly independent.

Then there exist $(\lambda_k)_{1 \leq k \leq p}$ that satisfy

$$\begin{cases} \nabla f(a) + \sum_{k=1}^{p} \lambda_k \nabla g_k(a) = 0 \\ g_k(a) = 0 \text{ (for } k \in E) \\ g_k(a) \leq 0 \text{ (for } k \in I) \\ \lambda_k \geq 0 \text{ (for } k \in I_a^0) \\ \lambda_k g_k(a) = 0 \text{ (for } k \in E \cup I) \end{cases}$$

either active constraint or $\lambda_k = 0$

Descent Methods

[methods for unconstrained problems]

Descent Methods

General principle

1. choose an initial point $\boldsymbol{x}_0$, set $t = 0$
2. while not happy
   - choose a descent direction $\boldsymbol{d}_t \neq 0$
   - line search:
     - choose a step size $\sigma_t > 0$
     - set $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \sigma_t \boldsymbol{d}_t$
   - set $t = t + 1$

Remaining questions

- how to choose $\boldsymbol{d}_t$?
- how to choose σ_t?

Gradient Descent

Rationale: $\boldsymbol{d}_t = -\nabla f(\boldsymbol{x}_t)$ is a descent direction

indeed for f differentiable

$$f\big(x - \sigma\nabla f(x)\big) = f(x) - \sigma||\nabla f(x)||^2 + o(\sigma||\nabla f(x)||)$$
$$< f(x) \text{ for } \sigma \text{ small enough}$$

Step-size

- optimal step-size: $\sigma_t = \underset{\sigma}{\operatorname{argmin}} f(\boldsymbol{x}_t - \sigma\nabla f(\boldsymbol{x}_t))$
- **Line Search:** total or partial optimization w.r.t. σ
  Total is however often too "expensive" (needs to be performed at each iteration step)
  Partial optimization: execute a limited number of trial steps until a loose approximation of the optimum is found. Typical rule for partial optimization: Armijo rule (see next slides)

Typical stopping criterium:

norm of gradient smaller than ϵ

The Armijo-Goldstein Rule

Choosing the step size:

- Only decreasing f-value is not enough to converge (quickly)
- Want to have a reasonably large decrease in f

Armijo-Goldstein rule:

- also known as backtracking line search
- starts with a (too) large estimate of σ and reduces it until f is reduced enough
- what is enough?
  - assuming a linear f e.g. $m_k(x) = f(x_k) + \nabla f(x_k)^T(x - x_k)$
  - expected decrease if step of σ_k is done in direction $\boldsymbol{d}$: $\sigma_k \nabla f(x_k)^T \boldsymbol{d}$
  - actual decrease: $f(x_k) - f(x_k + \sigma_k \boldsymbol{d})$
  - stop if actual decrease is at least constant times expected decrease (constant typically chosen in [0, 1])

The Armijo-Goldstein Rule

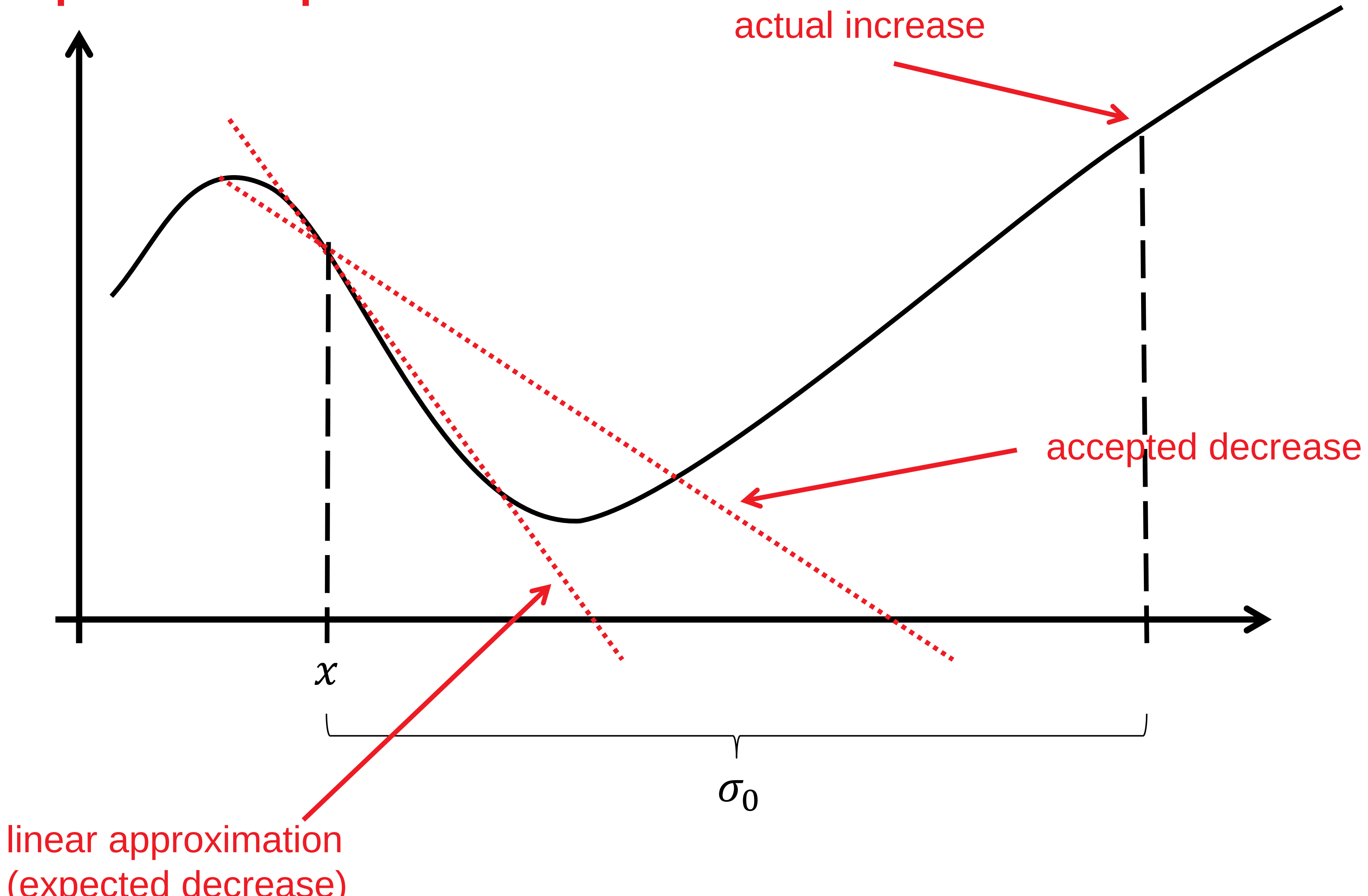
The Actual Algorithm:

Input: descent direction $\mathbf{d}$, point $\mathbf{x}$, objective function $f(\mathbf{x})$ and its gradient $\nabla f(\mathbf{x})$, parameters $\sigma_0 = 10$, $\theta \in [0,1]$ and $\beta \in (0,1)$
Output: step-size σ

Initialize σ: $\sigma \leftarrow \sigma_0$
while $f(\mathbf{x} + \sigma\mathbf{d}) > f(\mathbf{x}) + \theta\sigma\nabla f(\mathbf{x})^T\mathbf{d}$ **do**
    $\sigma \leftarrow \beta\sigma$
end while

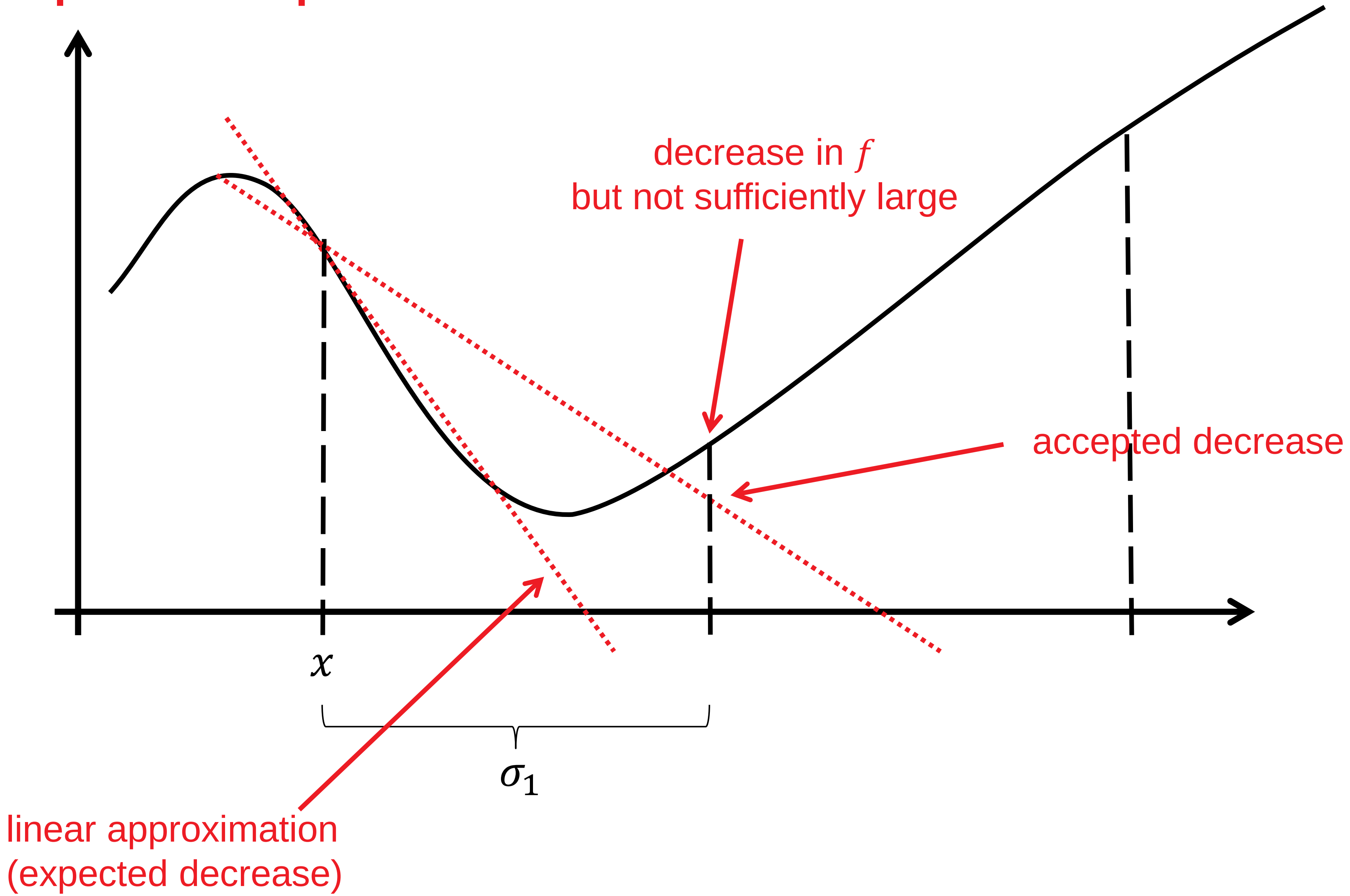
Armijo, in his original publication chose $\beta = \theta = 0.5$.
Choosing $\theta = 0$ means the algorithm accepts any decrease.

Graphical Interpretation



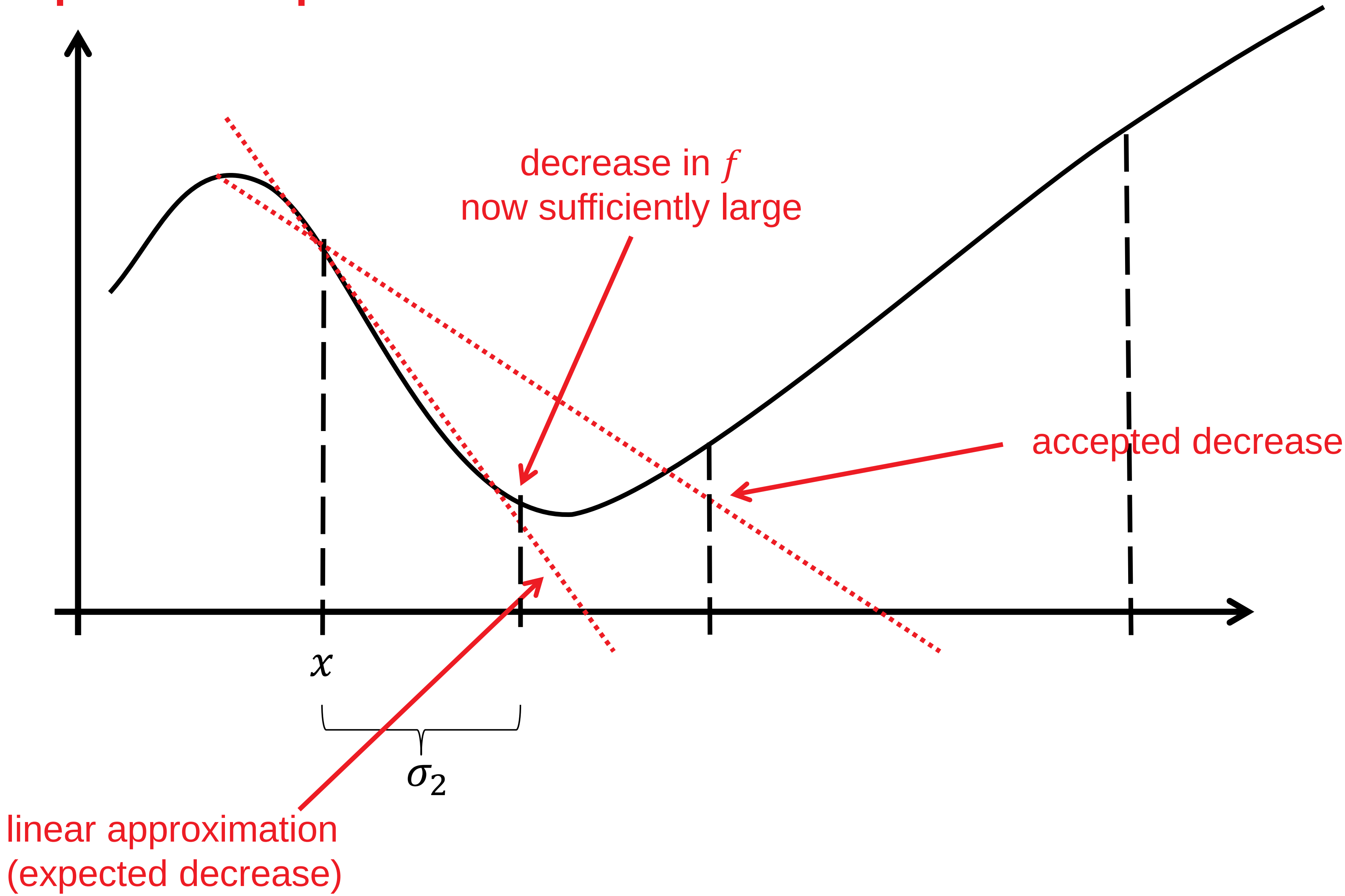
The Armijo-Goldstein Rule

Graphical Interpretation



The Armijo-Goldstein Rule

Graphical Interpretation



Gradient Descent: Simple Theoretical Analysis

Assume f is twice continuously differentiable, convex and that $\mu I_d \preccurlyeq \nabla^2 f(x) \preccurlyeq L I_d$ with $\mu > 0$ holds, assume a fixed step-size $\sigma_t = \frac{1}{L}$

Note: $A \preccurlyeq B$ means $x^T A x \leq x^T B x$ for all x

$$x_{t+1} - x^* = x_t - x^* - \sigma_t \nabla^2 f(y_t)(x_t - x^*) \text{ for some } y_t \in [x_t, x^*]$$

$$x_{t+1} - x^* = \left(I_d - \frac{1}{L} \nabla^2 f(y_t) \right) (x_t - x^*)$$

$$\text{Hence } ||x_{t+1} - x^*||^2 \leq |||I_d - \tfrac{1}{L} \nabla^2 f(y_t)|||^2 \; ||x_t - x^*||^2$$

$$\leq \left(1 - \frac{\mu}{L}\right)^2 ||x_t - x^*||^2$$

Linear convergence: $||x_{t+1} - x^*|| \leq \left(1 - \frac{\mu}{L}\right) ||x_t - x^*||$

algorithm slower and slower with increasing condition number

Non-convex setting: convergence towards stationary point

Newton Method

- descent direction: $-[\nabla^2 f(x_k)]^{-1}\nabla f(x_k)$ [so-called Newton direction]
- The Newton direction:
  - minimizes the best (locally) quadratic approximation of f: $\tilde{f}(x+\Delta x) = f(x) + \nabla f(x)^T \Delta x + \frac{1}{2}(\Delta x)^T \nabla^2 f(x) \Delta \mathrm{x}$
  - points towards the optimum on $f(x) = (x - x^*)^T A (x - x^*)$
- however, Hessian matrix is expensive to compute in general and its inversion is also not easy

quadratic convergence

$$\left(\text{i.e.} \quad \lim_{k \to \infty} \frac{|x_{k+1} - x^*|}{|x_k - x^*|^2} = \mu > 0\right)$$

Remark: Affine Invariance

Affine Invariance: same behavior on $f(x)$ and $f(Ax+b)$ for $A \in \mathrm{GLn}(\mathbb{R})$ = set of all invertible $n \times n$ matrices over $\mathbb{R}$

- Newton method is affine invariant

  see **http://users.ece.utexas.edu/~cmcaram/EE381V_2012F/Lecture_6_Scribe_Notes.final.pdf**

- same convergence rate on all convex-quadratic functions
- Gradient method not affine invariant

Quasi-Newton Method: BFGS

$x_{t+1} = x_t - \sigma_t H_t \nabla f(x_t)$ where H_t is an approximation of the inverse Hessian

Key idea of Quasi Newton:

successive iterates x_t, x_{t+1} and gradients $\nabla f(x_t)$, $\nabla f(x_{t+1})$ yield second order information

$$q_t \approx \nabla^2 f(x_{t+1}) p_t$$

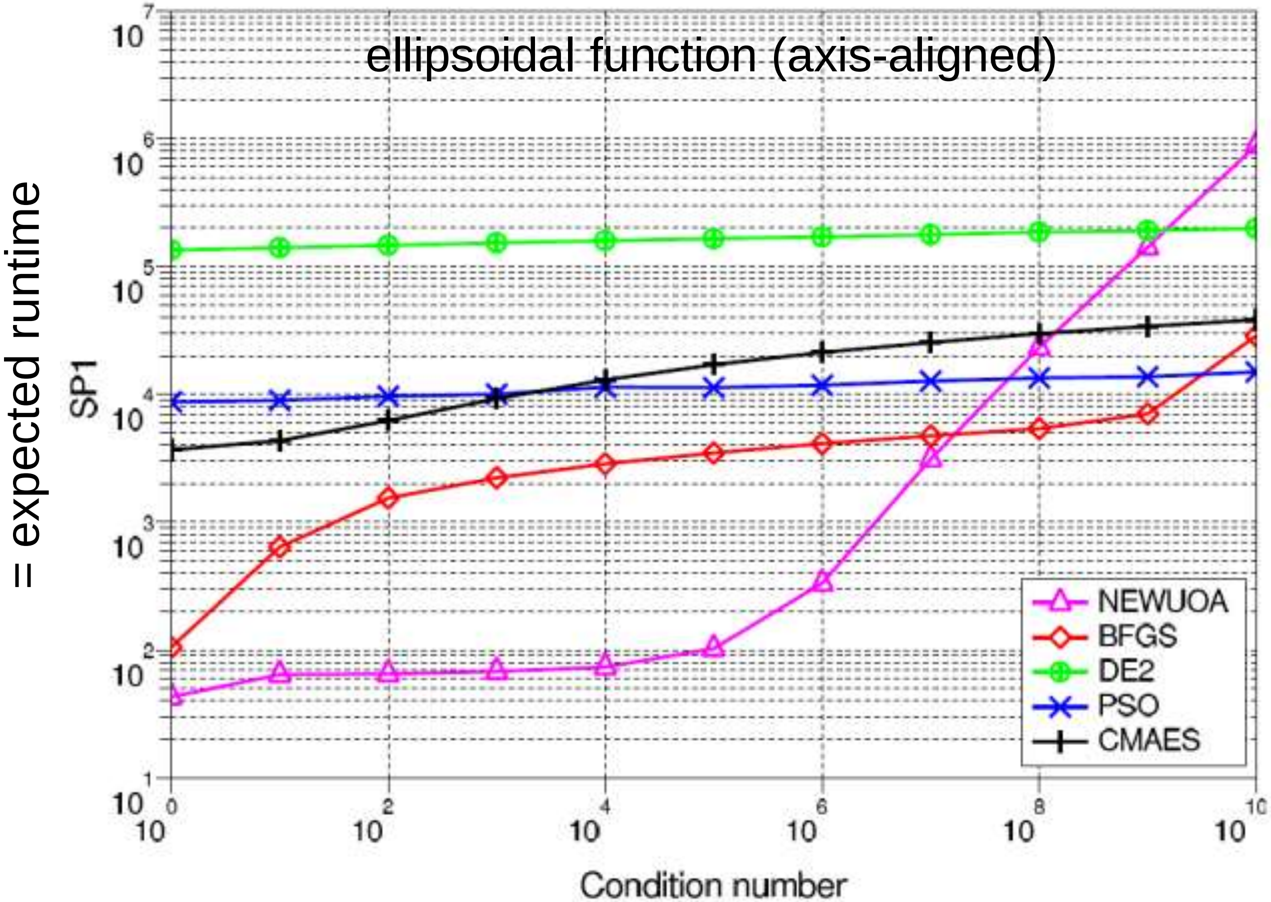
where $p_t = x_{t+1} - x_t$ and $q_t = \nabla f(x_{t+1}) - \nabla f(x_t)$

Most popular implementation of this idea: Broyden-Fletcher-Goldfarb-Shanno (BFGS)

- default in MATLAB's **`fminunc`** and python's **`scipy.optimize.minimize`**

Once Again: Invariances

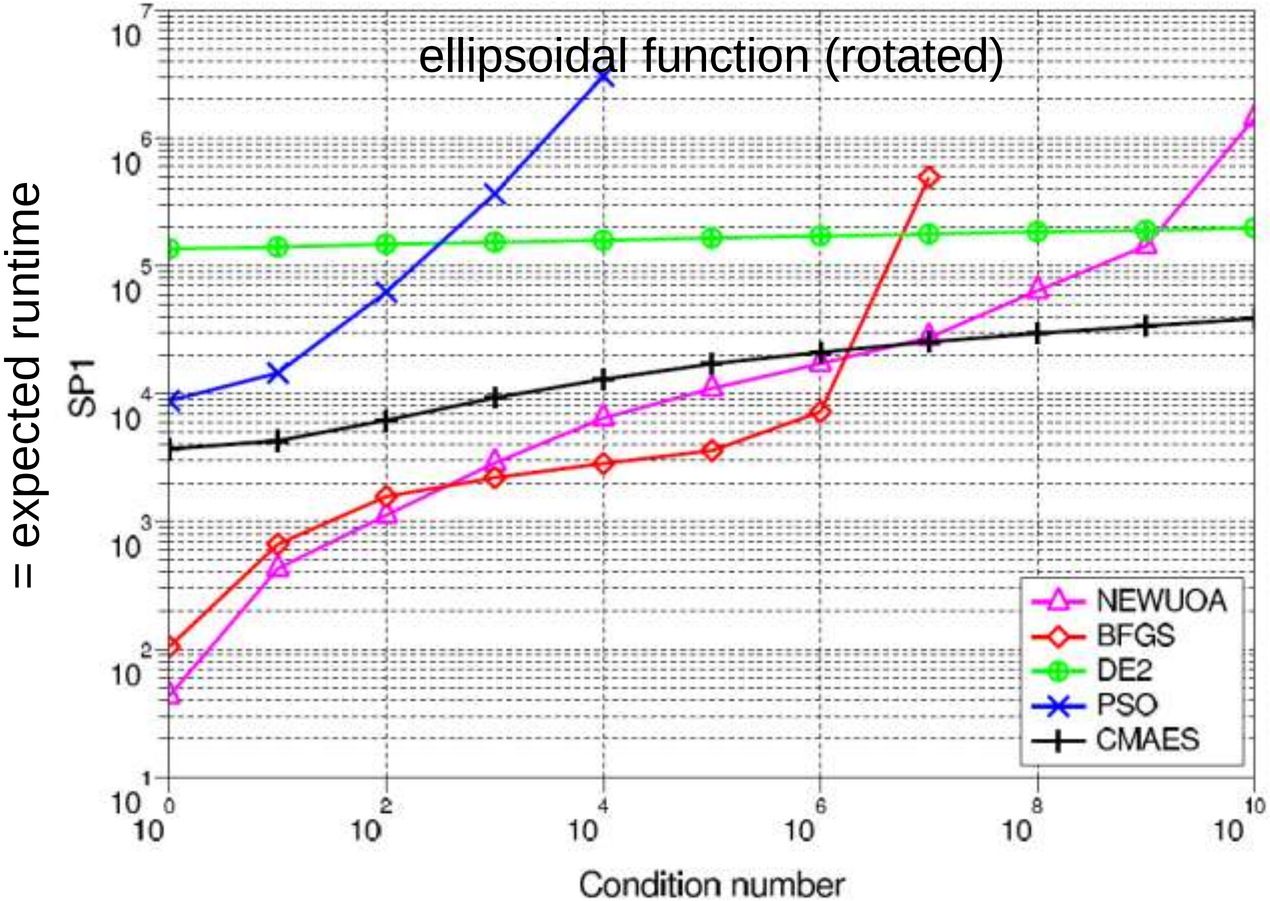
- Newton as well as BFGS are affine invariant in theory
- However, numerics might play an important role



Picture taken from “Experimental Comparisons of Derivative Free Optimization Algorithms” by A. Auger, N. Hansen, J. M. Perez Zerpa, R. Ros, and M. Schoenauer. In: *International Symposium on Experimental Algorithms*. Springer, Berlin, Heidelberg, 2009.

Once Again: Invariances

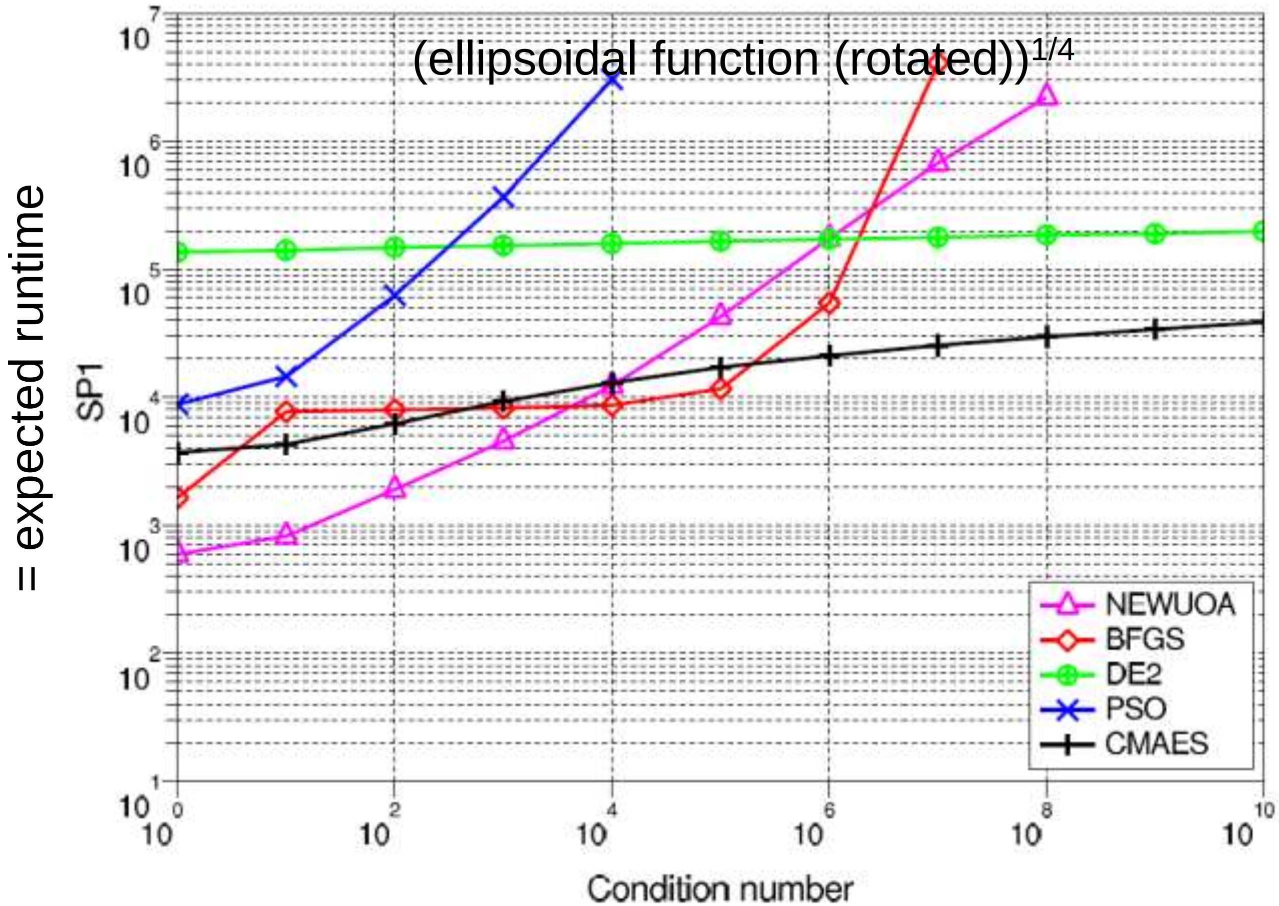
- Newton as well as BFGS are affine invariant in theory
- However, numerics might play an important role



Picture taken from “Experimental Comparisons of Derivative Free Optimization Algorithms” by A. Auger, N. Hansen, J. M. Perez Zerpa, R. Ros, and M. Schoenauer. In: *International Symposium on Experimental Algorithms*. Springer, Berlin, Heidelberg, 2009.

Once Again: Invariances

Other invariances might be nice as well, for example in function-value-free algorithms…



Picture taken from "Experimental Comparisons of Derivative Free Optimization Algorithms" by A. Auger, N. Hansen, J. M. Perez Zerpa, R. Ros, and M. Schoenauer. In: *International Symposium on Experimental Algorithms*. Springer, Berlin, Heidelberg, 2009.

discussion advanced exercise on invariance

see jupyter notebook in Edunao

Conclusions

I hope it became clear so far...

...what is the difference between gradient and Newton direction and gradient descent and (quasi-)Newton algorithms

... that adapting the step size in descent algorithms is crucial and

… what invariances are and why they are important in optimization (more details on that in the CMA-ES lecture)

Linear Optimization

[optimization with linear objective and linear constraints functions]

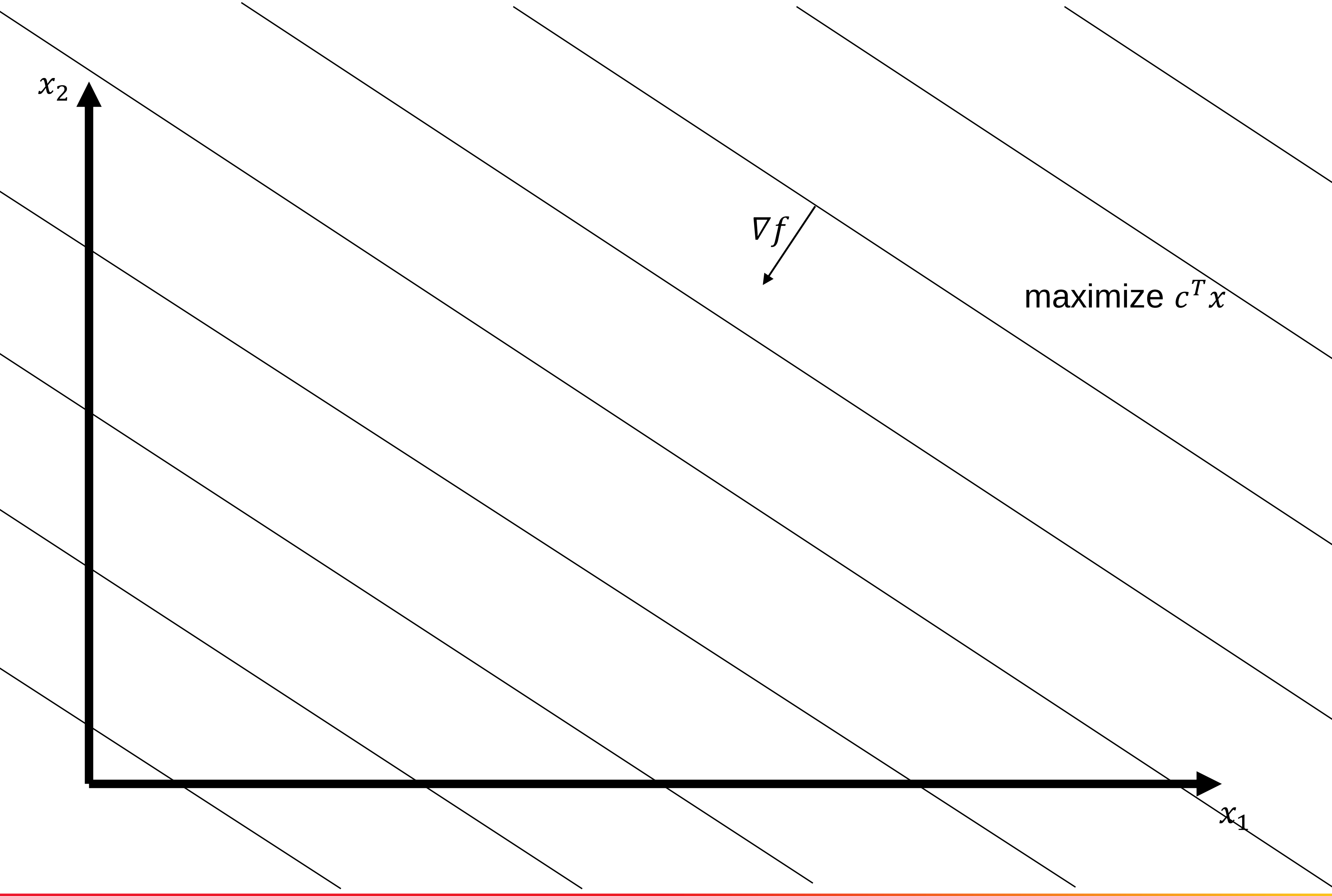
Linear Programming

Linear programming = linear optimization

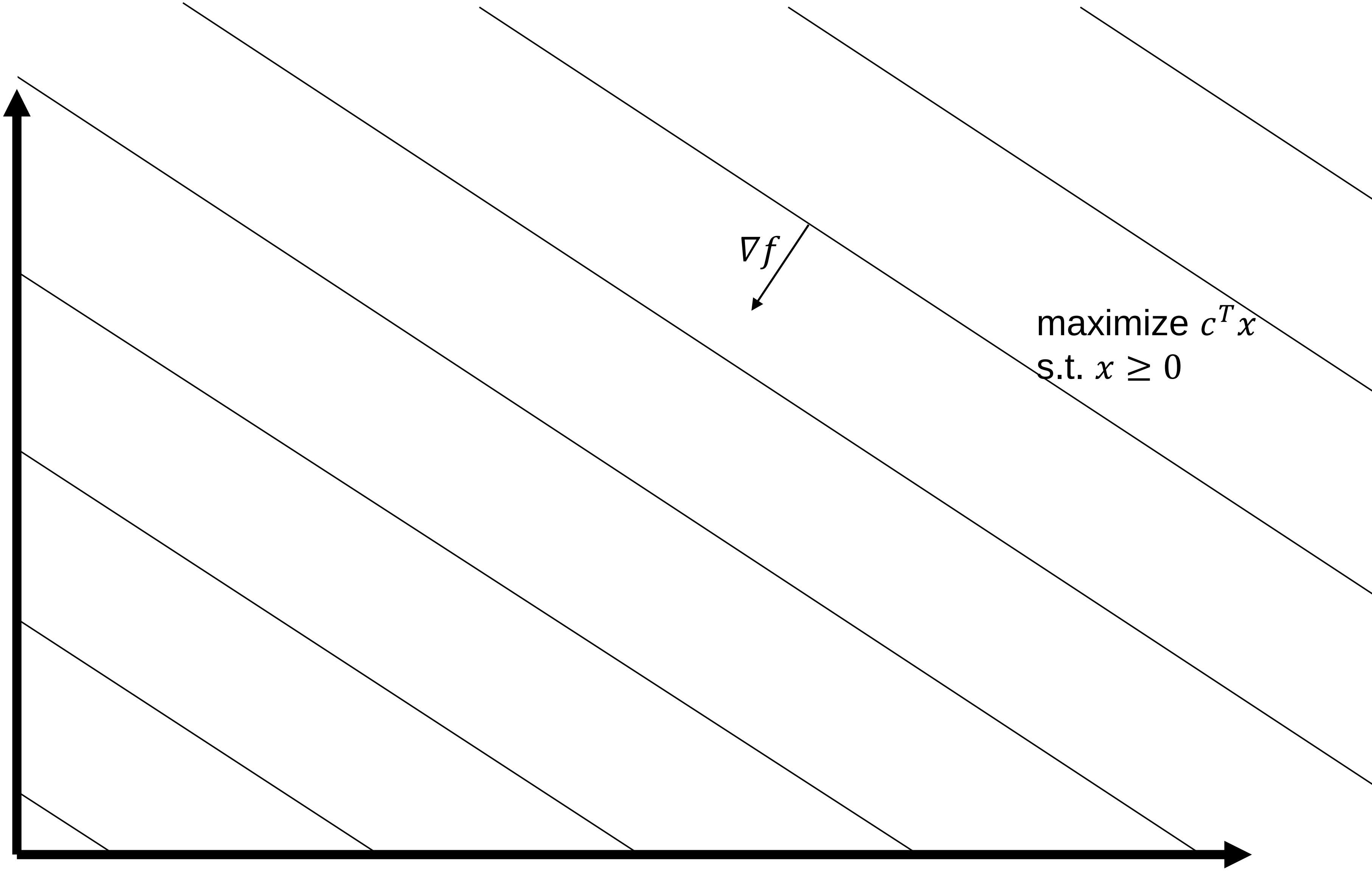
Find a vector x that

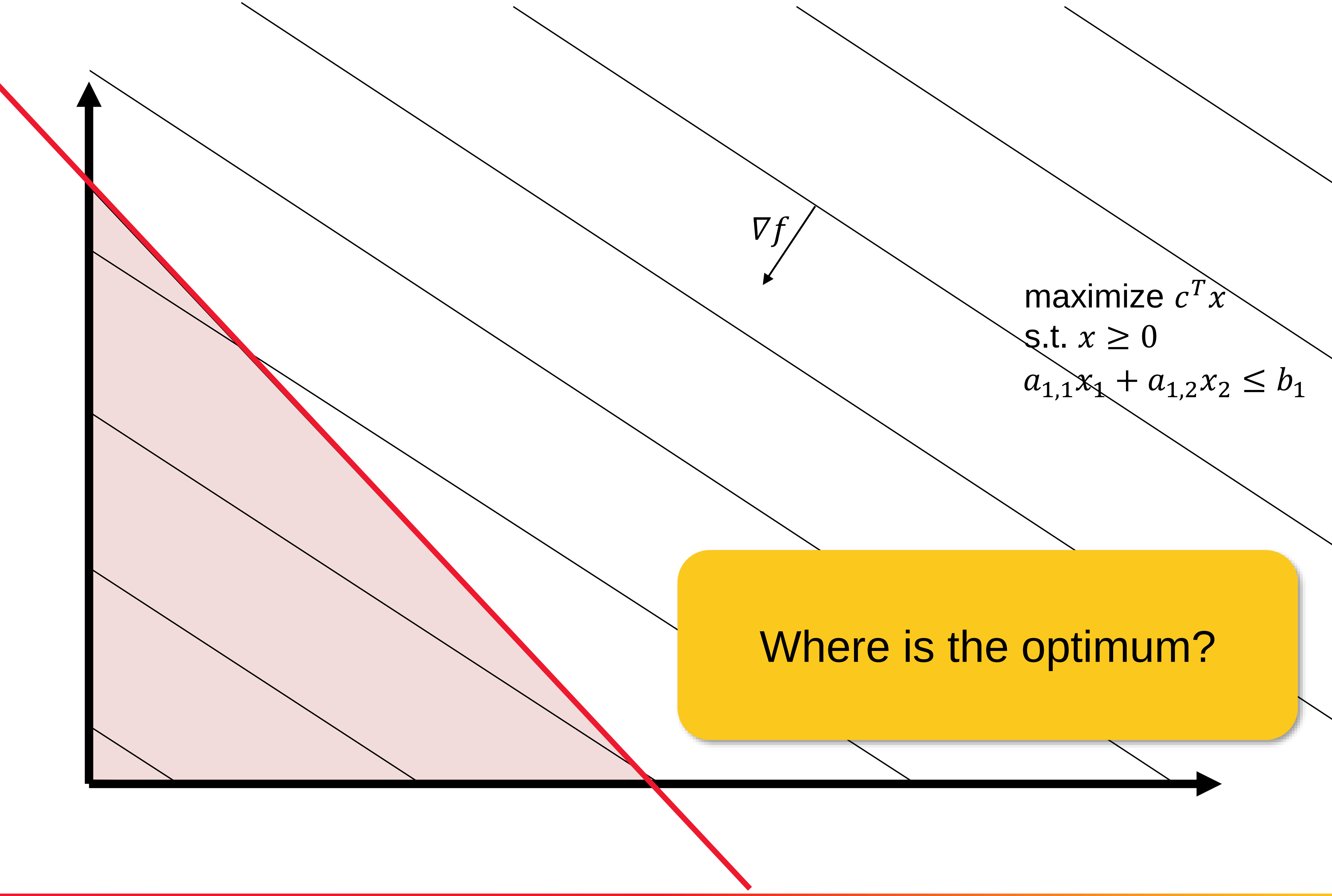
- maximizes $c^T x$
- s.t. $Ax \leq b$
- and $x \geq 0$

Graphical Illustration



Graphical Illustration

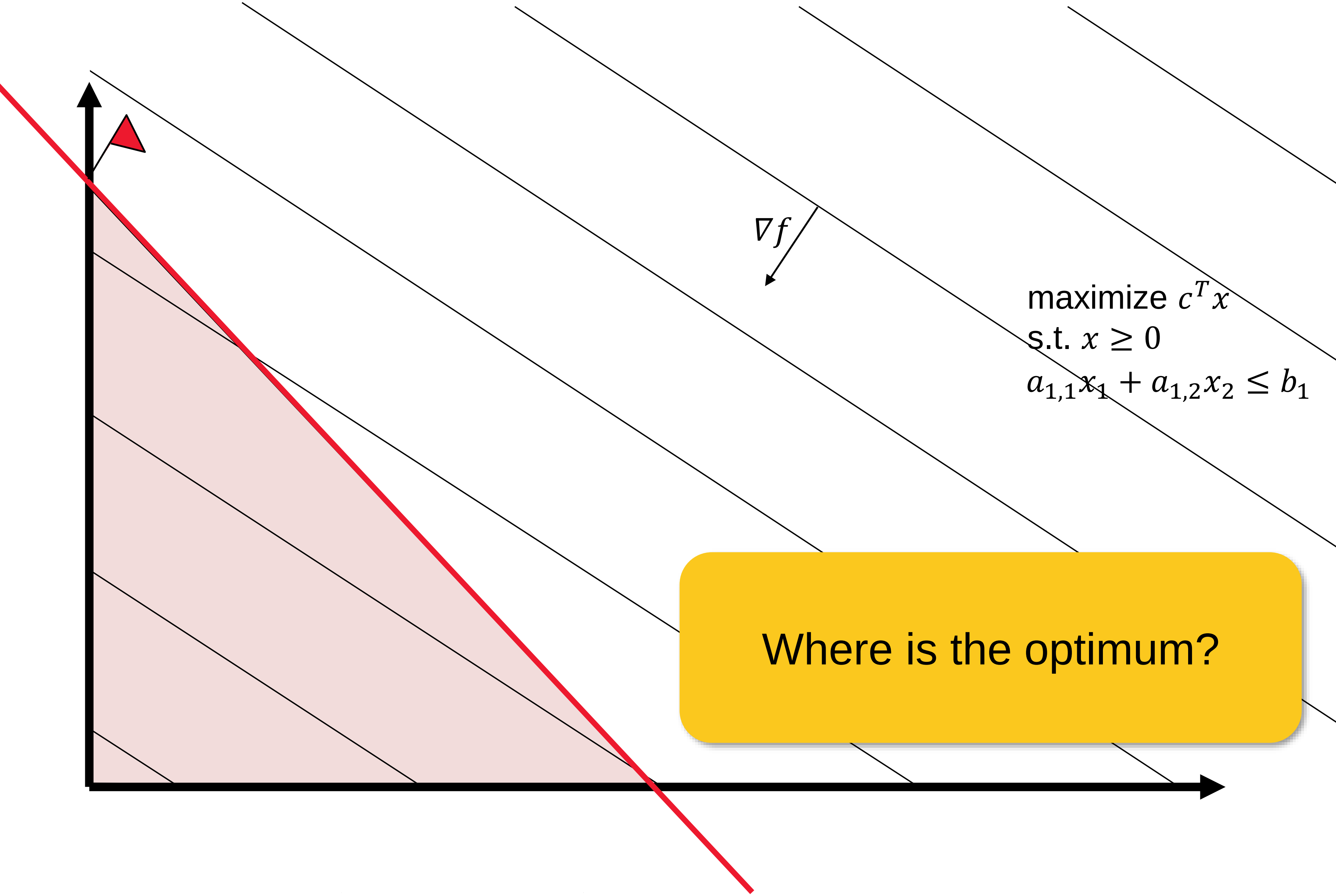




maximize $c^T x$
s.t. $x \geq 0$
$a_{1,1}x_1 + a_{1,2}x_2 \leq b_1$

Where is the optimum?

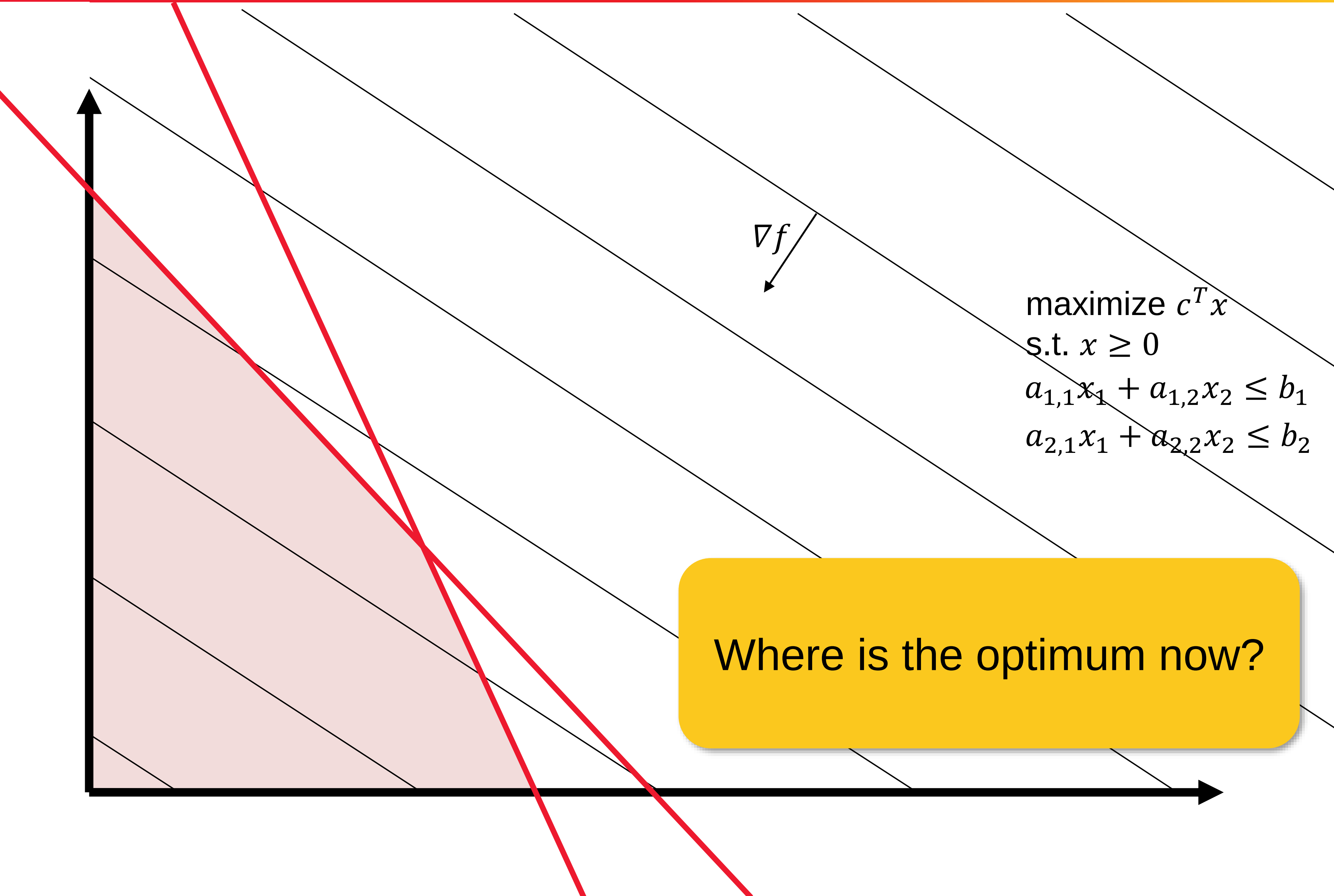
Graphical Illustration



maximize $c^T x$
s.t. $x \geq 0$
$a_{1,1}x_1 + a_{1,2}x_2 \leq b_1$

Where is the optimum?

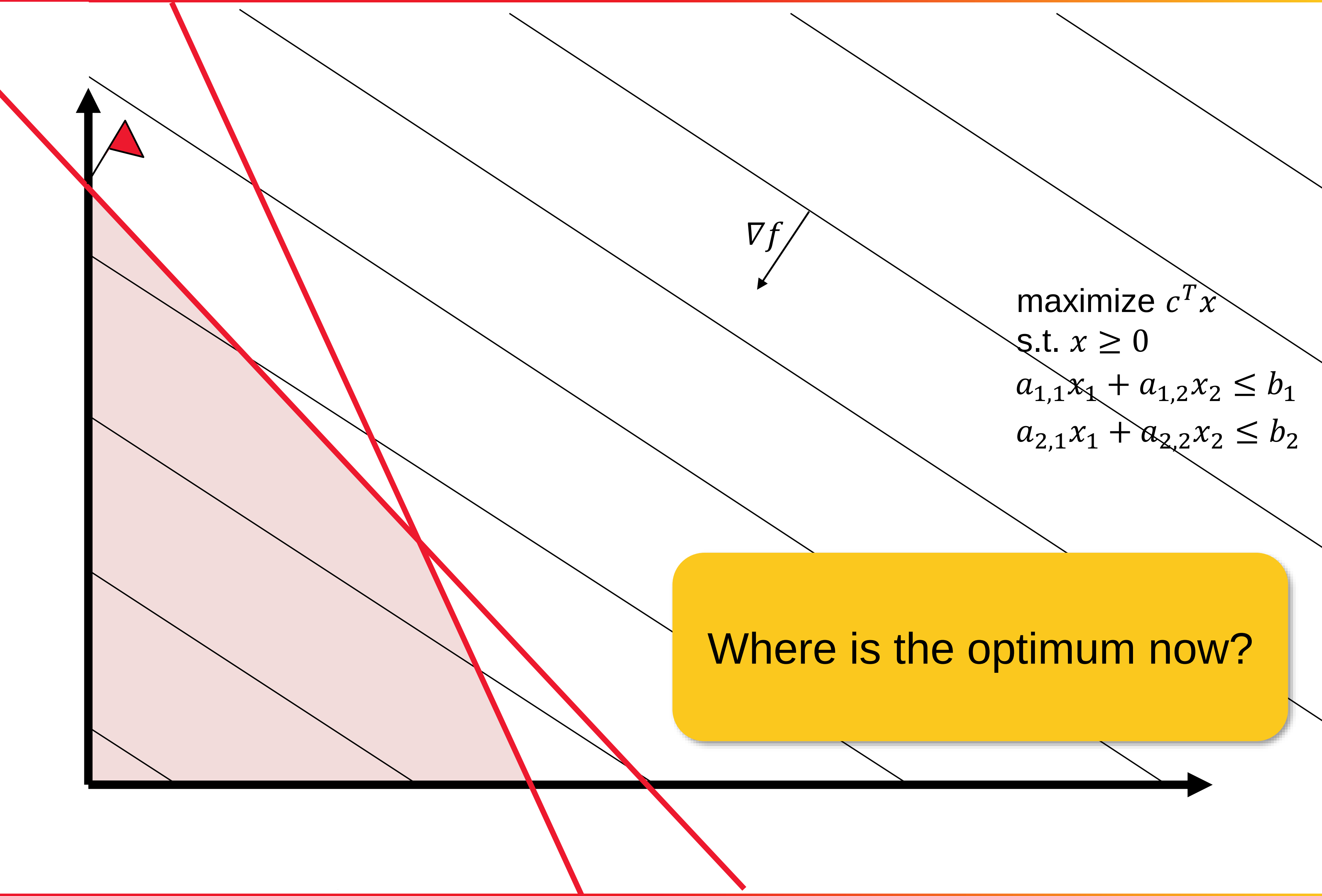
Graphical Illustration



maximize $c^T x$
s.t. $x \geq 0$
$a_{1,1}x_1 + a_{1,2}x_2 \leq b_1$
$a_{2,1}x_1 + a_{2,2}x_2 \leq b_2$

Where is the optimum now?

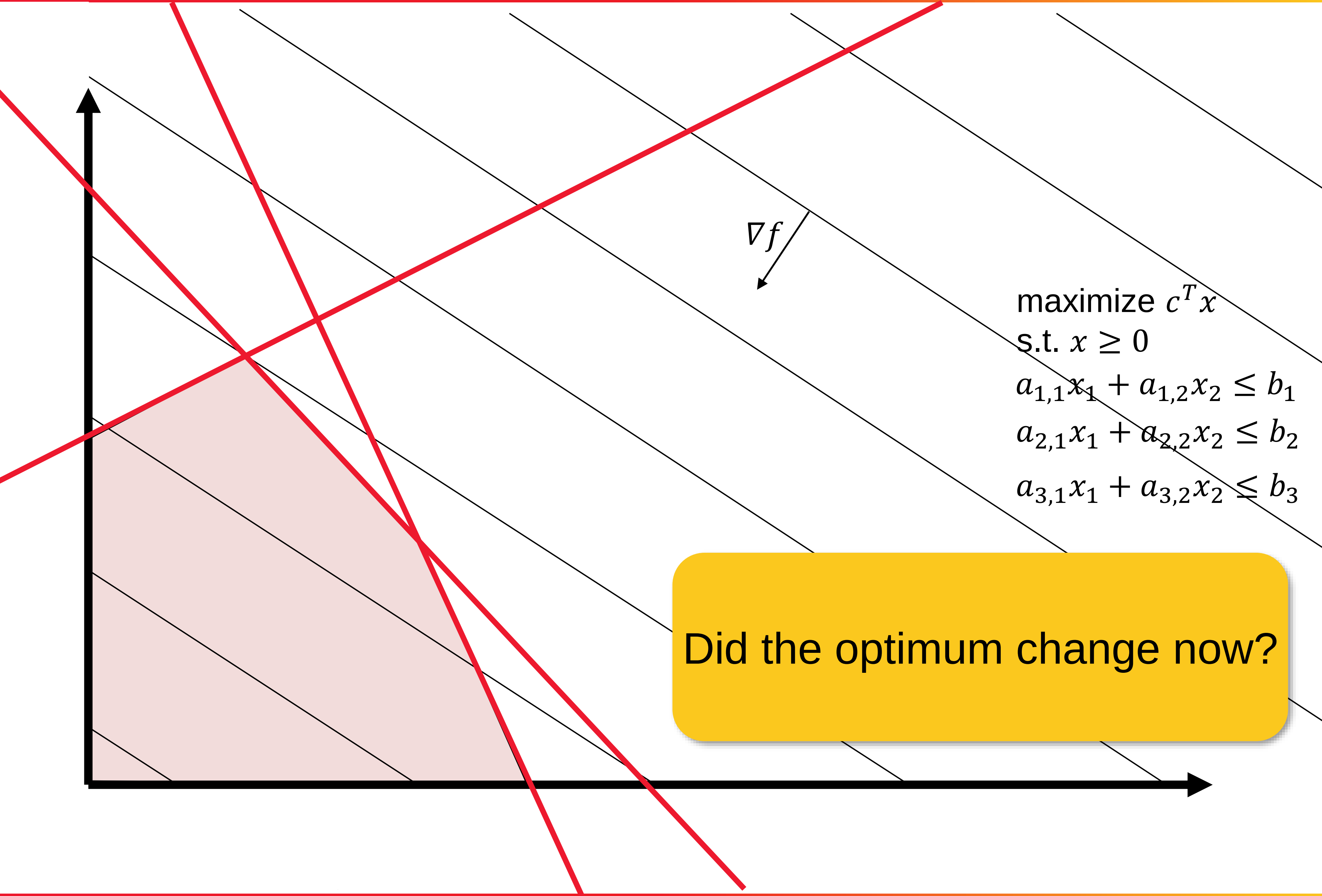
Graphical Illustration



maximize $c^T x$
s.t. $x \geq 0$
$a_{1,1}x_1 + a_{1,2}x_2 \leq b_1$
$a_{2,1}x_1 + a_{2,2}x_2 \leq b_2$

Where is the optimum now?

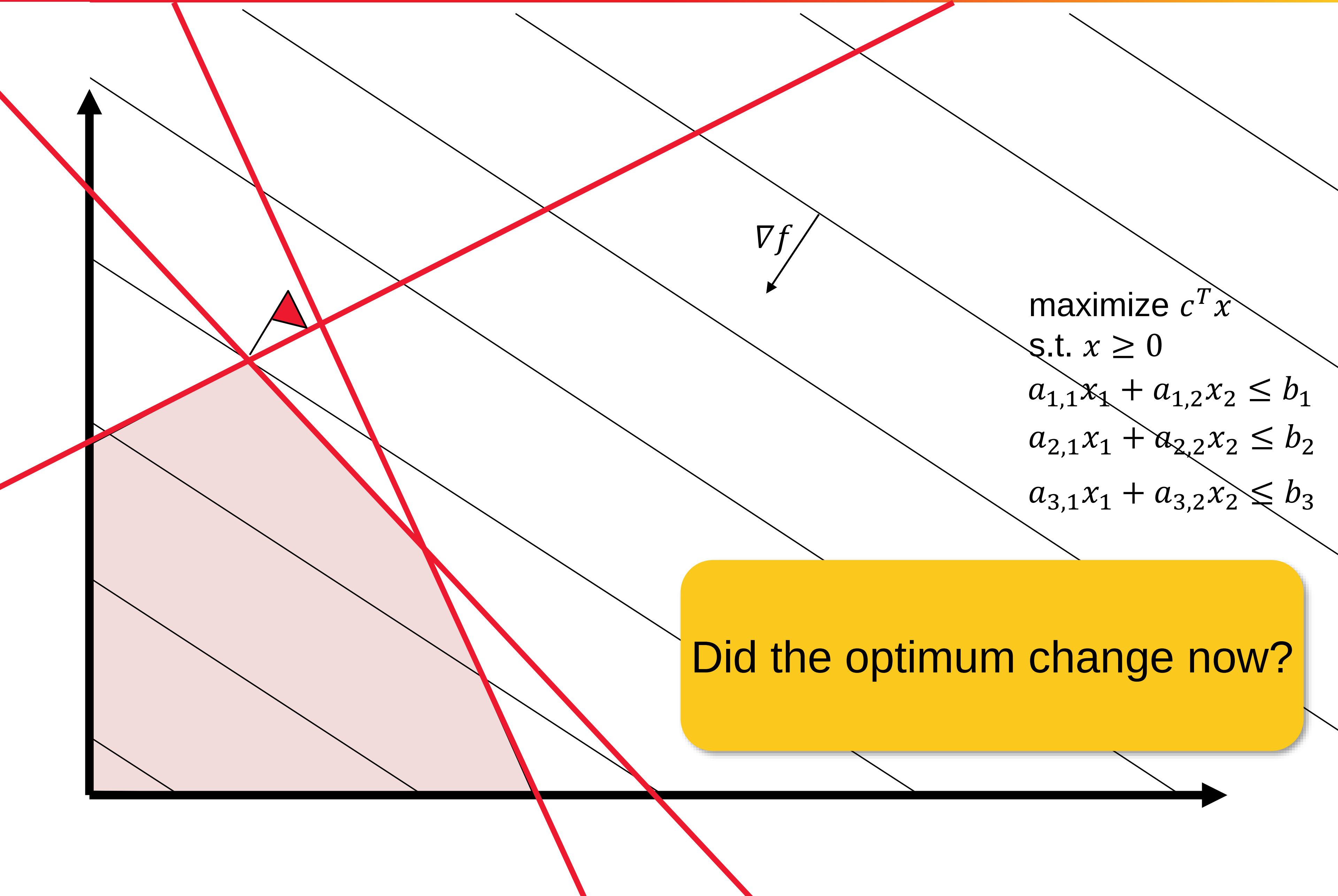
Graphical Illustration



maximize $c^T x$
s.t. $x \geq 0$
$a_{1,1}x_1 + a_{1,2}x_2 \leq b_1$
$a_{2,1}x_1 + a_{2,2}x_2 \leq b_2$
$a_{3,1}x_1 + a_{3,2}x_2 \leq b_3$

Did the optimum change now?

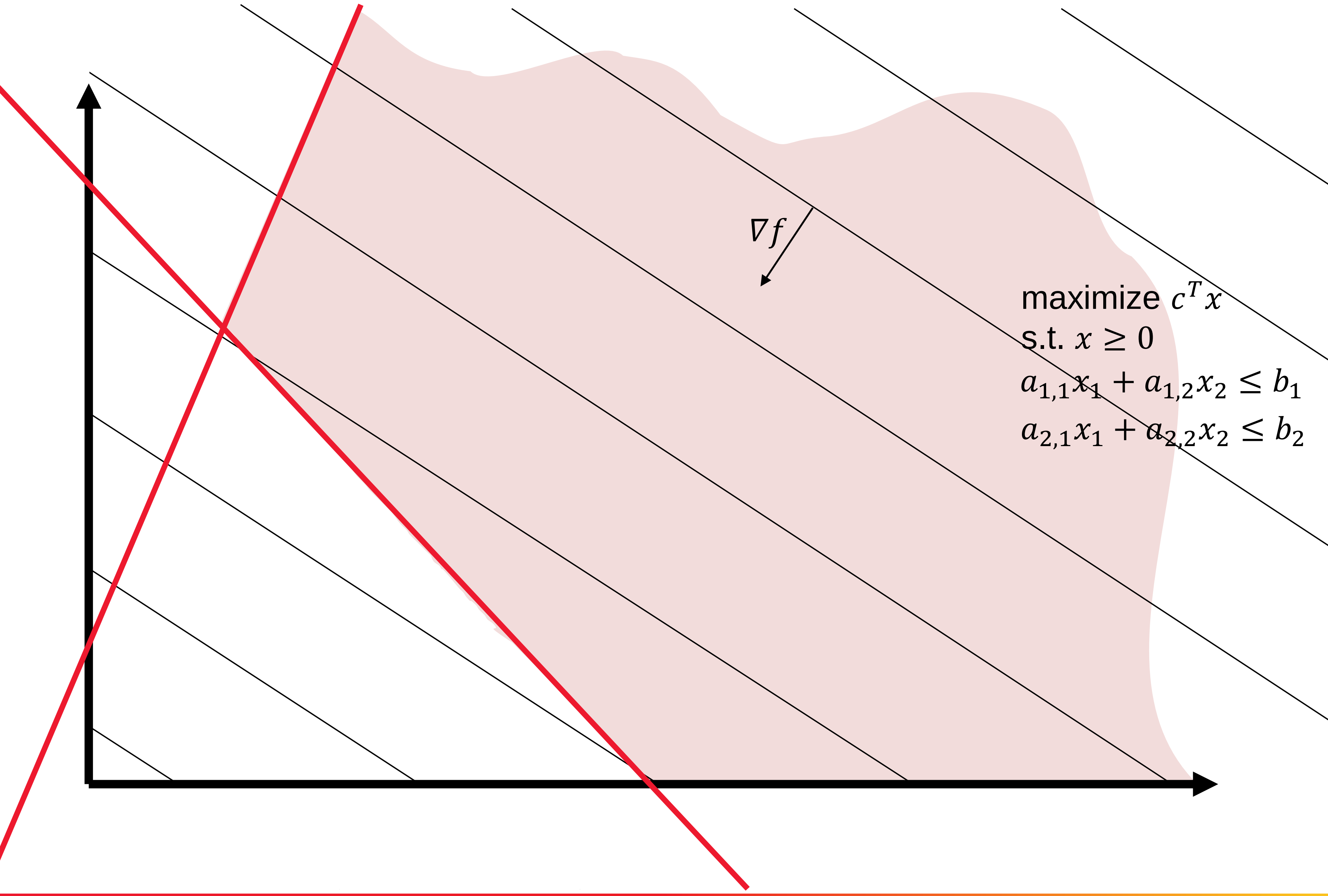
Graphical Illustration



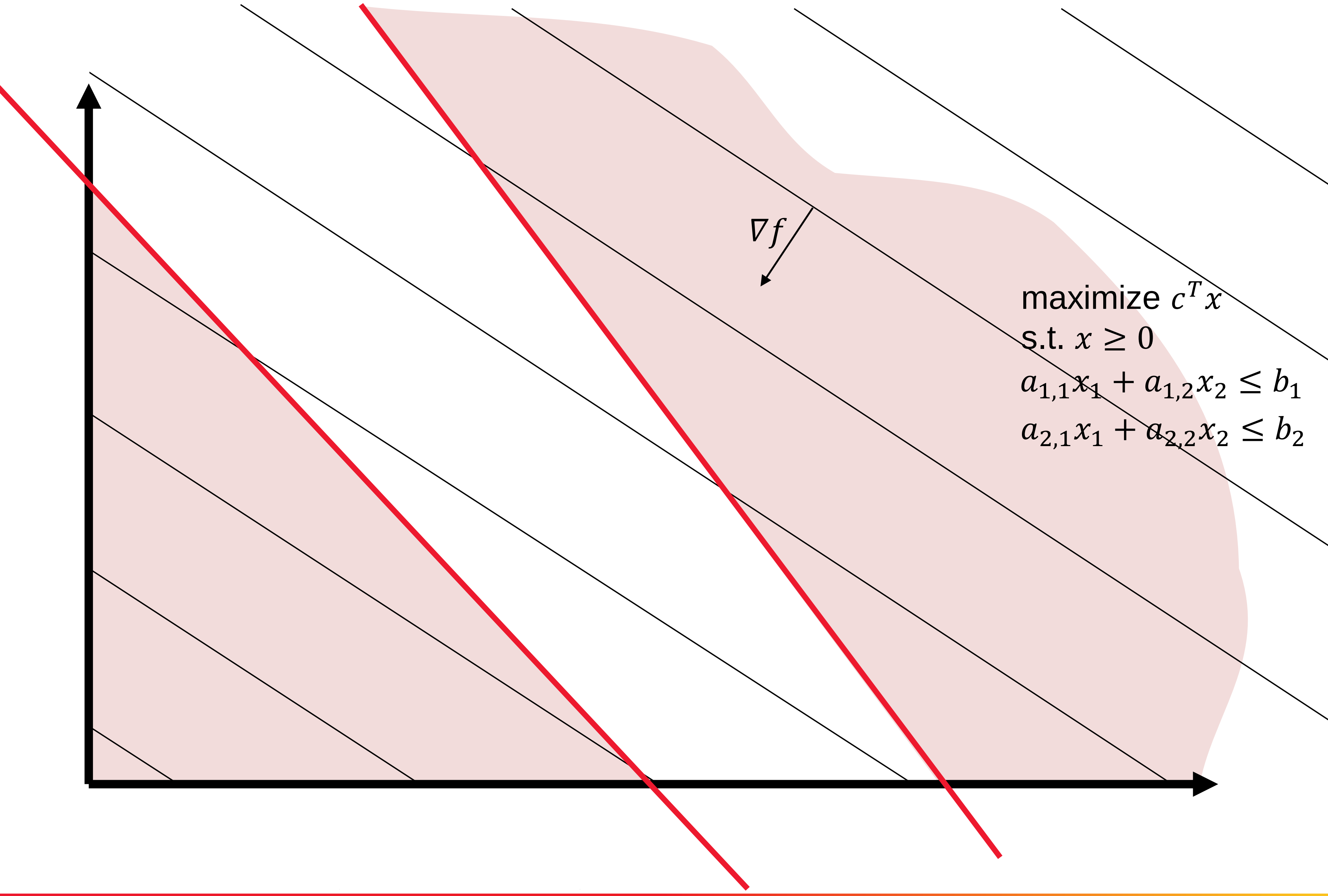
maximize $c^T x$
s.t. $x \geq 0$
$a_{1,1}x_1 + a_{1,2}x_2 \leq b_1$
$a_{2,1}x_1 + a_{2,2}x_2 \leq b_2$
$a_{3,1}x_1 + a_{3,2}x_2 \leq b_3$

Did the optimum change now?

What Can Happen: Unboundedness



What Can Happen: Infeasibility



How to Solve Linear Programs?

Simplex method (Dantzig, 1940s)

- fast in practice, but exponential in worst case

Interior point methods

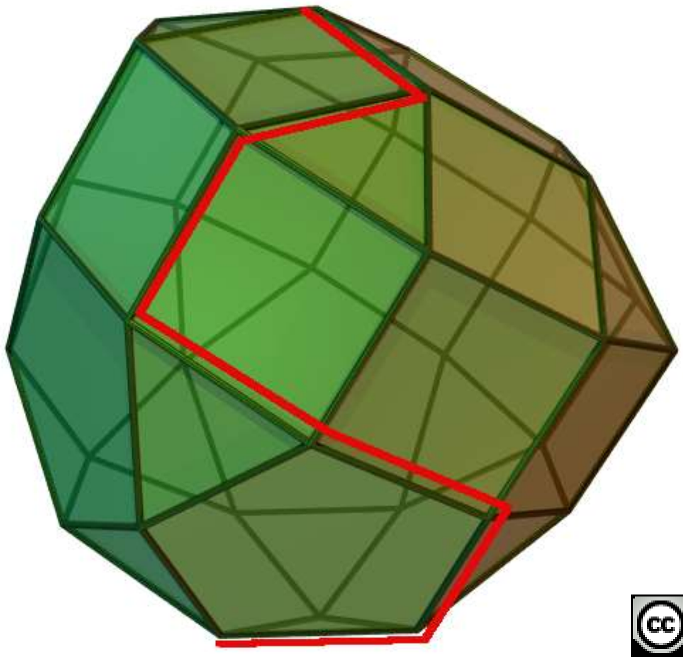
- Khachiyan, 1979: first polynomial algorithm, $O(n^6 L)$

  n: #variables, L: #input bits
- Karmarkar, 1984: $O(n^{3.5} L)$
- Vaidya, 1989: $O(n(n+d)^{1.5} L) = O(n^{2.5})$ for constant d

  d: #constraints

Idea Behind Simplex Algorithm

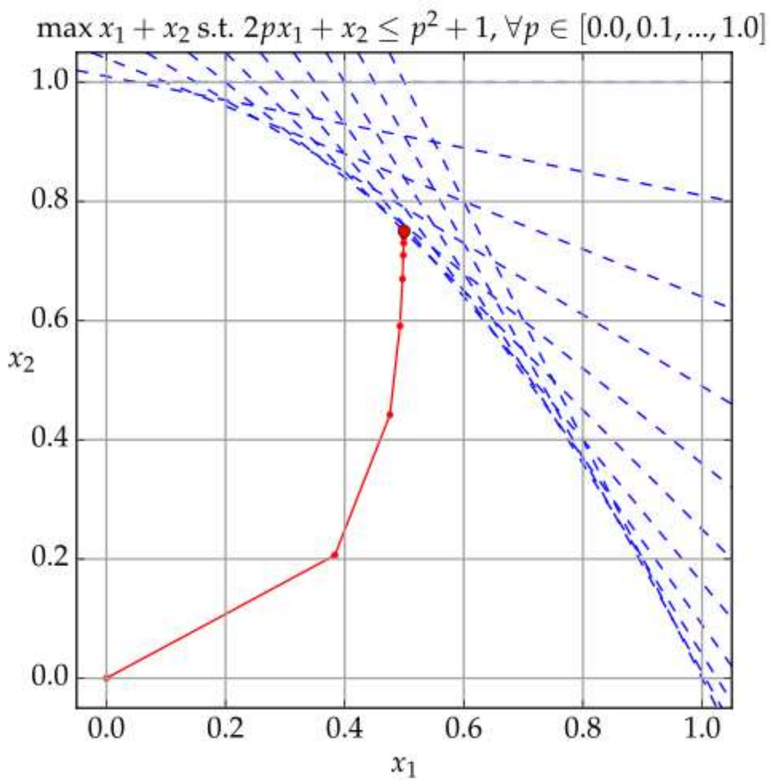
- Move along linear facets from corner to corner
- If corner not optimal, there is always a neighbor which is better
- Corresponds to equality constraints (inequality constraints need to be transformed accordingly via “slack variables”)



Idea Behind Interior Point Methods

- evaluate inside the simplex and move towards the edges
- works with inequality constraints
- solve $f(x) - 1/t \sum_{i=1}^{m} \log(g_i(x))$ iteratively with increasing t

given m inequality constraints $g_i(x) \geq 0$



Conclusions

I hope it became clear...

... what linear programming is and

... what are the ideas behind the simplex algorithm and interior point methods

Next time:

idea of duality

stochastic algorithms: stochastic gradient descent and CMA-ES