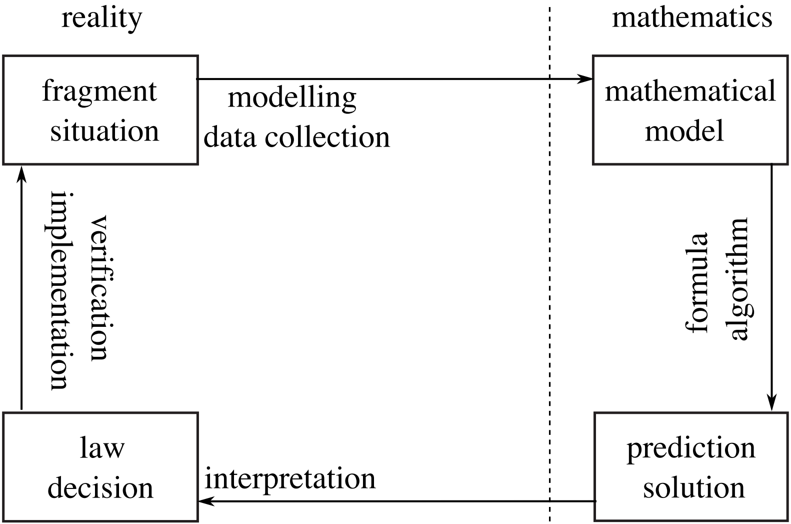
A Bird's Eye on Optimization

Antonio Frangioni

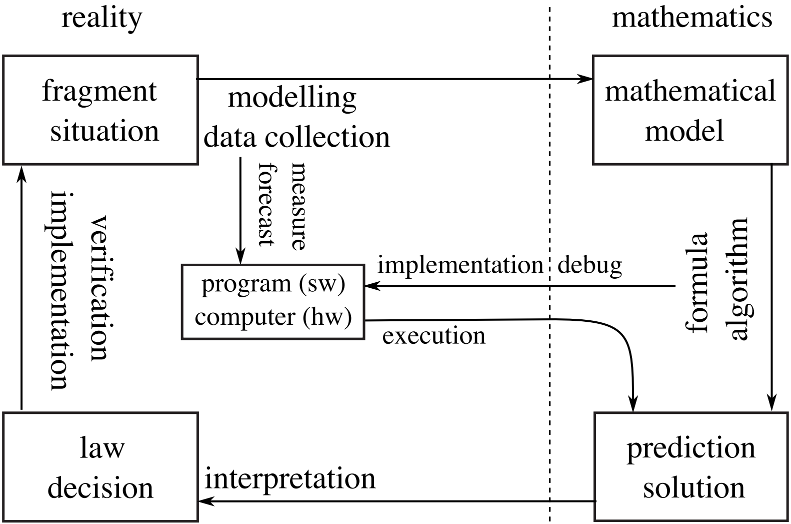
Dipartimento di Informatica, Università di Pisa

Model-Driven Decision-Making Methods (666AA)

AY 2021/22



- The fundamental cycle



- The fundamental cycle and its implementation

- Descriptive model: tells how the world (supposedly) is
- Prescriptive model: tells how the world (supposedly) should be a.k.a. optimization problem:

$$(P) \qquad f_* = \min \{\, f(x) \,:\, x \in X \,\}$$

  - arbitrary set X = feasible region of possible choices x
  - typically X specified by $G \supset X$ (ground set) + constraints dictating required properties of feasible solutions $x \in X$ [$\Longrightarrow x \in G \setminus X$ = unfeasible solution (??)]
  - $f : X \to \mathbb{R}$ objective function mapping preferences (cost)
  - optimal value $f_* \leq f(x) \forall x \in X$, $\forall v > f_* \, \exists x \in X$ s.t. $f(x) < v$
  - we want optimal solution: $x_* \in X$ s.t. $f(x_*) = f_*$
- Everything looks pretty straightforward

- Descriptive model: tells how the world (supposedly) is
- Prescriptive model: tells how the world (supposedly) should be
  a.k.a. optimization problem:

$$(P) \qquad f_* = \min \{\, f(x) \;:\; x \in X \,\}$$

  - arbitrary set X = feasible region of possible choices x
  - typically X specified by $G \supset X$ (ground set) + constraints dictating required properties of feasible solutions $x \in X$ [$\Longrightarrow x \in G \setminus X$ = unfeasible solution (??)]
  - $f : X \to \mathbb{R}$ objective function mapping preferences (cost)
  - optimal value $f_* \leq f(x) \forall x \in X$, $\forall v > f_* \, \exists x \in X$ s.t. $f(x) < v$
  - we want optimal solution: $x_* \in X$ s.t. $f(x_*) = f_*$
- Everything looks pretty straightforward . . . or is it?

- "Bad case" I: $X = \emptyset$ ("empty")

  $$\min\{\, x \ : \ x \in \mathbb{R} \wedge x \leq -1 \wedge x \geq 1 \,\}$$

  there just is no solution (which may be important to know)

- "Bad case" II: $\forall M \, \exists x_M \in X$ s.t. $f(\, x_M \,) \leq M$ ("unbounded [below]")

  $$\min\{\, x \ : \ x \in \mathbb{R} \wedge x \leq 0 \,\}$$

  there are solutions as good as you like (which may be important to know)

- Not really bad cases, just things that can happen

- Solving an optimization problem actually three different things:
  - Finding x_* and proving it is optimal (how??)
  - Proving $X = \emptyset$ (how??)
  - Constructively prove $\forall M \, \exists x_M \in X$ s.t. $f(\, x_M \,) \leq M$ (how??)

- Let's just stick to nonempty and bounded $X \implies \exists x_*$

- “Bad case” I: $X = \emptyset$ (“empty”)

  $$\min\{\, x \;:\; x \in \mathbb{R} \wedge x \leq -1 \wedge x \geq 1 \,\}$$

  there just is no solution (which may be important to know)

- “Bad case” II: $\forall M \, \exists x_M \in X$ s.t. $f(\, x_M \,) \leq M$ (“unbounded [below]”)

  $$\min\{\, x \;:\; x \in \mathbb{R} \wedge x \leq 0 \,\}$$

  there are solutions as good as you like (which may be important to know)

- Not really bad cases, just things that can happen

- Solving an optimization problem actually three different things:
  - Finding x_* and proving it is optimal (how??)
  - Proving $X = \emptyset$ (how??)
  - Constructively prove $\forall M \, \exists x_M \in X$ s.t. $f(\, x_M \,) \leq M$ (how??)

- Let’s just stick to nonempty and bounded $X \Longrightarrow \exists x_*$ . . . or does it?

- Things can be worse: not empty, not unbounded, but no x_* either:
  - $\min\{\, x \,:\, x \in \mathbb{R} \wedge x > 0 \,\}$ ("bad" X)
  - $\min\{\, 1 / x \,:\, x \in \mathbb{R} \wedge x > 0 \,\}$ ("bad" f and X)
  - $\min \left\{ f(x) = \begin{cases} x & \text{if } x > 0 \\ 1 & \text{if } x = 0 \end{cases} \,:\, x \in [\, 0 \,,\, 1 \,] \right\}$ ("bad" f)
- Still $\exists$ approximately optimal $\bar{x}$ for given $\varepsilon > 0$:

  $f(\,\bar{x}\,) - f_* \leq \varepsilon$ (absolute) or $(\, f(\,\bar{x}\,) - f_*\,) \,/\, |\, f_* \,| \leq \varepsilon$ (relative)
- Good enough for us (and anyway can't to better in general)
- Then optimizaton problems are simple objects: "just" a set and a function

- Things can be worse: not empty, not unbounded, but no x_* either:
  - $\min\{\, x \;:\; x \in \mathbb{R} \wedge x > 0 \,\}$ (“bad” X)
  - $\min\{\, 1 / x \;:\; x \in \mathbb{R} \wedge x > 0 \,\}$ (“bad” f and X)
  - $\min \left\{ f(x) = \begin{cases} x & \text{if } x > 0 \\ 1 & \text{if } x = 0 \end{cases} \;:\; x \in [\,0\,,\,1\,] \right\}$ (“bad” f)
- Still $\exists$ approximately optimal $\bar{x}$ for given $\varepsilon > 0$:

  $$f(\,\bar{x}\,) - f_* \leq \varepsilon \text{ (absolute)} \quad \text{or} \quad (\, f(\,\bar{x}\,) - f_*\,) \,/\, |\, f_* \,| \leq \varepsilon \text{ (relative)}$$

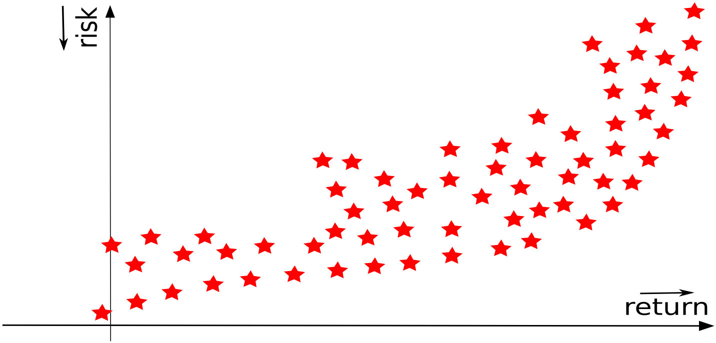
- Good enough for us (and anyway can’t to better in general)
- Then optimizaton problems are simple objects: “just” a set and a function
- Or are they?

Another Way in Which $f(\cdot)$ May Be Nasty

- Often, the actual problem is $\min \{ [f_1(x), f_2(x)] : x \in X \}$
  more than one objective, with incomparable units (apples & oranges)

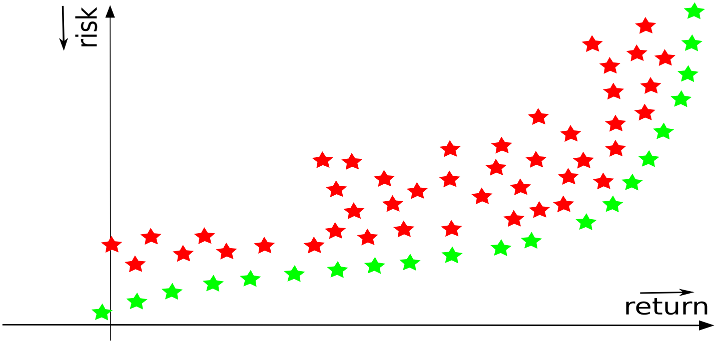
Another Way in Which $f(\cdot)$ May Be Nasty

- Often, the actual problem is $\min \{ [f_1(x), f_2(x)] : x \in X \}$
  more than one objective, with incomparable units (apples & oranges)
- Textbook example: portfolio selection problem



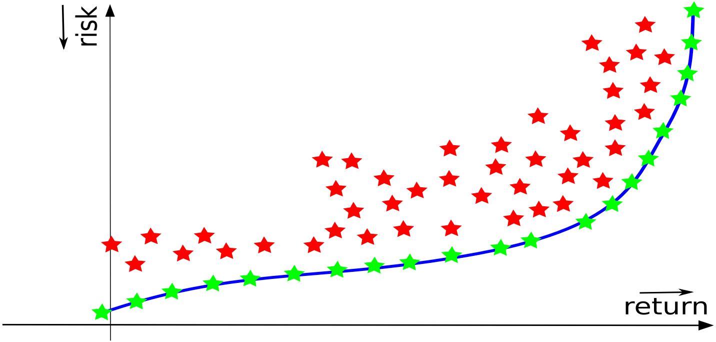
- No "best" solution, only

- Often, the actual problem is $\min\{\, [\, f_1(x)\, ,\, f_2(x)\,] \;:\; x \in X \,\}$
  more than one objective, with incomparable units (apples & oranges)
- Textbook example: portfolio selection problem



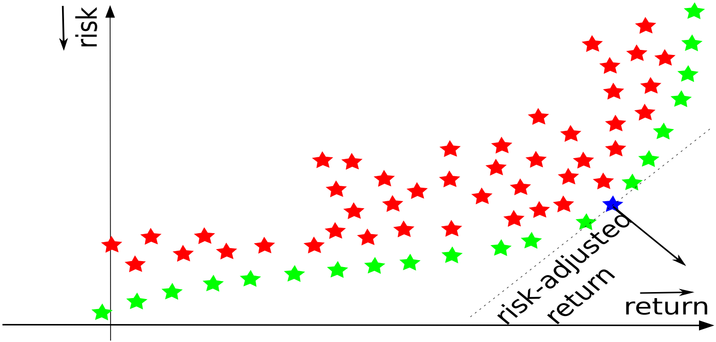
- No "best" solution, only non-dominated ones on the

- Often, the actual problem is $\min\{\, [\, f_1(x)\,,\, f_2(x)\,] \;:\; x \in X \,\}$
  more than one objective, with incomparable units (apples & oranges)
- Textbook example: portfolio selection problem



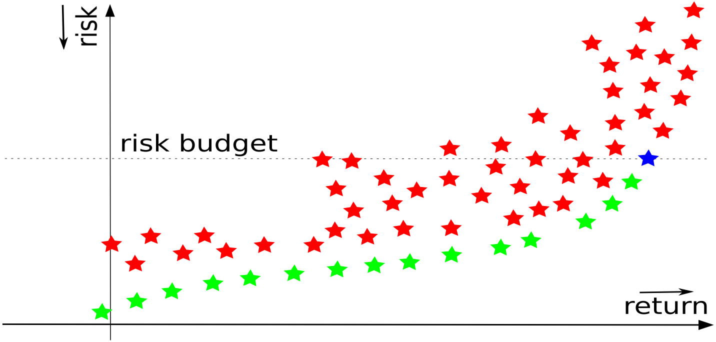
- No “best” solution, only non-dominated ones on the Pareto frontier
- Two practical solutions:

- Often, the actual problem is $\min \{ [f_1(x), f_2(x)] : x \in X \}$
  more than one objective, with incomparable units (apples & oranges)
- Textbook example: portfolio selection problem



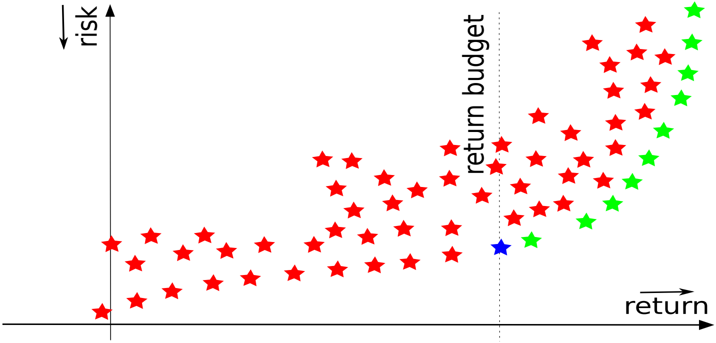
- No "best" solution, only non-dominated ones on the Pareto frontier
- Two practical solutions: maximize risk-adjusted return,
  a.k.a. scalarization $\min \{ f_1(x) + \alpha f_2(x) : x \in X \}$ (which α??)

- Often, the actual problem is $\min \{ [\, f_1(x), f_2(x) \,] : x \in X \}$
  more than one objective, with incomparable units (apples & oranges)
- Textbook example: portfolio selection problem



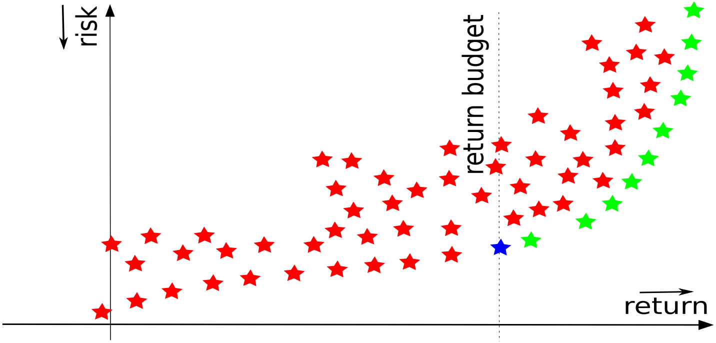
- No "best" solution, only non-dominated ones on the Pareto frontier
- Two practical solutions: maximize return with budget on maximum risk,
  a.k.a. budgeting $\min \{ f_1(x) : f_2(x) \leq \beta_2 \,,\, x \in X \}$ (which β_2??)

- Often, the actual problem is $\min \{ [\, f_1(x) \,,\, f_2(x) \,] \,:\, x \in X \}$
  more than one objective, with incomparable units (apples & oranges)
- Textbook example: portfolio selection problem



- No "best" solution, only non-dominated ones on the Pareto frontier
- Two practical solutions: minimize risk with budget on minimum return, a.k.a. budgeting $\min \{ f_2(x) \,:\, f_1(x) \leq \beta_1 \,,\, x \in X \}$ (which β_1??)

- Often, the actual problem is $\min \{ [f_1(x), f_2(x)] \ : \ x \in X \}$
  more than one objective, with incomparable units (apples & oranges)
- Textbook example: portfolio selection problem



- No "best" solution, only non-dominated ones on the Pareto frontier
- Two practical solutions: minimize risk with budget on minimum return,
  a.k.a. budgeting $\min \{ f_2(x) \ : \ f_1(x) \leq \beta_1 , \ x \in X \}$ (which β_1??)
- All a bit fuzzy, but it's the nature of the beast

- OK, let's assume $f : X \to \mathbb{R}$: then (P) is easy

- OK, let's assume $f : X \to \mathbb{R}$: then (P) is easy ... or is it?
- $X \subset G \;\equiv\;$ (indicator) function $\imath_X : G \to \{\, 0 \,,\, \infty \,\}$
- $x \in X \;\equiv\; \imath_X(\, x \,) \leq 0$ (constraint)
- All the difficulty lies in computing function values:

  $$(P) \;\equiv\; \min\left\{\, f_X(\, x \,) = f(\, x \,) + \imath_X(\, x \,) \,\right\}$$

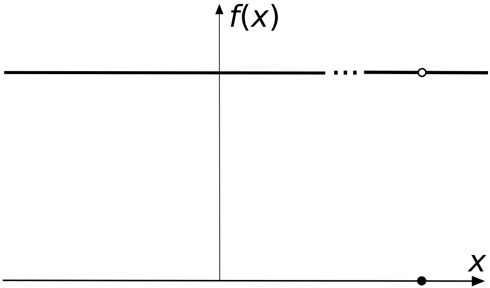
  essential objective f_X takes up all the complexity
- Vice-versa also true: f can always be linear (with complex X)

  $$(P) \equiv \min\left\{\, v \;:\; x \in X \;\;,\;\; v \geq f(\, x \,) \,\right\}$$
- Functions can be demonstrably impossible to compute $\Longrightarrow$ (P) demonstrably impossible to solve
- Even if not impossible, computing a function can be very hard

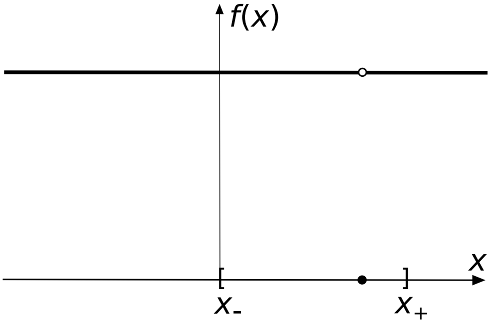
- OK, let's assume f_X is "easy to compute": then (P) is easy

- OK, let's assume f_X is "easy to compute": then (P) is easy . . . or is it?

- OK, let's assume f_X is "easy to compute": then (P) is easy ... or is it?
- Impossible even in one dimension because isolated minima can be anywhere

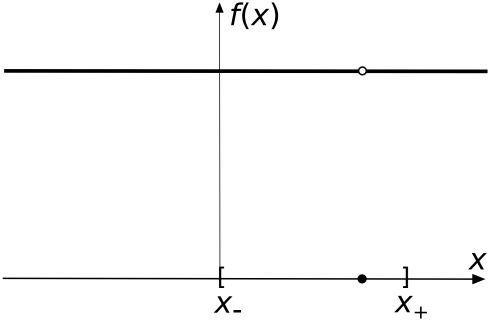


- OK, let's assume f_X is "easy to compute": then (P) is easy ... or is it?
- Impossible even in one dimension because isolated minima can be anywhere



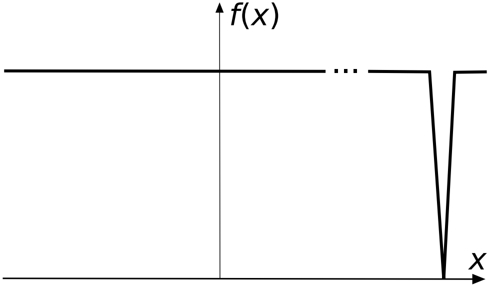
- Does it help restricting to $x \in X = [\, x_- \,,\, x_+ \,]$ $(-\infty < x_- < x_+ < +\infty)$?
- No: still uncountably many points to try

- OK, let's assume f_X is "easy to compute": then (P) is easy ... or is it?
- Impossible even in one dimension because isolated minima can be anywhere



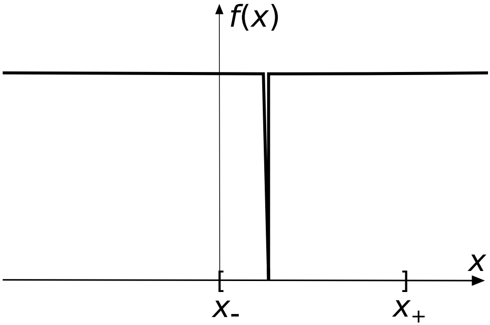
- Does it help restricting to $x \in X = [\, x_- \, , \, x_+ \,]$ $(-\infty < x_- < x_+ < +\infty)$?
- No: still uncountably many points to try
- Is it because f "jumps"?

- OK, let's assume f_X is "easy to compute": then (P) is easy ... or is it?
- Impossible even in one dimension because isolated minima can be anywhere



- Does it help restricting to $x \in X = [\, x_- \, , \, x_+ \,]$ $(-\infty < x_- < x_+ < +\infty)$?
- No: still uncountably many points to try
- Is it because f "jumps"? No, f can have isolated $\downarrow$ spikes anywhere

- OK, let's assume f_X is "easy to compute": then (P) is easy ... or is it?
- Impossible even in one dimension because isolated minima can be anywhere



- Does it help restricting to $x \in X = [\, x_- \,,\, x_+ \,]$ $(-\infty < x_- < x_+ < +\infty)$?
- No: still uncountably many points to try
- Is it because f "jumps"? No, f can have isolated $\downarrow$ spikes anywhere
- ... even on $X = [\, x_- \,,\, x_+ \,]$ as spikes can be aribtrarily narrow

- Impose $X = [\, x_- \,,\, x_+ \,]$ with $D = x_+ - x_- < \infty$ (finite diameter)
- Impose spikes can't be arbitrarily narrow $\equiv f$ cannot change too fast $\equiv$ f Lipschitz continuous (L-c) on X $\exists L > 0$ s.t.

  $$|\, f(\, x\,) - f(\, y\,)\,| \leq L|\, x - y\,| \qquad \forall x, y \in X$$
- f L-c $\Longrightarrow$ a fortiori f does not "jump" (continuous)
- f L-c $\Longrightarrow$ one ε-optimum can be found with $O(\, LD/\varepsilon\,)$ evaluations: uniformly sample X with step $2\varepsilon/L$
- Bad news: no algorithm can work in less than $\Omega(\, LD/\varepsilon\,)$
- # steps inversely proportional to accuracy, just not doable for "small" ε
- Even very dramatically worse if $X \subset \mathbb{R}^n$ (will see)
- Also, L generally unknown and not easy to estimate (will see) but algorithms actually require/use it

- (P) where $f(\cdot)$ and $\imath_X(\cdot)$ are "just any function" $\equiv$ complex mathematical model with no closed formulæ (most of them):
  - numerical integration
  - systems of PDEs
  - electromagnetic propagation models (ray-tracing, ...)
  - heat propagation models (heating/cooling of buildings, ...)
  - systems with complex management procedures (storage/plant design with route/machine optimization ...)
  - systems with stochastic components (+ possibly complex management) (queues in ERs, users of cellular networks, ...)
- A.k.a. simulation-based optimization: the system can only be numerically simulated as opposed to algebraically described
- Computation of $f_X(x)$ costly (can do few 100s/1000s of them)
- No information about the behaviour of $f(\cdot)$ "close" to x

- Typically require bound constraints: w.l.o.g. $X = [\,0\,,\,1\,]^n$ and other constraints "hidden" in $f(\,\cdot\,)$

- Basically only (clever) "shotgun approach": fire enough rounds and eventually a good solution happens

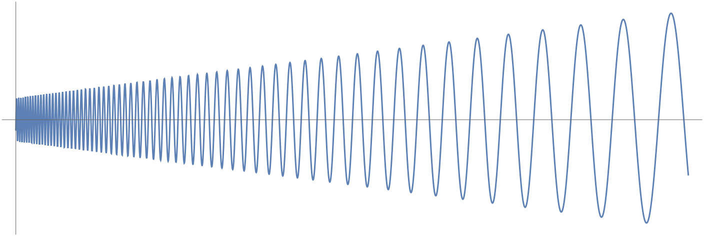
- Good playground for population-based approaches (genetic algorithms, particle swarm, ...)

- Any other standard search (simulated annealing, taboo search, GRASP, variable-neighbourhood search, ...)

- Better idea: construct a model of $f(\,\cdot\,)$ out of past iterates to drive the search (regression, kriging, radial-basis functions, SVR, ML, ...)

- Bad news: none of these can possibly work efficiently (in theory)

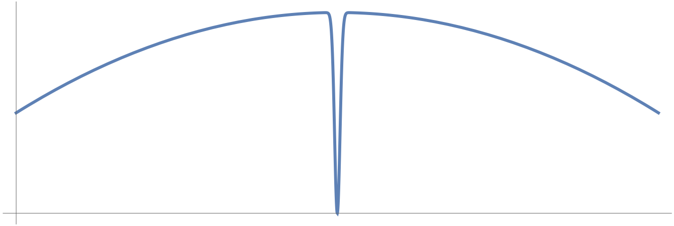
- If $f(\cdot)$ "swings wildly", things can be arbitrarily bad



- Assume $f : \mathbb{R}^n \to \mathbb{R}$ L-c with known constant L
- For each algorithm $\exists\, f(\,\cdot\,)$ s.t. finding ε-optimal solution requires $\Omega(\,L\,/\,\varepsilon\,)^n$ evaluations — that's very bad
- No free lunch theorem says "all algorithms equally bad"
- In practice is not as bad, but cost indeed grows very rapidly with n
- $n \approx 10 - 100$ if $f(\,\cdot\,)$ very costly, perhaps $n \approx 1000$ if not too costly

- $f(x)$ may be a random process:
  average performance computed via Montecarlo out of simulations
- Many examples:
  - behaviour of users
  - impact of weather on energy production/consumption
  - errors in measurement/impurity of materials ...
- Interesting tidbit: almost all approaches are inherently randomized
  (if you don't know anything, you may as well throw dices)
- Good part: can be trivially parallelized (as all Montecarlo do)
- Bad part: many runs = costly to compute average with high accuracy
- Intuitively, high accuracy only needed close to x_*
- But how do I tell if I'm close x_*? And which x_*?

- In a nutshell: if everything goes very, very well
  - you don't have many parameters (n in the few tens, ...)
  - you don't really need the best solution, a good one is OK
  - you have a lot of time and/or a supercomputer at hand
  - f is "nice enough": Lipschitz continuous, no isolated local minima, ...



- Good news: plenty of general-purpose black-box solvers, simple to use
- Bad news: difficult to choose/tune, none will ever scale to large-size
- In many cases, it is just what is needed
- Can we do better? Yes, we can if we have more structure

Let's Pry Open That Black Box

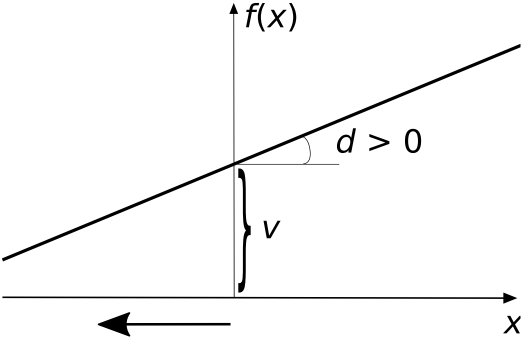
- Fundamental concept: if you know the structure of $f(\cdot)\,/\,X$, exploit it
- Very important structure: Partial Differential Equations
- Model disparate phenomena as such as:
  - sound, heat, diffusion
  - electromagnetism (Maxwell's equations)
  - fluid dynamics (Navier–Stokes equations)
  - elasticity, ...
- Countless many applications:
  - weather forecast, ocean currents, pollution diffusion, ...
  - flows in pipes (water, gas, blood, ...)
  - air flow (airplane wing, car, wind turbine, ...)
  - behaviour of complex materials/objects (buildings, seismic models, ...)
- Optimal design/operation of many systems has PDE-defined $f(\cdot)/X$: PDE-Constrained Optimization (PDE-CO) problem

- General form of the problem:
  $$\text{(PDE-CO)} \quad \min \{ f(c, s) \,:\, \mathcal{H}(c, s) = 0 \;,\; \mathcal{G}(c, s) \geq 0 \}$$
  $x = [c, s]$, explicit description of X:
  - $s =$ state (pressure/velocity of air, force in material, ...)
  - $c =$ controls (shape of wing/blade, position of actuators, ...)
  - $f(c, s) =$ measure of function $\Longrightarrow$ typically involves integrals
  - $\mathcal{H}(c, s) =$ PDE constraints (Navier–Stokes equations, ...)
  - $\mathcal{G}(c, s) =$ "other" algebraic constraints (min/max size/position, ...)
- Each $s_i : \mathbb{R}^k \to \mathbb{R}$ a function: $X \subset \mathbb{F}^n$
- Often k small-ish: 2D/3D coordinates, fields, time (optimal control)
- Controls may be functions or "simple" reals ($\equiv$ linear functions)
- $\mathbb{F}^n$ is a whole lot bigger than even $\mathbb{R}^n$ (all functions vs. linear ones): Banach space, infinite-dimensional while $\mathbb{R}^n$ has finite dimension n
- What did I gain from knowing f, $\mathcal{H}$, $\mathcal{G}$?

- (the) "Space ($\mathbb{F}^n$) is big. Really big. You just won't believe how vastly, hugely, mind-bogglingly big it is."

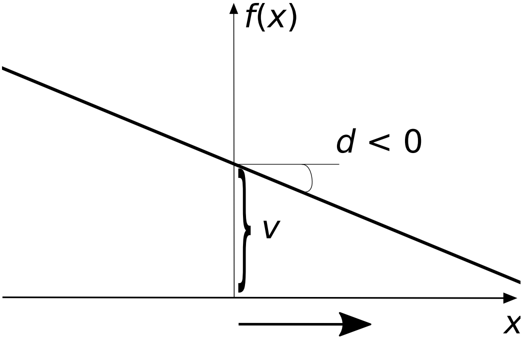
- (the) "Space ($\mathbb{F}^n$) is big. Really big. You just won't believe how vastly, hugely, mind-bogglingly big it is." Which way is x_*?

- (the) "Space ($\mathbb{F}^n$) is big. Really big. You just won't believe how vastly, hugely, mind-bogglingly big it is." Which way is x_*?



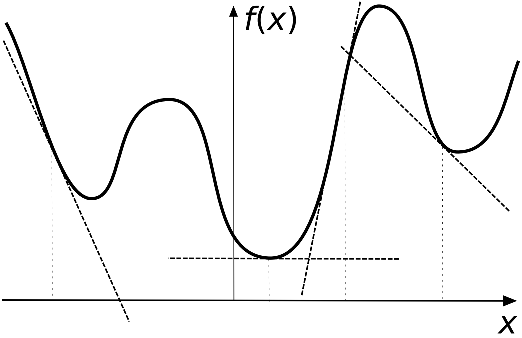
- $f(x) := dx + v : \mathbb{R} \to \mathbb{R}$ (linear) easy: always left if $d > 0$,

- (the) "Space ($\mathbb{F}^n$) is big. Really big. You just won't believe how vastly, hugely, mind-bogglingly big it is." Which way is x_*?



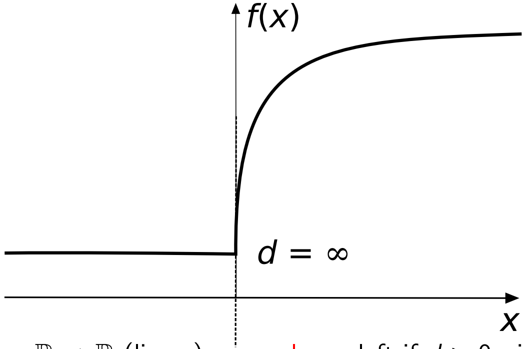
- $f(x) := dx + v : \mathbb{R} \to \mathbb{R}$ (linear) easy: always left if $d > 0$, right if $d < 0$

- (the) "Space ($\mathbb{F}^n$) is big. Really big. You just won't believe how vastly, hugely, mind-bogglingly big it is." Which way is x_*?



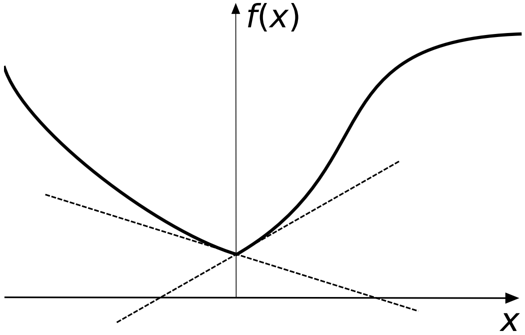
- $f(x) := dx + v : \mathbb{R} \to \mathbb{R}$ (linear) easy: always left if $d > 0$, right if $d < 0$
- Obvious idea: use the linear function that best locally approximates f
- Trusty old derivative $d = f'(x) = \lim_{t\to 0}[f(x+t) - f(x)]/t$ (putting a lot under the carpet even in $\mathbb{R}^n$, not to mention $\mathbb{F}^n$)

- (the) "Space ($\mathbb{F}^n$) is big. Really big. You just won't believe how vastly, hugely, mind-bogglingly big it is." Which way is x_*?

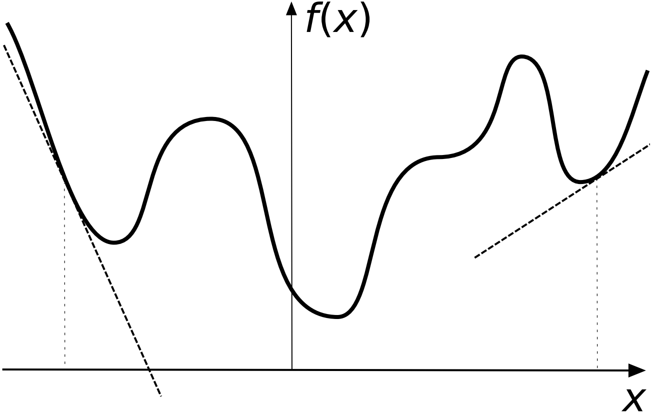


- $f(x) := dx + v : \mathbb{R} \to \mathbb{R}$ (linear) easy: always left if $d > 0$, right if $d < 0$
- Obvious idea: use the linear function that best locally approximates f
- Trusty old derivative $d = f'(x) = \lim_{t \to 0}[f(x + t) - f(x)] / t$ (putting a lot under the carpet even in $\mathbb{R}^n$, not to mention $\mathbb{F}^n$)
- Provided it exists

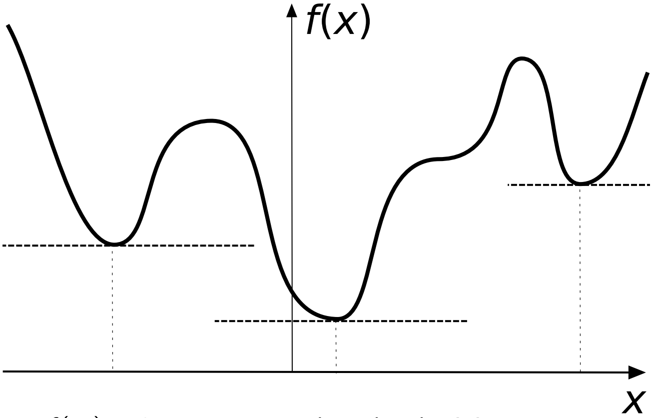
- (the) "Space ($\mathbb{F}^n$) is big. Really big. You just won't believe how vastly, hugely, mind-bogglingly big it is." Which way is x_*?



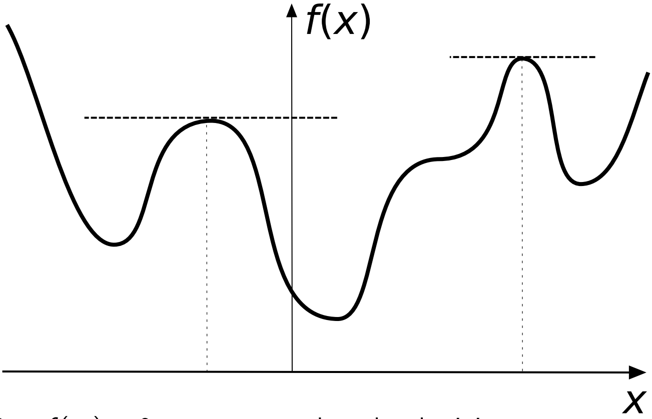
- $f(x) := dx + v : \mathbb{R} \to \mathbb{R}$ (linear) easy: always left if $d > 0$, right if $d < 0$
- Obvious idea: use the linear function that best locally approximates f
- Trusty old derivative $d = f'(x) = \lim_{t\to 0}[f(x+t) - f(x)]/t$ (putting a lot under the carpet even in $\mathbb{R}^n$, not to mention $\mathbb{F}^n$)
- Provided it exists . . . and it is unique



- $f'(x) < 0$ or $f(x) > 0 \Longrightarrow x$ cannot be a local minimum

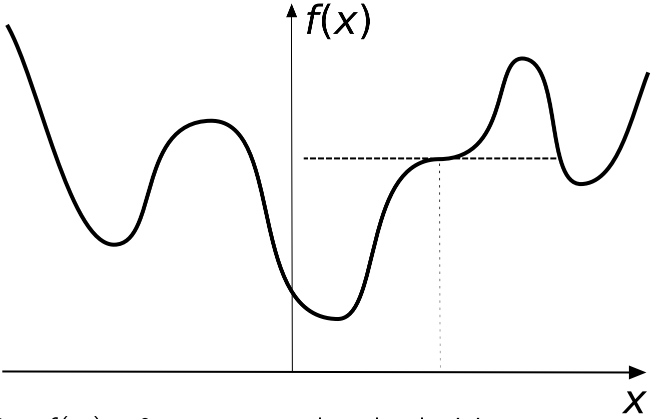


- $f'(x) < 0$ or $f(x) > 0 \Longrightarrow x$ cannot be a local minimum
- $f'(x) = 0$ in all local minima $\Longrightarrow$ in the global one



- $f'(x) < 0$ or $f(x) > 0 \Longrightarrow x$ cannot be a local minimum
- $f'(x) = 0$ in all local minima $\Longrightarrow$ in the global one
- However, $f'(x) = 0$ also in local (global) maxima

(Local) Optimality and Derivatives



- $f'(x) < 0$ or $f(x) > 0 \Longrightarrow x$ cannot be a local minimum
- $f'(x) = 0$ in all local minima $\Longrightarrow$ in the global one
- However, $f'(x) = 0$ also in local (global) maxima and in saddle points

- To find x_*, try finding stationary point x s.t. $f'(x) = 0$
- $f'(x) = 0$ (necessary, not sufficient) optimality condition: optimization is a system of (nonlinear) equations
- Still a lot hidden under the carpet:
  - what exactly is f' when $X \neq \mathbb{R}$?
  - lots of "ifs" and "buts" (f' has to exist, X has to be "nice", ...)
- If all goes well, (local) optimality can be detected using derivatives (we'll see more details later in a simpler setting) $\Longrightarrow$ optimality conditions for PDE-CO is a PDE system. But:

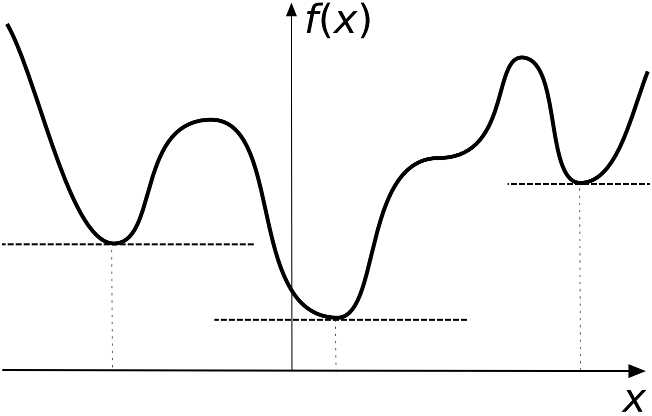
- To find x_*, try finding stationary point x s.t. $f'(x) = 0$
- $f'(x) = 0$ (necessary, not sufficient) optimality condition: optimization is a system of (nonlinear) equations
- Still a lot hidden under the carpet:
  - what exactly is f' when $X \neq \mathbb{R}$?
  - lots of "ifs" and "buts" (f' has to exist, X has to be "nice", ...)
- If all goes well, (local) optimality can be detected using derivatives (we'll see more details later in a simpler setting) $\Longrightarrow$ optimality conditions for PDE-CO is a PDE system. But:
  - significantly more complex than $\mathcal{H}$ itself

- To find x_*, try finding stationary point x s.t. $f'(x) = 0$
- $f'(x) = 0$ (necessary, not sufficient) optimality condition: optimization is a system of (nonlinear) equations
- Still a lot hidden under the carpet:
  - what exactly is f' when $X \neq \mathbb{R}$?
  - lots of “ifs” and “buts” (f' has to exist, X has to be “nice”, ...)
- If all goes well, (local) optimality can be detected using derivatives (we'll see more details later in a simpler setting) $\Longrightarrow$ optimality conditions for PDE-CO is a PDE system. But:
  - significantly more complex than $\mathcal{H}$ itself
  - mathematical details far from easy

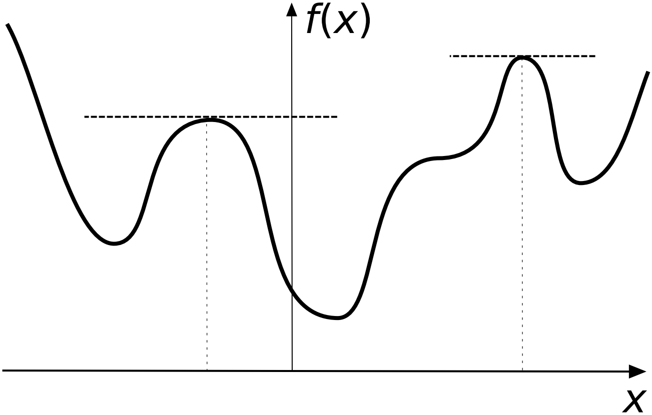
- To find x_*, try finding stationary point x s.t. $f'(x) = 0$
- $f'(x) = 0$ (necessary, not sufficient) optimality condition: optimization is a system of (nonlinear) equations
- Still a lot hidden under the carpet:
  - what exactly is f' when $X \neq \mathbb{R}$?
  - lots of "ifs" and "buts" (f' has to exist, X has to be "nice", ...)
- If all goes well, (local) optimality can be detected using derivatives (we'll see more details later in a simpler setting) $\Longrightarrow$ optimality conditions for PDE-CO is a PDE system. But:
  - significantly more complex than $\mathcal{H}$ itself
  - mathematical details far from easy
  - PDE systems have no closed-form solution anyway

- To find x_*, try finding stationary point x s.t. $f'(x) = 0$
- $f'(x) = 0$ (necessary, not sufficient) optimality condition: optimization is a system of (nonlinear) equations
- Still a lot hidden under the carpet:
  - what exactly is f' when $X \neq \mathbb{R}$?
  - lots of "ifs" and "buts" (f' has to exist, X has to be "nice", ...)
- If all goes well, (local) optimality can be detected using derivatives (we'll see more details later in a simpler setting) $\Longrightarrow$ optimality conditions for PDE-CO is a PDE system. But:
  - significantly more complex than $\mathcal{H}$ itself
  - mathematical details far from easy
  - PDE systems have no closed-form solution anyway $\Longrightarrow$ have to discretize the PDE and solve approximately

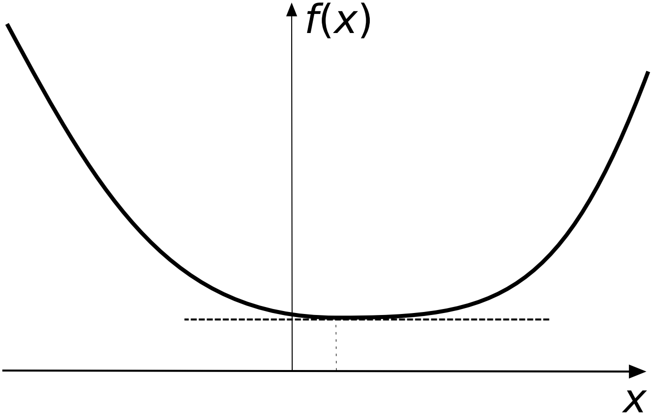
- Still back to "have to compute $f(\cdot)$ numerically"
- However, can now prove that x ($= [\,c\,,\,u\,]$) is a (local) minimum
- Also, algorithms that use derivatives are vastly more efficient
  (we'll see more details later in a simpler setting)
- Can quickly reach a (local) minimum and stop there:
  no more random moves for fear of having missed a better point nearby
- $|\,f'(\,x\,)\,| \approx$ "distance" from x_*, useful to choose accuracy of simulation
- Explicit optimality conditions leads to multiple strategies:
  - first discretize, then write optimality conditions
  - first write optimality conditions, then discretize
- However, $f'(\,x\,) = 0$ only tells x may be a local minimum



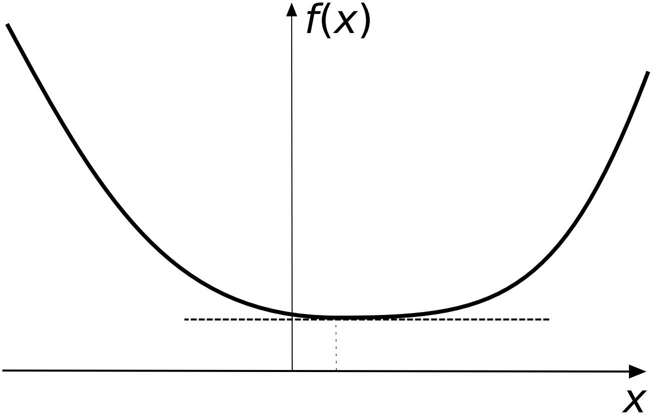
- If f has local minima



- If f has local minima and maxima, f' must first $\nearrow$ and then $\searrow$
- What if f' monotone?



- If f has local minima and maxima, f' must first $\nearrow$ and then $\searrow$
- What if f' monotone? f convex function $\Longrightarrow$



- If f has local minima and maxima, f' must first $\nearrow$ and then $\searrow$
- What if f' monotone? f convex function $\Longrightarrow$ stationary point $\Longrightarrow$ local minimum $\Longrightarrow$ global minimum
- $f(\cdot)$ and X convex $\Longrightarrow$ x_* can be found "easily"
- Rules to construct $f(\cdot)$ and X convex

- In a nutshell: if everything goes very well
  - f, $\mathcal{H}$ and $\mathcal{G}$ must have the right properties
  - details have to be worked out, options be wisely chosen
  - local optimality must be OK (or the problem convex to start with)
- There is no general-purpose PDE-OC solver,
  each case has to be dealt with individually
- However, tools are there, knowledge is there
- Problems of scale required by practical applications can be solved
  - with a little help from my (PDE-CO-savvy) friends
  - and possibly a supercomputer at hand
- As always, structure is your friend
  (e.g., optimal control has many specialized approaches exploiting time)
- Is it worth? In quite many cases, it is

- "Easier" problem: no PDE constraints, "only" algebraic ones

  $$X = \{ x \in \mathbb{R}^n \,:\, g_i(x) \leq 0 \;\; i \in \mathcal{I} \,, \quad h_j(x) = 0 \;\; j \in \mathcal{J} \}$$

  $\mathcal{I}$ = set of inequality constraints, $\mathcal{J}$ = set of equality constraints

- $G(x) = [\, g_i(x) \,]_{i \in \mathcal{I}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{I}|}$, $H(x) = [\, h_i(x) \,]_{i \in \mathcal{J}} : \mathbb{R}^n \to \mathbb{R}^{|\mathcal{J}|}$

  $$X = \{ x \in \mathbb{R}^n \,:\, G(x) \leq 0 \,,\, H(x) = 0 \}$$

  $G(\cdot)$, $H(\cdot)$ algebraic (vector-valued, multivariate) real functions

- Could always assume $|\, \mathcal{I} \,| = 1$ and $|\, \mathcal{J} \,| = 0$:
  - $h_j(x) = 0 \;\equiv\; h_j(x) \leq 0 \,\wedge\, -h_j(x) \leq 0$
  - $G(x) \leq 0 \;\equiv\; \max\{\, g_i(x) \,:\, i \in \mathcal{I} \,\} = g(x) \leq 0$

  but good reasons not to (exploit structure when is there)

- What does this gives to me? You know the drill: derivatives

- $f : \mathbb{R}^n \rightarrow \mathbb{R}$, partial derivative of f w.r.t. x_i at $x \in \mathbb{R}^n$:

  $$\frac{\partial f}{\partial x_i}(\, x\,) = \lim_{t \rightarrow 0} \frac{f(\, x_1\,, \ldots, x_{i-1}\,, x_i + t\,, x_{i+1}\,, \ldots, x_n\,) - f(\, x\,)}{t}$$

  just $f'(\, x_1\,, \ldots\,, x_{i-1}\,, x\,, x_{i+1}\,, \ldots\,, x_n\,)$ treating x_j for $j \neq i$ as constants
- Good news: computing derivatives mechanic, Automatic Differentiation software will do it for you, not only from formulæ but from code

- $f : \mathbb{R}^n \to \mathbb{R}$, partial derivative of f w.r.t. x_i at $x \in \mathbb{R}^n$:

$$\frac{\partial f}{\partial x_i}(x) = \lim_{t \to 0} \frac{f(x_1, \ldots, x_{i-1}, x_i + t, x_{i+1}, \ldots, x_n) - f(x)}{t}$$

  just $f'(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_n)$ treating x_j for $j \neq i$ as constants

- Good news: computing derivatives mechanic, Automatic Differentiation software will do it for you, not only from formulæ but from code ... provided they exist ($f(x) = |x|$, $f'(x) = ???$)

- Gradient = vector of all partial derivatives all important in optimization

$$\nabla f(x) := \left[\frac{\partial f}{\partial x_1}(x), \ldots, \frac{\partial f}{\partial x_n}(x) \right]$$

- f differentiable at $x \approx \forall i \; \frac{\partial f}{\partial x_i}(\cdot)$ continuous ($\Longleftarrow \exists$)

- $f \in C^1$: $\nabla f(x)$ continuous $\equiv f$ differentiable ($\Longrightarrow f$ continuous) $\forall x$

- $f \in C^1 \Longrightarrow$ finding stationary point) "easy":
  just go in the other direction ($-\nabla f(x) =$ steepest descent direction)

- $f : \mathbb{R}^n \to \mathbb{R}^m$, $f(x) = [\, f_1(x), f_2(x), \ldots, f_m(x) \,]$
- Partial derivative: usual stuff, except with extra index

$$\frac{\partial f_j}{\partial x_i}(x) = \lim_{t \to 0} \frac{f_j(x_1, \ldots, x_{i-1}, x_i + t, x_{i+1}, \ldots, x_n) - f_j(x)}{t}$$

- $\nabla f(x) : \mathbb{R}^n \to \mathbb{R}^n$ itself has a gradient: Hessian of f

$$\nabla^2 f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \ldots & \frac{\partial^2 f}{\partial x_n \partial x_1}(x) \\ \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \frac{\partial^2 f}{\partial x_2^2}(\,, x) & \ldots & \frac{\partial^2 f}{\partial x_n \partial x_2}(x) \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_n}(\,, x) & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) & \ldots & \frac{\partial^2 f}{\partial x_n^2}(x\,,) \end{bmatrix}$$

second-order partial derivative (just do it twice) $\qquad \dfrac{\partial^2 f}{\partial x_j \partial x_i} \qquad \dfrac{\partial^2 f}{\partial x_i \partial x_i} = \dfrac{\partial^2 f}{\partial x_i^2}$

- $f \in C^2$: $\nabla^2 f(x)$ continuous ($\Longrightarrow$ symmetric) $\equiv \nabla f$ differentiable $\forall\, x$
- $f \in C^2 \Longrightarrow$ finding local minimum (stationary point) “super-easy”

- $X = \mathbb{R}^n \Longrightarrow$ all depends on quality of derivatives of f:
  - $f \notin C^1 \Longrightarrow$ subgradient methods $\Longrightarrow$ sublinear convergence (error at step $k \approx 1/\sqrt{k}$, $O(\,1\,/\,\varepsilon^2\,)$ iterations)
  - $f \in C^1 \Longrightarrow$ gradient methods $\Longrightarrow$ linear convergence (error at step $k \approx \gamma^k$ with $\gamma < 1$, $O(\,1\,/\,\log(\,\varepsilon\,)\,)$ iterations)
  - $f \in C^2 \Longrightarrow$ Newton-type methods $\Longrightarrow$ superlinear/quadratic convergence (error at step $k \approx \gamma^{k^2}/\gamma^{2^k}$ with $\gamma < 1$, $\approx O(\,1\,)$ iterations)
- All bounds $\approx$ independent from n (can be "hidden in the constants") $\Longrightarrow$ good for large-scale problems (n very large)
- Not all trivial, line search/trust region, globalization, ...
- Gradient methods can be rather slow in practice ($\gamma \approx 1$), need to cure zig-zagging (heavy ball, fast gradients, ...)
- Hessian a big guy, inverting it $O(n^3)$ a serious issue for large-scale: quasi-Newton/conjugate gradient only $O(n^2)$ / $O(kn)$ (but trade-offs)
- All in all, local (unconstrained) convergence very well dealt with

- Local optimality still "easy" to characterize via derivatives
- Karush-Kuhn-Tucker conditions: $\exists\, \lambda \in \mathbb{R}_+^{|\mathcal{I}|}$ and $\mu \in \mathbb{R}^{|\mathcal{J}|}$ s.t.

$$g_i(\,x\,) \leq 0 \;\; i \in \mathcal{I} \quad , \quad h_j(\,x\,) = 0 \;\; j \in \mathcal{J} \tag{KKT-F}$$

$$\nabla f(x) + \textstyle\sum_{i \in \mathcal{I}} \lambda_i \nabla g_i(\,x\,) + \sum_{j \in \mathcal{J}} \mu_j \nabla h_j(\,x\,) = 0 \tag{KKT-G}$$

$$\textstyle\sum_{i \in \mathcal{I}} \lambda_i g_i(\,x\,) = 0 \tag{KKT-CS}$$

$\equiv x$ stationary point of Lagrangian function (in x, λ / μ parameters)

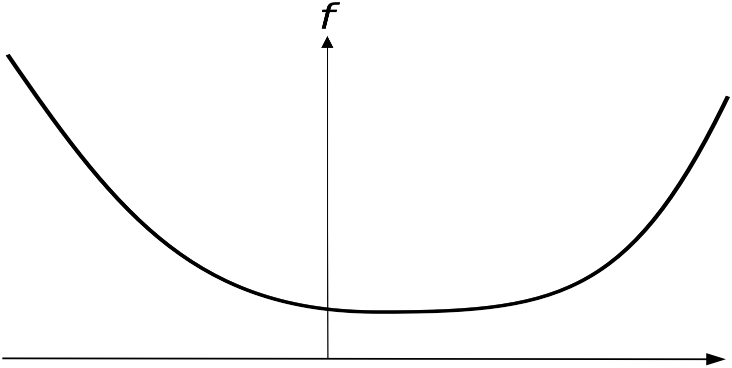
$L(\,x\,;\,\lambda\,,\,\mu\,) = f(\,x\,) + \sum_{i \in \mathcal{I}} \lambda_i g_i(\,x\,) + \sum_{j \in \mathcal{I}} \mu_j h_j(\,x\,)$ ($\rightsquigarrow$ duality ...)

- KKT Theorem: x local optimum + constraint qualifications $\Longrightarrow$ (KKT)
- (P) convex problem: (KKT) $\Longrightarrow x$ global optimum
- Otherwise, quite involved second-order optimality conditions ...

Meaning What, Algorithmically?

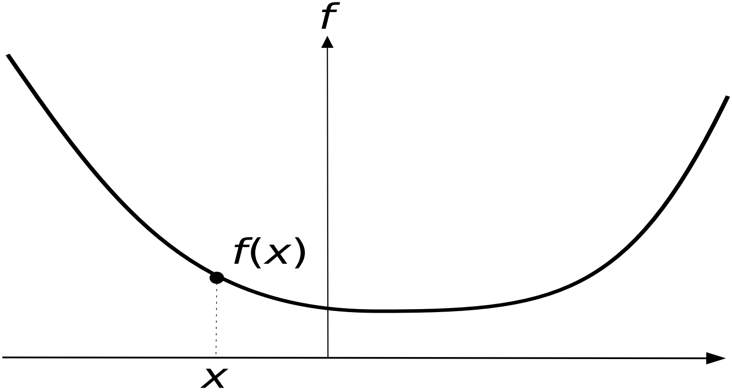
- In a nutshell, that $\exists$ efficient local algorithms
- At the very least, (KKT) $\approx x$ local minimum, stop the search
- Checking if (KKT) holds "easy" (Farkas' Lemma ...)
- Optimization $\equiv$ solving systems of nonlinear equations and inequalities
- Does not mean that algorithms are obvious:
  - several different forms (primal, dual, ...)
  - several different ideas (active set, projection, barrier, penalty, ...)
  - combinatorial aspects (active set choice) may make them inefficient
- Yet, provably and practically efficient algorithms are there if data of the problem nice (f, $G \in C^1/C^2$, H affine ...)
- Particularly relevant/elegant class: primal-dual interior point methods
- (Reasonably) robust and efficient implementations available, although numerical issues (linear algebra accuracy/cost) still nontrivial

- Unfortunately an entirely different game: sifting through all X required
- Derivatives a local object, can't give global information
  except in the convex case, where they actually do



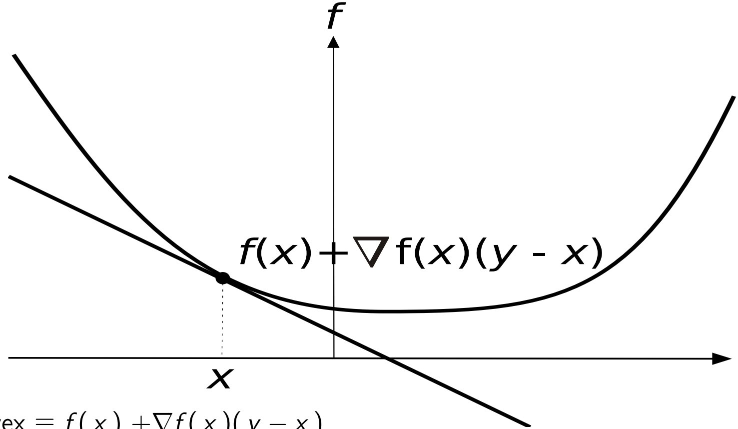
- f convex $\equiv$

- Unfortunately an entirely different game: sifting through all X required
- Derivatives a local object, can't give global information
  except in the convex case, where they actually do



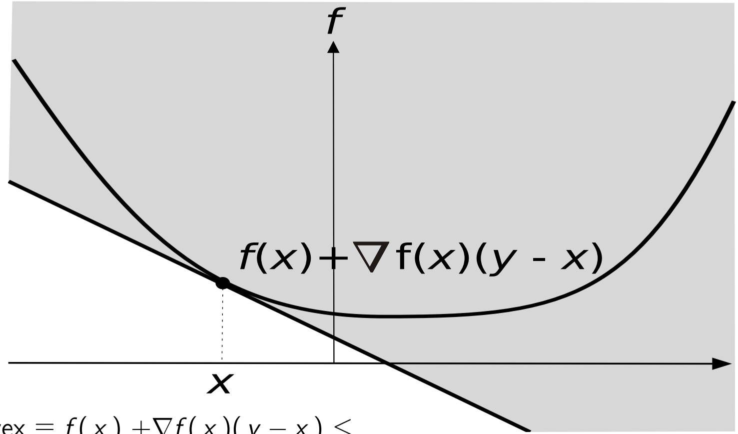
- f convex $\equiv f(\,x\,)$

- Unfortunately an entirely different game: sifting through all X required
- Derivatives a local object, can't give global information
  except in the convex case, where they actually do



- f convex $\equiv f(x) + \nabla f(x)(y - x)$

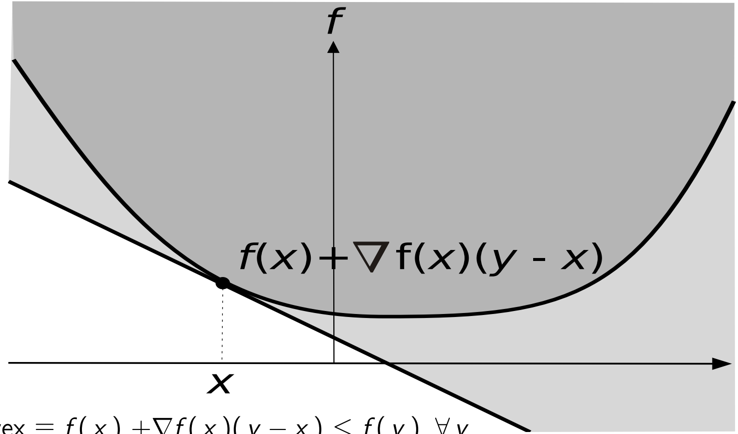
- Unfortunately an entirely different game: sifting through all X required
- Derivatives a local object, can't give global information
  except in the convex case, where they actually do



- f convex $\equiv f(x) + \nabla f(x)(y - x) \leq$

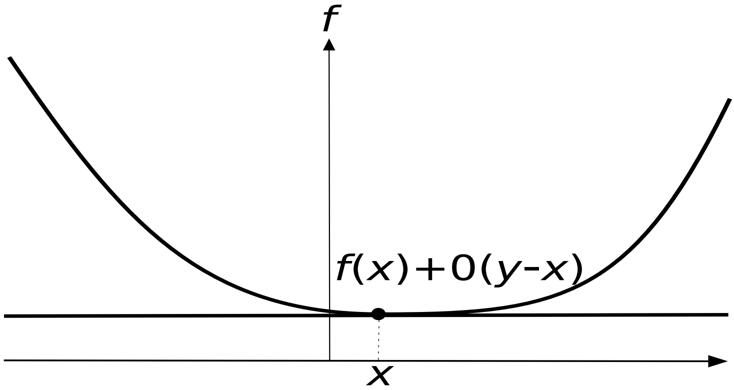
- Unfortunately an entirely different game: sifting through all X required
- Derivatives a local object, can't give global information

  except in the convex case, where they actually do



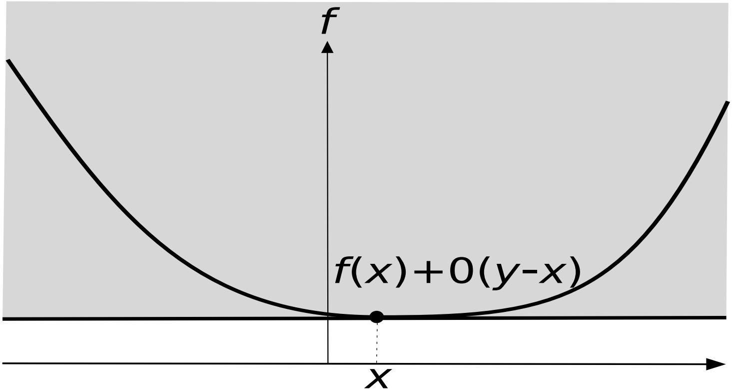
- f convex $\equiv f(x) + \nabla f(x)(y - x) \leq f(y) \;\; \forall\, y$

- Unfortunately an entirely different game: sifting through all X required
- Derivatives a local object, can't give global information
  except in the convex case, where they actually do



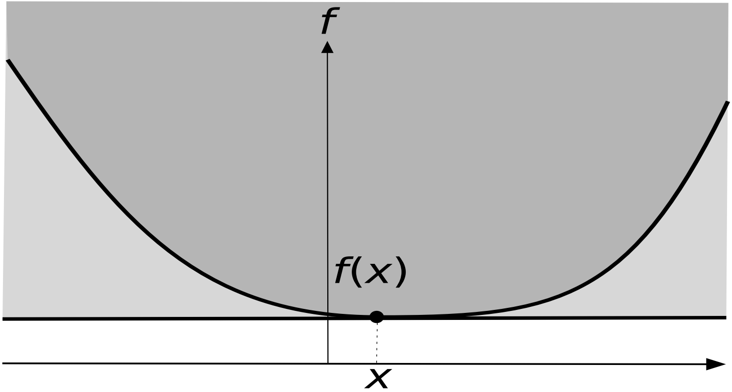
- f convex $\equiv f(x) + \nabla f(x)(y - x) \leq f(y) \;\; \forall y$
- $\nabla f(x) = 0$

- Unfortunately an entirely different game: sifting through all X required
- Derivatives a local object, can't give global information
  except in the convex case, where they actually do



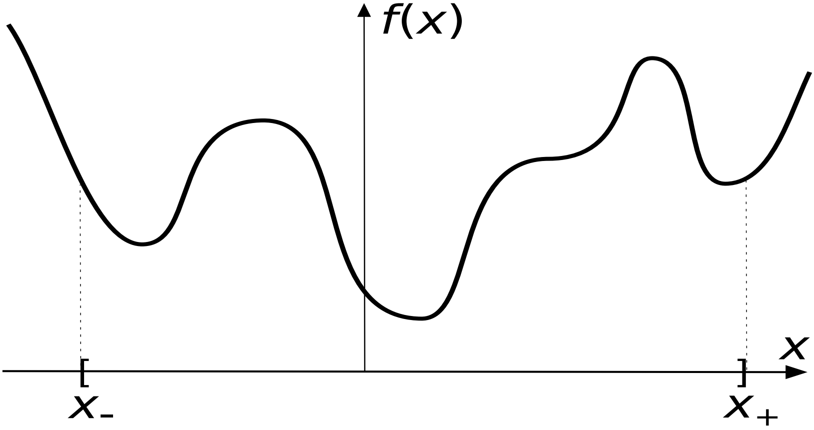
- f convex $\equiv f(x) + \nabla f(x)(y - x) \leq f(y) \;\; \forall\, y$
- $\nabla f(x) = 0 \Longrightarrow f(x)$

- Unfortunately an entirely different game: sifting through all X required
- Derivatives a local object, can't give global information
  except in the convex case, where they actually do

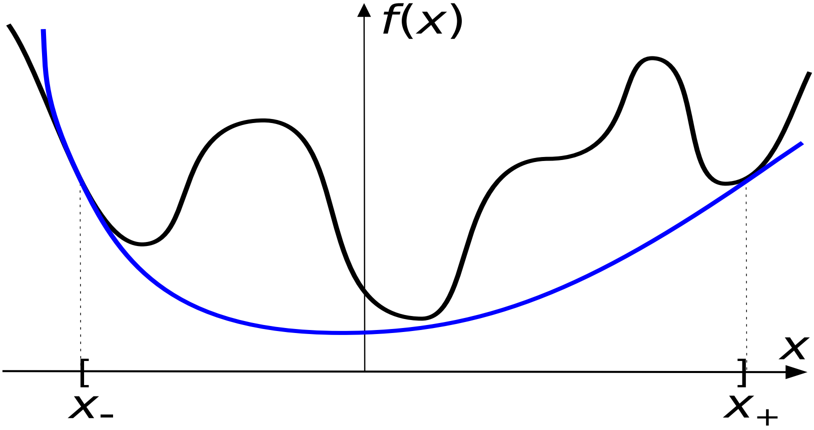


- f convex $\equiv f(x) + \nabla f(x)(y - x) \leq f(y) \;\; \forall\, y$
- $\nabla f(x) = 0 \implies f(x) \; [+0(y - x)] \leq f(y) \;\; \forall\, y$

- Sift through all $X = [\, x_- \,,\, x_+ \,]$, but using a clever guide

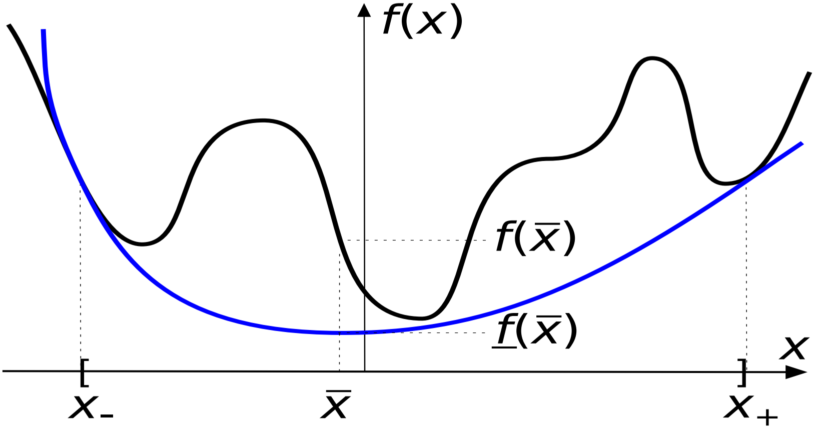


- Sift through all $X = [\, x_- \, , \, x_+ \,]$, but using a clever guide



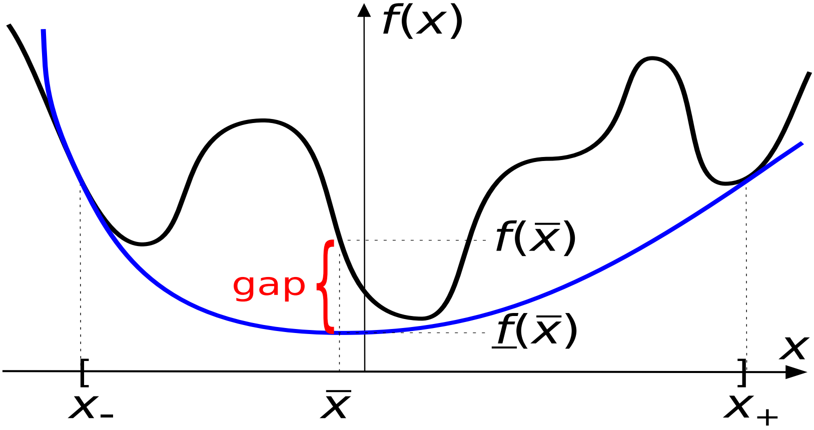
- Convex lower approximation $\underline{f}$ of nonconvex f on X

- Sift through all $X = [\, x_- \,,\, x_+ \,]$, but using a clever guide



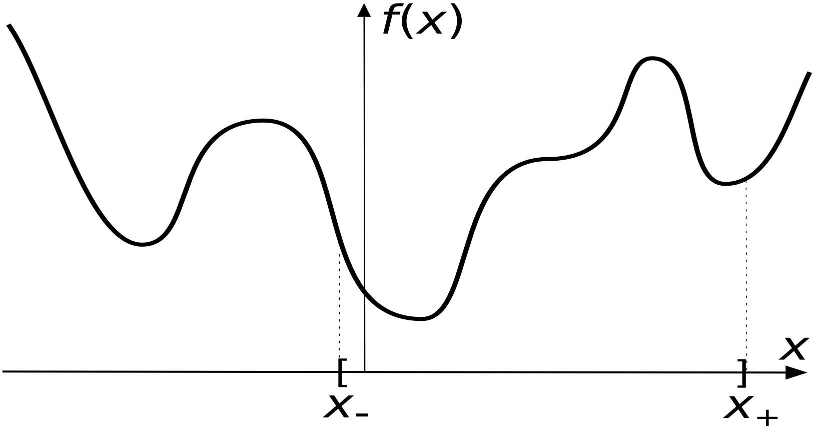
- Convex lower approximation $\underline{f}$ of nonconvex f on X
- "Easily" find local $\equiv$ global minimum $\bar{x}$, giving $\underline{f}(\, \bar{x}\,) \leq f_* \leq f(\, \bar{x}\,)$

- Sift through all $X = [\, x_- \, , \, x_+ \,]$, but using a clever guide



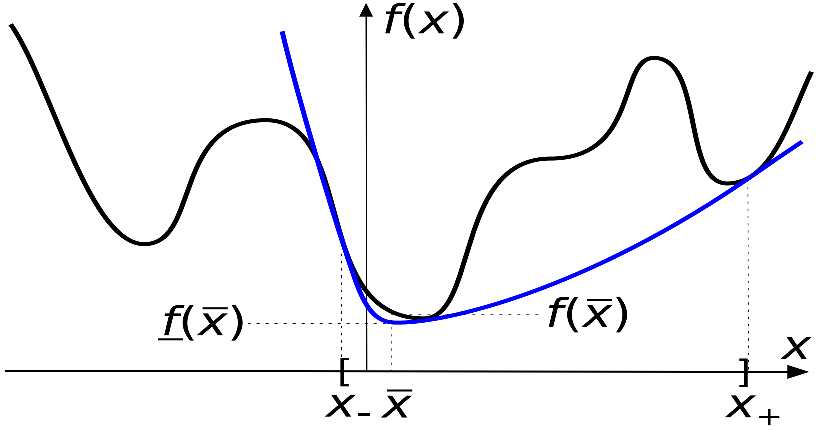
- Convex lower approximation $\underline{f}$ of nonconvex f on X
- "Easily" find local $\equiv$ global minimum $\bar{x}$, giving $\underline{f}(\,\bar{x}\,) \le f_* \le f(\,\bar{x}\,)$
- If gap $f(\,\bar{x}\,) - \underline{f}(\,\bar{x}\,)$ too large,

- Sift through all $X = [\, x_- \,,\, x_+ \,]$, but using a clever guide



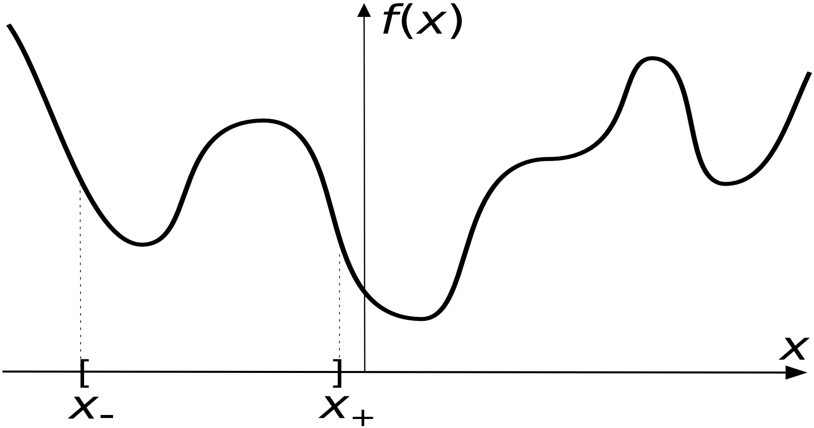
- Convex lower approximation $\underline{f}$ of nonconvex f on X
- "Easily" find local $\equiv$ global minimum $\bar{x}$, giving $\underline{f}(\,\bar{x}\,) \leq f_* \leq f(\,\bar{x}\,)$
- If gap $f(\,\bar{x}\,) - \underline{f}(\,\bar{x}\,)$ too large, partition X and iterate

- Sift through all $X = [\, x_- \, , \, x_+ \,]$, but using a clever guide



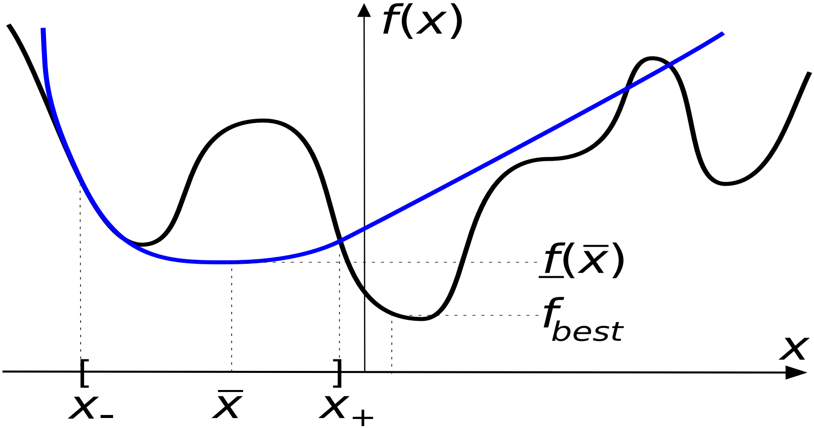
- Convex lower approximation $\underline{f}$ of nonconvex f on X
- "Easily" find local $\equiv$ global minimum $\bar{x}$, giving $\underline{f}(\,\bar{x}\,) \leq f_* \leq f(\,\bar{x}\,)$
- If gap $f(\,\bar{x}\,) - \underline{f}(\,\bar{x}\,)$ too large, partition X and iterate
- $\underline{f}$ depends on partition, smaller partition (hopefully) $\Longrightarrow$ better gap

- Sift through all $X = [\, x_- \,,\, x_+ \,]$, but using a clever guide



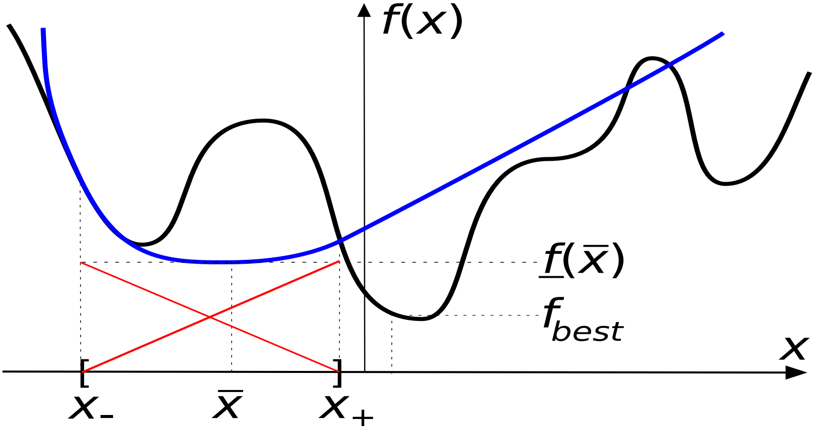
- Convex lower approximation $\underline{f}$ of nonconvex f on X
- "Easily" find local $\equiv$ global minimum $\bar{x}$, giving $\underline{f}(\,\bar{x}\,) \le f_* \le f(\,\bar{x}\,)$
- If gap $f(\,\bar{x}\,) - \underline{f}(\,\bar{x}\,)$ too large, partition X and iterate
- If on some partition

- Sift through all $X = [\, x_- \, , \, x_+ \,]$, but using a clever guide



- Convex lower approximation $\underline{f}$ of nonconvex f on X
- "Easily" find local $\equiv$ global minimum $\bar{x}$, giving $\underline{f}(\,\bar{x}\,) \leq f_* \leq f(\,\bar{x}\,)$
- If gap $f(\,\bar{x}\,) - \underline{f}(\,\bar{x}\,)$ too large, partition X and iterate
- If on some partition $\underline{f}(\,\bar{x}\,) \geq$ best f-value so far,

- Sift through all $X = [\, x_- \, , \, x_+ \,]$, but using a clever guide



- Convex lower approximation $\underline{f}$ of nonconvex f on X
- "Easily" find local $\equiv$ global minimum $\bar{x}$, giving $\underline{f}(\,\bar{x}\,) \leq f_* \leq f(\,\bar{x}\,)$
- If gap $f(\,\bar{x}\,) - \underline{f}(\,\bar{x}\,)$ too large, partition X and iterate
- If on some partition $\underline{f}(\,\bar{x}\,) \geq$ best f-value so far, partition killed for good

Is Something Like This Efficient?

- In a word? No
- Worst-case: keep dicing and slicing X until pieces "very small" ($\approx (\varepsilon / L)^n$)
- However, in practice it depends on:
  - "how much nonconvex" f really is
  - how good $\underline{f}$ is as a lower approximation of f
- Best possible lower approximation: convex envelope
- Bad news: computing convex envelopes is hard
- Typical approach:
  - rewrite the expression of f in terms of unary/binary functions
  - apply specific convexification formulæ for each function
- Tedious job, bounds often rather weak
- Good news: implemented in available, well-engineered solvers
- Good news: immensely less inefficient in practice than blind search (at least, bounds allow to cut away whole regions for good)

- In a nutshell: if everything goes quite well
  - f, G and H must have the right properties
  - the less nonconvex they are, the better
  - the less "complicated" they are, the better
- Yet, there are general-purpose nonconvex MINLP solvers which can solve the problem to proven optimality
- Using them nontrivial, formulating the problem well crucial ($\equiv$ so that a "good $\underline{f}$ is available")
- Not really "large-scale", but 100s/1000s of variables often doable quickly enough with off-the-shelf tools
- Much larger problems also possible with special tools/effort
- As always, structure is your friend
- Is it worth? In quite many cases, it is (ask chemical Engineers)

- Computing "good" convex approximation both complex and difficult
- One very relevant case in which at least "easy": integrality constraints
- $x_i \in \mathbb{Z}$, most often $x_i \in \{\,0\,,\,1\,\}$ very convenient for discrete choices (start machine/don't, make trip/don't, . . .)
- Clearly nonconvex, $\exists$ nonlinear versions ($x_i(\,1 - x_i\,) \leq 0$, . . .)
- Actually quite powerful: many different nonlinear nonconvex structures can be expressed via that + "simple" (linear) constraints
- Yet, this requires some rather weird formulation tricks $z = xy \equiv [\,(z \leq x) \wedge (z \leq y) \wedge (z \geq x + y - 1)\,]$ if all $x, y, z \in \{\,0\,,\,1\,\}$
- If all the rest in the problem convex, then a convex relaxation very easy: continuous relaxation ($x_i \in \mathbb{Z} \longrightarrow x_i \in \mathbb{R}$)
- This does not mean convex relaxation is good, but it may be
- At least makes life a lot easier to solution algorithms

- Finding good relaxations crucial for practical efficiency
- Solvers helped a lot by having few, well-controlled nonconvexities
- Mixed-Integer Convex Problems the easiest class of hard problems
- Especially famous special case: Mixed-Integer Linear Program

  $$(MILP) \quad \min \{ \, cx \, : \, Ax \geq b \; , \; x_i \in \mathbb{Z} \; \; i \in I \, \}$$

  $\Longrightarrow$ continuous relaxation $\equiv$ Linear Program
- Very stable and efficient algorithms, some $\approx$ unique (simplex methods)
- Very powerful methods to improve relaxation quality (valid inequalities)
- Countless many results about special combinatorial structures (paths, trees, cuts, matchings, cliques, covers, knapsacks, ...)
- Clever approaches to exploit structure, though some work for MINLP too (Column/Row Generation, Dantzig-Wolfe/Benders' Decomposition, ...)

- Fundamental point: formulating the problem well is crucial
- Almost anything can be written as a MILP, albeit to some $\approx$ (not always a good idea: some nonlinearities "nice")
- Many different ways to write the same problem: apparently minor changes can make orders-of-magnitude difference
- Several of the best formulation weird and/or very large (appropriate tricks to only generate the strictly required part)
- Doing it "by hand" should not be required: solvers should be able to automatically find the best formulation (reformulate)
- Good news: the "perfect" formulation provably exists
- Bad news: it is provably ($\mathcal{NP}$-)hard to construct
- Doing it automatically is clearly difficult (but we should try harder)
- Meanwhile, a well-trained eye can make a lot of difference

- Good news: can "hide" many nonlinearities in a Linear Program
- Conic Program: $(P)\ \min\{\, cx \,:\, Ax \geq_K b \,\}$
  where $x \geq_K y \equiv x - y \in K$, K pointed convex cone, e.g.
  - $K = \mathbb{R}^n_+ \equiv$ sign constraints $\equiv$ Linear Program
  - $K = \mathbb{L} = \{\, x \in \mathbb{R}^n \,:\, x_n \geq \sqrt{\sum_{i=1}^{n-1} x_i^2} \,\} \equiv$ Second-Order Cone Program
  - $K = \mathbb{S}_+ = \{\, A \succeq 0 \,\} \equiv$ "$\succeq$" constraints $\equiv$ SemiDefinite Program
- Exceedingly smart idea: everything is linear, but the cone is not
  $\equiv$ a nonlinear program disguised as a linear one
- Contains as special case convex quadratic functions
- Many interesting (convex) nonlinear functions have a conic representation (but have to learn some even weirder formulation trick)
- Continuous relaxation almost as efficient as Linear Program
- Many combinatorial MILP tricks extend di MI-SOCP (valid surfaces, . . .)
- Support in general-purpose software growing, already quite advanced

- In a nutshell: unless something goes very bad
  - data of the problem by definition is nice
  - a feasible relaxation always there, bounds can be quite good
  - lots of good ideas (cutting planes, general-purpose heuristics, ...)
- Plenty of general-purpose, well-engineered MILP/MI-SOCP solvers which can solve the problem to proven optimality
- Lots of useful supporting software: algebraic modelling languages, (there for MINLP too), IDEs, interfaces with database/spreadsheet, ...
- 10000/100000 variables often doable in minutes/hours on stock hw/sw if you write the right model
- Much larger problems (10^6 / 10^9) also possible with special tools/effort
- As always, structure is your friend, and many known forms of structures
- Is it worth? In very many cases, it is

Conclusions

- Optimization problems are difficult

- Optimization problems are difficult . . . but there are $\neq$ kinds of “difficult”
- Many problems have structure that can be exploited
- First crucial choice: which class of optimization problems
- Trade-off model accuracy vs. model complexity not trivial
- However, apparently very complex problems may not be that difficult if one knows the right set of modelling tricks
- Lots of stable, well-developed software (even open-source), especially for the most “tractable” problems
- A lot depends on how the problem is written
- The hand who rocks the model is the hand who rules the world