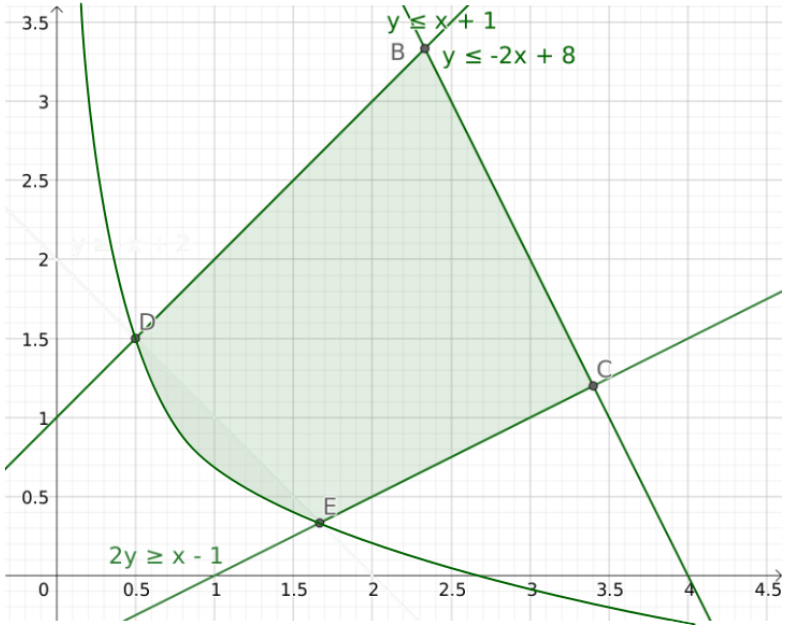
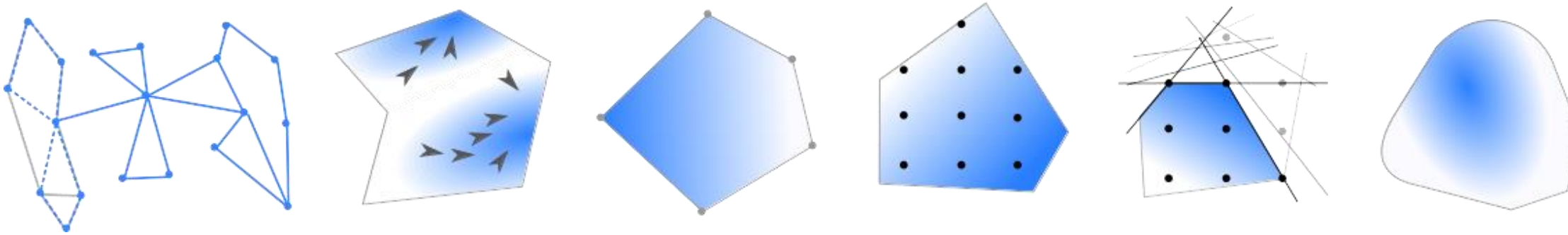
# Convex Optimisation 1

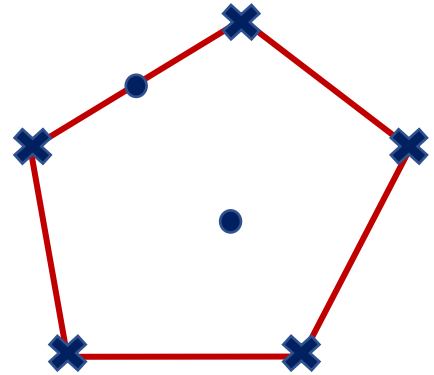
## COMP4691 / 8691



# Convex Optimisation

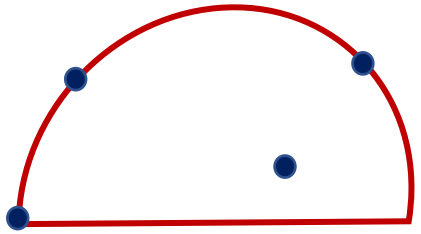
The objective is a *convex function* (for min) over a convex feasible set.



Linear programming fits these requirements.

In general **compared to LP**:

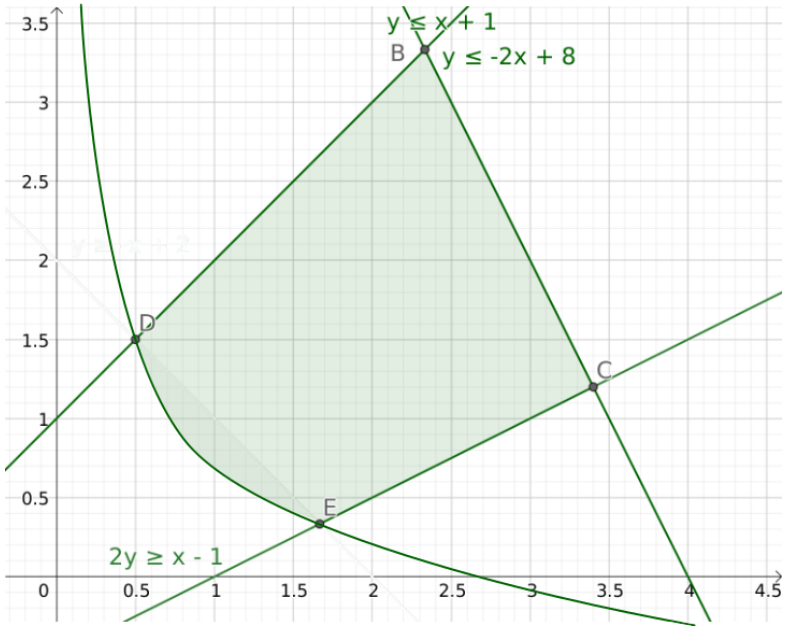
- Feasible region need not be a polyhedron
- Optimal solution can be anywhere on the boundary of the feasible region (and need not be at an extreme point)
- Optimal solution can be in the interior of the feasible region



Simplex algorithm won't work.

# Convex Optimisation Outline

- Convexity
  - Convex Functions
  - Convergence Rates
  - Background: Vector Calculus
- Unconstrained Optimisation
- Constrained Optimisation
- Lagrangian Duality
- KKT Conditions
- Interior Point Method




In some cases we will look at more general non-convex problems, as some of the theory applies there also.

# Disclaimer and Resources

We have limited time. I'm going to cover some of the **key concepts** and **algorithms**, and will avoid things like proofs on rates of convergence.

You will have enough to understand how the key algorithms work, how to form convex optimisation problems and how to use some of the tools.

| Online resources: |
| --- |
| ❖ **Boyd and Vandenberghe 2014** Convex Optimization<br>*What the Advanced Topics in Machine Learning course COMP4680 closely follows when the topic is convex optimisation.*<br>❖ **Bubeck 2015** Convex Optimization: Algorithms and Complexity<br>*Overview of convergence rates for different classes of convex problems.* |

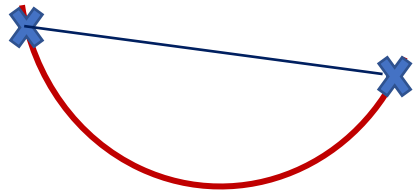
# Convex Functions

We've introduced convex sets. We can also talk about the convexity of functions, which is useful for determining whether our objective and constraints represent a convex optimisation problem.

A real-valued **function is convex** if and only if the line segment between any two points on its surface is **on or above** its *graph*:

$f : D \to \mathbb{R}$ $D$ A convex set in some real vector space

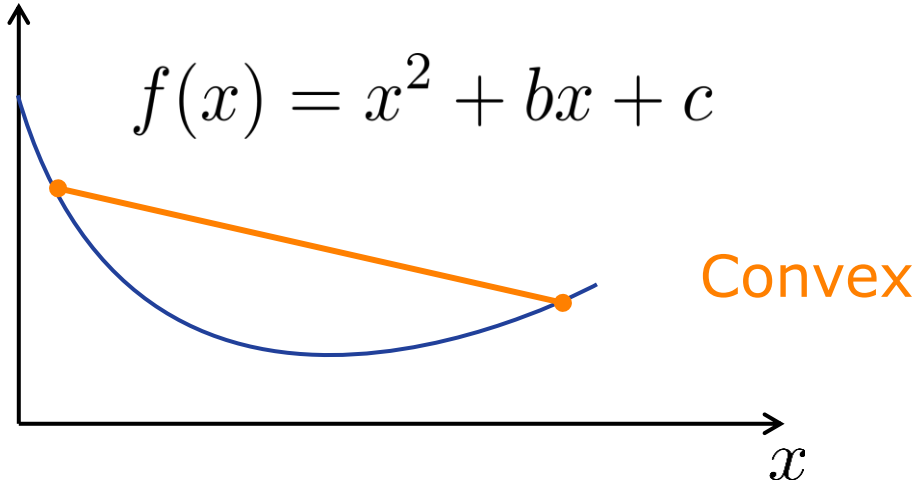
$$\forall x, y \in D : \quad f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y) \quad \forall \alpha \in [0,1]$$



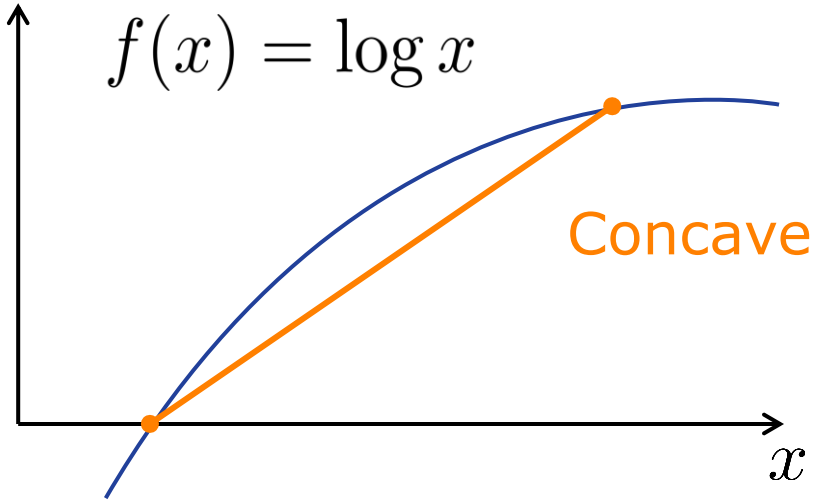
strict convexity if a strict inequality for **open** line segment

Conversely: a real-valued **function is concave** if and only if the line segment between any two points on its surface is **on or below** its *graph*.

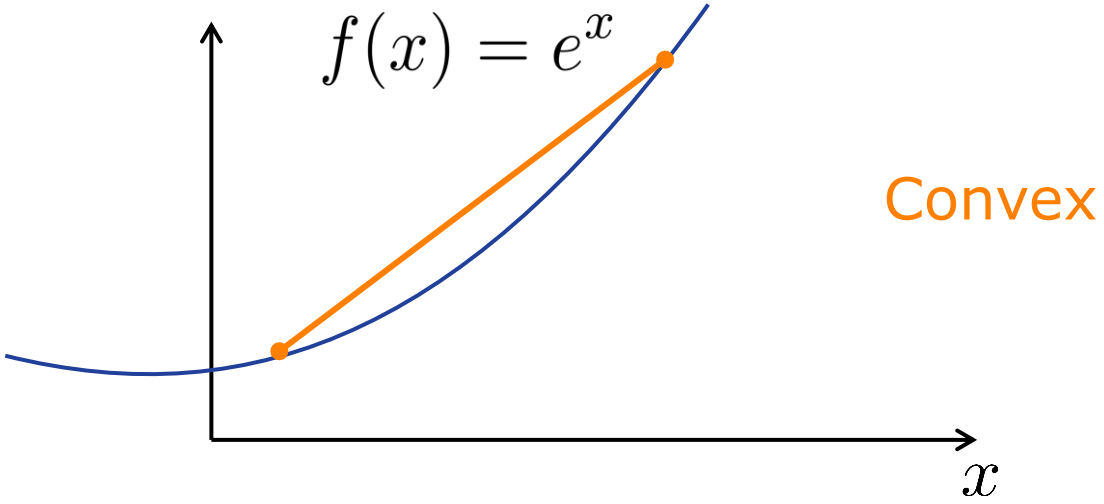
# Examples



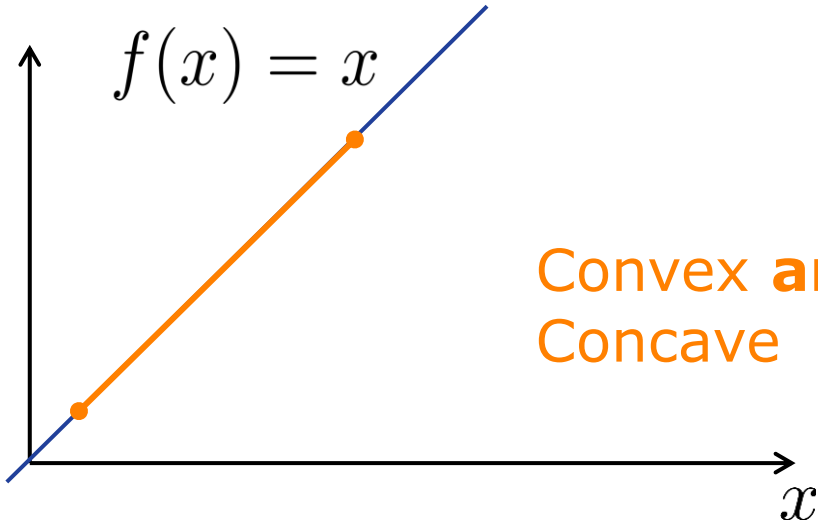

Convex




Concave




Convex

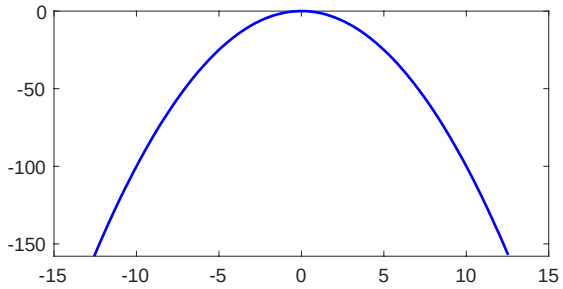



Convex **and** Concave

# Examples

Is the function $f(x) = -x^2$ convex or concave?

Concave




Is a linear (or affine) function convex or concave?

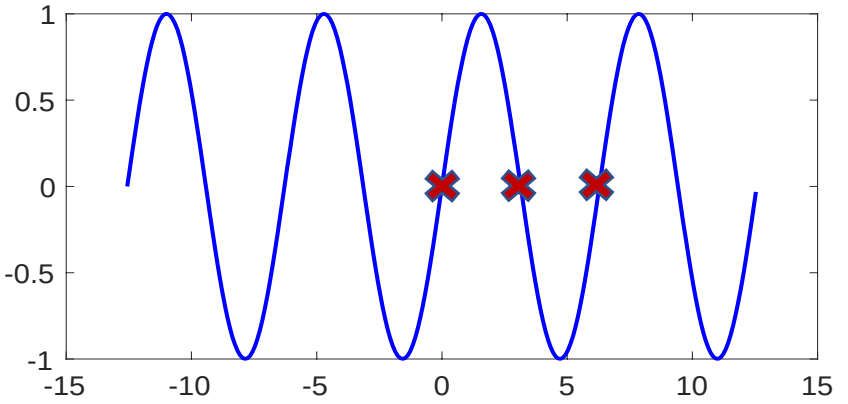
Both!

Affine: $f(x) = Ax + b$

Linear: $f(x) = Ax$

Is sine convex or concave?

Neither!

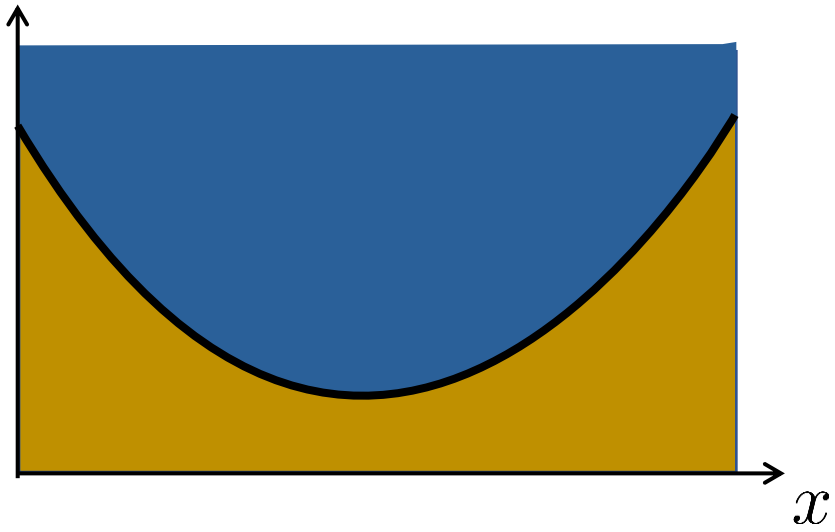


# Convex Functions

A **convex** function's *epigraph* is a convex set.

A **concave** function's *hypograph* is a convex set.

**Epigraph** (**hypograph**) = set of points on or **above** (**below**) the *graph* of a function.



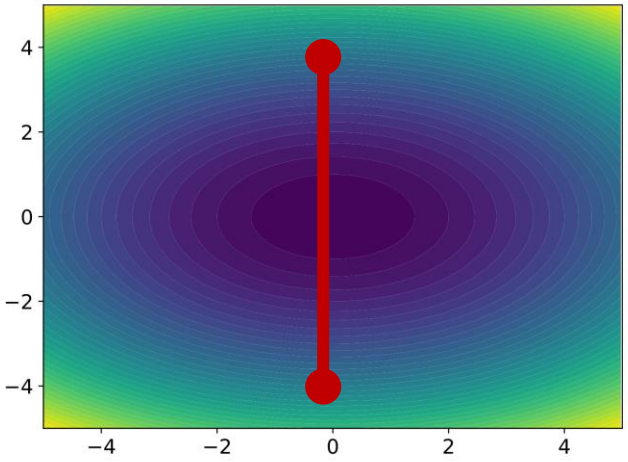

Note that might only be interested in the convexity of a function over an interval or convex set (single or multi-variate) rather than the full space.

# Multi-variate and Vector-valued

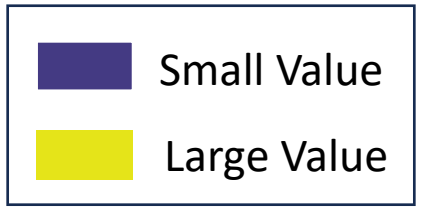
Same definition applies for multi-variate functions (multiple arguments / vector inputs).

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}$$

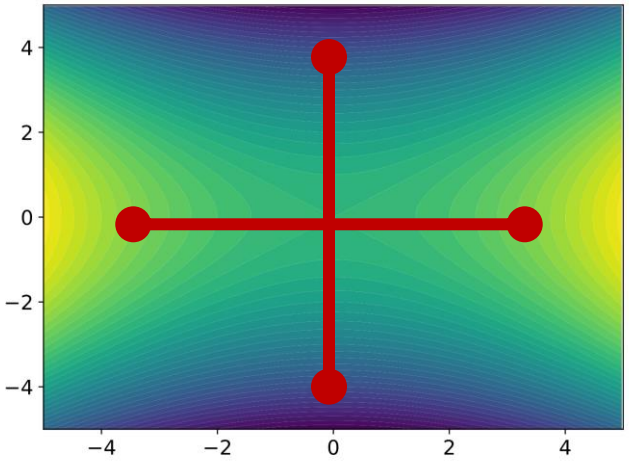
$$f(x, y) := x^2 + 2y^2$$




convex




$$f(x, y) := x^2 - 2y^2$$




nonconvex

A convex vector-value function (vector output): each output is convex.

$$f : \mathbb{R} \longrightarrow \mathbb{R}^m$$

$$f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_m(x) \end{bmatrix}$$

all convex

# Combining Functions

Recall, intersection of convex sets is itself a convex set.
Another important property:

The **sum of convex functions** results in **a convex function**.

$f(x) = x^2 + e^x - \log(x)$ is convex over the interval $x \in (0, \infty)$ because each term is convex

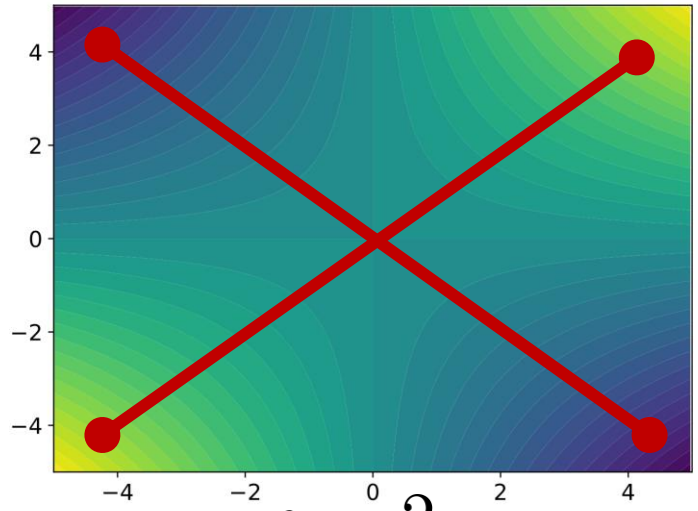
$f(x, y) = x^2 + y^2$ both terms are convex again

We can also multiply functions by a positive scalar without changing convexity.

Multiplying by a negative scalar swaps from convex to concave and vice versa.

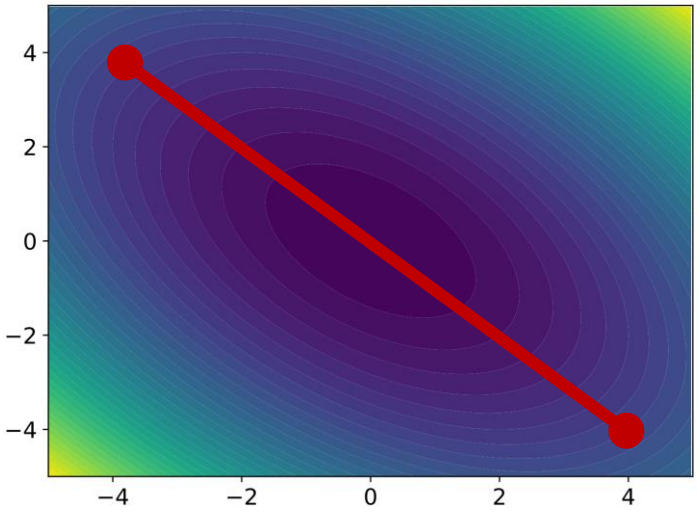
# Combining Functions

Bilinear: $xy$

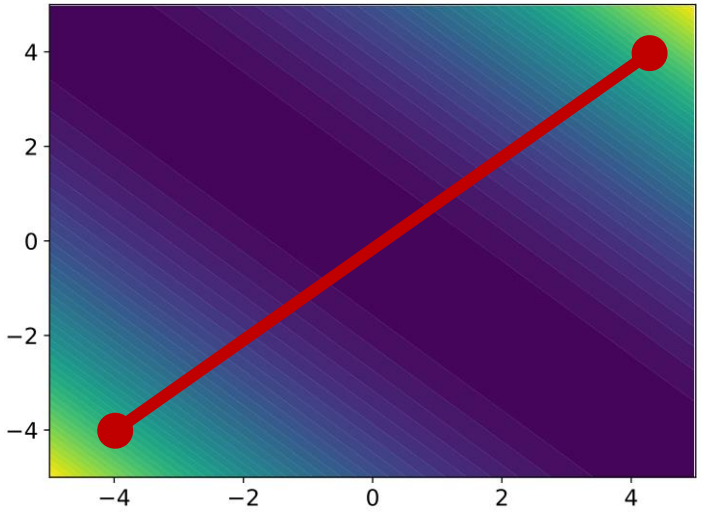



$x^2 + y^2 + axy$

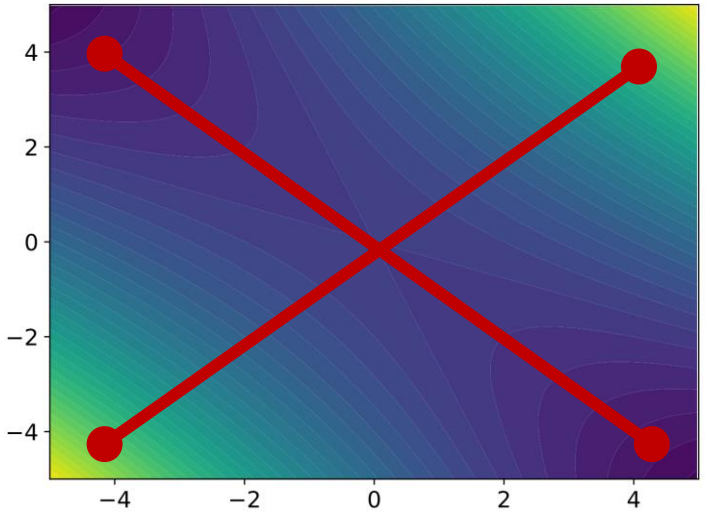
$a = 1$




$a = 2$

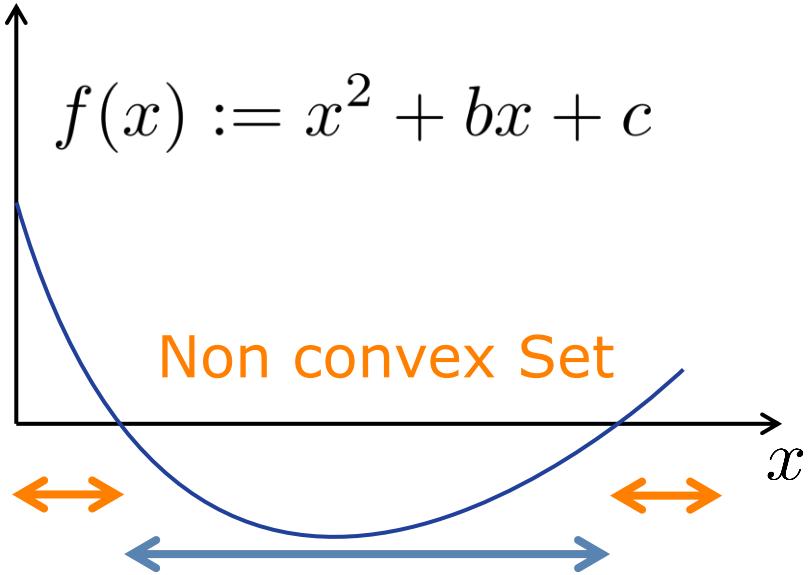



$a = 3$



# Functions in Constraints

Convex Func

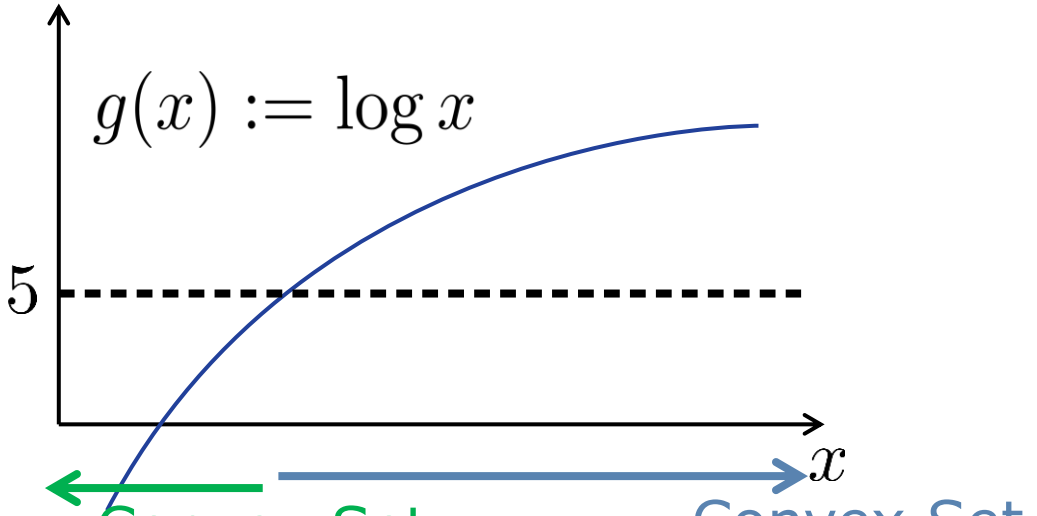



$f(x) \leq 0$ $g(x) \geq 5$ Convex Set

How do we interpret these?

Feasible region for x (the sub / super-level set) is convex.

Concave Func




$f(x) \geq 0$ $g(x) \leq 5$

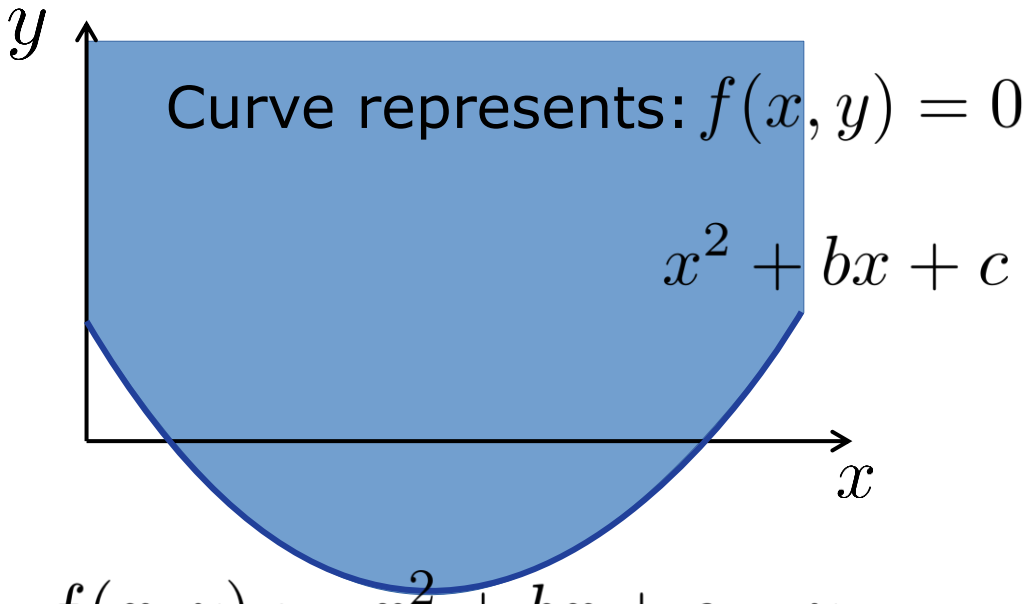
What about these?

Feasible region for x is convex in one case, nonconvex in the other.

In general, if f convex, g concave, these are the only relations *guaranteed* to encode a convex feasible region: $f(x) \leq b$ $g(x) \geq b$

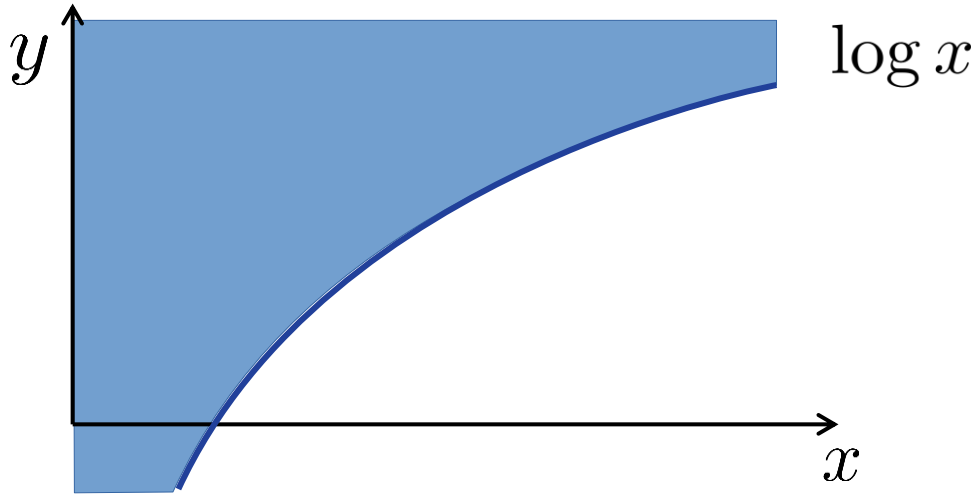
# M-Var Functions in Constraints

Just visualising the sub / super level sets now, rather than graph of function (that would be on a third dimension).




$$f(x,y) := x^2 + bx + c - y$$

$$f(x,y) \leq 0 \implies y \geq x^2 + bx + c$$




$$g(x,y) := \log x - y$$

$$g(x,y) \leq 0 \implies y \geq \log x$$

# Convex Optimisation Forms

$$\min_x \ f(x)$$ convex

$$\text{s.t. } g(x) \leq 0$$ convex

$$\max_x \ f(x)$$ concave

$$\text{s.t. } g(x) \leq 0$$ convex

$$\min_x \ f(x)$$ convex

$$\text{s.t. } h(x) = 0$$

$$x \geq 0$$ convex and concave: affine! (or if slack variables being used to mimic inequality, can be convex or concave depending)

$$\min_x \ f(x)$$ convex

$$\text{s.t. } g(x) \geq 0$$ concave

$$h(x) = 0$$ affine

Assume the domain of all functions is $\mathbb{R}^n$

State the requirements on the functions in each form in order for the problem to be *guaranteed* convex

The objective is a convex function (for min) over a convex feasible set.

# Some Classes

- Linear Programming (LP)
- Quadratic programming (QP)
- Quadratically constrained quadratic programming (QCQP)
- Second-order cone programming (SOCP)
- Semidefinite programming (SDP)
- Conic optimisation
- …

Some of the above classes (e.g., QP) are only convex when particular restrictions are put on them.

Mixed-integer variants of the above exist.

# Computational Complexity

Many common and useful classes of convex optimisation problems, are known to be "solvable" in polynomial time, but not all.

What is a useful notion of complexity when we are dealing with real numbers that we can't exactly represent in a computer anyway?

A common way is to consider **Information-Based Complexity:**

We want to work out how many *iterations* it takes to find an objective value within a certain tolerance of the optimal:

$$|f(x_k) - f(x^*)| \leq \epsilon \qquad k \leq O(d(\epsilon, n, \ldots)) \quad \text{e.g.,} \quad k \leq O\left(\frac{1}{\epsilon}\right)$$

This ignores most of the computation done within an iteration, and hence how the size of the problem factors in unless it contributes to the growth in the number of iterations in a significant way.

**Nemirovski 1995** *Information-based complexity of convex programming*

# Rates of Convergence

A more course measure is to look at the limiting behaviour of algorithms:

$$\epsilon_k := f(x_k) - f(x^*)$$

rate of convergence

$$\lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|} = \mu$$

$\mu = 1$ algorithm converges **sublinearly**

$0 < \mu < 1$ algorithm converges **linearly**

$\mu = 0$ algorithm converges **superlinearly**

$$\lim_{k\to\infty} \frac{|\epsilon_{k+1}|}{|\epsilon_k|^q} < M$$

$M > 0, q > 1$ algorithm converges **superlinearly** with **order q**

$M > 0, q = 2$ algorithm converges **quadratically**

# Rates of Convergence

| | | |
|---|---|---|
| $k \leq O\left(\frac{1}{\epsilon}\right)$ | $\epsilon \leq O\left(\frac{1}{k}\right)$ | **sublinear** |
| $k \leq O\left(\frac{1}{\sqrt{\epsilon}}\right)$ | $\epsilon \leq O\left(\frac{1}{k^2}\right)$ | **sublinear** |
| $k \leq O\left(\log \frac{1}{\epsilon}\right)$ | $\epsilon \leq O\left(e^{-k}\right)$ | **linear** |
| $k \leq O\left(\log_2 \log \frac{1}{\epsilon}\right)$ | $\epsilon \leq O\left(e^{-2^k}\right)$ | **quadratic** |

As a wild generalisation (depends on exact problem class and algorithm):

***First-order methods*** converge either sublinearly or linearly

***Second-order methods*** converge superlinearly or quadratically

# Vector Calculus: Gradient

The gradient is a generalisation of the derivative to multivariate functions

For a differentiable scalar function: $f : \mathbb{R}^n \longrightarrow \mathbb{R}$

$\nabla$ "nabla" / "del"

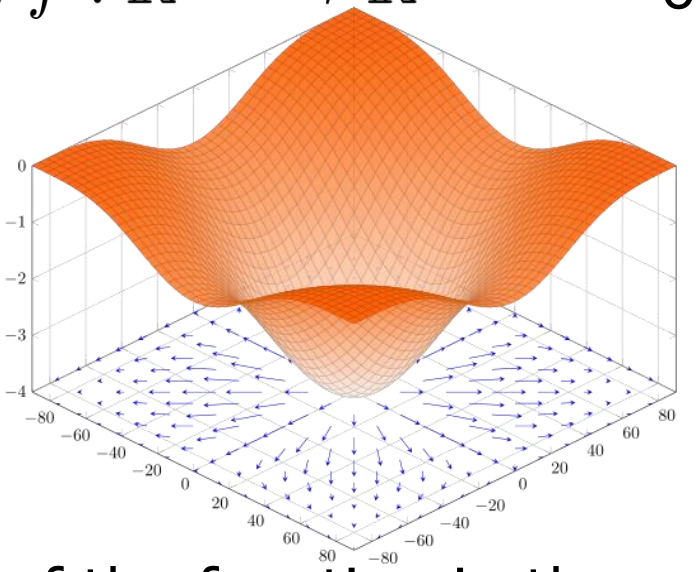
$\nabla f$ the **gradient** of f: $\nabla f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$

$\nabla f(x)$ the gradient of f at x

$$\nabla f(x) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \vdots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}$$

$\nabla f(x)^\intercal u$ the rate of change of the function in the direction of the unit vector u

A vector that gives the direction of greatest increase in the function and strength of that increase.



In the hills analogy, it gives the direction that would lead us up the slope and the steepness of that slope.

$\nabla f(x) = 0$?

# Vector Calculus: Jacobian

The Jacobian generalises the gradient to vector valued functions:

$f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$

$J_f$ the **Jacobian matrix** of f $J_f : \mathbb{R}^n \longrightarrow \mathbb{R}^{m \times n}$

$J_f(x)$ the **Jacobian matrix** of f at x

$$J_f(x) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(x) & \dots & \frac{\partial f_1}{\partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(x) & \dots & \frac{\partial f_m}{\partial x_n}(x) \end{bmatrix} = \begin{bmatrix} \nabla f_1(x)^\intercal \\ \vdots \\ \nabla f_m(x)^\intercal \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) & \dots & \frac{\partial f}{\partial x_n}(x) \end{bmatrix}$$

If m = 1: $J_f(x) = \nabla f(x)^\intercal$

# Vector Calculus: Hessian

The Hessian is a kind of second derivative of a multi-variate scalar function:

$$f : \mathbb{R}^n \longrightarrow \mathbb{R}$$

$H_f$ the **Hessian matrix** of f $\quad H_f : \mathbb{R}^n \longrightarrow \mathbb{R}^{n \times n}$

$H_f(x)$ the **Hessian matrix** of f at x

$$H_f(x) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \vdots & & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(x) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(x) \end{bmatrix}$$

A square matrix that will be **symmetric** if all second partial derivatives are continuous.

$H_f = J_{\nabla f}$ It is the Jacobian of the gradient!

$$f(x, y) = x^2 + y^2 + 4xy$$

$H_f$?

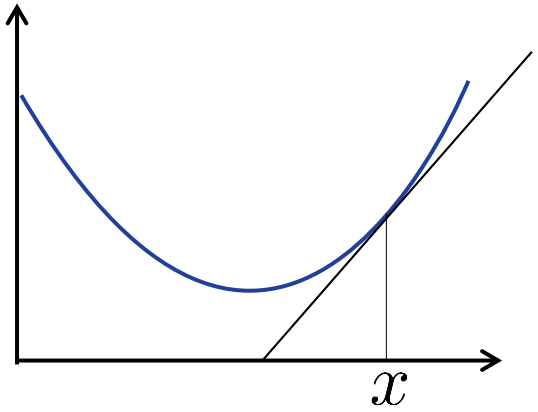
| | x | y |
|---|---|---|
| x | 2 | 4 |
| y | 4 | 2 |

# Differentiable Convex Functions

Convex iff function lies on or above all of its tangents (or tangent planes):

$$f(y) - f(x) \geq \nabla f(x)^\intercal (y - x)$$

$$\forall x, y \in \mathbb{R}^n$$




**Single variate function**

Convex iff: Second derivative everywhere **non-negative**

Strictly convex iff: Second derivative everywhere **strictly positive**

**Multi-variate function**

Convex iff: Hessian matrix is **positive semidefinite** everywhere

Strictly convex iff: Hessian matrix is **positive definite** everywhere

# Definiteness of Matrix

$A \in \mathbb{R}^{n \times n}$ an n by n **symmetric** matrix

**Positive semidefinite** $A \succeq 0$

iff $x^\intercal A x \geq 0 \quad \forall x \in \mathbb{R}^n$

**Positive definite** $A \succ 0$

iff $x^\intercal A x > 0 \quad \forall x \in \mathbb{R}^n \setminus 0$

**Negative (semi-)definite** for the opposite direction, and **indefinite** if none of these

$f(x, y) = x^2 + y^2 + 4xy$

Is f convex?

$$H_f = \begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix}$$

**Constant, so only one Hessian to evaluate**

$$[u, v] \begin{bmatrix} 2 & 4 \\ 4 & 2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = 2u^2 + 2v^2 + 8uv$$

$$u = 1, v = 1 \implies 2 + 2 + 8 = 12 > 0$$

$$u = -1, v = 1 \implies 2 + 2 - 8 = -4 < 0$$

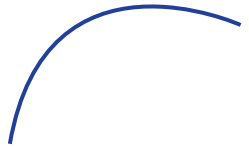
Its Hessian is indefinite, therefore f is non-convex.

# Definiteness

The second derivative / Hessian represents the rate of change of the gradient at a particular point. This indicates that locally:

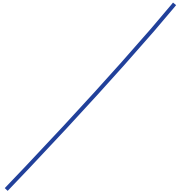
Positive semidefinite: upward facing “cup”

Negative semidefinite: downward facing “cup”



Indefinite: some kind of saddle shape

Zero: flat / linear



Definiteness of Hessian at a stationary point indicates whether or not it is a minimum (pos definite), maximum (neg definite) or saddle point (indefinite). We will use this shortly.

# Quadratic Programming

$$\min_x \ \frac{1}{2}x^\intercal Q x + c^\intercal x$$
$$\text{s.t. } Ax \leq b$$

Quadratic Program (QP)

$Q$ Symmetric

$P_i$

$$\min_x \ \frac{1}{2}x^\intercal Q x + c^\intercal x$$
$$\text{s.t. } \frac{1}{2}x^\intercal P_i x + a_i^\intercal x \leq b_i \quad \forall i \in 1, \ldots, m$$

Quadratically Constrained Quadratic Program (QCQP)

What is the Hessian of the objective? $Q$

When will these problems be convex? $Q, P_i$ Positive semidefinite

# Background: Norms

**A p-norm is:**

$$\|x\|_p = \left( \sum_{i=1}^{n} |x_i|^p \right)^{\frac{1}{p}}$$

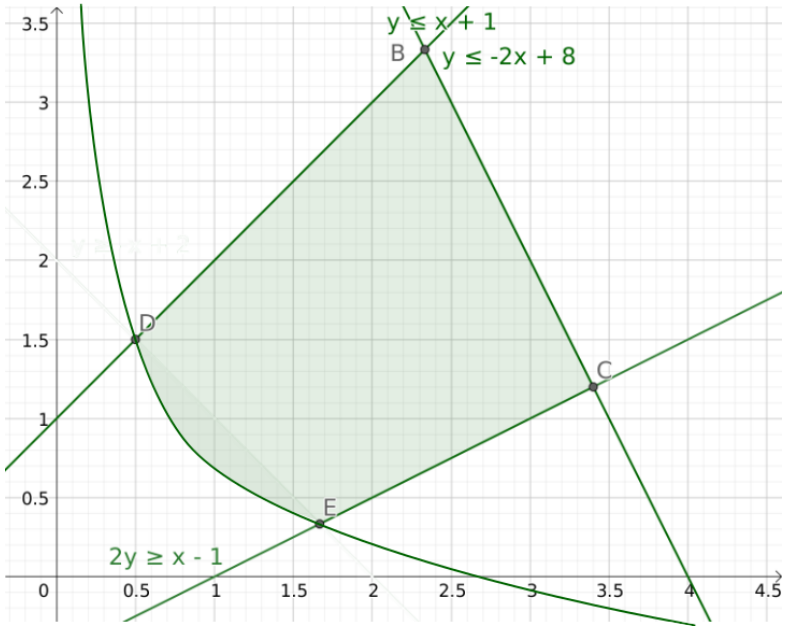
**Euclidean norm:**

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

$$\|x\|_2^2 = \sum_{i=1}^{n} x_i^2 = x^\intercal x$$

# Convex Optimisation Outline

- Convexity
- Unconstrained Optimisation
  - Optimality
  - Gradient Descent
  - Newton's Method
  - Gradient Projection
  - Penalty Method
- Constrained Optimisation
- Lagrangian Duality
- KKT Conditions
- Interior Point Method



# Optimality

For a **constrained** problem with feasible set $X$ and $x^* \in X$ :

$f(x^*)$ is a **global minimum** if $f(x^*) \leq f(x) \quad \forall x \in X$

$f(x^*)$ is a **local minimum** if $\exists \epsilon > 0$ such that:

$$f(x^*) \leq f(x) \quad \forall x \in \{x \in X : \|x - x^*\| \leq \epsilon\}$$

For an **unconstrained problem** replace $X$ with $\mathbb{R}^n$

The beauty of **convex optimisation problems** is that any local minimum is also a global minimum.

Unfortunately the above is not really that useful for **finding** minima.

# Unconstrained Optimality

An **unconstrained**, **twice-differentiable** function has a local minimum at a point if:

$$\boxed{\nabla f(x^*) = 0} \quad H_f(x^*) \succeq 0$$

**Together** these are **sufficient** conditions for a local minimum.

**Separately they are only necessary**, but not sufficient.

First-order **necessary** condition for unconstrained problem local optimum

Since a convex function has a positive semidefinite Hessian everywhere:

If the function is convex, then first-order condition becomes a **sufficient** condition

# Unconstrained Optimisation

$$\min_{x \in \mathbb{R}^n} f(x)$$

If differentiable, we can attempt to solve the first-order necessary condition for x: $\nabla f(x) = 0$

Again, when convex this is sufficient to find a local and global optimal. If nonconvex, this could be a stationary point: local minimum only or maximum or saddle point.

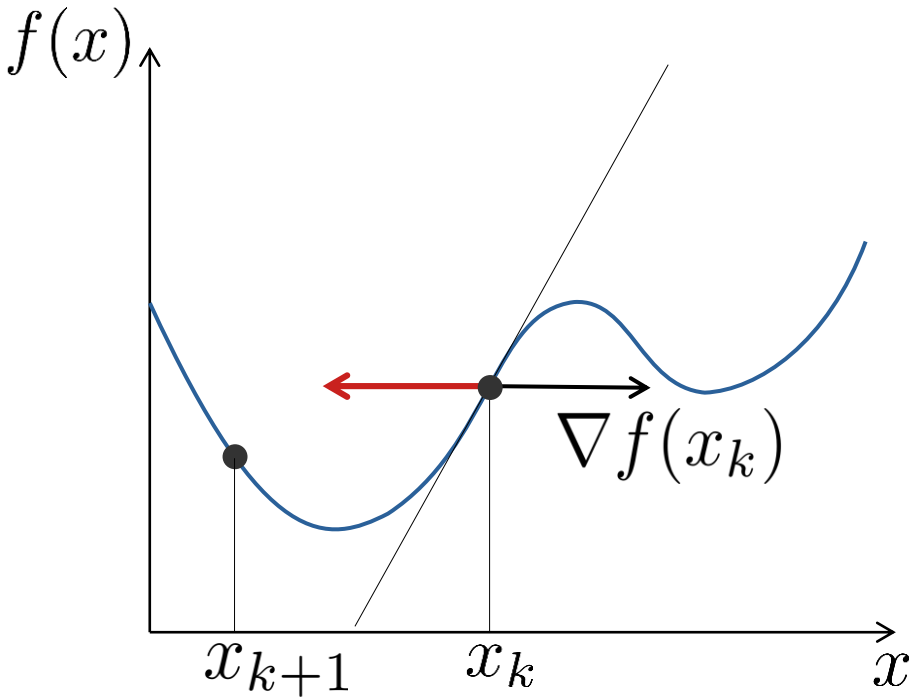
How challenging is it to find the zeros / roots?

In special circumstances we can do it analytically (small number of vars, small degree polynomials), but in general (Abel–Ruffini theorem) we will need to resort to **numerical methods**.

# Gradient Descent

For unconstrained differentiable problems.

$\min_{x \in \mathbb{R}^n} f(x)$ $f(x)$ differentiable




$x_{k=0}$ a starting value

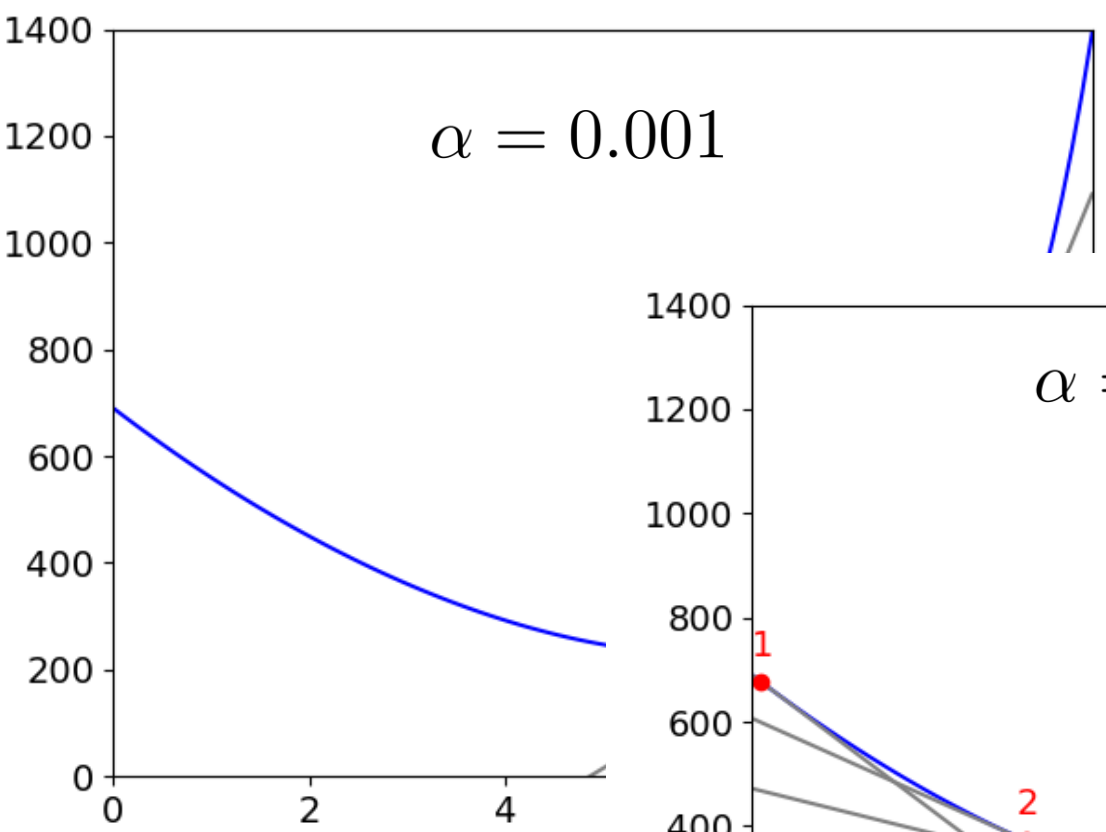
$\nabla f(x_k)$ the gradient for the k-th iterate

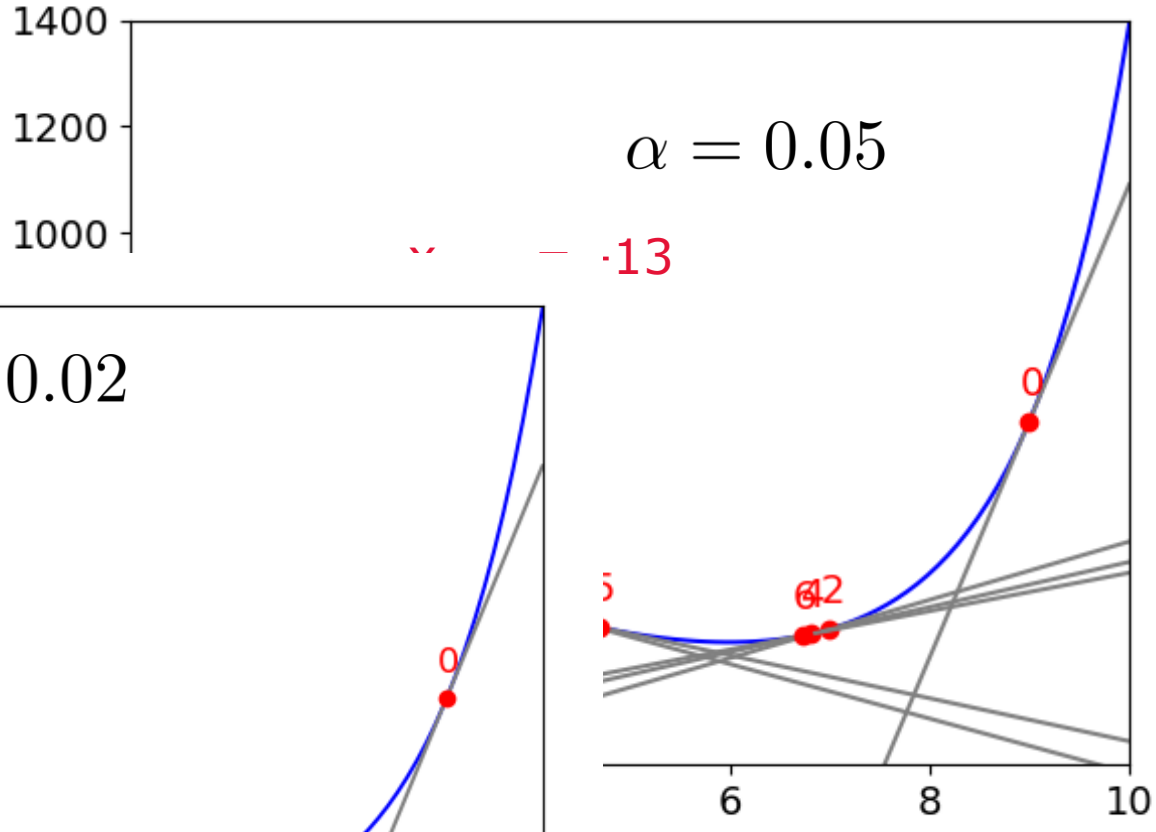
Points in the direction that would take us up the steepest part of the hill, and has magnitude equal to that slope.

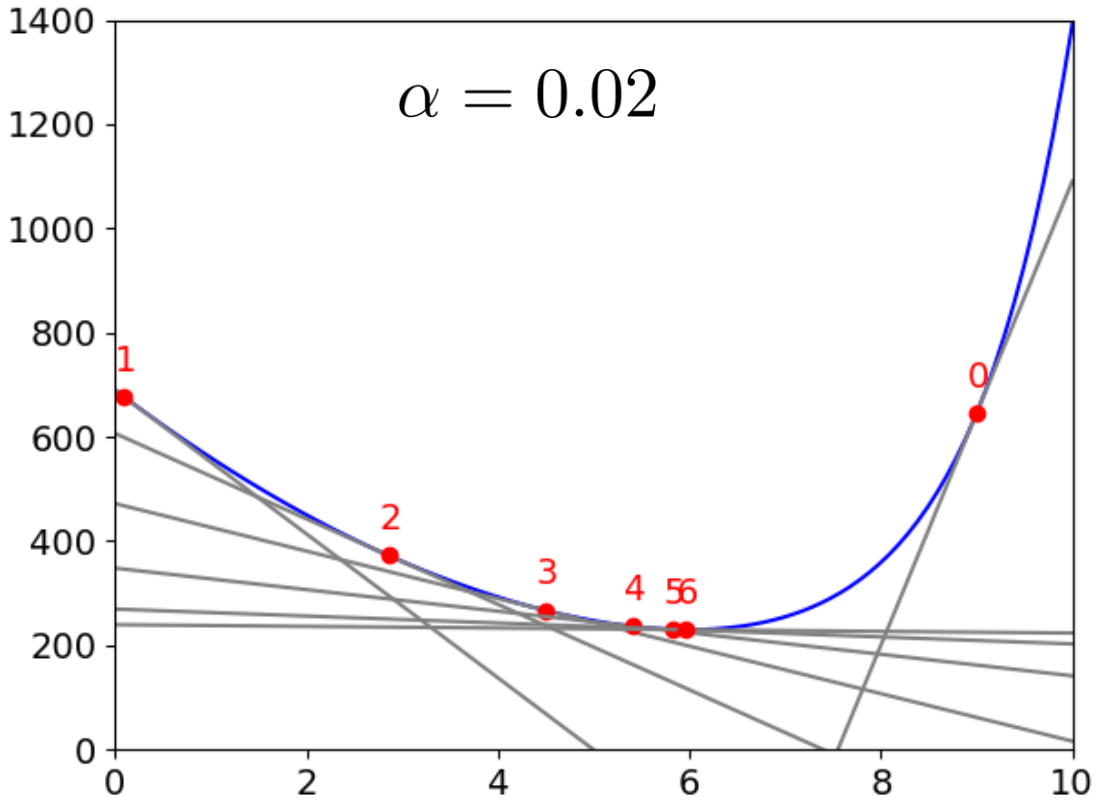
If we are minimising, we need to step in the opposite direction:

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$ for some step size $\alpha_k$

# Gradient Descent



# Stepping

The way this **step size** is selected / updated is critical to guarantee convergence, and to getting a fast convergence. Some options:

**Fixed** step size

- Potentially very slow if it converges at all.

**Rule-based** step size that has been proven to converge.

- Might use information from previous iterates.

**Line Search**: Optimise the the step in the descent direction.

- Gathers more information by looking at previously un-explored points.
- Can be optimised **Exactly**, or **Inexactly**.
- Often still have some rules about what steps we can accept in order to guarantee / improve convergence.

# Rule-based Step Size for GD

To give you an idea, **two** options from **Barzilai and Borwein 1988** *Two-Point Step Size Gradient Methods:*

1. $$\alpha_k := \frac{(x_k - x_{k-1})^\intercal(\nabla f(x_k) - \nabla f(x_{k-1}))}{\|\nabla f(x_k) - \nabla f(x_{k-1})\|^2}$$

*Smaller steps when gradient is changing rapidly, either in magnitude or direction.*

2. $$\alpha_k := \frac{\|x_k - x_{k-1}\|^2}{(x_k - x_{k-1})^\intercal(\nabla f(x_k) - \nabla f(x_{k-1}))}$$

Use iterate and gradient values from last two iterates.

# Line Search

Solve problem:

$$\alpha_k := \arg\min_{\alpha} f(x_k + \alpha d_k)$$

optimising a single variable (moving in a straight line from our current iterate)

our search direction

where for gradient descent:

$$d_k = -\nabla f(x_k)$$

Either **exactly**, which might take time, or **inexactly** using an approach such as **backtracking line search**.

We might not want to spend too much time doing this exactly because our current search direction might not be that great to start with!

**Backtracking line search**:

1. Start with a large step estimate.
2. Iteratively shrink the step size until some conditions are met: i.e. the objective value has "adequately" improved.
3. Continue onto calculating a new search direction.

# Methods

Gradient descent is a **first-order** method (recall general statements in rate of convergence slides).

**First-order** methods locally approximate function as being linear, in deciding the step direction (using **first derivative** information).

**Second-order** methods locally approximate function as being quadratic, in deciding the step direction (using **second derivative** information).

In the next lecture, we will take a look at a second-order method for unconstrained problems.

# Image Attributions

By MartinThoma - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=71375503