

AI505  
Optimization

## Derivatives and Gradients

Marco Chiarandini

Department of Mathematics & Computer Science  
University of Southern Denmark

# Outline

Derivatives  
Numerical Differentiation  
Automatic Differentiation

1. Derivatives

2. Numerical Differentiation

3. Automatic Differentiation

# Definitions

- $[a, b] = \{x \in \mathbb{R} \mid a \leq x \leq b\}$  closed interval  
 $(a, b) = \{x \in \mathbb{R} \mid a < x < b\}$  open interval
- column vectors and matrices  
 scalar product:  $\mathbf{y}^T \mathbf{x} = \sum_{i=1}^n y_i x_i$
- $A\mathbf{x}$  column vector combination of the columns of  $A$ ;  
 $\mathbf{u}^T A$  row vector combination of the rows of  $A$
- **linear combination**

$$\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k \in \mathbb{R}^n$$

$$\boldsymbol{\lambda} = [\lambda_1, \dots, \lambda_k]^T \in \mathbb{R}^k$$

$$\mathbf{x} = \lambda_1 \mathbf{v}_1 + \dots + \lambda_k \mathbf{v}_k = \sum_{i=1}^k \lambda_i \mathbf{v}_i$$

moreover:

$$\lambda \geq 0$$

**conic combination**

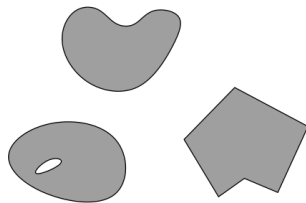
$$\boldsymbol{\lambda}^T \mathbf{1} = 1$$

**affine combination**

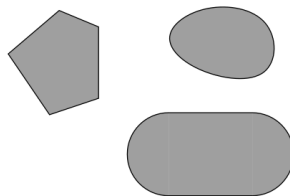
$$\left( \sum_{i=1}^k \lambda_i = 1 \right)$$

# Definitions

- convex set:** if  $\mathbf{x}, \mathbf{y} \in S$  and  $0 \leq \lambda \leq 1$  then  $\lambda \mathbf{x} + (1 - \lambda) \mathbf{y} \in S$

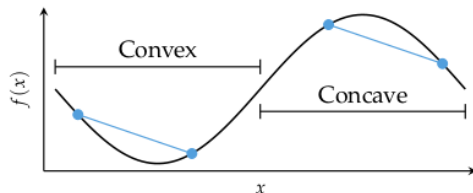


nonconvex

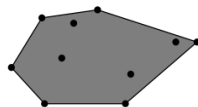
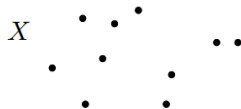


convex

- convex function** if its epigraph  $\{(x, y) \in \mathbb{R}^2 : y \geq f(x)\}$  is a convex set or if  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  and if  $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n, \alpha \in [0, 1]$  it holds that  $f(\alpha \mathbf{x} + (1 - \alpha) \mathbf{y}) \leq \alpha f(\mathbf{x}) + (1 - \alpha) f(\mathbf{y})$



- For a set of points  $S \subseteq \mathbb{R}^n$ 
  - $\text{lin}(S)$  linear hull (span)
  - $\text{cone}(S)$  conic hull
  - $\text{aff}(S)$  affine hull
  - $\text{conv}(S)$  convex hull



the convex hull of  $X$

$$\text{conv}(X) = \{ \lambda_1 \mathbf{x}_1 + \lambda_2 \mathbf{x}_2 + \dots + \lambda_n \mathbf{x}_n \mid \mathbf{x}_i \in X, \lambda_1, \dots, \lambda_n \geq 0 \text{ and } \sum_i \lambda_i = 1 \}$$

# Norms

Def. A **norm** is a function that assigns a length to a vector.

A function  $f$  is a norm if:

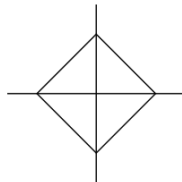
1.  $f(\mathbf{x}) = 0$  if and only if  $\mathbf{x}$  is the zero vector
2.  $f(a\mathbf{x}) = |a|f(\mathbf{x})$ , such that lengths scale
3.  $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$ , also known as triangle inequality

$L_p$  norms are commonly used set of norms parameterized by a scalar  $p \geq 1$ :

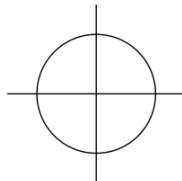
$$\|\mathbf{x}\|_p = \lim_{\rho \rightarrow p} (|\mathbf{x}_1|^\rho + |\mathbf{x}_2|^\rho + \dots + |\mathbf{x}_n|^\rho)^{\frac{1}{\rho}}$$

$L_\infty$  is also called the max norm, Chebyshev distance or chessboard distance.

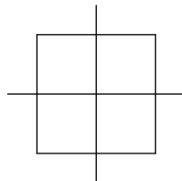
$$L_1: \|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_n|$$



$$L_2: \|\mathbf{x}\|_2 = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$$



$$L_\infty: \|\mathbf{x}\|_\infty = \max(|x_1|, |x_2|, \cdots, |x_n|)$$



# Outline

**Derivatives**  
Numerical Differentiation  
Automatic Differentiation

1. Derivatives

2. Numerical Differentiation

3. Automatic Differentiation

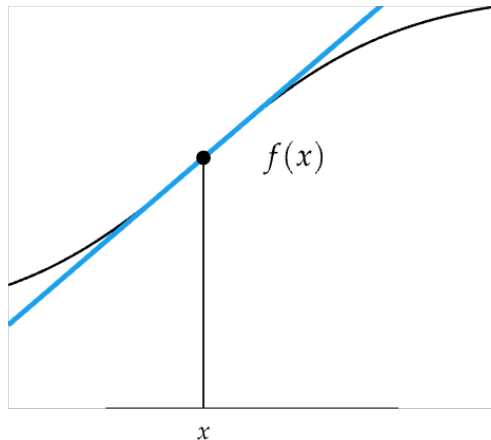


# Derivatives

- Derivatives tell us which direction to search for a solution
- Slope of Tangent Line

$$f'(x) := \frac{df(x)}{dx}$$

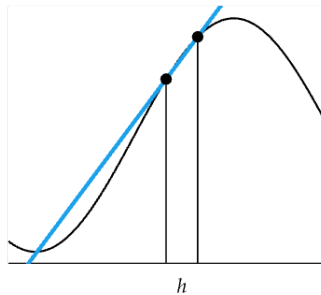
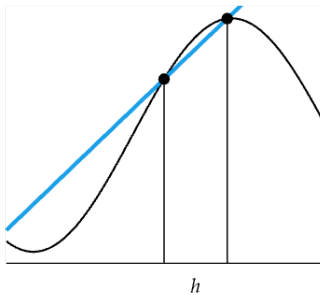
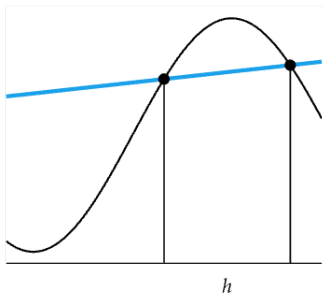
(Leibniz notation)



# Derivatives

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x$$

$$f'(x) = \frac{\Delta y}{\Delta x}$$



# Symbolic Differentiation

$$f'(x) \equiv \lim_{h \rightarrow 0} \underbrace{\frac{f(x+h) - f(x)}{h}}_{\text{forward difference}} = \lim_{h \rightarrow 0} \underbrace{\frac{f(x+h/2) - f(x-h/2)}{h}}_{\text{central difference}} = \lim_{h \rightarrow 0} \underbrace{\frac{f(x) - f(x-h)}{h}}_{\text{backward difference}}$$

# Symbolic Differentiation

```
import sympy as sp

# Define the variable
x = sp.symbols('x')

# Define the function
f = x**2 + x/2 - sp.sin(x)/x

# Compute the derivative
df_dx = sp.diff(f, x)

# Display the result
print("The symbolic derivative of f is:")
print(df_dx)
```

derivative.py

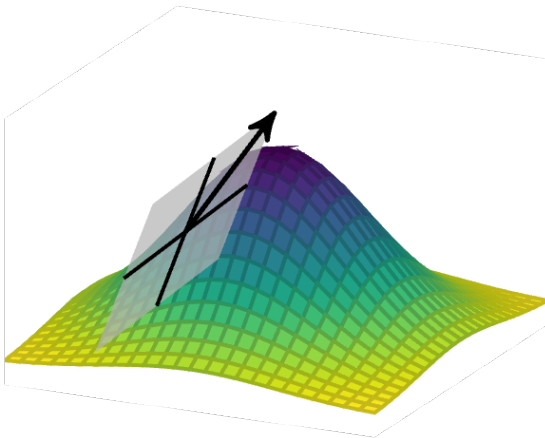
# Derivatives in Multiple Dimensions

- Gradient Vector

$$\nabla f(\mathbf{x}) = \left[ \frac{\partial f(\mathbf{x})}{\partial x_1}, \frac{\partial f(\mathbf{x})}{\partial x_2}, \dots, \frac{\partial f(\mathbf{x})}{\partial x_n} \right]$$

- Hessian Matrix

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



# Directional derivative

The **directional derivative**  $\nabla_s f(\mathbf{x})$  of a multivariate function  $f$  is the instantaneous rate of change of  $f(\mathbf{x})$  as  $\mathbf{x}$  is moved with velocity  $\mathbf{s}$ .

$$\nabla_s f(\mathbf{x}) \equiv \underbrace{\lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{s}) - f(\mathbf{x})}{h}}_{\text{forward difference}} = \underbrace{\lim_{h \rightarrow 0} \frac{f(\mathbf{x} + h\mathbf{s}/2) - f(\mathbf{x} - h\mathbf{s}/2)}{h}}_{\text{central difference}} = \underbrace{\lim_{h \rightarrow 0} \frac{f(\mathbf{x}) - f(\mathbf{x} - h\mathbf{s})}{h}}_{\text{backward difference}}$$

To compute  $\nabla_s f(\mathbf{x})$ :

- compute  $\nabla_s f(\mathbf{x}) = \nabla f(\mathbf{x})^T \mathbf{s}$
- $g(\alpha) := f(\mathbf{x} + \alpha \mathbf{s})$  and then compute  $g'(0)$

# Matrix Calculus

Common gradient:

$$\nabla_{\mathbf{x}} \mathbf{b}^T \mathbf{x} = ?$$

$$\mathbf{b}^T \mathbf{x} = [b_1 x_1 + b_2 x_2 + \dots + b_n x_n]$$

$$\frac{\partial \mathbf{b}^T \mathbf{x}}{\partial x_i} = b_i$$

$$\nabla_{\mathbf{x}} \mathbf{b}^T \mathbf{x} = \nabla_{\mathbf{x}} \mathbf{x}^T \mathbf{b} = \mathbf{b}$$

# Matrix Calculus

Common gradient:

$$\nabla_{\mathbf{x}} \mathbf{x}^T A \mathbf{x} = ?$$

$$\mathbf{x}^T A \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^T \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}^T \begin{bmatrix} x_1 a_{11} + x_2 a_{12} + \dots + x_n a_{1n} \\ x_1 a_{21} + x_2 a_{22} + \dots + x_n a_{2n} \\ \vdots \\ x_1 a_{n1} + x_2 a_{n2} + \dots + x_n a_{nn} \end{bmatrix}$$

$$\begin{aligned} & x_1^2 a_{11} + x_1 x_2 a_{12} + \dots + x_1 x_n a_{1n} + \\ & x_1 x_2 a_{21} + x_2^2 a_{22} + \dots + x_2 x_n a_{2n} + \\ & = \quad \quad \quad \vdots \\ & x_1 x_n a_{n1} + x_2 x_n a_{n2} + \dots + x_n^2 a_{nn} \end{aligned}$$



$$\frac{\partial}{\partial x_i} \mathbf{x}^T A \mathbf{x} = \sum_{j=1}^n x_j (a_{ij} + a_{ji})$$

$$\nabla_{\mathbf{x}} \mathbf{x}^T A \mathbf{x} = \begin{bmatrix} \sum_{j=1}^n x_j (a_{1j} + a_{j1}) \\ \sum_{j=1}^n x_j (a_{2j} + a_{j2}) \\ \vdots \\ \sum_{j=1}^n x_j (a_{nj} + a_{jn}) \end{bmatrix} = \begin{bmatrix} a_{11} + a_{11} & a_{12} + a_{21} & \dots & a_{1n} + a_{n1} \\ a_{21} + a_{12} & a_{22} + a_{22} & \dots & a_{2n} + a_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} + a_{1n} & a_{n2} + a_{2n} & \dots & a_{nn} + a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = (A + A^T) \mathbf{x}$$

Def. The **smoothness** of a function is a property measured by the number of continuous derivatives (differentiability class) it has over its domain.

A function of **class**  $C^k$  is a function of smoothness at least  $k$ ; that is, a function of class  $C^k$  is a function that has a  $k$ th derivative that is continuous in its domain.

The term **smooth function** refers to a  $C^\infty$ -function. However, it may also mean “sufficiently differentiable” for the problem under consideration.

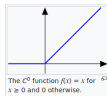
- Let  $U$  be an open set on the real line and a function  $f$  defined on  $U$  with real values. Let  $k$  be a non-negative integer.
- The function  $f$  is said to be of **differentiability class**  $C^k$  if the derivatives  $f', f'', \dots, f^{(k)}$  exist and are continuous on  $U$ .
- If  $f$  is  $k$ -differentiable on  $U$ , then it is at least in the class  $C^{k-1}$  since  $f', f'', \dots, f^{(k-1)}$  are continuous on  $U$ .
- The function  $f$  is said to be **infinitely differentiable**, **smooth**, or of **class**  $C^\infty$ , if it has derivatives of all orders (continuous) on  $U$ .
- The function  $f$  is said to be of **class**  $C^\omega$ , or **analytic**, if  $f$  is smooth and its Taylor series expansion around any point in its domain converges to the function in some neighborhood of the point.
- There exist functions that are smooth but not analytic;  $C^\omega$  is thus strictly contained in  $C^\infty$ . Bump functions are examples of functions with this property.

**Example: continuous ( $C^0$ ) but not differentiable** [\[ edit \]](#)

The function

$$f(x) = \begin{cases} x & \text{if } x \geq 0, \\ 0 & \text{if } x < 0 \end{cases}$$

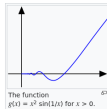
is continuous, but not differentiable at  $x = 0$ , so it is of class  $C^0$ , but not of class  $C^1$ .

**Example: finitely-times differentiable ( $C^k$ )** [\[ edit \]](#)

For each even integer  $k$ , the function

$$f(x) = |x|^{k+1}$$

is continuous and  $k$  times differentiable at all  $x$ . At  $x = 0$ , however,  $f$  is not  $(k+1)$  times differentiable, so  $f$  is of class  $C^k$ , but not of class  $C^j$  where  $j > k$ .

**Example: differentiable but not continuously differentiable (not  $C^1$ )** [\[ edit \]](#)

The function

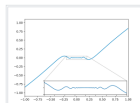
$$g(x) = \begin{cases} x^2 \sin\left(\frac{1}{x}\right) & \text{if } x \neq 0, \\ 0 & \text{if } x = 0 \end{cases}$$

is differentiable, with derivative

$$g'(x) = \begin{cases} -\cos\left(\frac{1}{x}\right) + 2x \sin\left(\frac{1}{x}\right) & \text{if } x \neq 0, \\ 0 & \text{if } x = 0. \end{cases}$$

Because  $\cos(1/x)$  oscillates as  $x \rightarrow 0$ ,  $g'(x)$  is not continuous at zero.

Therefore,  $g(x)$  is differentiable but not of class  $C^1$ .



The function  $f: \mathbb{R} \rightarrow \mathbb{R}$  with  $f(x) = x^2 \sin\left(\frac{1}{x}\right)$  for  $x \neq 0$  and  $f(0) = 0$  is differentiable. However, this function is not continuously differentiable.

**Example: differentiable but not Lipschitz continuous** [\[ edit \]](#)

The function

$$h(x) = \begin{cases} x^{1/3} \sin\left(\frac{1}{x}\right) & \text{if } x \neq 0, \\ 0 & \text{if } x = 0 \end{cases}$$

is differentiable but its derivative is unbounded on a [compact set](#). Therefore,  $h$  is an example of a function that is differentiable but not locally [Lipschitz continuous](#).

**Example: analytic ( $C^\omega$ )** [\[ edit \]](#)

The [exponential function](#)  $e^x$  is [analytic](#), and hence falls into the class  $C^\omega$  (where  $\omega$  is the smallest [transfinite ordinal](#)). The [trigonometric functions](#) are also analytic wherever they are defined, because they are [linear combinations of complex exponential functions](#)  $e^{itx}$  and  $e^{-itx}$ .

**Example: smooth ( $C^\infty$ ) but not analytic ( $C^\omega$ )** [\[ edit \]](#)

The [bump function](#)

$$f(x) = \begin{cases} e^{-\frac{1}{1-x^2}} & \text{if } |x| < 1, \\ 0 & \text{otherwise} \end{cases}$$

is smooth, so of class  $C^\infty$ , but it is not analytic at  $x = \pm 1$ , and hence is not of class  $C^\omega$ . The function  $f$  is an example of a smooth function with [compact support](#).

# Positive Definiteness

Def. A symmetric matrix  $A$  is **positive definite** if  $\mathbf{x}^T A \mathbf{x}$  is positive for all points other than the origin:  $\mathbf{x}^T A \mathbf{x} > 0$  for all  $\mathbf{x} \neq 0$ .

Def. A symmetric matrix  $A$  is **positive semidefinite** if  $\mathbf{x}^T A \mathbf{x}$  is always non-negative:  $\mathbf{x}^T A \mathbf{x} \geq 0$  for all  $\mathbf{x}$ .

If the matrix  $A$  is positive definite in the function  $f(\mathbf{x}) = \mathbf{x}^T A \mathbf{x}$ , then  $f$  has a unique global minimum.

Recall that the second order Taylor approximation of a twice-differentiable function  $f$  at  $\mathbf{x}_0$  is

$$f(\mathbf{x}) \approx f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)^T (\mathbf{x} - \mathbf{x}_0) + \frac{1}{2} (\mathbf{x} - \mathbf{x}_0)^T H_0 (\mathbf{x} - \mathbf{x}_0)$$

where  $H_0$  is the Hessian evaluated at  $\mathbf{x}_0$ . If  $(\mathbf{x} - \mathbf{x}_0)^T H_0 (\mathbf{x} - \mathbf{x}_0)$  has a unique global minimum, then the overall approximation has a unique global minimum.

We wish to compute the directional derivative of  $f(\mathbf{x}) = x_1x_2$  at  $\mathbf{x} = [1, 0]$  in the direction  $\mathbf{s} = [-1, -1]$ :

$$\nabla f(\mathbf{x}) = \left[ \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2} \right] = [x_2, x_1]$$

$$\nabla_{\mathbf{s}} f(\mathbf{x}) = \nabla f(\mathbf{x})^\top \mathbf{s} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} -1 \\ -1 \end{bmatrix} = -1$$

We can also compute the directional derivative as follows:

$$g(\alpha) = f(\mathbf{x} + \alpha \mathbf{s}) = (1 - \alpha)(-\alpha) = \alpha^2 - \alpha$$

$$g'(\alpha) = 2\alpha - 1$$

$$g'(0) = -1$$

# Outline

Derivatives  
**Numerical Differentiation**  
Automatic Differentiation

1. Derivatives

2. Numerical Differentiation

3. Automatic Differentiation

## Finite Difference Method

- Neighboring points are used to approximate the derivative

$$f'(x) \approx \underbrace{\frac{f(x+h) - f(x)}{h}}_{\text{forward difference}} \approx \underbrace{\frac{f(x+h/2) - f(x-h/2)}{h}}_{\text{central difference}} \approx \underbrace{\frac{f(x) - f(x-h)}{h}}_{\text{backward difference}}$$

- $h$  too small causes numerical cancellation errors (square root or cube root of the machine precision for floating point values: `sys.float_info.epsilon` difference between 1 and closest representable number)



# Derivation

from Taylor series expansion:

$$f(x+h) = f(x) + \frac{f'(x)}{1!}h + \frac{f''(x)}{2!}h^2 + \frac{f'''(x)}{3!}h^3 + \dots$$

We can rearrange and solve for the first derivative:

$$f'(x)h = f(x+h) - f(x) - \frac{f''(x)}{2!}h^2 - \frac{f'''(x)}{3!}h^3 - \dots$$

$$f'(x) = \frac{f(x+h) - f(x)}{h} - \frac{f''(x)}{2!}h - \frac{f'''(x)}{3!}h^2 - \dots$$

$$f'(x) \approx \frac{f(x+h) - f(x)}{h}$$

- forward difference has error term  $O(h)$ , linear error as  $h$  approaches zero
- central difference has error term is  $O(h^2)$

```
import sys
import numpy as np

def diff_forward(f, x: float, h: float=np.sqrt(sys.float_info.epsilon)) -> float:
    return (f(x+h) - f(x))/h

def diff_central(f, x: float, h: float=np.cbrt(sys.float_info.epsilon)) -> float:
    return (f(x+h/2) - f(x-h/2))/h

def diff_backward(f, x: float, h: float=np.sqrt(sys.float_info.epsilon)) -> float:
    return (f(x) - f(x-h))/h

# Example usage
def func(x):
    return x**2 + np.sin(x)

x0 = 1.0
print(f"The derivative at x = {x0} is {diff_forward(func, x0)}")
```

# Numerical Differentiation

## Complex step method

Uses one single function evaluation after taking a step in the imaginary direction.

$$f(x + ih) = f(x) + ihf'(x) - h^2 \frac{f''(x)}{2!} - ih^3 \frac{f'''(x)}{3!} + \dots$$

$$\operatorname{Im}(f(x + ih)) = hf'(x) - h^3 \frac{f'''(x)}{3!} + \dots$$

$$\begin{aligned}\Rightarrow f'(x) &= \frac{\operatorname{Im}(f(x + ih))}{h} + h^2 \frac{f'''(x)}{3!} - \dots \\ &= \frac{\operatorname{Im}(f(x + ih))}{h} + O(h^2) \text{ as } h \rightarrow 0\end{aligned}$$

$$\operatorname{Re}(f(x + ih)) = f(x) - h^2 \frac{f''(x)}{2!} + \dots$$

$$\Rightarrow f(x) = \operatorname{Re}(f(x + ih)) + h^2 \frac{f''(x)}{2!} - \dots$$

```
import numpy as np

def diff_complex(f, x: float, h: float=1e-20) -> float:
    return np.imag(f(x + h * 1j)) / h

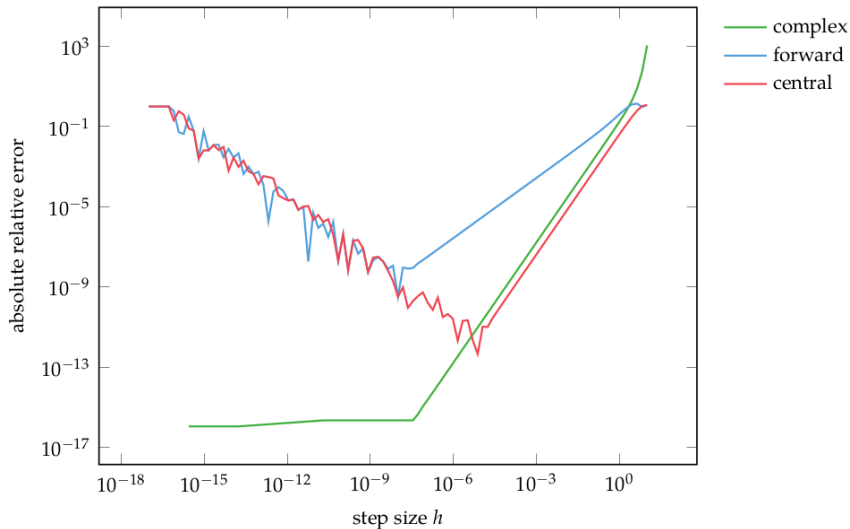
# Example usage
def func(x):
    return x**2 + np.sin(x)

x0 = 1.0
print(f"The derivative at x = {x0} is {diff_complex(func, x0)}")
```

complex\_diff.py

# Numerical Differentiation Error Comparison

Derivatives  
**Numerical Differentiation**  
Automatic Differentiation



# Outline

Derivatives  
Numerical Differentiation  
**Automatic Differentiation**

1. Derivatives

2. Numerical Differentiation

3. Automatic Differentiation

Evaluate a function and compute partial derivatives simultaneously using the **chain rule** of differentiation

$$\frac{d}{dx}f(g(x)) = \frac{d}{dx}f \circ g(x) = \frac{df}{dg} \frac{dg}{dx}$$

- Forward Accumulation is equivalent to expanding a function using the chain rule and computing the derivatives inside-out
- Requires  $n$ -passes to compute  $n$ -dimensional gradient
- Example:

$$f(a, b) = \ln(ab + \max(a, 2))$$

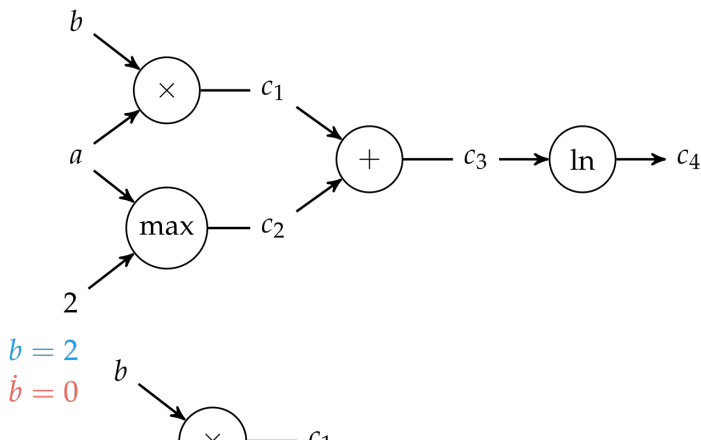
$$\begin{aligned} \frac{\partial f}{\partial a} &= \frac{\partial}{\partial a} \ln(ab + \max(a, 2)) \\ &= \frac{1}{ab + \max(a, 2)} \frac{\partial}{\partial a} (ab + \max(a, 2)) \\ &= \frac{1}{ab + \max(a, 2)} \left[ \frac{\partial(ab)}{\partial a} + \frac{\partial \max(a, 2)}{\partial a} \right] \\ &= \frac{1}{ab + \max(a, 2)} \left[ \left( b \frac{\partial a}{\partial a} + a \frac{\partial b}{\partial a} \right) + \left( (2 > a) \frac{\partial 2}{\partial a} + (2 < a) \frac{\partial a}{\partial a} \right) \right] \\ &= \frac{1}{ab + \max(a, 2)} [b + (2 < a)] \end{aligned}$$



# Automatic Differentiation

**Computational graph:** nodes are operations and the edges are input-output relations. leaf nodes of a computational graph are input variables or constants, and terminal nodes are values output by the function

**Forward accumulation** for  $f(a, b) = \ln(ab + \max(a, 2))$



# Dual numbers

- Dual numbers can be expressed mathematically by including the abstract quantity  $\epsilon$ , where  $\epsilon^2$  is defined to be 0.
- Like a complex number, a dual number is written  $a + b\epsilon$  where  $a$  and  $b$  are both real values.
- $(a + b\epsilon) + (c + d\epsilon) = (a + c) + (b + d)\epsilon$   
 $(a + b\epsilon) \times (c + d\epsilon) = (ac) + (ad + bc)\epsilon$
- by passing a dual number into any smooth function  $f$ , we get the evaluation and its derivative.  
 We can show this using the Taylor series:

$$\begin{aligned}
 f(x) &= \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!} (x - a)^k & = f(a) + bf'(a)\epsilon + \epsilon^2 \sum_{k=2}^{\infty} \frac{f^{(k)}(a)b^k}{k!} \epsilon^{(k-2)} \\
 f(a + b\epsilon) &= \sum_{k=0}^{\infty} \frac{f^{(k)}(a)}{k!} (a + b\epsilon - a)^k & = f(a) + bf'(a)\epsilon \\
 &= \sum_{k=0}^{\infty} \frac{f^{(k)}(a)b^k \epsilon^k}{k!}
 \end{aligned}$$

# Automatic Differentiation

- **Reverse accumulation** is performed in single run using two passes over an  $n$ -dimensional function (forward and back)
- implemented through two different operation overloading functions (for forward and backward)
- Note: this is central to the backpropagation algorithm used to train neural networks
- Many open-source software implementations are available: eg, Tensorflow

$$\frac{df}{dx} = \frac{df}{dc_4} \frac{dc_4}{dx} = \frac{df}{dc_4} \left( \frac{dc_4}{dc_3} \frac{dc_3}{dx} \right) = \frac{df}{dc_4} \left( \frac{dc_4}{dc_3} \left( \frac{dc_3}{dc_2} \frac{dc_2}{dx} + \frac{dc_3}{dc_1} \frac{dc_1}{dx} \right) \right)$$

$$\frac{df}{dx} = \frac{df}{dc_4} \frac{dc_4}{dx} = \left( \frac{df}{dc_3} \frac{dc_3}{dc_4} \right) \frac{dc_4}{dx} = \left( \left( \frac{df}{dc_2} \frac{dc_2}{dc_3} + \frac{df}{dc_1} \frac{dc_1}{dc_3} \right) \frac{dc_3}{dc_4} \right) \frac{dc_4}{dx}$$

- Derivatives are useful in optimization because they provide information about how to change a given point in order to improve the objective function
- For multivariate functions, various derivative-based concepts are useful for directing the search for an optimum, including the gradient, the Hessian, and the directional derivative
- One approach to numerical differentiation includes finite difference approximations
- Complex step method can eliminate the effect of subtractive cancellation error when taking small steps
- Analytic differentiation methods include forward and reverse accumulation on computational graphs