



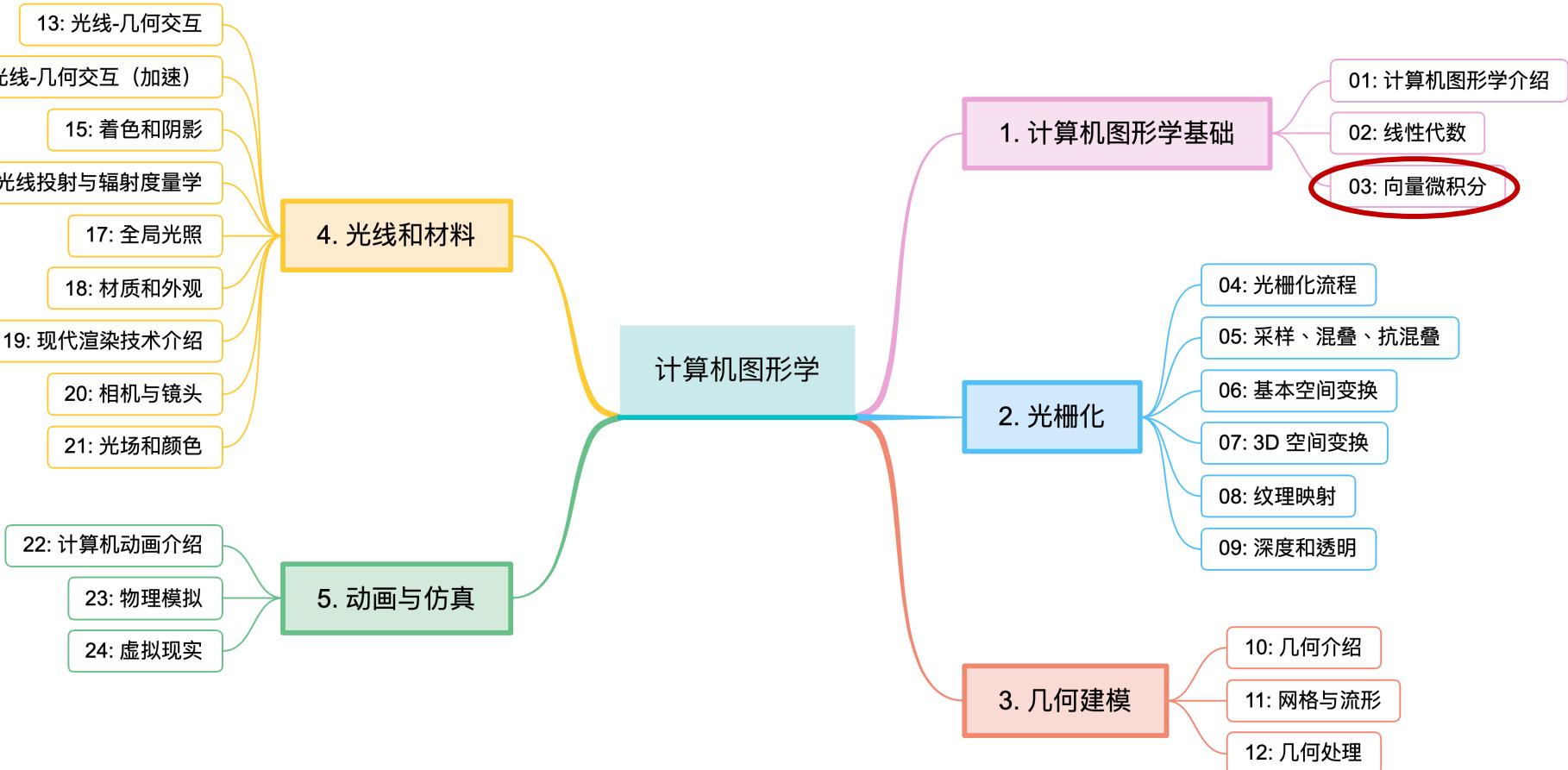
Lecture 03: 向量微积分

SSE315: 计算机图形学
Computer Graphics

陈壮彬

软件工程学院

chenzhb36@mail.sysu.edu.cn



Today's topics

□ 叉积 Cross product

□ 微分算子 Differential operators

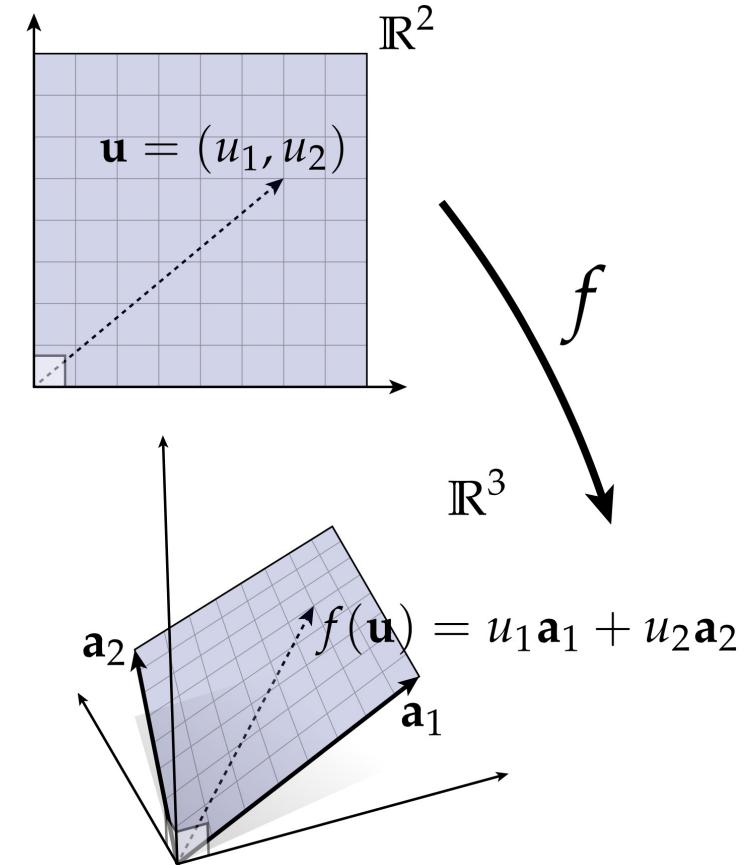
□ 向量场 Vector fields

上节课回顾：线性代数

□ 我们学习了很多内容

向量 & 向量空间
范数 / 内积
线性系统
线性映射
矩阵
...

□ 还有很多内容！



3Blue1Brown — Essence of Linear Algebra
Robert Ghrist — Calculus Blue

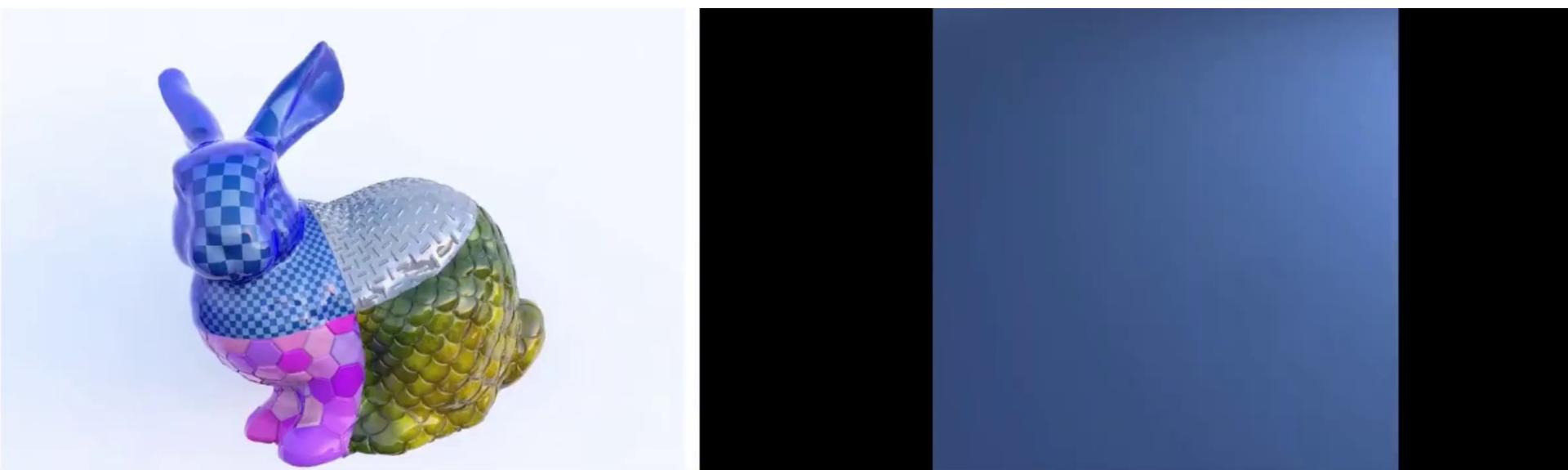
...

<https://www.bilibili.com/video/BV1u4411H7Ry/>
<https://www.bilibili.com/video/BV15E411k7VF/>

计算机图形学中的向量微积分

口为什么向量微积分对计算机图形学很重要？

- 是讨论**空间关系 (spatial relationship)**、**变换 (transformations)** 等的基本语言
- 许多现代图形学（基于物理的动画、几何处理等）是用**偏微分方程 (partial differential equations, PDE)** 表示的



欧几里德范数 Euclidean norm

口上一节课我们讨论了范数 (norm) 的概念，用于测量尺寸 size、长度 length、体积 volume、强度 intensity 等

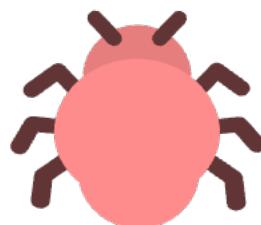
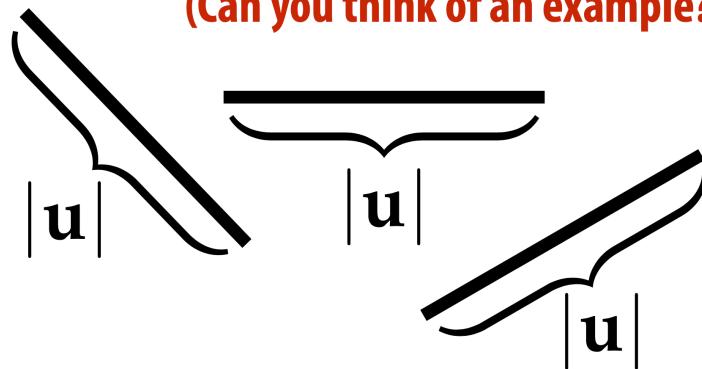
口对于几何计算，**欧几里得范数**是我们最关心的范数

口欧几里得范数是一个测量长度的方式，其特点是在空间进行旋转、平移或反射操作后，原有的长度信息仍被保留

口在正交基中，

$$|\mathbf{u}| := \sqrt{u_1^2 + \cdots + u_n^2}$$

Not true for all norms!
(Can you think of an example?)



WARNING: 除非在正交基上，否则 $|\mathbf{u}|$ 不具有几何长度的意义 (常见的 bug)

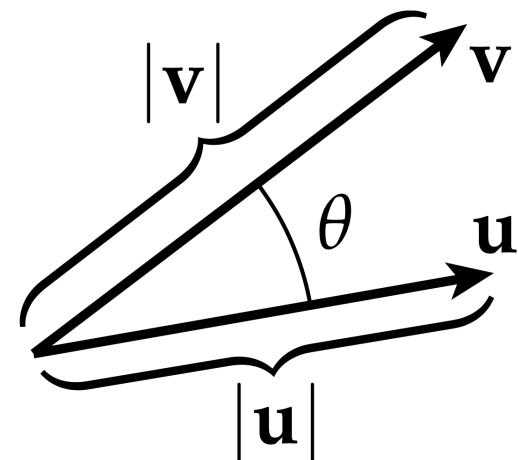
欧几里德内积/点积

口许多不同的内积直观上测量一些“对齐 alignment”的概念
口对于几何运算，希望通过内积来捕捉一些关于几何的信息
口对于 n 维向量，**欧几里得内积 (Euclidean inner product)** 定义为

$$\langle \mathbf{u}, \mathbf{v} \rangle := |\mathbf{u}| |\mathbf{v}| \cos(\theta)$$

口在正交笛卡尔坐标系中，内积可以通过**点积 (dot product)** 表示

$$\mathbf{u} \cdot \mathbf{v} := u_1 v_1 + \cdots + u_n v_n$$



WARNING: 与欧几里得范数一样，除非在正交基中，否则欧几里德内积没有几何意义

叉积 Cross product

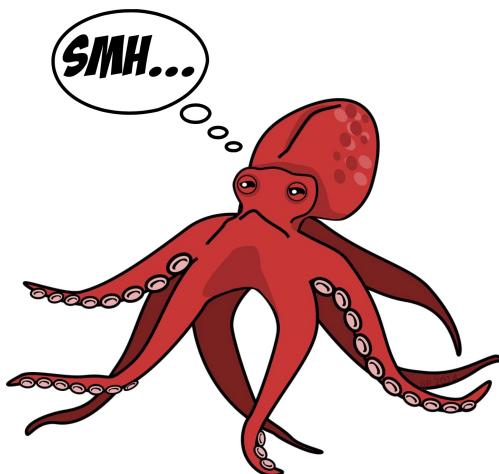
□ 内积取两个向量 (vector) 并产生一个**标量 (scalar)**

□ 在 3D 中，叉积取两个向量并得到一个**向量 (vector)**，
写为 $\mathbf{u} \times \mathbf{v}$

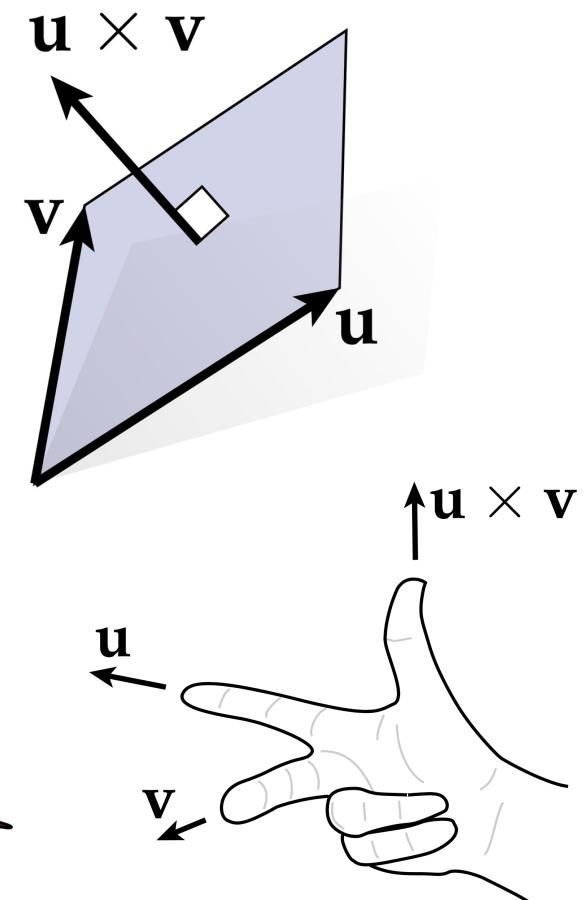
□ 在几何中， $\mathbf{u} \times \mathbf{v}$ 的

- **大小**等于平行四边形的面积
- **方向**垂直于两个向量
- 但具体是哪个方向呢？

□ “右手法则”



**Q: 为什么向量叉积
(通常) 仅在 3D 中
才有意义？**



叉积、行列式和角度

□ 更精确的定义 (不需要借助手) :

$$\sqrt{\det(\mathbf{u}, \mathbf{v}, \mathbf{u} \times \mathbf{v})} = |\mathbf{u}| |\mathbf{v}| \sin(\theta)$$

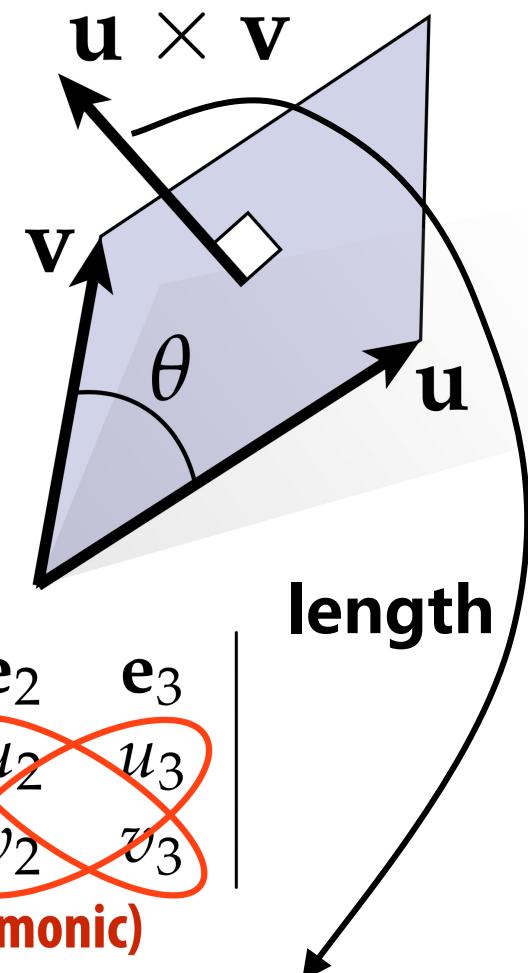
□ θ 是 \mathbf{u} 和 \mathbf{v} 之间的角度

□ \det 是三个列向量的行列式

□ 唯一确定叉积坐标的公式 (在正交基中):

$$\mathbf{u} \times \mathbf{v} := \begin{bmatrix} u_2 v_3 - u_3 v_2 \\ u_3 v_1 - u_1 v_3 \\ u_1 v_2 - u_2 v_1 \end{bmatrix}$$

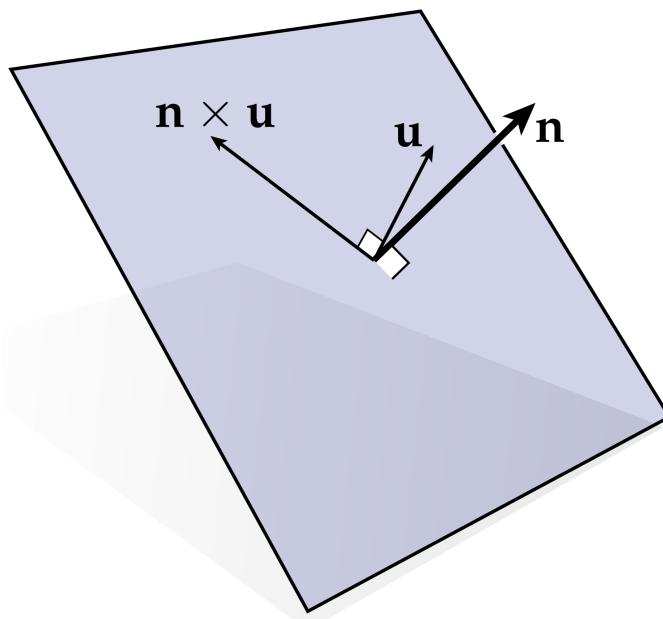
$$\begin{vmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix} \quad (\text{mnemonic})$$



□ 2D 中的表示 (符号滥用) : $\mathbf{u} \times \mathbf{v} := u_1 v_2 - u_2 v_1$

四分之一旋转的叉积

□ 在三维中操作向量，若向量 \mathbf{u} 与单位向量 \mathbf{n} 垂直，则 \mathbf{u} 与 \mathbf{n} 的叉积相当于在法线为 \mathbf{n} 的平面中的 90 度旋转



□ Q: $\mathbf{n} \times (\mathbf{n} \times \mathbf{u}) = ?$

□ Q: 给定 \mathbf{u} 和正交的单位向量 \mathbf{n} , 如何将 \mathbf{u} 绕 \mathbf{n} 旋转一个任意的角度 θ , 而不仅是 90 度?

点积的矩阵表示

□ 通过矩阵积来表示点积通常很方便 (比如写代码)

$$\mathbf{u} \cdot \mathbf{v} = \mathbf{u}^\top \mathbf{v} = \begin{bmatrix} u_1 & \cdots & u_n \end{bmatrix} \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \sum_{i=1}^n u_i v_i$$

□ 其他形式的内积呢?

□ 比如 $\langle \mathbf{u}, \mathbf{v} \rangle := 2 \mathbf{u}_1 \mathbf{v}_1 + \mathbf{u}_1 \mathbf{v}_2 + \mathbf{u}_2 \mathbf{v}_1 + 3 \mathbf{u}_2 \mathbf{v}_2$

$$\underbrace{\begin{bmatrix} u_1 & u_2 \end{bmatrix}}_{\mathbf{u}^\top} \underbrace{\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} v_1 \\ v_2 \end{bmatrix}}_{\mathbf{v}} = \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} 2v_1 + v_2 \\ v_1 + 3v_2 \end{bmatrix}$$
$$= (2u_1v_1 + u_1v_2) + (u_2v_1 + 3u_2v_2).$$

Q: 为什么代表内积的矩阵总是对称的 (即 $A^\top = A$) ?

叉积的矩阵表示

□ 我们同样可以用矩阵乘法来表示叉积

$$\mathbf{u} := (u_1, u_2, u_3) \Rightarrow \hat{\mathbf{u}} := \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix}$$

$$\mathbf{u} \times \mathbf{v} = \hat{\mathbf{u}}\mathbf{v} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix}$$

□ Q: 不构建新矩阵, 我们如何表示 $\mathbf{v} \times \mathbf{u}$?

□ A: 注意到 $\mathbf{v} \times \mathbf{u} = -\mathbf{u} \times \mathbf{v}$ (为什么?), 因此:

$$\mathbf{v} \times \mathbf{u} = -\hat{\mathbf{u}}\mathbf{v} = \hat{\mathbf{u}}^\top \mathbf{v}$$

行列式 Determinant

□Q: 如何计算一个矩阵的行列式?

$$\mathbf{A} := \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

□A: 应用一个很久以前有人告诉我的算法

$$\begin{bmatrix} a & b & c \\ d & \cancel{e} & \cancel{f} \\ g & \cancel{h} & \cancel{i} \end{bmatrix}$$

$$\begin{bmatrix} a & b & c \\ \cancel{d} & e & \cancel{f} \\ \cancel{g} & h & \cancel{i} \end{bmatrix}$$

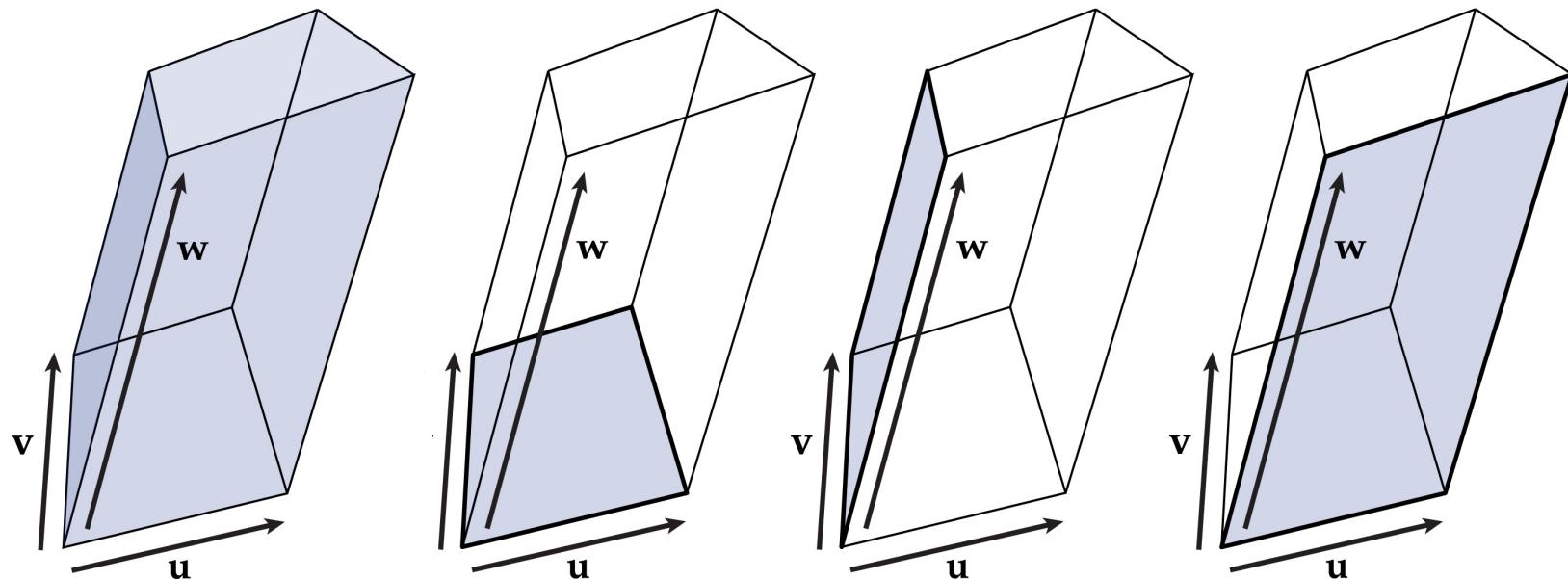
$$\begin{bmatrix} a & b & c \\ \cancel{d} & \cancel{e} & f \\ \cancel{g} & \cancel{h} & i \end{bmatrix}$$

$$\det(\mathbf{A}) = a(ei - fh) + b(fg - di) + c(dh - eg)$$

□这个数字到底是什么意思?

行列式、体积与三乘积

□更直观的答案是： $\det(\mathbf{u}, \mathbf{v}, \mathbf{w})$ 表示用向量 $\mathbf{u}, \mathbf{v}, \mathbf{w}$ 构成的平行六面体的（有符号的）体积



$$\det(\mathbf{u}, \mathbf{v}, \mathbf{w}) = (\mathbf{u} \times \mathbf{v}) \cdot \mathbf{w} = (\mathbf{v} \times \mathbf{w}) \cdot \mathbf{u} = (\mathbf{w} \times \mathbf{u}) \cdot \mathbf{v}$$

□被称为“三乘积公式 (triple product formula)”的关系

□Q: 如果我们颠倒叉积的顺序会怎么样?

线性映射的行列式

□如果一个矩阵 A 编码了一个线性映射 f , 那么 $\det(A)$ 有什么含义?

**首先需要回顾如何将线性
映射用矩阵表示!**

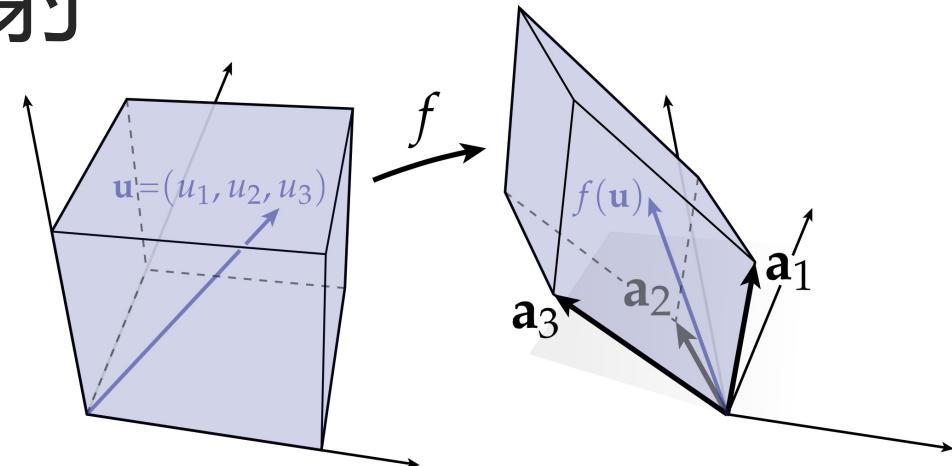
用矩阵表示线性映射

□ 假设我有如下线性映射

$$f(\mathbf{u}) = u_1 \mathbf{a}_1 + u_2 \mathbf{a}_2 + u_3 \mathbf{a}_3$$

□ Q：如何把它编码成矩阵？

□ A：直接将向量 \mathbf{a}_i 变成矩阵的列：



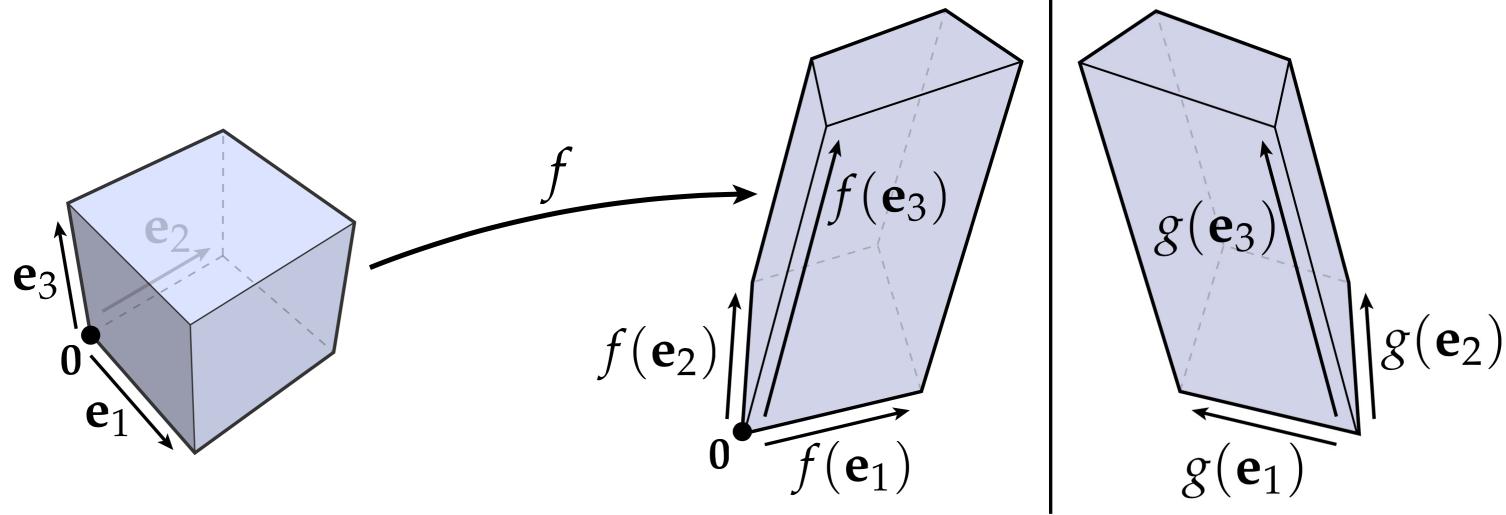
$$A := \begin{bmatrix} & & \\ \mathbf{a}_1 & \mathbf{a}_2 & \mathbf{a}_3 \\ & & \end{bmatrix} = \begin{bmatrix} a_{1,x} & a_{2,x} & a_{3,x} \\ a_{1,y} & a_{2,y} & a_{3,y} \\ a_{1,z} & a_{2,z} & a_{3,z} \end{bmatrix}$$

□ 原本的线性映射可以用矩阵乘法表示

$$A \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} = \begin{bmatrix} a_{1,x}u_1 + a_{2,x}u_2 + a_{3,x}u_3 \\ a_{1,y}u_1 + a_{2,y}u_2 + a_{3,y}u_3 \\ a_{1,z}u_1 + a_{2,z}u_2 + a_{3,z}u_3 \end{bmatrix} = u_1 \mathbf{a}_1 + u_2 \mathbf{a}_2 + u_3 \mathbf{a}_3$$

线性映射的行列式

□ 如果一个矩阵 A 编码了一个线性映射 f , 那么 $\det(A)$ 有什么含义?



□ A: 代表了**体积的变化**

□ Q: 在这种情况下, 行列式的符号 (sign) 告诉我们什么?

□ A: 它告诉我们方向是否反转 ($\det(A) < 0$)

其他的三重乘积 Triple products

□ Super useful for working w/ vectors in 3D.

□ E.g., **Jacobi identity** for the cross product:

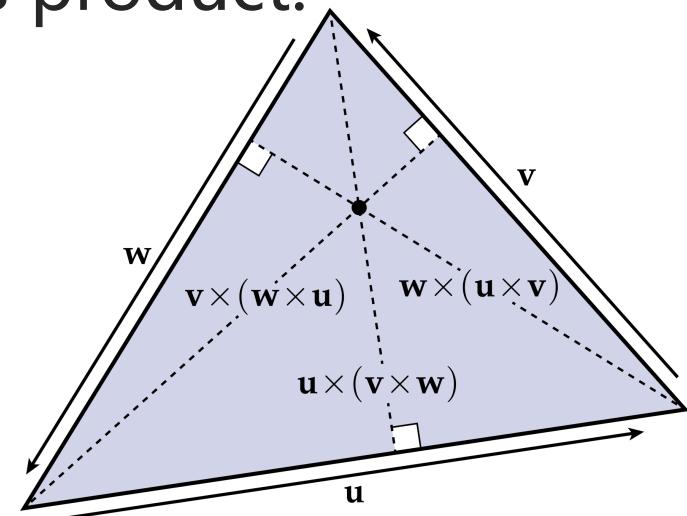
$$\begin{aligned}\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) &+ \\ \mathbf{v} \times (\mathbf{w} \times \mathbf{u}) &+ \\ \mathbf{w} \times (\mathbf{u} \times \mathbf{v}) &= 0\end{aligned}$$

□ 为什么它在几何上是正确的?

□ 有一个几何上的原因, 但并不像 \det 那么明显: 与三角形的三条高交汇于一点有关

□ 另一个三重乘积: **Lagrange' identify**

$$\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) = \mathbf{v}(\mathbf{u} \cdot \mathbf{w}) - \mathbf{w}(\mathbf{u} \cdot \mathbf{v})$$



Today's topics

□ 叉积 Cross product

□ 微分算子 Differential operators

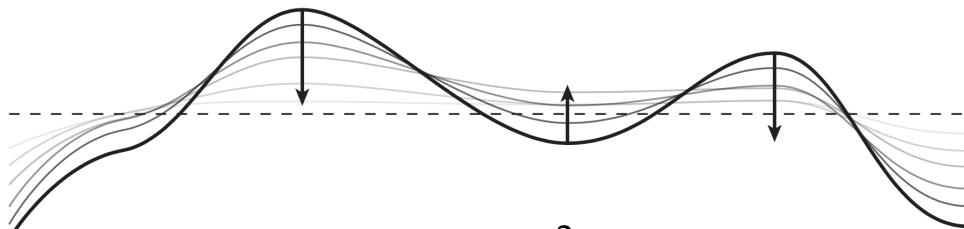
□ 向量场 Vector fields

微分算子 Differential operators

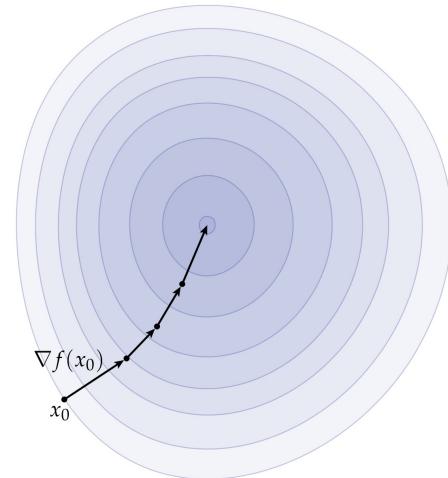
□ 接下来我们将学习**微分算子 Differential operators**

□ 为什么它们对计算机图形学有用？

- 许多物理/几何问题用**相对变化率**（**常微分方程 ODEs** 或 **偏微分方程 PDEs**）表示
- 这些工具还为**数值优化 (numerical optimization)** 提供了基础 - 例如，通过遵循某个目标的梯度进行最小化



$$\frac{d}{dt} \phi(x) = \frac{d^2}{dx^2} \phi(x)$$



导数 (derivative) 作为斜率

- 考虑一个函数 $f(x): R \rightarrow R$
- $f(x)$ 的导数 f' 的含义是什么？
- 一个解释是 “rise over run”， 斜率：

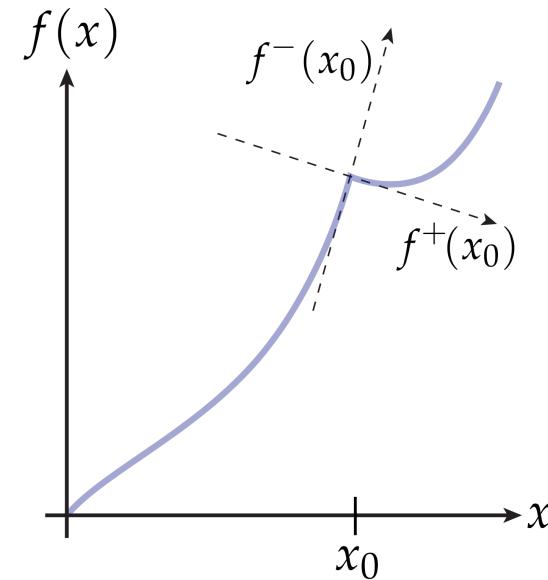
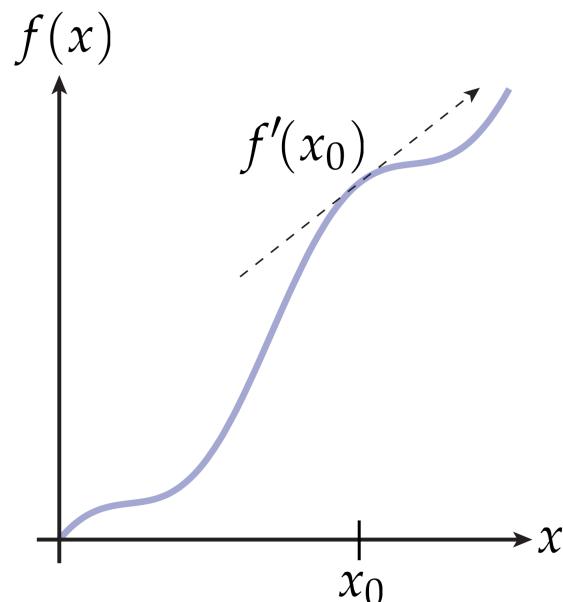
$$f'(x_0) := \lim_{\varepsilon \rightarrow 0} \frac{f(x_0 + \varepsilon) - f(x_0)}{\varepsilon}$$

- 如果不同方向的斜率不一样怎么办？

$$f^+(x_0) := \lim_{\varepsilon \rightarrow 0} \frac{f(x_0 + \varepsilon) - f(x_0)}{\varepsilon}$$

$$f^-(x_0) := \lim_{\varepsilon \rightarrow 0} \frac{f(x_0) - f(x_0 - \varepsilon)}{\varepsilon}$$

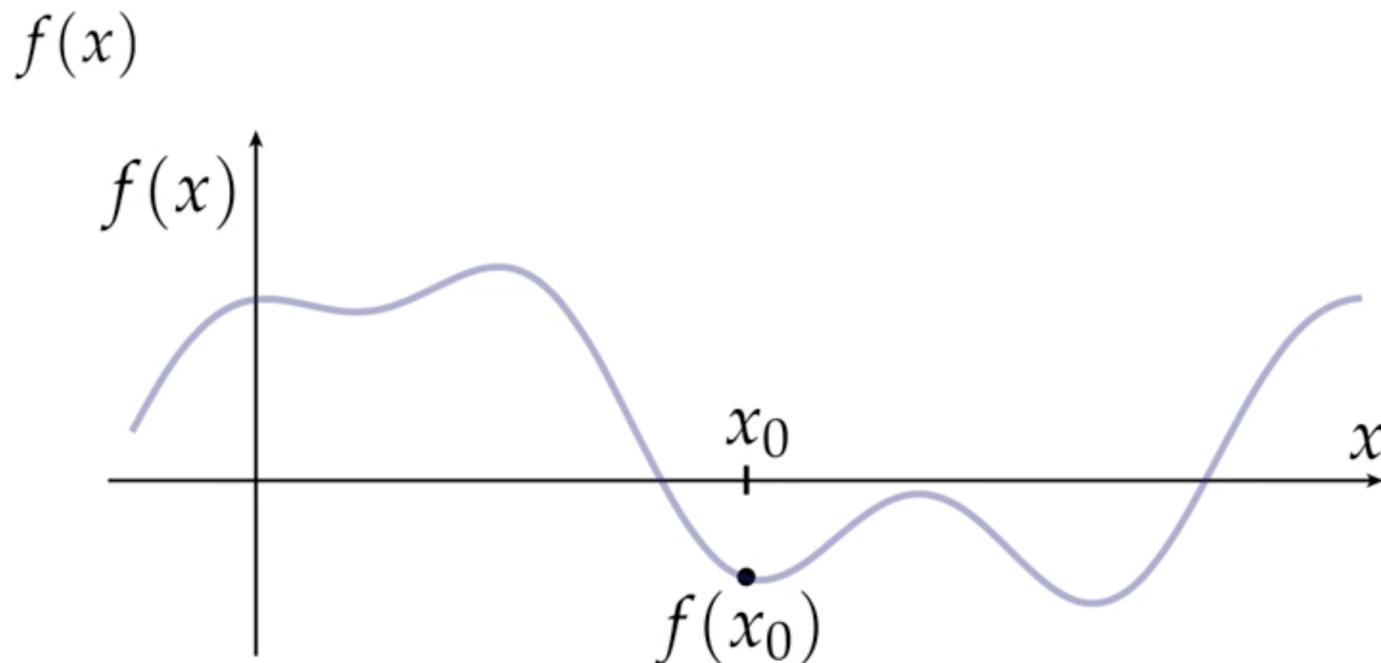
- 如果 $f^+ = f^-$ ， 则 f 在 x_0 处可导



Many functions in graphics are NOT differentiable!

导数作为最佳线性近似

任何光滑函数 $f(x)$ 都可表示为泰勒级数 (Taylor series)

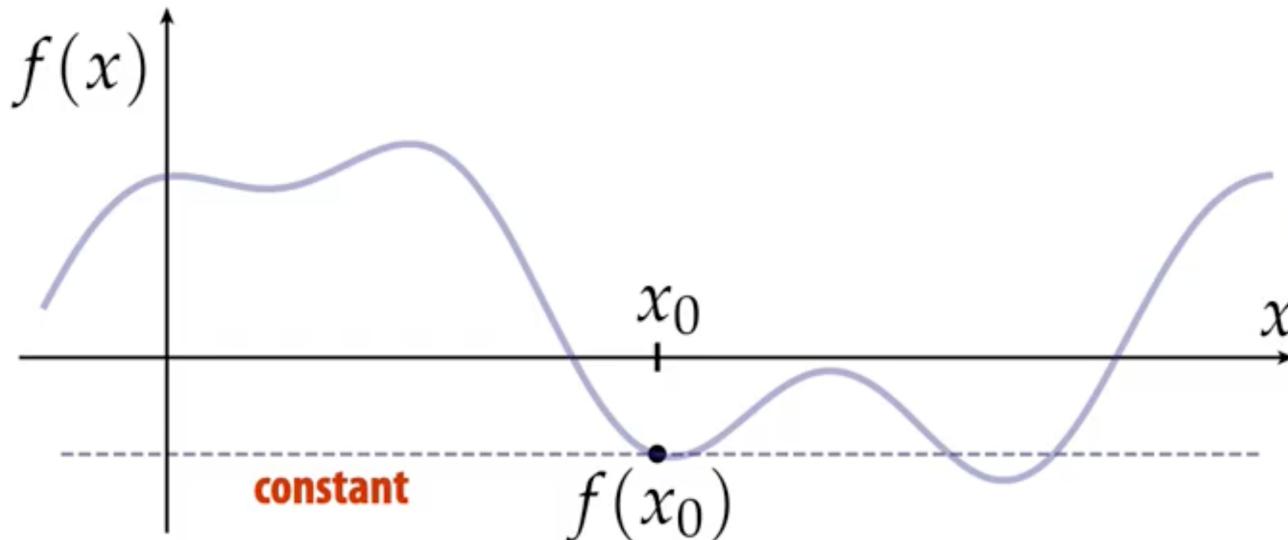


导数作为最佳线性近似

任何光滑函数 $f(x)$ 都可表示为泰勒级数 (Taylor series)

constant

$$f(x) = f(x_0)$$

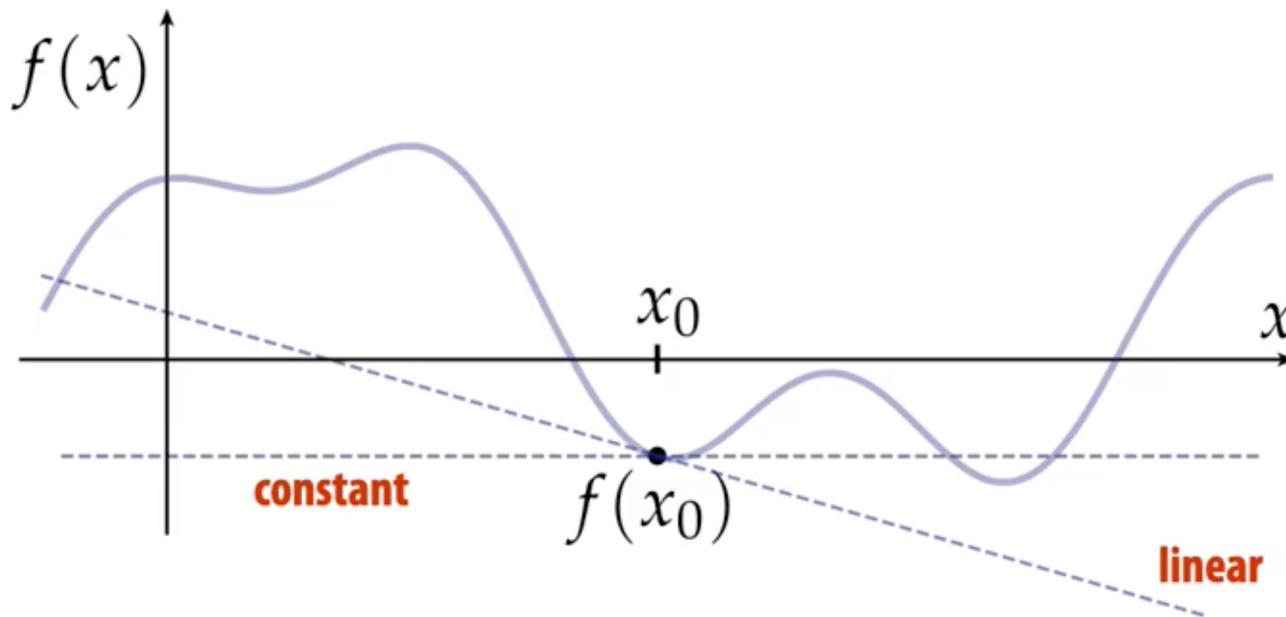


导数作为最佳线性近似

任何光滑函数 $f(x)$ 都可表示为泰勒级数 (Taylor series)

constant **linear**

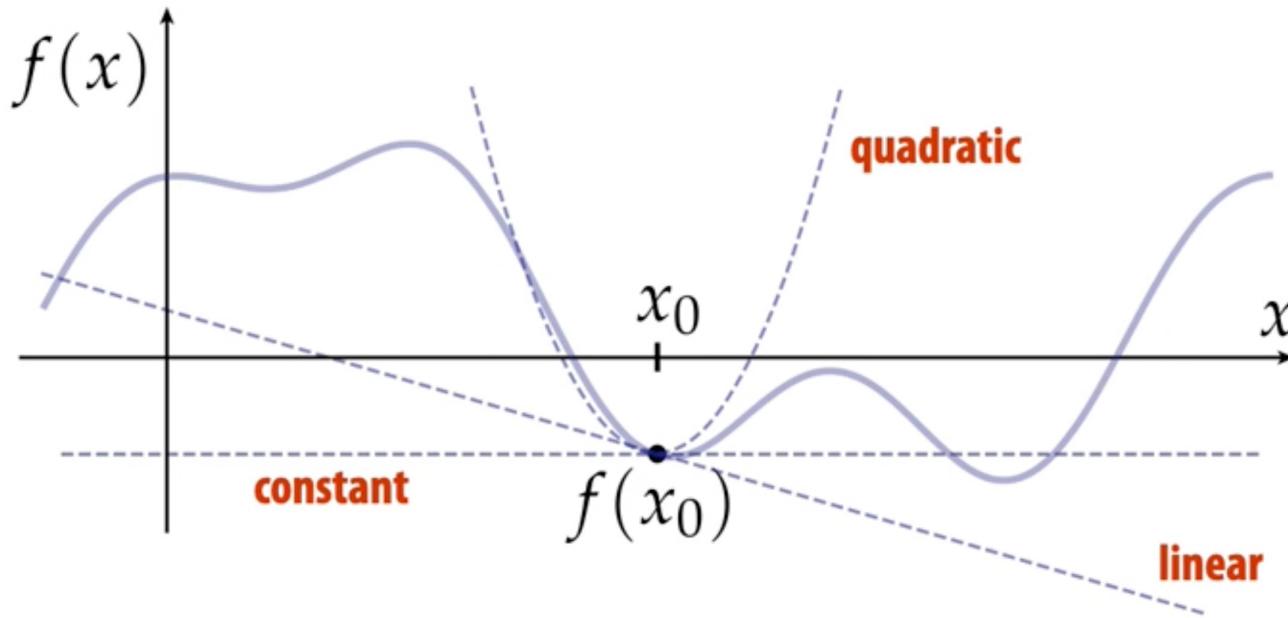
$$f(x) = f(x_0) + f'(x_0)(x - x_0)$$



导数作为最佳线性近似

任何光滑函数 $f(x)$ 都可表示为泰勒级数 (Taylor series)

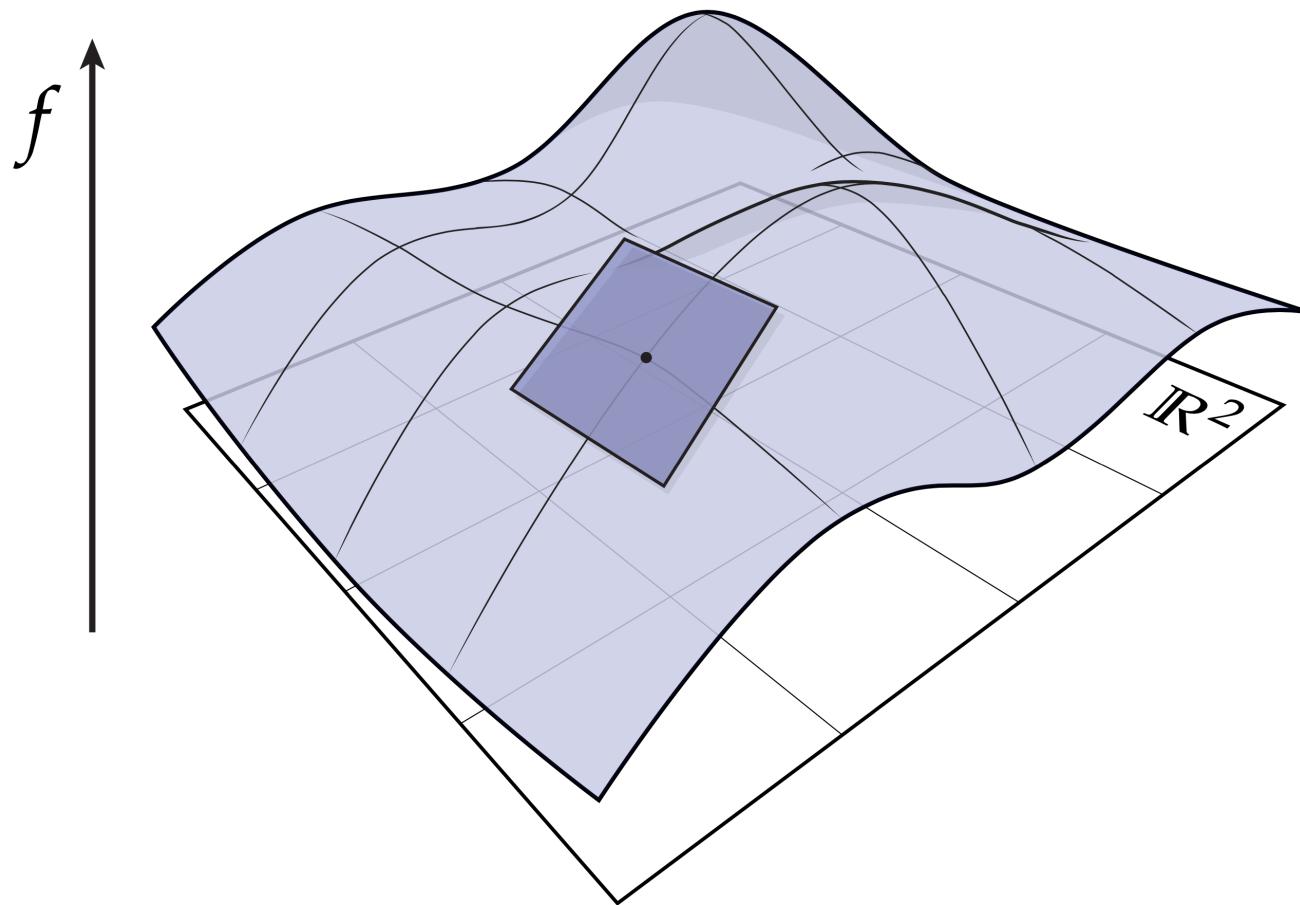
constant	linear	quadratic
$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{(x-x_0)^2}{2!}f''(x_0) + \dots$		



我们将会看到很多例子，用线性（有时是二次项）近似代替复杂函数是图形算法中的一个强大的技巧

导数作为最佳线性近似

口直觉上，同样的想法适用于多变量函数 (functions of multiple variables)

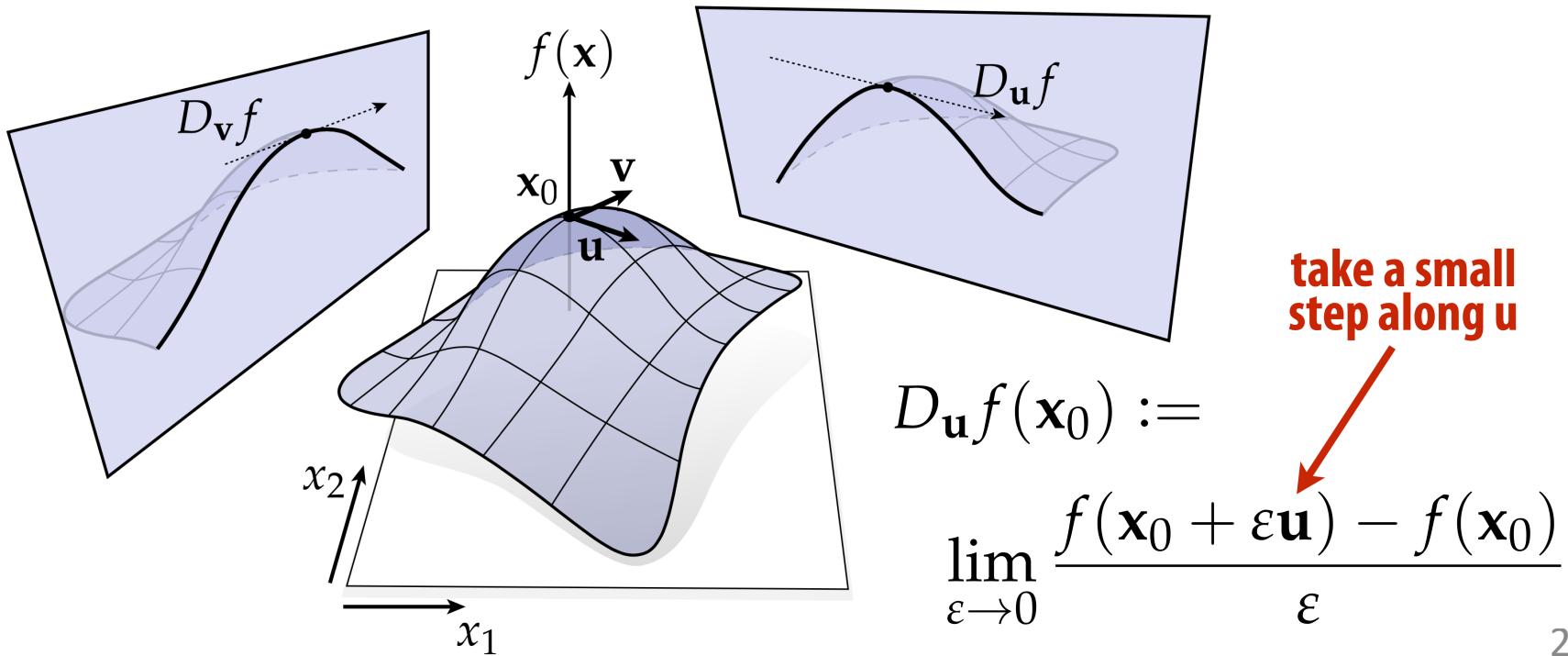


**如何考慮具有多个变量的
函数的导数？**

方向导数 Directional derivative

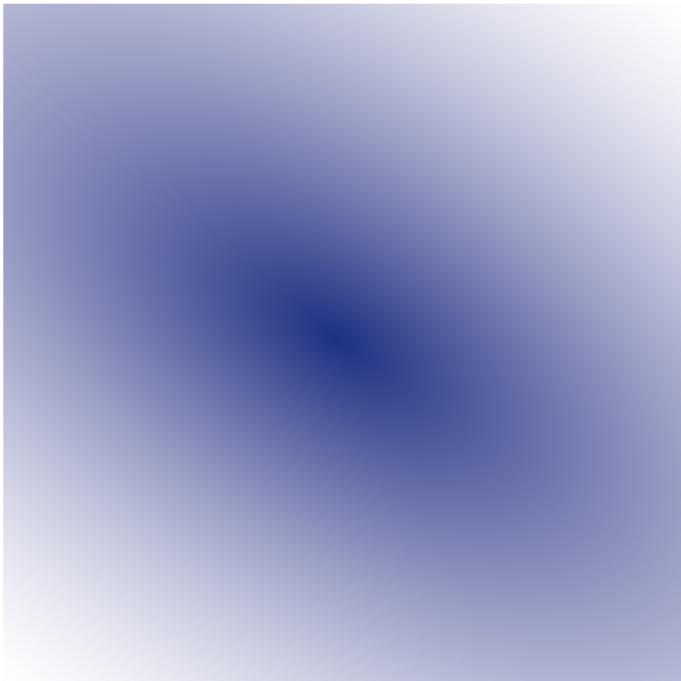
假设我们有一个函数 $f(x_1, x_2)$

- 沿着某条线对函数进行“切片”
- 应用一维中常见的导数
- 便得到方向导数 (directional derivative)

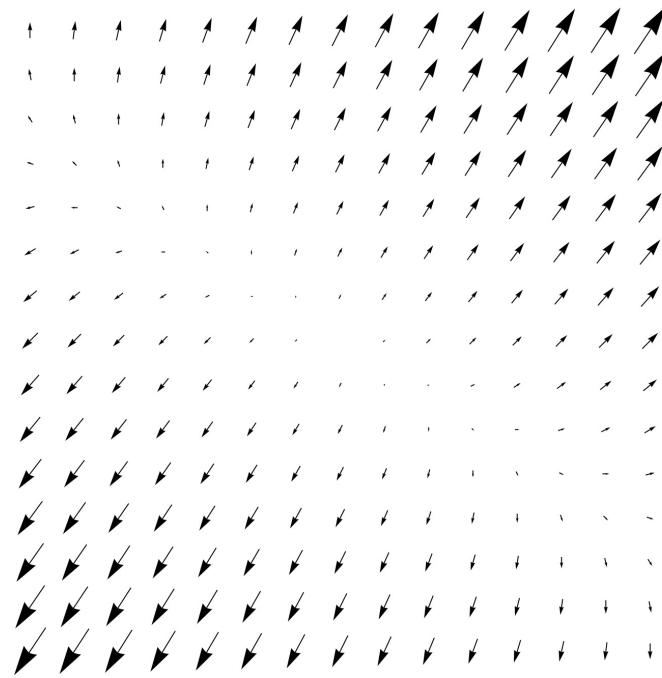


梯度 Gradient

□ 对于一个多变量函数 $f(\mathbf{x})$, 其**梯度** $\nabla f(\mathbf{x})$ 在每个点指定一个向量 (函数值增加最快的方向, 大小表达了函数值的变化率)



$$f(\mathbf{x})$$



$$\nabla f(\mathbf{x})$$

□ 这些梯度中的向量具体是哪些?

坐标系中的梯度

□ 最熟悉的定义：所有偏导数 (partial derivatives) 的列表

□ 依次假设除一个坐标外，所有坐标都是常数 (constant)，并取常规的导数

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

□ 这么理解梯度会有两个可能的问题

- 没有办法区分函数的函数 $F(f)$ ，因为我们没有坐标 x_1, x_2, \dots, x_n
- 内积的作用不清楚

□ 但是，这仍然是计算梯度的一个非常常用的办法

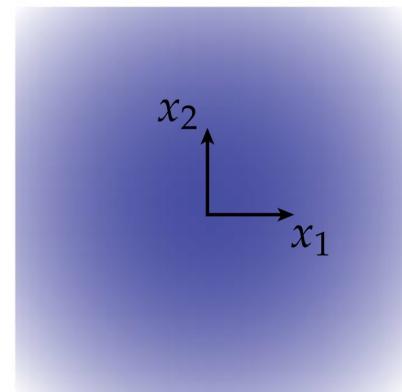
例子：坐标系中的梯度

$$f(\mathbf{x}) := x_1^2 + x_2^2$$

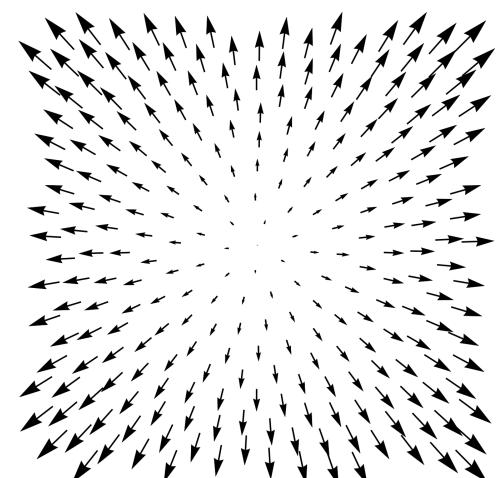
$$\frac{\partial f}{\partial x_1} = \frac{\partial}{\partial x_1} x_1^2 + \frac{\partial}{\partial x_1} x_2^2 = 2x_1 + 0$$

$$\frac{\partial f}{\partial x_2} = \frac{\partial}{\partial x_2} x_1^2 + \frac{\partial}{\partial x_2} x_2^2 = 0 + 2x_2$$

$$\nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = 2\mathbf{x}$$



$$f(\mathbf{x})$$

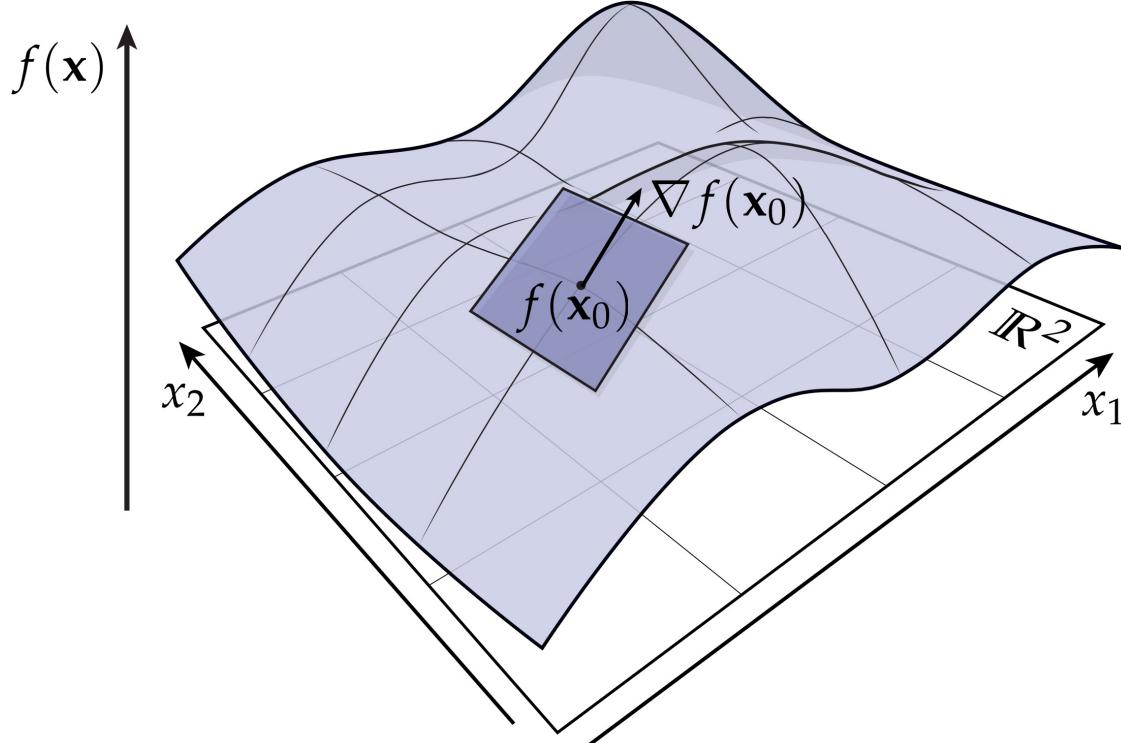


$$\nabla f(\mathbf{x})$$

梯度作为最佳线性近似

另一种看待梯度的角度是：在每个点 x_0 ，梯度 $\nabla f(x_0)$ 是实现最佳近似的向量

$$f(x) \approx f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle$$



Starting at x_0 , this term gets:

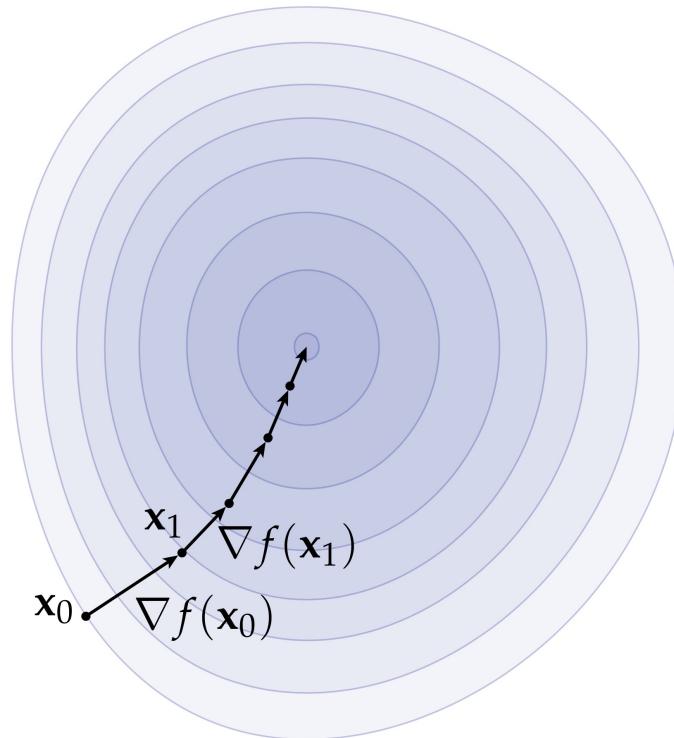
- bigger if we move in the direction of the gradient,
- smaller if we move in the opposite direction, and
- doesn't change if we move orthogonal to gradient.

梯度带你爬坡

口另一种看待梯度的角度：上升最快（最陡）的方向

口也就是说，为了尽快增加函数的值，应该朝梯度方向前进

口这是我们常用的优化算法的思想，也常用于图形学中



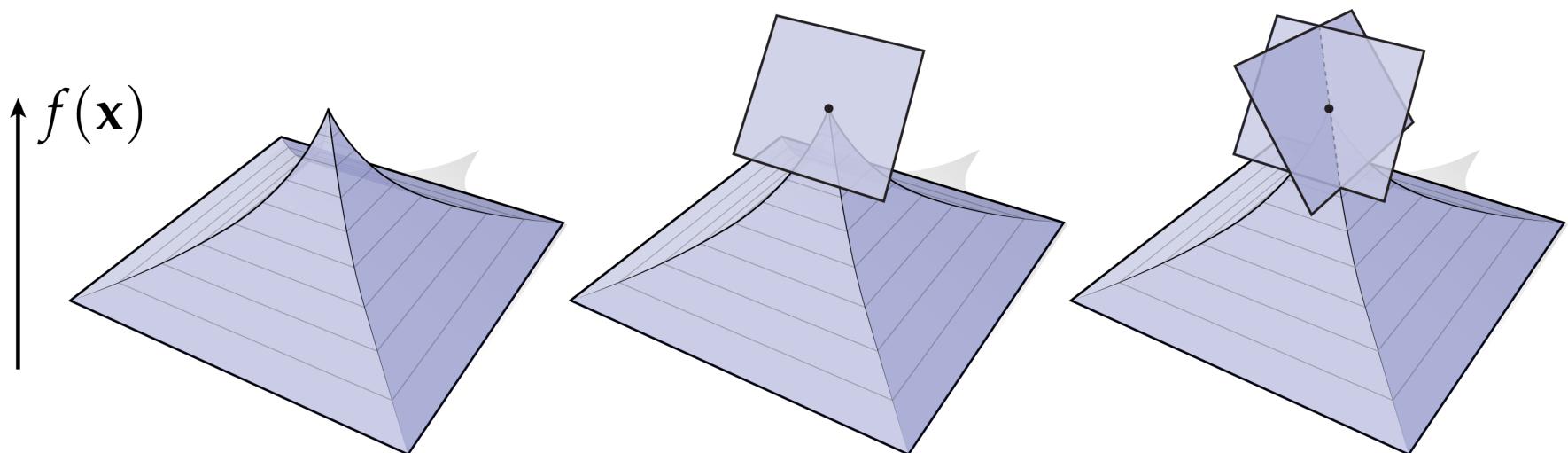
梯度和方向导数的关系

□ 在每一个点 x , 梯度是唯一的向量 $\nabla f(x)$, 使得

$$\langle \nabla f(x), \mathbf{u} \rangle = D_{\mathbf{u}} f(x) \text{ for all } \mathbf{u}$$

即, 计算方向 \mathbf{u} 与梯度的内积, 可以得到 \mathbf{u} 的方向导数
不需要逐个方向去算

对于不可微的函数不成立



例子：点积的梯度

□ 考虑用矩阵表示的点积

$$f := \mathbf{u}^\top \mathbf{v}$$

□ f 相对于 \mathbf{u} 的梯度是多少？

□ 一种方式是：用坐标写出来：

$$\mathbf{u}^\top \mathbf{v} = \sum_{i=1}^n u_i v_i$$

(equals zero unless $i = k$)

$$\frac{\partial}{\partial u_k} \sum_{i=1}^n u_i v_i = \sum_{i=1}^n \frac{\partial}{\partial u_k} (u_i v_i) = v_k$$

In other words:

$$\nabla_{\mathbf{u}} (\mathbf{u}^\top \mathbf{v}) = \mathbf{v}$$

$$\Rightarrow \nabla_{\mathbf{u}} f = \begin{bmatrix} v_1 \\ \dots \\ v_n \end{bmatrix}$$

Not so different from $\frac{d}{dx}(xy) = y$!

矩阵表达式的梯度

- 能够对矩阵表达式求微分，在计算机图形学中非常有用
- 最终，表达式看起来很像普通的导数

For any two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ and symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$:

MATRIX DERIVATIVE	LOOKS LIKE
$\nabla_{\mathbf{x}}(\mathbf{x}^T \mathbf{y}) = \mathbf{y}$	$\frac{d}{dx} xy = y$
$\nabla_{\mathbf{x}}(\mathbf{x}^T \mathbf{x}) = 2\mathbf{x}$	$\frac{d}{dx} x^2 = 2x$
$\nabla_{\mathbf{x}}(\mathbf{x}^T A \mathbf{y}) = A\mathbf{y}$	$\frac{d}{dx} axy = ay$
$\nabla_{\mathbf{x}}(\mathbf{x}^T A \mathbf{x}) = 2A\mathbf{x}$	$\frac{d}{dx} ax^2 = 2ax$
...	...

Excellent resource: Petersen & Pedersen, "The Matrix Cookbook"

函数的函数的梯度

□ 考虑一个函数的函数 $F(f)$, 即**范函 functional**

□ F 相对于 f 的梯度是什么?

□ 我们不能求偏导数, 不能分别对 x_1, x_2, \dots, x_n 进行求导

□ 相反地, 我们应该求一个函数 ∇F , 使得对于所有函数 u :

$$\langle\langle \nabla F, u \rangle\rangle = D_u F$$

□ 函数的函数的方向导数是什么?

□ 使用之前的极限:

$$D_u F(f) = \lim_{\varepsilon \rightarrow 0} \frac{F(f + \varepsilon u) - F(f)}{\varepsilon}$$

□ 一些例子能便于理解这些概念

范函数的计算机图形学例子

口路径选择 Path selection

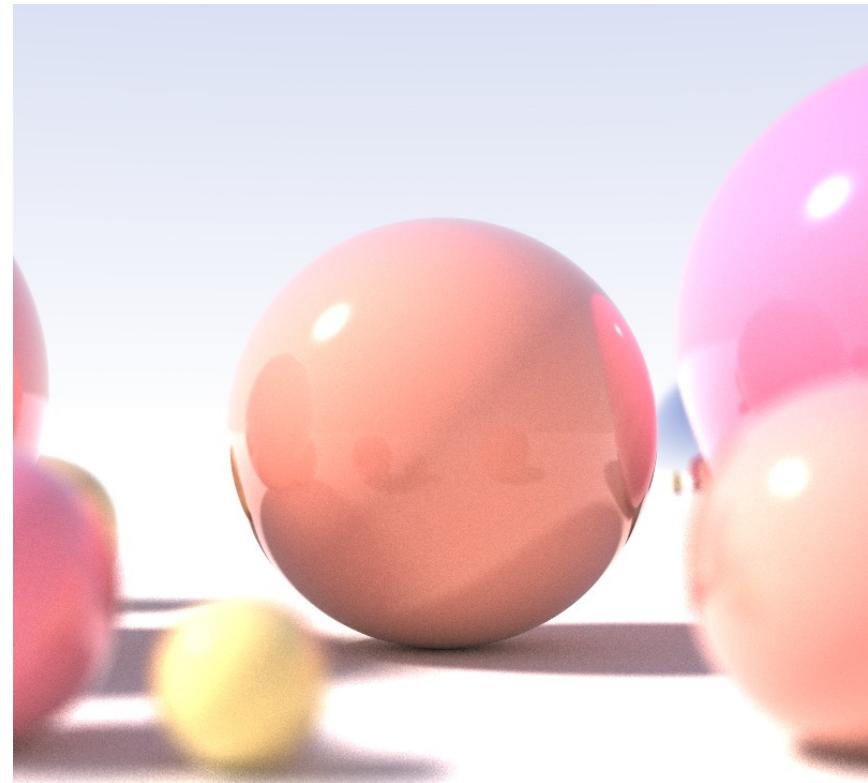
- 给定起点和终点，计算两点间的距离、用时等
- f : 路径函数
- F : 根据路径计算距离、用时



范函数的计算机图形学例子

口光线追踪 Ray tracing

- 通过追踪图像平面中像素的光线路径，并模拟其与虚拟对象的相遇效果来生成图像
- f : 光线路径函数
- F : 根据路径返回一个颜色



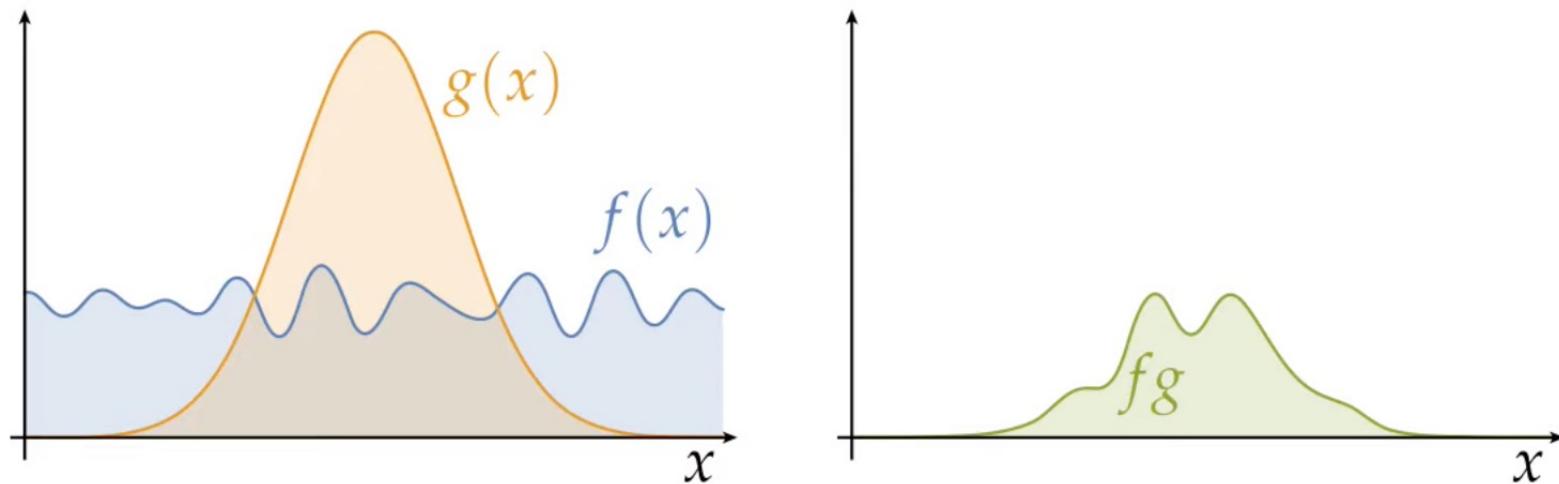
L^2 梯度：可视化的例子

□ 对于 $f, g: [0, 1] \rightarrow \mathbb{R}$, 考虑一个函数 $F(f, g) := \langle\langle f, g \rangle\rangle$

□ 我们猜测 $\nabla F = g$

□ 这凭直觉有意义吗？我们如何尽快地增加内积？

- 内积衡量函数“对齐”的程度
- g 是与 f 最佳对齐的函数



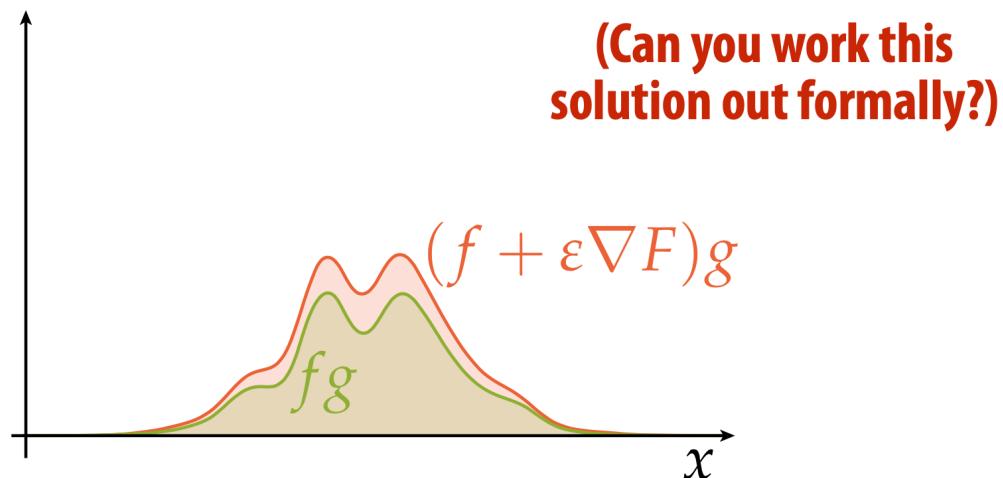
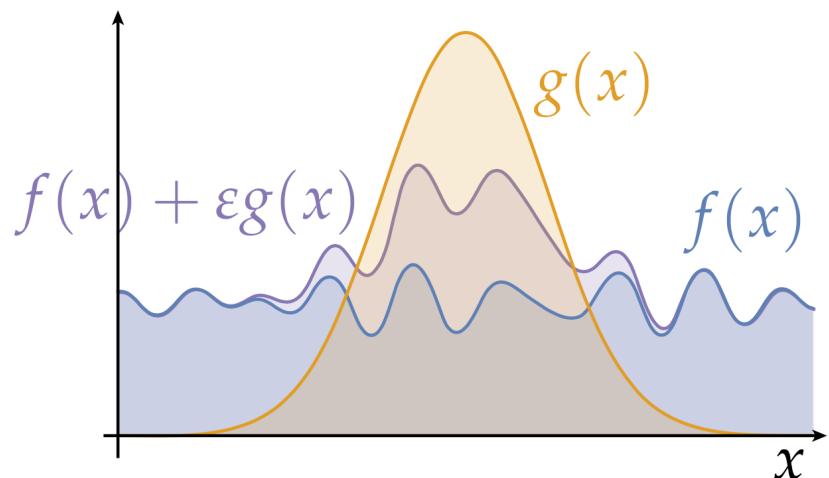
L^2 梯度：可视化的例子

□ 对于 $f, g: [0, 1] \rightarrow \mathbb{R}$, 考虑一个函数 $F(f, g) := \langle\langle f, g \rangle\rangle$

□ 我们猜测 $\nabla F = g$

□ 这凭直觉有意义吗？我们如何尽快地增加内积？

- 内积衡量函数“对齐”的程度
- g 是与 f 最佳对齐的函数
- 所以为了解释增加内积，我们只需在 f 上加一点 g



L^2 梯度：例子

□ 对于 $f, g: [0, 1] \rightarrow \mathbb{R}$, 考虑一个函数 $F(f) := \|f\|^2$

□ 对于一个“点” f_0 , 我们希望函数 ∇F 对于所有函数 u

$$\langle\langle \nabla F(f_0), u \rangle\rangle = \lim_{\varepsilon \rightarrow 0} \frac{F(f_0 + \varepsilon u) - F(f_0)}{\varepsilon}$$

□ 展开分子的第一项, 我们得到

$$\|f_0 + \varepsilon u\|^2 = \|f_0\|^2 + \varepsilon^2 \|u\|^2 + 2\varepsilon \langle\langle f_0, u \rangle\rangle$$

□ 因此, 极限变成

$$\lim_{\varepsilon \rightarrow 0} (\varepsilon \|u\|^2 + 2 \langle\langle f_0, u \rangle\rangle) = 2 \langle\langle f_0, u \rangle\rangle$$

□ 对所有的 u , 的唯一解为

$$\boxed{\nabla F(f_0) = 2f_0}$$

not much different from $\frac{d}{dx} x^2 = 2x!$

Key idea

一旦你掌握了普通函数的梯度，
对于矩阵、函数的函数等更奇特
的对象来说（表面上）就不难了

Today's topics

□ 叉积 Cross product

□ 微分算子 Differential operators

□ 向量场 Vector fields

向量场 Vector fields

□ 梯度是向量场的一个例子

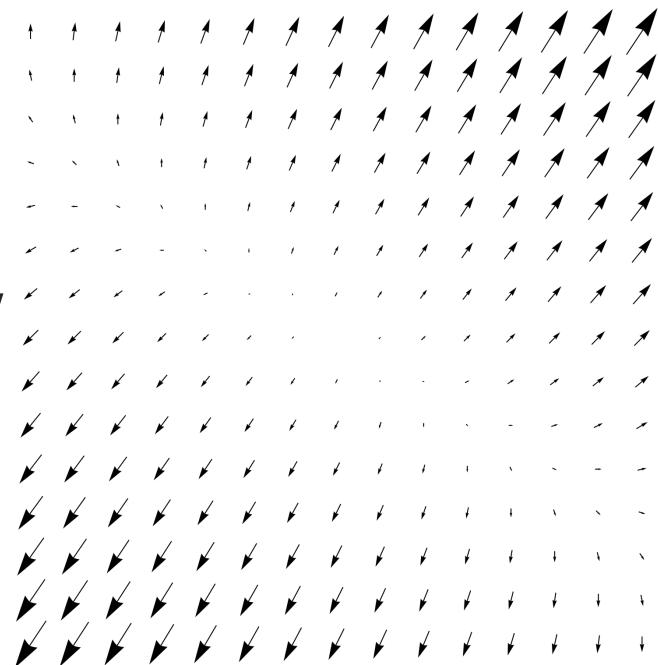
□ A vector field is a function that assigns a vector to each point in a subset of space

□ 例如，可以将平面中的二维向量场视为映射

$$X : \mathbb{R}^2 \rightarrow \mathbb{R}^2$$

□ 比如说，对于函数 $f(x, y) = x^2 + y^2$ ，
其梯度场为

$$\nabla f(x, y) = (2x, 2y)$$



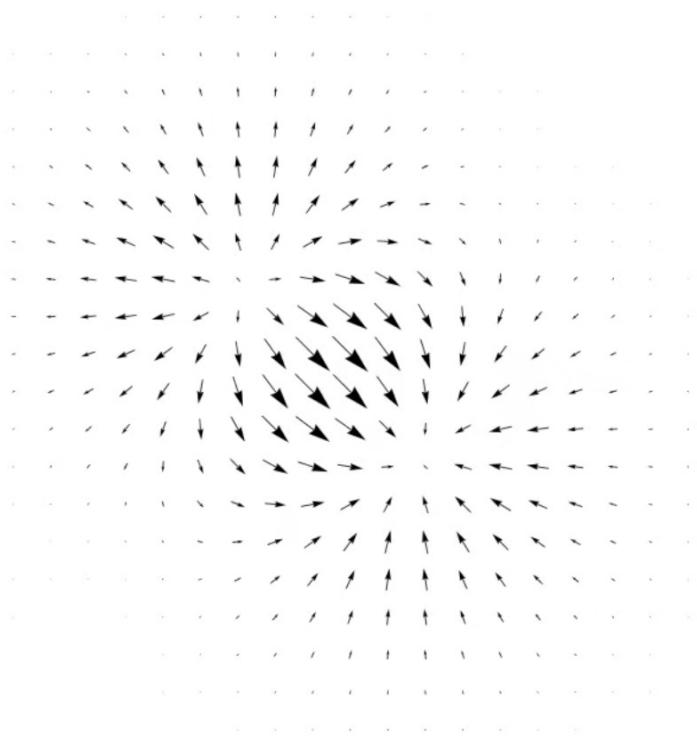
$$\nabla f(\mathbf{x})$$

**Q：我们如何测量向量场
的变化？**

散度和旋度 Divergence and Curl

口向量场的两个基本导数：

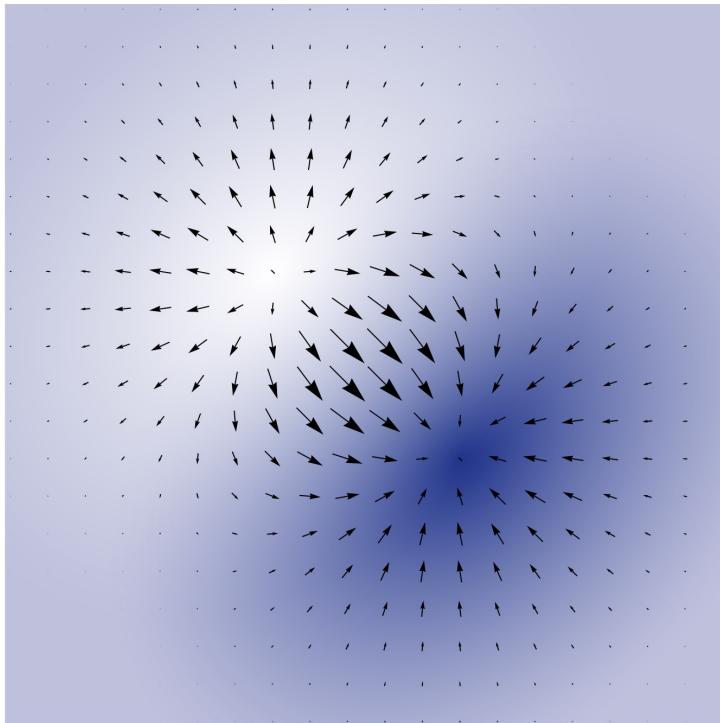
X



散度和旋度 Divergence and Curl

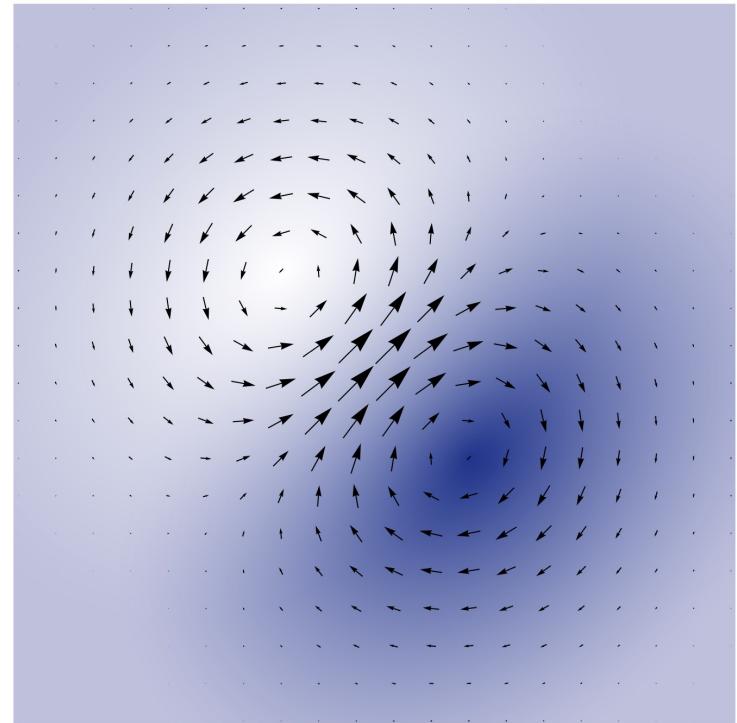
口向量场的两个基本导数：

向量场缩小/扩大了多少？



$\text{div } X$

向量场旋转了多少？



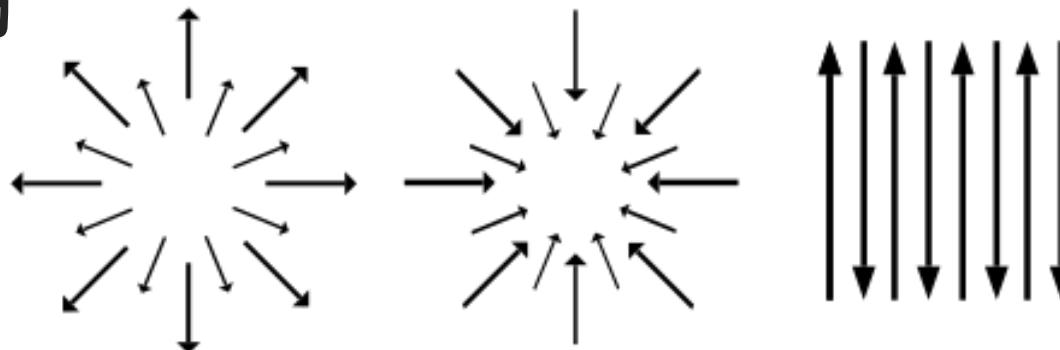
$\text{curl } Y$

散度 Divergence – 直观理解

正散度 (Positive Divergence): 在池塘的某一点放了一个水源，水从这个点不断涌出并向四周流开，代表了通量的净流出

负散度 (Negative Divergence): 你在某一点放了一个排水口，周围的水都向这个点汇集并消失，代表了通量的净流入

零散度 (Zero Divergence): 在一个既没有水源也没有排水口的区域，流进去的水和流出来的水一样多，比如在一条均匀流动的河里，任何一点的散度都为零。这种场被称为**无源场**或**不可压缩场**



$$\frac{\partial}{\partial x}(\mathbf{V}_x) > 0$$

$$\frac{\partial}{\partial y}(\mathbf{V}_y) > 0$$

$$\nabla \cdot (\mathbf{V}) > 0$$

$$\frac{\partial}{\partial x}(\mathbf{V}_x) < 0$$

$$\frac{\partial}{\partial y}(\mathbf{V}_y) < 0$$

$$\nabla \cdot (\mathbf{V}) < 0$$

$$\frac{\partial}{\partial x}(\mathbf{V}_x) = 0$$

$$\frac{\partial}{\partial y}(\mathbf{V}_y) = 0$$

$$\nabla \cdot (\mathbf{V}) = 0$$

散度 Divergence

□ 常写作 $\nabla \cdot X$

□ 它提供了一个散度的坐标定义

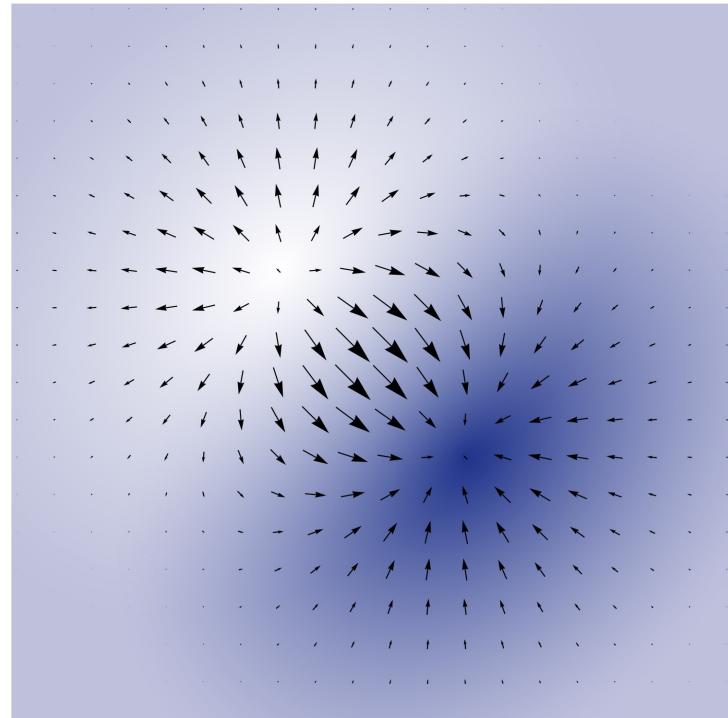
□ 把 ∇ 当作是梯度向量 (vector of derivatives)

$$\nabla = \left(\frac{\partial}{\partial u_1}, \dots, \frac{\partial}{\partial u_n} \right)$$

□ 把 X 当作是函数向量 (vector of functions)

$$X(\mathbf{u}) = (X_1(\mathbf{u}), \dots, X_n(\mathbf{u}))$$

□ 那么散度为 $\nabla \cdot X := \sum_{i=1}^n \partial X_i / \partial u_i$



散度实际上是函数在各个方向的变化率之和

$$\nabla \cdot X$$

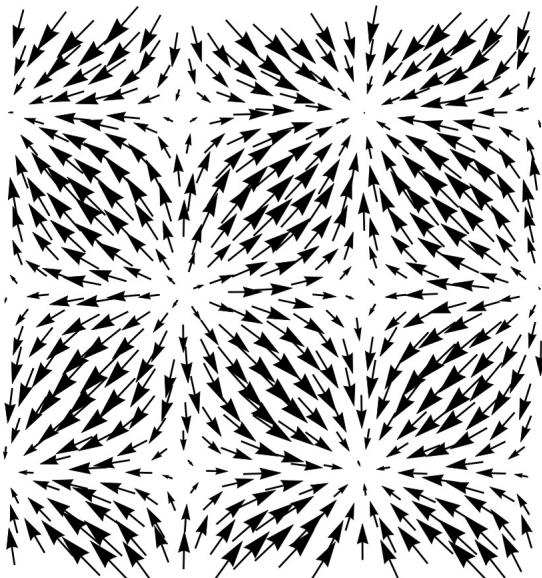
散度 - 例子

□ 考慮向量場 $X(u, v) := (\cos(u), \sin(v))$

Both are a scalar

□ 其散度為

$$\nabla \cdot X = \frac{\partial}{\partial u} \cos(u) + \frac{\partial}{\partial v} \sin(v) = -\sin(u) + \cos(v)$$



X



$\nabla \cdot X$

旋度 Curl – 直观理解

□ 在水流中的任意一点，放一个极小的、可自由旋转的小桨轮

□ **非零旋度 (Non-zero Curl)**：如果这个小桨轮开始旋转，那么该点的旋度就不为零

- **旋度的大小**：代表桨轮旋转的速度
- **旋度的方向**：代表桨轮的**旋转轴**（遵循右手法则：四指指向旋转方向，大拇指指向旋度的方向）

□ **零旋度 (Zero Curl)**：如果小桨轮只会被水流平推着走，自身不发生旋转，那么该点的旋度就为零

- 这种场被称为**无旋场或保守场 (conservative field)**

旋度 Curl

□ 常写作 $\nabla \times X$

□ 它提供了一个旋度的坐标定义

□ 把 ∇ 当作是一个包含三个梯度的向量

$$\nabla = \left(\frac{\partial}{\partial u_1}, \frac{\partial}{\partial u_2}, \frac{\partial}{\partial u_3} \right)$$

□ 把 X 当作是一个包含三个函数的向量

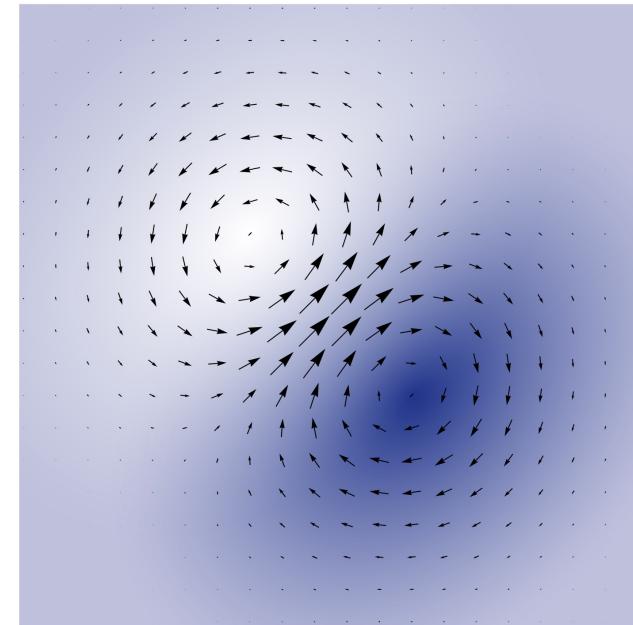
$$X(\mathbf{u}) = (X_1(\mathbf{u}), X_2(\mathbf{u}), X_3(\mathbf{u}))$$

□ 那么旋度为

$$\nabla \times X := \begin{bmatrix} \partial X_3 / \partial u_2 - \partial X_2 / \partial u_3 \\ \partial X_1 / \partial u_3 - \partial X_3 / \partial u_1 \\ \partial X_2 / \partial u_1 - \partial X_1 / \partial u_2 \end{bmatrix}$$

(2D “curl”: $\nabla \times X := \partial X_2 / \partial u_1 - \partial X_1 / \partial u_2$)

旋度是函数在各个方向变化率的差异产生的



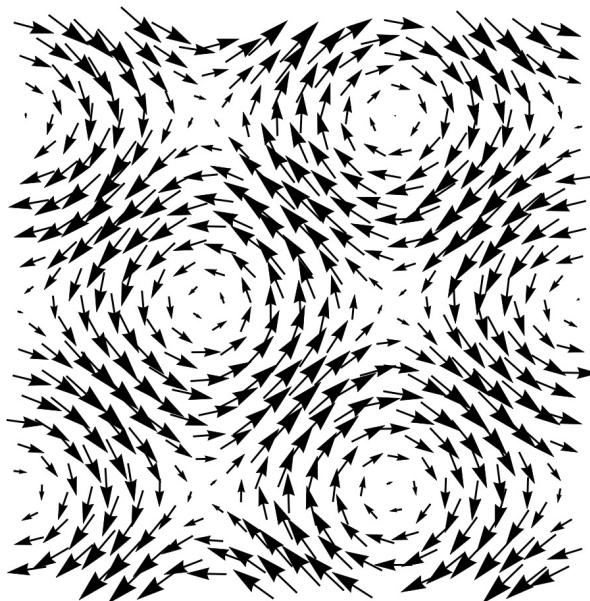
$$\nabla \times X$$

旋度 - 例子

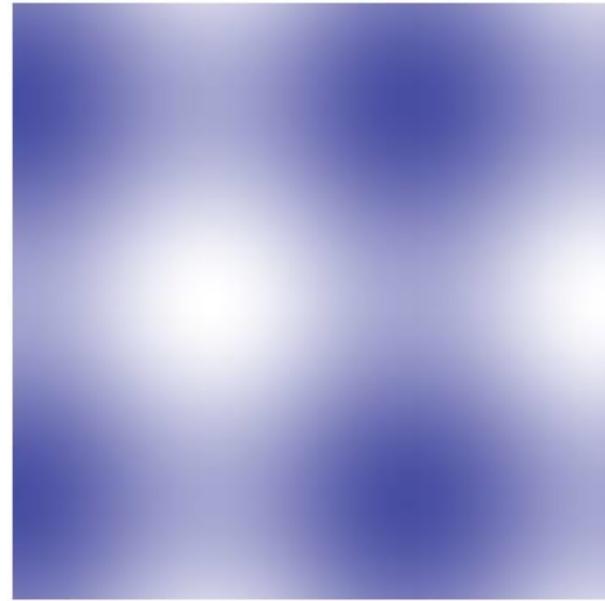
□ 考虑一个向量场 $X(u, v) := (-\sin(v), \cos(u))$

□ (2D) 旋度则为

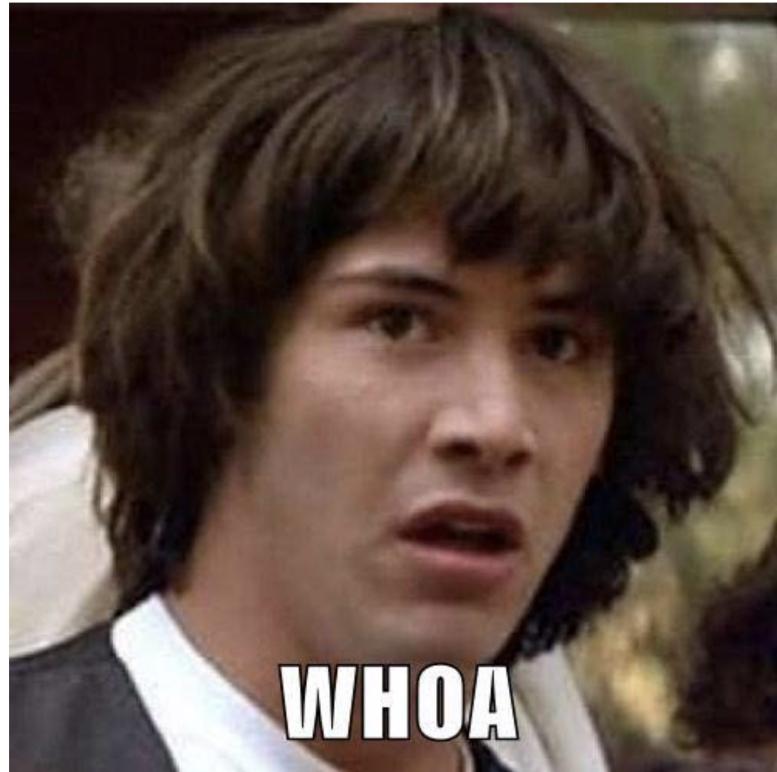
$$\nabla \times X = \frac{\partial}{\partial u} \cos(u) - \frac{\partial}{\partial v} (-\sin(v)) = -\sin(u) + \cos(v)$$



X



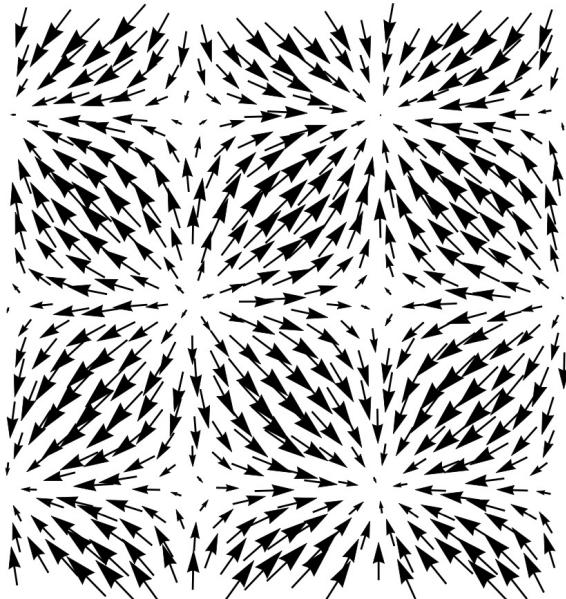
$\nabla \times X$



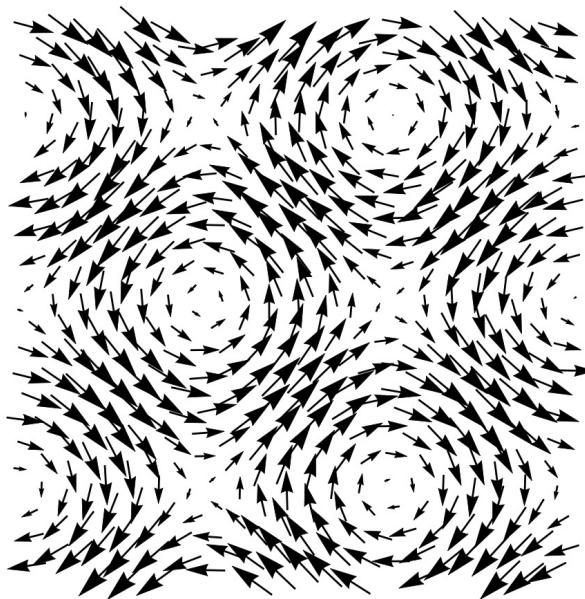
注意到旋度和散度之间的关系了吗？

散度 vs. 旋度 (2D)

□ X 的散度与 X 旋转 90 度的旋度相同：



X



X^\perp

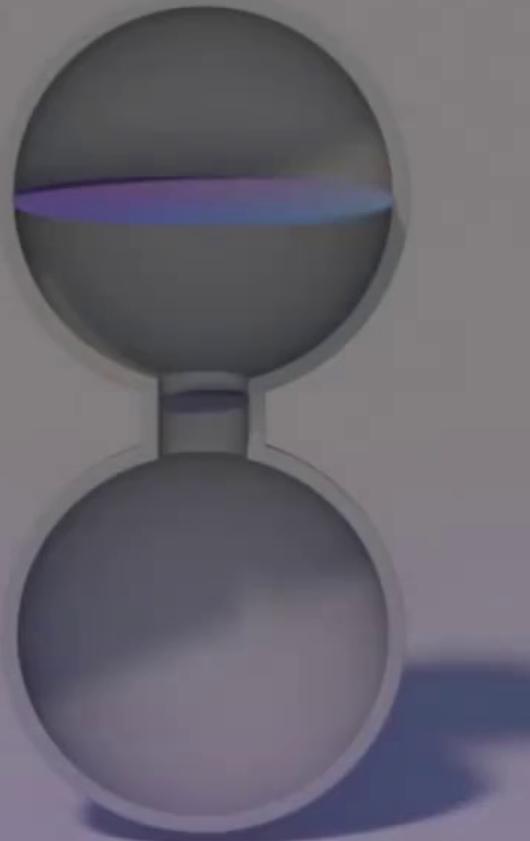


$$\nabla \cdot X = \nabla \times X^\perp$$

□ 基于向量场的各类操作在各类图形学算法中扮演重要的角色（比如说模拟液体）

例子：使用流函数模拟液体

Our method



Single-phase
Pressure solver



$$\min_{\Psi} ||u^* - \nabla \times \Psi||^2$$

$$u = \nabla \times \Psi$$

$$\Delta p = \nabla \cdot u^*$$

$$u = u^* - \nabla p$$



中山大學 软件工程学院
SUN YAT-SEN UNIVERSITY SCHOOL OF SOFTWARE ENGINEERING

谢谢

陈壮彬
软件工程学院
chenzhb36@mail.sysu.edu.cn