Math (P)Review Part II:
Vector Calculus

Computer Graphics
CMU 15-462/662
Last Time: Linear Algebra

- Touched on a variety of topics:
  - vectors & vector spaces
  - norm
  - $L^2$ norm/inner product
  - span
  - Gram-Schmidt
  - linear systems
  - quadratic forms
  - vectors as functions
  - inner product
  - linear maps
  - basis
  - frequency decomposition
  - bilinear forms
  - matrices
  - ...

- Don’t have time to cover everything!

- But there are some fantastic lectures online:
  - 3Blue1Brown — Essence of Linear Algebra
  - Robert Ghrist — Calculus Blue
  - ...
  - (Let us know about others online!)
Vector Calculus in Computer Graphics

- Today’s topic: *vector calculus*.

- Why is vector calculus important for computer graphics?
  - Basic language for talking about spatial relationships, transformations, etc.
  - Much of modern graphics (physically-based animation, geometry processing, etc.) formulated in terms of *partial differential equations* (PDEs) that use div, curl, Laplacian...
  - As we saw last time, vector-valued data is everywhere in graphics!
Euclidean Norm

- Last time, developed idea of norm, which measures total size, length, volume, intensity, etc.
- For geometric calculations, the norm we most often care about is the Euclidean norm.
- Euclidean norm is any notion of length preserved by rotations/translations/reflections of space.
- In orthonormal coordinates:

\[ |u| := \sqrt{u_1^2 + \cdots + u_n^2} \]

WARNING: This quantity does not encode geometric length unless vectors are encoded in an orthonormal basis. (Common source of bugs!)
Euclidean Inner Product / Dot Product

- Likewise, lots of possible inner products—intuitively, measure some notion of “alignment.”

- For geometric calculations, want to use inner product that captures something about geometry!

- For n-dimensional vectors, Euclidean inner product defined as

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

- In orthonormal Cartesian coordinates, can be represented via the dot product

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

- **WARNING**: As with Euclidean norm, no geometric meaning unless coordinates come from an orthonormal basis.
Cross Product

- Inner product takes two vectors and produces a scalar
- In 3D, cross product is a natural way to take two vectors and get a vector, written as “u x v”

- Geometrically:
  - magnitude equal to parallelogram area
  - direction orthogonal to both vectors
  - …but which way?
- Use “right hand rule”

(Q: Why only 3D?)
Cross Product, Determinant, and Angle

- More precise definition (that does not require hands):

\[ \sqrt{\det(u, v, u \times v)} = |u||v| \sin(\theta) \]

- \( \theta \) is angle between \( u \) and \( v \)
- "det" is determinant of three column vectors
- Uniquely determines coordinate formula:

\[
\begin{bmatrix}
  u_2v_3 - u_3v_2 \\
  u_3v_1 - u_1v_3 \\
  u_1v_2 - u_2v_1
\end{bmatrix}
\]

- Useful abuse of notation in 2D: \( u \times v \) := \( u_1v_2 - u_2v_1 \)
Cross Product as Quarter Rotation

- Simple but useful observation for manipulating vectors in 3D: cross product with a unit vector $N$ is equivalent to a quarter-rotation in the plane with normal $N$:

- $Q$: What is $N \times (N \times u)$?

- $Q$: If you have $u$ and $N \times u$, how do you get a rotation by some arbitrary angle $\theta$?
Matrix Representation of Dot Product

- Often convenient to express dot product via matrix product:

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

- By the way, what about some other inner product?
  - E.g., \( \langle \mathbf{u}, \mathbf{v} \rangle := 2u_1v_1 + u_1v_2 + u_2v_1 + 3u_2v_2 \)

\[
\begin{bmatrix} u_1 \\ u_2 \end{bmatrix}^\top \begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} u_1 & u_2 \end{bmatrix} \begin{bmatrix} 2v_1 + v_2 \\ v_1 + 3v_2 \end{bmatrix}
\]

\[= (2u_1v_1 + u_1v_2) + (u_2v_1 + 3u_2v_2). \quad \checkmark \]

Q: Why is matrix representing inner product always symmetric \((\mathbf{A}^\top = \mathbf{A})\)?
Matrix Representation of Cross Product

- Can also represent cross product via matrix multiplication:

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

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

- Q: Without building a new matrix, how can we express \( \mathbf{v} \times \mathbf{u} \)?
- A: Useful to notice that \( \mathbf{v} \times \mathbf{u} = -\mathbf{u} \times \mathbf{v} \) (why?). Hence,

\[ \mathbf{v} \times \mathbf{u} = -\hat{\mathbf{u}} \mathbf{v} = \hat{\mathbf{u}}^\top \mathbf{v} \]
Determinant

Q: How do you compute the determinant of a matrix?

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

A: Apply some algorithm somebody told me once upon a time:

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

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

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

\[
\det(A) = a(ei - fh) + b(fg - di) + c(dh - eg)
\]

Totally obvious… right?

Q: No! What the heck does this number mean?!
Better answer: \( \det(u, v, w) \) encodes (signed) volume of parallelepiped with edge vectors \( u, v, w \).

\[
\det(u, v, w) = (u \times v) \cdot w = (v \times w) \cdot u = (w \times u) \cdot v
\]

Relationship known as a “triple product formula”

(Q: What happens if we reverse order of cross product?)
Determinant of a Linear Map

Q: If a matrix $A$ encodes a linear map $f$, what does $\text{det}(A)$ mean?

(First: need to recall how a matrix encodes a linear map!)
Representing Linear Maps via Matrices

- Key example: suppose I have a linear map

\[ f(u) = u_1 a_1 + u_2 a_2 + u_3 a_3 \]

- How do I encode as a matrix?

- Easy: “a” vectors become matrix columns:

\[
A := \begin{bmatrix}
    a_1 & a_2 & a_3 \\
\end{bmatrix} = \begin{bmatrix}
    a_{1,x} & a_{2,x} & a_{3,x} \\
    a_{1,y} & a_{2,y} & a_{3,y} \\
    a_{1,z} & a_{2,z} & a_{3,z}
\end{bmatrix}
\]

- Now, matrix-vector multiply recovers original map:

\[
A \begin{bmatrix}
    u_1 \\
    u_2 \\
    u_3
\end{bmatrix} = \begin{bmatrix}
    a_{1,x}u_1 + a_{2,x}u_2 + a_{3,x}u_3 \\
    a_{1,y}u_1 + a_{2,y}u_2 + a_{3,y}u_3 \\
    a_{1,z}u_1 + a_{2,z}u_2 + a_{3,z}u_3
\end{bmatrix} = u_1 a_1 + u_2 a_2 + u_3 a_3
\]
Determinant of a Linear Map

Q: If a matrix $A$ encodes a *linear map* $f$, what does $\det(A)$ mean?

A: It measures the *change in volume*.

Q: What does the *sign* of the determinant tell us, in this case?

A: It tells us whether *orientation* was reversed ($\det(A) < 0$)

(Do we really need a *matrix* in order to talk about the determinant of a linear map?)
Other Triple Products

- Super useful for working w/ vectors in 3D.
- E.g., **Jacobi identity** for the cross product:

\[
\begin{align*}
\mathbf{u} \times (\mathbf{v} \times \mathbf{w}) & + \\
\mathbf{v} \times (\mathbf{w} \times \mathbf{u}) & + \\
\mathbf{w} \times (\mathbf{u} \times \mathbf{v}) & = 0
\end{align*}
\]

- Why is it true, geometrically?
- There is a geometric reason, but not nearly as obvious as det: has to do w/ fact that triangle's altitudes meet at a point.

- Yet another triple product: **Lagrange's identity**

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

(Can you come up with a geometric interpretation?)
Differential Operators - Overview

- Next up: differential operators and vector fields.
- Why is this useful for computer graphics?
  - Many physical/geometric problems expressed in terms of relative rates of change (ODEs, PDEs).
  - These tools also provide foundation for numerical optimization—e.g., minimize cost by following the gradient of some objective.

\[ \frac{d}{dt} \phi(x) = \frac{d^2}{dx^2} \phi(x) \]
Derivative as Slope

Consider a function \( f(x) : \mathbb{R} \rightarrow \mathbb{R} \).

What does its derivative \( f' \) mean?

One interpretation “rise over run”

Corresponds to standard definition:

\[
f'(x_0) := \lim_{\varepsilon \to 0} \frac{f(x_0 + \varepsilon) - f(x_0)}{\varepsilon}
\]

Careful! What if slope is different when we walk in opposite direction?

\[
f^+(x_0) := \lim_{\varepsilon \to 0} \frac{f(x_0 + \varepsilon) - f(x_0)}{\varepsilon}
\]

\[
f^-(x_0) := \lim_{\varepsilon \to 0} \frac{f(x_0) - f(x_0 - \varepsilon)}{\varepsilon}
\]

Differentiable at \( x_0 \) if \( f^+ = f^- \).

Many functions in graphics are NOT differentiable!
Derivative as Best Linear Approximation

- Any smooth function $f(x)$ can be expressed as a Taylor series:

$$ f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{(x-x_0)^2}{2!} f''(x_0) + \cdots $$

- Replacing complicated functions with a linear (and sometimes quadratic) approximation is a powerful trick in graphics algorithms—we’ll see many examples.
Derivative as Best Linear Approximation

- Intuitively, same idea applies for functions of multiple variables:
How do we think about derivatives for a function that has multiple variables?
One way: suppose we have a function $f(x_1, x_2)$
- Take a “slice” through the function along some line
- Then just apply the usual derivative!
- Called the **directional derivative**

$$D_{\mathbf{u}}f(x_0) := \lim_{\varepsilon \rightarrow 0} \frac{f(x_0 + \varepsilon \mathbf{u}) - f(x_0)}{\varepsilon}$$
Gradient

- Given a multivariable function \( f(x) \), gradient \( \nabla f(x) \) assigns a vector at each point:

- (Ok, but which vectors, exactly?)
Gradient in Coordinates

- Most familiar definition: list of partial derivatives

  - I.e., imagine that all but one of the coordinates are just constant values, and take the usual derivative

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

- Two potential problems:
  - Role of inner product is not clear (more later!)
  - No way to differentiate functions of functions \( F(f) \) since we don’t have a finite list of coordinates \( x_1, \ldots, x_n \)

- Still, extremely common way to calculate the gradient...
Example: Gradient in Coordinates

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

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

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

\[ \nabla f(\mathbf{x}) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix} = 2\mathbf{x} \]
Gradient as Best Linear Approximation

Another way to think about it: at each point $x_0$, gradient is the vector $\nabla f(x_0)$ that leads to the best possible approximation

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

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.
The gradient takes you uphill...

- Another way to think about it: direction of “steepest ascent”
- I.e., what direction should we travel to increase value of function as quickly as possible?
- This viewpoint leads to algorithms for optimization, commonly used in graphics.
Gradient and Directional Derivative

At each point $x$, gradient is unique vector $\nabla f(x)$ such that

$$\langle \nabla f(x), u \rangle = D_u f(x)$$

for all $u$. In other words, such that taking the inner product w/ this vector gives you the directional derivative in any direction $u$.

Can’t happen if function is not differentiable!

(Notice: gradient also depends on choice of inner product...)
Example: Gradient of Dot Product

Consider the dot product expressed in terms of matrices:

\[ f := \mathbf{u}^T \mathbf{v} \]

What is gradient of \( f \) with respect to \( \mathbf{u} \)?

One way: write it out in coordinates:

\[ \mathbf{u}^T \mathbf{v} = \sum_{i=1}^{n} u_i v_i \]

In other words:

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

\[ \Rightarrow \nabla_{\mathbf{u}} f = \left[ \begin{array}{c} v_1 \\ \vdots \\ v_n \end{array} \right] \]

In other words:

\[ \nabla_{\mathbf{u}} (\mathbf{u}^T \mathbf{v}) = \mathbf{v} \]

Not so different from \( \frac{d}{dx} (xy) = y \!
Gradients of Matrix-Valued Expressions

- **EXTREMELY** useful in graphics to be able to differentiate expressions involving matrices

- Ultimately, expressions look much like ordinary derivatives

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

<table>
<thead>
<tr>
<th>Matrix Derivative</th>
<th>Looks Like</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\nabla_x (x^T y) = y$</td>
<td>$\frac{d}{dx} xy = y$</td>
</tr>
<tr>
<td>$\nabla_x (x^T x) = 2x$</td>
<td>$\frac{d}{dx} x^2 = 2x$</td>
</tr>
<tr>
<td>$\nabla_x (x^T Ay) = Ay$</td>
<td>$\frac{d}{dx} axy = ay$</td>
</tr>
<tr>
<td>$\nabla_x (x^T Ax) = 2Ax$</td>
<td>$\frac{d}{dx} ax^2 = 2ax$</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

  Excellent resource: Petersen & Pedersen, “The Matrix Cookbook”

- At least once in your life, work these out meticulously in coordinates (to convince yourself they’re true).

- Then... forget about coordinates altogether!
Advanced*: L² Gradient

- Consider a function of a function \( F(f) \)
- What is the gradient of F with respect to f?
- Can’t take partial derivatives anymore!
- Instead, look for function \( \nabla F \) such that for all functions u,

\[
\langle \langle \nabla F, u \rangle \rangle = D_u F
\]

- What is directional derivative of a function of a function??
- Don’t freak out—just return to good old-fashioned limit:

\[
D_u F(f) = \lim_{\varepsilon \to 0} \frac{F(f + \varepsilon u) - F(f)}{\varepsilon}
\]

- This strategy becomes much clearer w/ a concrete example...

*as in, NOT on the test! (But perhaps somewhere in the test of life...)
Consider function $F(f) := \langle f, g \rangle$ for $f, g: [0, 1] \rightarrow \mathbb{R}$

I claim the gradient is: $\nabla F = g$

Does this make sense intuitively? How can we increase inner product with $g$ as quickly as possible?

- inner product measures how well functions are "aligned"
- $g$ is definitely function best-aligned with $g$!
- so to increase inner product, add a little bit of $g$ to $f$

(Can you work this solution out formally?)
Consider function $F(f) := \|f\|^2$ for arguments $f: [0,1] \rightarrow \mathbb{R}$.

At each “point” $f_0$, we want function $\nabla F$ such that for all functions $u$

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

Expanding 1st term in numerator, we get

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

Hence, limit becomes

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

The only solution to $\langle \nabla F(f_0), u \rangle = 2 \langle f_0, u \rangle$ for all $u$ is

$$\nabla F(f_0) = 2f_0$$

not much different from $\frac{d}{dx}x^2 = 2x$!
Key idea:
Once you get the hang of taking the gradient of ordinary functions, it’s *(superficially)* not much harder for more exotic objects like matrices, functions of functions, ...
Vector Fields

- Gradient was our first example of a vector field.
- In general, a vector field assigns a vector to each point in space.
- E.g., can think of a 2-vector field in the plane as a map

\[ X : \mathbb{R}^2 \rightarrow \mathbb{R}^2 \]

- For example, we saw a gradient field

\[ \nabla f(x, y) = (2x, 2y) \]

(for the function \( f(x,y) = x^2 + y^2 \))
Q: How do we measure the change in a vector field?
Divergence and Curl

Two basic derivatives for vector fields:

“How much is field shrinking/expanding?”

“How much is field spinning?”

div X  curl Y
Divergence

- Also commonly written as $\nabla \cdot X$
- Suggests a coordinate definition for divergence
- Think of $\nabla$ as a “vector of derivatives”
  \[
  \nabla = \left( \frac{\partial}{\partial u_1}, \ldots, \frac{\partial}{\partial u_n} \right)
  \]
- Think of $X$ as a “vector of functions”
  \[
  X(u) = (X_1(u), \ldots, X_n(u))
  \]
- Then divergence is
  \[
  \nabla \cdot X := \sum_{i=1}^{n} \frac{\partial X_i}{\partial u_i}
  \]
Divergence - Example

- Consider the vector field \( X(u, v) := (\cos(u), \sin(v)) \)
- Divergence is then

\[
\nabla \cdot X = \frac{\partial}{\partial u} \cos(u) + \frac{\partial}{\partial v} \sin(v) = -\sin(u) + \cos(v).
\n\]
Curl

- Also commonly written as $\nabla \times X$
- Suggests a coordinate definition for curl
- This time, think of $\nabla$ as a vector of just three derivatives:

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

- Think of $X$ as vector of three functions:

$$X(u) = (X_1(u), X_2(u), X_3(u))$$

- Then curl is

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

(2D “curl”: $\nabla \times X := \frac{\partial X_2}{\partial u_1} - \frac{\partial X_1}{\partial u_2}$)
Curl - Example

- Consider the vector field \( X(u, v) := (-\sin(v), \cos(u)) \)
- (2D) Curl is then

\[
\nabla \times X = \frac{\partial}{\partial u} \cos(u) - \frac{\partial}{\partial v} (-\sin(v)) = -\sin(u) + \cos(v).
\]
Notice anything about the relationship between curl and divergence?
Divergence vs. Curl (2D)

- Divergence of $X$ is the same as curl of 90-degree rotation of $X$:

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

- Playing these kinds of games w/ vector fields plays an important role in algorithms (e.g., fluid simulation)

- (Q: Can you come up with an analogous relationship in 3D?)
Example: Fluids w/ Stream Function

Our method

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

\[
u = \nabla \times \Psi
\]

Single-phase Pressure solver

\[
\Delta p = \nabla \cdot u^*
\]

\[
u = u^* - \nabla p
\]

Laplacian

- One more operator we haven’t seen yet: the Laplacian
- Unbelievably important object in graphics, showing up across geometry, rendering, simulation, imaging
  - basis for Fourier transform / frequency decomposition
  - used to define model PDEs (Laplace, heat, wave equations)
  - encodes rich information about geometry
Laplacian—Visual Intuition

Q: For ordinary function $f(x)$, what does 2nd derivative tell us?

Likewise, Laplacian measures “curvature” of a function.

For further interpretations of the Laplacian, see https://youtu.be/oEq9R0I9Umk
Laplacian—Many Definitions

- Maps a scalar function to another scalar function (*linearly*)!

- Usually* denoted by $\Delta$ "Delta"

- Many starting points for Laplacian:
  - divergence of gradient $\Delta f := \nabla \cdot \nabla f = \text{div}(\text{grad} f)$
  - sum of 2nd partial derivatives $\Delta f := \sum_{i=1}^{n} \frac{\partial^2 f}{\partial x_i^2}$
  - gradient of Dirichlet energy $\Delta f := -\nabla f \left( \frac{1}{2} \| \nabla f \| ^2 \right)$
  - by analogy: graph Laplacian
  - variation of surface area
  - trace of Hessian …

*Or by $\nabla^2$, but we’ll reserve this symbol for the Hessian.
Laplacian—Example

Let’s use coordinate definition: \[ \Delta f := \sum_i \frac{\partial^2 f}{\partial x_i^2} \]

Consider the function \[ f(x_1, x_2) := \cos(3x_1) + \sin(3x_2) \]

We have

\[ \frac{\partial^2}{\partial x_1^2} f = \frac{\partial^2}{\partial x_1^2} \cos(3x_1) + \frac{\partial^2}{\partial x_1^2} \sin(3x_2) = 0 \]

\[ -3 \frac{\partial}{\partial x_1} \sin(3x_1) = -9 \cos(3x_1). \]

Hence,

\[ \Delta f = -9(\cos(3x_1) + \sin(3x_2)) = -9f \]

Interesting! Does this always happen?
Hessian

- Our final differential operator—**Hessian** will help us locally approximate complicated functions by a few simple terms.

- Recall our **Taylor series**
  
  \[ f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{(x-x_0)^2}{2!} f''(x_0) + \cdots \]

- How do we do this for multivariable functions?

- Already talked about best **linear** approximation, using gradient:

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

**Hessian** gives us next, "quadratic" term.
Hessian in Coordinates

- Typically denote Hessian by symbol $\nabla^2$

- Just as gradient was "vector that gives us partial derivatives of the function," Hessian is "operator that gives us partial derivatives of the gradient":

$$ (\nabla^2 f) \mathbf{u} := D_{\mathbf{u}} (\nabla f) $$

- For a function $f(\mathbf{x}): \mathbb{R}^n \rightarrow \mathbb{R}$, can be more explicit:

$$ \nabla^2 f := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix} $$

Q: Why is this matrix always symmetric?
Taylor Series for Multivariable Functions

- Using Hessian, can now write 2nd-order approximation of any smooth, multivariable function $f(x)$ around some point $x_0$:

$$
\begin{align*}
f(x) & \approx f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle + \langle \nabla^2 f(x_0)(x - x_0), x - x_0 \rangle / 2 \\
& = c + b^T (x - x_0) + A(x - x_0)^T(x - x_0)
\end{align*}
$$

- Can write this in matrix form as

$$
f(u) \approx \frac{1}{2} u^T A u + b^T u + c, \quad u := x - x_0
$$

Will see later on how this approximation is very useful for optimization!
Next time: Rasterization

- Next time, we’ll talk about how to draw triangles
- A lot more interesting (and difficult!) than it might seem…
- Leads to a deep understanding of modern graphics hardware