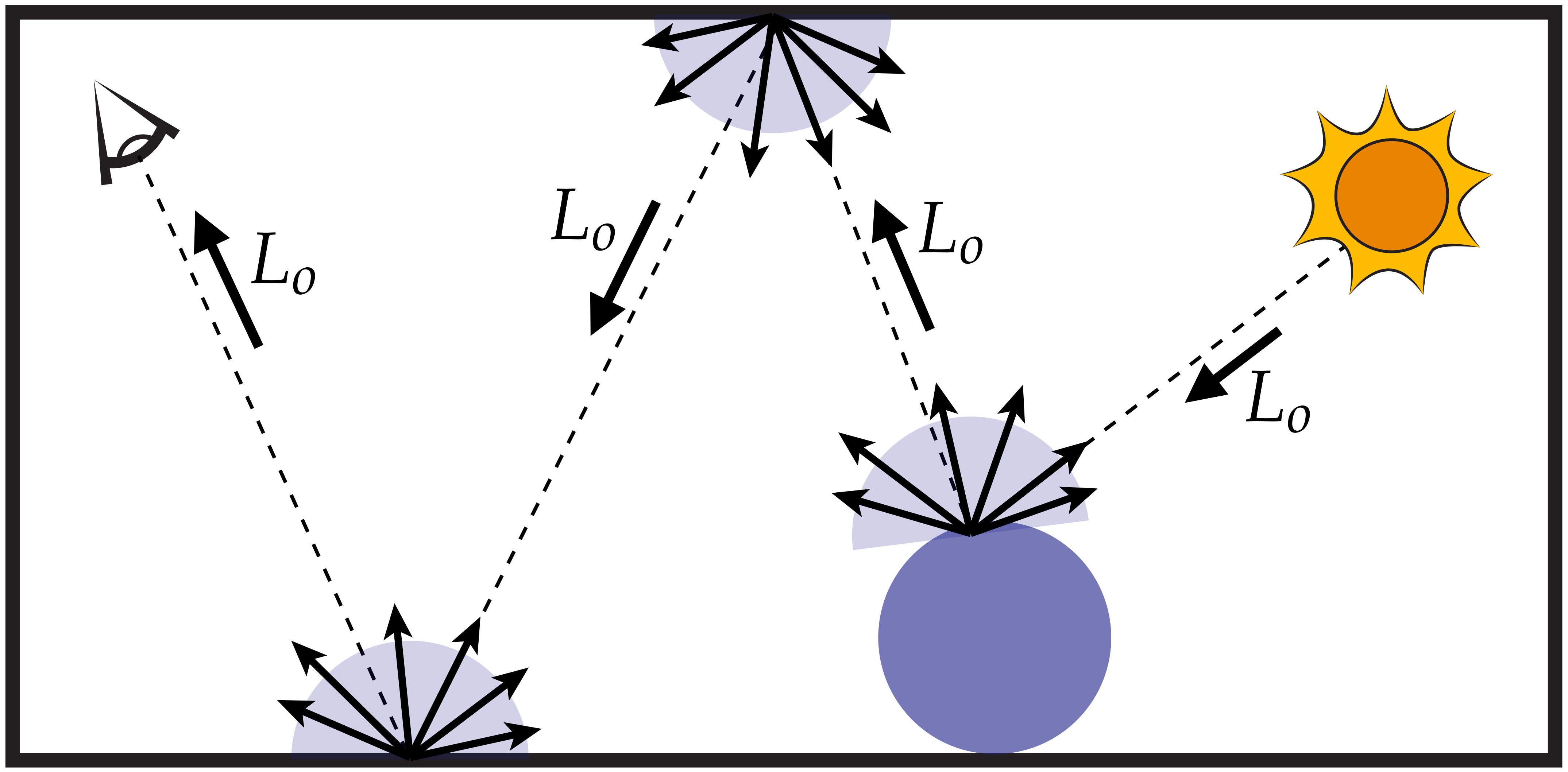
Numerical Integration

Computer Graphics
CMU 15-462/15-662

Motivation: The Rendering Equation

- **Recall the rendering equation, which models light "bouncing around the scene":**



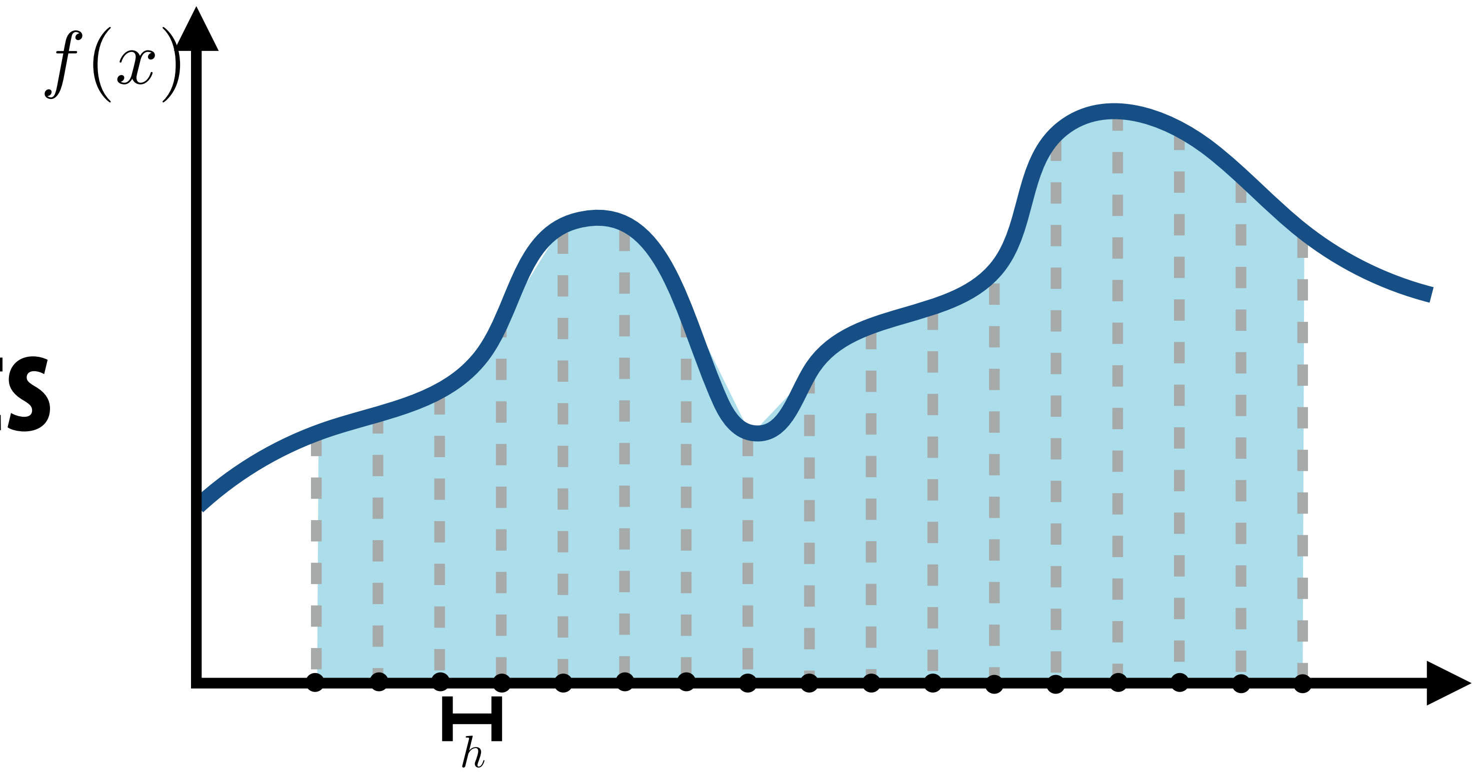
$$L_o(\mathbf{p}, \omega_o) = L_e(\mathbf{p}, \omega_o) + \int_{\mathcal{H}^2} f_r(\mathbf{p}, \omega_i \to \omega_o) L_i(\mathbf{p}, \omega_i) \cos\theta \, d\omega_i$$

How can we possibly evaluate this integral?

Numerical Integration—Overview

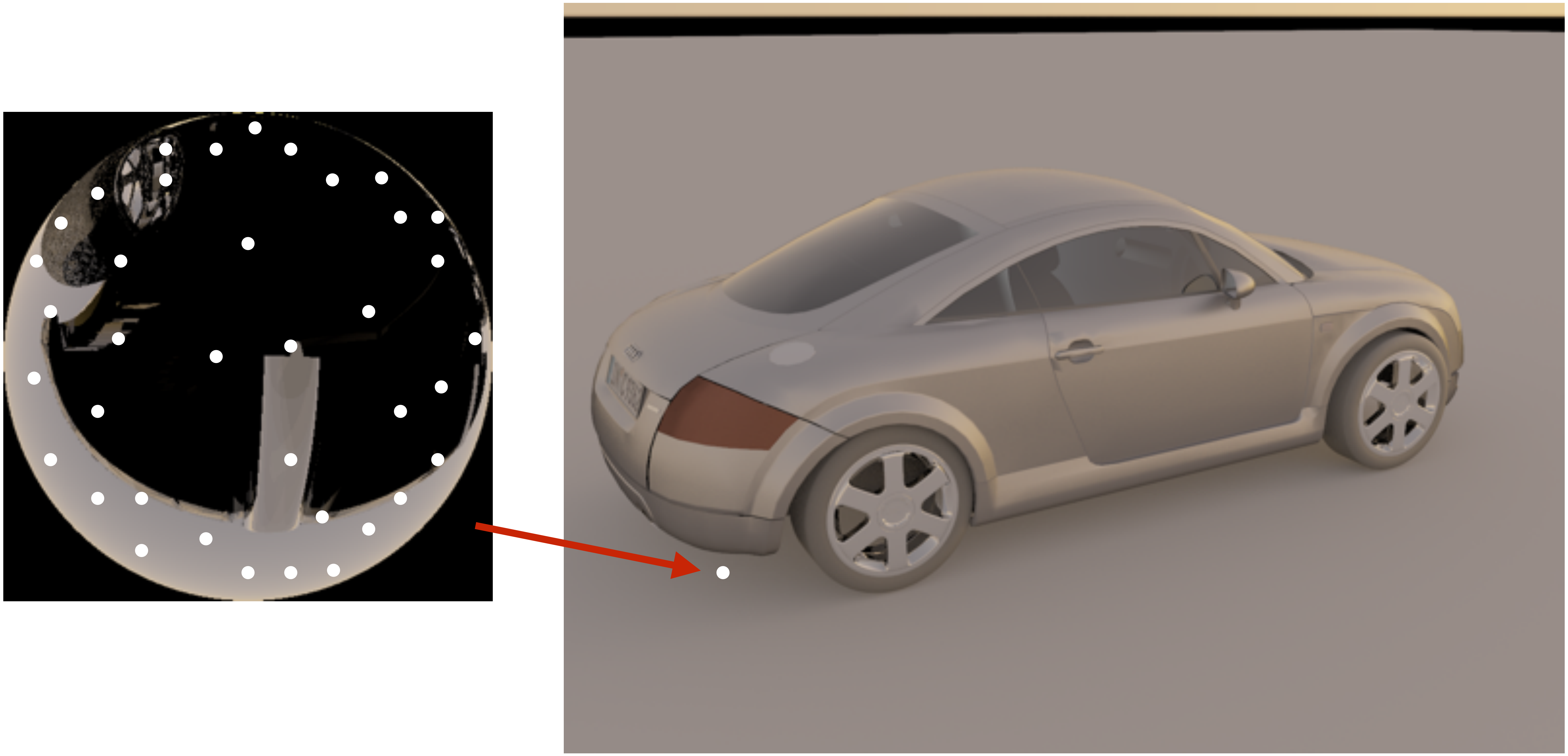
- In graphics, many quantities we're interested in are naturally expressed as integrals (total brightness, total area, ...)
- For very, very simple integrals, we can compute the solution analytically
- For everything else, we have to compute a numerical approximation
- Basic idea:
  - integral is "area under curve"
  - sample the function at many points
  - integral is approximated as weighted sum

$$\int_0^1 \frac{1}{3}x^2\,dx = \left[x^3\right]_0^1 = 1$$



Rendering: what are we integrating?

- Recall this view of the world:

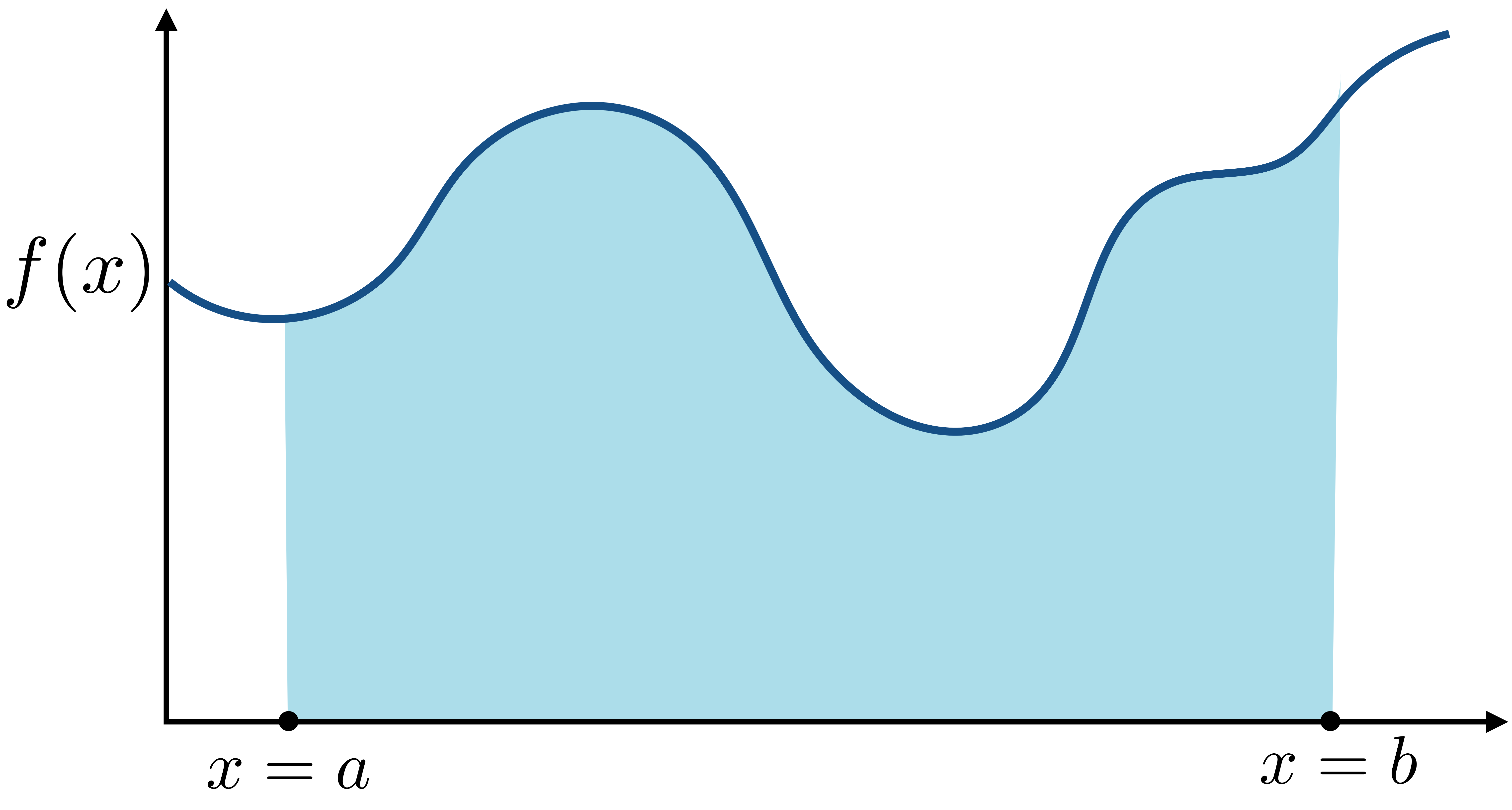


Want to "sum up"—i.e., integrate!—light from all directions

(But let's start a little simpler…)

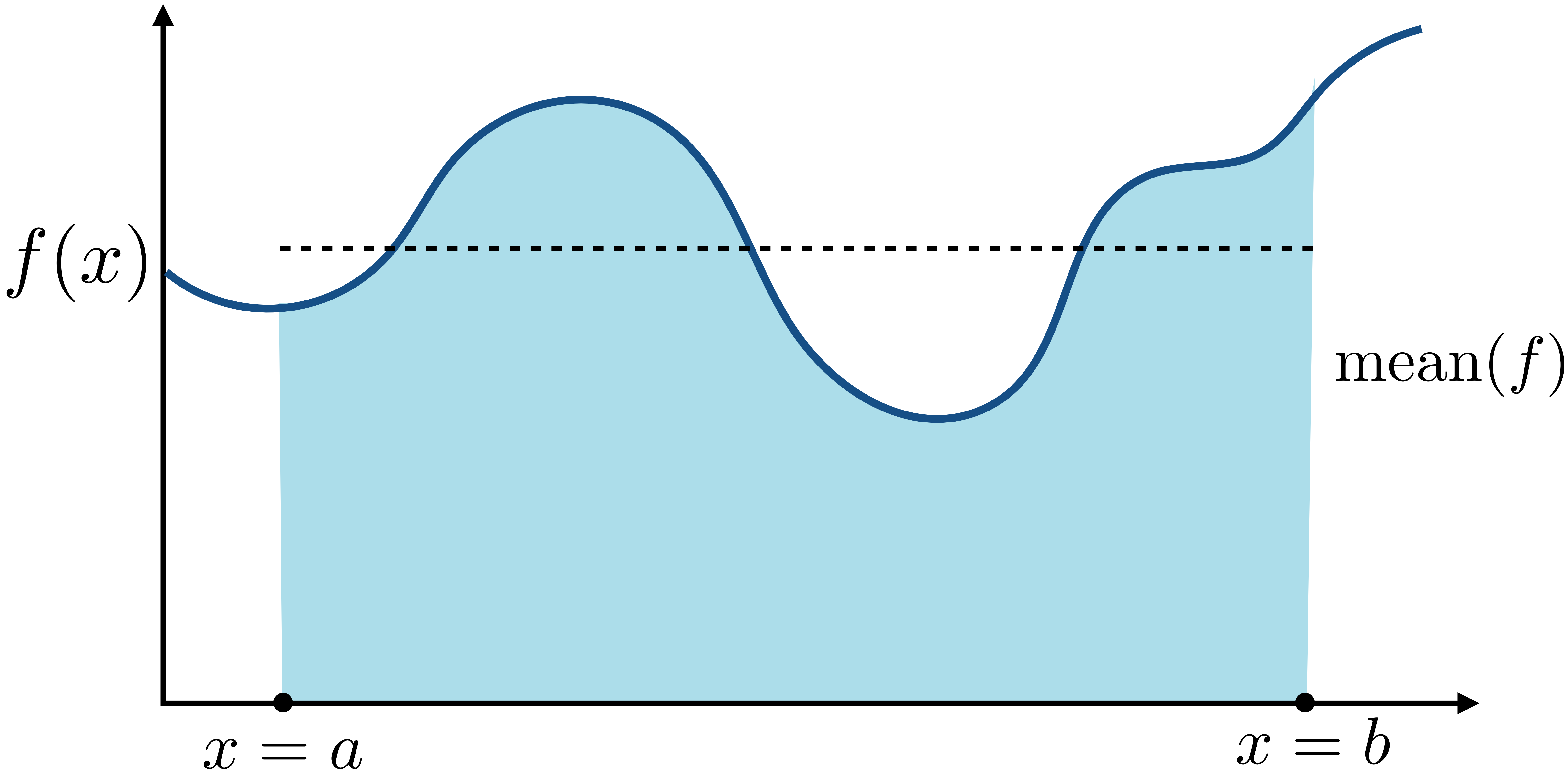
Review: integral as “area under curve”

$$\int_a^b f(x)dx$$



Or: average value times size of domain

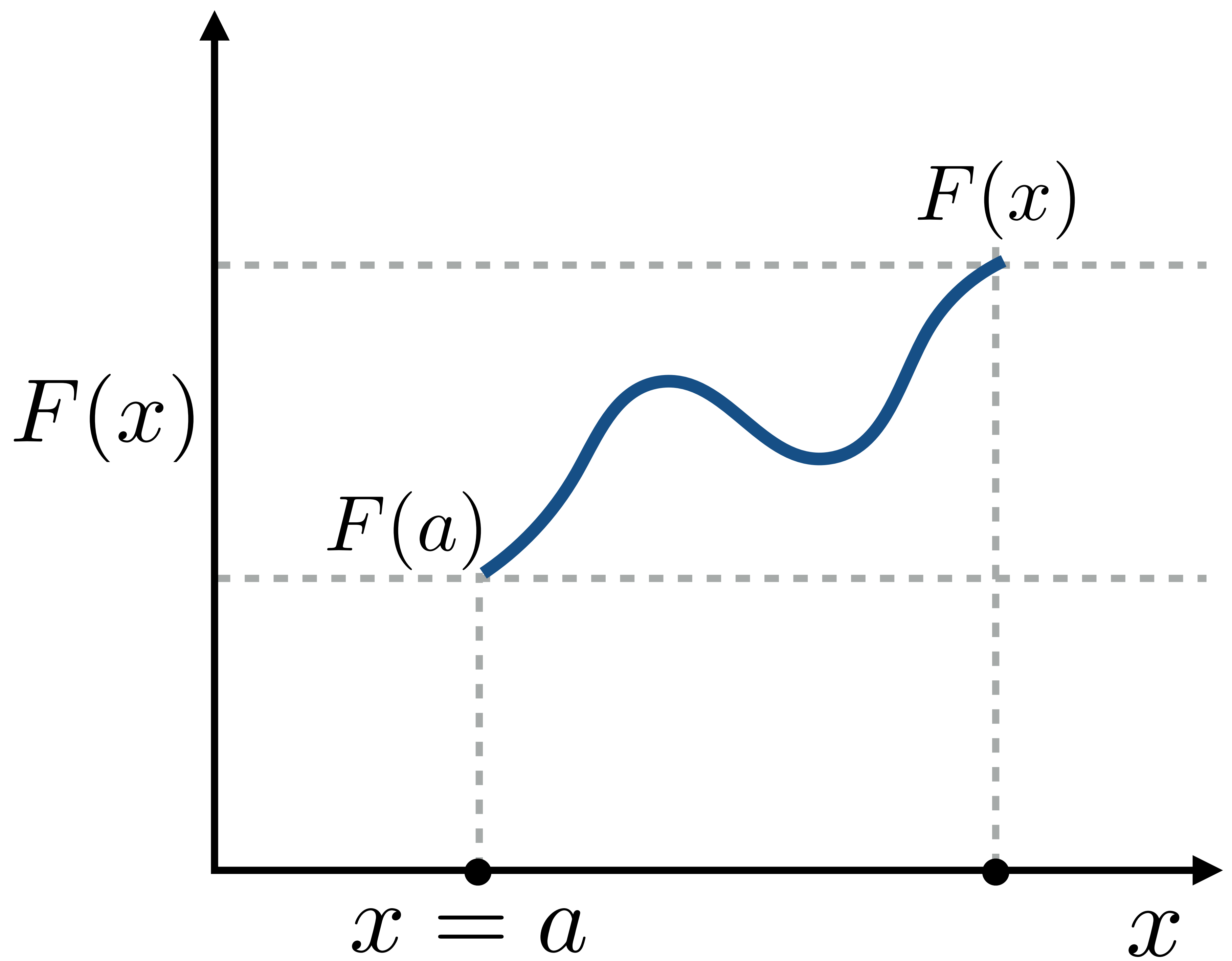
$$\int_a^b f(x)dx = (b-a)\text{mean}(f)$$



Review: fundamental theorem of calculus

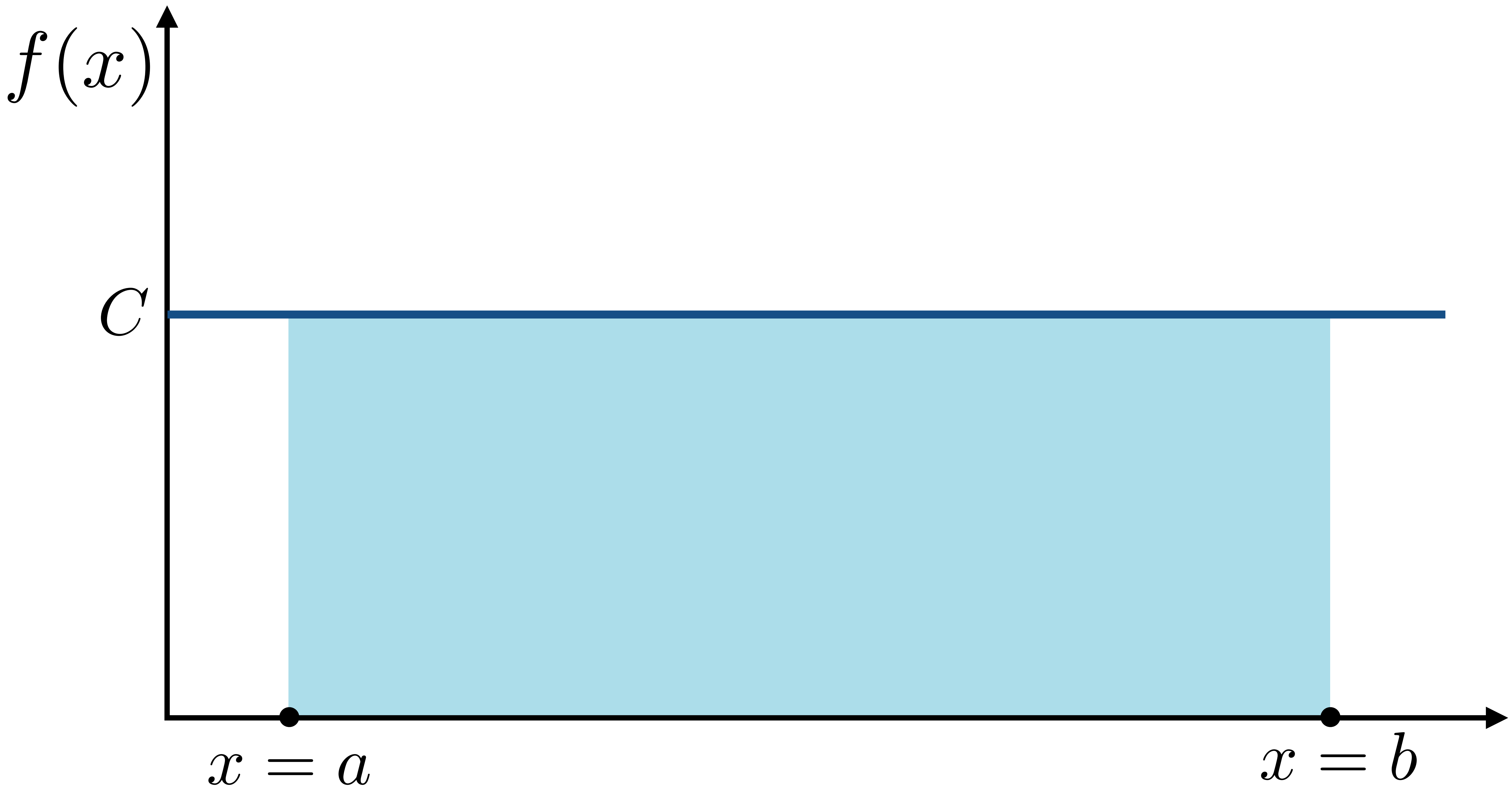
$$\int_a^b f(x)dx = F(b) - F(a)$$

$$f(x) = \frac{d}{dx}F(x)$$



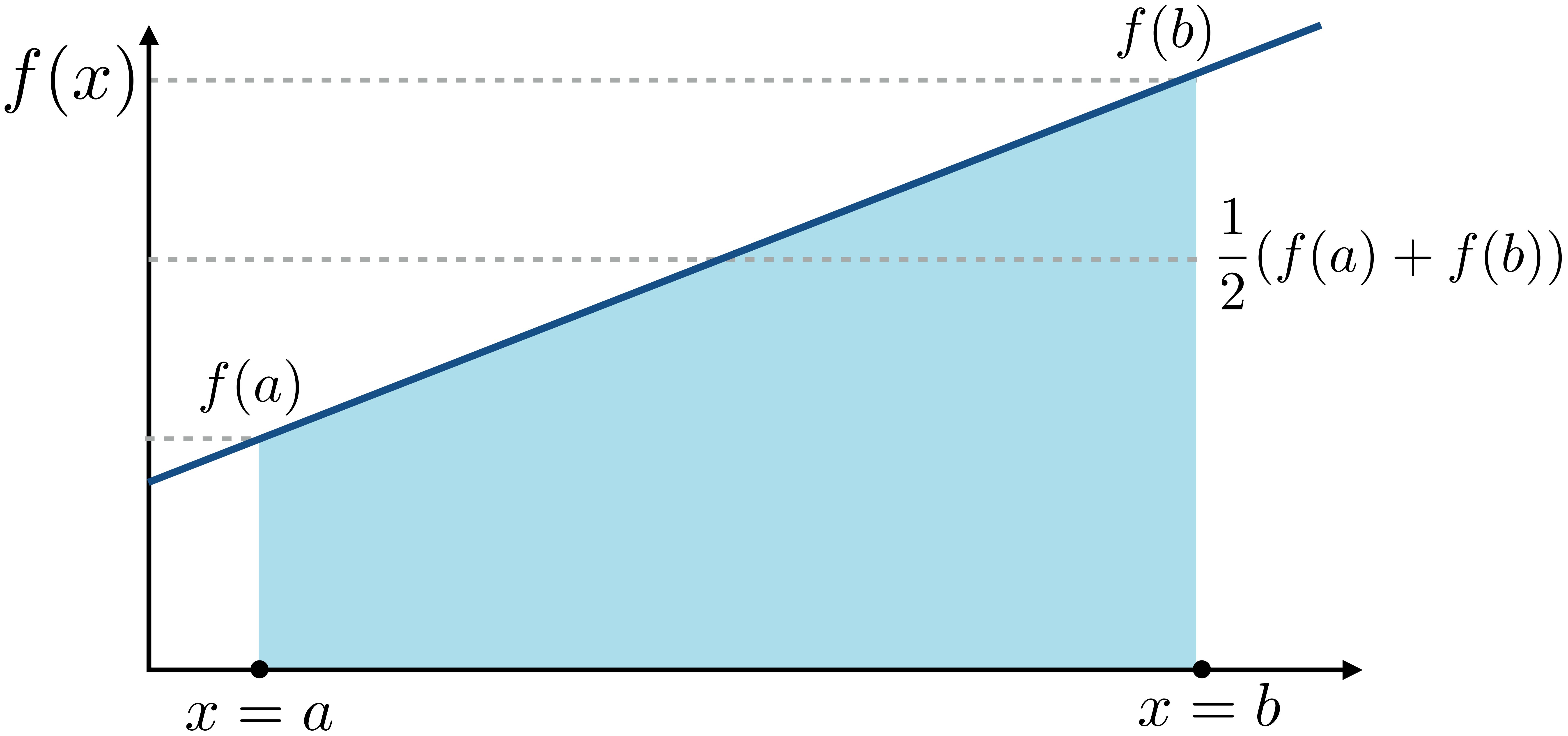
Simple case: constant function

$$\int_a^b C\,dx = (b-a)C$$



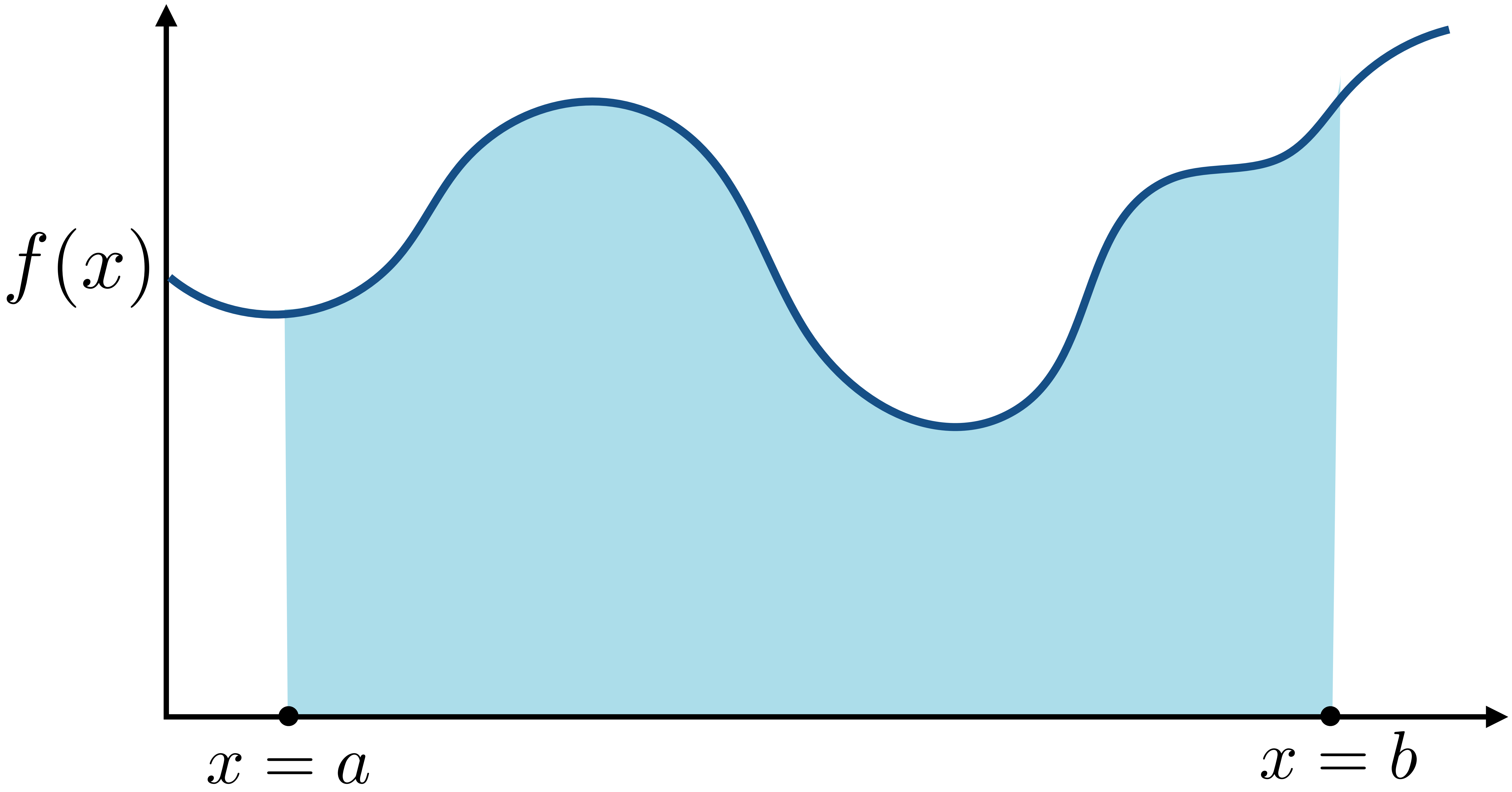
Affine function: $f(x) = cx + d$

$$\int_a^b f(x)dx = \frac{1}{2}(f(a) + f(b))(b - a)$$



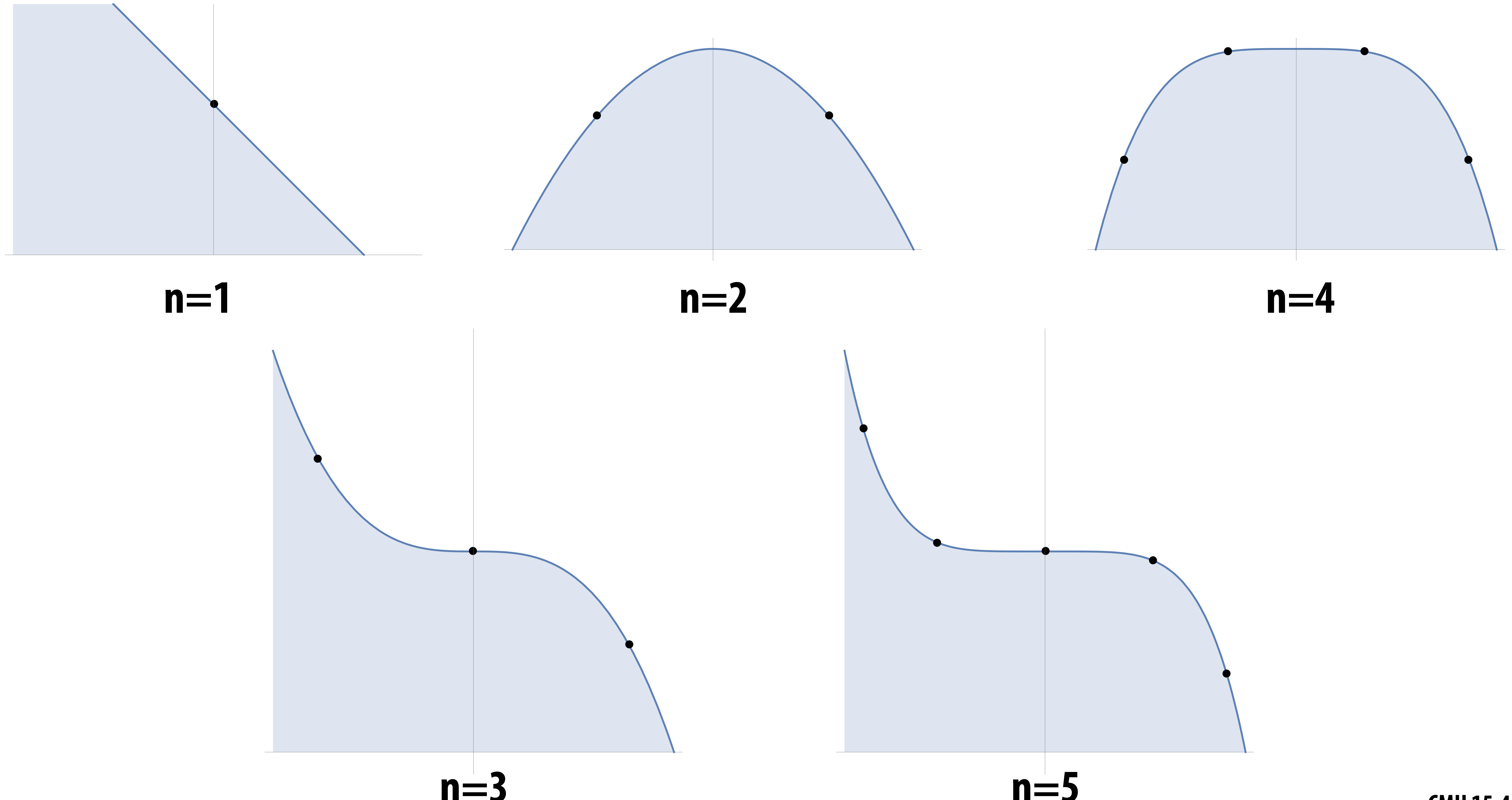
Need only one sample of the function (at just the right place…)

More general polynomials?



Gauss Quadrature

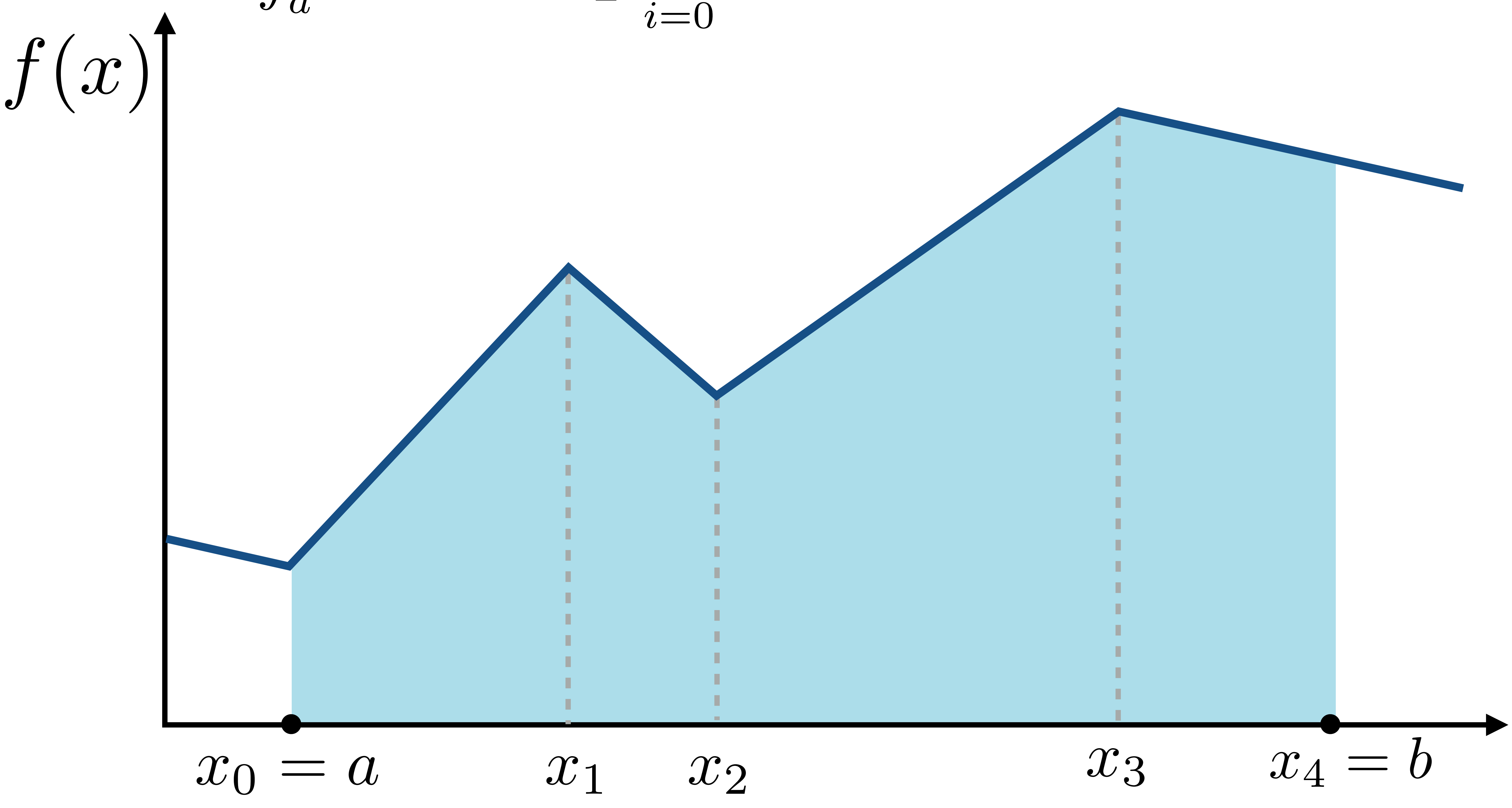
- **For any polynomial of degree 2n-1 or less, we can always obtain the exact integral by sampling at a special set of n points and taking a special weighted combination**



Piecewise affine function

For piecewise functions, just sum integral of each piece:

$$\int_a^b f(x)dx = \frac{1}{2}\sum_{i=0}^{n-1}(x_{i+1} - x_i)(f(x_i) + f(x_{i+1}))$$



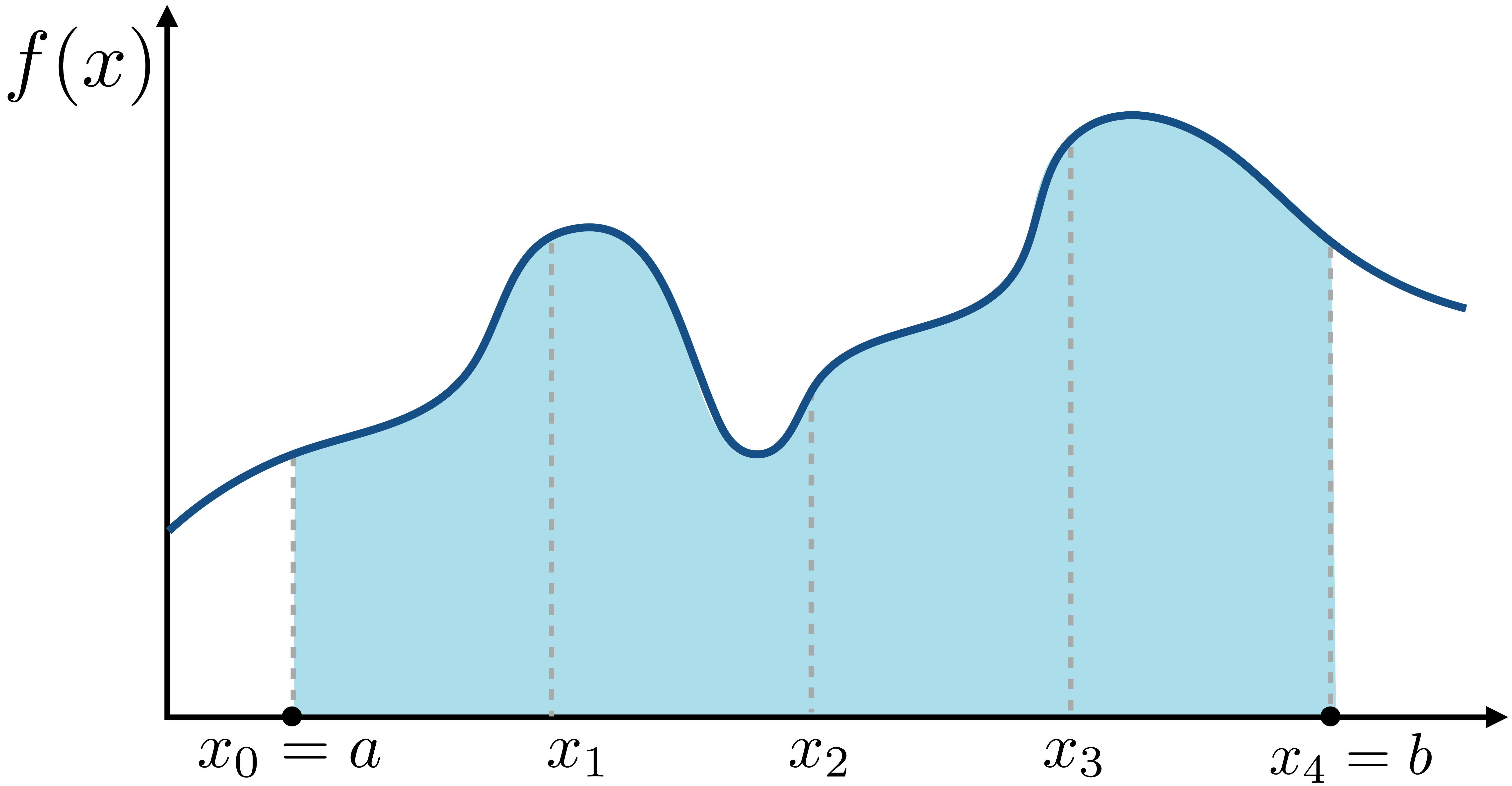
Key idea so far:
To approximate an integral, we need

(i) quadrature points, and

(ii) weights for each point

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} w_i f(x_i)$$

Arbitrary function f(x)?

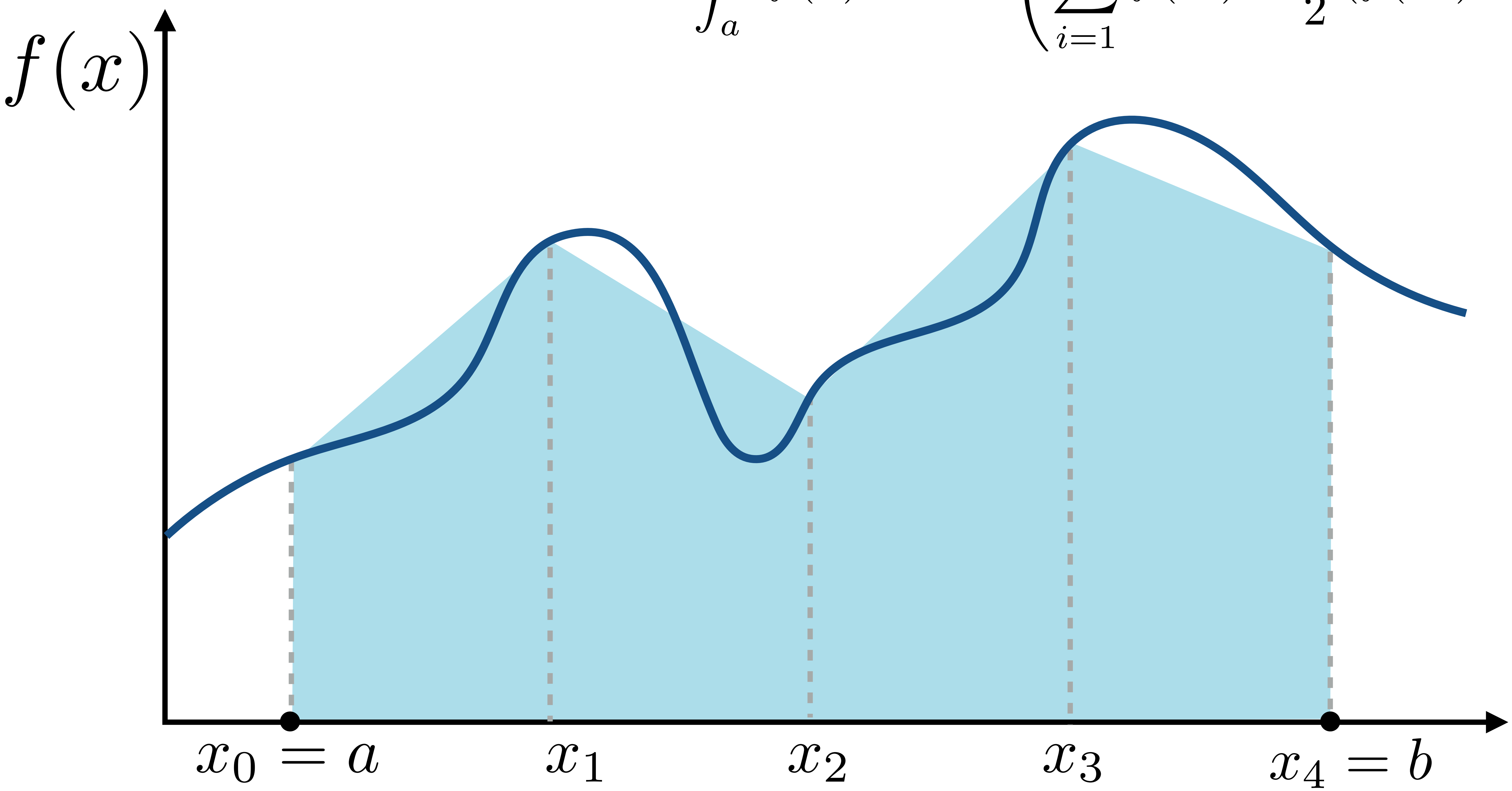


Trapezoid rule

Approximate integral of f(x) by pretending function is piecewise affine

For equal length segments: $h = \dfrac{b-a}{n-1}$

$$\int_a^b f(x)dx = h\left(\sum_{i=1}^{n-1} f(x_i) + \frac{1}{2}\left(f(x_0) + f(x_n)\right)\right)$$



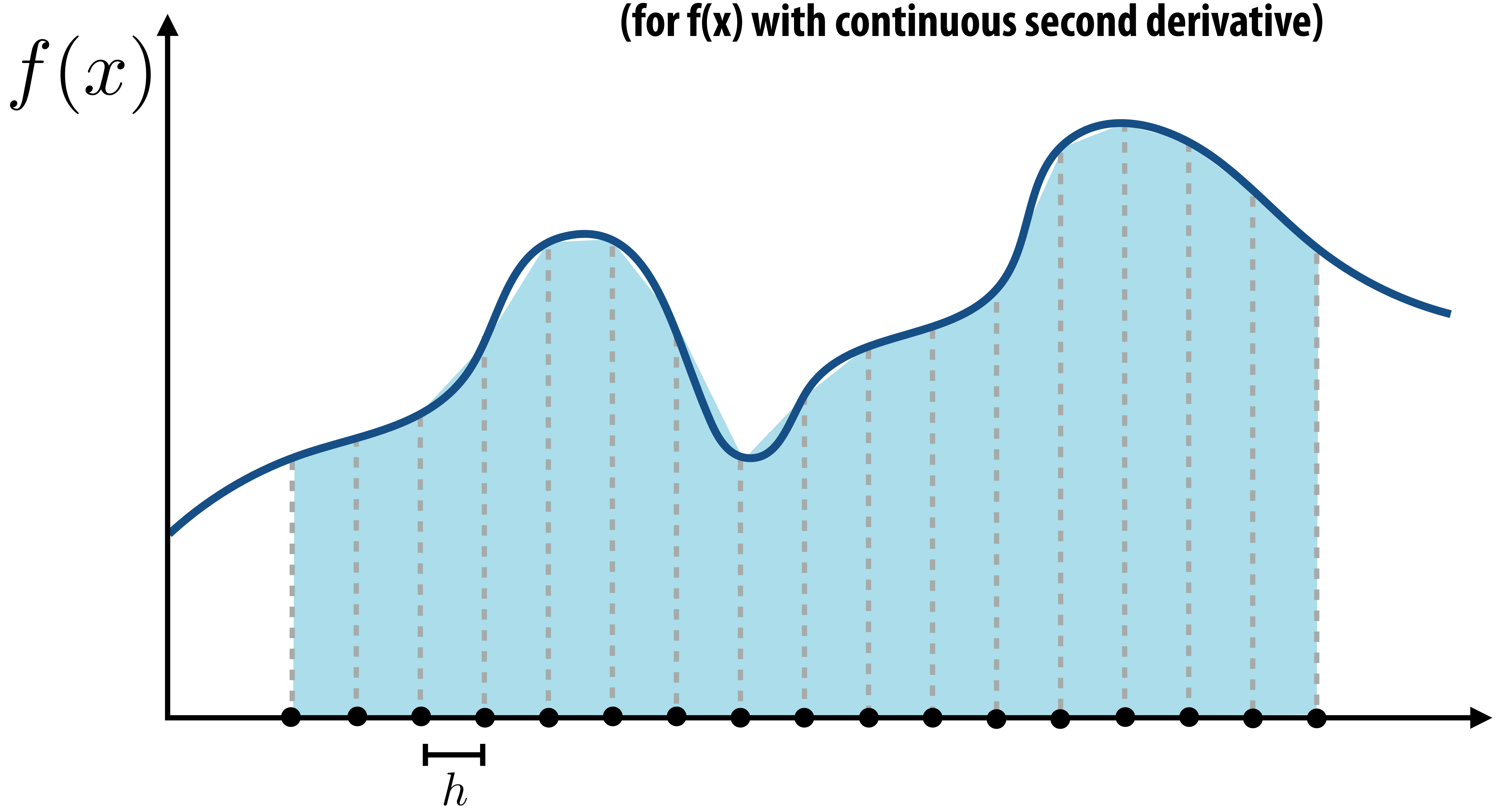
Trapezoid rule

Consider cost and accuracy of estimate as $n \to \infty$ **(or** $h \to 0$ **)**

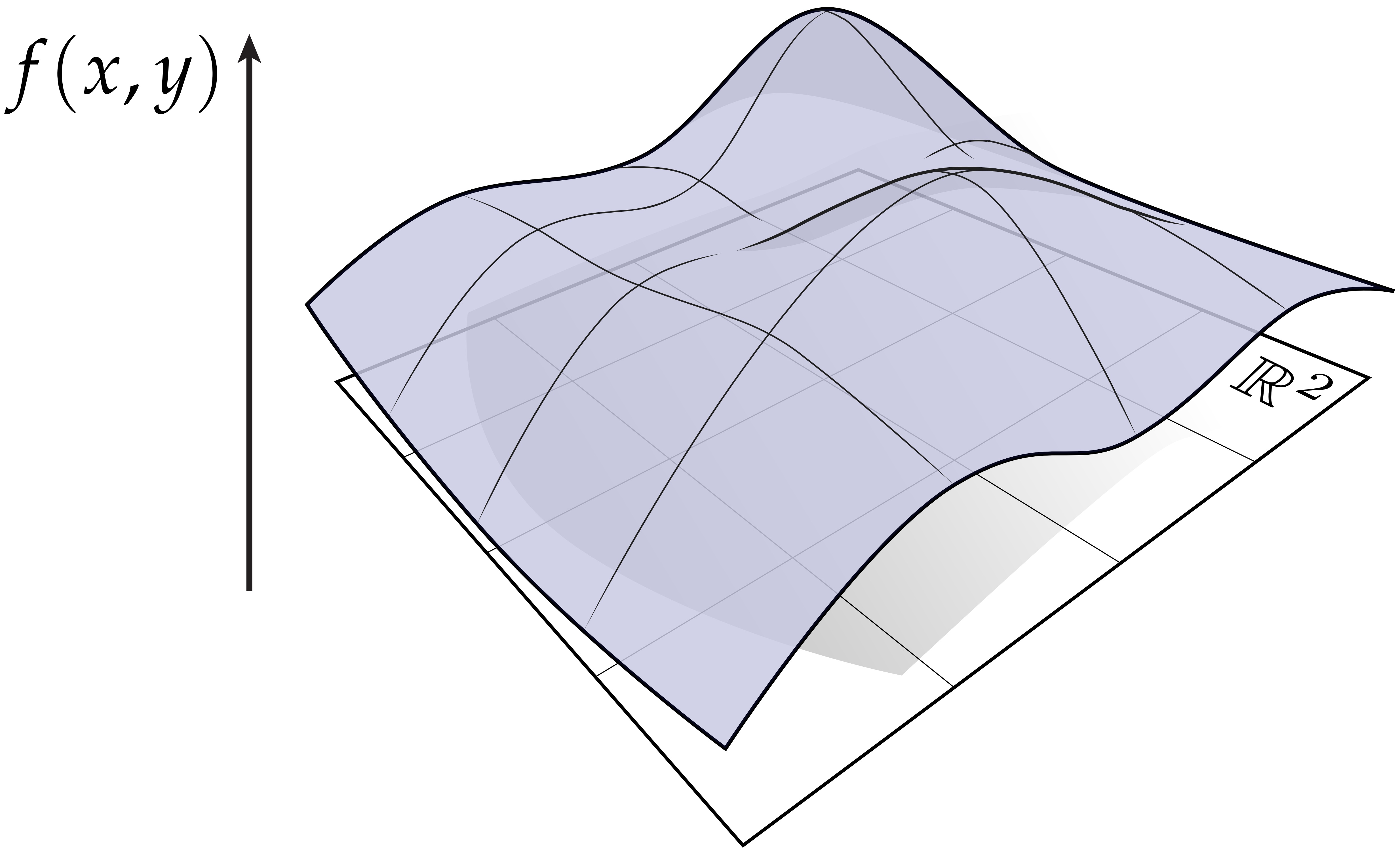
Work: $O(n)$

Error can be shown to be: $O(h^2) = O(\frac{1}{n^2})$

(for f(x) with continuous second derivative)



What about a 2D function?



How should we approximate the area underneath?

Integration in 2D

Consider integrating $f(x,y)$ using the trapezoidal rule

(apply rule twice: when integrating in x and in y)

$$\int_{a_y}^{b_y}\int_{a_x}^{b_x} f(x,y)dxdy = \int_{a_y}^{b_y}\left(\boxed{O(h^2)+\sum_{i=0}^{n}A_i f(x_i,y)}\right)dy$$

First application of rule

$$= O(h^2)+\sum_{i=0}^{n}A_i\int_{a_y}^{b_y}f(x_i,y)dy$$

$$= O(h^2)+\sum_{i=0}^{n}A_i\left(\boxed{O(h^2)+\sum_{j=0}^{n}A_j f(x_i,y_j)}\right)$$

Second application

$$= O(h^2)+\sum_{i=0}^{n}\sum_{j=0}^{n}A_iA_j f(x_i,y_j)$$

Errors add, so error still: $O(h^2)$

But work is now: $O(n^2)$

(n x n set of measurements)

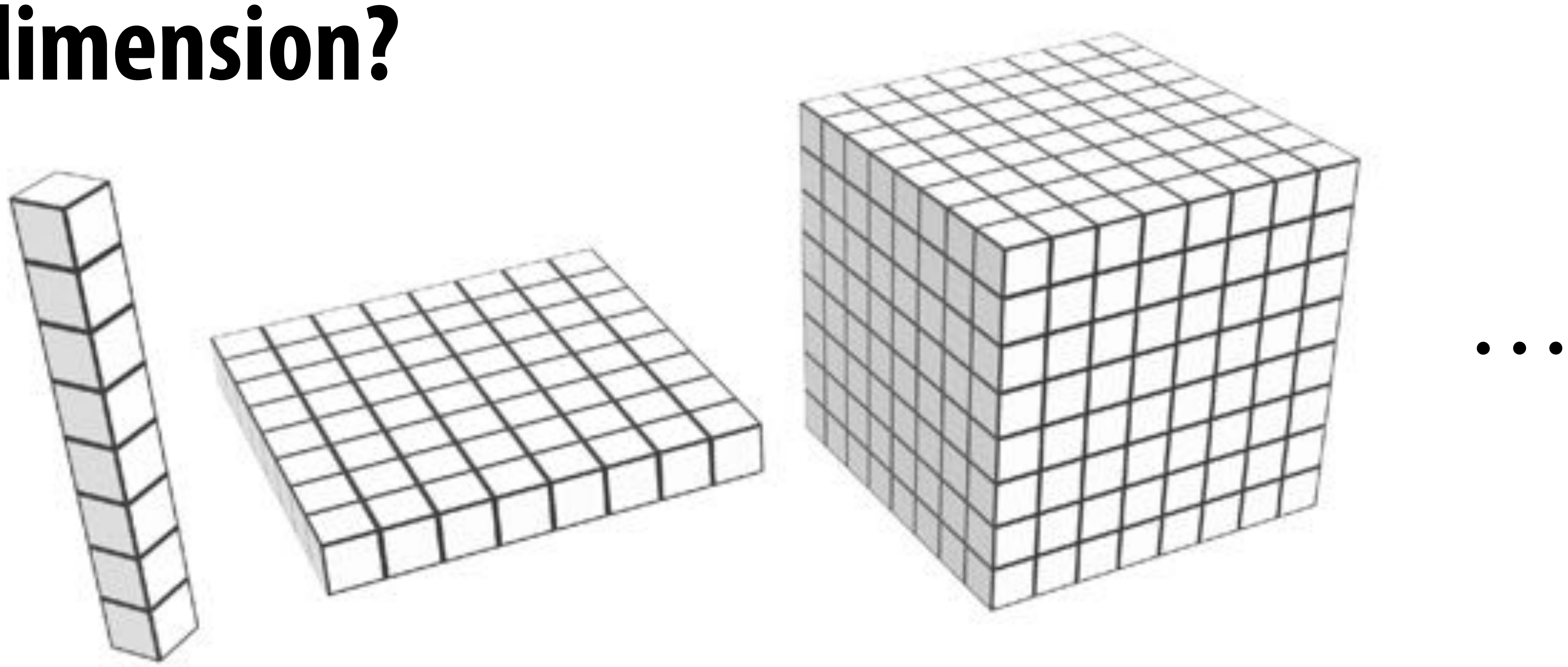
Must perform much more work in 2D to get same error bound on integral!

In K-D, let $N = n^k$

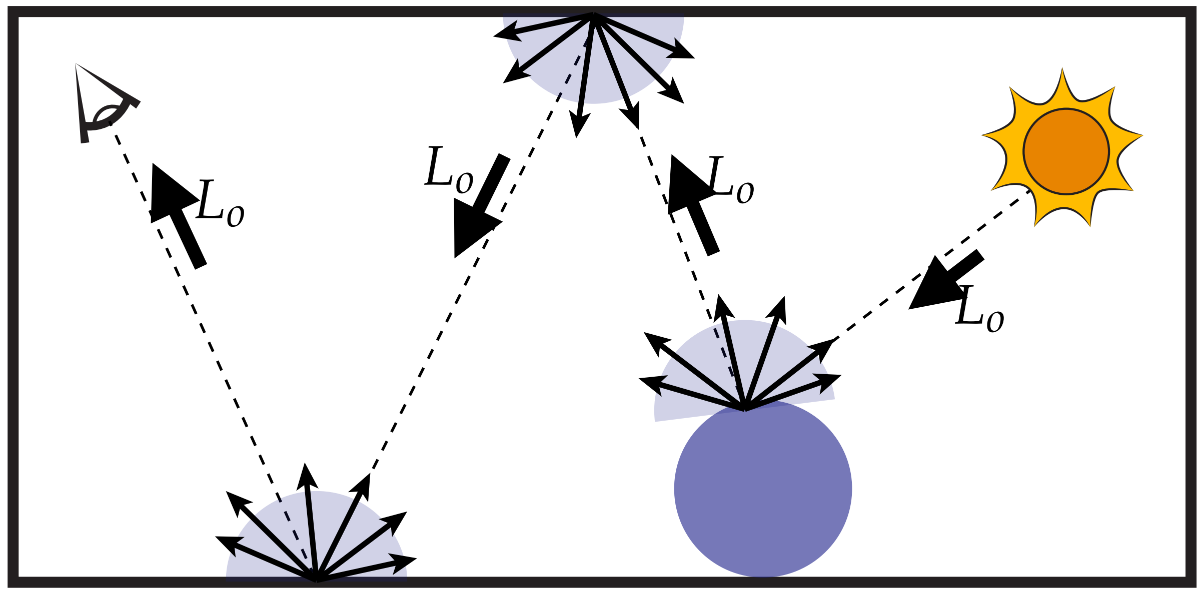
Error goes as: $O\left(\frac{1}{N^{2/k}}\right)$

Curse of Dimensionality

- How much does it cost to apply the trapezoid rule as we go up in dimension?
  - 1D: O(n)
  - 2D: O(n^2)
  - …
  - kD: O(n^k)



- For many problems in graphics (like rendering), k is very, very big (e.g., tens or hundreds or thousands)



- Applying trapezoid rule does not scale!
- Need a fundamentally different approach…

Monte Carlo Integration

Credit: many of these slides were created by Matt Pharr and Pat Hanrahan

Monte Carlo Integration

So far we've discussed techniques that use a fixed set of sample points (e.g., uniformly spaced, or obtained by finding roots of polynomial (Gaussian quadrature))

- Estimate value of integral using random sampling of function
  - Value of estimate depends on random samples used
  - But algorithm gives the correct value of integral "on average"
- Only requires function to be evaluated at random points on its domain
  - Applicable to functions with discontinuities, functions that are impossible to integrate directly
- Error of estimate is independent of the dimensionality of the integrand
  - Depends on the number of random samples used: $O\left(\frac{1}{\sqrt{n}}\right)$

Review: random variables

X random variable. Represents a distribution of potential values

$X \sim p(x)$ probability density function (PDF). Describes relative probability of a random process choosing value x

Uniform PDF: all values over a domain are equally likely

e.g., for an unbiased die

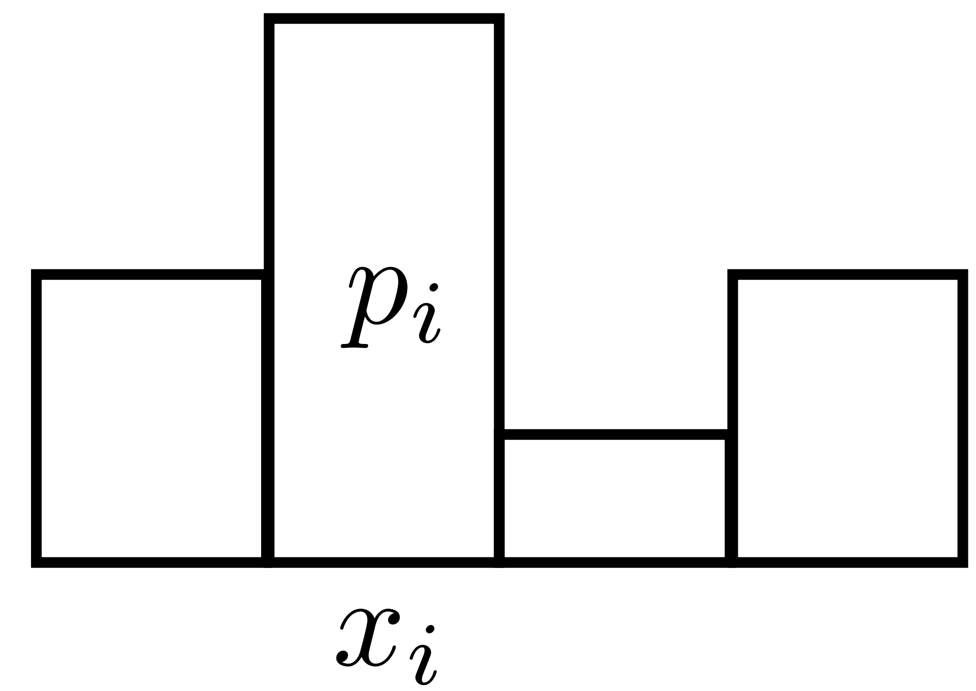
X **takes on values 1,2,3,4,5,6**

$$p(1) = p(2) = p(3) = p(4) = p(5) = p(6)$$

Discrete probability distributions

n discrete values x_i

With probability p_i



Requirements of a PDF:

$$p_i \geq 0$$

$$\sum_{i=1}^{n} p_i = 1$$

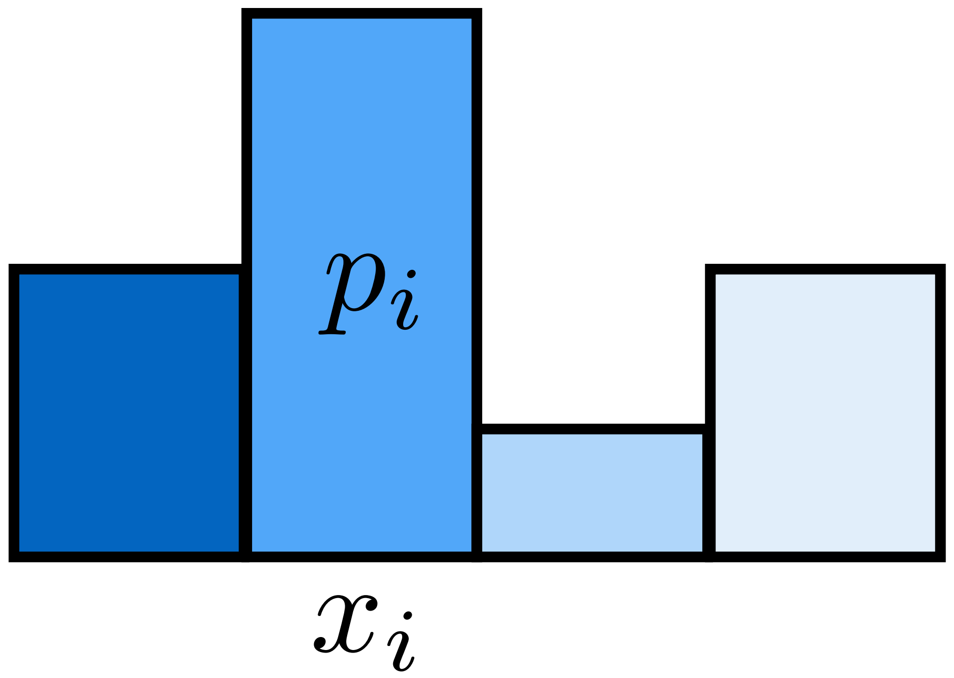
Six-sided die example: $p_i = \frac{1}{6}$

Think: p_i **is the probability that a random measurement of** X **will yield the value** x_i

X **takes on the value** x_i **with probability** p_i

Cumulative distribution function (CDF)

(For a discrete probability distribution)

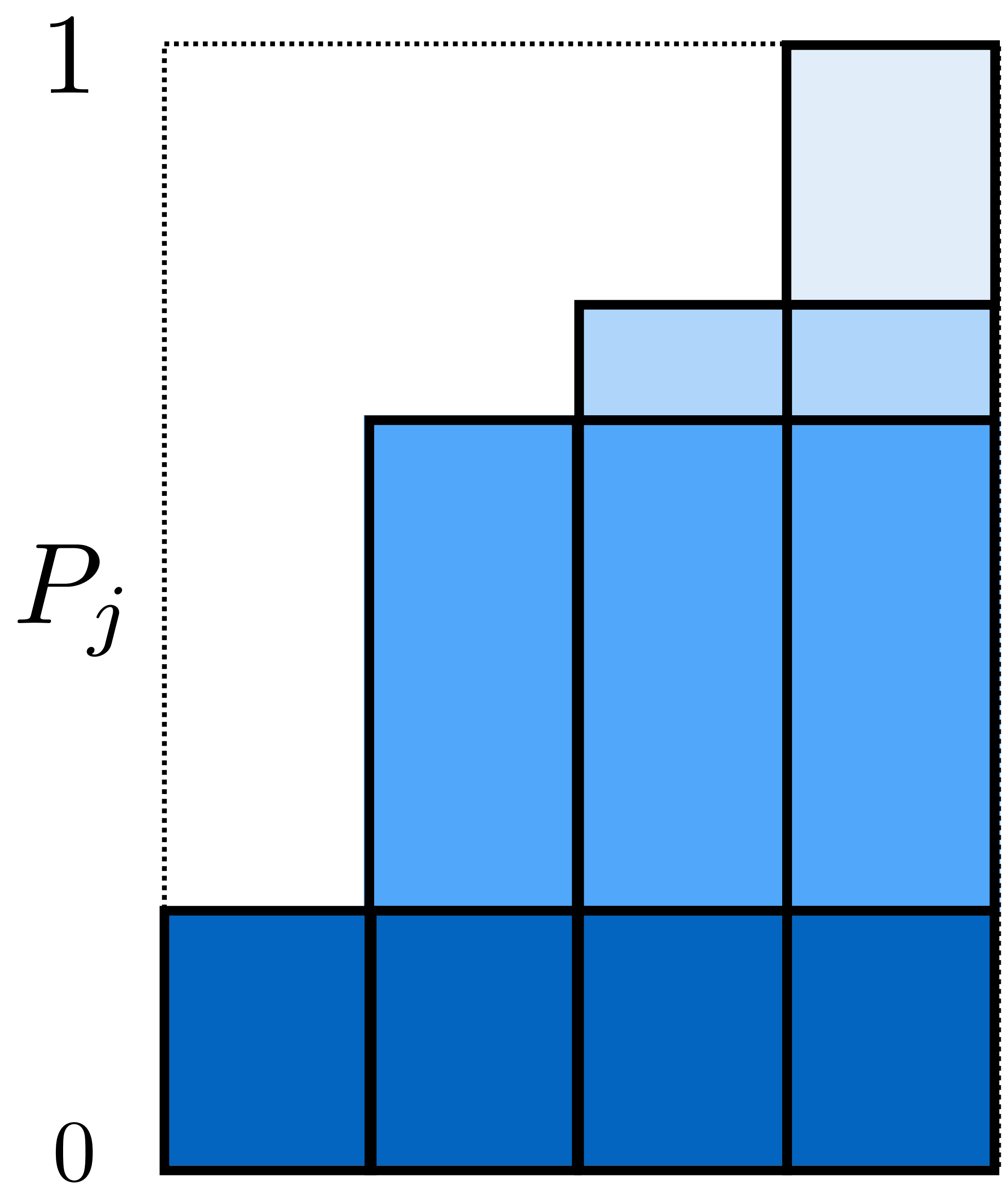


Cumulative PDF: $P_j = \sum_{i=1}^{j} p_i$

where:

$$0 \leq P_i \leq 1$$

$$P_n = 1$$



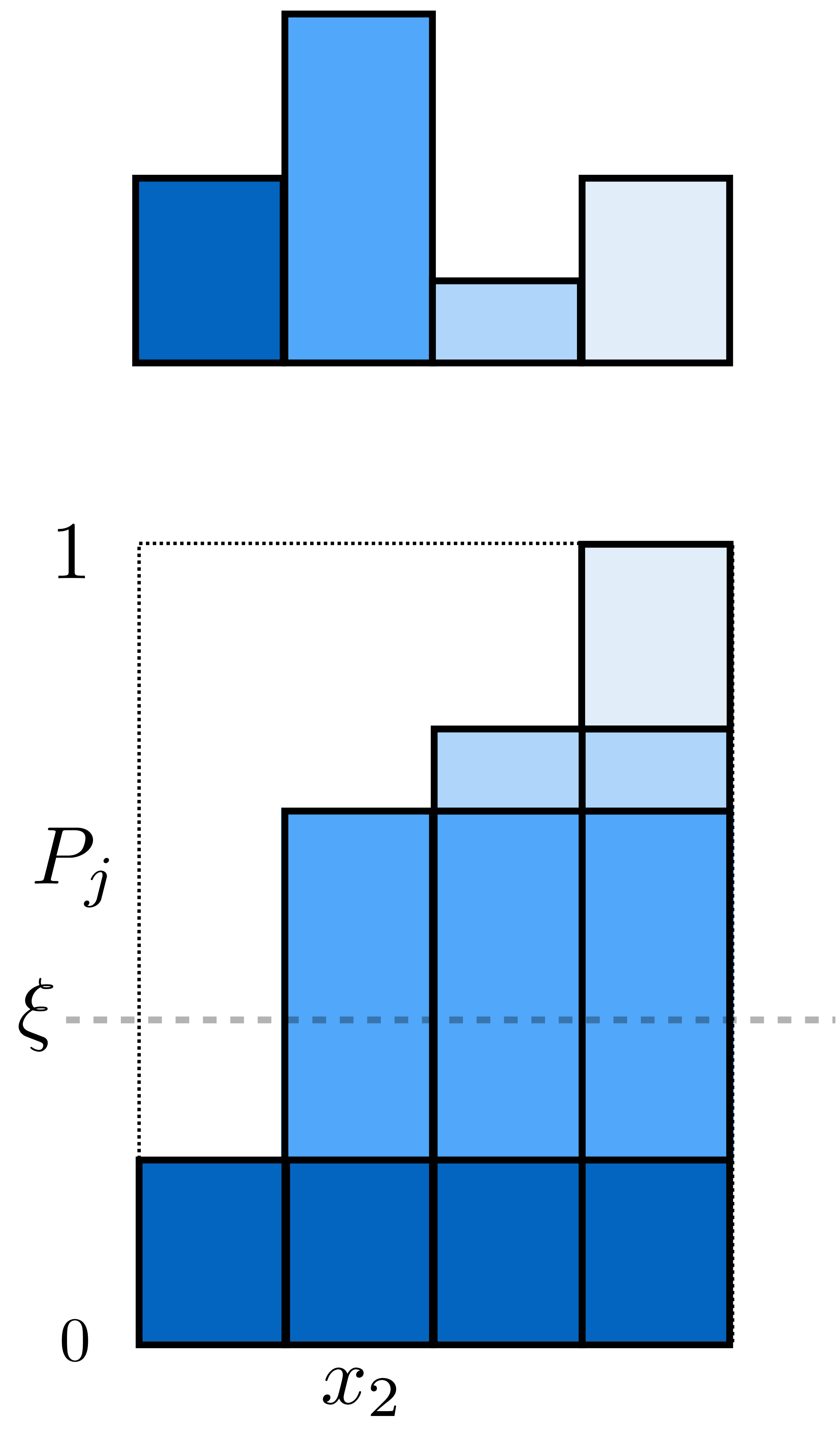
How do we generate samples of a discrete random variable (with a known PDF?)

Sampling from discrete probability distributions

To randomly select an event, select x_i if

$$P_{i-1} < \xi \leq P_i$$

Uniform random variable $\in [0, 1)$



Continuous probability distributions

PDF $p(x)$

$$p(x) \geq 0$$

CDF $P(x)$

$$P(x) = \int_0^x p(x)\,\mathrm{d}x$$

$$P(x) = \Pr(X < x)$$

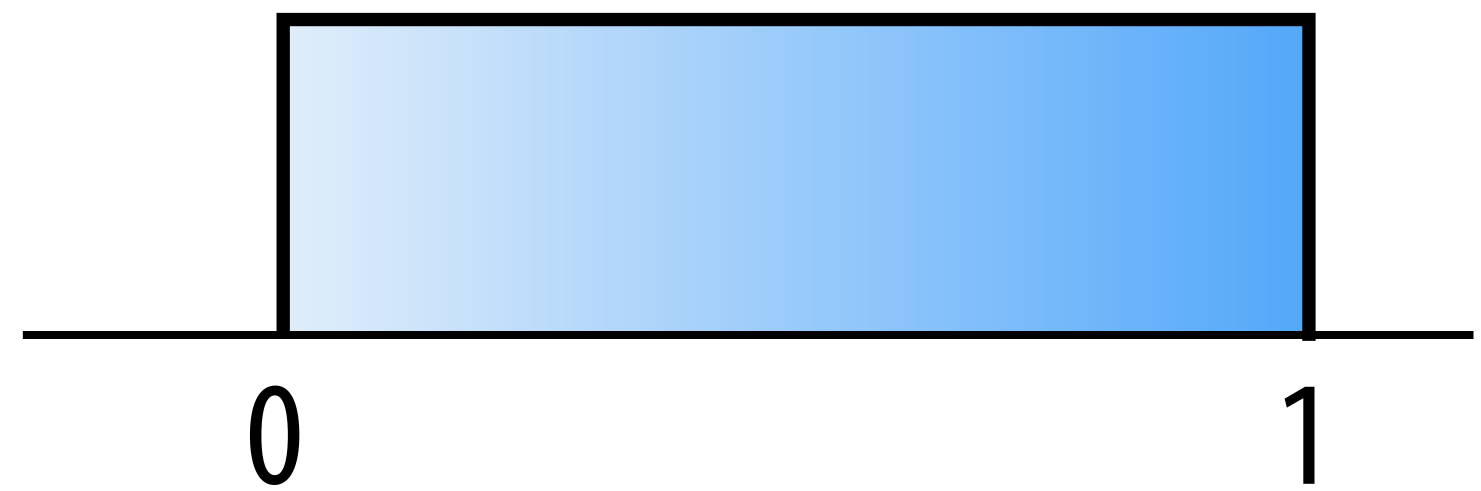
$$P(1) = 1$$

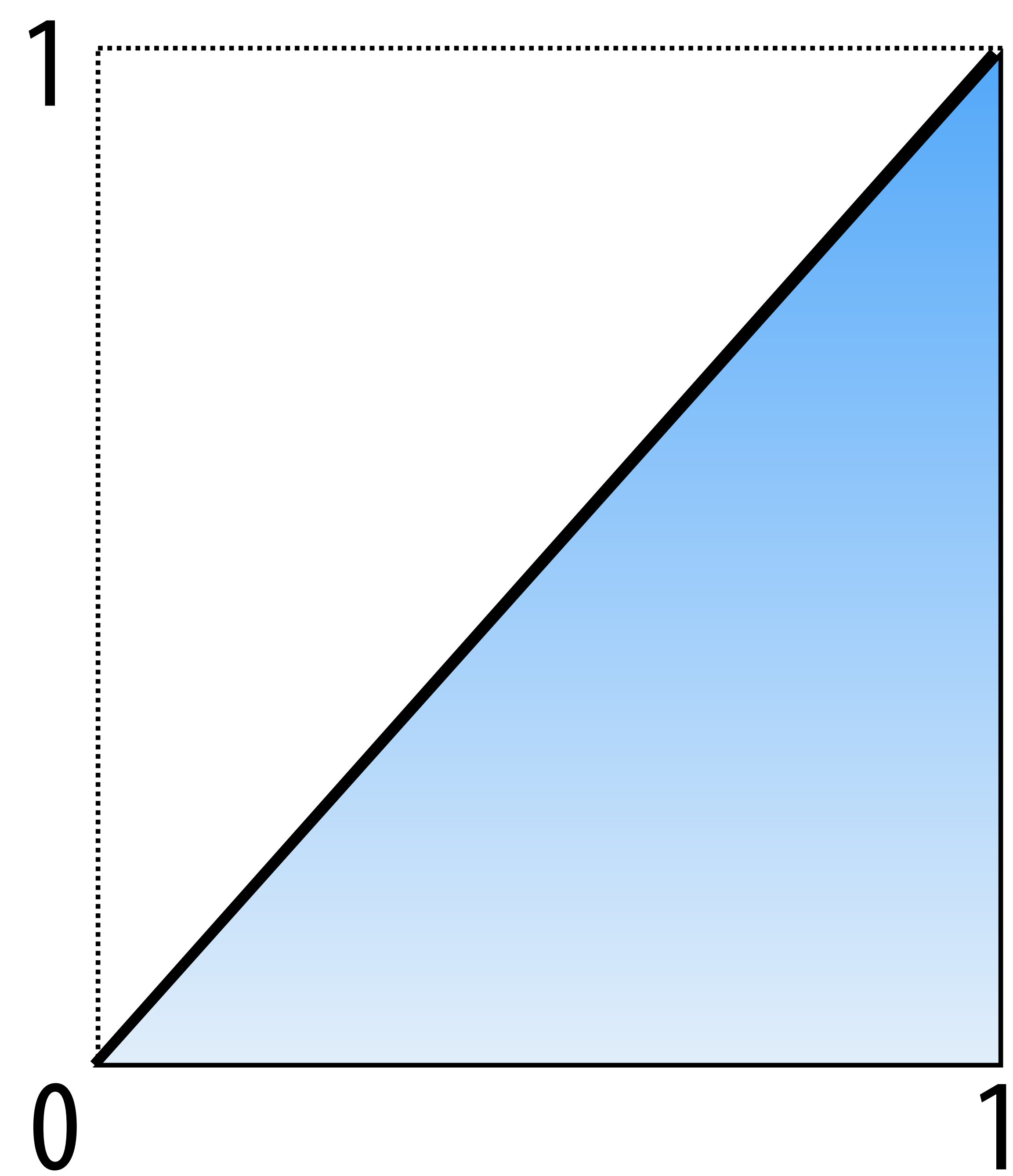
$$\Pr(a \leq X \leq b) = \int_a^b p(x)\,\mathrm{d}x$$

$$= P(b) - P(a)$$

Uniform distribution

(for random variable X defined on [0,1] domain)

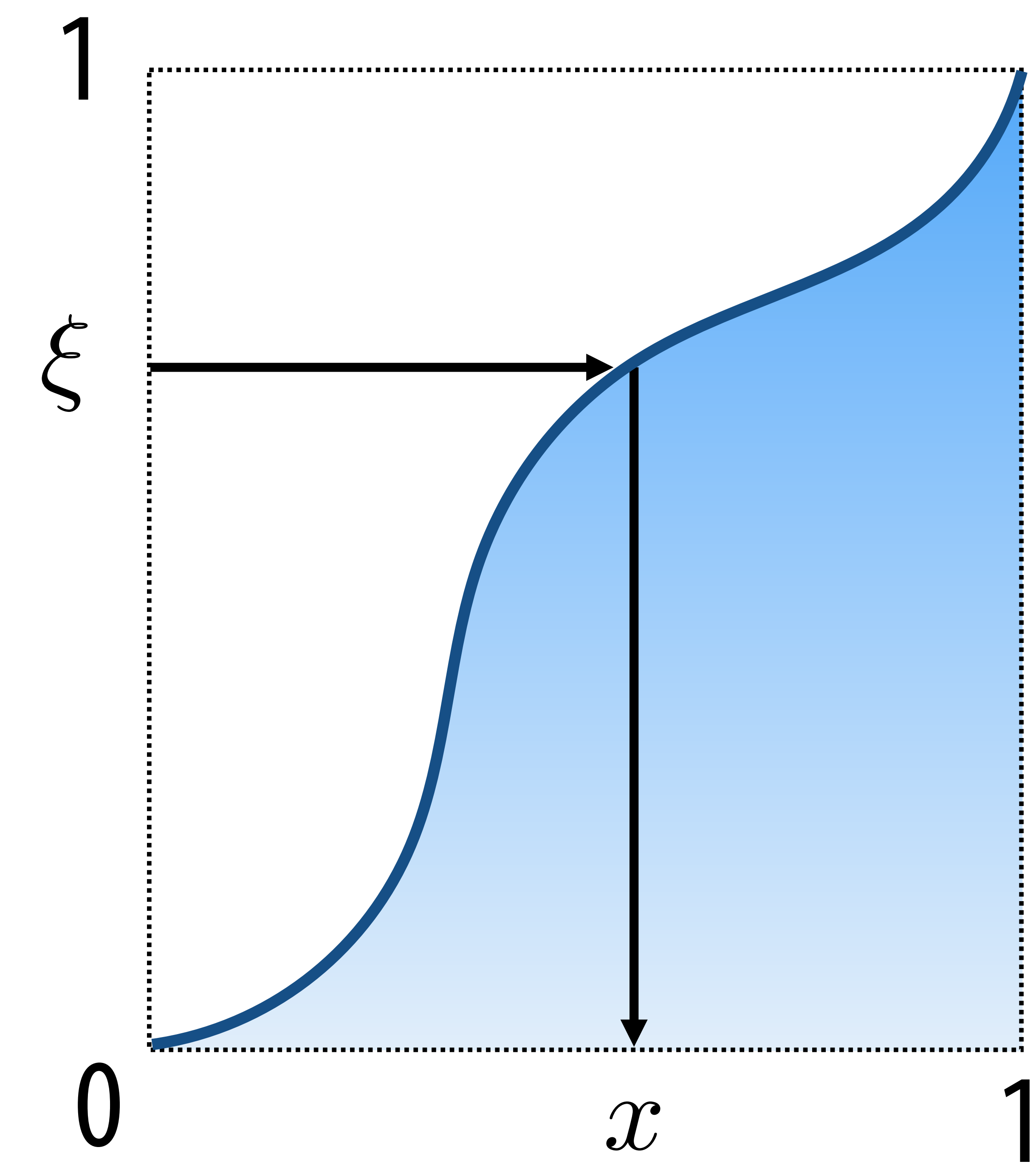




Sampling continuous random variables using the inversion method

Cumulative probability distribution function

$$P(x) = \Pr(X < x)$$



Construction of samples:

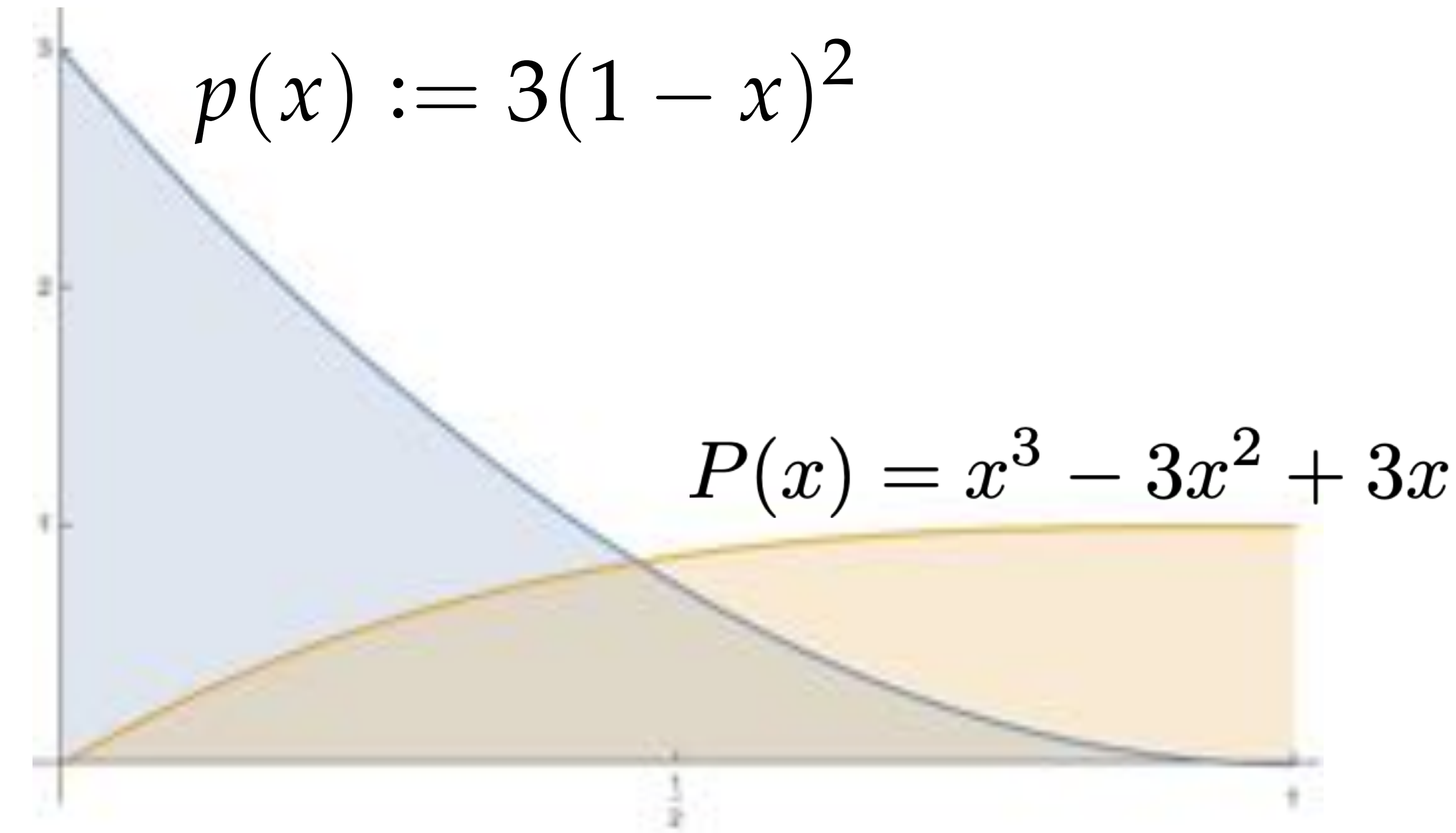
Solve for $x = P^{-1}(\xi)$

Must know the formula for:

1. **The integral of** $p(x)$
2. **The inverse function** $P^{-1}(\xi)$

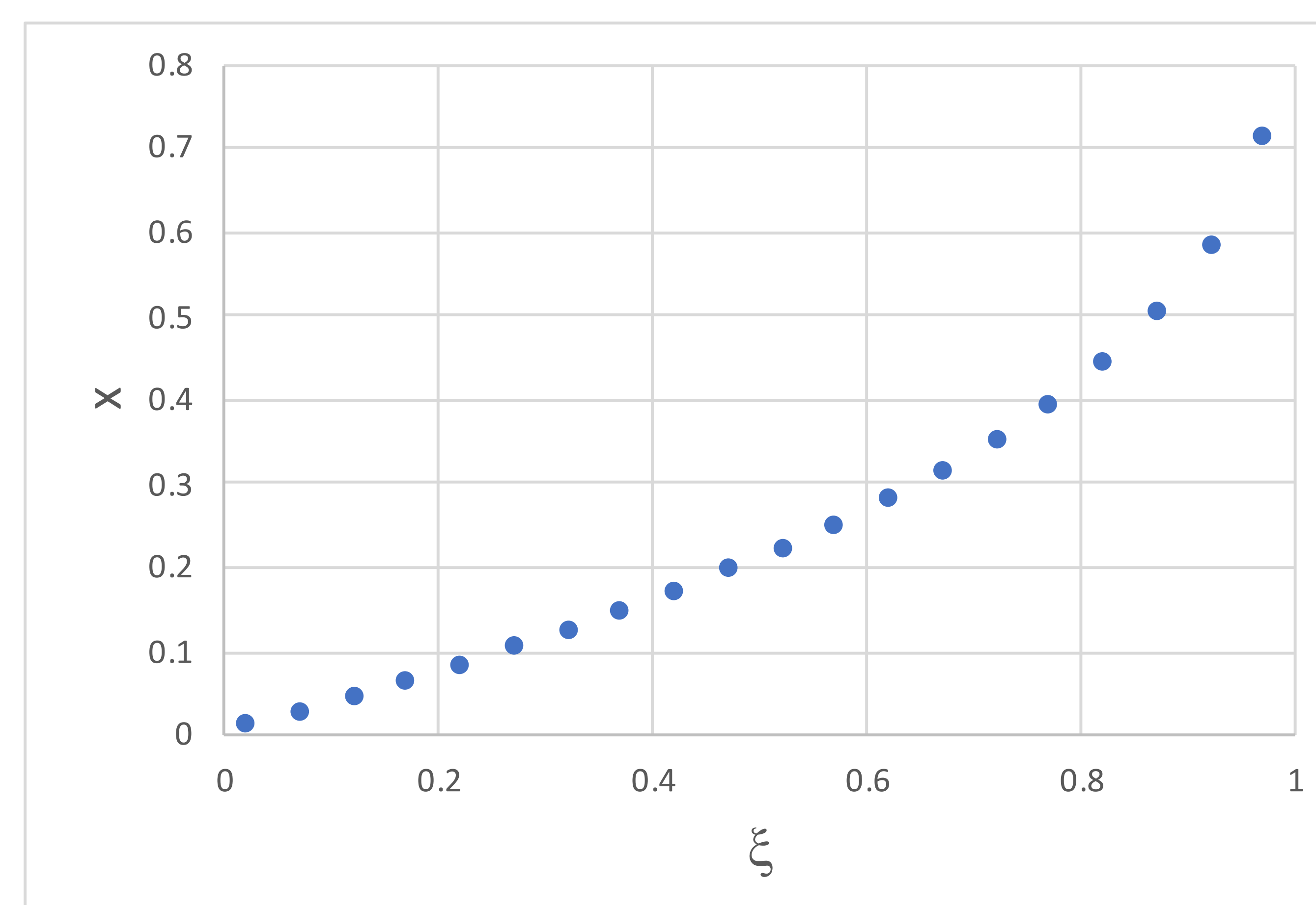
Example—Sampling Quadratic Distribution

- As a toy example, consider the simple probability distribution p(x) := 3(1-x)^2 over the interval [0,1]
- How do we pick random samples distributed according to p(x)?
- First, integrate probability distribution p(x) to get cumulative distribution P(x)
- Invert P(x) by solving $\xi = P(x)$ for x
- Finally, plug uniformly distributed random values ξ in [0,1] into this expression

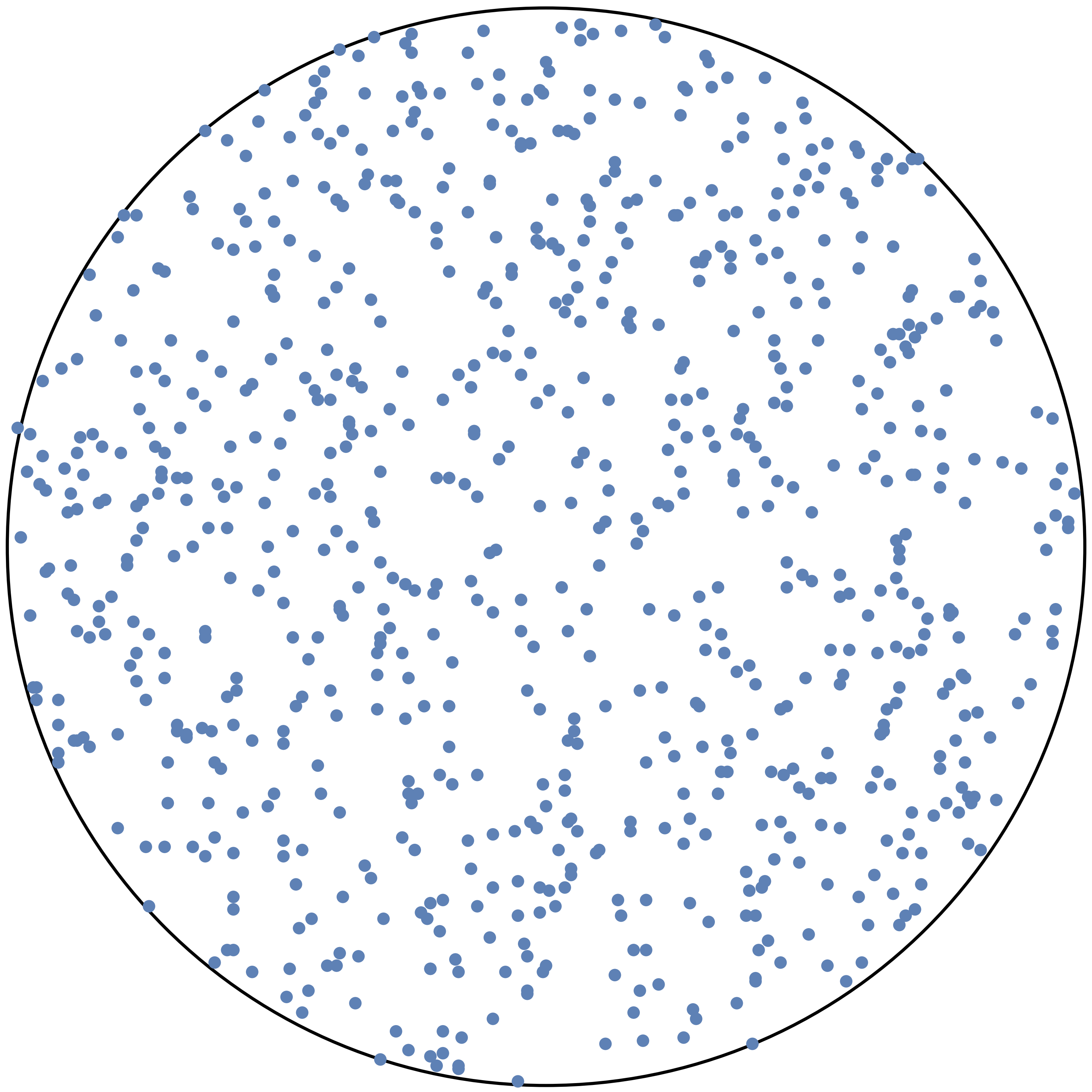


$$\int_0^x 3(1-s)^2 ds = x^3 - 3x^2 + 3x$$

$$x = P^{-1}(\xi) = 1 - (1-\xi)^{\frac{1}{3}}$$



How do we uniformly sample the unit circle?



I.e., choose any point P=(px, py) in circle with equal probability)

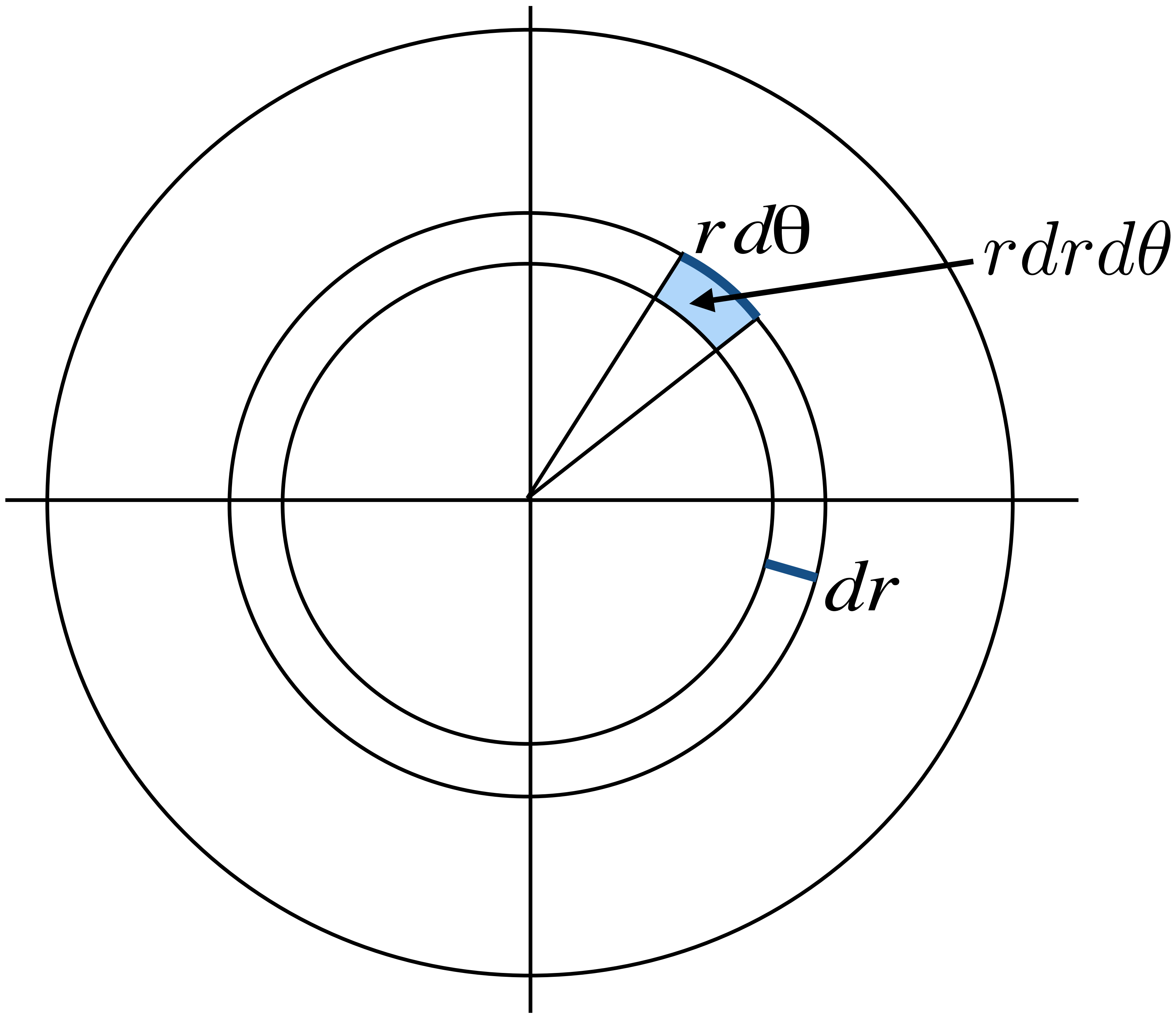
Uniformly sampling unit circle: first try

- θ = uniform random angle between 0 and 2π
- r = uniform random radius between 0 and 1
- Return point: $(r \cos\,\theta, r \sin\,\theta)$

This algorithm <u>does not</u> produce the desired uniform sampling of the area of a circle. Why?

Because sampling is not uniform in area!

Points farther from center of circle are less likely to be chosen



$$\theta = 2\pi\xi_1 \qquad r = \xi_2$$

So how should we pick samples? Well, think about how we integrate over a disk in polar coordinates…

Sampling a circle (via inversion in 2D)

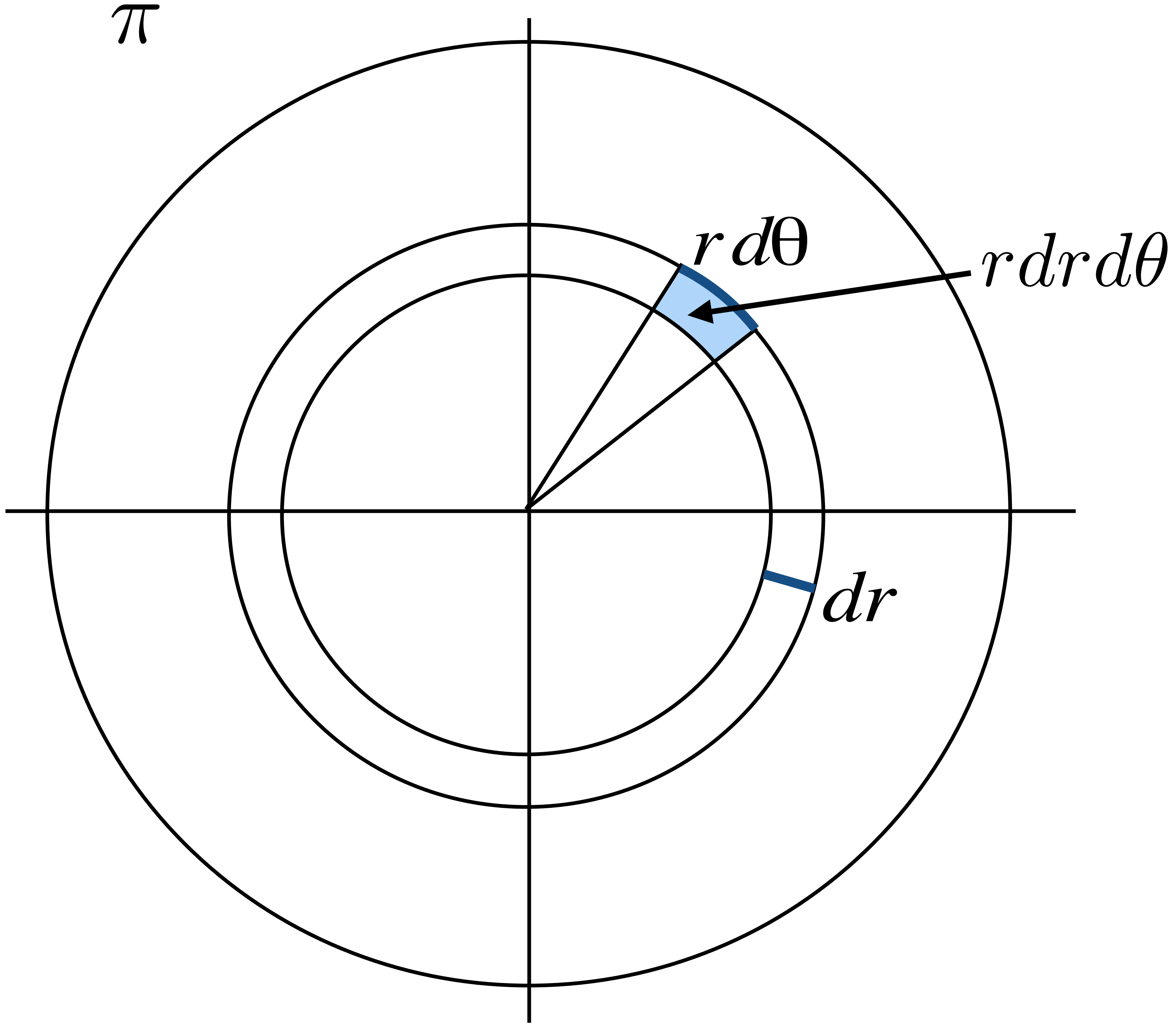
$$A = \int_0^{2\pi}\int_0^1 r\,\mathrm{d}r\,\mathrm{d}\theta = \int_0^1 r\,\mathrm{d}r \int_0^{2\pi} \mathrm{d}\theta = \left(\frac{r^2}{2}\right)\Big|_0^1 \theta\Big|_0^{2\pi} = \pi$$

$$p(r,\theta)\,\mathrm{d}r\,\mathrm{d}\theta = \frac{1}{\pi} r\,\mathrm{d}r\,\mathrm{d}\theta \rightarrow p(r,\theta) = \frac{r}{\pi}$$

so that we integrate to 1 instead of area

$$p(r,\theta) = p(r)p(\theta)$$

← r, θ **independent**



$$p(\theta) = \frac{1}{2\pi}$$

$$P(\theta) = \frac{1}{2\pi}\theta$$

$$\xi_1 = P(\theta) = \frac{\theta}{2\pi}$$

$$\theta = 2\pi\xi_1$$

$$p(r) = 2r$$

$$P(r) = r^2$$

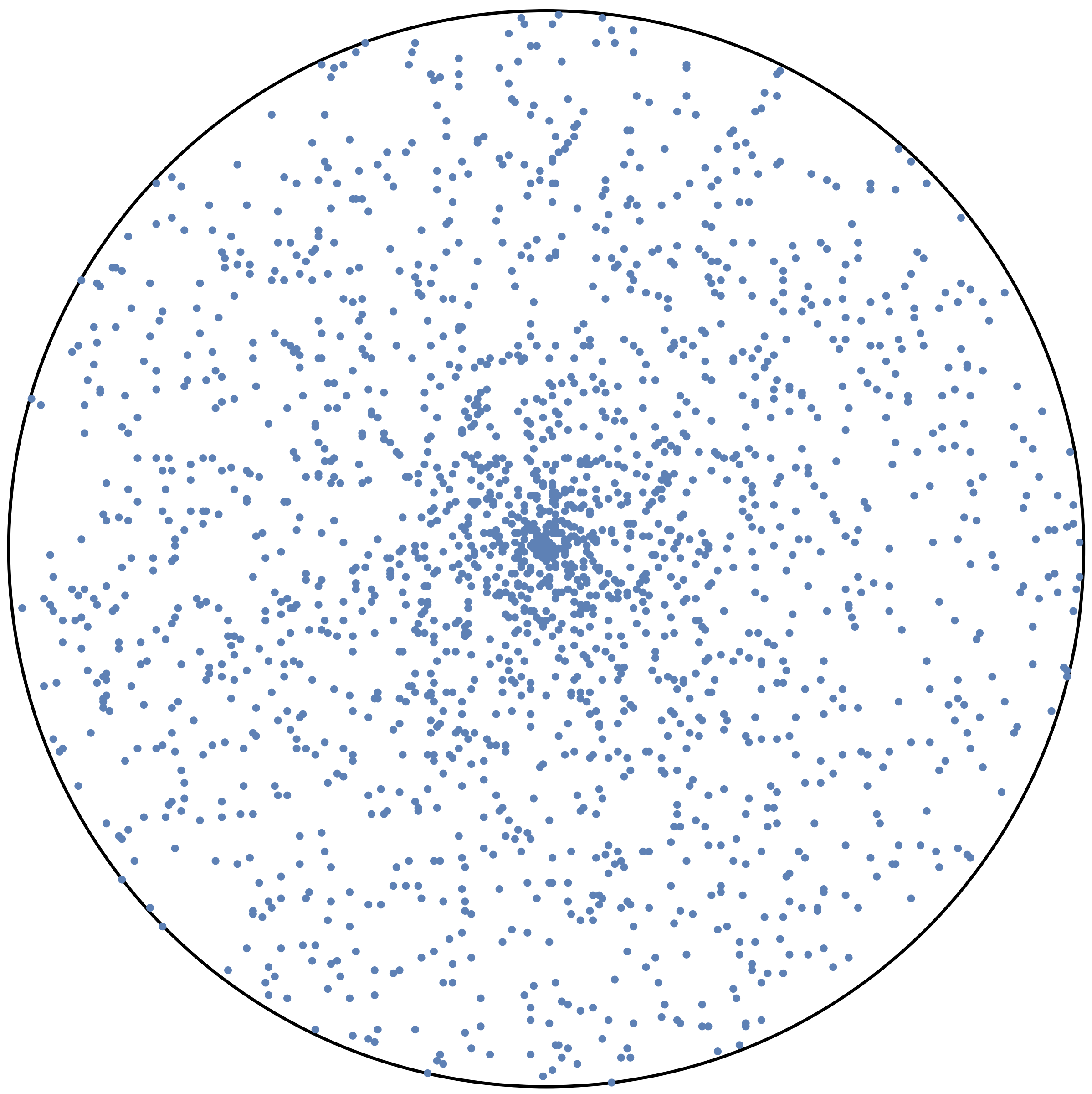
$$\xi_2 = P(r) = r^2$$

$$r = \sqrt{\xi_2}$$

Uniform area sampling of a circle

WRONG

probability is uniform; samples are not!

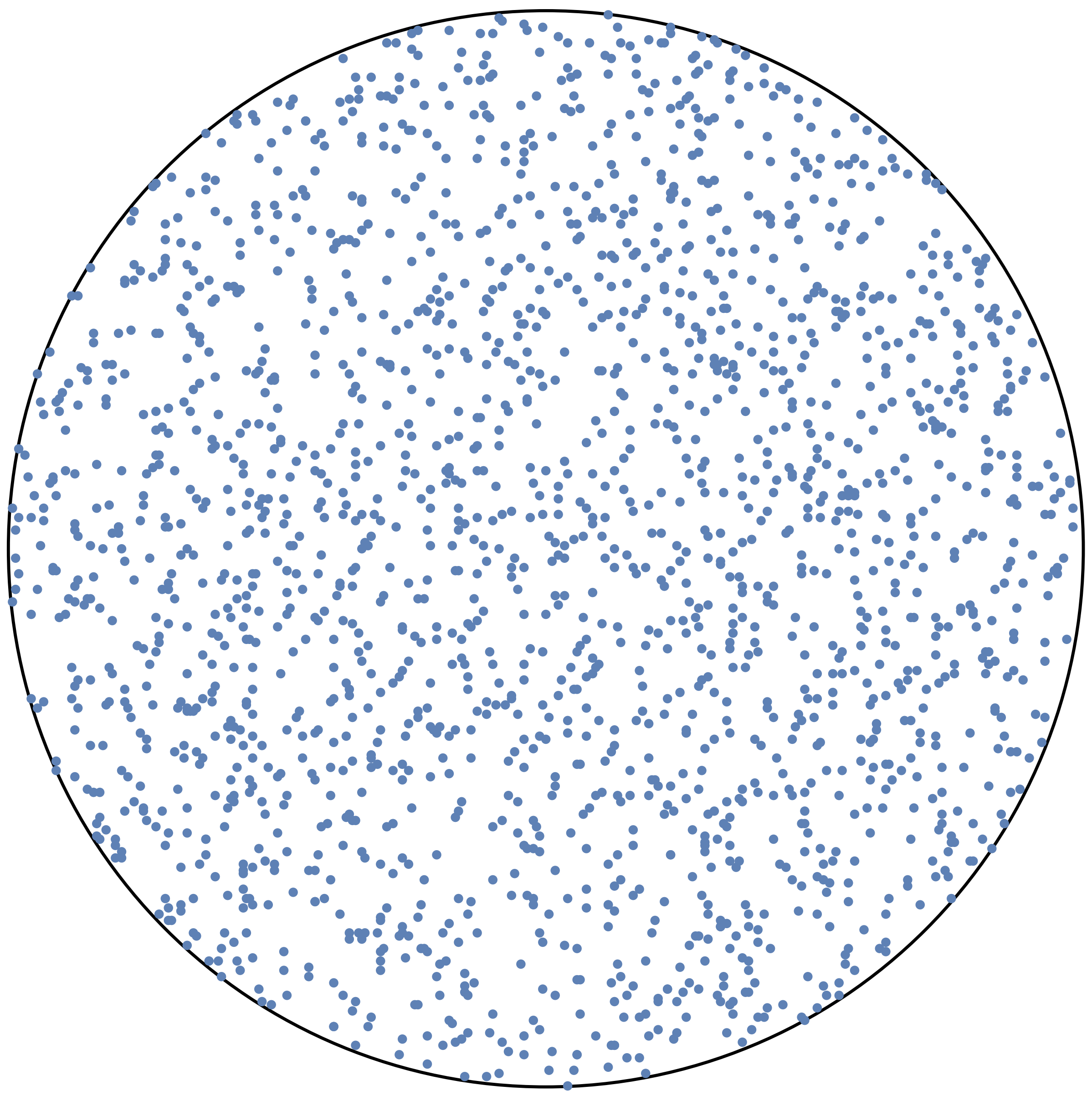


$$\theta = 2\pi\xi_1$$

$$r = \xi_2$$

RIGHT

probability is nonuniform; samples are uniform



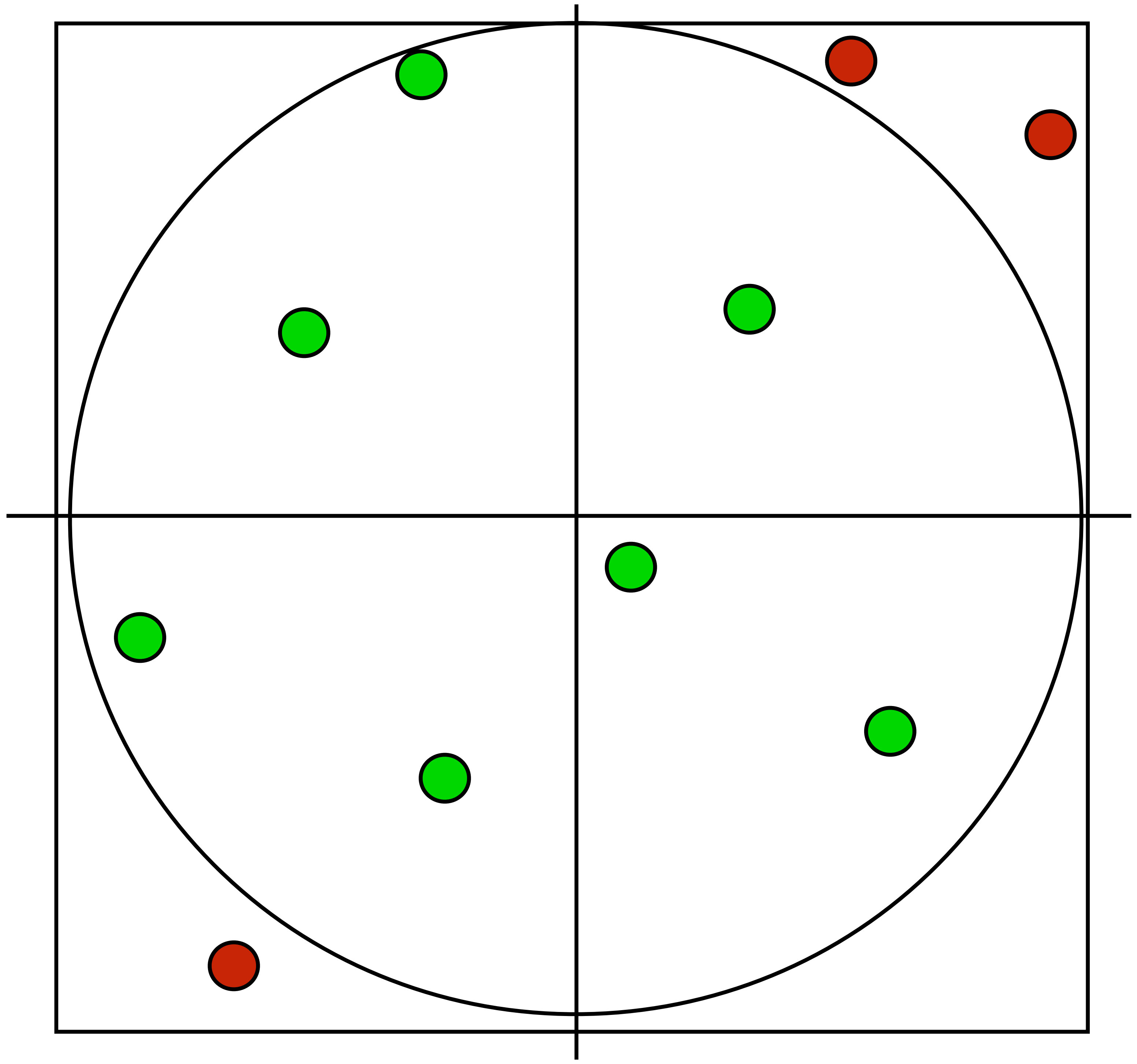
$$\theta = 2\pi\xi_1$$

$$r = \sqrt{\xi_2}$$

Uniform sampling via rejection sampling

Completely different idea: pick uniform samples in square (easy)

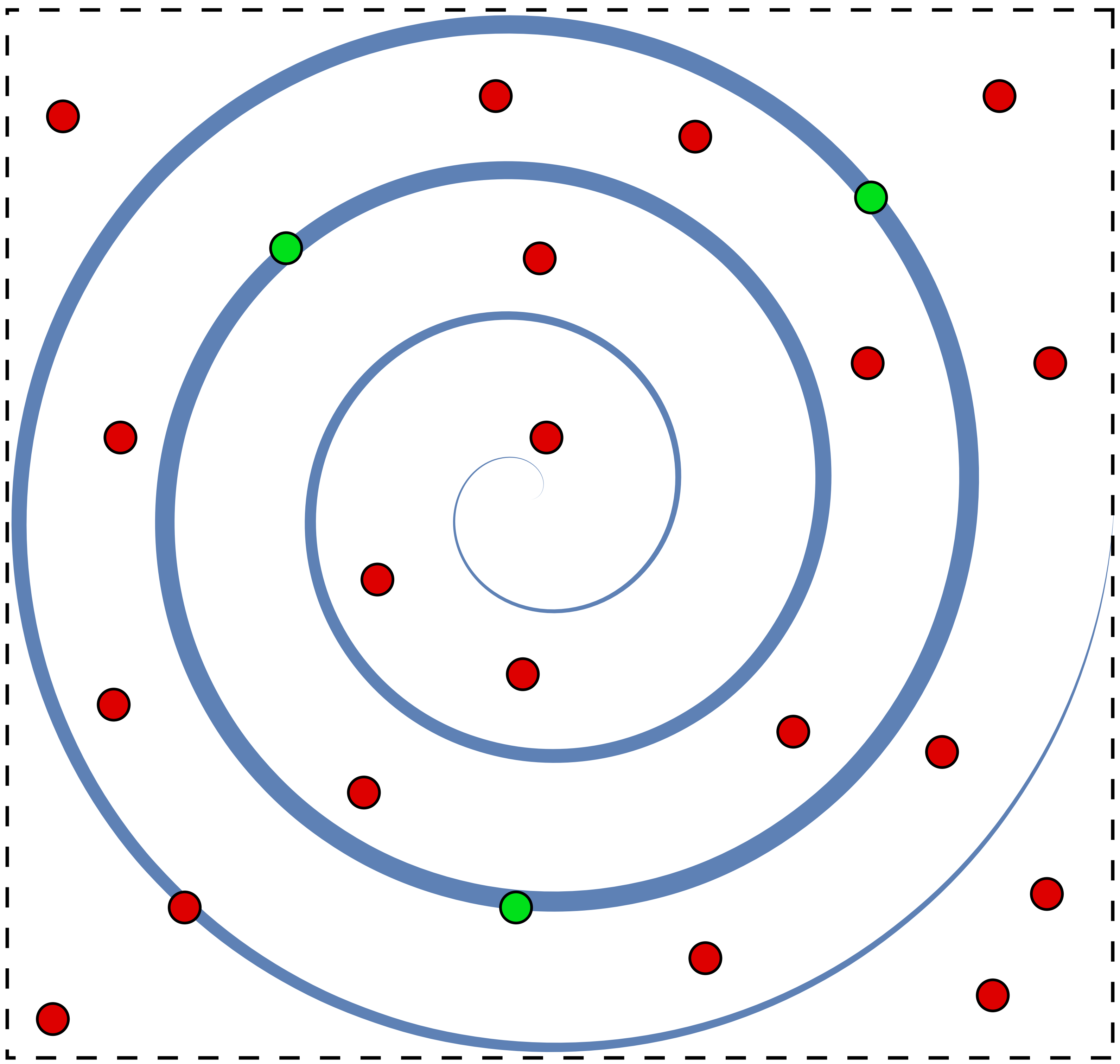
Then toss out any samples not in square (easy)



Efficiency of technique: area of circle / area of square

Efficiency of Rejection Sampling

- **If the region we care about covers only a very small fraction of the region we're sampling, rejection is probably a bad idea:**



Smarter in this case to "warp" our random variables to follow the spiral.

So how do we use numerical integration to do rendering?

Monte Carlo Rendering

- Goal: render a photorealistic image
- Put together many of the ideas we've studied:
  - color
  - materials
  - radiometry
  - numerical integration
  - geometric queries
  - spatial data structures
  - rendering equation

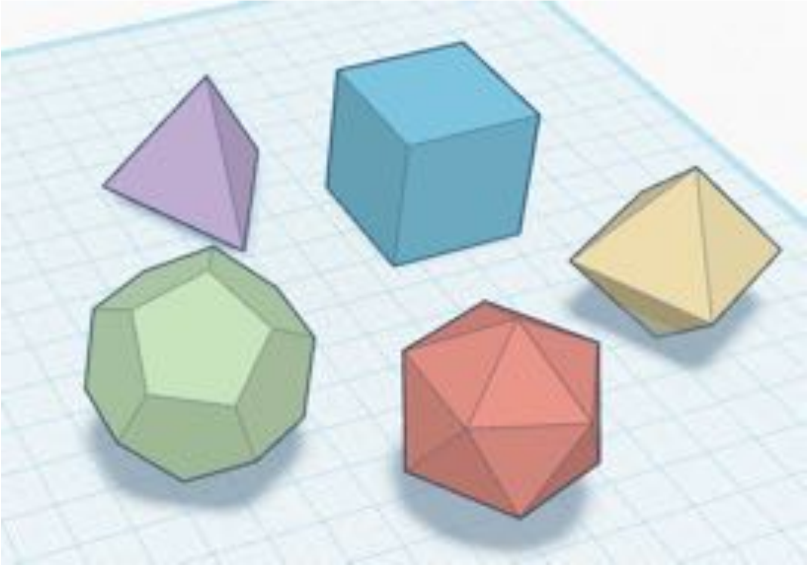
- Combine into final Monte Carlo ray tracing algorithm
- Alternative to rasterization, lets us generate much more realistic images (usually at much greater cost…)

Photorealistic Rendering—Basic Goal

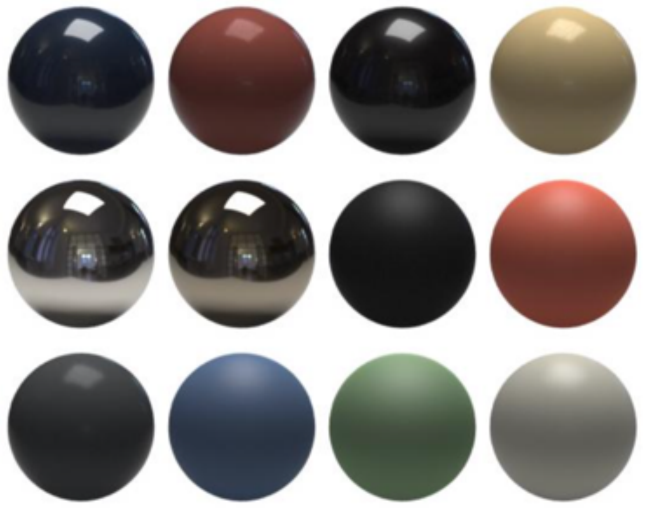
What are the INPUTS and OUTPUTS?

camera

geometry



materials



lights

("scene")

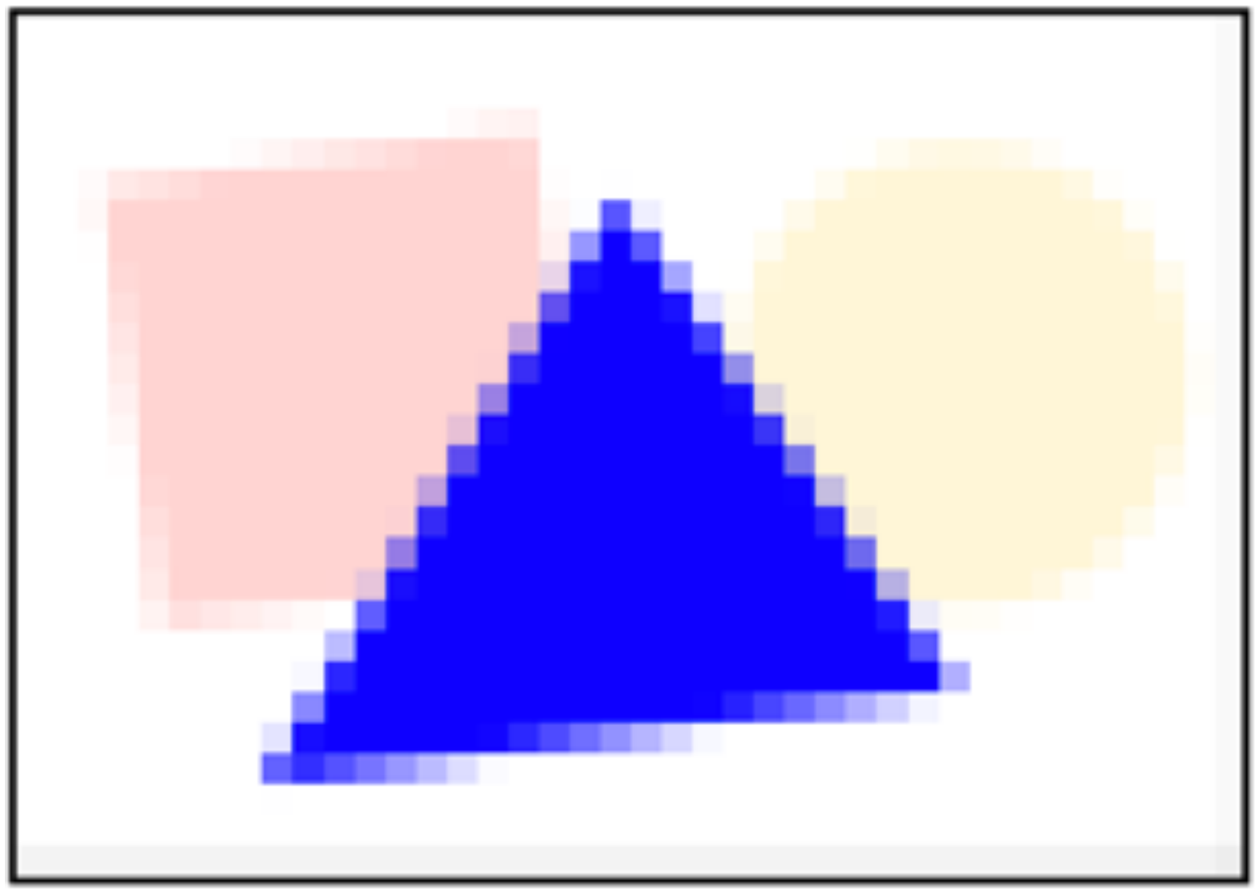
Ray Tracer

image

Ray Tracing vs. Rasterization—Order

- Both rasterization & ray tracing will generate an image
- What's the difference?
- One basic difference: order in which we process samples

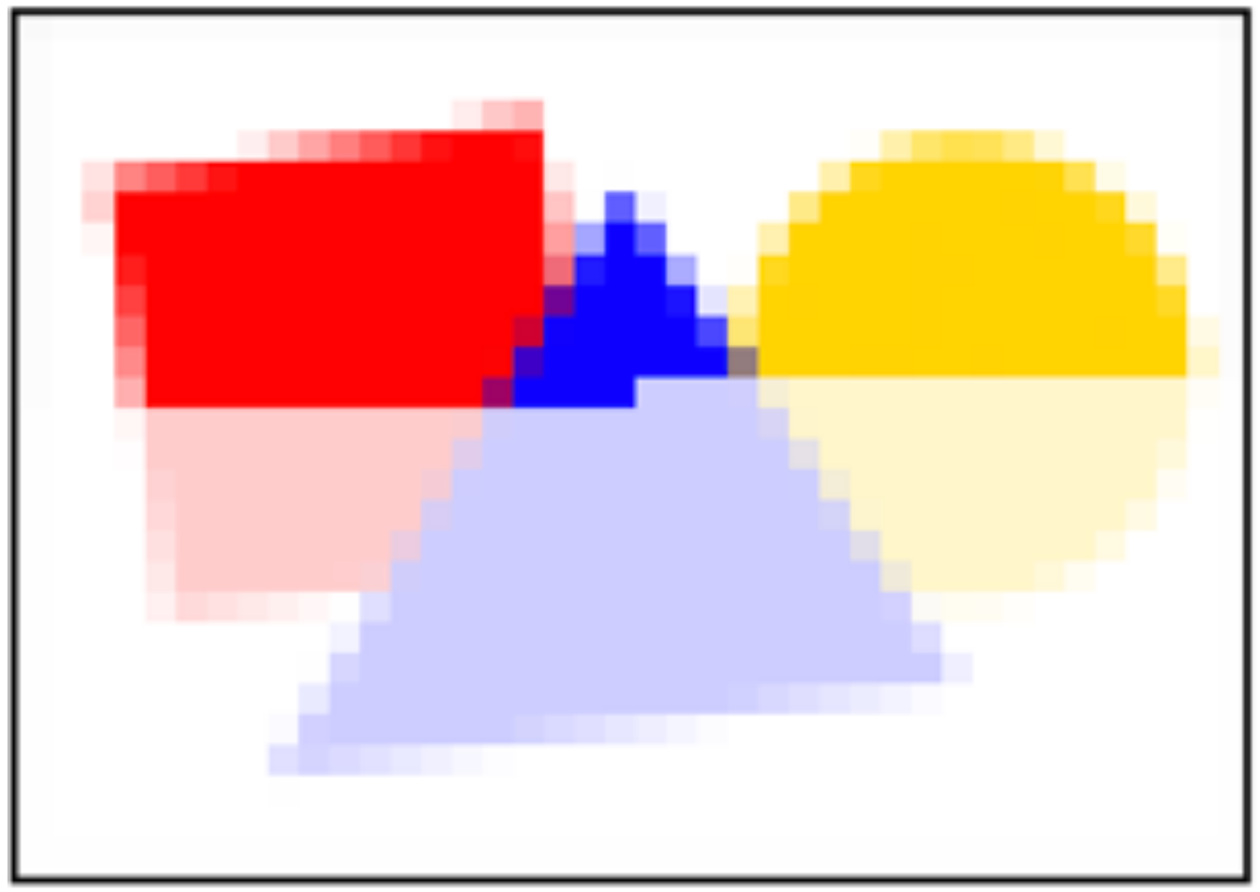
RASTERIZATION



```
for each primitive:
    for each sample:
        determine coverage
        evaluate color
```

(Use Z-buffer to determine which primitive is visible)

RAY TRACING



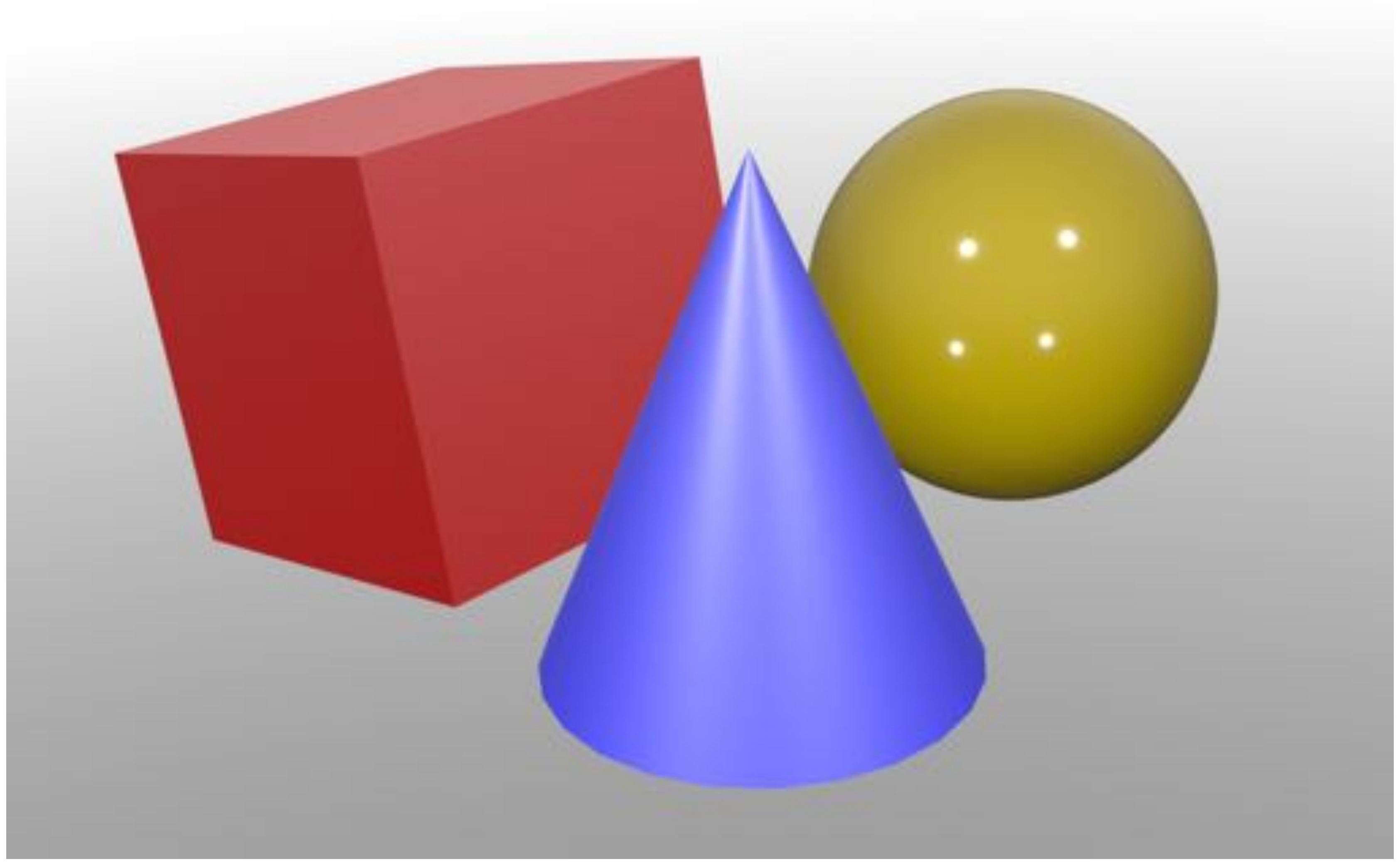
```
for each sample:
    for each primitive:
        determine coverage
        evaluate color
```

(Use spatial data structure like BVH to determine which primitive is visible)

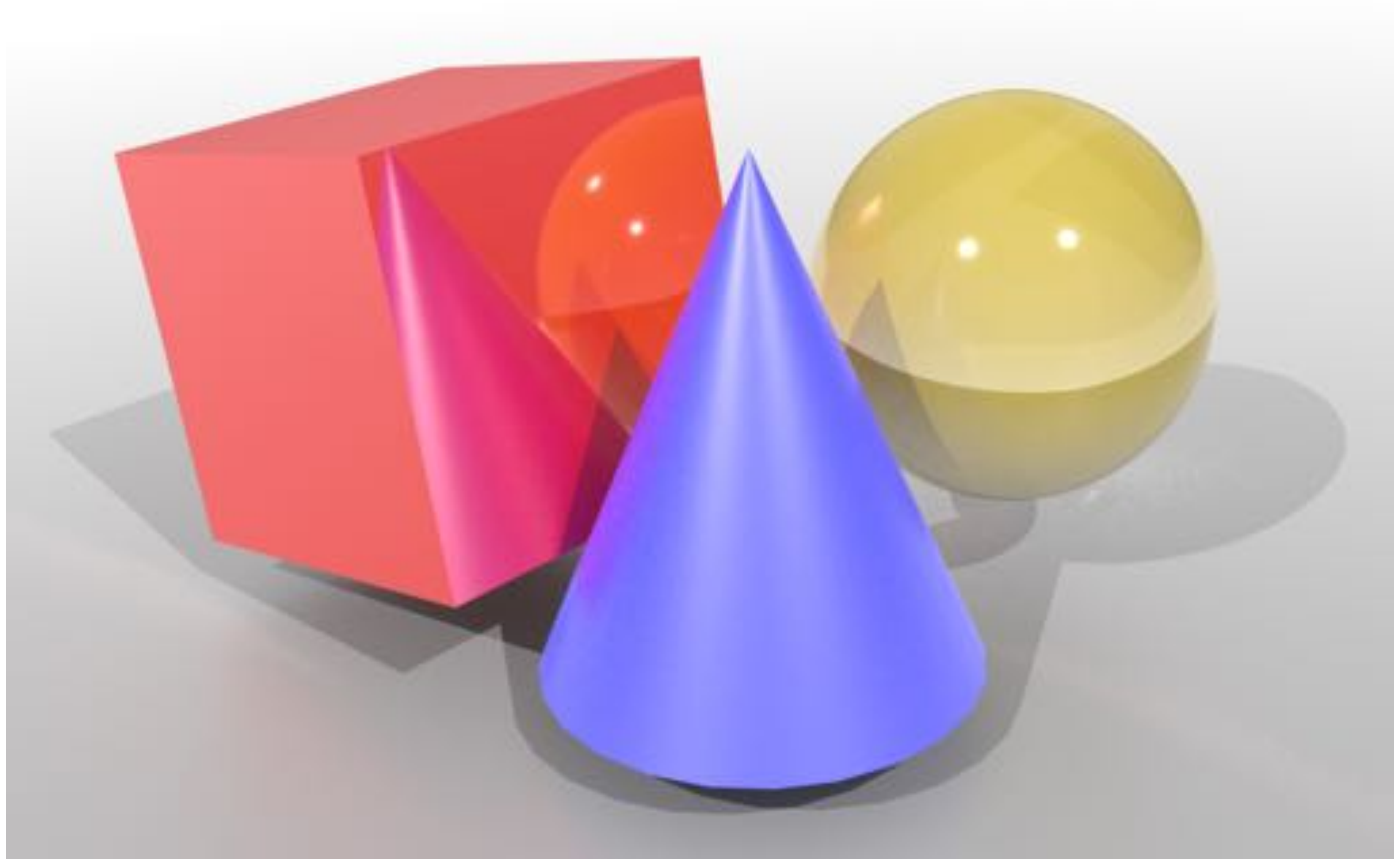
Ray Tracing vs. Rasterization—Illumination

- More major difference: sophistication of illumination model
  - [LOCAL] rasterizer processes one primitive at a time; hard* to determine things like "A is in the shadow of B"
  - [GLOBAL] ray tracer processes on ray at a time; ray knows about everything it intersects, easy to talk about shadows & other "global" illumination effects

RASTERIZATION



RAY TRACING

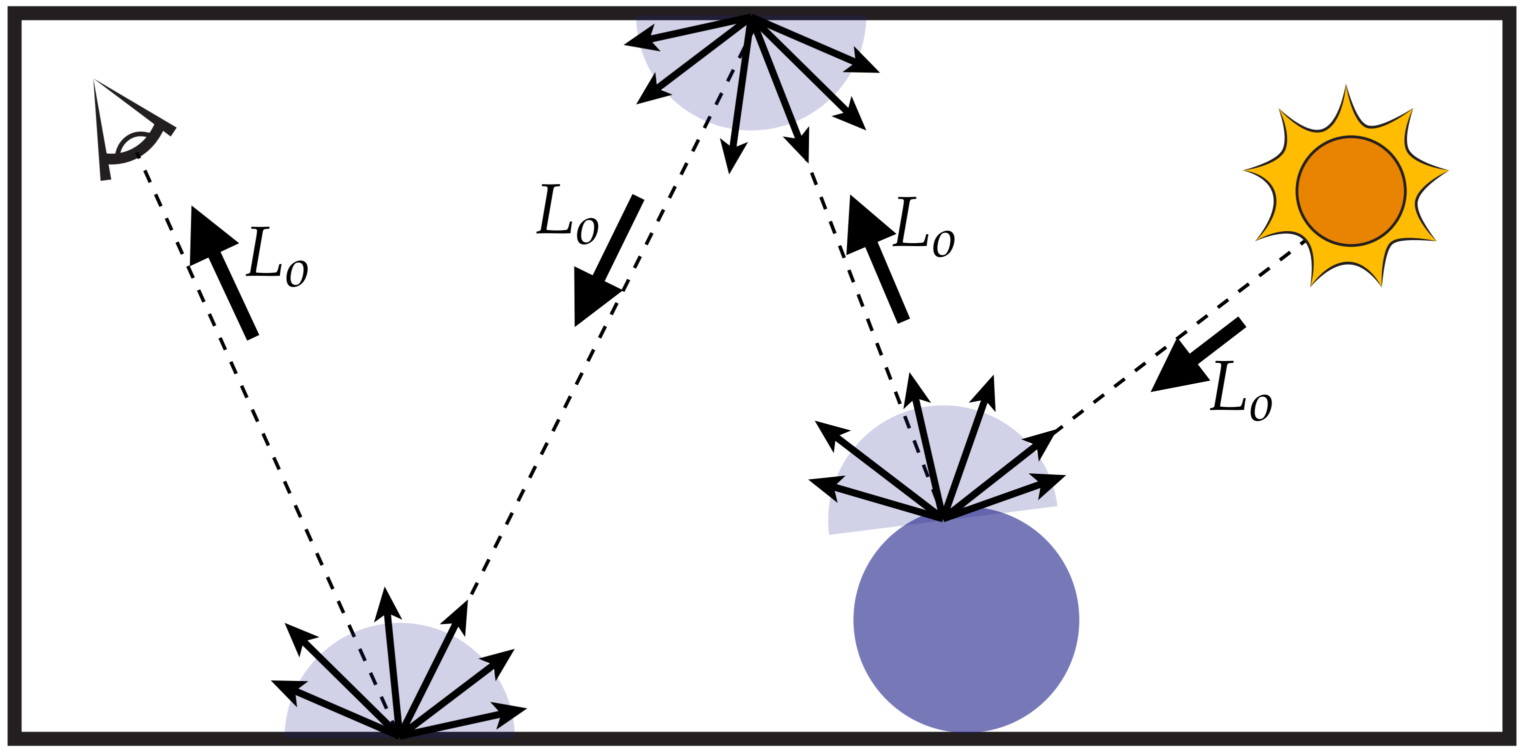


Q: What illumination effects are missing from the image on the left?

*But not impossible to do <u>some</u> things with rasterization (e.g., shadow maps)… just results in more complexity

Monte Carlo Ray Tracing

- To develop a full-blown photorealistic ray tracer, will need to apply Monte Carlo integration to the rendering equation
- To determine color of each pixel, integrate incoming light
- What function are we integrating?
  - illumination along different paths of light
- What does a "sample" mean in this context?
  - each path we trace is a sample



$$L_o(\mathbf{p}, \omega_o) = L_e(\mathbf{p}, \omega_o) + \int_{\mathcal{H}^2} f_r(\mathbf{p}, \omega_i \to \omega_o) L_i(\mathbf{p}, \omega_i) \cos\theta \, d\omega_i$$

Monte Carlo Integration

- Started looking at Monte Carlo integration in our lecture on numerical integration
- Basic idea: take average of random samples
- Will need to flesh this idea out with some key concepts:
  - EXPECTED VALUE — what value do we get on average?
  - VARIANCE — what's the expected deviation from the average?
  - IMPORTANCE SAMPLING — how do we (correctly) take more samples in more important regions?

$$\boxed{\lim_{N\to\infty} \frac{|\Omega|}{N} \sum_{i=1}^{N} f(X_i) = \int_\Omega f(x)\, dx}$$

Expected Value

Intuition: what value does a random variable take, on average?

- E.g., consider a fair coin where heads = 1, tails = 0
- Equal probability of heads & is tails (1/2 for both)
- Expected value is then (1/2)•1 + (1/2)•0 = 1/2

expected value of random variable Y

number of possible outcomes

$$E(Y) := \sum_{i=1}^{k} p_i y_i$$

value of ith outcome

probability of ith outcome

Properties of expectation:

$$E\left[\sum_i Y_i\right] = \sum_i E[Y_i]$$

$$E[aY] = aE[Y]$$

(Can you show these are true?)

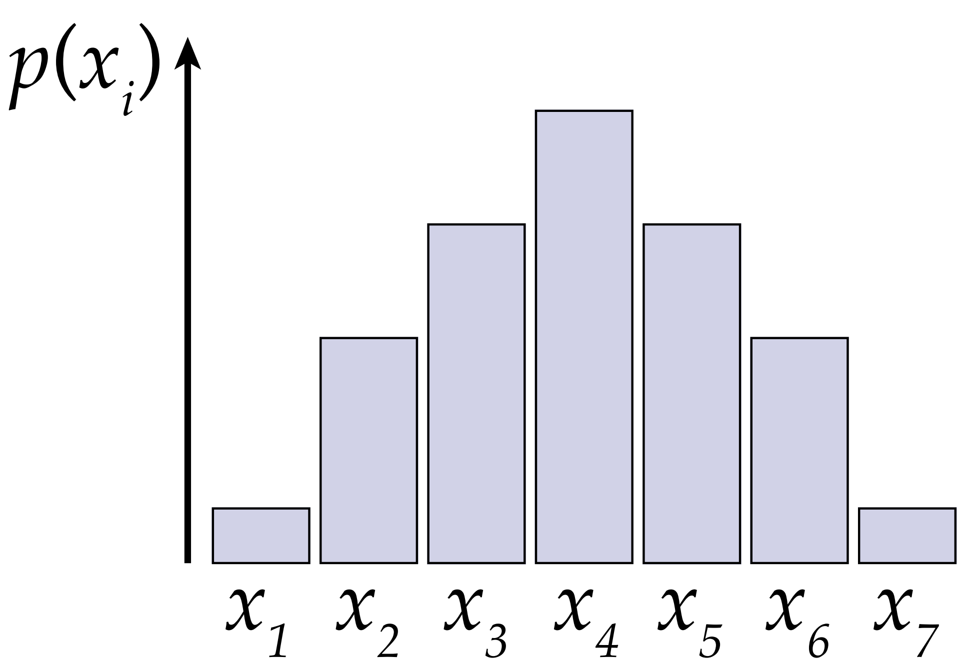
Variance

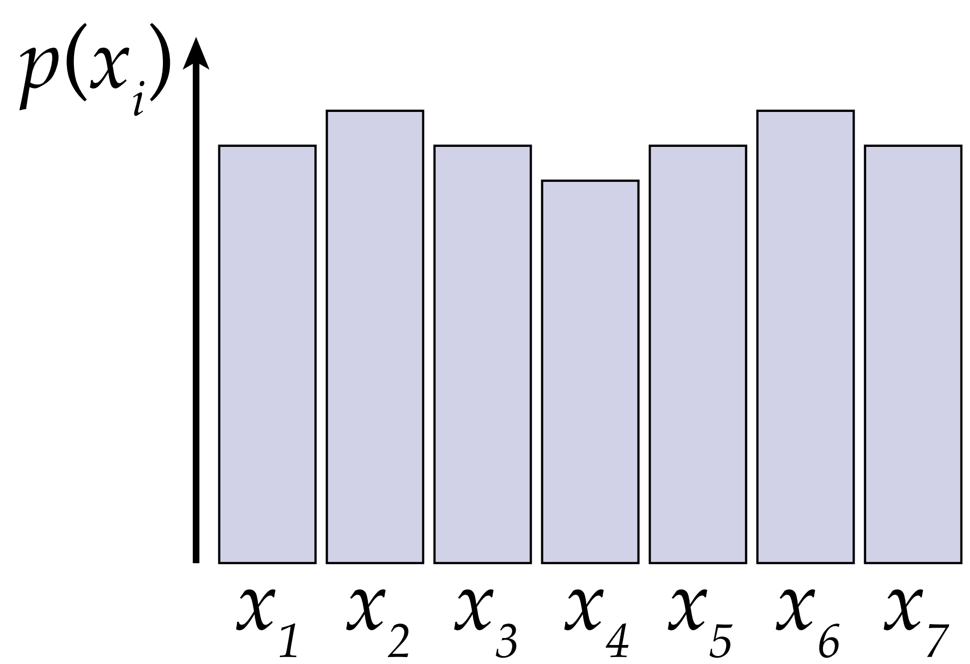
Intuition: how far are our samples from the average, on average?

Definition

$$V[Y] = E[(Y - E[Y])^2]$$

Q: Which of these has higher variance?





Properties of variance:

$$V[Y] = E[Y^2] - E[Y]^2$$

$$V\left[\sum_{i=1}^{N} Y_i\right] = \sum_{i=1}^{N} V[Y_i]$$

$$V[aY] = a^2 V[Y]$$

(Can you show these are true?)

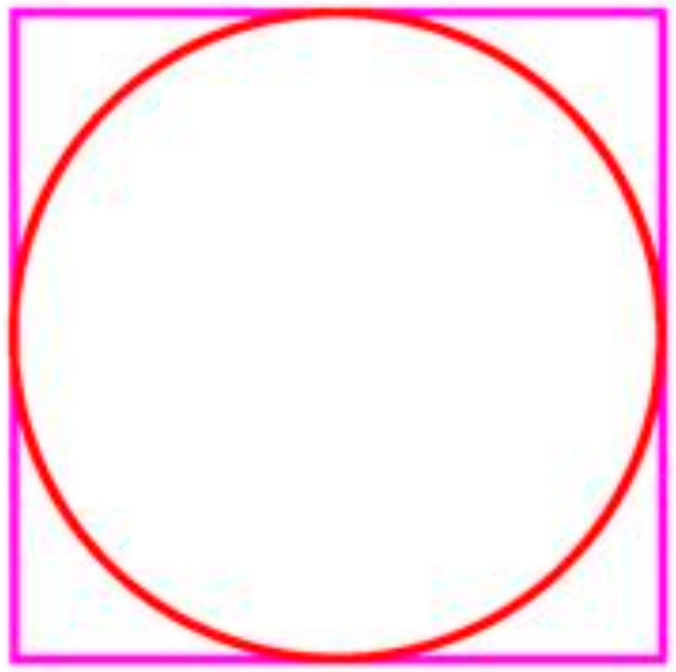
Law of Large Numbers

- **Important fact: for any random variable, the average value of N trials approaches the expected value as we increase N**
- **Decrease in variance is always linear in N:**

$$V\left[\frac{1}{N}\sum_{i=1}^{N} Y_i\right] = \frac{1}{N^2}\sum_{i=1}^{N} V[Y_i] = \frac{1}{N^2} N\, V[Y] = \frac{1}{N} V[Y]$$

Consider a coconut…

nCoconuts	estimate of π
1	4.000000
10	3.200000
100	3.240000
1000	3.112000
10000	3.163600
100000	3.139520
1000000	3.141764



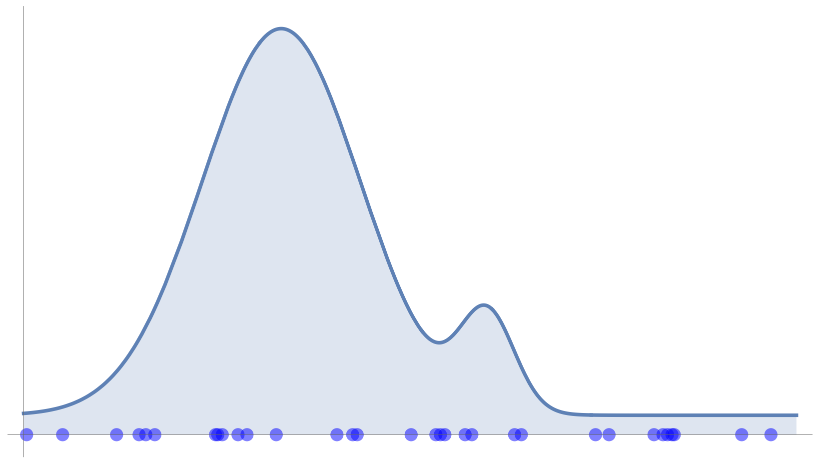
Q: Why is the law of large numbers important for Monte Carlo ray tracing?

A: No matter how hard the integrals are (crazy lighting, geometry, materials, etc.), can always* get the right image by taking more samples.

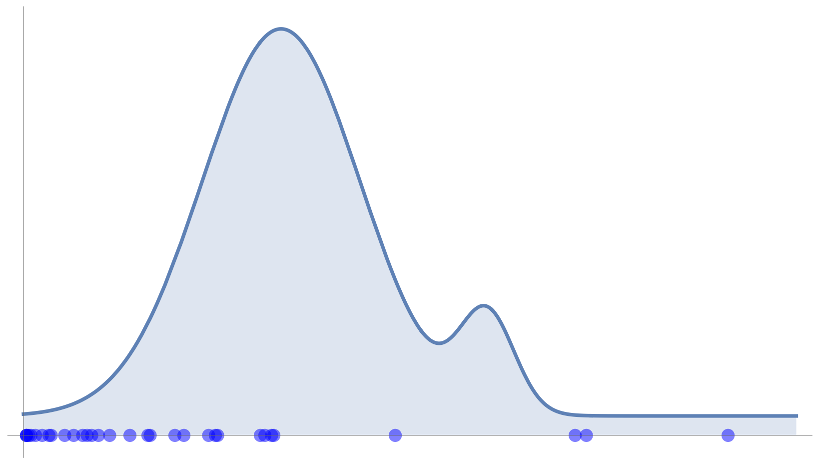
*As long as we make sure to sample all possible kinds of light paths…

Biasing

- So far, we've picked samples uniformly from the domain (every point is equally likely)
- Suppose we pick samples from some other distribution (more samples in one place than another)
- Q: Can we still use samples f(Xi) to get a (correct) estimate of our integral?
- A: Sure! Just weight contribution of each sample by how likely we were to pick it
- Q: Are we correct to divide by p? Or… should we multiply instead?
- A: Think about a simple example where we sample RED region 8x as often as BLUE region
  - average color over square should be purple
  - if we multiply, average will be TOO RED
  - if we divide, average will be JUST RIGHT

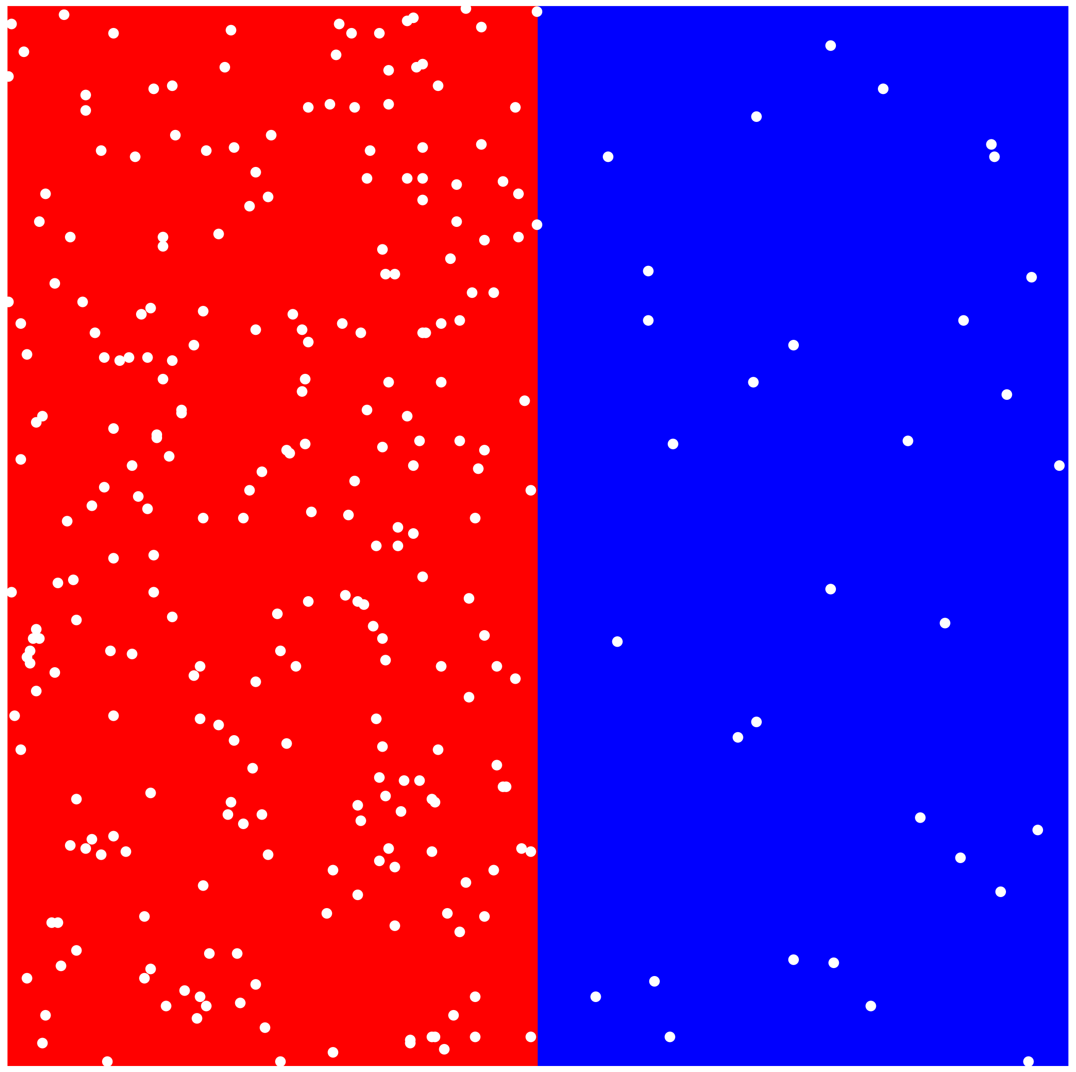


(uniform)



$X_i \sim p$

$$\int_\Omega f(x)\,dx \approx \frac{1}{N}\sum_{i=1}^{N}\frac{f(X_i)}{p(X_i)}$$



Next Time: Use biasing for Importance Sampling, along with other aspects of effective Monte Carlo Raytracing!