# High Dimensional Integration

M.Sc. in Mathematical Modelling & Scientific Computing,
Practical Numerical Analysis

Michaelmas Term 2019, Lecture 8

# Integration

Suppose we want to compute

$$I(f) \quad = \quad \int_{\Omega} f(\mathbf{x}) \mathrm{d}\mathbf{x}$$

where $\Omega \subset \mathbb{R}^d$ and $\mathbf{x} = (x_1, x_2, \ldots, x_d)$.

We know how to do this in 1D using one of the quadrature rules discussed in Lecture 3. Then

$$I(f) \quad \approx \quad I_n(f) \; = \; \sum_{k=0}^{n} w_k f(x_k) \; .$$

In dD if $\Omega$ is a hypercube, e.g. $\Omega = (0,1)^d$, we could compute using tensor product rules. So in 2D we could use

$$I(f) \quad \approx \quad I_n(f) \; = \; \sum_{k,\ell=0}^{n} w_k w_\ell f(x_k, x_\ell) \; .$$

# Integration

The trouble is that, as $d$ grows, so does the number of function evaluations required — we need to evaluate $f(\mathbf{x})$ at $N := (n+1)^d$ points.

Then, assuming the function to be integrated is smooth, we have

$$\text{error} \;=\; \mathcal{O}\left(\frac{1}{n^2}\right) \;=\; \mathcal{O}\left(\frac{1}{N^{2/d}}\right) \;=\; \mathcal{O}(N^{-2/d})\,.$$

# 2D Example

The composite trapezium rule in 1D is

$$\begin{aligned}\int_0^1 g(x)\mathrm{d}x &= \frac{1}{n}\left(g(0) + 2\sum_{k=1}^{n-1} g(x_k) + g(1)\right) \\ &= \sum_{k=0}^{n} w_k g(x_k)\end{aligned}$$

where $x_k = k/n$ for $k = 0, 1, \ldots, n$ and $w_0 = w_n = 1/(2n)$, and $w_k = 1/n$ for $k = 1, \ldots, n-1$.

Thus, in 2D we use

$$I(f) = \int_0^1 \int_0^1 f(x,y)\mathrm{d}x\mathrm{d}y \approx I_n(f) = \sum_{k,\ell=0}^{n} w_k w_\ell f(x_k, x_\ell)\,.$$

# 2D Smooth Example

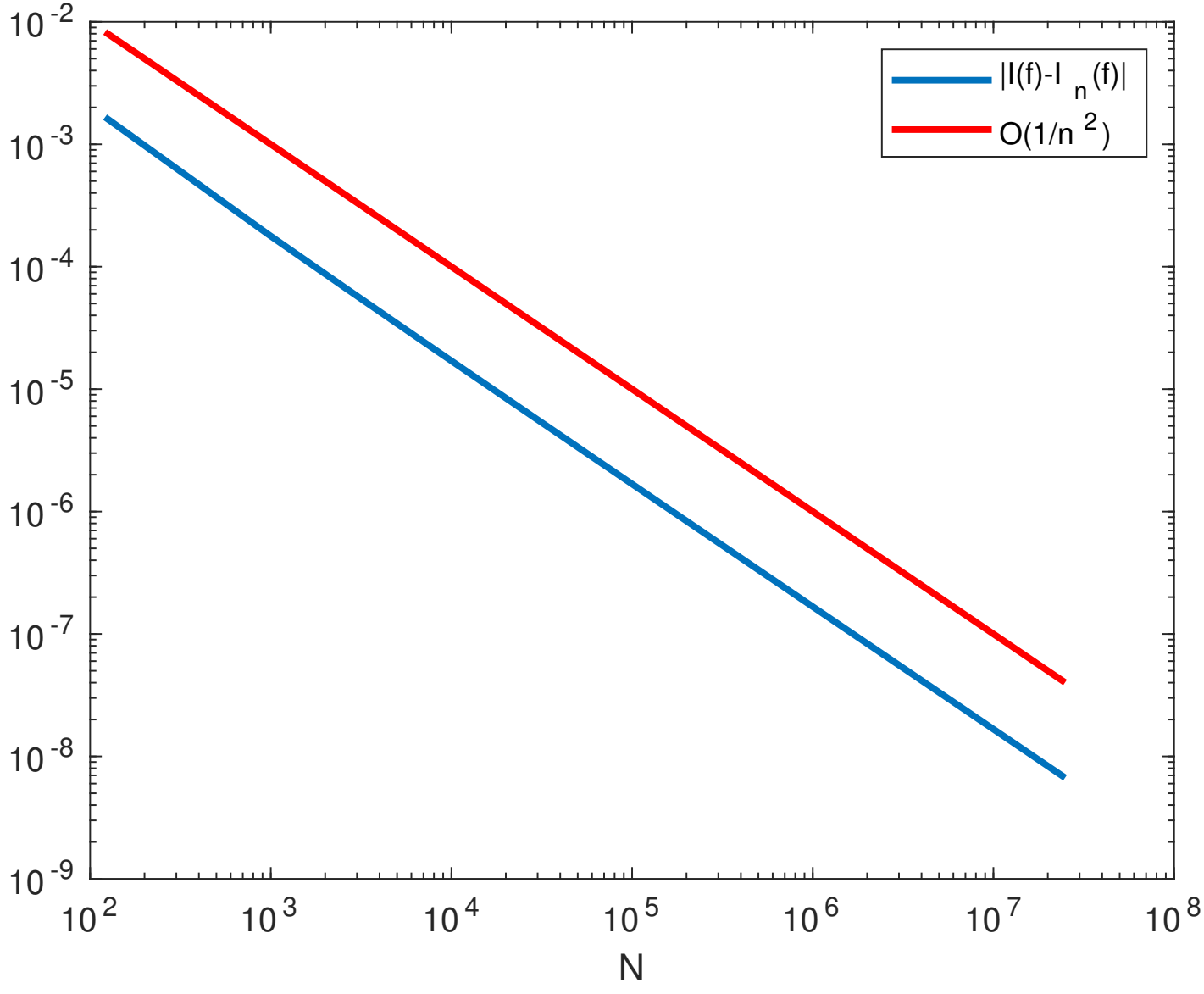
Define

$$f(x,y) \;=\; \cos\left(\frac{\pi x}{2}\right)\cos\left(\frac{\pi y}{2}\right)$$

so that

$$I(f) \;=\; \int_0^1\int_0^1 f(x,y)\mathrm{d}x\mathrm{d}y \;=\; \frac{4}{\pi^2}\,.$$

# 2D Smooth Example



# 2D Non-Smooth Example

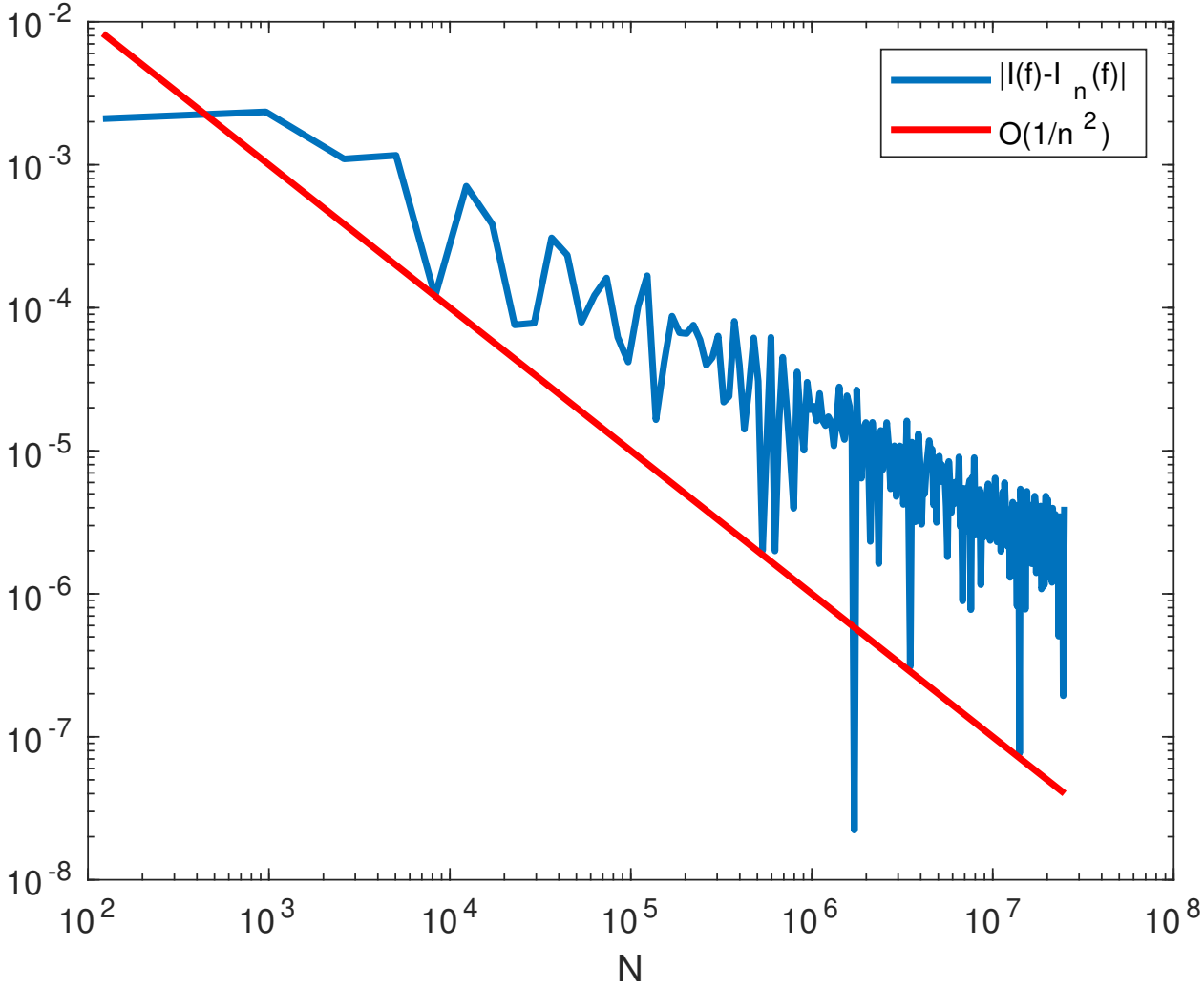
Define

$$f(x,y) = \begin{cases} 1 & 0 \le x^2 + y^2 \le 1 \\ 0 & \text{elsewhere} \end{cases}$$

Then

$$\begin{aligned} I(f) &= \int_0^1 \int_0^1 f(x,y)\mathrm{d}x\mathrm{d}y \\ &= \int_{x^2+y^2\le 1} \chi_{x\ge 0}\chi_{y\ge 0}\mathrm{d}x\mathrm{d}y \\ &= \frac{\pi}{4}\,. \end{aligned}$$

# 2D Non-Smooth Example




Convergence is worse than predicted because $f(x, y)$ is not smooth.

# Alternative Idea: Monte Carlo

Here the idea is that, with $\Omega = (0,1)^d$, we approximate

$$I(f) \approx I_N(f) = \frac{1}{N}\sum_{k=1}^{N} f(\mathbf{x}_k)$$

where $\mathbf{x}_k = (x_{k,1}, x_{k,2}, \ldots, x_{k,d})$ and the $x_{k,i}$ are independent samples from a uniform distribution on $[0,1]$.

Note that this is unbiased so $\mathbb{E}[I_N(f)] = I(f)$.

In addition the law of large numbers ensures that

$$\lim_{N\to\infty} I_N(f) = I(f)\,.$$

# Alternative Idea: Monte Carlo

The Central Limit Theorem proves that for large $N$

$$\epsilon_N := I(f) - I_N(f) \quad \sim \quad \sigma N^{-1/2} Z$$

where $Z \sim N(0,1)$ and

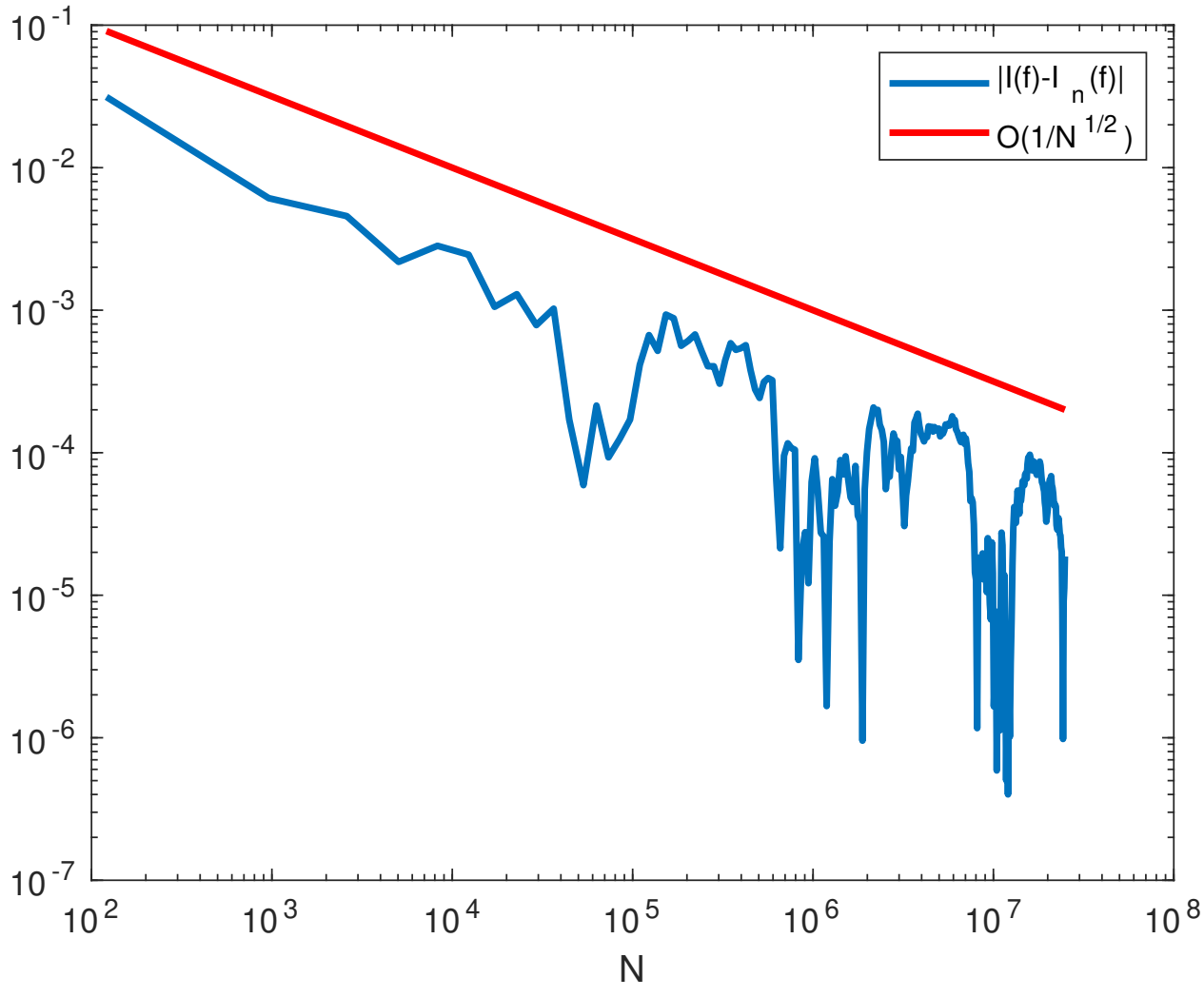
$$\sigma^2 \quad = \quad \mathbb{E}[(f - I(f))^2] \quad = \quad \int_{I^d} (f(\mathbf{x}) - I(f))^2 \mathrm{d}x \; .$$

Hence the error is $\mathcal{O}(N^{-1/2})$ for any $d$.

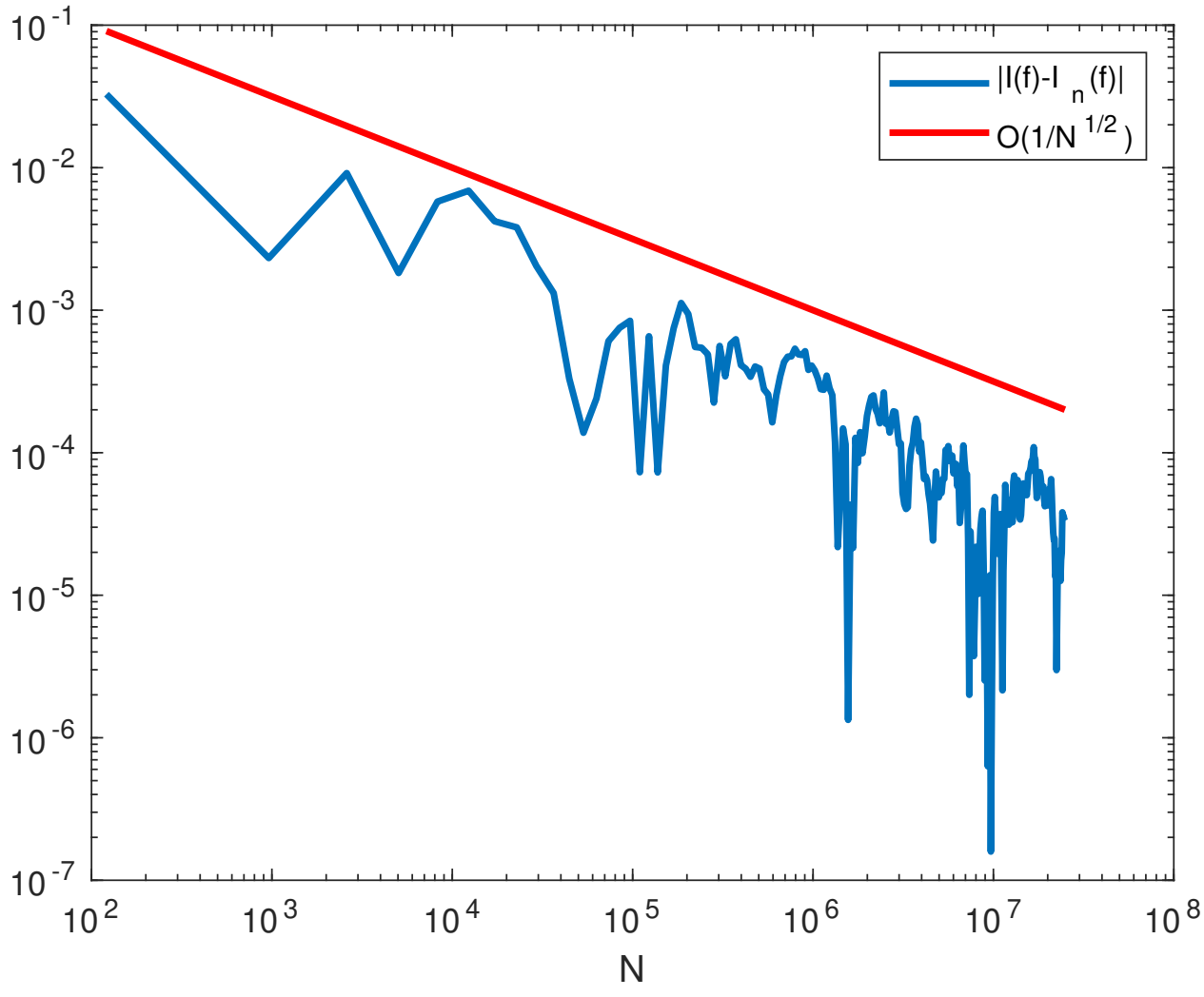
The sample variance is an unbiased estimate of $\sigma^2$ where the sample variance is

$$\begin{aligned} \hat{\sigma}_N^2 \quad &= \quad \frac{1}{N-1} \sum_{k=1}^{N} (f(\mathbf{x}_k) - I_N(f))^2 \\ &= \quad \frac{N}{N-1} \left( \frac{1}{N} \sum_{k=1}^{N} (f(\mathbf{x}_k))^2 - (I_N(f))^2 \right) \end{aligned}$$

# 2D Smooth Example using Monte Carlo



# 2D Non-Smooth Example using Monte Carlo



# How Many Samples Should be Used?

Recall that

$$\epsilon_N := I(f) - I_N(f) \quad \sim \quad \sigma N^{-1/2} Z \, .$$

Thus, if $\sigma$ is finite, as $N \to \infty$ we have

$$CDF(N^{1/2}\sigma^{-1}\epsilon_N) \quad \to \quad CDF(Z)$$

and so

$$\begin{aligned} P(N^{1/2}\sigma^{-1}\epsilon_N < s) \quad &\to \quad P(Z < s) \;=\; \Phi(s) \\ P(|N^{1/2}\sigma^{-1}\epsilon_N| > s) \quad &\to \quad P(|Z| > s) \;=\; 2\Phi(-s) \\ P(|N^{1/2}\sigma^{-1}\epsilon_N| < s) \quad &\to \quad P(|Z| < s) \;=\; 1 - 2\Phi(-s) \, . \end{aligned}$$

Here $\Phi(s)$ is the CDF of a normal distribution with mean 0 and variance 1 so

$$\Phi(s) \quad = \quad \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{s}{\sqrt{2}}\right)\right) \, .$$

# How Many Samples Should be Used?

We can use

$$P(|N^{1/2}\sigma^{-1}\epsilon_N| < s) \quad \rightarrow \quad P(|Z| < s) \;=\; 1 - 2\Phi(-s)\,,$$

to choose $N$ so that $P(|\epsilon_N| < s) \approx c$.

Let $c = 1 - 2\Phi(-s)$ so that $s(c) = \Phi^{-1}((1-c)/2)$. Then

$$P(|N^{1/2}\sigma^{-1}\epsilon_N| < s(c)) \quad \rightarrow \quad c$$

and we use

$$P(|\epsilon_N| < s(c)/(N^{1/2}\sigma^{-1})) \quad \approx \quad c\,.$$

So if we require $P(|\epsilon_N| < TOL) \approx c$ we should choose

$$N \;=\; \left(\frac{\sigma s(c)}{TOL}\right)^2$$

samples. In practice, $\sigma$ is unknown so we can use $\hat{\sigma}_N$ instead where $\hat{\sigma}_N$ is computed using a fairly small value of $N$.

# How Many Samples Should be Used?

We have $c = 1 - 2\Phi(-s)$, $s(c) = \Phi^{-1}((1-c)/2)$ and

$$\Phi(s) \;=\; \frac{1}{2}\left(1 + \mathrm{erf}\left(\frac{s}{\sqrt{2}}\right)\right)$$

so that $c = \mathrm{erf}(s/\sqrt{2})$ and $s = \sqrt{2}\mathrm{erf}^{-1}(c)$ (use `erfinv` in Matlab).

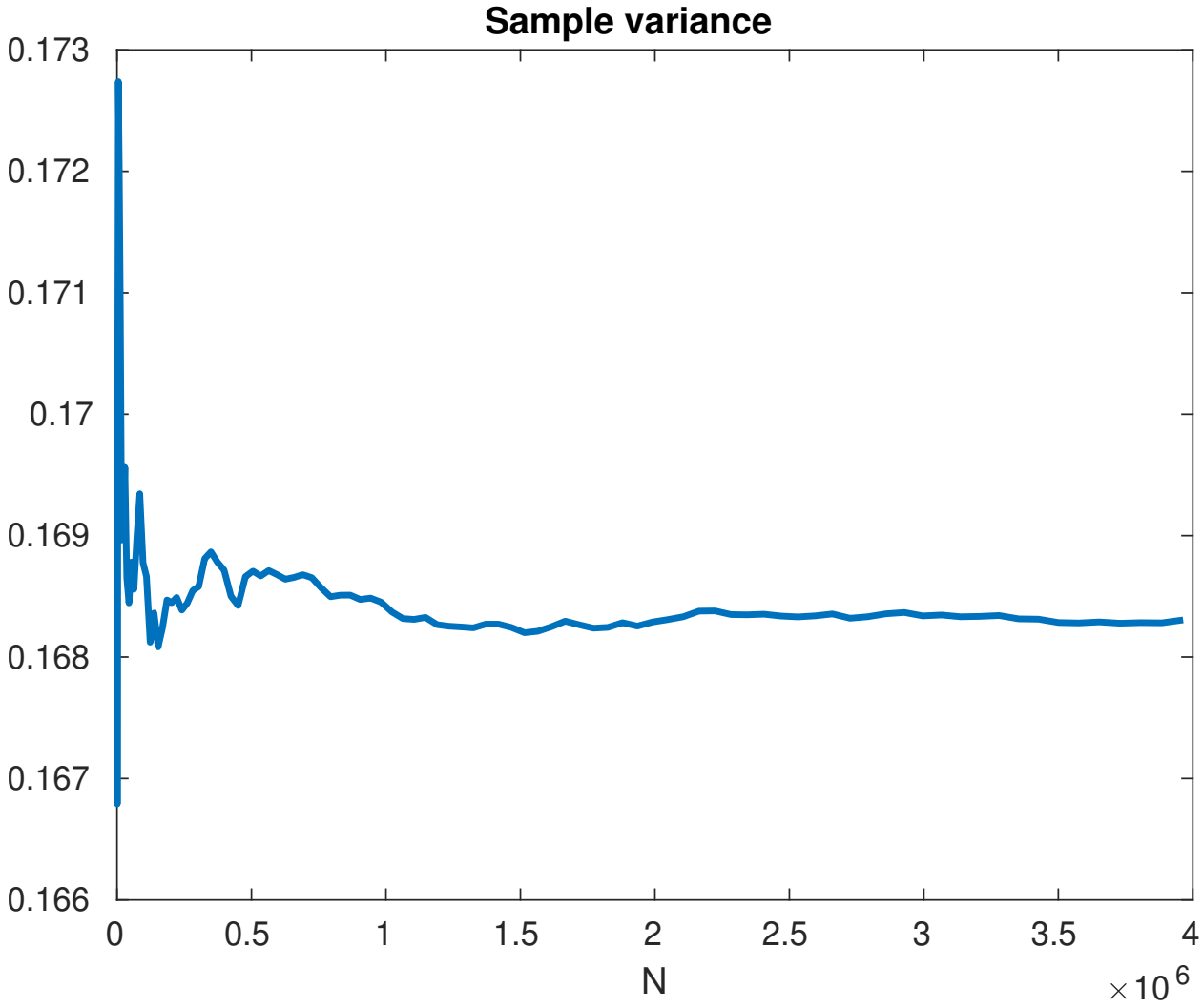
Hence we have

| $c$ | 0.9 | 0.99 | 0.999 | 0.9999 |
|---|---|---|---|---|
| $s$ | 1.6449 | 2.5758 | 3.2905 | 3.8906 |

Thus if we require $P(|\epsilon_N| < 0.01) \approx 0.999$ we choose

$$N \;=\; \left(\frac{3.2905\sigma}{0.01}\right)^2 \;\approx\; 108274\sigma^2 \,.$$

# Sample Variance



# Non-Smooth Example

With $N = 100$ we calculate the sample variance to be $\hat{\sigma}_N^2 = 0.1555$ so if we require $P(|\epsilon_N| < 0.01) \approx 0.999$ we choose

$$N = \left(\frac{3.2905}{0.01}\right)^2 \hat{\sigma}_N^2 = 16832\,.$$

Then we calculate

$$I_N(f) = 0.780121197718631$$

with

$$|I_N(f) - I(f)| = 0.005276965678817 < 0.01\,.$$

(Note that because we use random numbers, this is just one set of results — re-running the code would generate a different sample variance and a different approximation $I_N(f)$.)

# Trapezium vs Monte Carlo

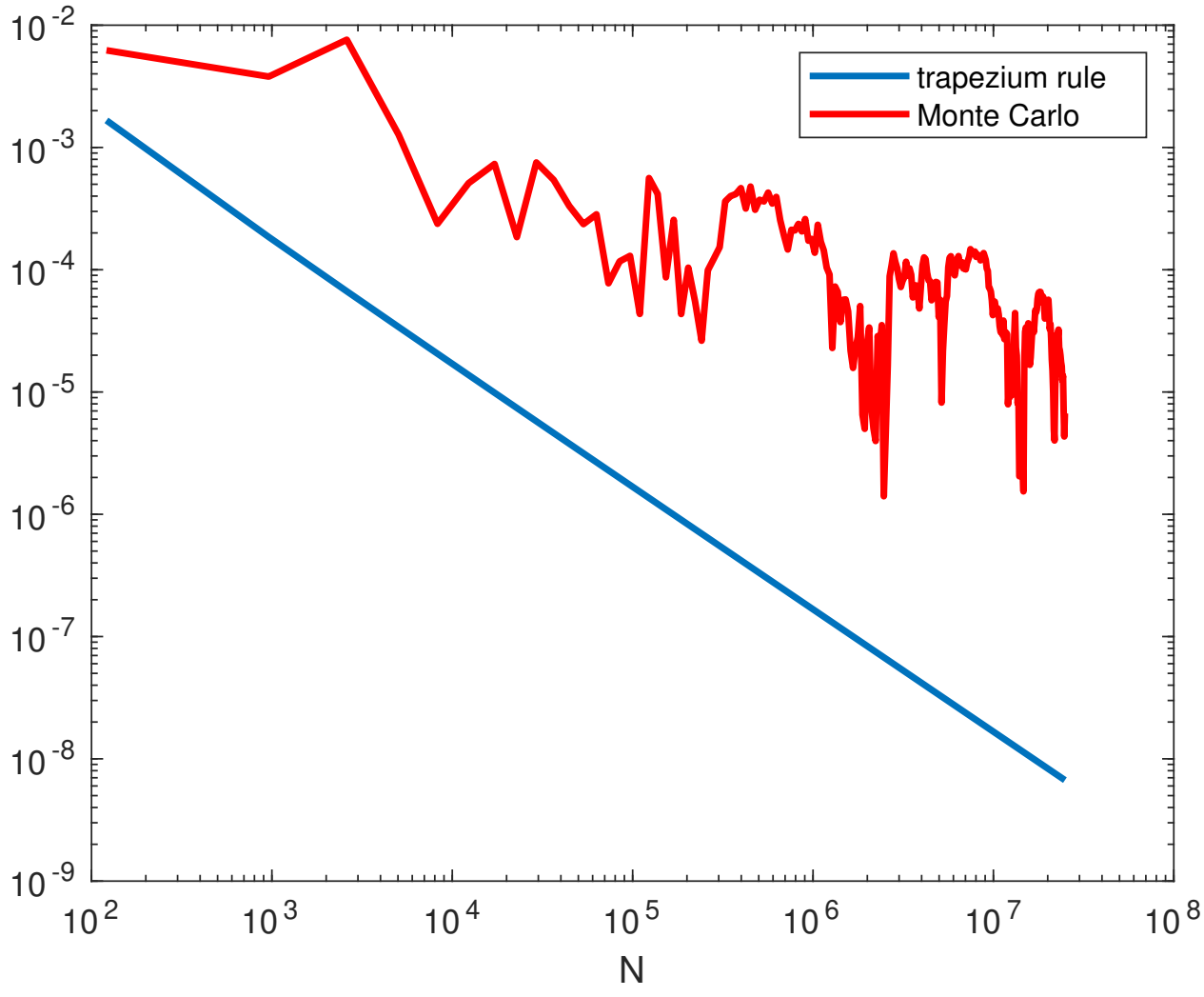
If we use $N$ sample points for the trapezium rule and for Monte Carlo, then the CPU time will be similar. However, we have

$$\begin{aligned} \text{Trapezium rule error} &\sim N^{-2/d} \\ \text{Monte Carlo error} &\sim N^{-1/2} \end{aligned}$$

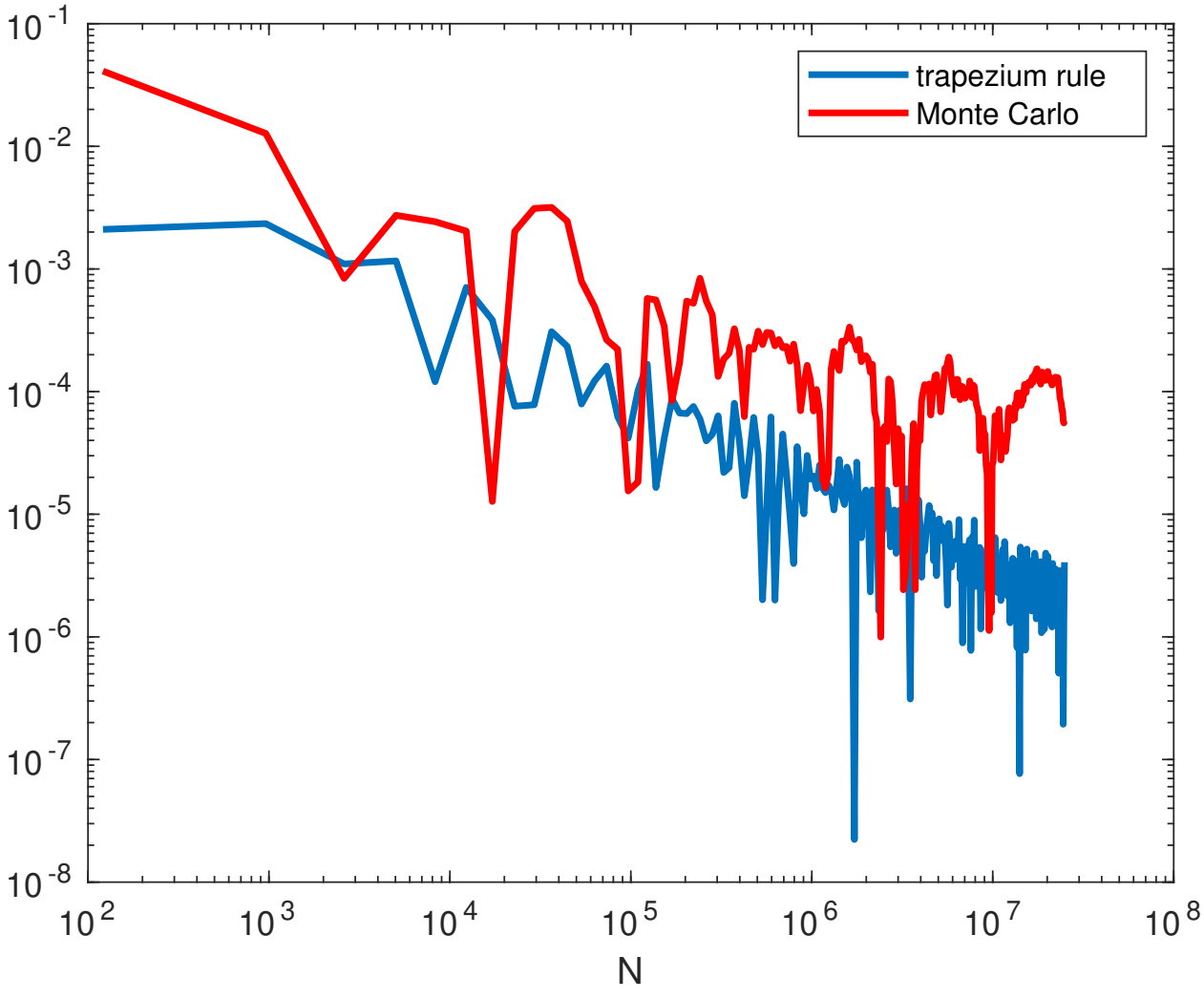
assuming the integrand is sufficiently smooth.

Thus, if $d = 1, 2, 3$ the trapezium rule is better, if $d = 4$ the errors are the same order, and if $d > 4$ Monte Carlo is better. Again this assumes the integrand is sufficiently smooth.

# Smooth Example



# Non-Smooth Example



# Reduction of Number of Sample Points

Recall that if we require $P(|\epsilon_N| < TOL) \approx c$ we should choose

$$N = \left(\frac{\sigma s(c)}{TOL}\right)^2$$

samples. Thus, if we can reduce the variance, we can reduce the number of sample points needed.

# Stratified Sampling in 1D

If we want to approximate

$$I(f) \quad = \quad \int_0^1 f(x)\mathrm{d}x$$

then we could split $[0, 1]$ into $M$ 'strata' of equal width and take $L$ samples in each. Let $x_{i,j}$ be the $i$th sample from the $j$th strata.

Then let

$$\bar{F}_j \quad = \quad \frac{1}{L}\sum_{i=1}^{L} f(x_{i,j})$$

be the average from strata $j$, and the overall average is

$$\bar{F} \quad = \quad \frac{1}{M}\sum_{j=1}^{M} \bar{F}_j \; .$$

# Stratified Sampling in 1D

If we also let

$$\begin{aligned} \mu_j &= \mathbb{E}[f(x)|x \in \text{strata } j] \\ \sigma_j^2 &= \mathbb{V}[f(x)|x \in \text{strata } j] \end{aligned}$$

then

$$\mathbb{E}[\bar{F}] = \frac{1}{M}\sum_{j=1}^{M}\mathbb{E}[\bar{F}_j] = \frac{1}{M}\sum_{j=1}^{M}\mu_j = \mu$$

so it is unbiased.

Also the variance is

$$\mathbb{V}[\bar{F}] = \frac{1}{M^2}\sum_{j=1}^{M}\mathbb{V}[\bar{F}_j] = \frac{1}{M^2}\frac{1}{L}\sum_{j=1}^{M}\sigma_j^2 = \frac{1}{MN}\sum_{j=1}^{M}\sigma_j^2$$

where $N = ML$ is the total number of samples.

# Stratified Sampling in 1D

On the other hand, without stratified sampling $\mathbb{V}[\bar{F}] = \sigma^2/N$ with

$$\begin{aligned}
\sigma^2 &= \mathbb{E}[f^2] - \mu^2 \\
&= \frac{1}{M}\sum_{j=1}^{M} \mathbb{E}[f(x)^2 | x \in \text{strata } j] - \mu^2 \\
&= \frac{1}{M}\sum_{j=1}^{M} (\mu_j^2 + \sigma_j^2) - \mu^2 \\
&= \frac{1}{M}\sum_{j=1}^{M} ((\mu_j - \mu)^2 + \sigma_j^2) \\
&\geq \frac{1}{M}\sum_{j=1}^{M} \sigma_j^2 \, .
\end{aligned}$$

# Stratified Sampling in 1D

Thus, with stratified sampling we have

$$\mathbb{V}[\bar{F}] = \frac{1}{MN}\sum_{j=1}^{M}\sigma_j^2$$

and without stratified sampling we have

$$\mathbb{V}[\bar{F}] \geq \frac{1}{MN}\sum_{j=1}^{M}\sigma_j^2$$

and we see that stratified sampling reduces the variance.

# Stratified Sampling in 1D

An alternative is to use $L_j$ samples in stratum $j$. Then it can be shown that the overall variance is

$$\frac{1}{M^2} \sum_{j=1}^{M} \frac{1}{L_j} \sigma_j^2 \ .$$

If we want the total number of samples $N = \sum_{j=1}^{M} L_j$ to be fixed, then the variance is minimised if $L_j$ is proportional to $\sigma_j$.

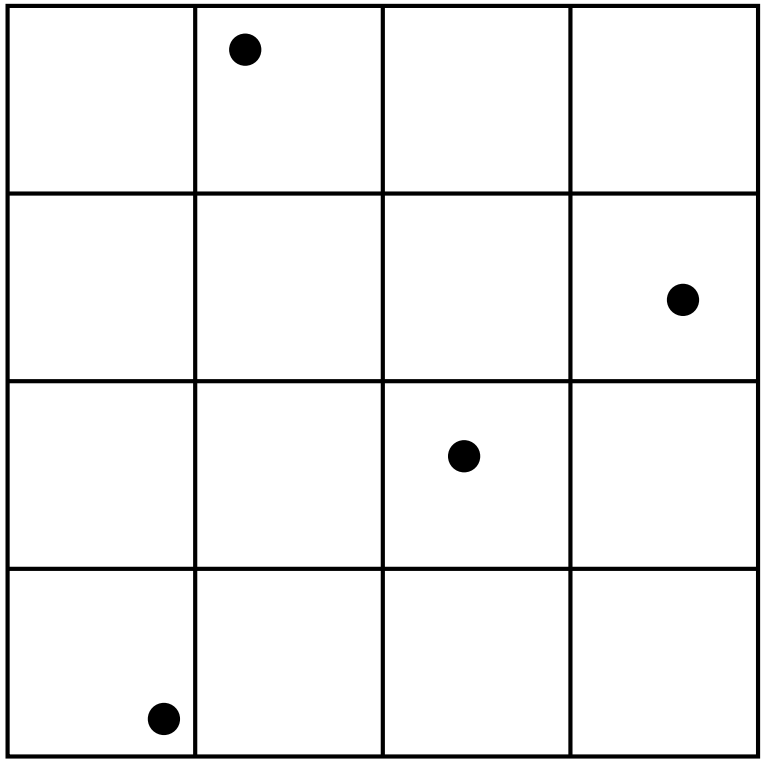
# Stratified Sampling in Higher Dimensions

A generalisation to $d$ dimensions could split $[0,1]^d$ into $M^d$ sub-cubes by splitting each dimension into $M$ strata. Then $L$ points could be used in each subcube. The problem is that this requires $LM^d$ function evaluations which grows very quickly with $d$ unless $M$ is small.

An alternative is to use Latin Hypercube sampling.

# Latin Hypercube sampling

Here the idea is to generate $M$ points, dimension by dimension, using 1D stratified sampling with 1 value per stratum assigning them randomly to the $M$ points to give precisely one point in each stratum.



# Latin Hypercube sampling

This gives one set of $M$ points, with average

$$\bar{f} \quad = \quad \frac{1}{M}\sum_{k=1}^{M} f(\mathbf{x}_k)\,.$$

Again this is unbiased, i.e. we have $\mathbb{E}[\bar{f}] = \mathbb{E}[f]$.

If we now take $L$ independently generated sets of points to get an average, $\bar{F}_\ell$, every time, we can compute an average of these

$$\frac{1}{L}\sum_{\ell=1}^{L} \bar{f}_\ell$$

which is again an unbiased estimate for $\mathbb{E}[f]$.

# Other Methods

Other methods for variance reduction include

- antithetic variables
- control variates
- importance sampling
- quasi-Monte Carlo methods
- ...