Bayesian estimation of convergence rates in numerical approximation

Bruno Degli Esposti[1] and Davide Fabbrico[2]

[1]Department of IE and Mathematics
University of Siena, Italy

[2]Department of Statistics
University of Florence, Italy

UNIVERSITÀ
DI SIENA
1240

Multivariate Splines for Inferential Data Science
14 May 2026, Granada, Spain

Introduction

Introduction

Main topic

We investigate the use of statistical tools to rigorously analyze data produced by numerical experiments, in particular to estimate error convergence rates. Typically, errors decay as $\mathcal{O}(h^k)$ with respect to the reduction of a discretization parameter $h \to 0^+$.

Real-world applications

The new approach allows us to

- Estimate the convergence rate k in both stochastic and "noisy" deterministic numerical methods
- Quantify uncertainty in the estimation of convergence rates
- Optimize the number of numerical experiments needed

Introduction

Error plots provide empirical evidence that a numerical method is convergent and has been implemented correctly. More than 30% of last week's arXiv papers in numerics feature convergence plots.

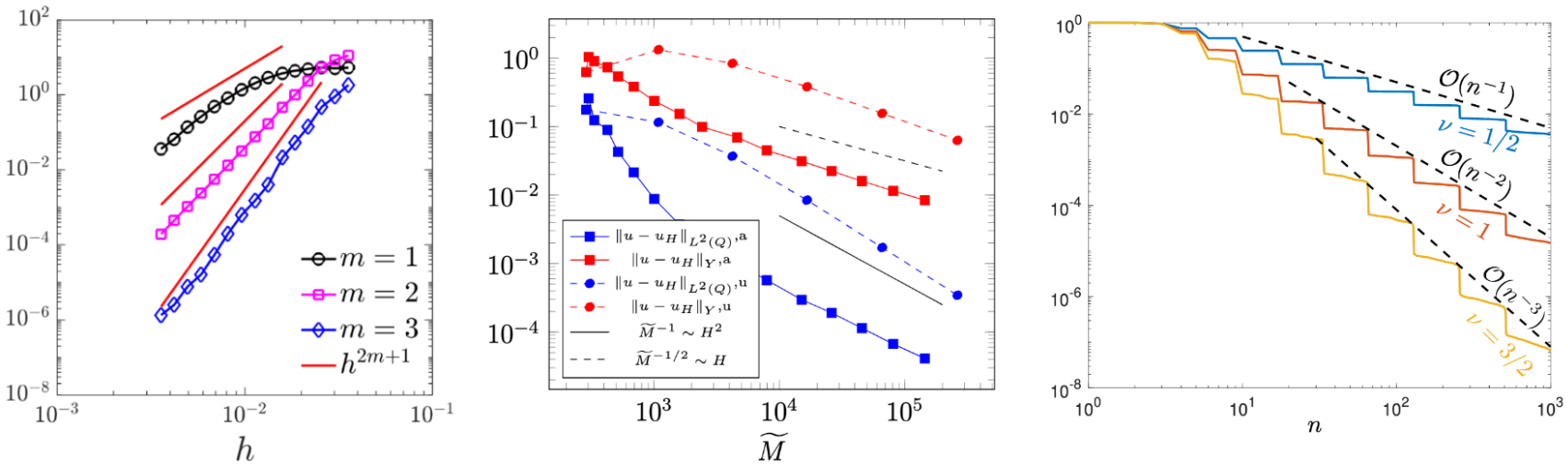


Figure: Convergence plots in the wild (authors: Law; Köthe, Steinbach; Jeong, Townsend). Notice the log-log scale and the reference slopes.

Introduction

There are two paths to clarity and understanding in numerical analysis: theories leading to proofs, and numerical experiments producing data.

As mathematicians, we have high standards of rigour for proofs, but often not as high when analyzing data coming out of our numerical experiments. It's extremely rare to see anything more than a simple reference slope. Why?

Introduction

There are two paths to clarity and understanding in numerical analysis: theories leading to proofs, and numerical experiments producing data.

As mathematicians, we have high standards of rigour for proofs, but often not as high when analyzing data coming out of our numerical experiments. It's extremely rare to see anything more than a simple reference slope. Why? Possible explanations:

- Numerical experiments are considered inferior to proofs

Introduction

There are two paths to clarity and understanding in numerical analysis: theories leading to proofs, and numerical experiments producing data.

As mathematicians, we have high standards of rigour for proofs, but often not as high when analyzing data coming out of our numerical experiments. It's extremely rare to see anything more than a simple reference slope. Why? Possible explanations:

- Numerical experiments are considered inferior to proofs
- It's always been done this way

Introduction

There are two paths to clarity and understanding in numerical analysis: theories leading to proofs, and numerical experiments producing data.

As mathematicians, we have high standards of rigour for proofs, but often not as high when analyzing data coming out of our numerical experiments. It's extremely rare to see anything more than a simple reference slope. Why? Possible explanations:

- Numerical experiments are considered inferior to proofs
- It's always been done this way
- Mathematicians often don't know statistics

Introduction

There are two paths to clarity and understanding in numerical analysis: theories leading to proofs, and numerical experiments producing data.

As mathematicians, we have high standards of rigour for proofs, but often not as high when analyzing data coming out of our numerical experiments. It's extremely rare to see anything more than a simple reference slope. Why? Possible explanations:

- Numerical experiments are considered inferior to proofs
- It's always been done this way
- Mathematicians often don't know statistics
- We don't need more to check convergence anyway

Introduction

Simplest approach: linear least-squares regression. It takes just one line of code in MATLAB:

```
fit(log10(h_array),log10(err_array),'poly1')
```

The slope of the line estimates the convergence rate k. The intercept estimates the (base 10 logarithm) of the constant C multiplying h^k.

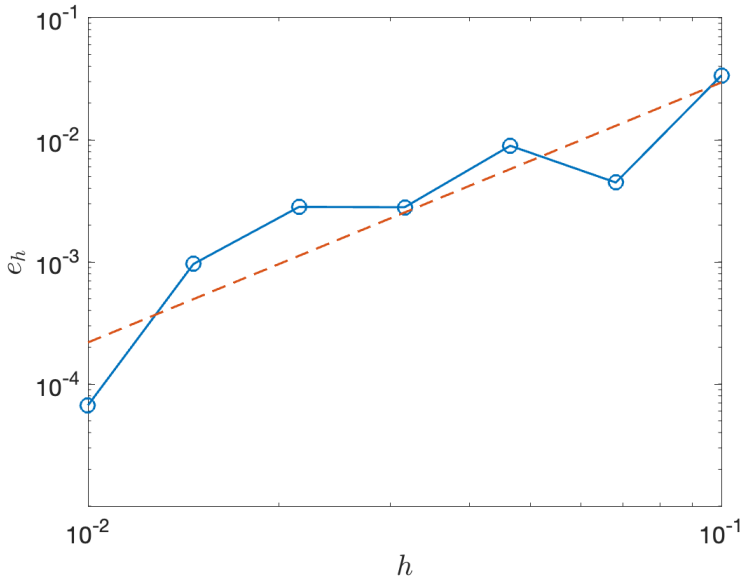
Examples where automatic estimation of k and C is needed:

- Software testing
- Tuning parameters with grid search
- Too many numerical experiments to plot
- Number of experiments required to achieve uncertainty threshold

Introduction

Sources of noise in convergence plots:

- Use of stochastic (sub)algorithms
- h may still be in the pre-asymptotic regime
- Roundoff errors, possibly magnified by poor conditioning
- Arbitrary choice of mesh/point cloud/initialization/etc

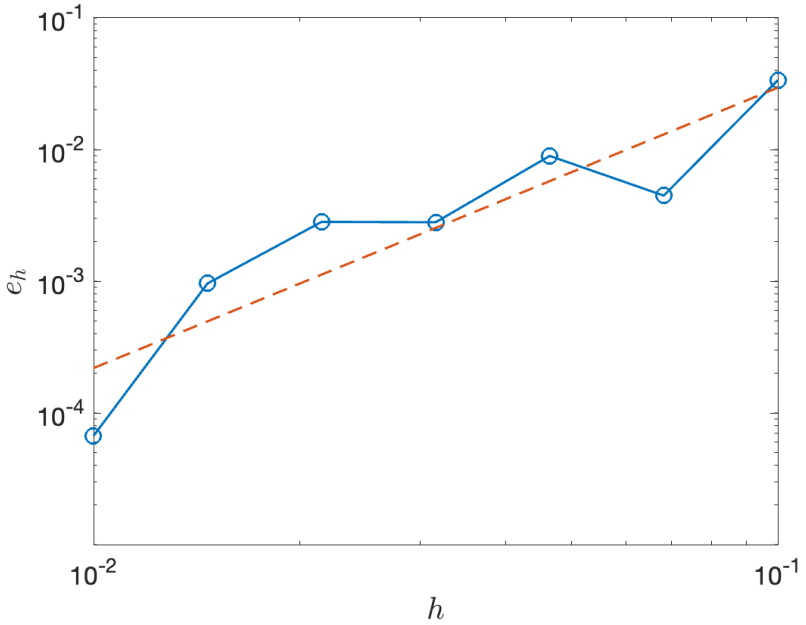


Example of noisy error plot.
Linear fit with slope 2.13.
The 95% confidence interval is (0.99, 3.27).

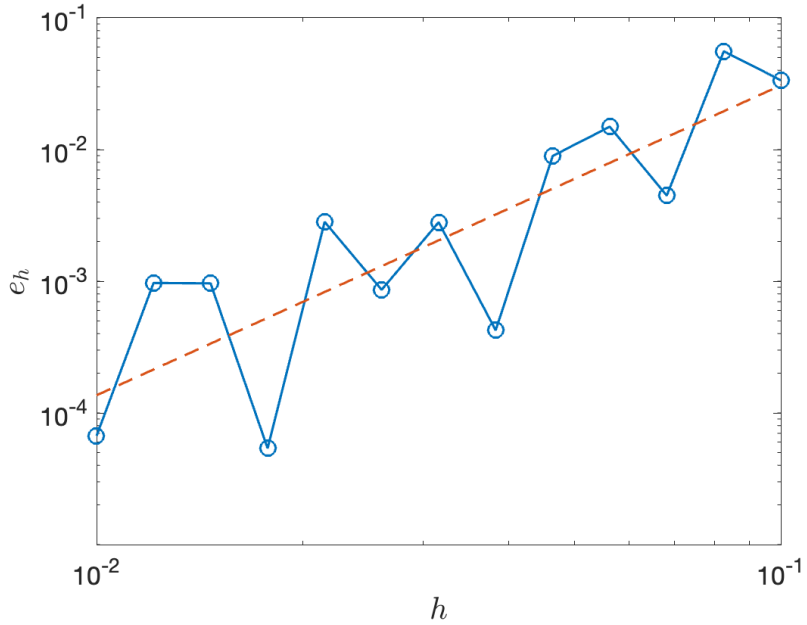
Can we claim h^2 convergence?

Introduction

Let's enlarge the dataset. Do things get better?



(a) 7 data points in $[10^{-2}, 10^{-1}]$.
Linear fit with slope 2.13.
95% CI (0.99, 3.27).



(b) 13 data points in $[10^{-2}, 10^{-1}]$.
Linear fit with slope 2.35.
95% CI (1.27, 3.44).

Modeling errors as random variables

Modeling errors as random variables

Fundamental intuition

Numerical errors should be modeled as random variables X_h not only in the case of stochastic algorithms (e.g. Monte Carlo integration), but also with deterministic methods subject to arbitrary choices which cannot be carried out in a unique or canonical way. Examples:

- Placement of scattered nodes in meshless methods
- Generation of a mesh with target element size h in FEM
- Choice of initialization in iterative algorithms

In principle, a numerical method can have two distinct (algebraic) rates of convergence k_1 and k_2: decay rate of the mean errors, and decay rate of the standard deviations of the errors:

$$\mathbb{E}(X_h) \approx C_\mu h^{k_1} \quad \sigma(X_h) \approx C_\sigma h^{k_2} \quad \text{for } h \to 0^+.$$

Error model for signed errors

Statistical inference of convergence order requires a model. For signed errors we use the model

$$X_h \sim \mathcal{N}\left(C_\mu h^k, (C_\sigma h^k)^2\right).$$

Modeling assumptions

1. Normally distributed errors. Intuition: the global error is often the combination of many small weakly-dependent local errors.
2. Independent errors. Does not apply to adaptive approximations or hierarchical refinement, because meshes are dependent.
3. Mean and std deviation of the errors decay at the same rate k. Note that the h-indexed family of random variables is heteroscedastic.

Error model for unsigned errors

For unsigned errors we use instead

$$X_h \sim \mathcal{N}^+\left(C_\mu h^k, (C_\sigma h^k)^2\right).$$

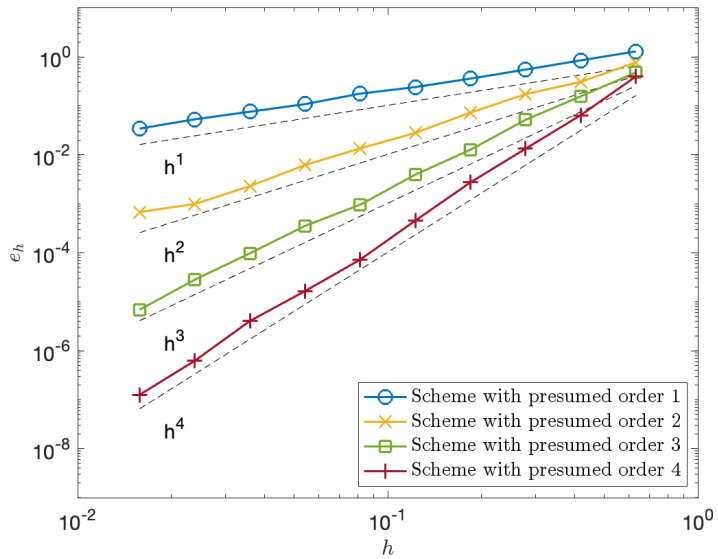
where $\mathcal{N}^+$ is the folded normal distribution (i.e. X_h is the absolute value of a normally distributed random variable).

Signed errors are appropriate when approximating integrals or any single real pointwise value.

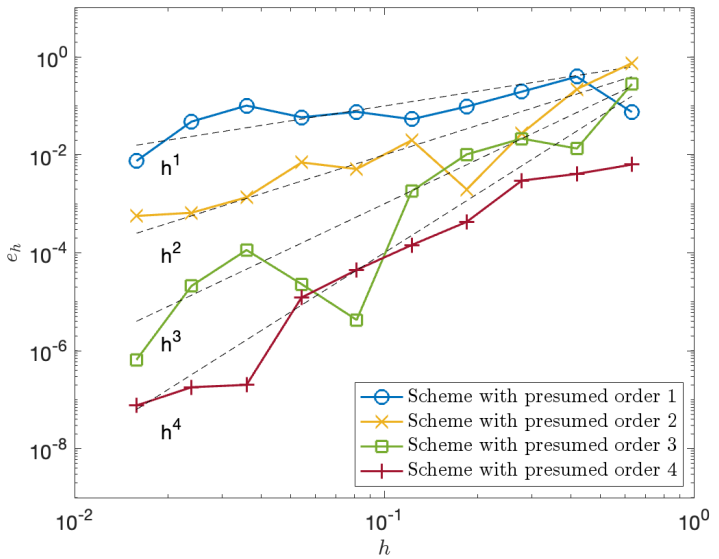
Unsigned errors are appropriate for distances or norms.

Convergence regimes

The models reproduce both smooth and noisy convergence plots, depending on the ratio C_μ / C_σ:



(a) $C_\mu = 8,\ C_\sigma = 1,\ k = 1, \ldots, 4.$



(b) $C_\mu = 8,\ C_\sigma = 8,\ k = 1, \ldots, 4.$

Figure: Sampling the unsigned error model with different ratios.

Convergence regimes

Recall that

$$X_h \sim \mathcal{N}^{(+)}\left(C_\mu h^k, (C_\sigma h^k)^2\right).$$

Rule of thumb

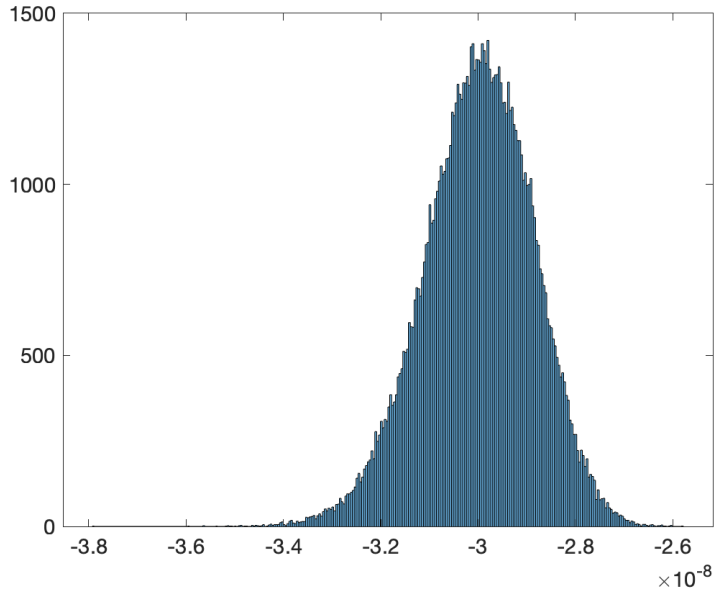
$\mathbb{E}(X_h)$ dominates $\sigma(X_h)$ and error plots will look smooth only if $C_\mu > 4C_\sigma$, in which case we say that we are in a mean-dominated convergence regime. Otherwise, we call it variance-dominated.

The ratio C_μ/C_σ depends on:

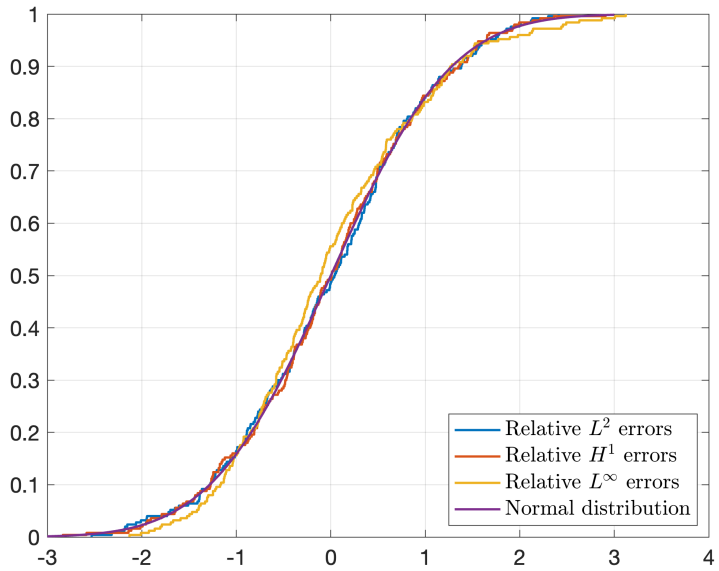
- Numerical method
- Choice of discretization
- Input data

Normality assumption

The normality assumption can be checked for mesh-based methods by randomly perturbating a given mesh, or by running the same algorithm on several meshes of equivalent quality.



(a) Trapezoidal quadrature in 1D



(b) FEM with P1 elements on torus

Figure: Error histogram and empirical CDF of two common numerical schemes on randomized meshes, validating the normality assumption.

Bayesian inference

Bayesian inference

Let θ be the vector of parameters in our model that we want to estimate from the observed data. Then, by Bayes' formula,

$$p(\theta \mid \text{data}) = \frac{p(\text{data} \mid \theta) \cdot p(\theta)}{p(\text{data})}$$

Posterior — Likelihood — Prior — Normalization

Bayesian inference delivers a probability distribution for θ, called the posterior distribution, from which we can compute mean, standard deviation, find its peak(s), etc.

Bayesian inference

We reformulate our model for e.g. signed errors as

$$X_h \sim \mathcal{N}\left((-1)^s 10^{\beta_\mu} h^k, \left(10^{\beta_\sigma} h^k\right)^2\right)$$

and define $\theta = (s, \beta_\mu, \beta_\sigma, k)$. We choose the uninformative prior

$$s \sim \text{Bernoulli}(0.5), \qquad \beta_\mu, \beta_\sigma, k \sim \text{Uniform}(-8, 8)$$

to reflect our belief that all orders of magnitude for C_μ, C_σ, k are equally likely before looking at the data.

Having fixed the prior distribution, which gives us $p(\theta)$, and the model, which gives us $p(\text{data}|\theta)$, we now know the posterior distribution $p(\theta|\text{data})$ up to the normalization factor $p(\text{data})$. How do we sample from the posterior distribution?

Bayesian inference via MCMC

Algorithm 1: Metropolis–Hastings

Initialize $\theta_0 = (s, \beta_\mu, \beta_\sigma, k)$ by sampling the prior distribution.

for $i = 1, \ldots, M$ **do**

Define θ' as a random perturbation of θ_{i-1}.

Calculate the acceptance ratio using prior and model:

$$\alpha = \frac{p(\text{data}|\theta')p(\theta')}{p(\text{data}|\theta)p(\theta)}.$$

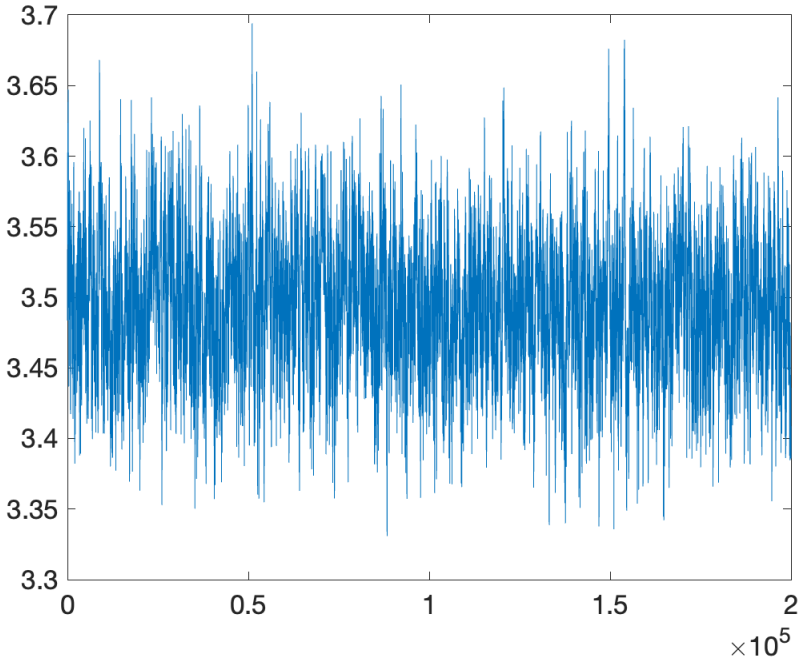
Generate a uniform random number $u \in [0, 1]$.

Set $\theta_i = \theta'$ if $u \leq \alpha$, otherwise set $\theta_i = \theta_{i-1}$.

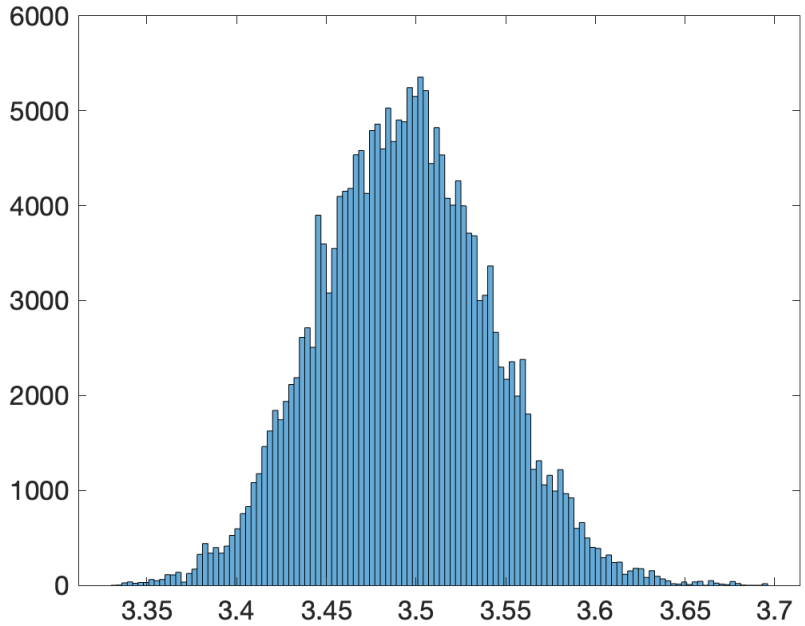
Output the sequence $\{\theta_i\}_{i=1,\ldots,N}$ after discarding the first $N/3$ values.

We use $M = 300000$, and the variance of random (normal) perturbations is tuned to accept new steps $\approx 23\%$ of the time.

Bayesian inference via MCMC



(a) Posterior samples for k.



(b) Posterior distribution of k.

Figure: Example on dataset with order of convergence 3.5

Bayesian inference via MCMC

Validation of the Bayesian approach

We fix $\bar{\theta}$, generate data using our normal model with parameters $\bar{\theta}$, and run the Metropolis algorithm to recover a probability distribution, from which we can compute mean and standard deviation.

N	$\bar{\theta}$	C_μ / C_σ	β_μ	β_σ	k
7	$(0, 1, 0, 3.5)$	10	0.89 ± 0.08	-0.01 ± 0.16	3.46 ± 0.04
13	$(0, 1, 0, 3.5)$	10	0.98 ± 0.03	-0.16 ± 0.10	3.49 ± 0.01
7	$(0, 0, 0, 3.5)$	1	-2.29 ± 2.62	0.19 ± 0.36	3.58 ± 0.17
13	$(0, 0, 0, 3.5)$	1	0.16 ± 1.34	0.24 ± 0.29	3.56 ± 0.13
101	$(0, 0, 0, 3.5)$	1	-0.04 ± 0.11	0.03 ± 0.10	3.49 ± 0.05

Table: N data points from $h =$ 1e-3 to 1e-1.
Means and standard deviations of posterior samples.

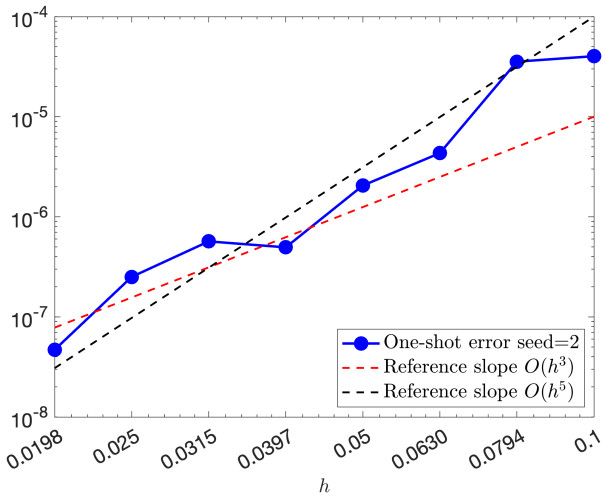
Case study: meshless moment-free quadrature

Case study: meshless moment-free quadrature

One of my research topics is the construction of quadrature formulas on scattered nodes (point clouds) over a bounded domain $\Omega \subset \mathbb{R}^d$.

We want to establish convergence of the errors as $h \to 0^+$, where h is the size of the largest gap in the point distribution (fill distance).

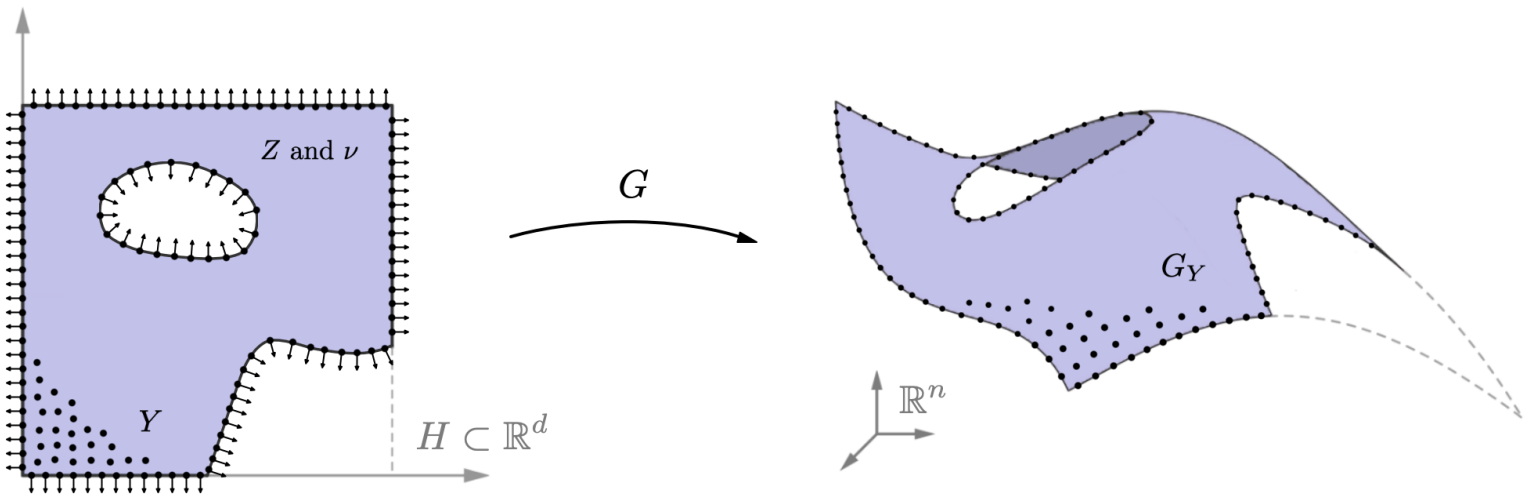
The convergence of meshless moment-free quadrature introduced in [1] often falls in the variance-dominated regime. Example: quadrature error on torus. What is the source of this noise? What is k?



[1] Oleg Davydov and Bruno Degli Esposti (2025). *Meshless moment-free quadrature formulas arising from numerical differentiation*, CMAME.

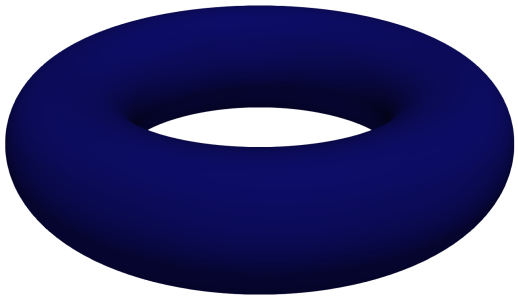
Case study: meshless moment-free quadrature

Scattered nodes Z over the boundary of Ω and Y over the interior of Ω are generated by an advancing front method which, at every step, calls a random number generator (RNG) to place new candidates for expansion around existing nodes at distance h.

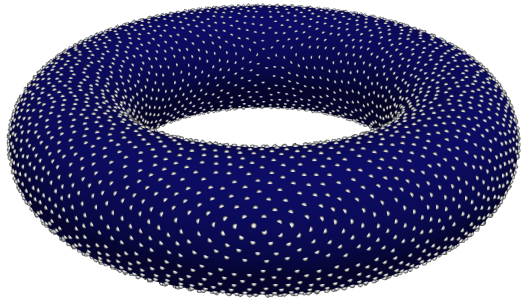


Quadrature errors are a function of RNG seed, i.e. a random variable. Errors on the plot are independent samples. Noise comes from variance!

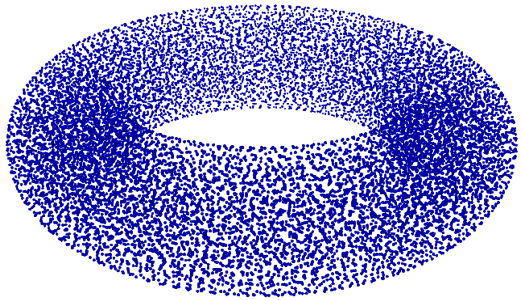
Case study: meshless moment-free quadrature



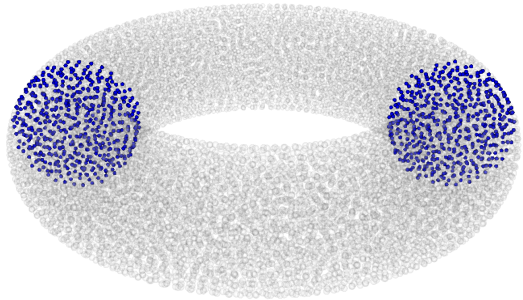
(a) Torus $r = 0.32, R = 1$.



(b) Boundary nodes.



(c) Interior nodes.

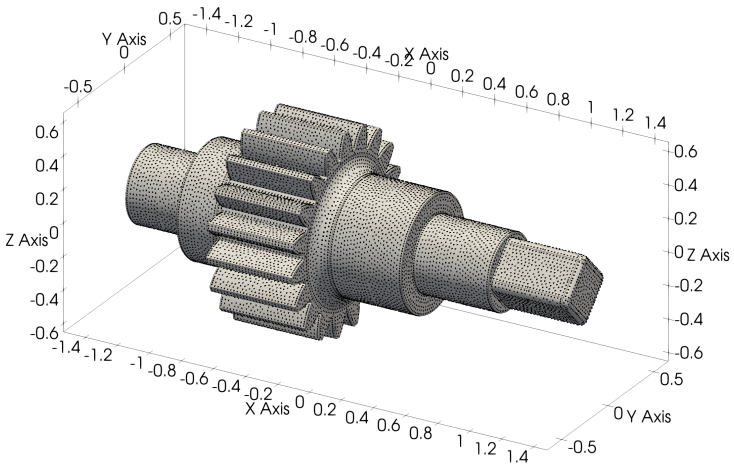


(d) Section of the nodes.

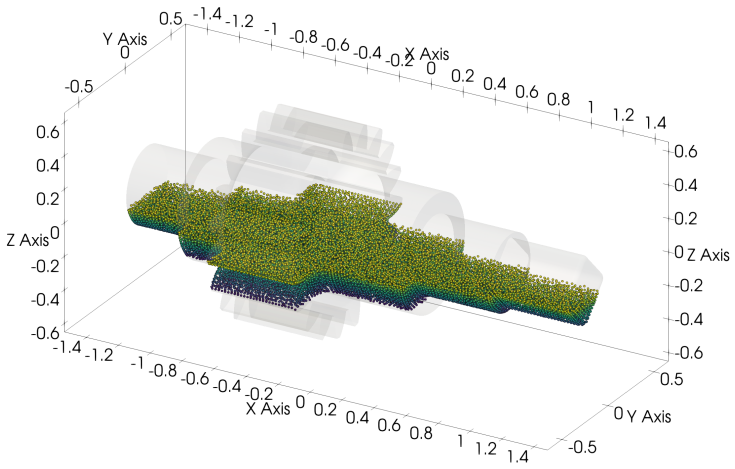
Figure: Nodes produced by advancing front method on a torus, $h = 0.05$.

Case study: meshless moment-free quadrature

Advancing front method available as C++/MATLAB library at
`https://github.com/BrunoDegliEsposti/NodeGenLib`
The code natively supports STEP files, the standard in CAD based on NURBS surface patches.



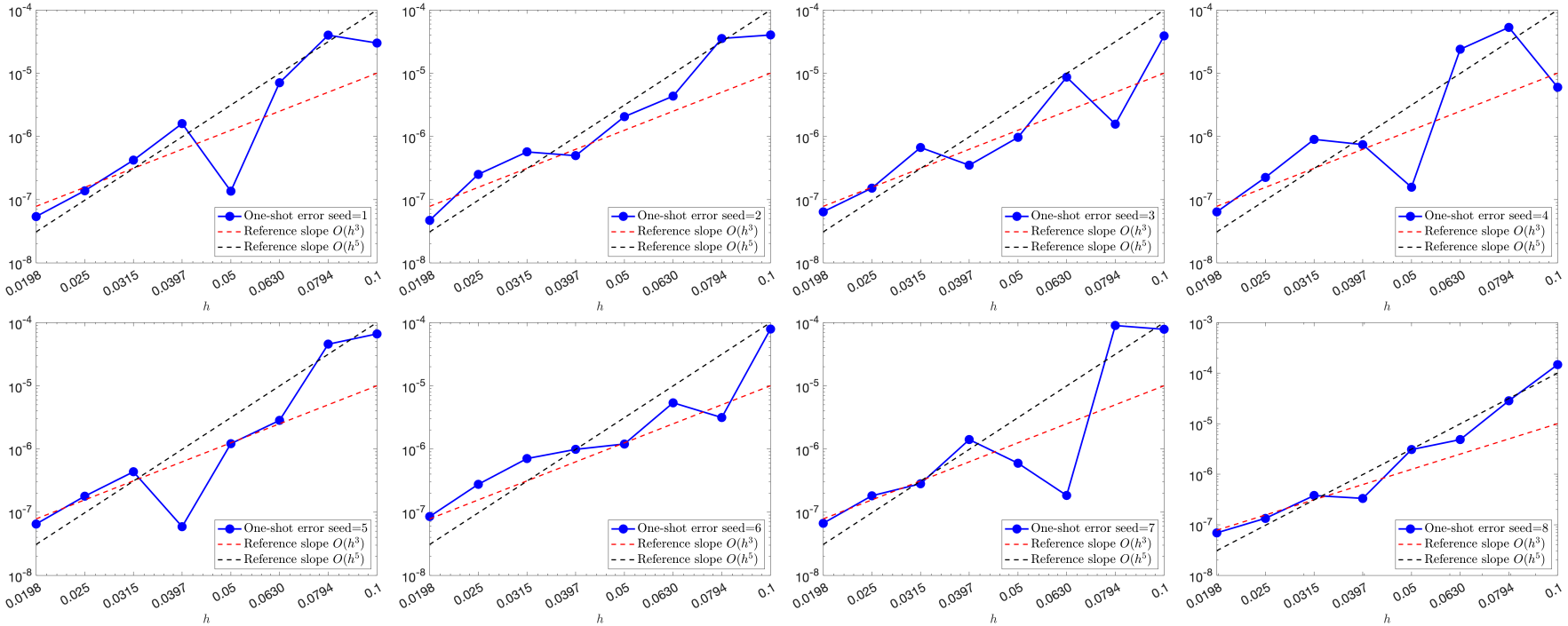
(a) Nodes on surface of gear shaft



(b) Nodes inside gear shaft

Case study: meshless moment-free quadrature

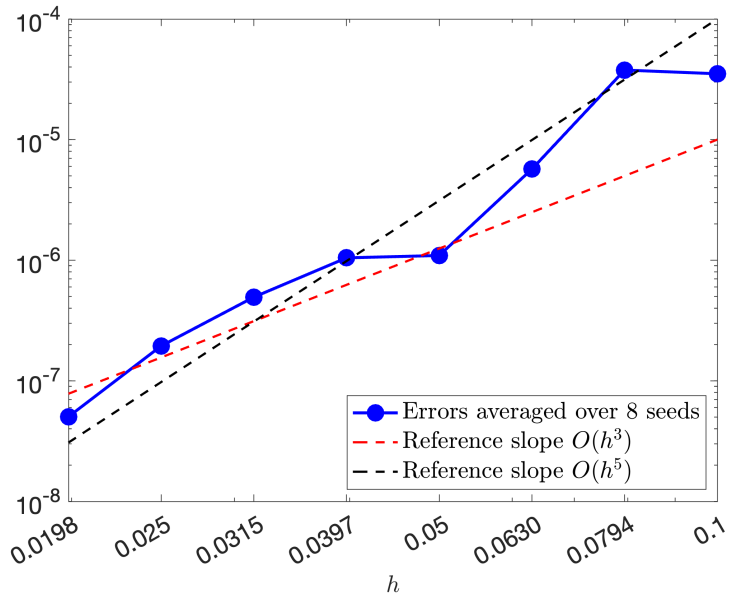
Quadrature errors on the torus for 8 different seeds:



Too noisy to publish! And cherry picking is scientific fraud. So standard fix in meshless community: average errors over seeds to reduce variance.

Case study: meshless moment-free quadrature

Plot of averaged errors. Least squares line fit estimates $k = 4.4$ with 95% confidence bounds $(3.433, 4.431)$. Order 4? Or more?



Pros: simple fix, enough to publish our paper.

Cons: runtime has grown by 8x, and the exact order is still unclear. How does the residual noise affect our estimate of k?

Case study: meshless moment-free quadrature

Consider again the quadrature errors on the torus for 8 different seeds. Each seed is used to run 8 simulations with h in the range 0.1 to 0.02.

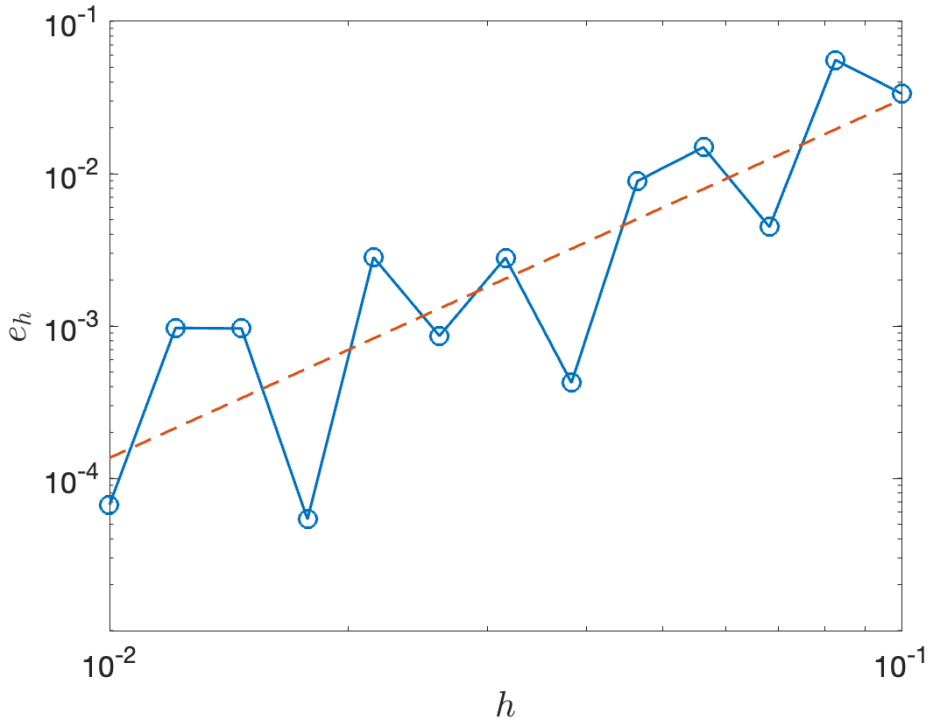
Theoretical analysis of the meshless quadrature scheme predicts $\mathcal{O}(h^4)$ convergence. However, the method might be superconvergent. Naive linear regression of averaged errors estimates $k = 4.4$. The evidence is not strong enough to claim that our method is superconvergent.

To support this claim, we need our new statistical approach:

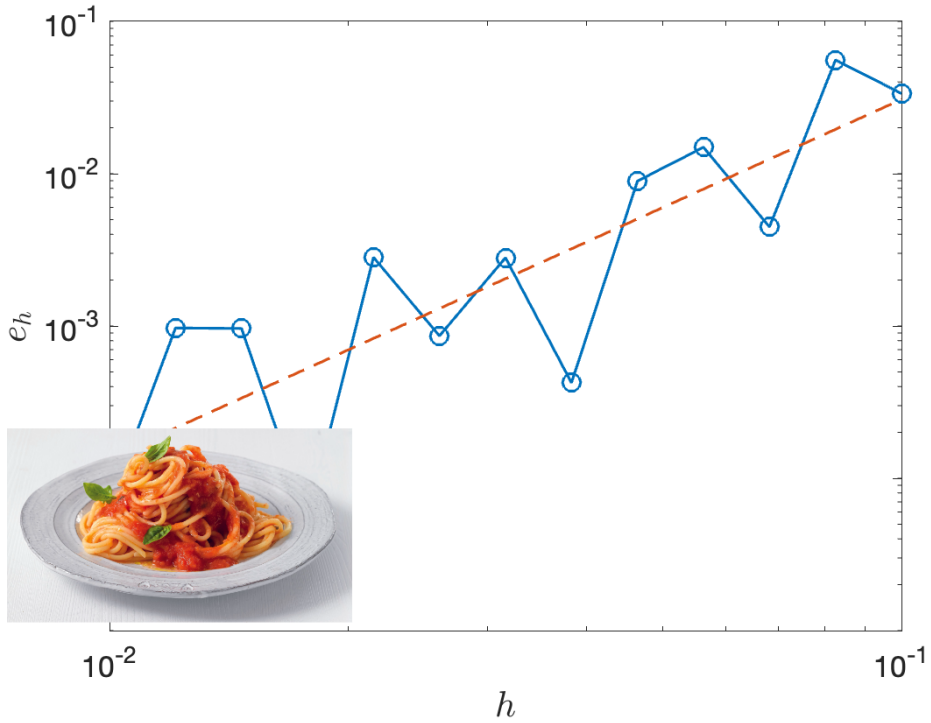
Seeds	N	k
1	8	4.69 ± 0.24
2	16	4.65 ± 0.21
4	32	4.73 ± 0.18
8	64	4.69 ± 0.15

Table: Means and standard deviations of posterior samples.

Case study: meshless moment-free quadrature



Case study: meshless moment-free quadrature



Thank you for your attention!