Bayesian estimation of convergence rates in numerical methods

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

[1]Department of Mathematics
University of Florence, Italy

[2]Department of Statistics
University of Florence, Italy

Fifth Edition of the Young Applied Mathematicians Conference
23 September 2025, Padova, Italy

Introduction

Introduction

Error plots provide empirical evidence that a numerical method is convergent. For this reason, they are everywhere in numerical analysis: more than 30% of the papers uploaded to arXiv last week have one.

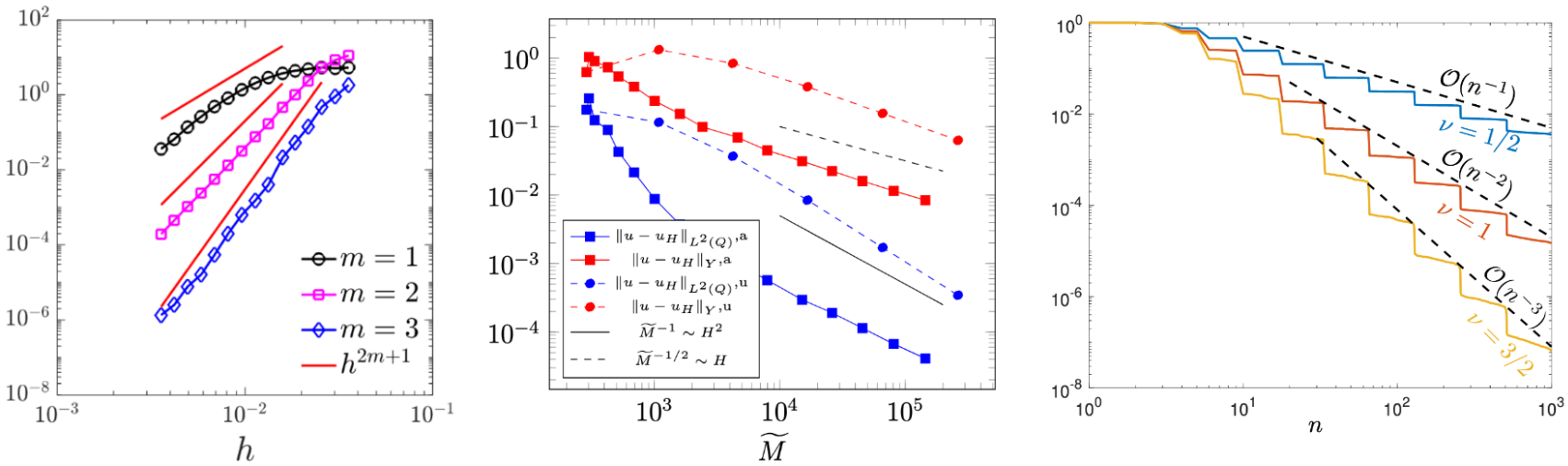


Figure: Convergence plots in the wild (authors: Law; Köthe, Steinbach; Jeong, Townsend). Notice the log-log scale and the reference slopes.

Introduction

Paradox #1

The complexity of numerical methods has increased tremendously in the last 80 years, but their performance is still naively evaluated by visual inspection. Linear regression has been known for 200+ years, and today can be done with one line of code.

Paradox #2

Mathematicians are unlikely to know and to use statistical methods, because mathematics is not an experimental science. And yet, the accuracy and stability of some algorithms can only be tested by performing numerical experiments.

Trying to avoid statistics can unfortunately lead to bad statistics!

Introduction

Research 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
- Optimize the number of numerical experiments needed
- Quantify uncertainty in the estimation of convergence rates

Motivation for statistical analysis

Meshless moment-free quadrature

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

$Y_h \subset \overline{\Omega}$ interior quadrature nodes with weights $w_h \in \mathbb{R}^{|Y_h|}$

$Z_h \subset \partial\Omega$ boundary quadrature nodes with weights $v_h \in \mathbb{R}^{|Z_h|}$

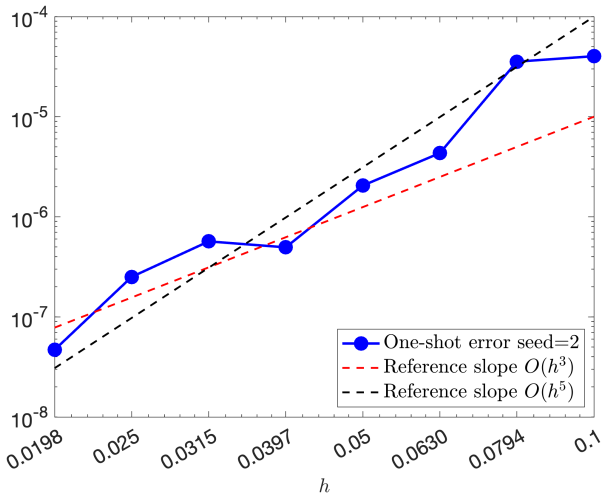
$$\int_\Omega f(x)\,dx \approx \sum_{i=1}^{|Y_h|} w_{h,i} f(y_{h,i}) \qquad \int_{\partial\Omega} g(x)\,d\boldsymbol{\sigma}(x) \approx \sum_{i=1}^{|Z_h|} v_{h,i} g(z_{h,i})$$

In the context of scattered nodes, h is typically defined as the fill distance, i.e. the size of the largest hole in the point distribution.

We want to establish convergence of the errors as $h \to 0^+$.

Meshless moment-free quadrature

In [1] we introduced a way to simultaneously compute quadrature weights w and v by discretizing the divergence theorem. As usual, to publish our discovery, we had to run numerical experiments and produce convergence plots. However, errors can decay erratically, and the rate of convergence is not always clear. Example: quadrature error on torus.

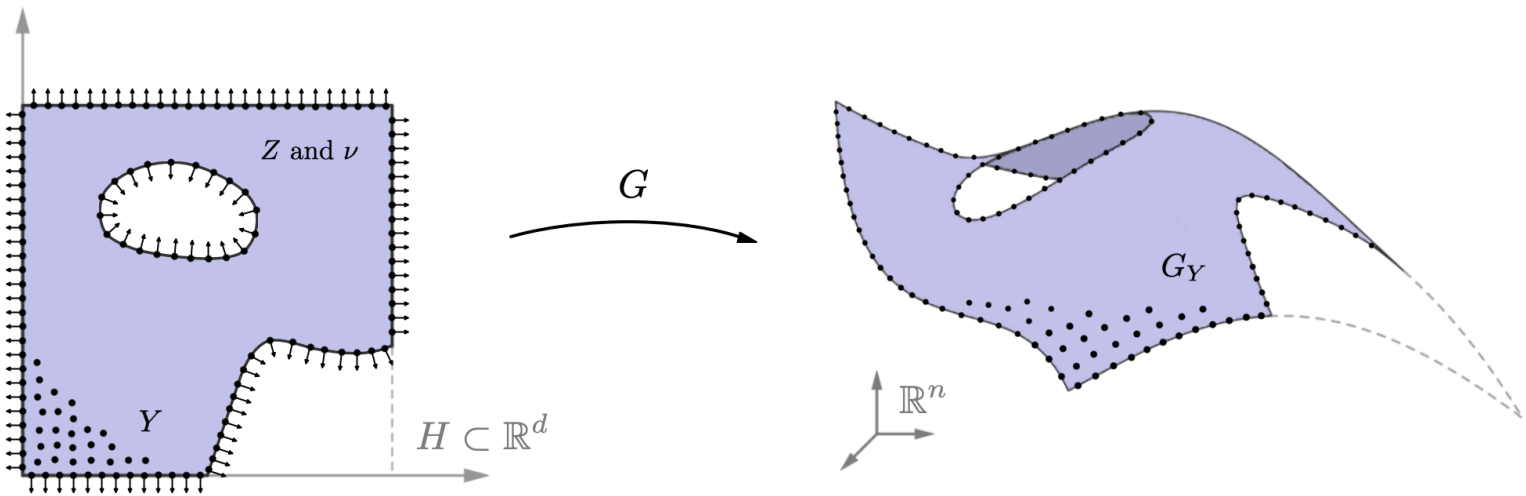


[1] Oleg Davydov and Bruno Degli Esposti (2025). *Meshless moment-free quadrature formulas arising from numerical differentiation*, CMAME.

What is the source of this noise?

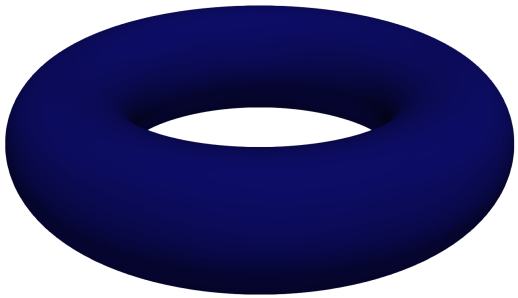
Node generation

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.

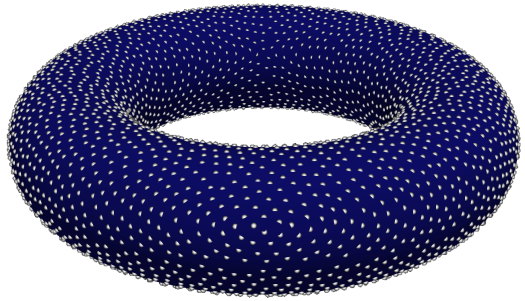


Quadrature errors are a function of RNG seed, i.e. a random variable. Errors on the plot are independent samples. Noise comes from variance!

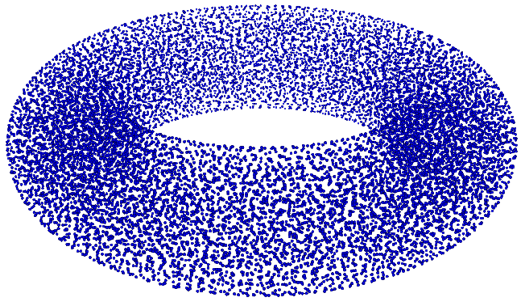
Node generation



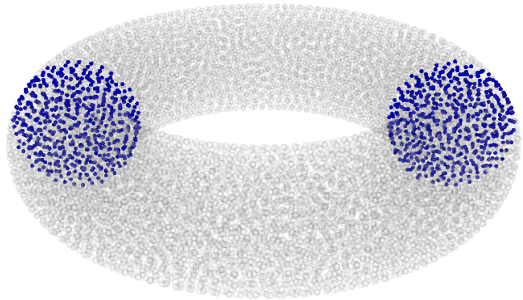
(a) Torus $r = 0.32, R = 1$.



(b) Boundary nodes.



(c) Interior nodes.

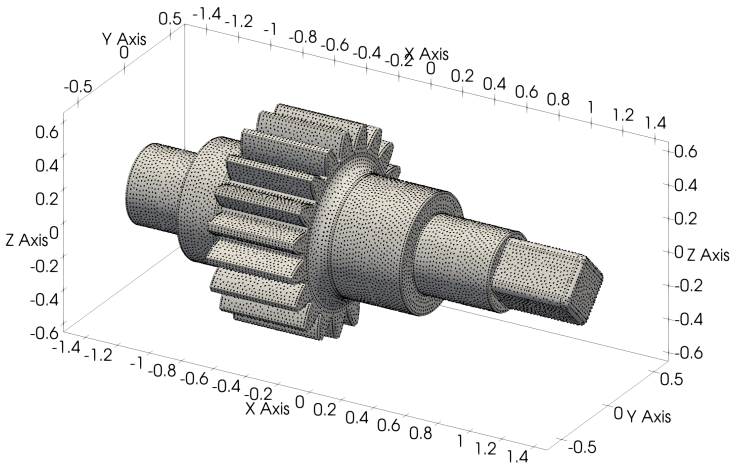


(d) Section of the nodes.

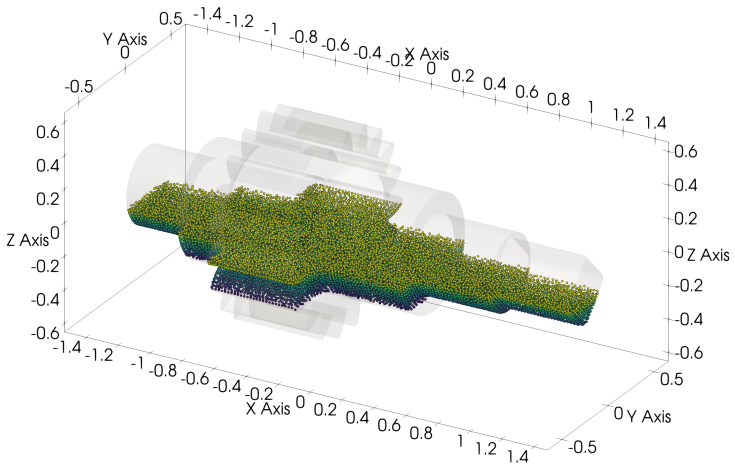
Figure: Nodes produced by advancing front method on a torus, $h = 0.05$.

Node generation (shameless advertisement)

Advancing front method available as C++/MATLAB library at
`https://github.com/BrunoDegliEsposti/NodeGenLib`
The code natively supports STEP files, the standard in CAD.
Does anyone here need to train PINNs on CAD models?



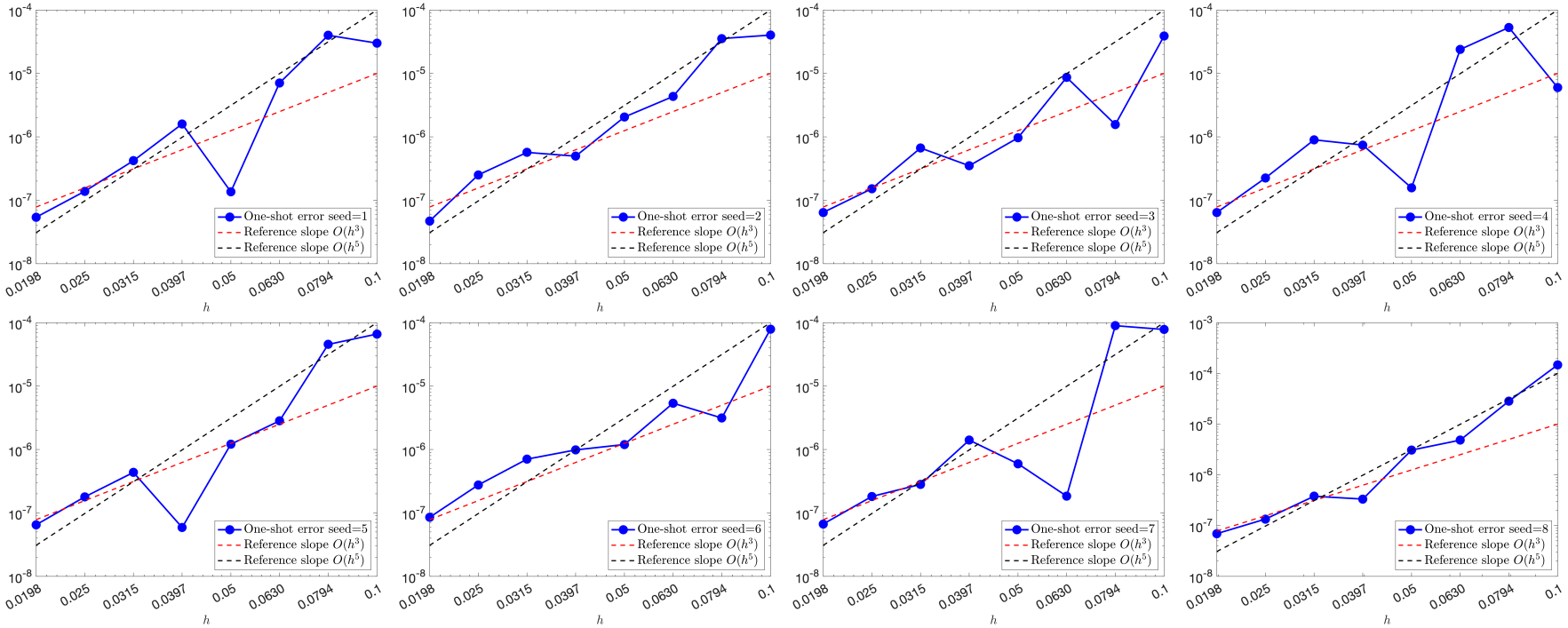
(a) Nodes on surface of gear shaft



(b) Nodes inside gear shaft

Noisy errors

Quadrature errors on the torus for 8 different seeds:

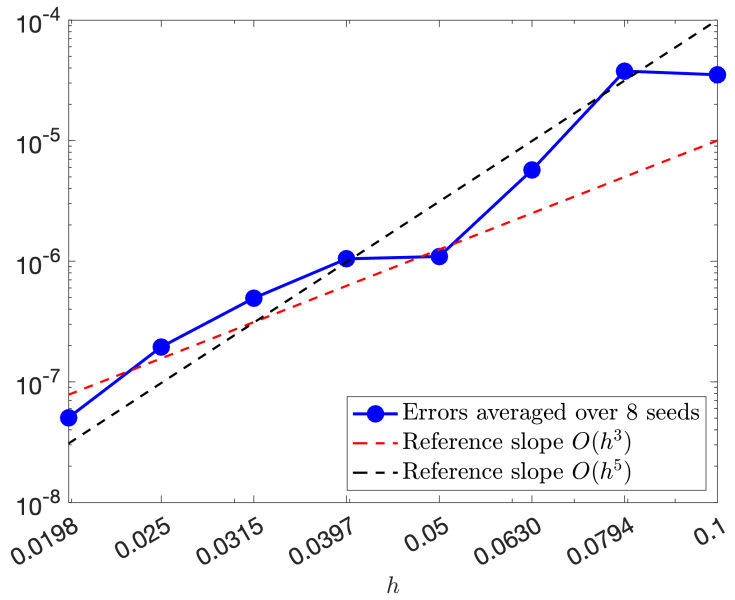


Too noisy to publish! Remember: cherry picking seeds is scientific fraud.

Standard fix: average errors over seeds to reduce variance.

Noisy errors

Plot of averaged errors. Least squares line fit estimates $k = 4.4$.



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?

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:

$$\mathbb{E}(X_h) \approx c_1 h^{k_1} \quad \sigma(X_h) \approx c_2 h^{k_2} \quad \text{for } h \to 0^+.$$

Normal model for signed errors

Statistical inference of convergence order requires a model. We use the simple three-parameter normal model

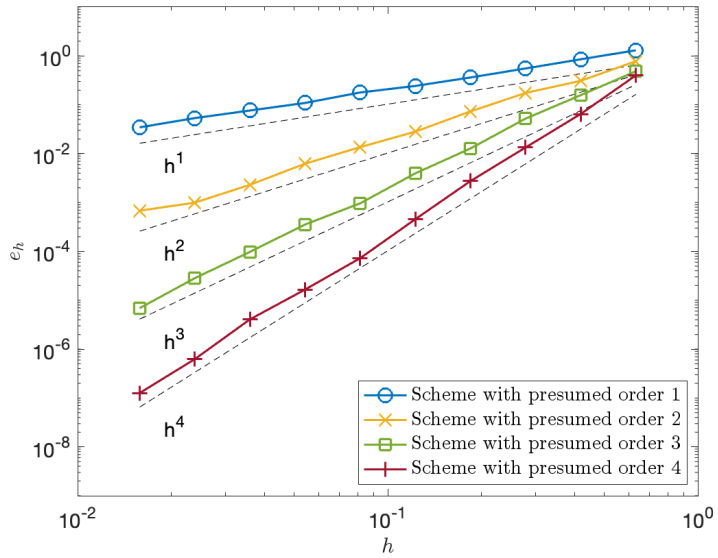
$$X_h \sim \mathcal{N}\left(c_1 h^k, (c_2 h^k)^2\right), \qquad \boldsymbol{\theta} = (c_1, c_2, k).$$

Modeling assumptions

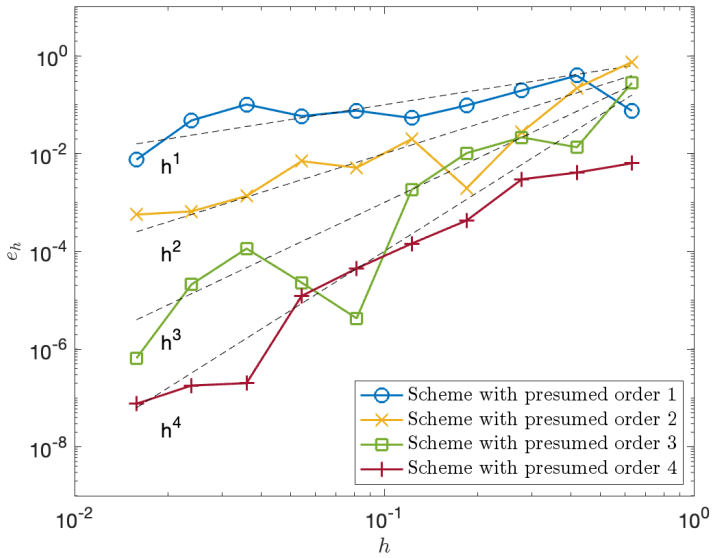
1. Errors are signed, which means that they can take on both positive and negative values. This is the case for quadrature errors.
2. Normally distributed errors. Intuition: the global error is often the combination of many small weakly-dependent local errors.
3. The mean and standard deviation of the errors decay at the same rate, to improve robustness and lower cost of inference.

Normal model for signed errors

The model reproduces both smooth and noisy convergence plots, depending on the ratio c_1/c_2.



(a) $c_1 = 8,\ c_2 = 1,\ k = 1, \ldots, 4.$



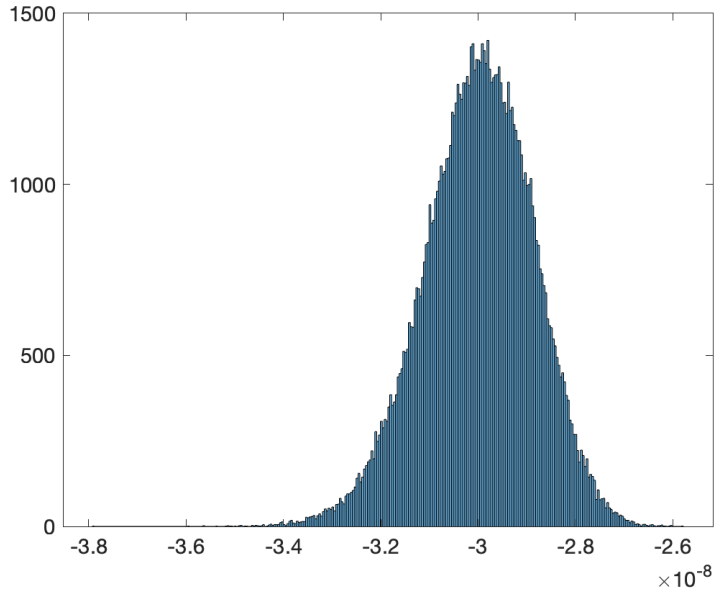
(b) $c_1 = 1,\ c_2 = 1,\ k = 1, \ldots, 4.$

Figure: Absolute values of independent samples of the normal error model.

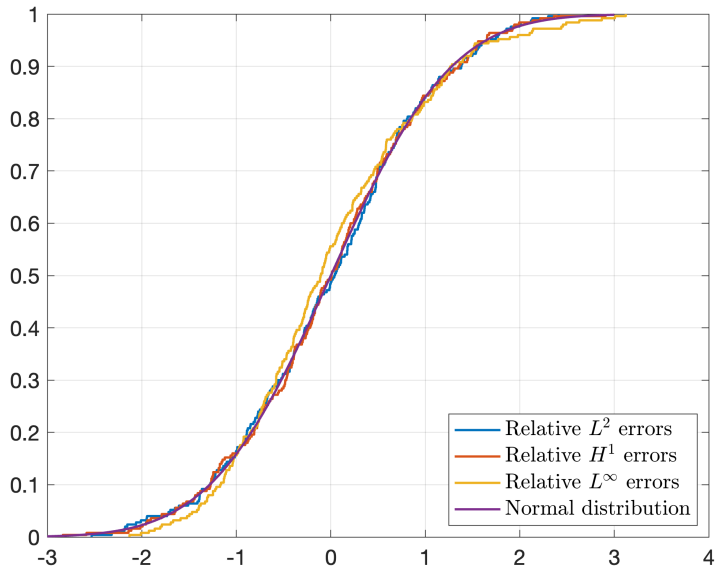
If $c_2 \gg c_1$, $\mathbb{E}(X_h)$ dominates $\sigma(X_h)$ and the error plot looks smooth. Otherwise, $\sigma(X_h)$ is large enough to make convergence erratic.

Normal model for signed errors

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, validating the normality assumption.

Bayesian inference and numerical results

Posterior inference via MCMC

Let $\boldsymbol{x} = (x_1, \ldots, x_n)$ be the measured numerical errors, and $\boldsymbol{h} = (h_1, \ldots, h_n)$ the corresponding discretization levels.

The log-likelihood under the assumed generative model is

$$\log\big(p(\boldsymbol{x} \mid \boldsymbol{\theta})\big) = -\frac{n}{2}\log(2\pi) - \sum_{i=1}^{n}\left[\log(c_2 h_i^k) + \frac{(x_i - c_1 h_i^k)^2}{2c_2^2 h_i^{2k}}\right].$$

We consider weakly informative priors

$$c_1 \sim \mathcal{N}(0,\, \sigma_c^2), \quad c_2^2 \sim \mathcal{IG}(a,\, b), \quad k \sim \mathcal{N}^+(0,\, \sigma_k^2).$$

In our numerical experiments, $\sigma_c^2 = 100$, $a = 2.1$, $b = 1.1$, $\sigma_k^2 = 25$.

Posterior inference for the parameters is performed via a *Metropolis-within-Gibbs* Markov Chain Monte Carlo algorithm.

Posterior inference via MCMC

Algorithm 1: Metropolis-within-Gibbs sampler

Initialize parameters $c_1^{(0)}, c_2^{(0)}, k^{(0)}$ by sampling prior distributions

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

Update c_1 via Metropolis-Hastings step with proposal variance v_c

Update c_2^2 via a Gibbs step using the full conditional distribution

$$c_2^2 \mid c_1, k, \boldsymbol{x} \sim \mathcal{IG}\left(a + \frac{n}{2},\ b + \frac{1}{2}\sum_{i=1}^{n} \frac{(x_i - c_1 h_i^k)^2}{h_i^{2k}}\right)$$

Update k via Metropolis-Hastings step with proposal variance v_k

Output posterior samples after burn-in period and thinning

In our numerical experiments $N = 10^6$, length of burn-in period is $N/2$, and thinning keeps 10% of the samples. Acceptance rate is around 23%.

Posterior inference via MCMC

Validation of the MCMC approach

We fix $\boldsymbol{\theta}$, generate $\boldsymbol{x}$ using our normal model for the errors with parameters $\boldsymbol{\theta}$, and run MCMC to recover $\boldsymbol{\theta}$. For example,

$$k = 2 \text{ and } c_1 \gg c_2 \quad \rightarrow \quad \text{Estimate } k = 2.009 \pm 0.002 \text{ for } n = 10$$

$$k = 2 \text{ and } c_1 \ll c_2 \quad \rightarrow \quad \text{Estimate } k = 2.022 \pm 0.058 \text{ for } n = 10$$

The assumption $k_1 = k_2 = k$ is critical for the success of our approach in the case $c_1 \ll c_2$: a four-parameter model cannot recover c_1 and k_1 because the signal drowns in noise.

Posterior inference via MCMC

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	c_1	c_2	k
1	8	3.95 ± 3.14	25.95 ± 27.63	4.69 ± 0.24
2	16	2.51 ± 1.93	23.85 ± 23.02	4.65 ± 0.21
4	32	2.86 ± 1.83	44.21 ± 31.61	4.73 ± 0.18
8	64	2.01 ± 1.37	36.30 ± 31.03	4.69 ± 0.15

Table: Means and standard deviations of posterior samples.

Conclusion and future work

Conclusions

Bayesian inference can be used to robustly estimate error convergence rates of numerical methods, even in the presence of noise.

- Our estimate of k comes with a measure of uncertainty
- We can decide if more numerical experiments need to be performed based on the uncertainty
- Coefficients c_1 and c_2 of competing numerical methods with the same order of convergence can be compared

Directions for future work

- Extend the model to unsigned errors using folded normals
- Investigate the case $k_1 \neq k_2$

Conclusion and future work

Does your numerical method produce noisy errors?
Drop me an email at `bruno.degliesposti@unifi.it`

Thank you for your attention!

Hopefully not you by now

$\longleftarrow$