# *Supplementary Material*

## 1 CONVEXITY PROOF

Consider a negative monotonic function $f\colon \mathbb{R}^+ \to \mathbb{R}^+$ which relates signal strength $s$ to a parameter credible interval $\sigma$. We assume that for strong signals a further improvement in signal strength should result in a smaller decrease of the credible interval than for a weak signal. This means that the rate by which the absolute derivative of $f$ changes $\frac{\mathrm{d}}{\mathrm{d}s}|\frac{\mathrm{d}f}{\mathrm{d}s}|$ must be negative. We can write

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}s}\Big|\underbrace{\frac{\mathrm{d}f}{\mathrm{d}s}}_{<0}\Big| < 0 \\ \Leftrightarrow\ &-\frac{\mathrm{d}^2 f}{\mathrm{d}s^2} < 0 \\ \Leftrightarrow\ &\frac{\mathrm{d}^2 f}{\mathrm{d}s^2} > 0 \end{aligned} \tag{S1}$$

which proves that $f$ must be convex.

## 2 SUPPLEMENTARY FIGURES

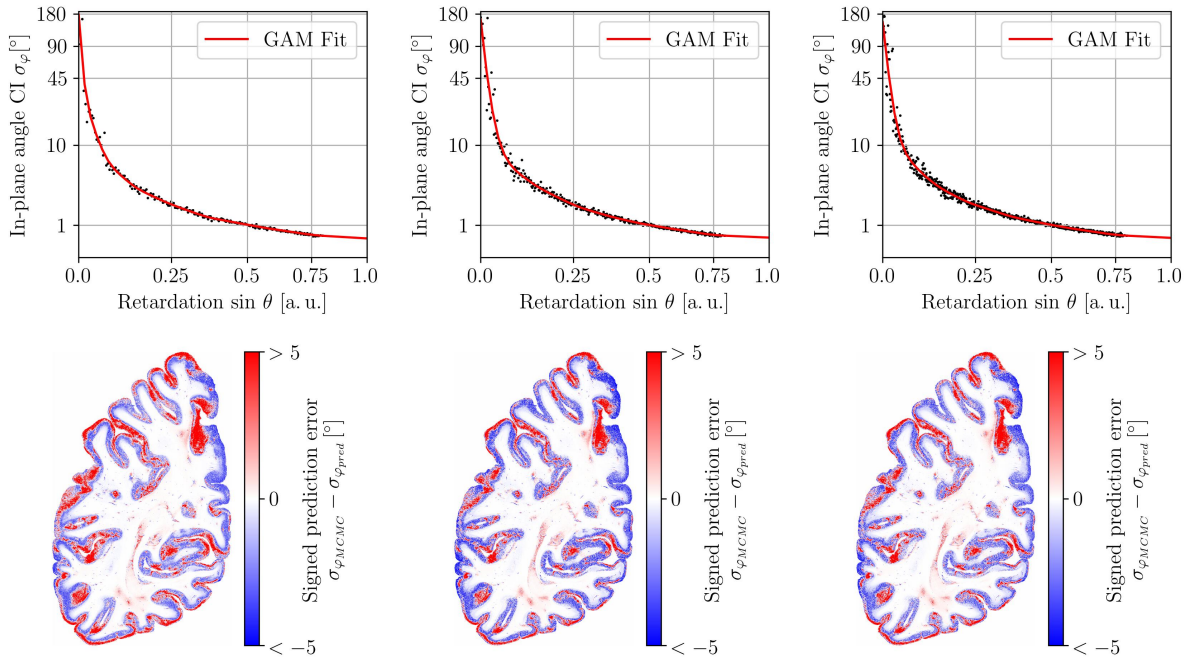


**Figure S1.** Evaluation of the GAM fit and prediction accuracy for the in-plane angle uncertainty $\sigma_\varphi$ for different sample sizes (200, 400, and 800). **Top row**: training data and GAM fit. **Bottom row**: prediction error map.

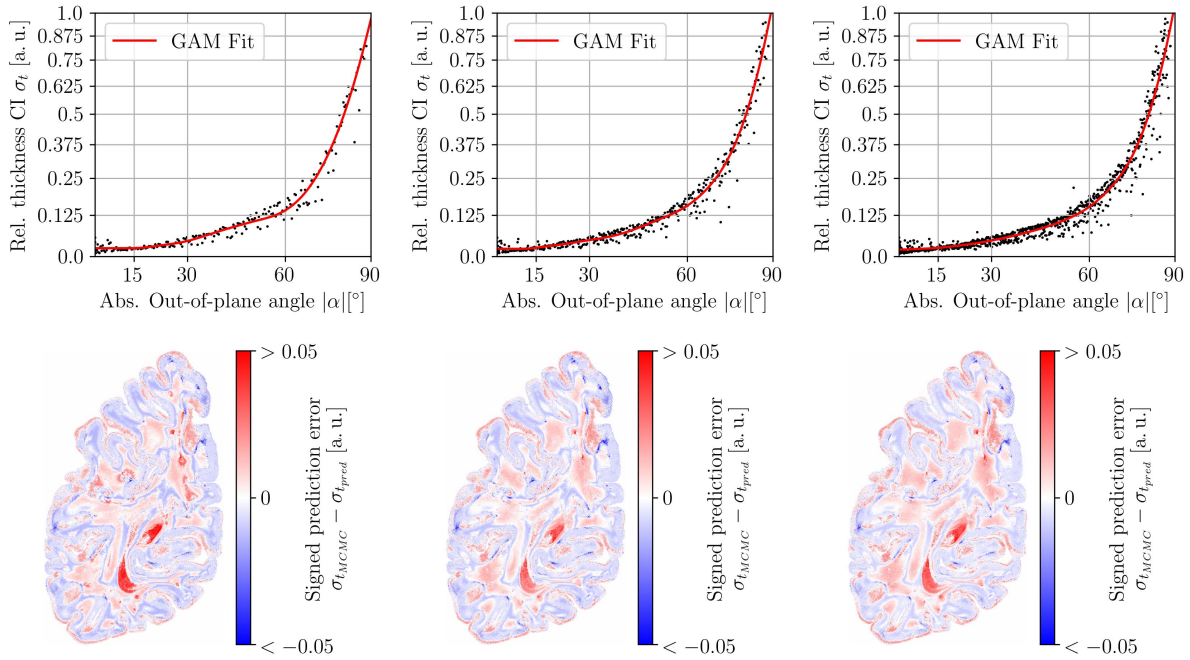


**Figure S2.** Evaluation of the GAM fit and prediction accuracy for the rel. thickness uncertainty $\sigma_t$ for different sample sizes (200, 400, and 800). **Top row**: GAM fits for one brain section with N = 200, 400, and 800 samples. **Top row**: training data and GAM fit. **Bottom row**: prediction error map.

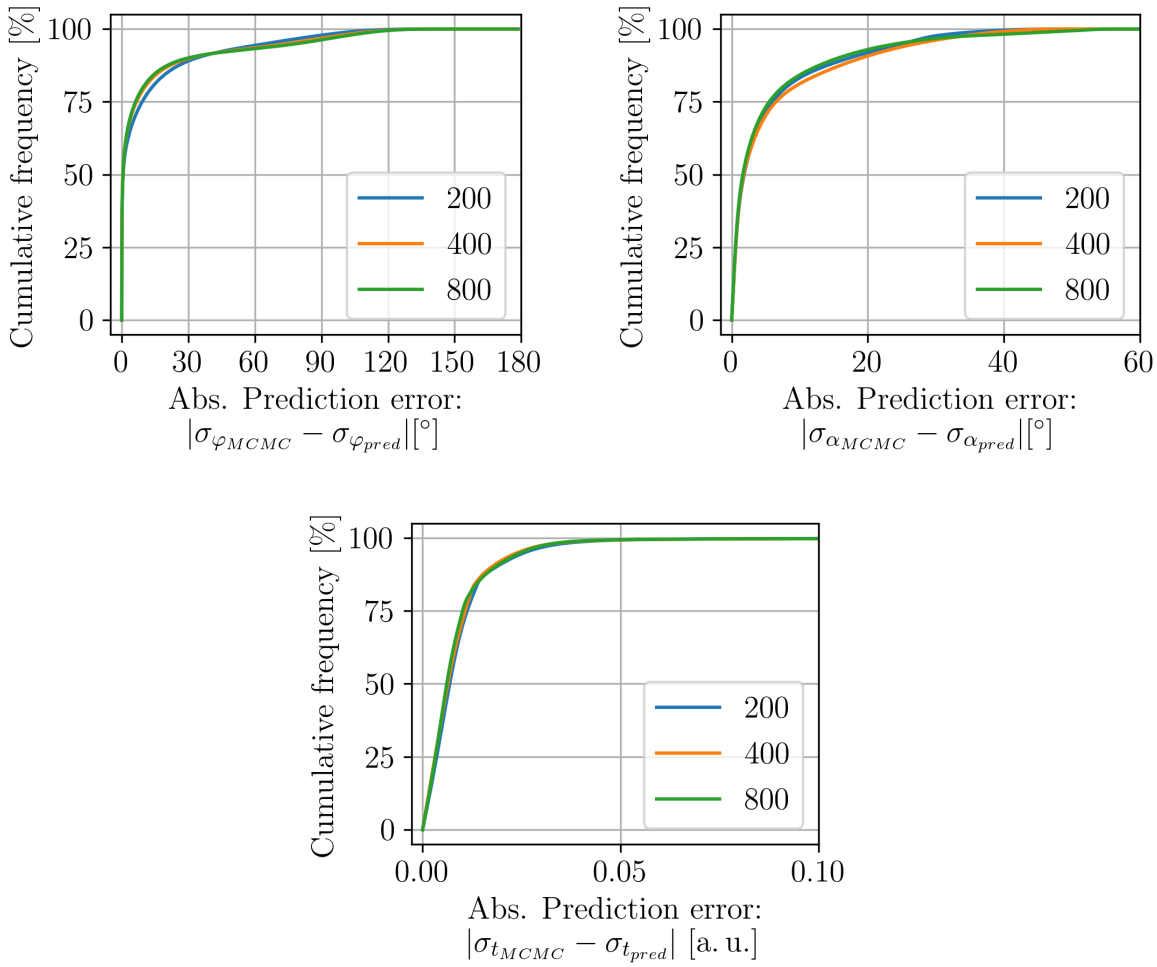


**Figure S3.** Cumulative histograms of the absolute prediction errors of one exemplary brain section for different training sample sizes.

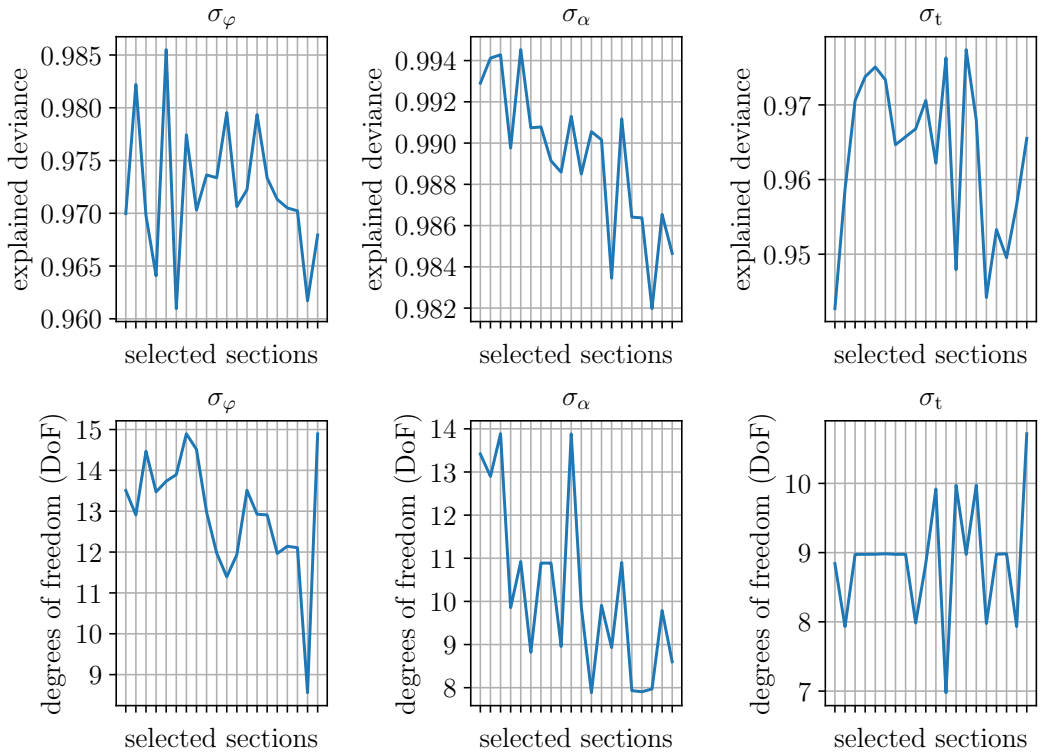


**Figure S4.** Statistical measures of the fitted GAM for sample size $N = 800$. **Top row**: Explained Deviance. **Bottom row**: Degrees of freedom (DoF).

## 3 COMPUTATIONAL DETAILS

In this section details regarding the implementation of MCMC sampling and GAM fitting which are relevant for the reported runtimes are presented. All processing was done in Python 3.6 using double floating point precision.

### 3.1 Hardware

The MCMC sampling was computed on JURECA [1], a modular supercomputer which consists of 1872 nodes with Two Intel Xeon E5-2680 v3 Haswell CPUs per node and 128 GiB, 256 GiB, and 512 GiB DDR4 memory (2133 MHz). Each processor has 12 cores at 2.5 GHz which support Intel Hyperthreading Technology (Simultaneous Multithreading) and offers AVX 2.0 ISA extensions.

### 3.2 MCMC sampling

For MCMC sampling, the `emcee` implementation of the ensemble sampler was utilized [2, 3]. The sampling parameters were set as 100 parallel chains and 500 sampling steps each resulting in 50.000 samples [4]. As the computational bottleneck of MCMC algorithms is the computation of the posterior probability, the posterior calculation was accelerated using the `numba` just-in-time compiler which achieves similar computation times as code written in compiled languages [5]. The runtime amounts to about 1.07 s per pixel. Computations were pixelwise parallelized using `mpi4py` [6].

### 3.3 Generalized Additive Model implementation

The GAM model is fitted using `pygam`'s *gridsearch* function. The parameter optimization via penalized iteratively reweighted least squares dominantly relies on linear algebra operations operating on sparse matrices. `pygam` implements these via `numpy` and `scipy` functions which call highly optimized Basic Linear Algebra Subroutines (BLAS) routines [7, 8, 9].

## REFERENCES

[1]Dorian Krause and Philipp Thörnig. Jureca: Modular supercomputer at jülich supercomputing centre. *Journal of large-scale research facilities JLSRF*, 4, 07 2018.

[2]D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman. emcee: The mcmc hammer. *PASP*, 125:306–312, 2013.

[3]Jonathan Goodman and Jonathan Weare. Ensemble samplers with affine invariance. *Commun. Appl. Math. Comput. Sci.*, 5(1):65–80, 2010.

[4]Daniel Schmitz, Thomas Lippert, Katrin Amunts, and Markus Axer. Quantification of fiber orientation uncertainty in polarized light imaging of the human brain. In Guang-Hong Chen and Hilde Bosmans, editors, *Medical Imaging 2020: Physics of Medical Imaging*, volume 11312, pages 781 – 789. International Society for Optics and Photonics, SPIE, 2020.

[5]Siu Kwan Lam, Antoine Pitrou, and Stanley Seibert. Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*, LLVM '15, pages 7:1–7:6, New York, NY, USA, 2015. ACM.

[6]Lisandro Dalcín, Rodrigo Paz, and Mario Storti. Mpi for python. *Journal of Parallel and Distributed Computing*, 65(9):1108 – 1115, 2005.

[7]Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30, 2011.

[8]P. Virtanen et al. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[9]L Susan Blackford, Antoine Petitet, Roldan Pozo, Karin Remington, R Clint Whaley, James Demmel, Jack Dongarra, Iain Duff, Sven Hammarling, Greg Henry, et al. An updated set of basic linear algebra subprograms (blas). *ACM Transactions on Mathematical Software*, 28(2):135–151, 2002.