Uncertainty quantification of experimentally-derived quantities

M.A. Chilenski, M. Greenwald, N.T. Howard, A.E. White,
J.W. Hughes, J.R. Walk, J.E. Rice, M.L. Reinke, and C. Gao

MIT PSFC/Alcator C-Mod

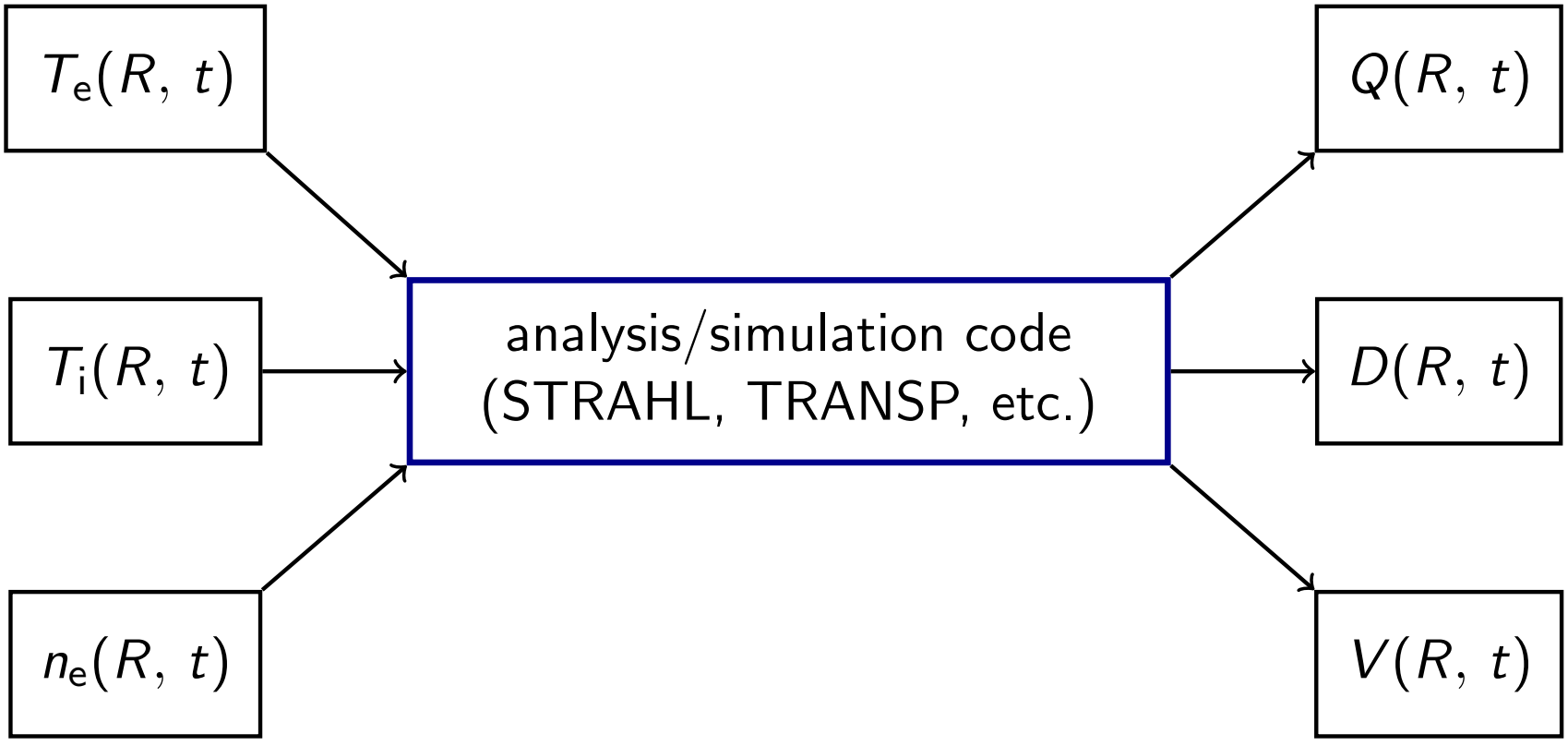
APS-DPP, Denver, CO
November 14, 2013
Poster TP8.00038

Supported by USDoE award DE-FC02-99ER54512. Supported in part by a DOE SCGF fellowship, administered by ORISE-ORAU under contract DE-AC05-06OR23100.

Overview: What do we know, *and how well do we know it?*

- Many quantities are not measured directly, instead are inferred using complicated analysis codes (e.g., Q_i, Q_e from TRANSP [1] or D_Z, V_Z from STRAHL [2]).
- Credible error estimates are critical when comparing these results to simulations/theory.
- A variety of techniques are being tested to obtain error estimates in a rigorous, automated manner.
- This poster presents recent progress in using Gaussian process regression (GPR) to fit profiles and extract samples.
- The samples are used with STRAHL to get uncertainties on impurity transport coefficients D, V for injected Ca in an Alcator C-Mod L-mode.

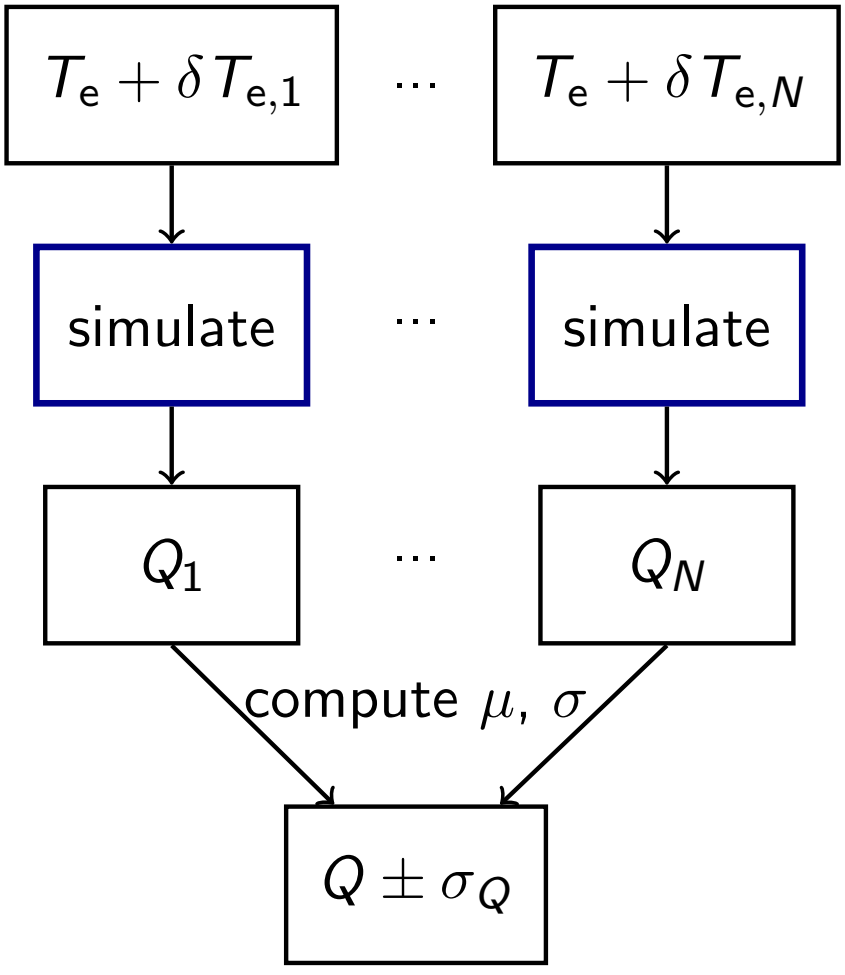
Motivation: Uncertainty quantification with profile inputs



Objective

Given the inputs and their uncertainties, measured at discrete points, what are the outputs and their uncertainties?

Typical scheme: sampling to estimate uncertainty

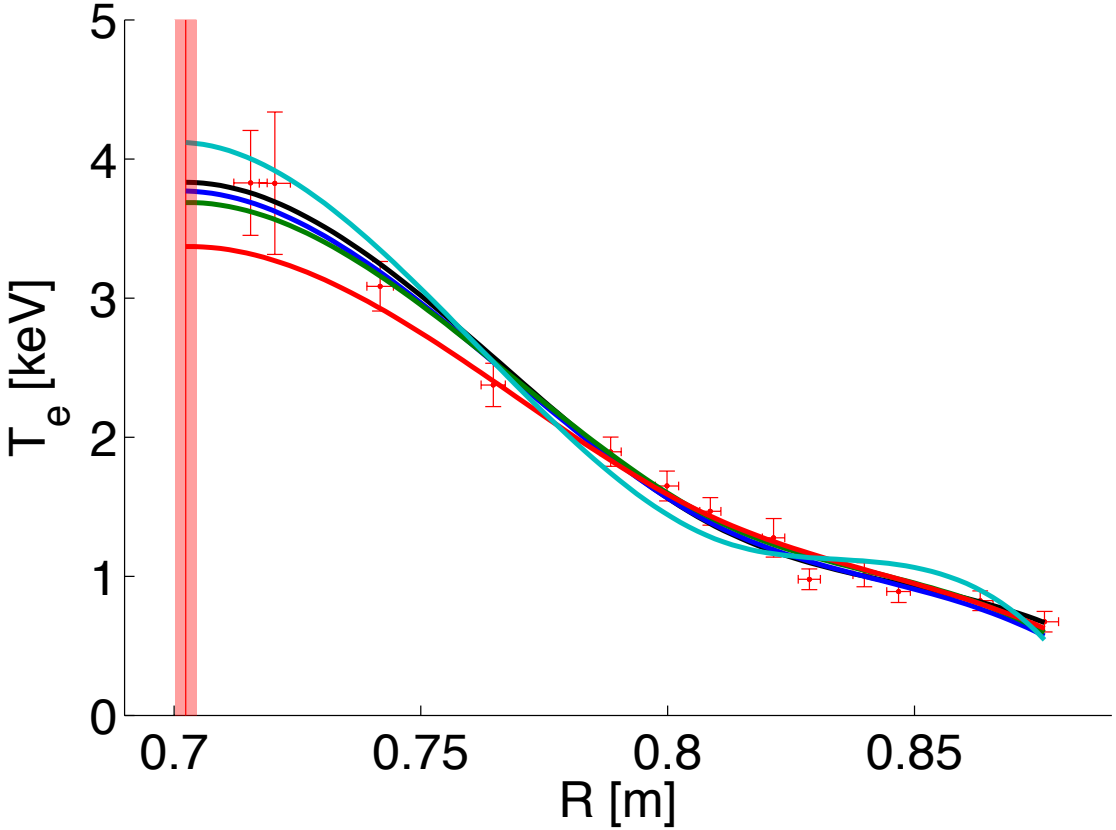


- T_e is the fitted estimate for the input quantity.
- $\delta T_{e,N}$ is a random perturbation, distributed according to the uncertainty estimate on T_e.
- The samples can be generated using a variety of strategies, including naive Monte Carlo, Latin hypercube and quasi-Monte Carlo sampling [3].
- Q_N is a possible realization of the the output quantity.

What advanced UQ techniques, GPR can contribute

- Provide better confidence in error estimates.
- Obtain reliable results with fewer expensive simulation runs.
- Provide statistically defensible, automated fits to *entire* profiles without the need for time-consuming manual tuning.

Option: Perturb individual data points, fit with spline



- Often requires manual supervision.
- Goodness versus complexity of fit requires extra care.
- Handling multivariate data ($f(R, Z, \phi, t)$) is painful.

Notes on the plots

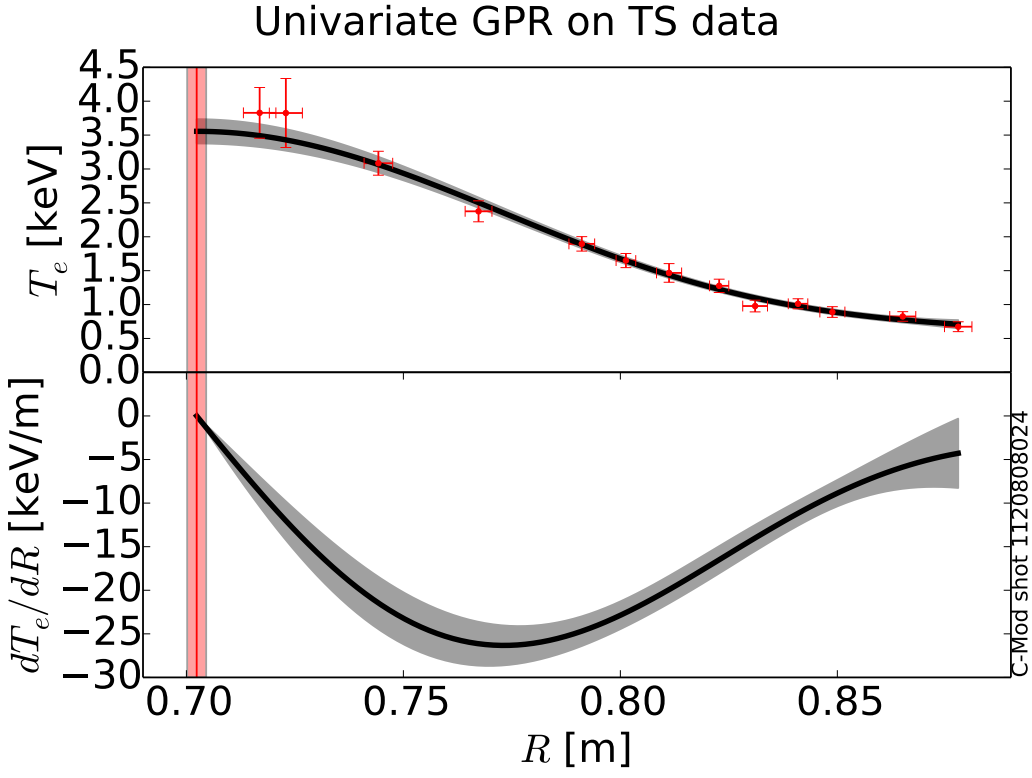
- Vertical red line indicates the magnetic axis.
- Data points are the average in a TS channel over the flattop
- Vertical error bars are $\pm 1\sigma$ of T_e within a channel. Horizontal error bars are $\pm 1\sigma$ of mapped R_{mid} *but are not included in the analysis*.

Gaussian process regression (GPR) provides a better way to fit profiles and produce input samples

GPR is a Bayesian non-parametric regression technique [4]

Bayesian: Prior encodes assumptions about the data.

Non-parametric: Data are not reduced into parameters.



- Distribution of functions, can sample directly.
- Variance gives the uncertainty in the fit.
- Simple to get gradients **and their uncertainties**.
- Generalization to multivariate data is trivial.

Terms and symbols used

GPR:	Gaussian process regression
training data:	The data to be fit: $\boldsymbol{y}(\mathrm{X})$, $\boldsymbol{y} \in \mathbb{R}^n$, $\mathrm{X} \in \mathbb{R}^{D \times n}$
test data:	The predicted points: $\boldsymbol{y}_*(\mathrm{X}_*)$, $\boldsymbol{y}_* \in \mathbb{R}^{n_*}$, $\mathrm{X}_* \in \mathbb{R}^{D \times n_*}$
covariance function:	Function giving the covariance between two points: $k(\boldsymbol{x}, \boldsymbol{x}')$, $\boldsymbol{x}$, $\boldsymbol{x}' \in \mathbb{R}^D$
covariance matrix:	k evaluated between all cases: $\mathrm{K} = k(\mathrm{X}, \mathrm{X})$, $\mathrm{K} \in \mathbb{R}^{n \times n}$, $\mathrm{K}_* = k(\mathrm{X}, \mathrm{X}_*)$, $\mathrm{K}_* \in \mathbb{R}^{n \times n_*}$
hyperparameters:	The parameters of the covariance function k.

In other words:

- n observations of quantity y are taken at n (D-dimensional) locations $\boldsymbol{x}$ and combined into the vector $\boldsymbol{y}$ and matrix X.
- Predictions y_* are then made at n_* locations $\boldsymbol{x}_*$ and combined into vector $\boldsymbol{y}_*$ and matrix X_*.
- The matrix K consists of the covariance function k evaluated pairwise between each of the points $\boldsymbol{x}$ in X.

A Gaussian process is a *distribution over functions*

$$f \sim \mathcal{GP}(m(\boldsymbol{x}),\, k(\boldsymbol{x},\, \boldsymbol{x}'))$$

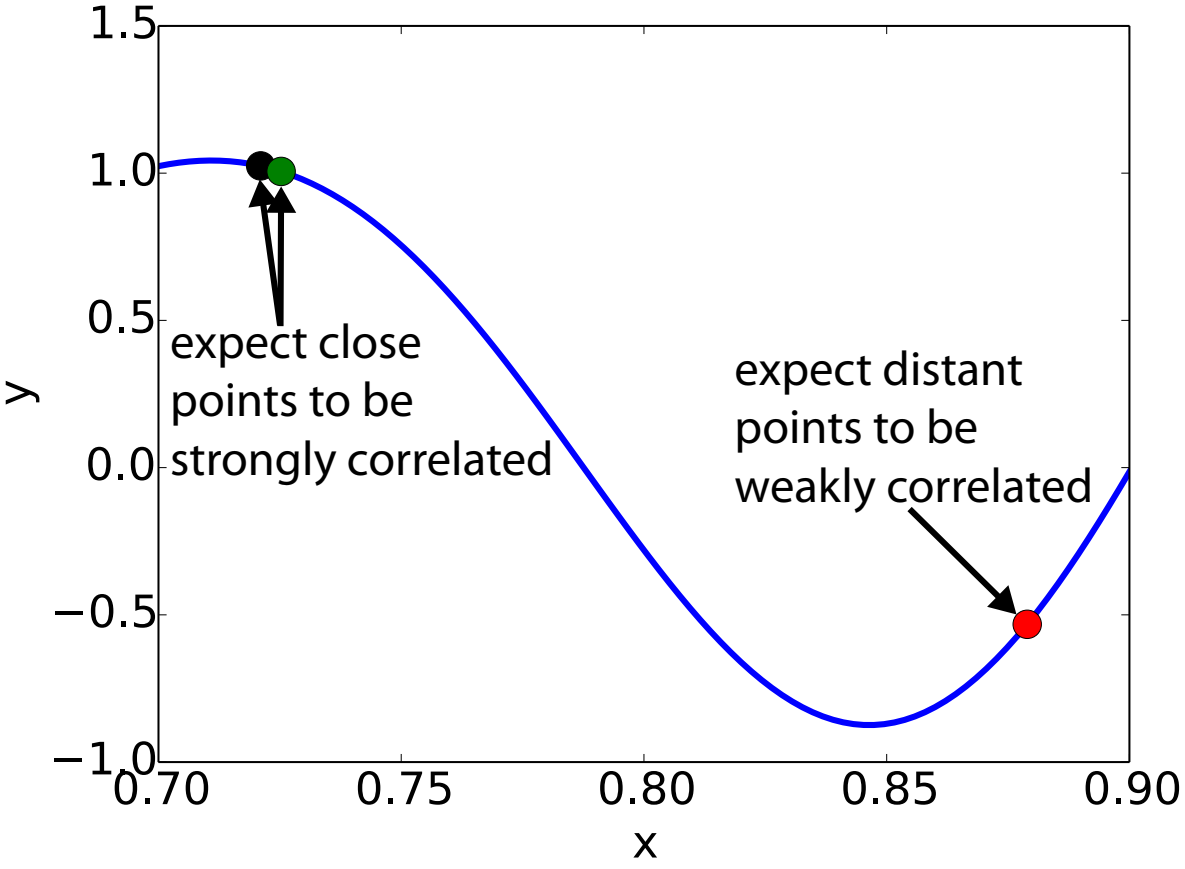
- $m(\boldsymbol{x})$ is the mean function
- $k(\boldsymbol{x},\, \boldsymbol{x}')$ is the covariance function

For any set of points X, the value of $\boldsymbol{y} = f(\mathrm{X})$ is distributed as

$$\boldsymbol{y} \sim \mathcal{N}(m(\mathrm{X}),\, k(\mathrm{X},\, \mathrm{X}))$$

This is an n-dimensional multivariate normal, where $\boldsymbol{y} \in \mathbb{R}^n$.

The covariance function determines the spatial correlation between points



Common choice: squared exponential (SE)

$$k_{\mathrm{SE}}(r) = \sigma_f^2 \exp\left(-\frac{r^2}{2\ell^2}\right)$$

$$r = \left|\boldsymbol{x} - \boldsymbol{x}'\right|$$

ℓ sets the length scale of drop-off in covariance, **is not the same as the gradient scale length!**

Conditioning the prior on the training data

Prior: zero-mean multivariate normal

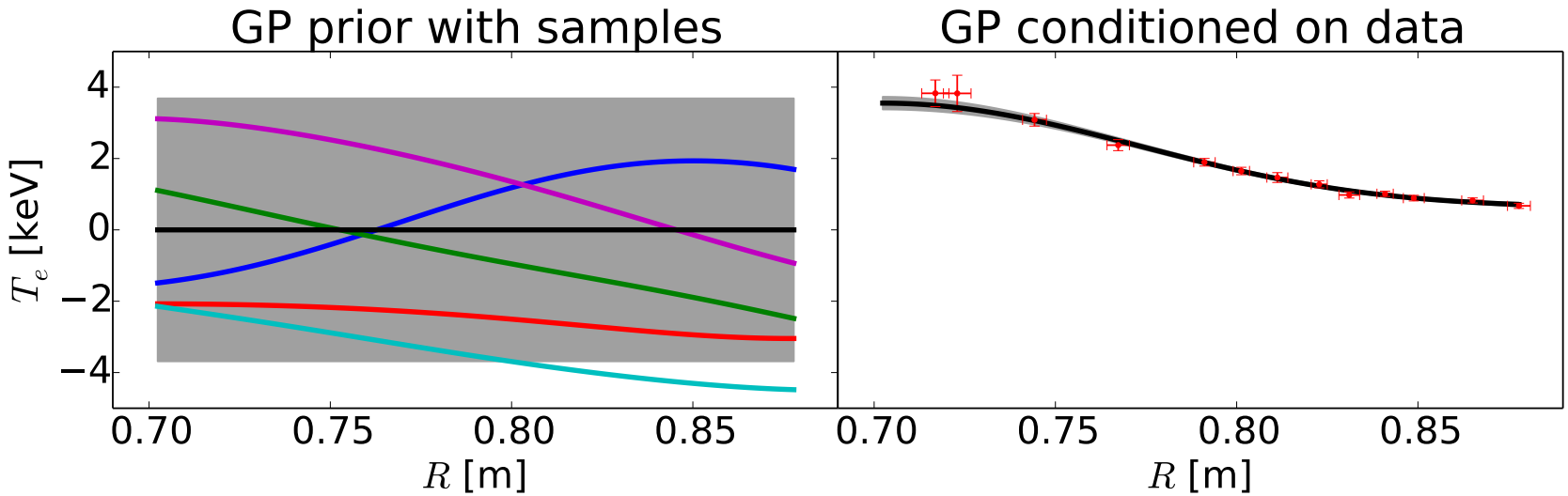
$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathsf{K}(\mathsf{X}, \mathsf{X}) + \Sigma_\mathsf{n} & \mathsf{K}(\mathsf{X}, \mathsf{X}_*) \\ \mathsf{K}(\mathsf{X}_*, \mathsf{X}) & \mathsf{K}(\mathsf{X}_*, \mathsf{X}_*) \end{bmatrix}\right)$$

Σ_n is the noise covariance matrix.

Conditioning

Predictive distribution: prior conditioned on the training data

$$\boldsymbol{y}_*|\mathsf{X}, \boldsymbol{y} \sim \mathcal{N}(\mathsf{K}(\mathsf{X}_*, \mathsf{X})[\mathsf{K}(\mathsf{X}, \mathsf{X}) + \Sigma_\mathsf{n}]^{-1}\boldsymbol{y},$$
$$\mathsf{K}(\mathsf{X}_*, \mathsf{X}_*) - \mathsf{K}(\mathsf{X}_*, \mathsf{X})[\mathsf{K}(\mathsf{X}, \mathsf{X}) + \Sigma_\mathsf{n}]^{-1}\mathsf{K}(\mathsf{X}, \mathsf{X}_*))$$



The predictive distribution contains the fit and its uncertainty

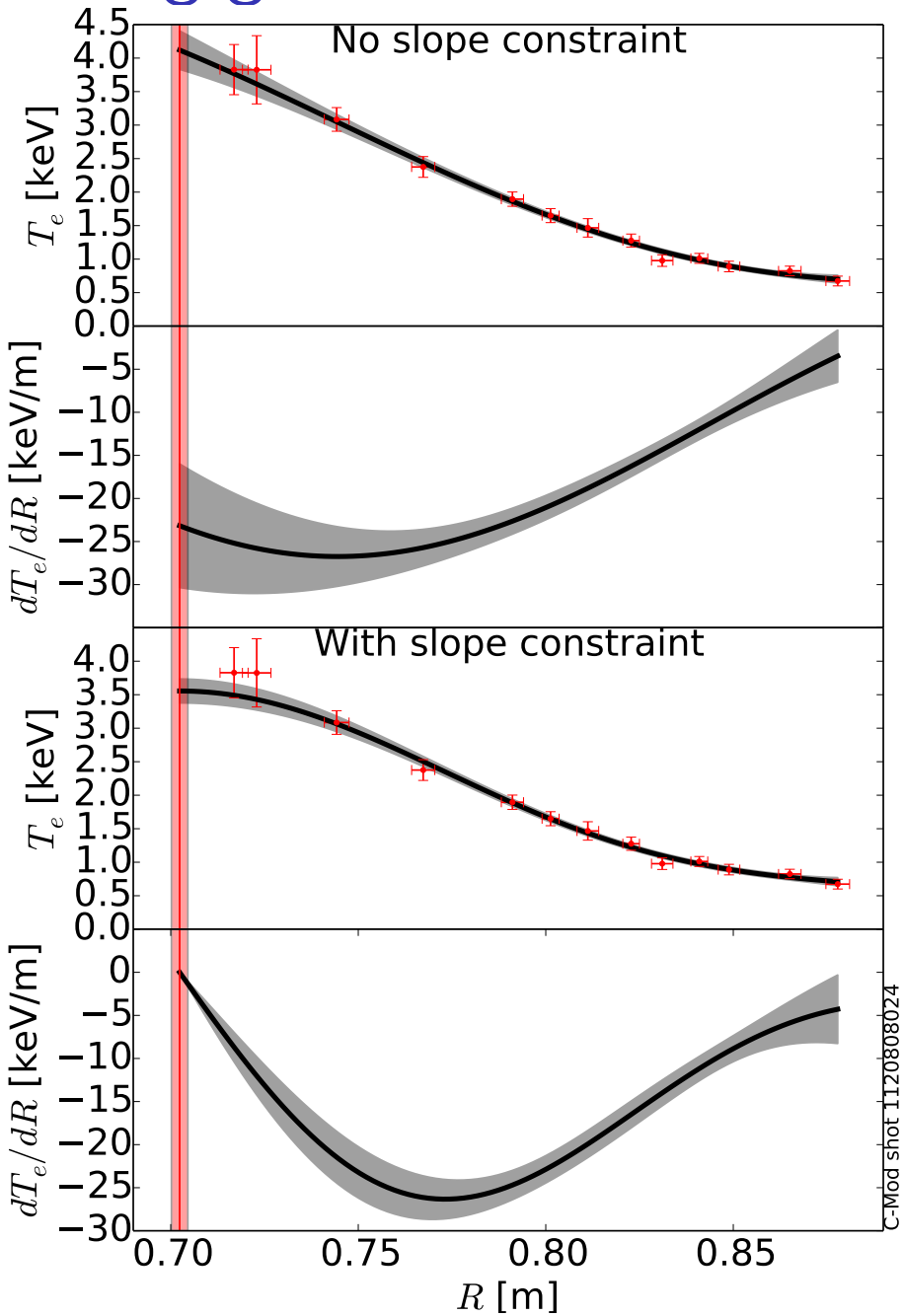
- Mean is linear predictor for $y_*(\boldsymbol{x}_*)$:

$$\begin{aligned}\overline{y}_*(\boldsymbol{x}_*) &= \mathrm{K}(\boldsymbol{x}_*, \mathrm{X})[\mathrm{K}(\mathrm{X}, \mathrm{X}) + \Sigma_\mathrm{n}]^{-1}\boldsymbol{y} \\ &= \sum_{i=1}^{n} \alpha_i k(\boldsymbol{x}_i, \boldsymbol{x}_*) \\ \boldsymbol{\alpha} &= [\mathrm{K}(\mathrm{X}, \mathrm{X}) + \Sigma_\mathrm{n}]^{-1}\boldsymbol{y}\end{aligned}$$

- Diagonal elements of covariance give the uncertainty in the fit:

$$\sigma^2_{y_*} = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \mathrm{K}(\boldsymbol{x}_*, \mathrm{X})[\mathrm{K}(\mathrm{X}, \mathrm{X}) + \Sigma_\mathrm{n}]^{-1}\mathrm{K}(\mathrm{X}, \boldsymbol{x}_*)$$

Getting gradients *and their uncertainties* is straightforward



The derivative of a GP is a GP:

$$\operatorname{cov}\left(y_i, \frac{\partial y_j}{\partial x_{dj}}\right) = \frac{\partial k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{dj}}$$

$$\operatorname{cov}\left(\frac{\partial y_i}{\partial x_{di}}, \frac{\partial y_j}{\partial x_{dj}}\right) = \frac{\partial^2 k(\boldsymbol{x}_i, \boldsymbol{x}_j)}{\partial x_{di}\, \partial x_{dj}}$$

- Derivative equality constraint: just add a datapoint!
- Derivative predictions: predictive distribution contains the uncertainty.

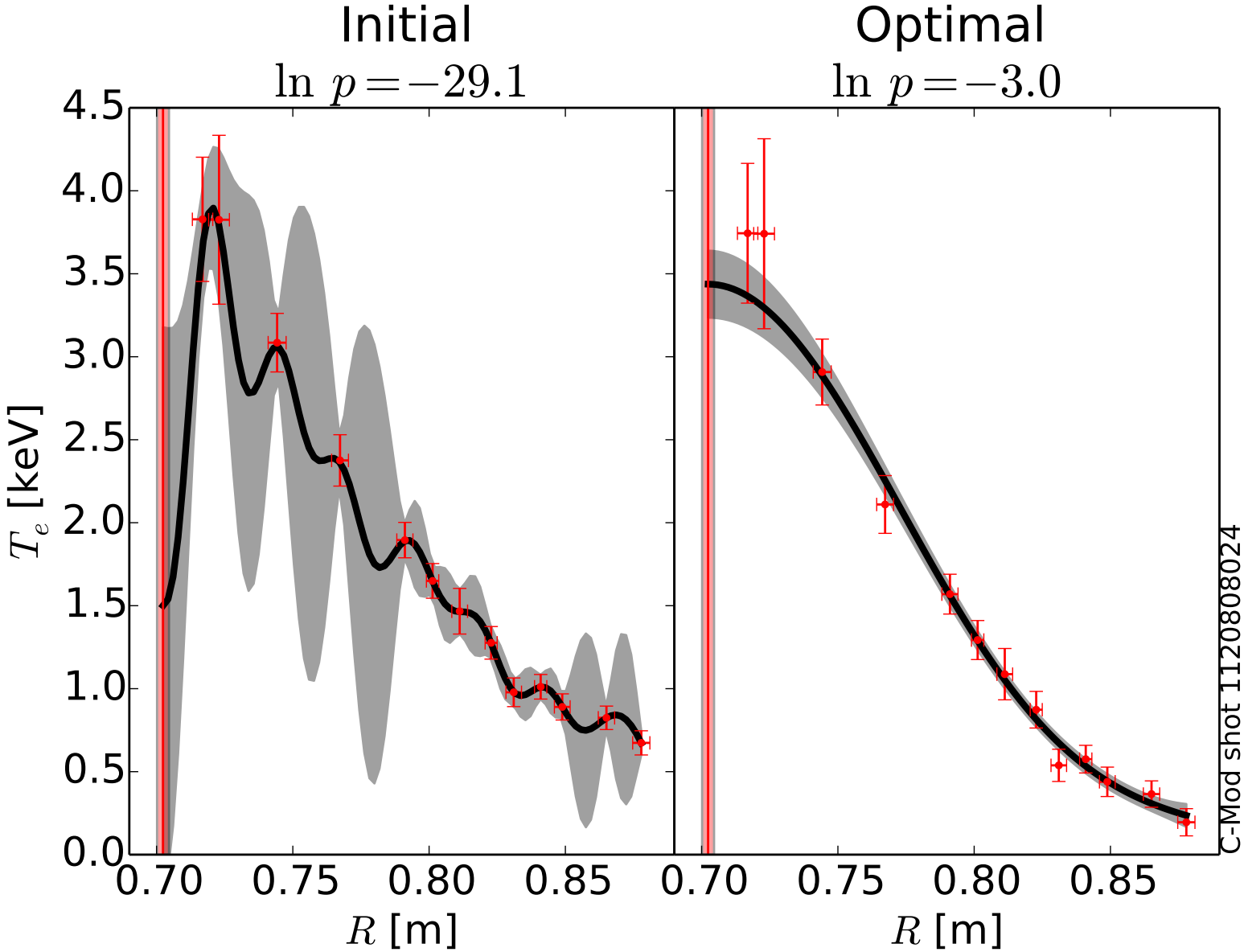
The hyperparameters can be estimated by maximizing the likelihood

Likelihood of the training data given k with hyperparameters $\boldsymbol{\theta} = [\sigma_f,\ \ell,\ \ldots]$:

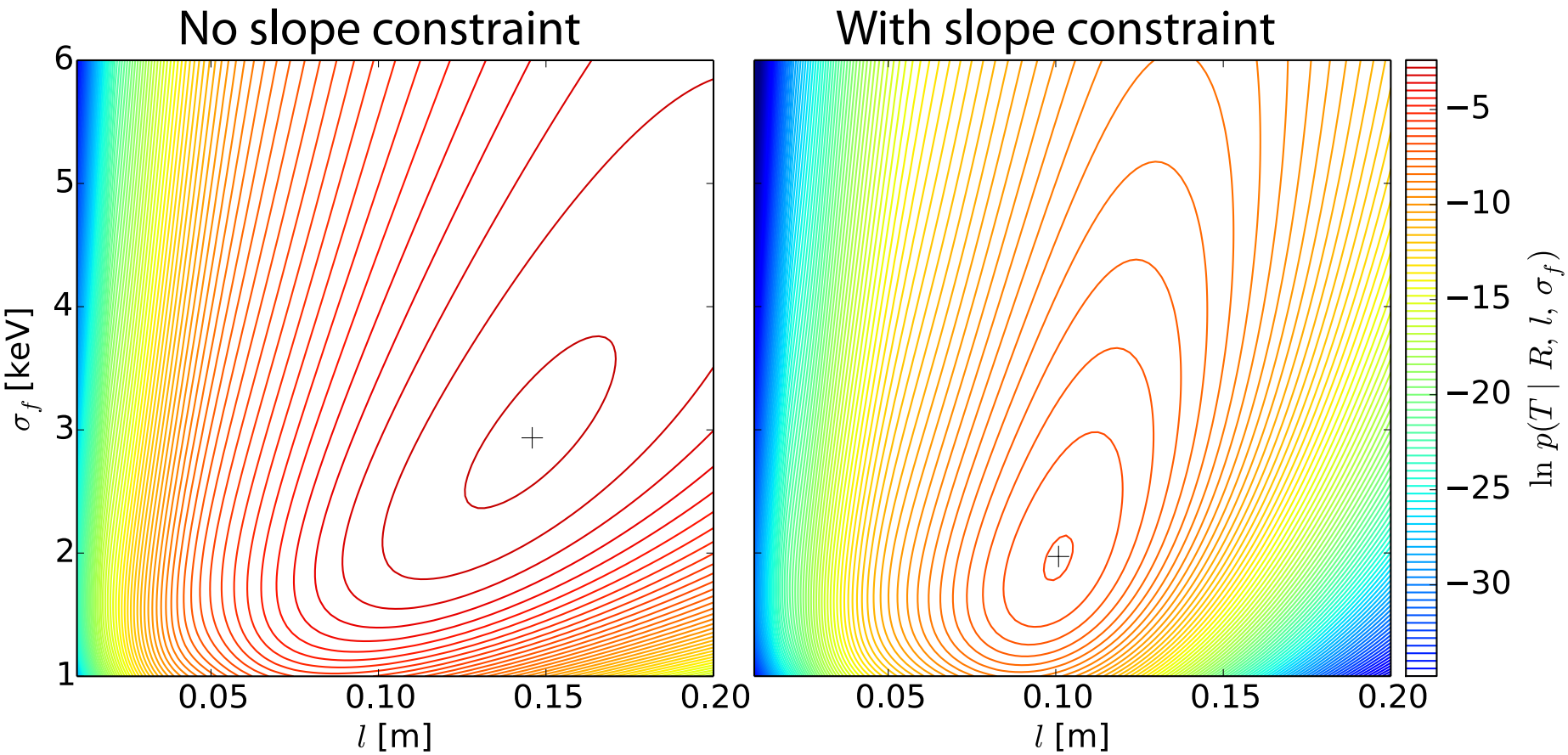
$$\ln p(\boldsymbol{y}|\mathrm{X}, \boldsymbol{\theta}) = -\frac{1}{2}\boldsymbol{y}^{\mathsf{T}}[\mathrm{K} + \Sigma_{\mathrm{n}}]^{-1}\boldsymbol{y} - \frac{1}{2}\ln|\mathrm{K} + \Sigma_{\mathrm{n}}| - \frac{n}{2}\ln 2\pi$$

- Maximize with respect to hyperparameter vector $\boldsymbol{\theta}$.
- Local maxima: different possible interpretations of the data. E.g., noisy and long-ℓ versus precise and short-ℓ
- Compare likelihoods to select the most appropriate kernel.

Bad versus good choices for the hyperparameters have a large effect on the likelihood



The hyperparameter space for the SE kernel appears to be well-behaved



Drawing random samples is straightforward

Just sampling a multivariate normal:

$$\widetilde{\boldsymbol{y}}_* = \boldsymbol{m} + \mathsf{L}\boldsymbol{u}$$

$$\begin{aligned} \boldsymbol{m} = \mathsf{K}(\mathsf{X}_*, \mathsf{X})[\mathsf{K}(\mathsf{X}, \mathsf{X}) + \Sigma_\mathsf{n}]^{-1}\boldsymbol{y} \quad & \text{(the predictive mean)} \\ \mathsf{K}_\mathsf{p} = \mathsf{L}\mathsf{L}^\mathsf{T} \quad & \text{(Cholesky decomposition)} \\ \boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \mathsf{I}) \quad & (n_* \text{ independent standard} \\ & \text{normal variables)} \end{aligned}$$

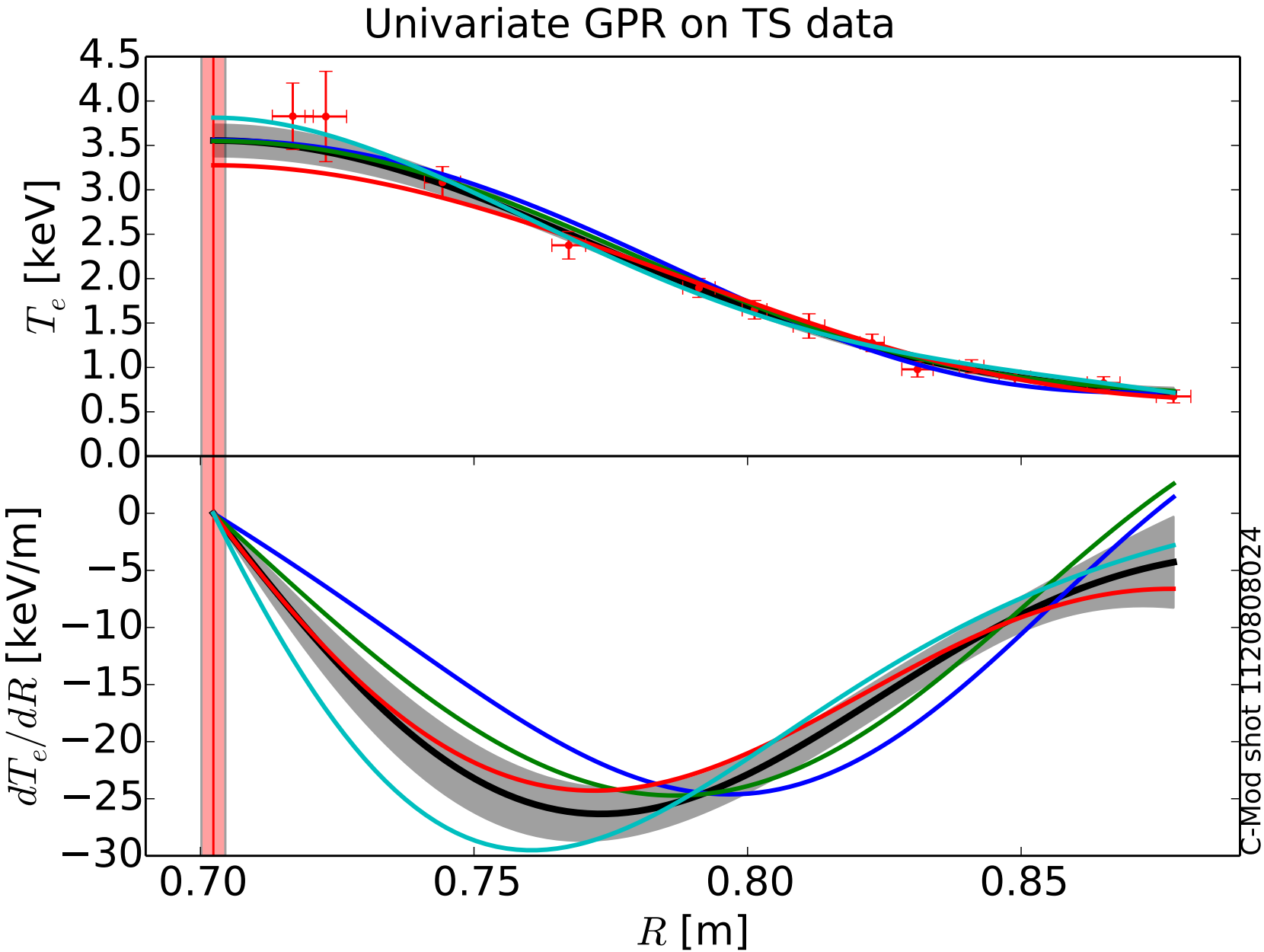
More powerful way of writing the matrix square root:

$$\mathsf{K}_\mathsf{p} = \mathsf{Q}\Lambda\mathsf{Q}^{-1} = \mathsf{Q}\Lambda^{1/2}(\mathsf{Q}\Lambda^{1/2})^\mathsf{T}$$

(Because K_p is symmetric, $\mathsf{Q}^{-1} = \mathsf{Q}^\mathsf{T}$.)

Can truncate eigendecomposition to reduce dimensionality.

Samples of T_e and dT_e/dR can be extracted together

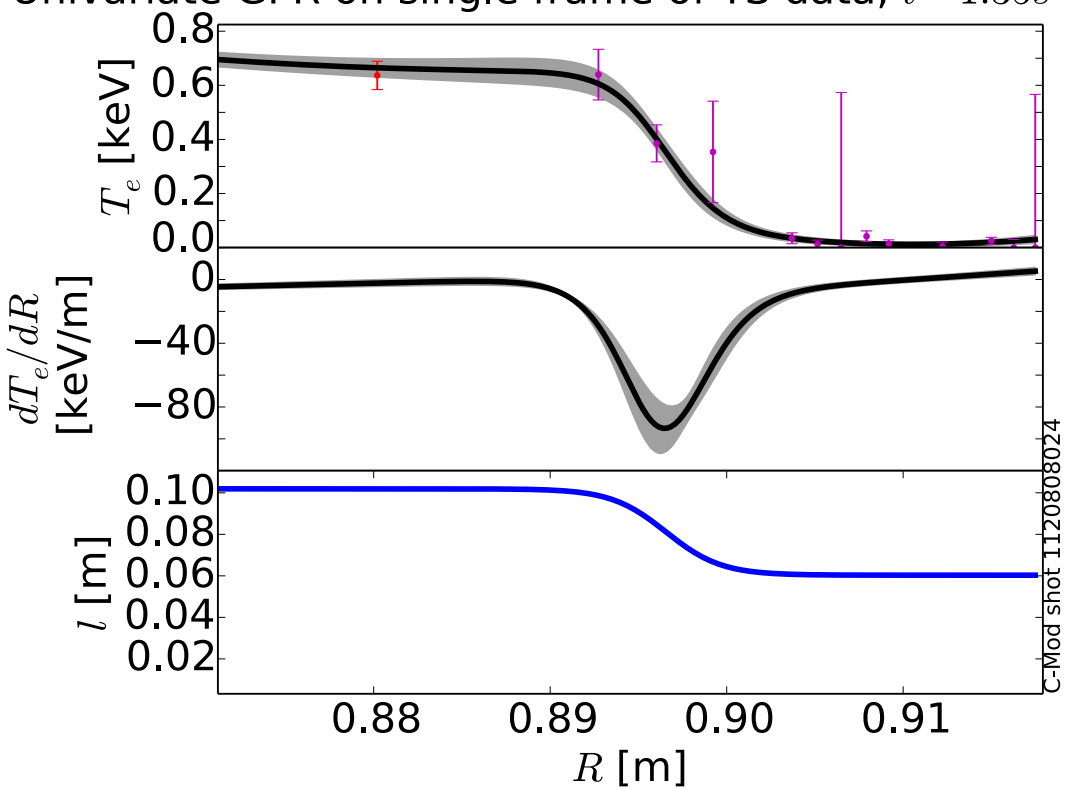


Capturing the pedestal requires a non-stationary kernel

Gibbs kernel [5]: ℓ is an arbitrary function of $\boldsymbol{x}$

$$k_{\mathrm{G}}(\boldsymbol{x}, \boldsymbol{x}') = \sigma_f^2 \left(\frac{2\ell(\boldsymbol{x})\ell(\boldsymbol{x}')}{\ell^2(\boldsymbol{x}) + \ell^2(\boldsymbol{x}')} \right)^{1/2} \exp\left(- \frac{|\boldsymbol{x} - \boldsymbol{x}'|^2}{\ell^2(\boldsymbol{x}) + \ell^2(\boldsymbol{x}')} \right)$$

Univariate GPR on single frame of TS data, $t = 1.35s$



- *Preliminary implementation*
- Length scale:

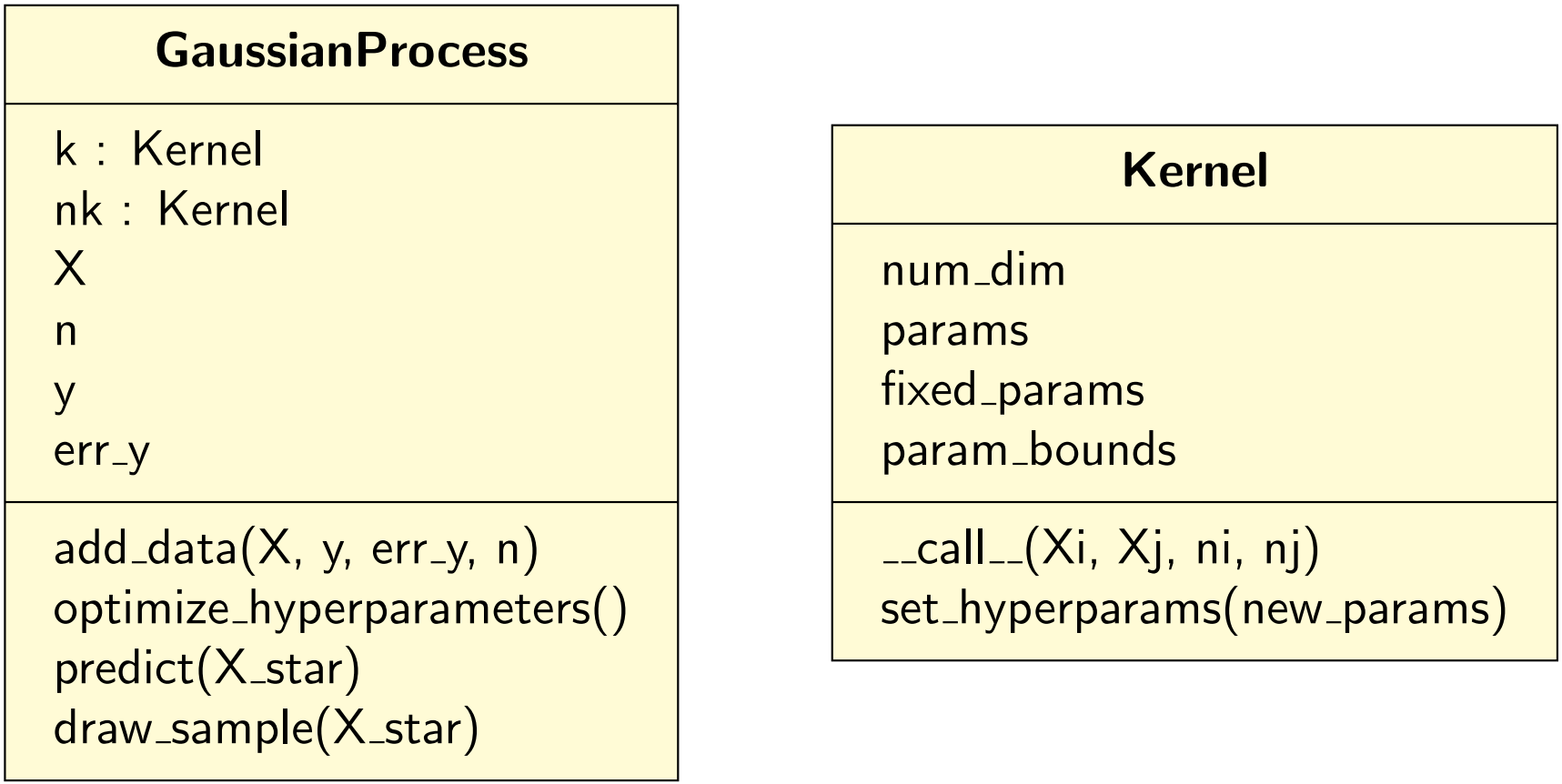
$$\ell = \frac{\ell_1 + \ell_2}{2} - \frac{\ell_1 - \ell_2}{2} \tanh \frac{x - x_0}{\ell_{\mathrm{w}}}$$

- Picked ℓ_1, ℓ_2, ℓ_{w} and x_0 by maximizing ln p.

gptools: An extensible, object-oriented Python package for multivariate GPR including gradients

- Available GPR codes lack one or more critical features:
  - Ability to both constrain and predict gradients.
  - Straightforward way to draw random samples.
- gptools was written to meet these needs:
  - Object-oriented structure.
  - Interface for easy data fusion and application of constraints.
  - SE, Gibbs, Matérn and RQ kernels *with support for arbitrary orders of differentiation*.
- Available on GitHub: www.github.com/markchil/gptools

gptools contains two classes for performing GPR



Impurity transport in Alcator C-Mod is explored using a laser blow-off impurity injector

Controlled injections are a powerful tool to probe transport

- Small (nonperturbative) injection of a non-intrinsic, non-recycling impurity (such as calcium) enables systematic study of impurity transport [6, 7].
- Larger injections are used to induce cold pulses to investigate non-local thermal transport (see C. Gao et al., TP8.00036).
- Injection of Mo used to probe poloidal asymmetries and their effects (see M.L. Reinke et al., TP8.00037).

Hardware overview

- Motorized steering for between-shot positioning.
- Piezoelectric steering for in-shot movement of beam.
- Fast steering and 10 Hz laser repetition rate enables multiple injections into a shot.

Impurity transport coefficients are inferred using STRAHL

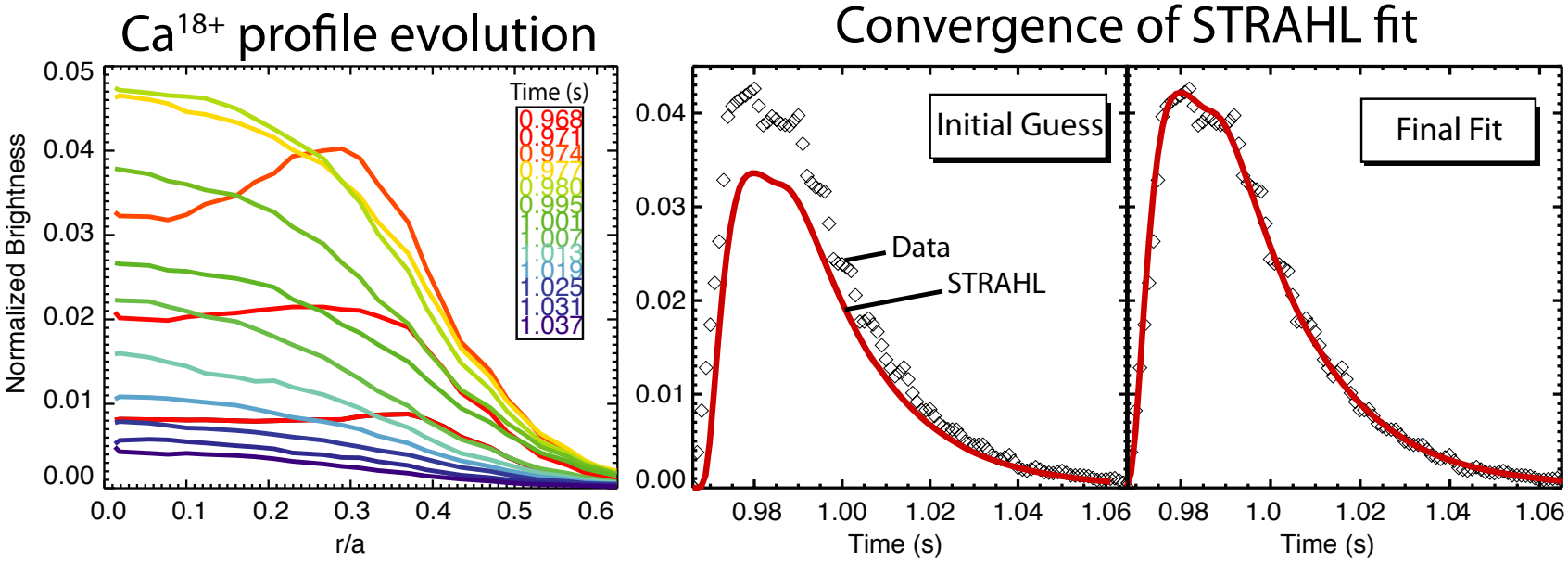
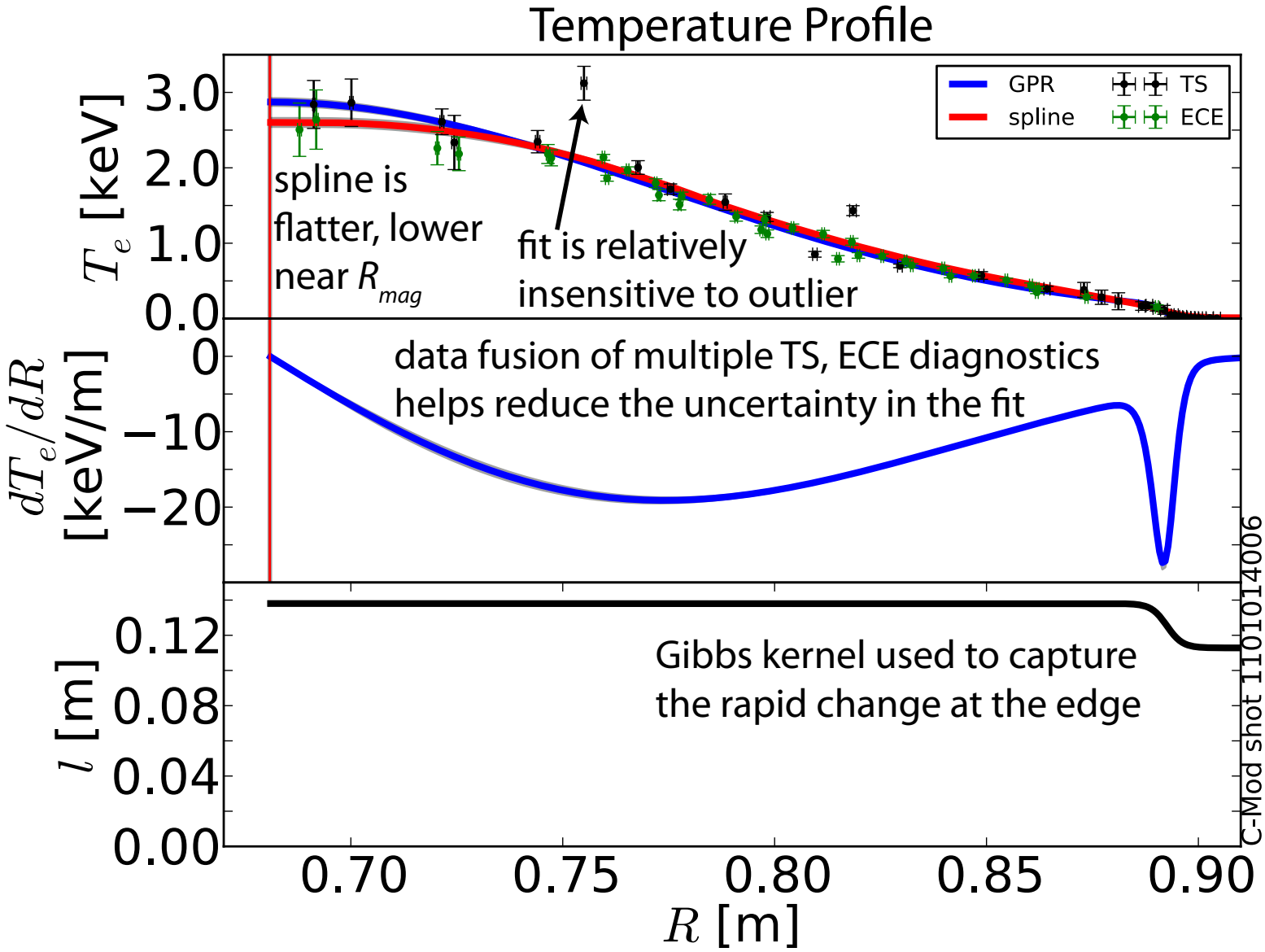
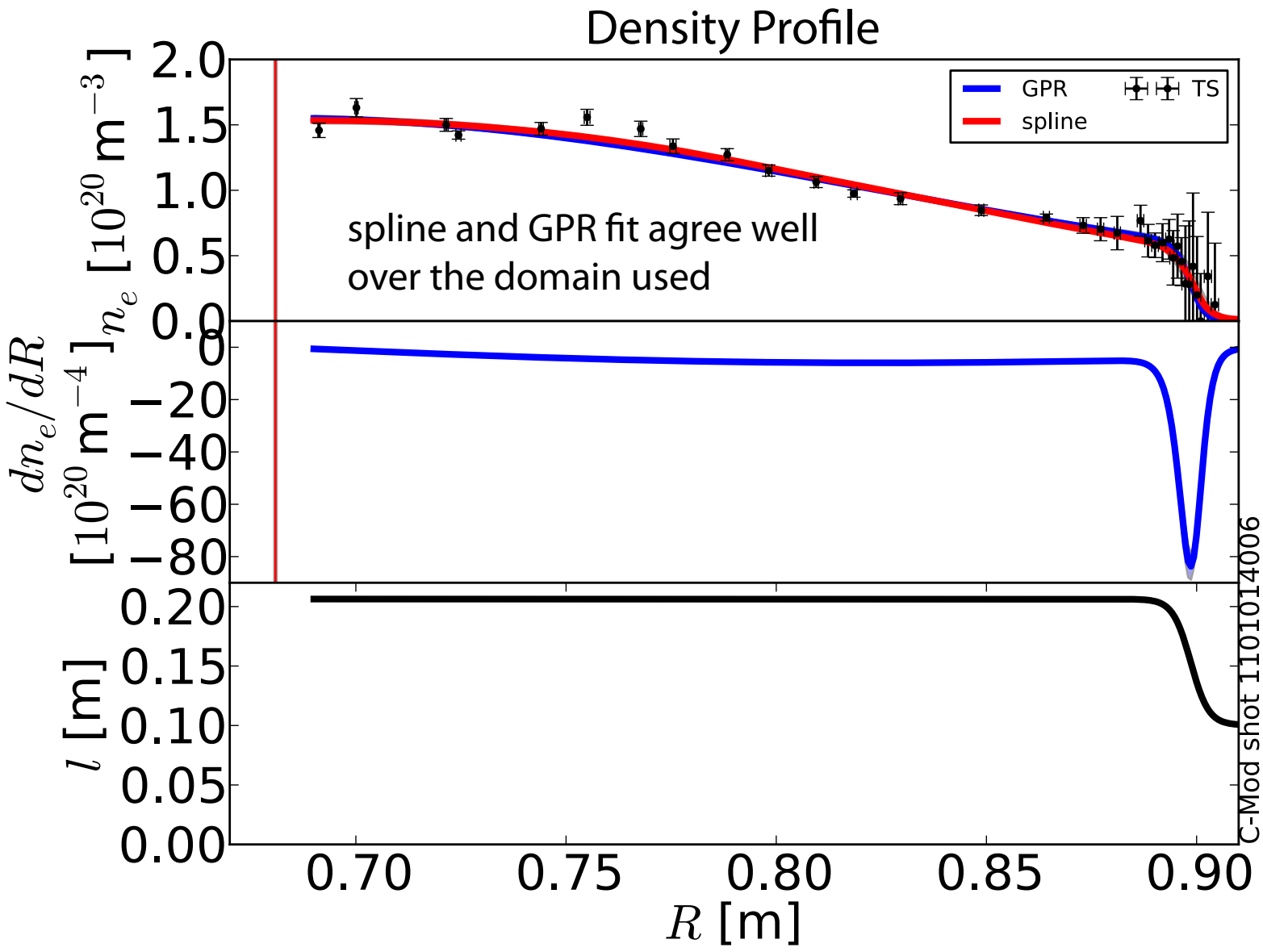


Figure originally from [6].

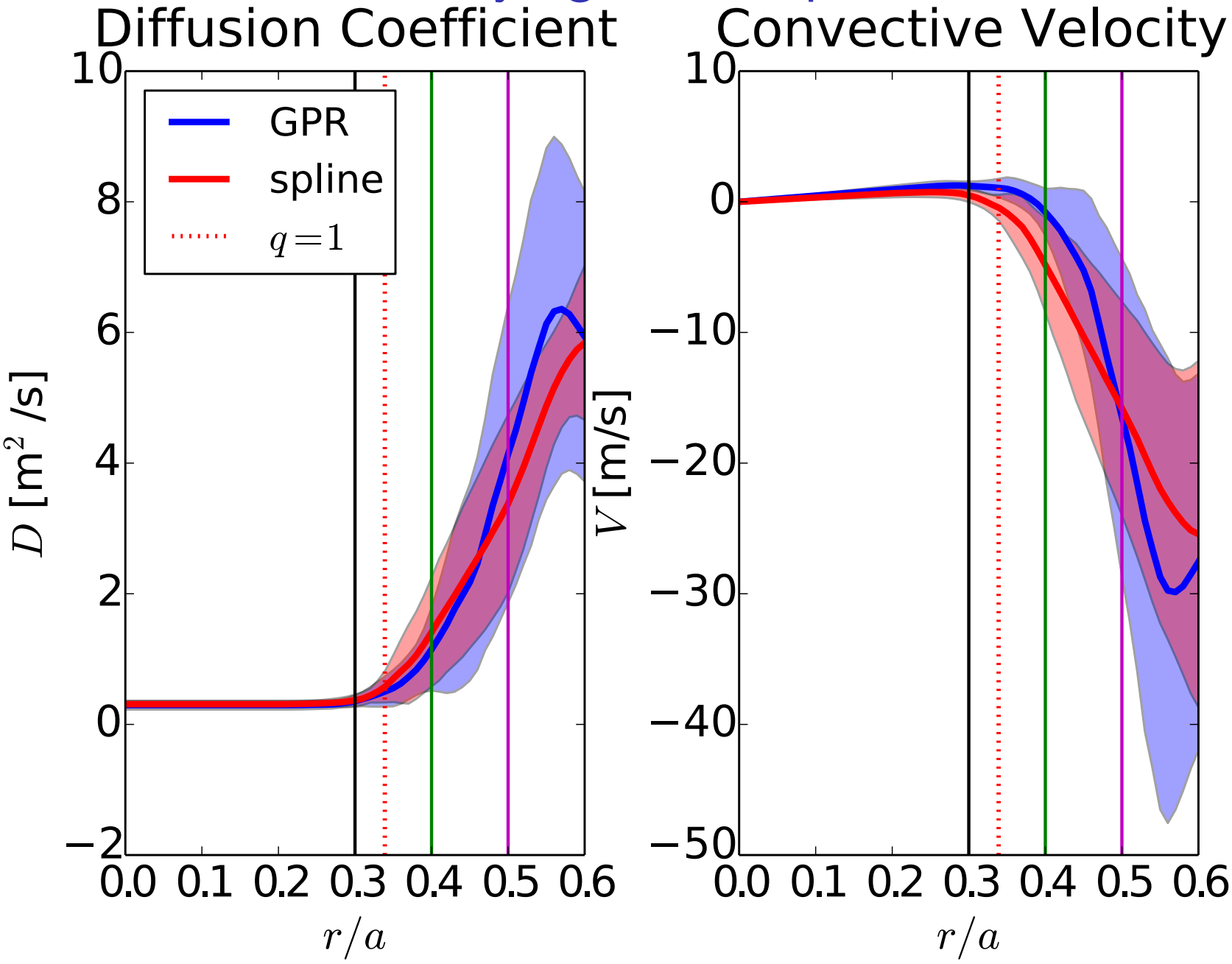
- Given a guess for the D, V profile, STRAHL produces a prediction of the evolution of the impurity density profile.
- This is converted to the Ca^{18+} emissivity profile, which is line-integrated and compared to the brightness measured with an x-ray imaging crystal spectrometer [8].
- The D and V profiles are iterated upon to find the best fit.

Naive Monte Carlo sampling with GPR has been applied to Ca transport in a C-Mod L-mode

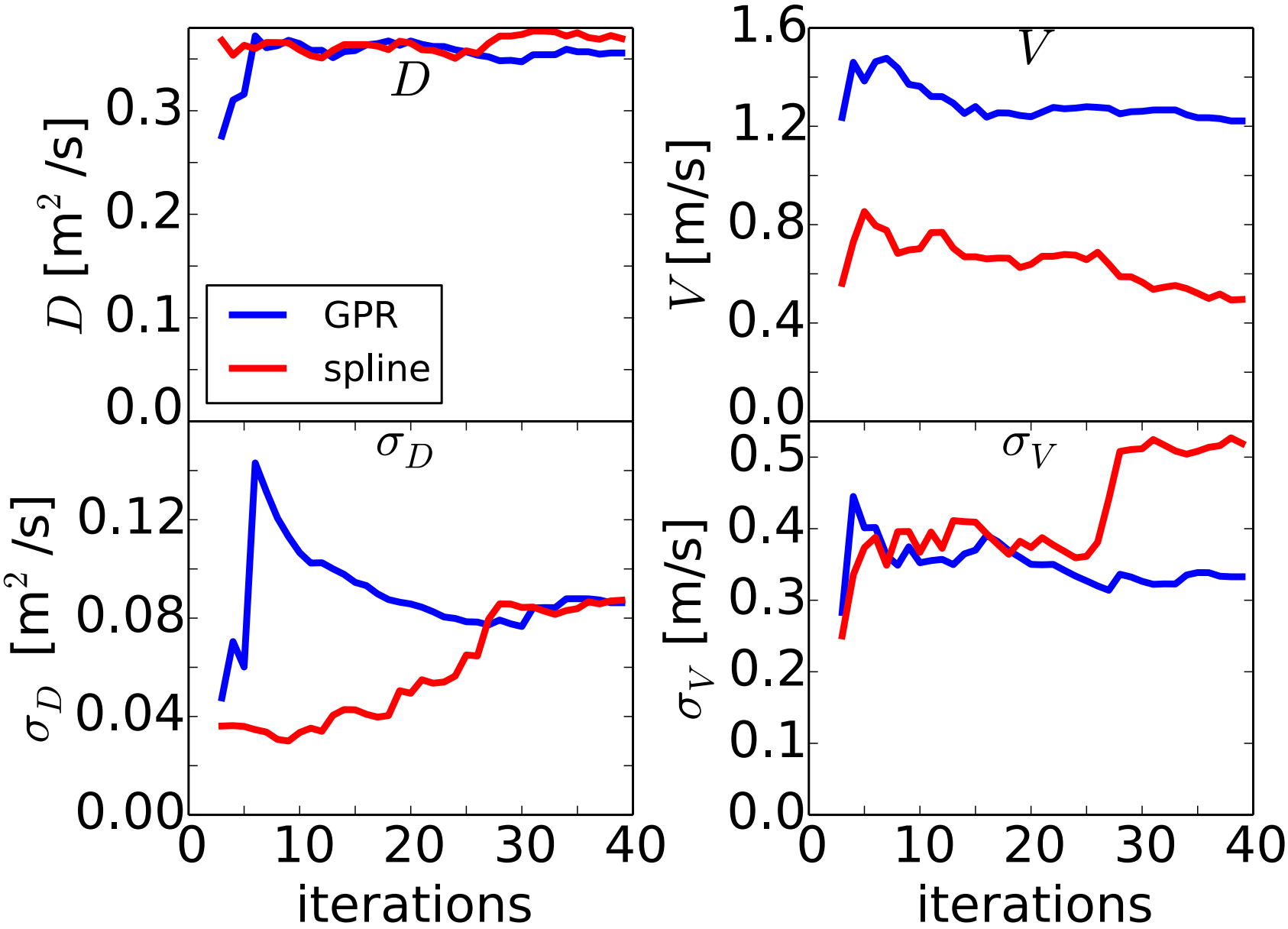



Density Profile
GPR
spline
TS
spline and GPR fit agree well
over the domain used
n_e [10^{20} m^{-3}]
dn_e/dR [10^{20} m^{-4}]
l [m]
R [m]
C-Mod shot 1101014006

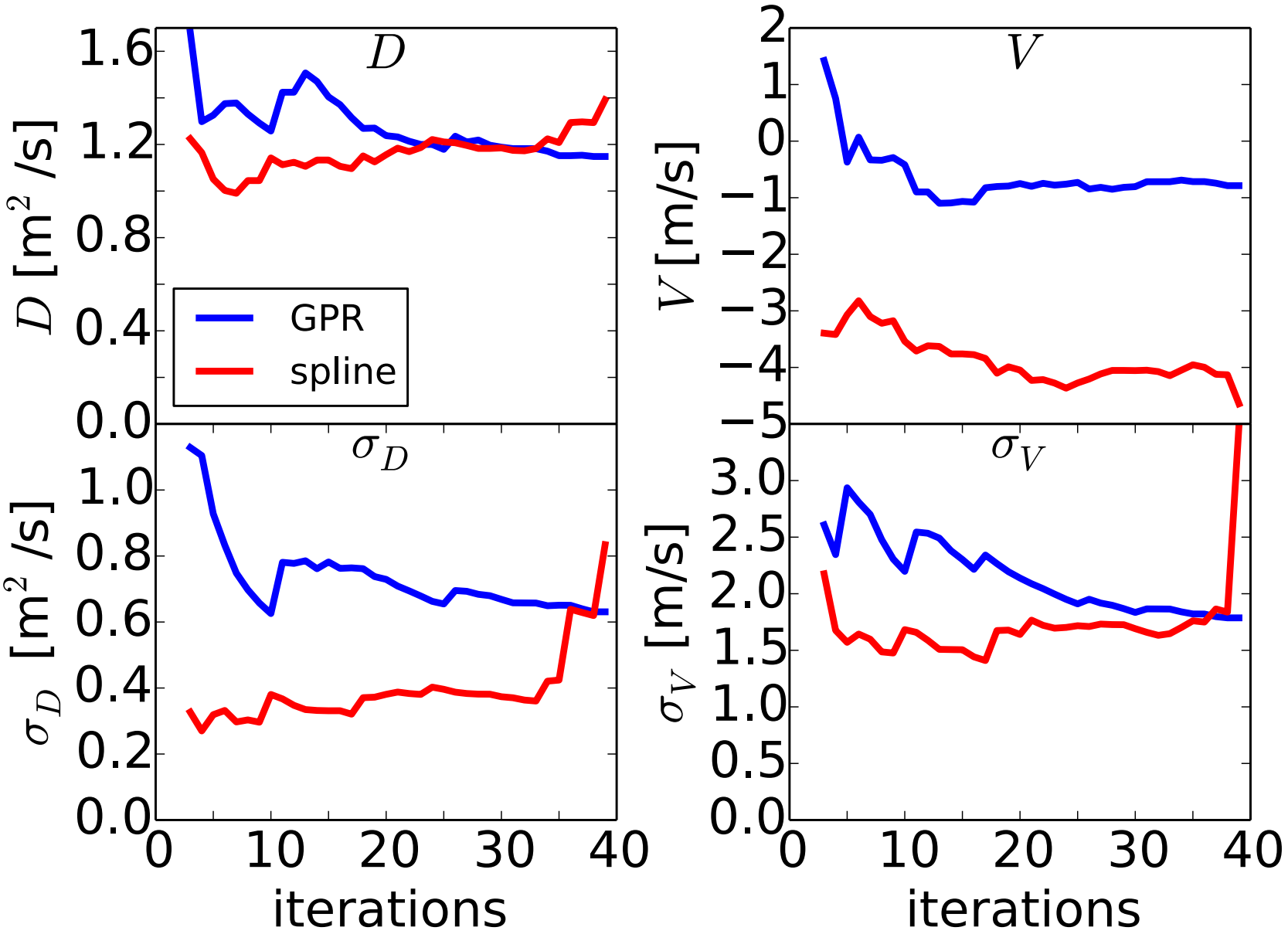
Initial GPR results mostly agree with previous results



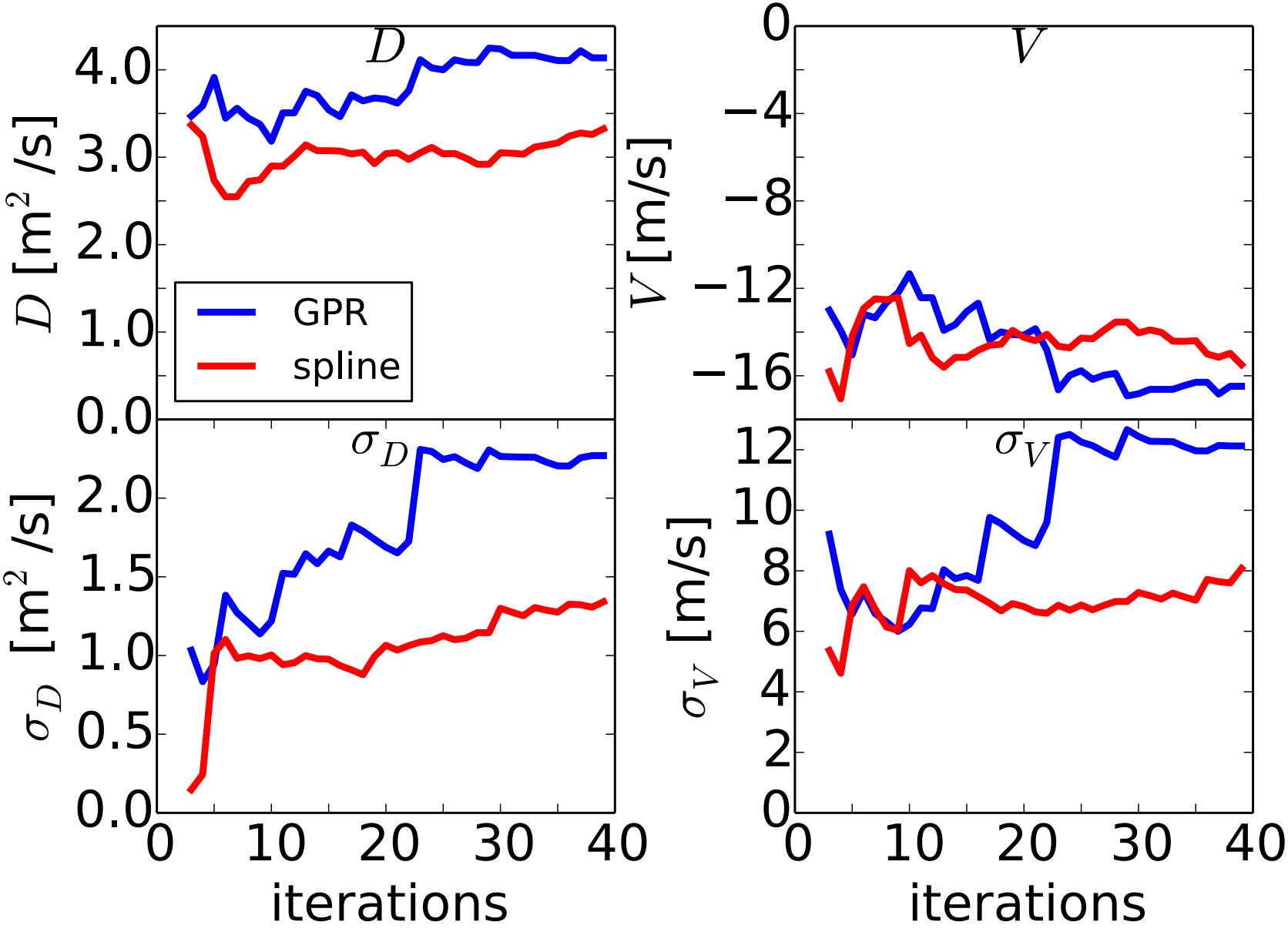
Convergence at $r/a = 0.3$



Convergence at $r/a = 0.4$



Convergence at $r/a = 0.5$



Discussion of initial STRAHL results

- The results are not trusted outside of the range shown ($0 < r/a < 0.6$).
- Within this domain, the spline-based approach and the GPR-based approach produce similar values, but GPR tends to predict a larger error.
- Neither approach included errors in R_{mid}, which will be considered in a future study.
- The previous spline-based calculation might not be fully converged.
- GPR seems to show a more stable convergence across the domain.
- Large jumps in the spline-based values between iterations could indicate that robust estimators/other outlier mitigation needs to be used.

GPR shows promise as a tool to make uncertainty estimation more rigorous, more automated

- GPR is a nonparametric Bayesian regression technique.
- Naive Monte Carlo sampling has been applied to GPR fits of the n_e, T_e profiles input into the STRAHL code to determine the uncertainties in the D, V profiles.
- The initial results seem to mostly agree with the previous uncertainty estimates using spline fits.
- GPR seems to show better convergence than the use of spline-based samples.

Future work

- Account for additional uncertainties: R_{mid} mapping, uncertainty in hyperparameters.
- Better characterize convergence.
- Assess advanced sampling methods (LHS, QMC) to improve convergence speed.
- Apply to other codes and other plasma conditions.

References

[1] TRANSP, http://w3.pppl.gov/transp/.

[2] R. Dux, STRAHL User Manual, Technical Report 10/30, IPP, 2006.

[3] C. Lemieux, *Monte Carlo and Quasi-Monte Carlo Sampling*, Springer, 2009.

[4] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2006.

[5] M. N. Gibbs, *Bayesian Gaussian Processes for Regression and Classification*, PhD thesis, University of Cambridge, 1997.

[6] N. T. Howard et al., Nucl. Fusion **52** (2012).

[7] N. T. Howard et al., Rev. Sci. Instrum. **82** (2011).

[8] A. Ince-Cushman et al., Rev. Sci. Instrum. **79** (2008).