Improved profile fitting and uncertainty quantification using Gaussian process regression

M.A. Chilenski, M. Greenwald, Y. Marzouk,[a]
N.T. Howard, A.E. White

MIT PSFC/Alcator C-Mod
[a]MIT Uncertainty Quantification Group

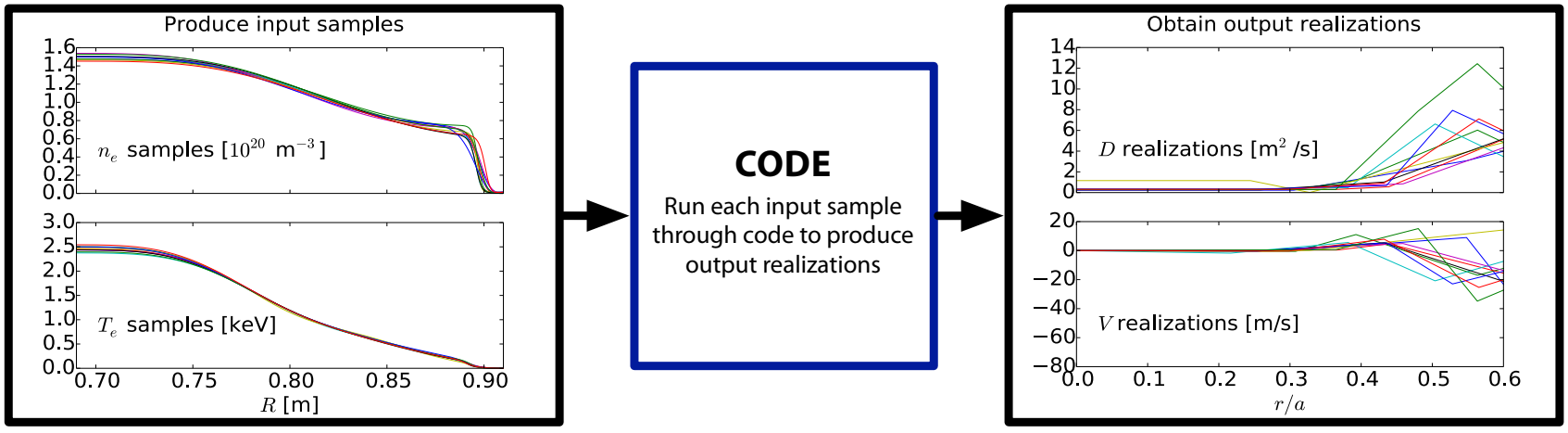
TTF, San Antonio, TX
April 24, 2014
Session II, Poster 28

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

Having better uncertainty estimates on input profiles and code outputs improves the validation process

- 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: Better profile fitting/sampling for improved code comparisons



- To quantify the uncertainty in the outputs of a code it is necessary to produce perturbed samples of profile inputs and run them through the code.
- This has traditionally been done with splines: see slide 6.
- This poster presents a better way of fitting profiles with built-in support for computing the uncertainty in the fit *and* its gradients as well as a straightforward way of drawing input samples.

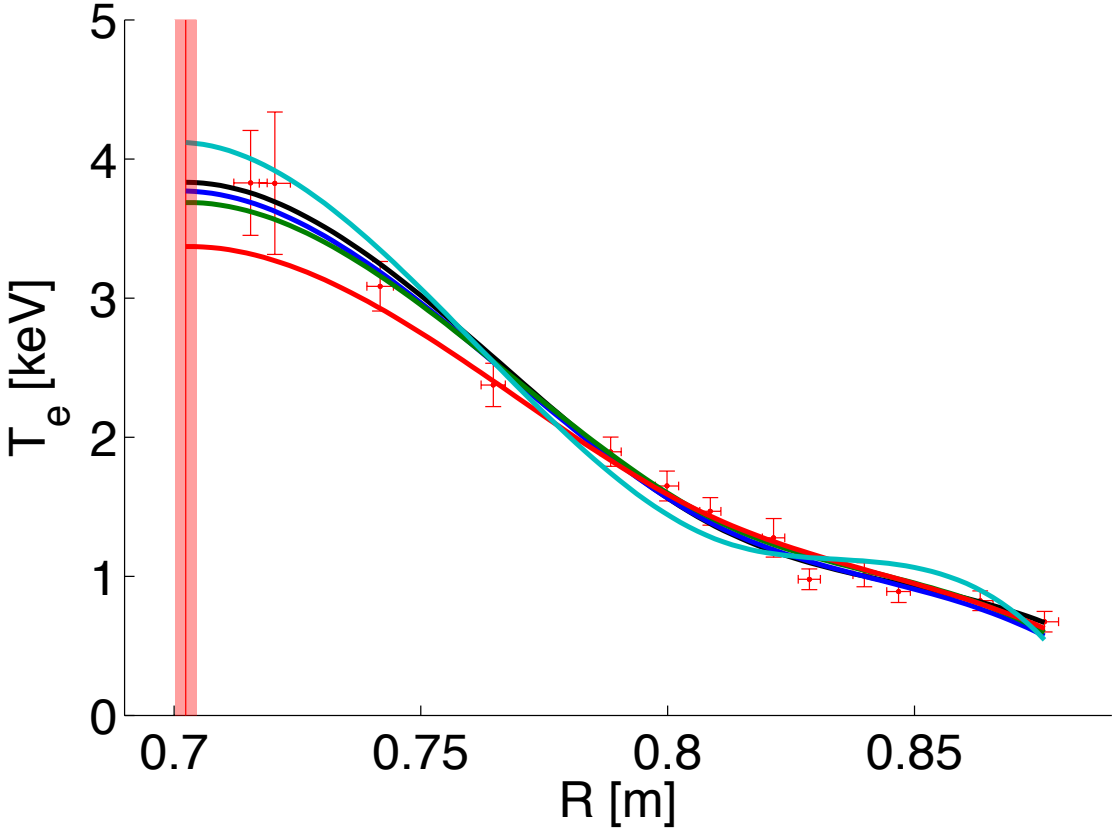
Overview: What Gaussian process regression (GPR) does

- GPR fits a smooth curve to a set of noisy data points – such as measurements of the T_e profile from Thomson scattering.
- Unlike other techniques (such as splines), GPR does this in a fundamentally probabilistic framework.
  - GPR provides built-in estimates of the uncertainty in the fit: no extra steps are required to put an uncertainty envelope on your fit.
  - It is very easy and efficient to draw random samples to be used in a Monte Carlo uncertainty quantification of a code that takes profiles as inputs.
- GPR lets you easily incorporate gradient information into the fit, both in terms of adding constraints and in terms of getting estimates of the uncertainty in the gradient of the fit.

What advanced uncertainty quantification techniques, Gaussian process regression 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 ($y(R, Z, \phi, t)$) is painful.

Notes on the plots

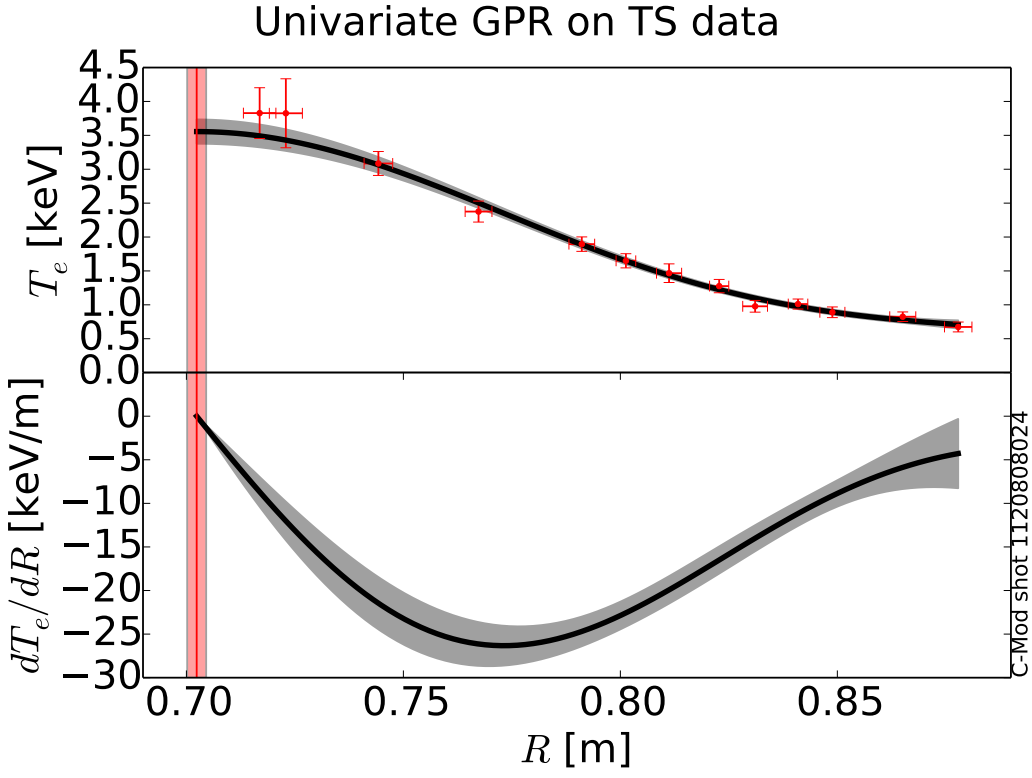
- Vertical red line indicates the magnetic axis.
- Data points are the average in a single channel over the flattop
- Vertical error bars are $\pm 1\sigma$ of T_e within a channel. Horizontal error bars are $\pm 1\sigma$ of mapped R_{mid} *but are not included in the analysis*.

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

GPR is a Bayesian non-parametric regression technique [3]

Bayesian: A probability distribution called the prior encodes assumptions about the properties of 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
observations:	The data to be fit: $\boldsymbol{y}(\mathrm{X})$, $\boldsymbol{y} \in \mathbb{R}^n$, $\mathrm{X} \in \mathbb{R}^{D \times n}$
predictions:	The values of the smooth fit: $\boldsymbol{y}_*(\mathrm{X}_*)$, $\boldsymbol{y}_* \in \mathbb{R}^{n_*}$, $\mathrm{X}_* \in \mathbb{R}^{D \times n_*}$
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:

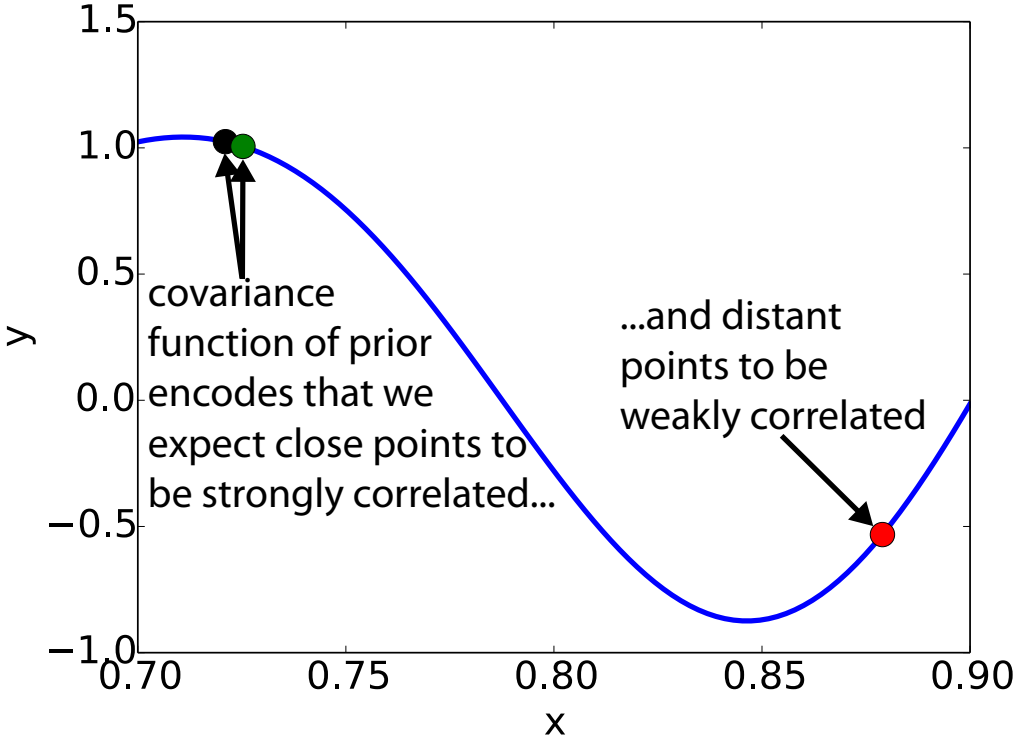
- n observations of quantity y are taken at n (D-dimensional) locations $\boldsymbol{x}$ and combined into vector $\boldsymbol{y}$ and matrix X.
- Predictions y_* are then made at n_* locations $\boldsymbol{x}_*$ and combined into vector $\boldsymbol{y}_*$ and matrix X_*.
- The matrix K consists of the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

A Gaussian process is a *distribution over functions*

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

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

- *This is an n-dimensional multivariate normal, where* $\boldsymbol{y} \in \mathbb{R}^n$.
- $m(\boldsymbol{x})$ is the mean function
- $k(\boldsymbol{x},\, \boldsymbol{x}')$ is the covariance function, which determines the spatial correlation between points:



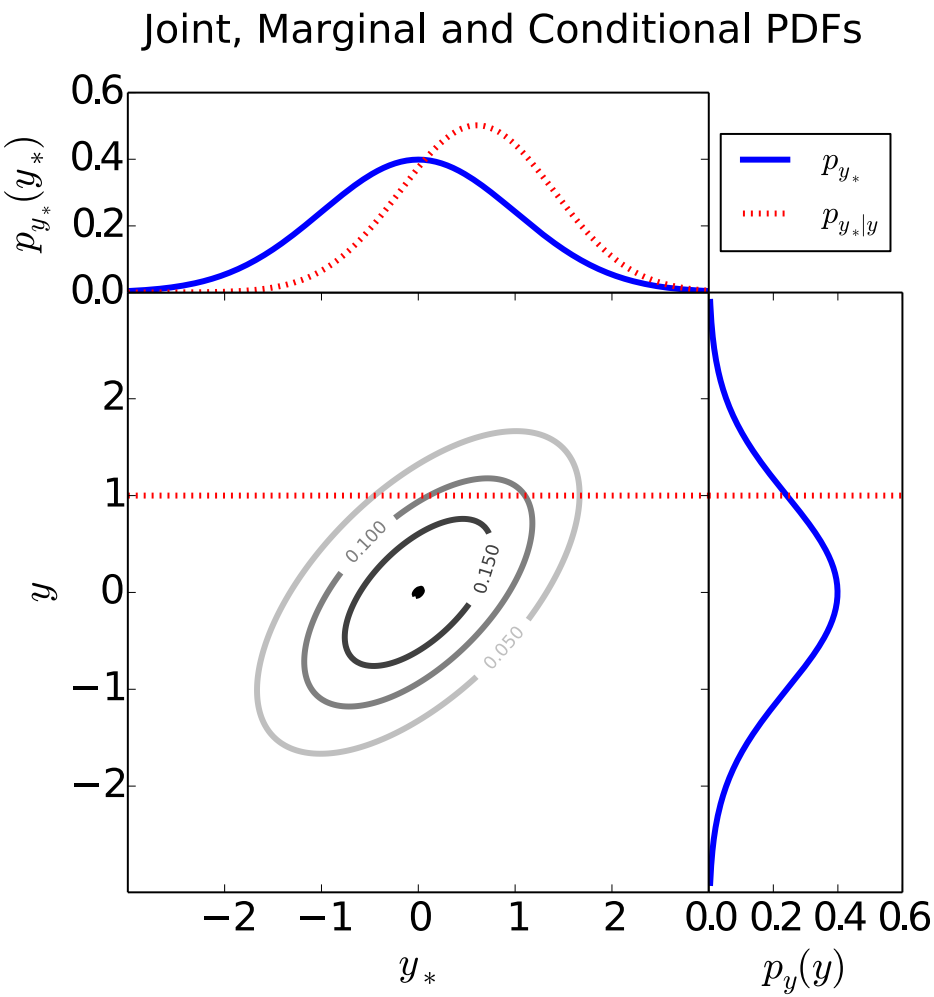
Common choice: squared exponential (SE)

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

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

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

Simple illustration: one observation and one prediction



- Start with the *prior* probability density between observations $\boldsymbol{y}(\mathrm{X})$ and predictions $\boldsymbol{y}_*(\mathrm{X}_*)$: $p(\boldsymbol{y}_*, \boldsymbol{y}) = \mathcal{N}(\mathbf{0}, k([\mathrm{X}_*, \mathrm{X}], [\mathrm{X}_*, \mathrm{X}]))$
- Condition on observations to get *posterior* for the predictions $\boldsymbol{y}_*$: $p(\boldsymbol{y}_*|\boldsymbol{y}) = p(\boldsymbol{y}_*, \boldsymbol{y})/p(\boldsymbol{y})$
- The posterior is also a multivariate normal, but with a more complicated mean and covariance function.
- The posterior mean is the fit, the posterior variance is the uncertainty in the fit.

Selection of hyperparameters is accomplished with standard statistical techniques

Several levels of sophistication have been explored to select the hyperparameters $\boldsymbol{\theta} = [\sigma_f,\ \ell,\ \ldots]$ (as in the k's on slides 9 and 15):

Simplest approach: maximum likelihood (ML) Maximize the likelihood of the observations $\boldsymbol{y}$ given hyperparameters $\boldsymbol{\theta}$: $\hat{\boldsymbol{\theta}}^{ML} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{y}|\boldsymbol{\theta})$

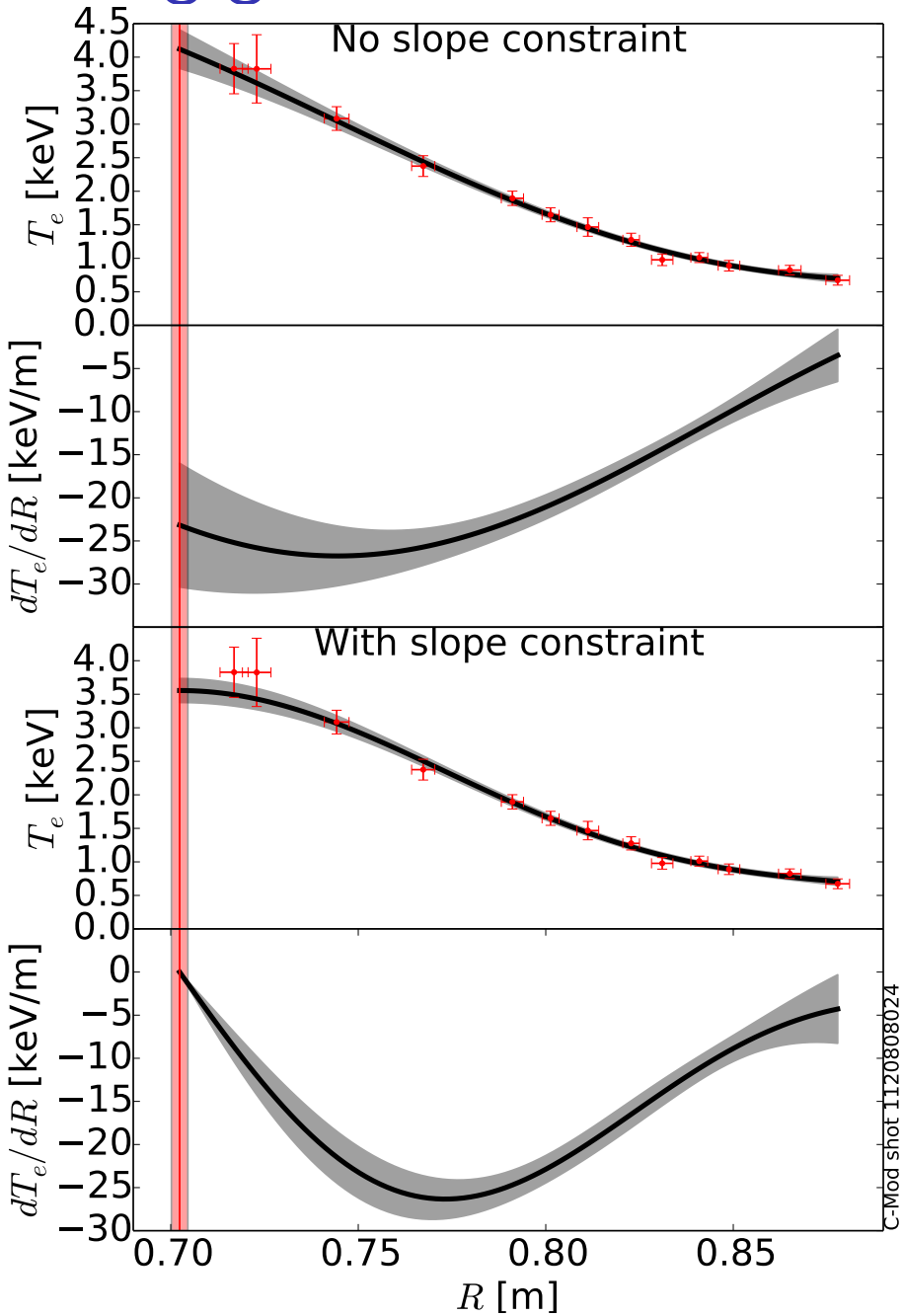
Next level: maximum a posteriori (MAP) Maximize the probability of the hyperparameters $\boldsymbol{\theta}$ given the observations $\boldsymbol{y}$: $\hat{\boldsymbol{\theta}}^{MAP} = \arg\max_{\boldsymbol{\theta}} p(\boldsymbol{\theta}|\boldsymbol{y})$

Most complete: marginalization *Marginalize* the hyperparameters $\boldsymbol{\theta}$ out of the posterior for the predictions $\boldsymbol{y}_*$ given the observations $\boldsymbol{y}$:

$$p(\boldsymbol{y}_*|\boldsymbol{y}) = \int p(\boldsymbol{y}_*|\boldsymbol{\theta}, \boldsymbol{y}) p(\boldsymbol{\theta}|\boldsymbol{y})\, \mathrm{d}\boldsymbol{\theta}$$

This integral is carried out efficiently using Markov chain Monte Carlo (MCMC) techniques [4, 5].

Getting gradients *and their uncertainties* is straightforward



The gradient of a Gaussian process is a Gaussian process:

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

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

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

Drawing random samples of T_e, dT_e/dR is straightforward

Just sampling a multivariate normal to draw a random sample $\widetilde{\boldsymbol{y}}_*$ from $p(\boldsymbol{y}_*|\boldsymbol{y}, \boldsymbol{\theta})$:

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

$$\boldsymbol{m} = \mathsf{K}(\mathsf{X}_*, \mathsf{X})[\mathsf{K}(\mathsf{X}, \mathsf{X}) + \Sigma_\mathsf{n}]^{-1}\boldsymbol{y} \quad \text{(the predictive mean)}$$

$$\mathsf{K}_\mathsf{p} = \mathsf{L}\mathsf{L}^\mathsf{T} \quad \text{(predictive covariance matrix)}$$

$$\boldsymbol{u} \sim \mathcal{N}(\mathbf{0}, \mathsf{I}) \quad (n_* \text{ independent standard normal variables})$$

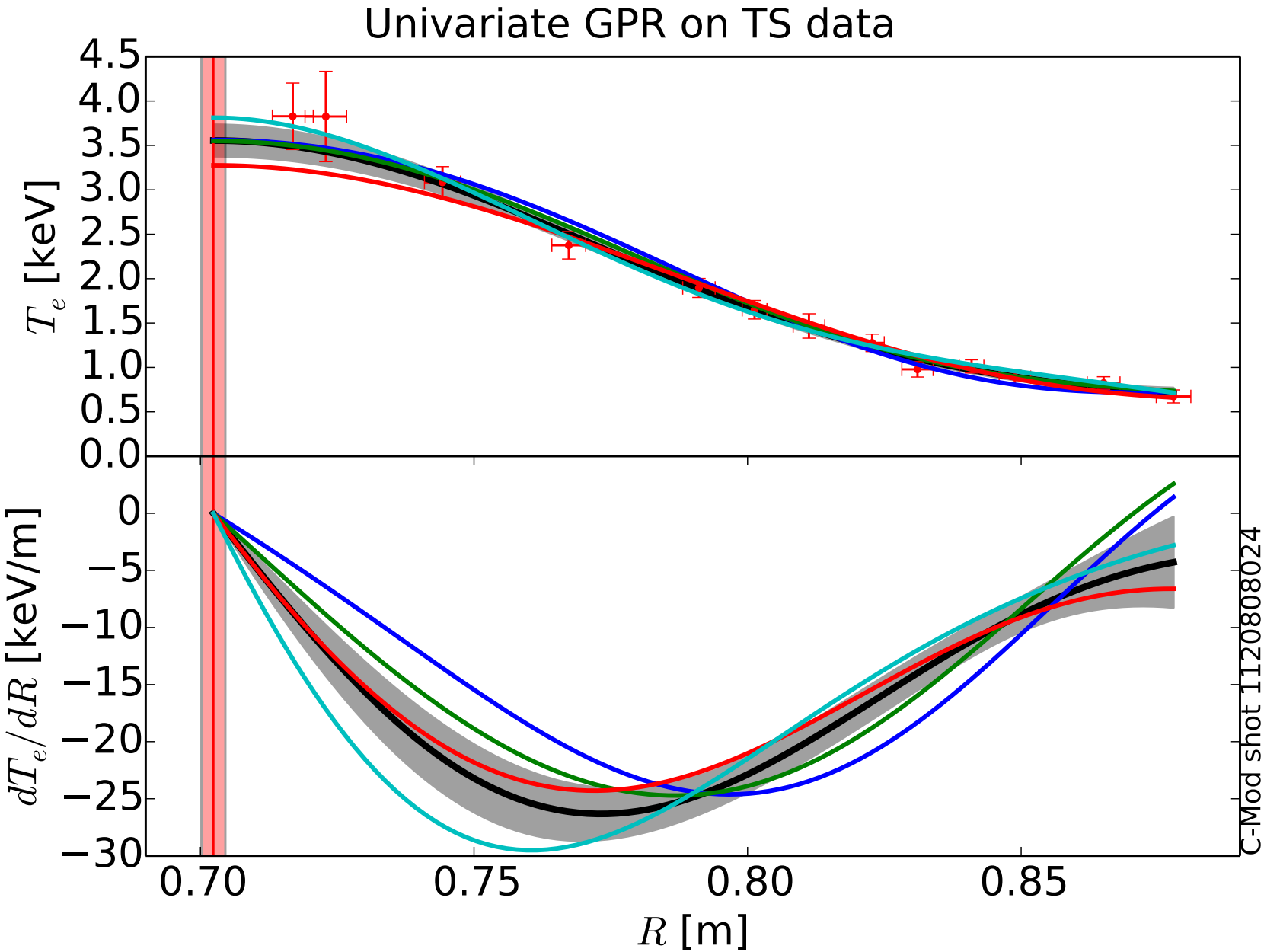
More powerful way of writing the matrix square root using eigendecomposition of predictive covariance matrix:

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

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

Can truncate eigendecomposition to reduce dimensionality.

Example of simultaneous random samples of T_e, dT_e/dR

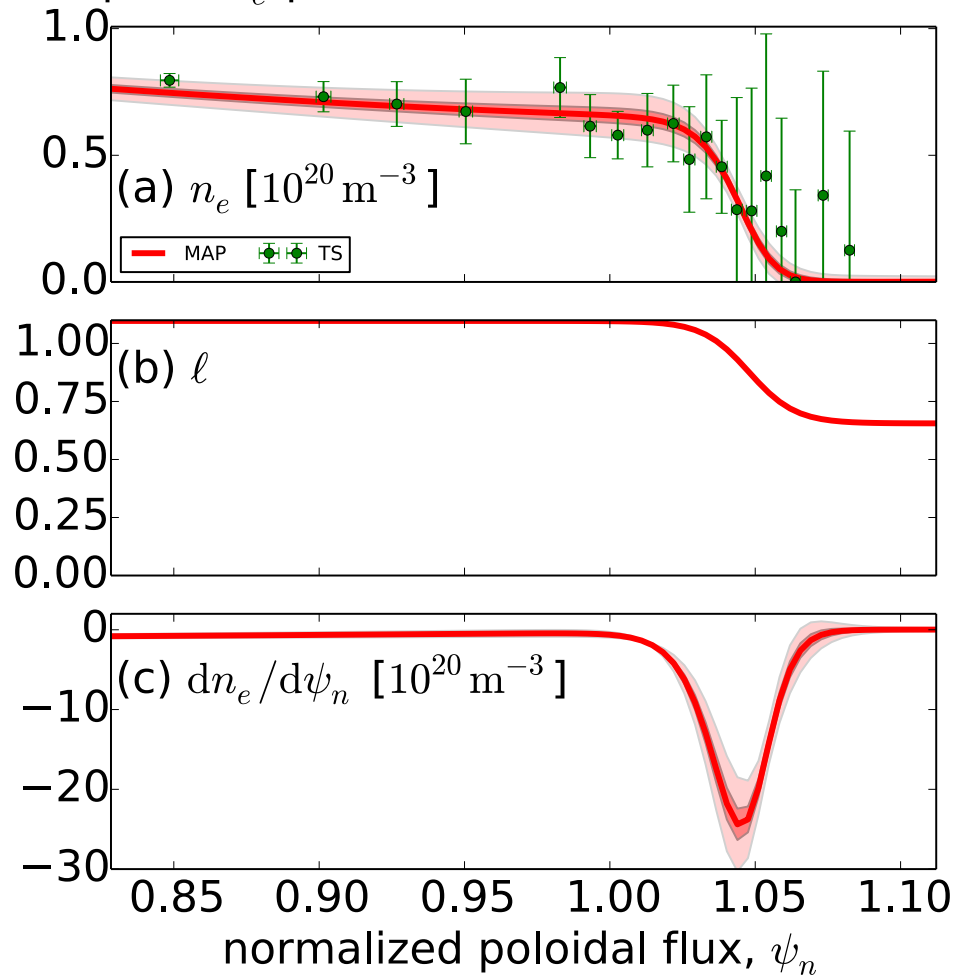


Fitting pedestal requires non-stationary covariance function

Gibbs kernel [6]: ℓ is an arbitrary function of $\boldsymbol{x}$

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

Complete n_e profile: Gibbs covariance function



- Length scale used here:

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

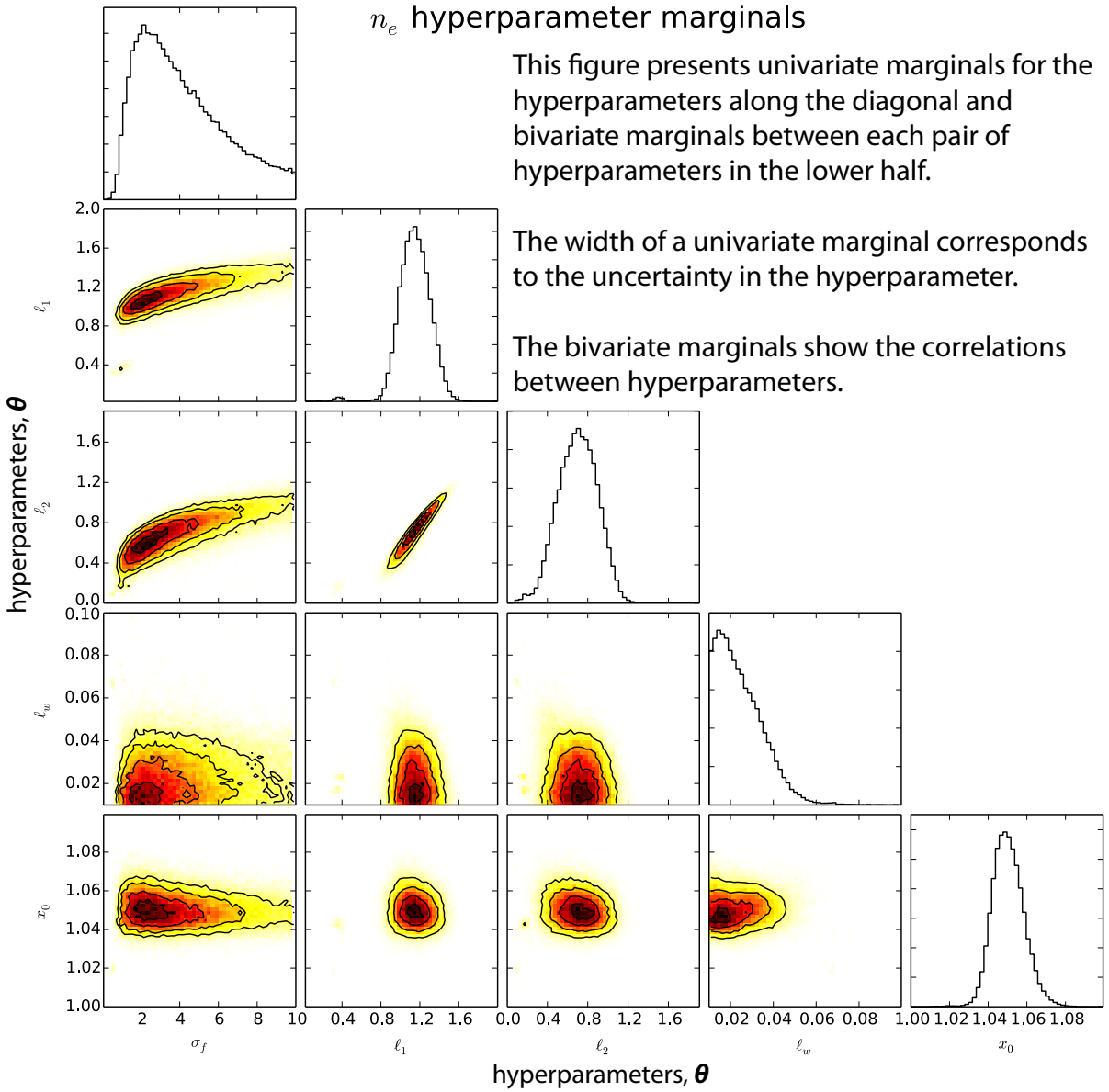
- Either pick σ_f, ℓ_1, ℓ_2, ℓ_w and x_0 through MAP (shown here) or by marginalizing with MCMC.
- n_e, $\mathrm{d}n_e/\mathrm{d}\psi_n$ were set to ~ 0 at $\psi_n = 1.1$, the midplane location of the limiter.

Marginalizing hyperparameters using MCMC: the goal

- MAP and ML estimators are *point estimates* – they do not account for the fact that there is uncertainty in the hyperparameters $\boldsymbol{\theta}$.
- The most complete accounting of uncertainty is obtained by marginalizing over the hyperparameters (see slide 11).
- Markov chain Monte Carlo (MCMC) [4, 5] is used to produce random draws $\widetilde{\boldsymbol{\theta}}$ of the hyperparameters distributed according to the posterior for the hyperparameters $\boldsymbol{\theta}$ given the observations $\boldsymbol{y}$: $\widetilde{\boldsymbol{\theta}} \sim p(\boldsymbol{\theta}|\boldsymbol{y})$.
- These draws are then used with the scheme to draw random samples given on slide 13 to produce random draws $\widetilde{\boldsymbol{y}}_*$ of the smoothed profile: $\widetilde{\boldsymbol{y}}_* \sim p(\boldsymbol{y}_*|\boldsymbol{y}, \boldsymbol{\theta} = \widetilde{\boldsymbol{\theta}})$.
- These draws are then either averaged to give the mean smoothed profile, or are used as input samples to a code.

Marginalizing hyperparameters using MCMC: the result

Visualization of 5-d posterior distribution for hyperparameters given observations, $p(\boldsymbol{\theta}|\boldsymbol{y})$:



- Univariate marginals are peaked: the observations provide sufficient information to dominate over the uniform hyperpriors.
- Marginals are unimodal: permits unambiguous interpretation of the fit.
- Profiles are given on slides 21 and 22

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

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 [7, 8].
- Larger injections are used to induce cold pulses to investigate non-local thermal transport.
- Injection of molybdenum used to probe poloidal asymmetries and their effects.

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

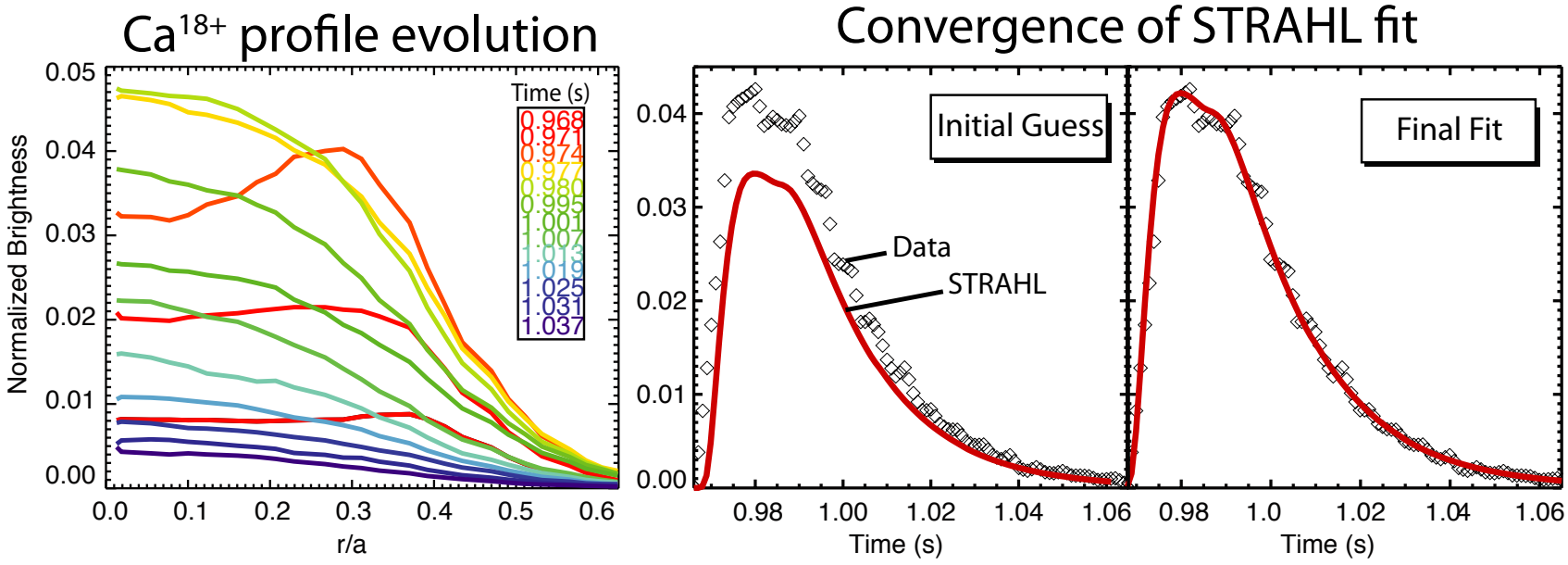
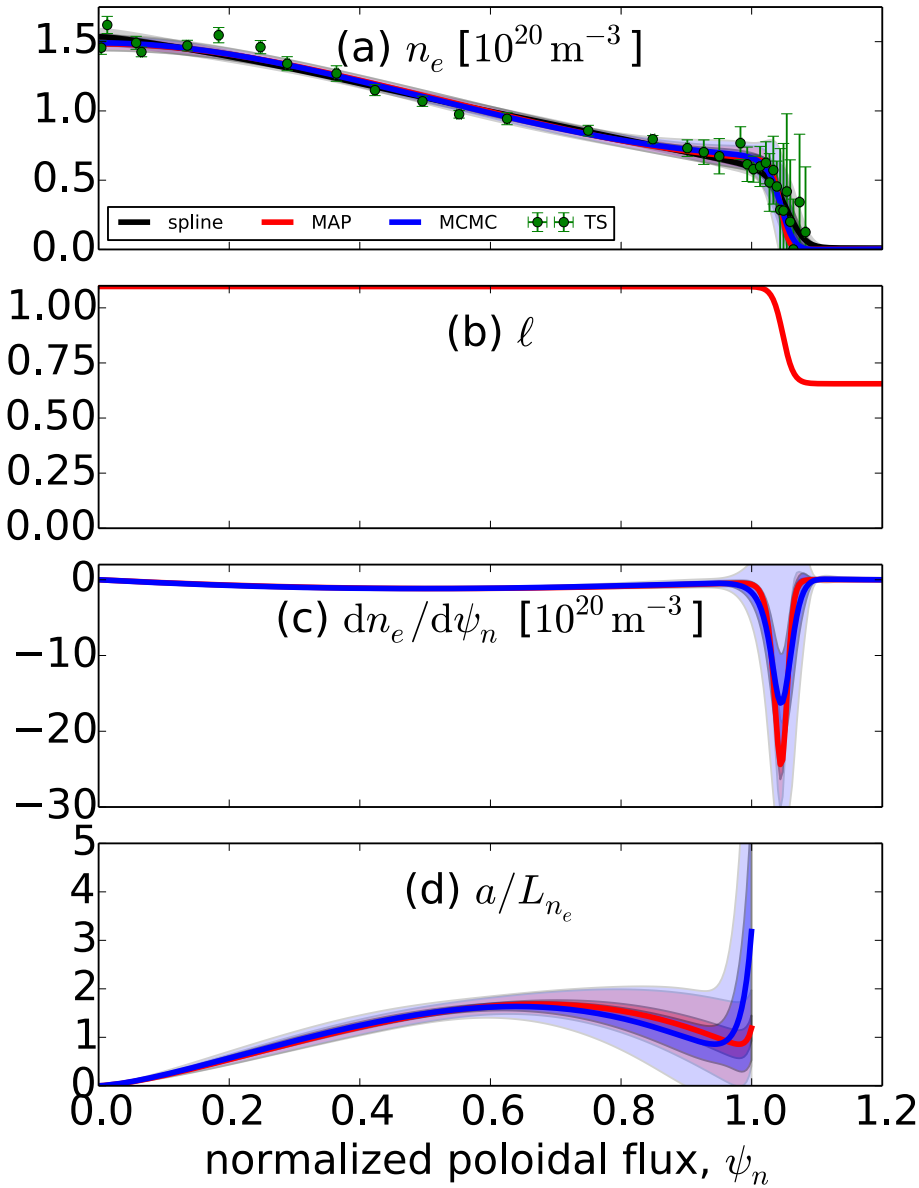


Figure originally from [7].

- Model the impurity flux as $\Gamma_Z = -D\nabla n_Z + V n_Z$
- Given the n_e, T_e profiles and a guess for the D, V profiles, 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 [9].
- The D and V profiles are iterated upon to find the best fit.

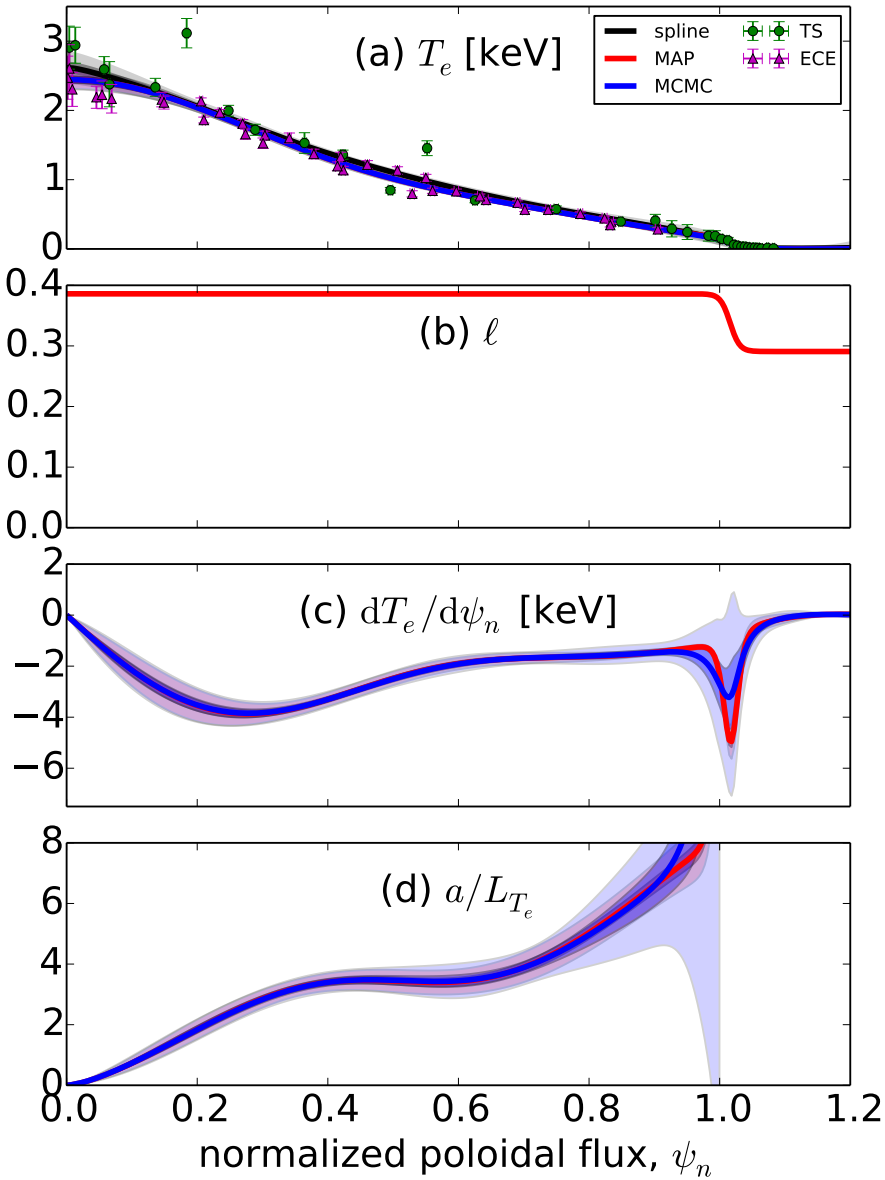
GPR sampling has been applied to Ca transport in C-Mod

Complete n_e profile: Gibbs covariance function



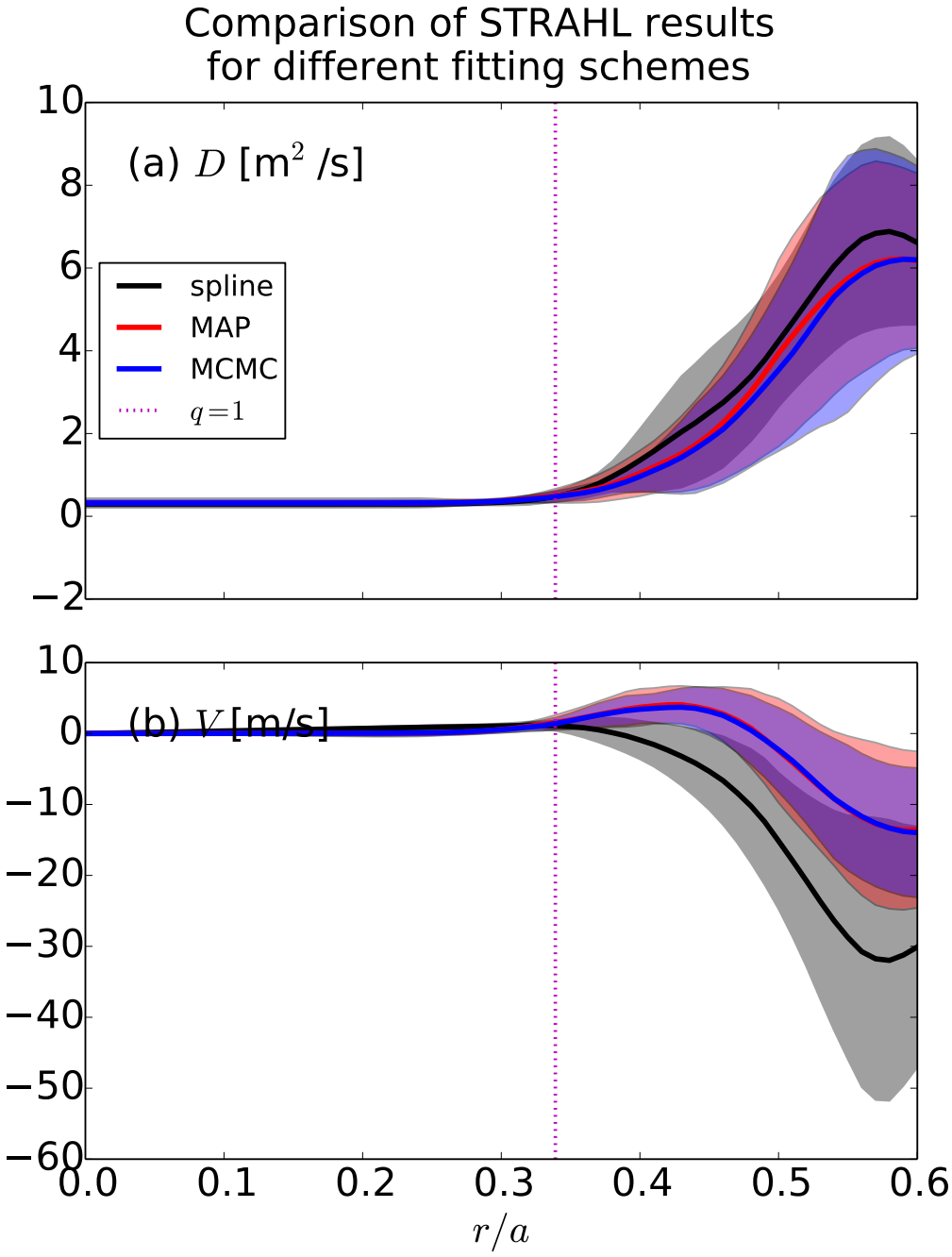
- 800 kA, 5.4 T L-mode with 1 MW ICRF.
- Light band is $\pm 3\sigma$, dark band is $\pm 1\sigma$.
- Shape of n_e profile is similar between the spline fit, the MAP estimate and the fully marginalized estimate.
- The uncertainty in $\mathrm{d}n_e/\mathrm{d}\psi_n$ and a/L_{n_e} increases noticeably with marginalization – **marginalization is essential to properly capture the uncertainty in the gradient**.

Complete T_e profile: Gibbs covariance function



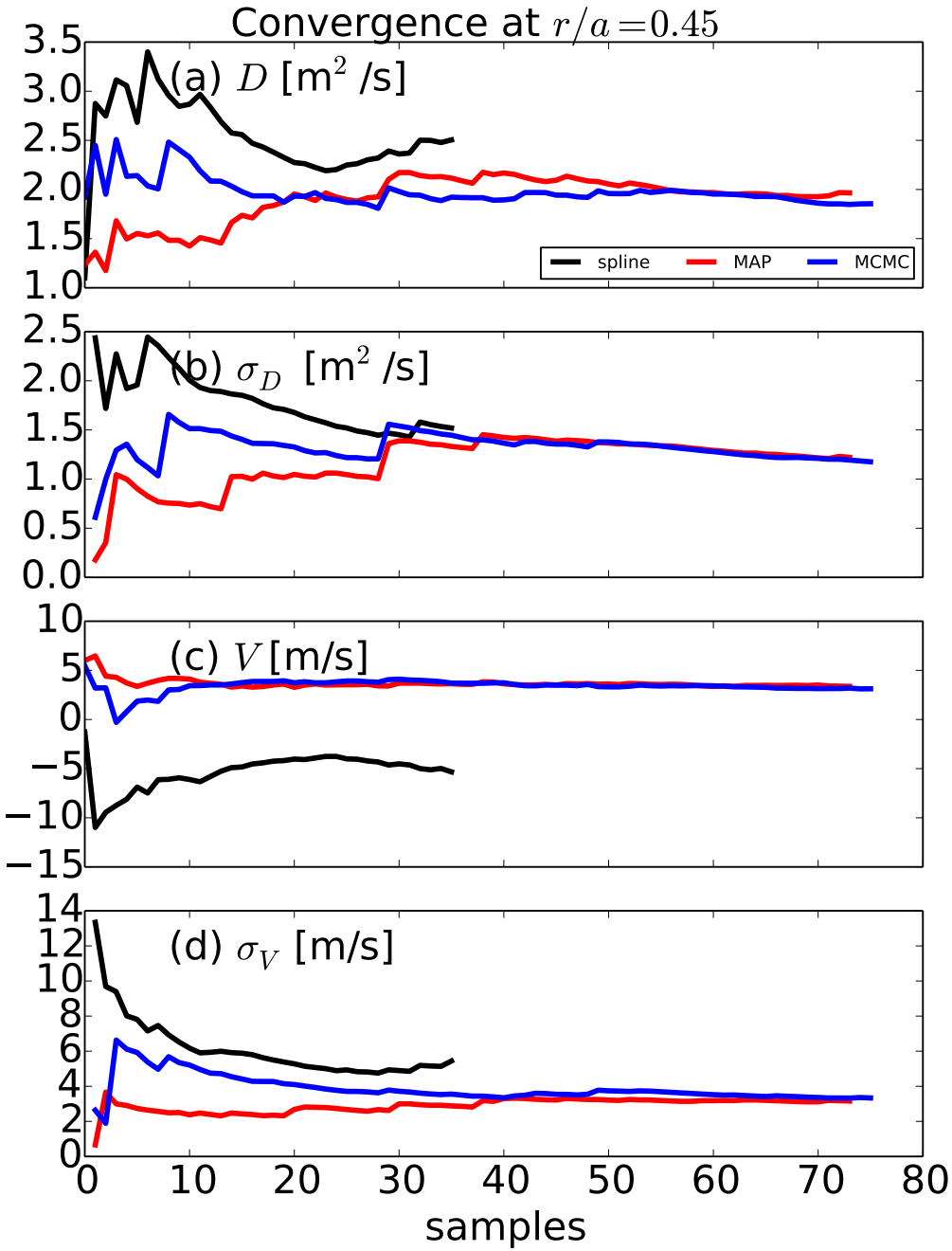
- Two Thomson scattering (TS) and three electron cyclotron emission (ECE) diagnostics were combined to reduce uncertainty.
- The fit is robust against the apparent outliers near $\psi_n = 0.2$ and 0.5.
- Marginalization over the hyperparameters again showed a substantial effect on the uncertainty in the gradient.

Initial GPR results mostly agree with previous results



- Only the $\pm 1\sigma$ uncertainty envelope is shown.
- The results are not trusted outside of the range shown ($0 < r/a < 0.6$).
- Within this domain, the spline-based approach and the Gaussian process-based approach produce similar values.

GPR-based approach shows better convergence



- The previous spline-based calculation might not be fully converged.
- Extra MAP, MCMC samples were run to ensure convergence.
- Adding samples is trivial with GPR, would have required nontrivial manual effort with previous workflow.
- MAP estimate appears to converge faster than MCMC at some radii, but also tends to have more jumps.

Gaussian process regression shows promise as a tool to make uncertainty estimation more rigorous, more automated

- GPR is a Bayesian nonparametric 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

- Assess advanced sampling methods (LHS, QMC) to improve convergence speed.
- Apply to other codes and other plasma conditions.
- **A paper on this work is in preparation and will be submitted for publication soon.**

References

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

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

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

[4] J. Goodman and J. Weare, Comm. App. Math. and Comp. Sci. **5**, 65 (2010).

[5] D. Foreman-Mackey, D. W. Hogg, D. Lang, and J. Goodman, Publ. Astron. Soc. Pac. **125**, 306 (2013).

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

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

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

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