Normalizing flows for likelihood-free inference with fusion simulations

R. Michael Churchill, *PPPL*

Chirag Furia, *Rutgers University*

PPPL PRINCETON PLASMA PHYSICS LABORATORY

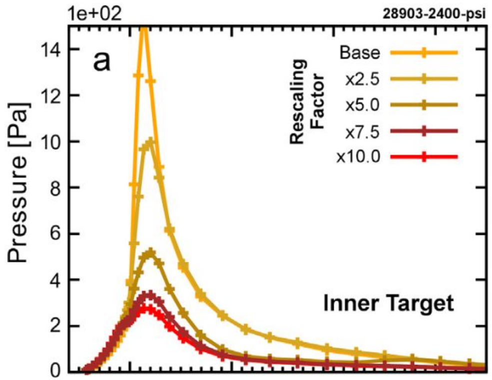
PRINCETON UNIVERSITY

Motivation: Fusion simulations often require ad-hoc inputs

- Ad-hoc simulation inputs (e.g. anomalous transport coefficients in scrape-off layer fluid codes like SOLPS) are valuable for engineering design and experimental comparison

- Traditionally ad-hoc inputs are hand tuned to match simulation with experiment

But how do we know their uniqueness or range of validity?



- Ex: Fluid codes like SOLPS often can't match upstream/downstream density and temperature simultaneously, tuning to match involves enhanced anomalous transport in divertor [Reimold, PSI 2017]

Motivation: Utilize simulation for fast statistical inference of physics parameters from experiment

- To apply Bayesian inference for deriving physics parameters from experiment, typically explicitly specify likelihood $p(x|\theta)$
  - Some simulators with randomness can't explicitly define a likelihood
- Traditional methods (e.g. MCMC) for statistical inference are sequential, require ~hours, don't scale to number of experimental points in fusion experiments

Physics parameters
θ
(e.g. D, χ)

Forward model
$p(x|\theta)$

Prediction →

Predicted observables
x
(e.g. n, T)

Inferred physics parameters
θ^*
(e.g. D, χ)

← Inference

Inverse model
$p(\theta|x)$

Actual experimental observables
x^*
(e.g. n, T)

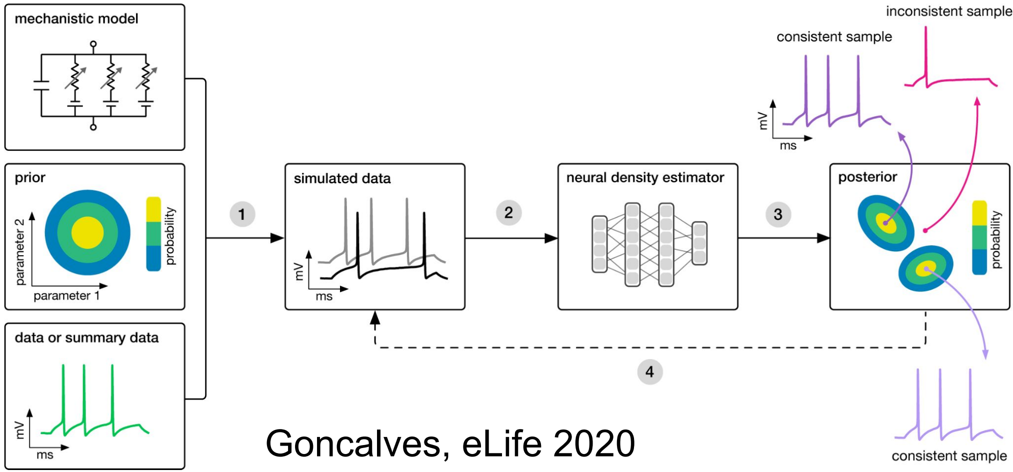
Baye's Thm

$$p(\theta|x) \propto p(x|\theta)p(\theta)$$

How can we utilize simulators without explicit likelihoods, and ensure fast inference?

Simulation-based inference (a.k.a. likelihood-free inference) using neural density estimator

- Procedure to use:
  - Generate sample physics parameters $\theta_i \sim p(\theta)$ from prior
  - Run through simulator ("forward model") to generate experimental observations $x_i \sim \text{Simulator}(\theta_i)$
  - Train neural network with dataset $\{\theta_i, x_i\}$'s to learn conditional density, e.g. the posterior $p(\theta|x)$
  - Can train amortized model for generic experimental targets (x), or sequentially update prior in rounds for a specific instance of (x_0)



Goncalves, eLife 2020

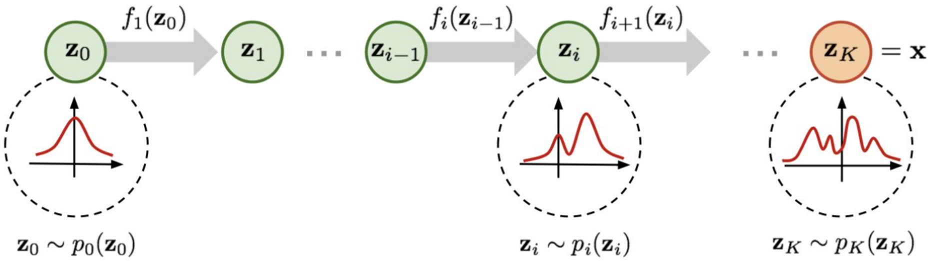
Normalizing Flows

- Normalizing flows are a bijective (invertible) transformation
  - Allows transforming a known distribution $p_Z(z)$ e.g. a Gaussian to a more complicate one $p_X(x)$

$$f : R^n \rightarrow R^n \quad \text{such that} \quad x = f(z) \quad \text{and} \quad z = f^{-1}(x)$$

$$p_X(x) = p_Z\left(f^{-1}(x)\right)\left|\det\left(\frac{\partial f^{-1}(x)}{\partial x}\right)\right| = p_Z(z)\left|\det\left(\frac{\partial f}{\partial z}\right)\right|^{-1}$$

$$\log p_X(x) = \log p_Z(z) - \log\left|\det\left(\frac{\partial f}{\partial z}\right)\right|$$

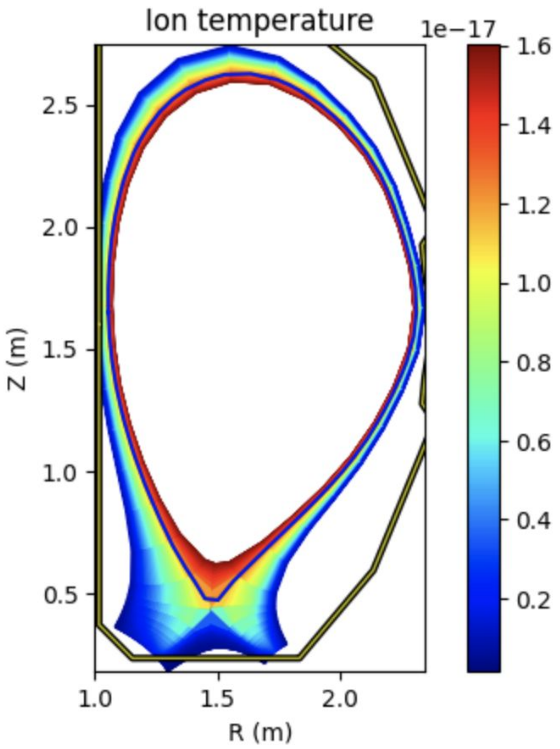


- Chaining bijective transforms also leads to a total bijective transform
- Neural networks are used to learn each bijective transform *f*
- Can be used for conditional density estimation of posterior $p(\theta|x)$

https://lilianweng.github.io/lil-log/2018/10/13/flow-based-deep-generative-models.html

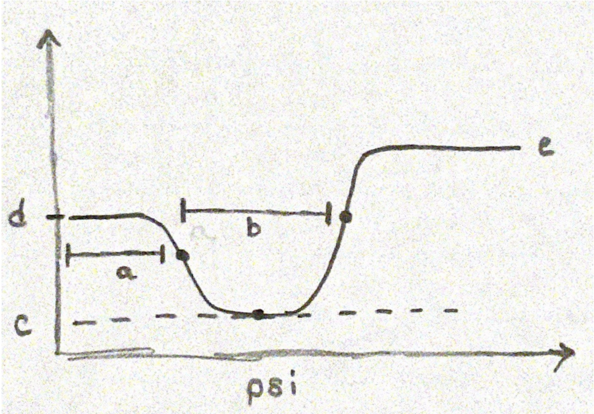
Simulator: UEDGE

- UEDGE is a fluid SOL code similar to SOLPS
  - Most often fluid neutrals used
- Utilizes ad-hoc inputs of anomalous perpendicular transport:
  - *D* particle diffusion, χ_e electron energy diffusion, χ_i ion energy diffusion
- Relatively fast on small grids, ~1 minute using pyUEDGE
- Here used 18 x 10 grid



Problem setup

- Begin with a simplified task:
  - sample a set of parameters $\theta_0 = \{D, \chi_e, \chi_i\}$
  - generate profiles $x_0 = \{n_{e,mid}, T_{e,mid}, T_{i,mid}, n_{e,div}, T_{e,div}, T_{i,div}\}$
  - apply simulation-based inference to learn posterior $p(\theta|x_0)$, comparing to original generating θ_0
- (Sequential) neural posterior estimation (SNPE) using masked autoregressive flows (mafs) implemented using sbi package in Python
- Two θ setups:
  - parameterized function for radial profiles of D, χe, χi . Embedding net for x.
  - Unparameterized, values at each grid-point. No embedding net.

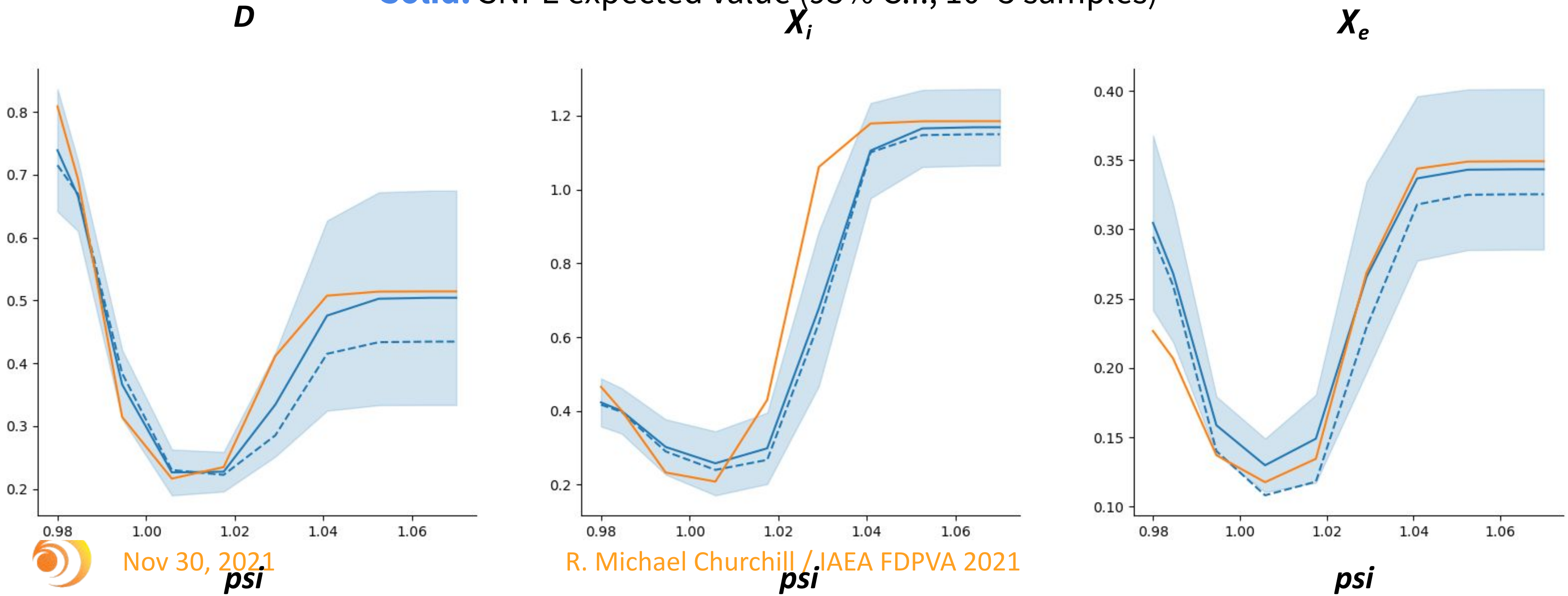


Results (parameterized θ_0)

N = 1000 simulations/round, 40 hidden features, 5 transforms **(Round 1)**

Orange: Actual ($\boldsymbol{\theta}_{\mathbf{o}}$) **Dashed:** SNPE maximum-a-posteriori

Solid: SNPE expected value (95% C.I., 10^5 samples)

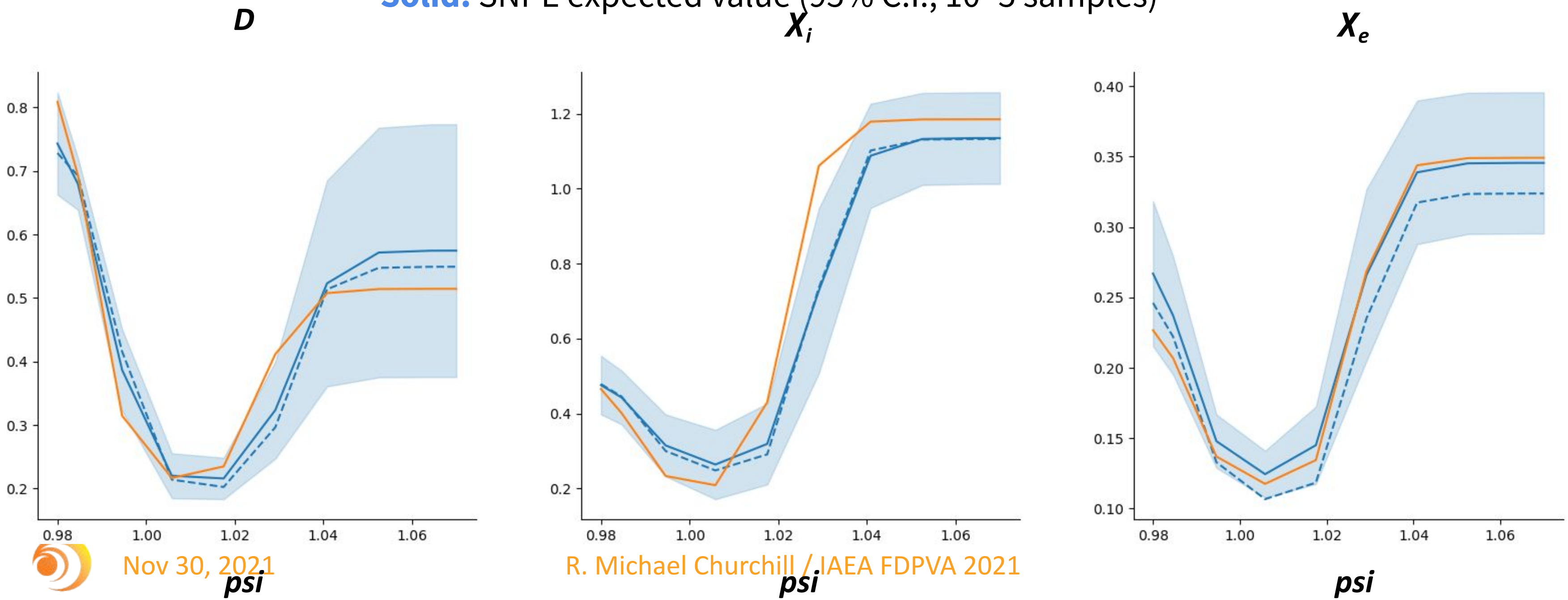


Results (parameterized θ_0)

N = 1000 simulations/round, 40 hidden features, 5 transforms **(Round 2)**

Orange: Actual ($\boldsymbol{\theta_o}$) **Dashed:** SNPE maximum-a-posteriori

Solid: SNPE expected value (95% C.I., 10^5 samples)



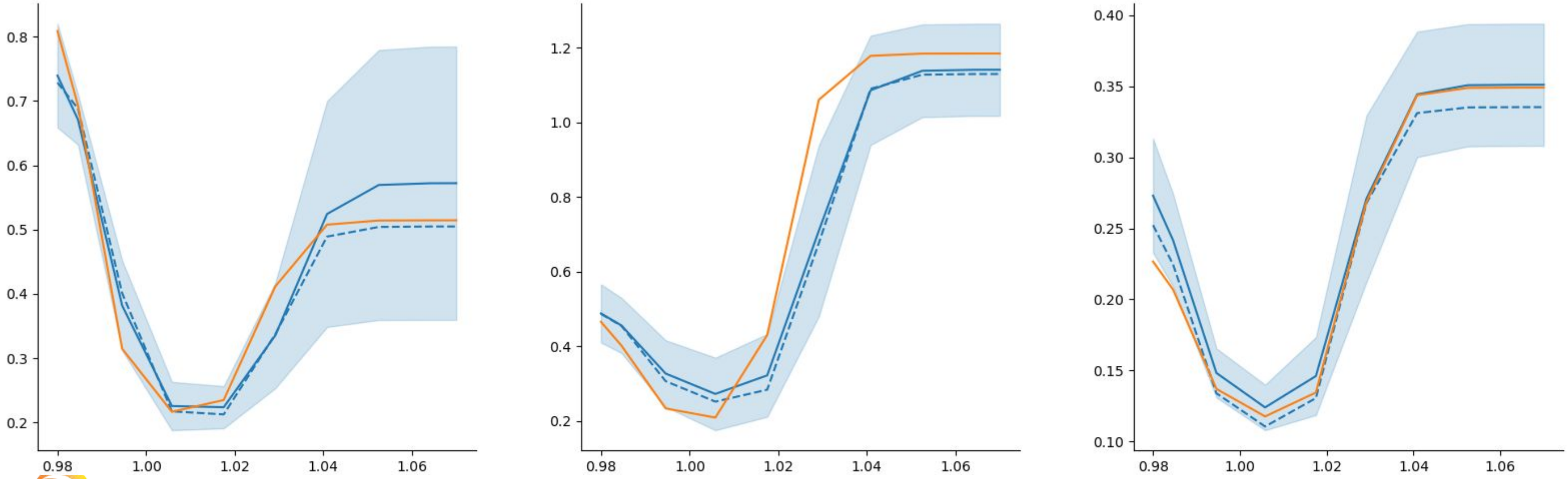
Results (parameterized θ_0)

N = 1000 simulations/round, 40 hidden features, 5 transforms **(Round 3)**

Orange: Actual ($\boldsymbol{\theta_o}$) **Dashed:** SNPE maximum-a-posteriori

Solid: SNPE expected value (95% C.I., 10^5 samples)

D ***X_i*** ***X_e***



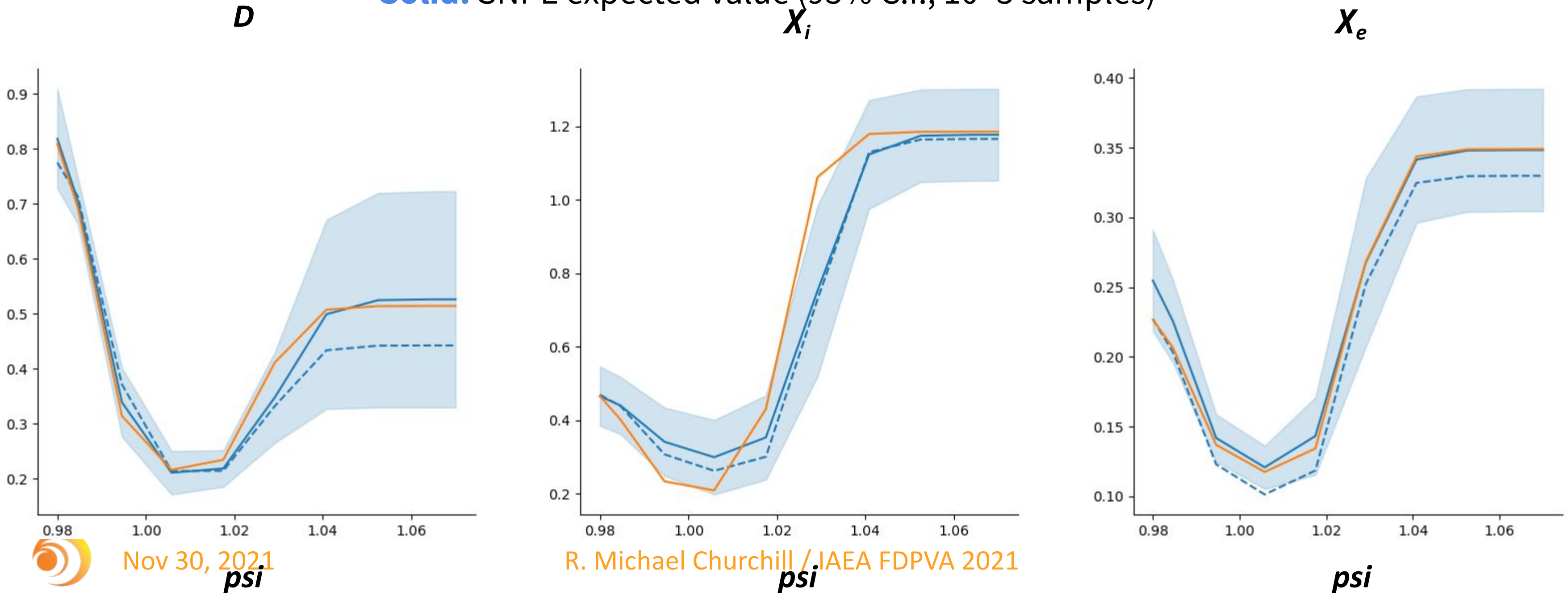
psi ***psi*** ***psi***

Results (parameterized θ_0)

N = 1000 simulations/round, 40 hidden features, 5 transforms **(Round 4)**

Orange: Actual ($\boldsymbol{\theta}_{\mathbf{o}}$) **Dashed:** SNPE maximum-a-posteriori

Solid: SNPE expected value (95% C.I., 10^5 samples)

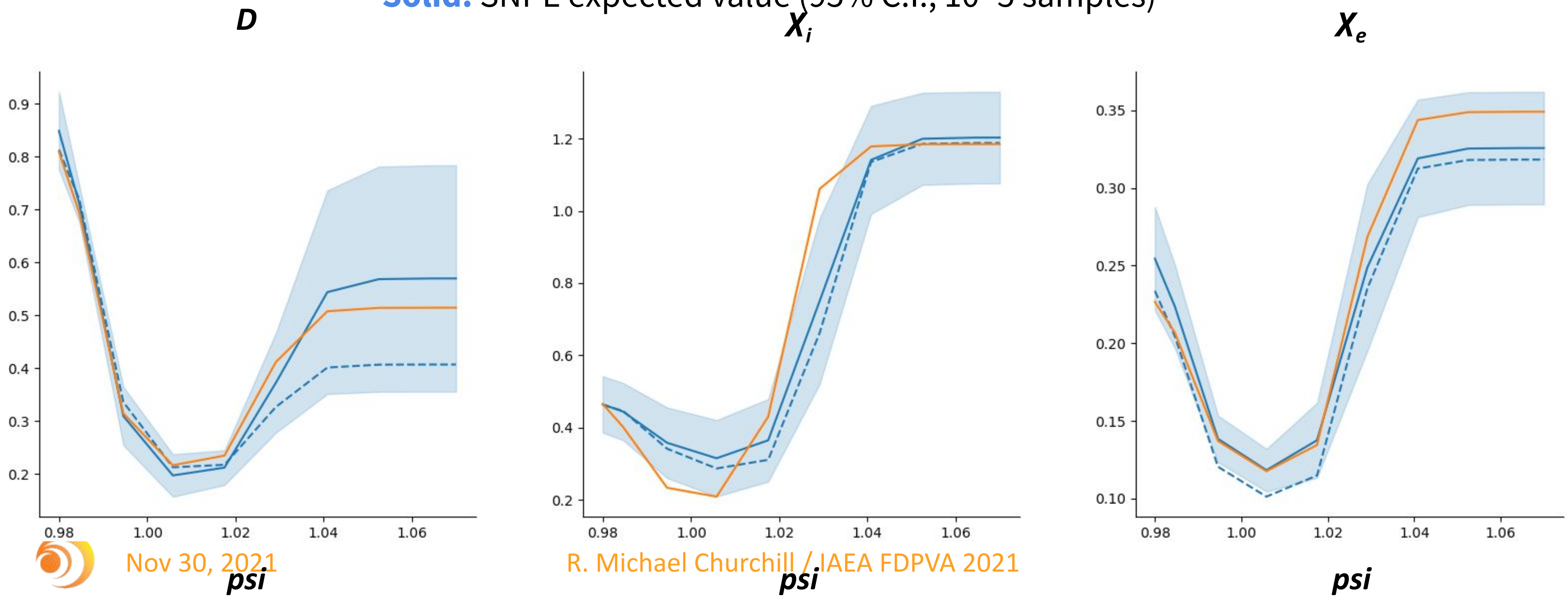


Results (parameterized θ_0)

N = 1000 simulations/round, 40 hidden features, 5 transforms **(Round 5)**

Orange: Actual ($\boldsymbol{\theta}_o$) **Dashed:** SNPE maximum-a-posteriori

Solid: SNPE expected value (95% C.I., 10^5 samples)

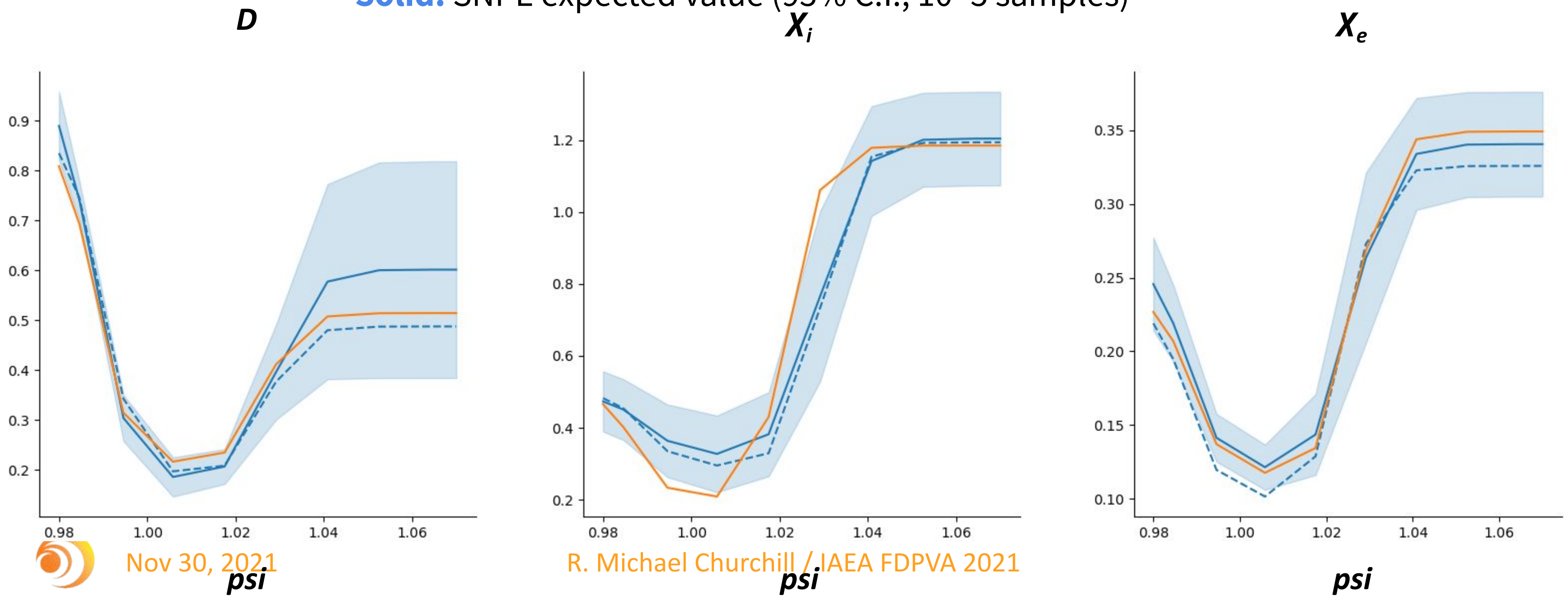


Results (parameterized θ_0)

N = 1000 simulations/round, 40 hidden features, 5 transforms **(Round 6)**

Orange: Actual ($\boldsymbol{\theta_o}$) **Dashed:** SNPE maximum-a-posteriori

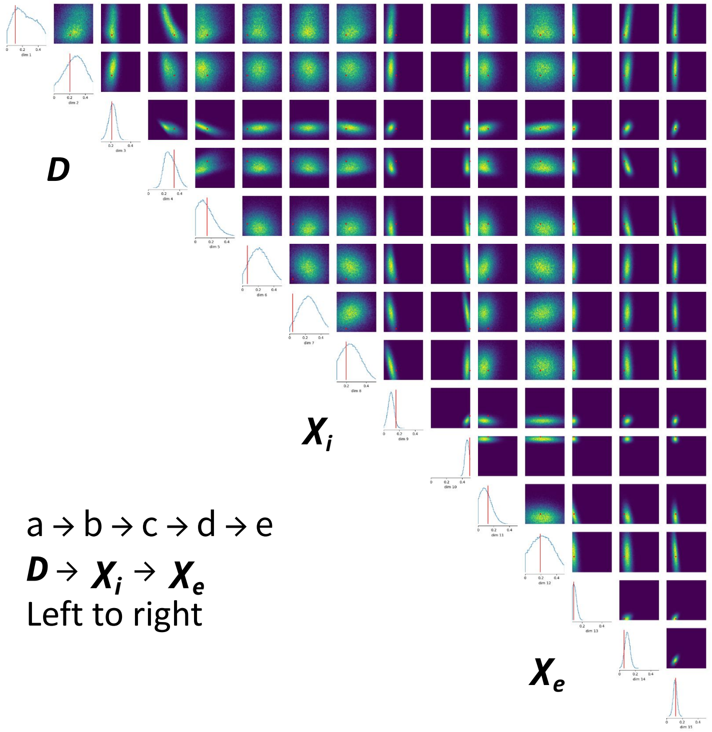
Solid: SNPE expected value (95% C.I., 10^5 samples)



Results (parameterized θ_0)

- SNPE is able to capture general shape for radial profiles
- Later rounds do not cause substantial shifts, increasing importance of first round
  - Leakage
- SOL height ("e" parameter) for ***D*** tends to be more diffuse
- Log-probability stabilizes after 2-3 rounds
  - Plateau?

Pairplot **(Round 1)**



a → b → c → d → e

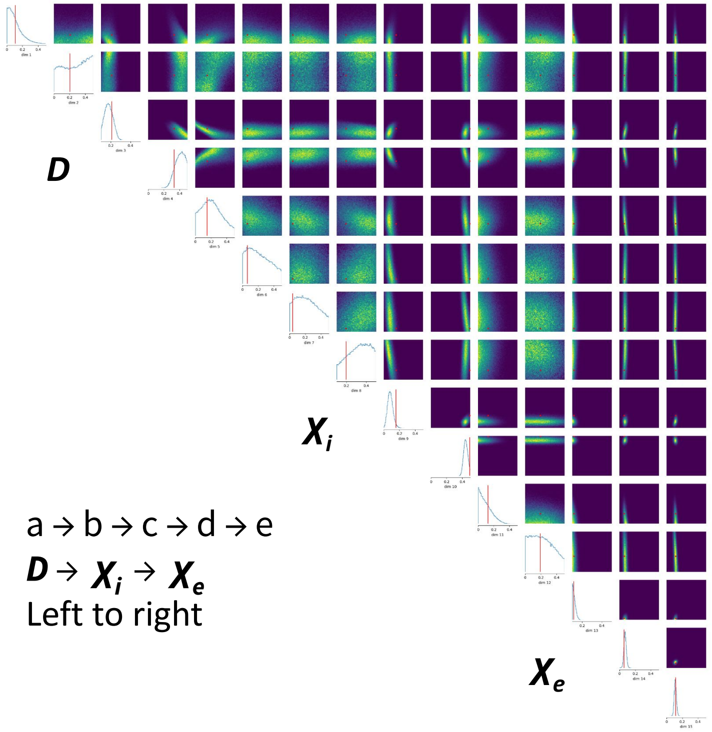
D → $\boldsymbol{X_i}$ → $\boldsymbol{X_e}$

Left to right

Results (parameterized θ_0)

- SNPE is able to capture general shape for radial profiles
- Later rounds do not cause substantial shifts, increasing importance of first round
    - Leakage
- SOL height ("e" parameter) for ***D*** tends to be more diffuse
- Log-probability stabilizes after 2-3 rounds
    - Plateau?

Pairplot **(Round 6)**



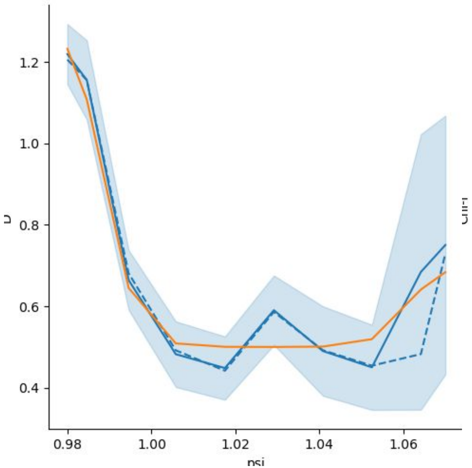
Results (unparameterized θ_0)

N = 2000 simulations/round, 80 hidden features, 10 transforms **(Round 1)**

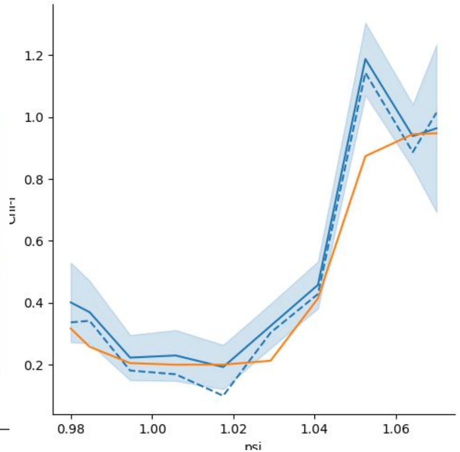
Orange: Actual ($\boldsymbol{\theta_o}$) **Dashed:** SNPE maximum-a-posteriori

Solid: SNPE expected value (95% C.I., 10^5 samples)

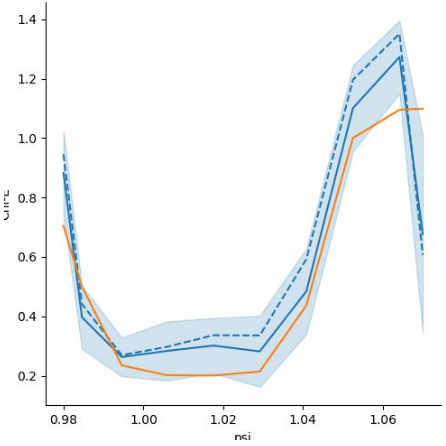
D



X_i



X_e



Results (unparameterized θ_0)

- Much sharper pair plots, possibly embedding net used previously was too restrictive
- Particle diffusion D still large uncertainties in far-SOL

Future work and conclusions

- Simulation-based inference (SBI) techniques show promise to leverage simulation models for statistical inference with experimental data
  - Gives uncertainties on ad-hoc parameters in simulations
  - Help see where simulator physics models break down
  - Reduces # of simulation runs needed for result
- Creating amortized models with large-scale number of simulations could lead to generic models which can be routinely applied to experimental output
- Need to determine dimensionality of θ that can be handled by neural density estimators based on normalizing flows
  - e.g. to use entire grid including poloidal, instead of flux surface average