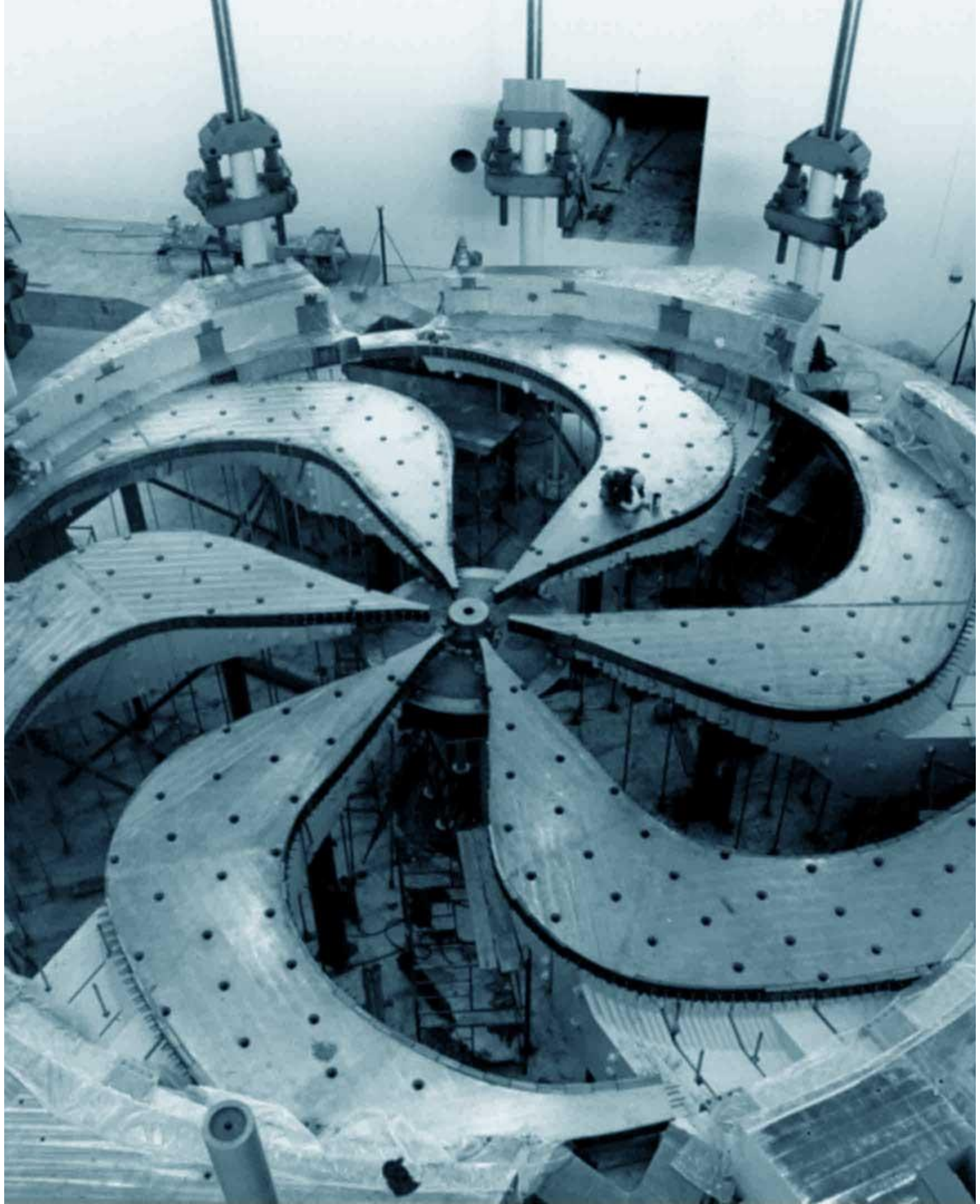
TRIUMF

Extrapolating nuclear many-body calculations with constrained Gaussian processes

Michael Gennari

P. Gysbers, W. Fedorko, P. Navrátil

2019-09-16



Discovery, accelerated

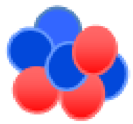
Ab initio nuclear theory and the no-core shell model (NCSM)

- Our goal is to solve the many-body Schrödinger equation

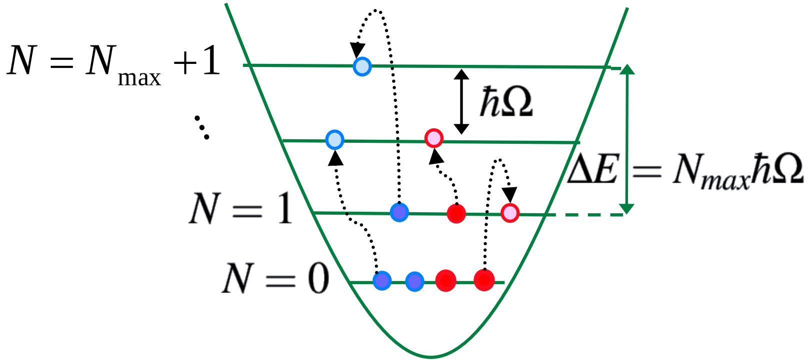
$$H|\Psi_k\rangle = E_k|\Psi_k\rangle \qquad H = \sum_{i}^{A} T_i + \sum_{i<j} V_{ij} + \sum_{i<j<f} V_{ijf} + \cdots$$

- NCSM is an *ab initio* non-relativistic approach with nucleons as the degrees of freedom
  - nuclear interactions are the only input
  - expand in anti-symmetrized products of harmonic oscillator single particle states (parameters N_{max} and $\hbar\Omega$)

A

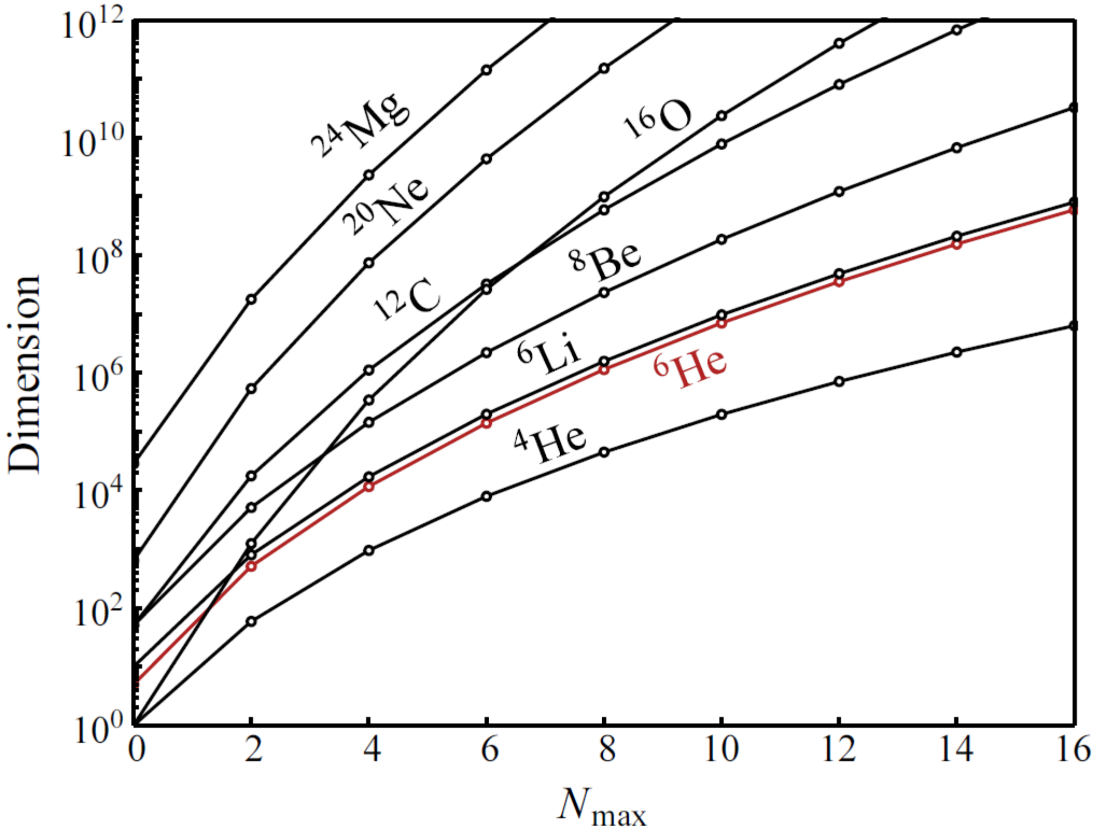


$$\Psi^A = \sum_{N=0}^{Nmax} \sum_{i} c_{Ni} \Phi_{Ni}^{A}$$



Problem…

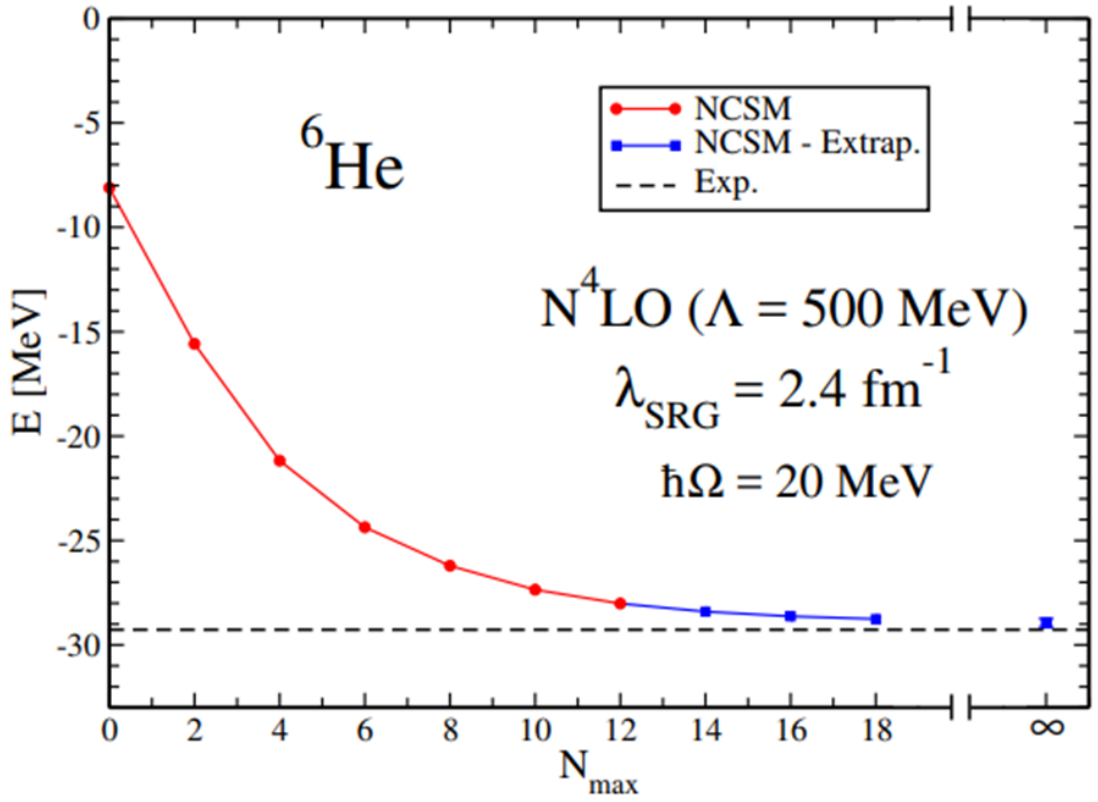
- NCSM is a rigorous model
- Computational complexity grows exponentially with basis truncation parameter N_{max}
- Calculations should converge as $N_{max} \rightarrow \infty$
- Meaningful calculations at very large N_{max} or for larger nuclear systems are computationally infeasible



More problems…

- Functional form of energy convergence curve with respect to N_{max} is unknown (near the $\hbar\Omega$ variational minimum)
- *Ad hoc* functions are used to attempt approximate extrapolations

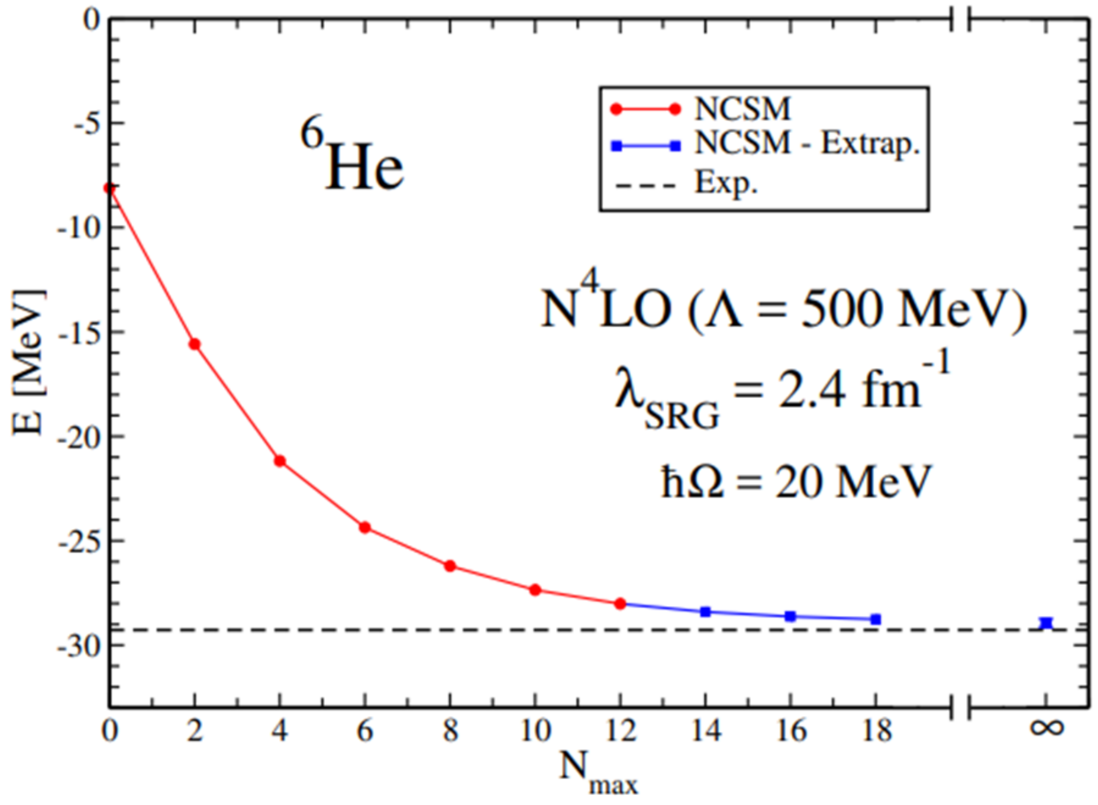
$$E = E_\infty + \alpha e^{-\beta N_{max}}$$



More problems…

- Functional form of energy convergence curve with respect to N_{max} is unknown (near the $\hbar\Omega$ variational minimum)
- *Ad hoc* functions are used to attempt approximate extrapolations

$$E = E_{\infty} + \alpha e^{-\beta N_{max}}$$

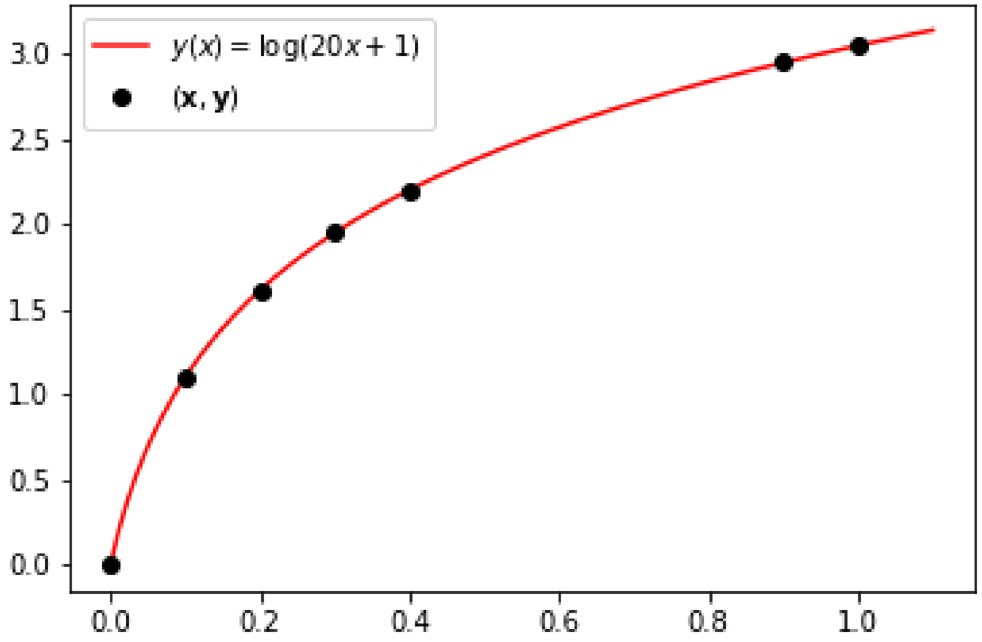


Require use of computational techniques to predict energy as $N_{max} \rightarrow \infty$ (want meaningful errors)

Mathematical problem statement

- Given some discrete data set $y = \{y_i\} = \{y(x_i)\}$, can we determine the underlying functional form $y(x)$
- Is it determined well enough to make predictions $y^* = y(x^*)$?

$y \sim -E$



$x \sim N_{max}$

Parametric and non-parametric models

Typical extrapolation - Parametric

- Select a functional form with some parameters $y(x) \sim f(x,p) + \epsilon(x)$
- Determine most likely values for parameters p given assumed error distribution $\epsilon(x)$

$$E = E_{\infty} + \alpha e^{-\beta N_{max}}$$

Gaussian process - Non-parametric

- Make assumptions on functional behaviour
- Consider conditional probability of predictions given data $p(y^*|y)$ to constrain function space further than a typical GP

Gaussian processes (GPs): Part 1

Overview

- GP is collection of infinite number of random variables with mean function $m(x)$ and covariance function $\mathrm{r}(x, x')$

$$\vec{y} = (y_1, \dots, y_n)^T \sim \mathcal{N}(\mu, \Sigma) \quad \rightarrow \quad y = y(x) \sim \mathcal{N}\big(m(x), r(x, x')\big)$$

- GPs are distributions over function spaces
  - provide interpolations with uncertainties
  - functional behaviour selected based on covariance of prediction sites and data points
  - can be improved by incorporating derivative constraints
- Extending work of Golchi et al [1], we attempt GP extrapolation by constraining first **and** second derivatives

Gaussian processes (GPs): Part 2

Key assumption on the prior

- Points $\{y_k\}$ drawn from multivariate Gaussian distribution (GD) with covariance function $C[y_i, y_j]$ defined by kernel choice

$$p\left(\begin{bmatrix} y_i \\ y_j \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mu[y_i] \\ \mu[y_j] \end{bmatrix}, \begin{bmatrix} C[y_i, y_i] & C[y_i, y_j] \\ C[y_j, y_i] & C[y_j, y_j] \end{bmatrix}\right)$$

$$C[y_i, y_j] = r(x_i, x_j) = \sigma^2 e^{-\frac{(x_i - x_j)^2}{2l^2}}$$

← Gaussian kernel

- Assumption of ‘smoothness’ on space of functions, nearby inputs have nearby outputs
- If $|x_i - x_j| \sim l$ then $|y_i - y_j| > \sigma$ is unlikely

Gaussian processes (GPs): Part 3

Calculation

- Extending to vectors, y (data) and y^* (function at select x^* points) form joint GD and are drawn with mean $\bar{\mu}$ and covariance matrix $\bar{\Sigma}$

$$p\left(\begin{bmatrix} y \\ y^* \end{bmatrix}\right) = \mathcal{N}(\bar{\mu}, \bar{\Sigma}) = \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu^* \end{bmatrix}, \begin{bmatrix} C & C_* \\ C_*^T & C_{**} \end{bmatrix}\right)$$

$$C^{(ij)} = C[y_i, y_j] = r(x_i, x_j)$$

$$C_*^{(ij)} = r(x_i, x_j^*) \qquad C_{**}^{(ij)} = r(x_i^*, x_j^*)$$

Gaussian 'trick'

- Can compute probability of function predictions y^* given input data

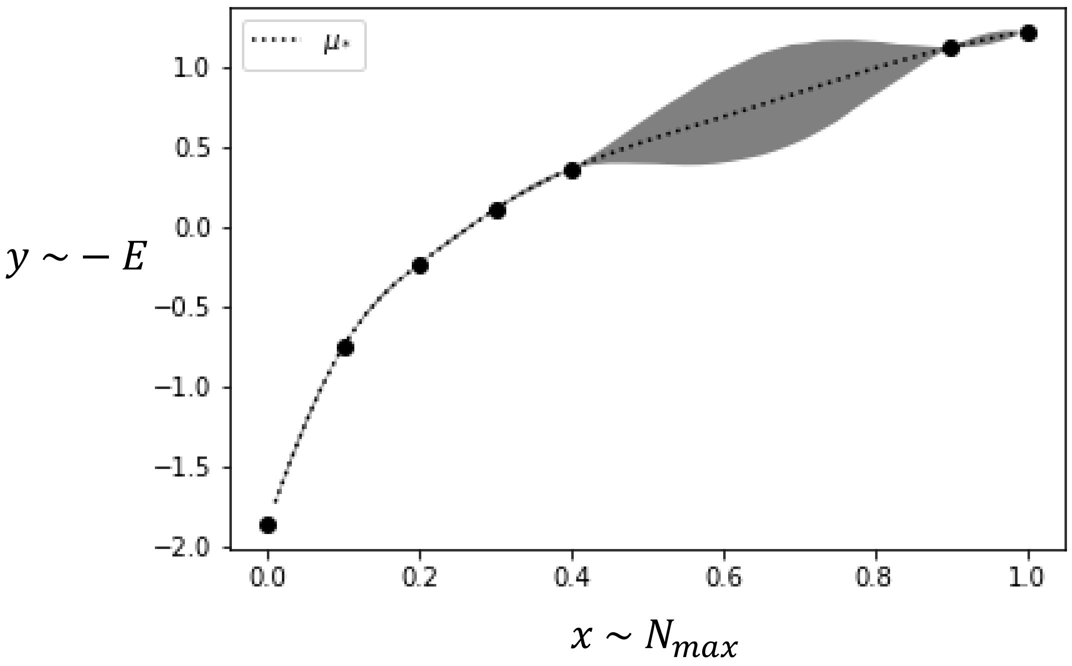
$$p(y^*|y) = \mathcal{N}(\mu_*, \Sigma_*)$$

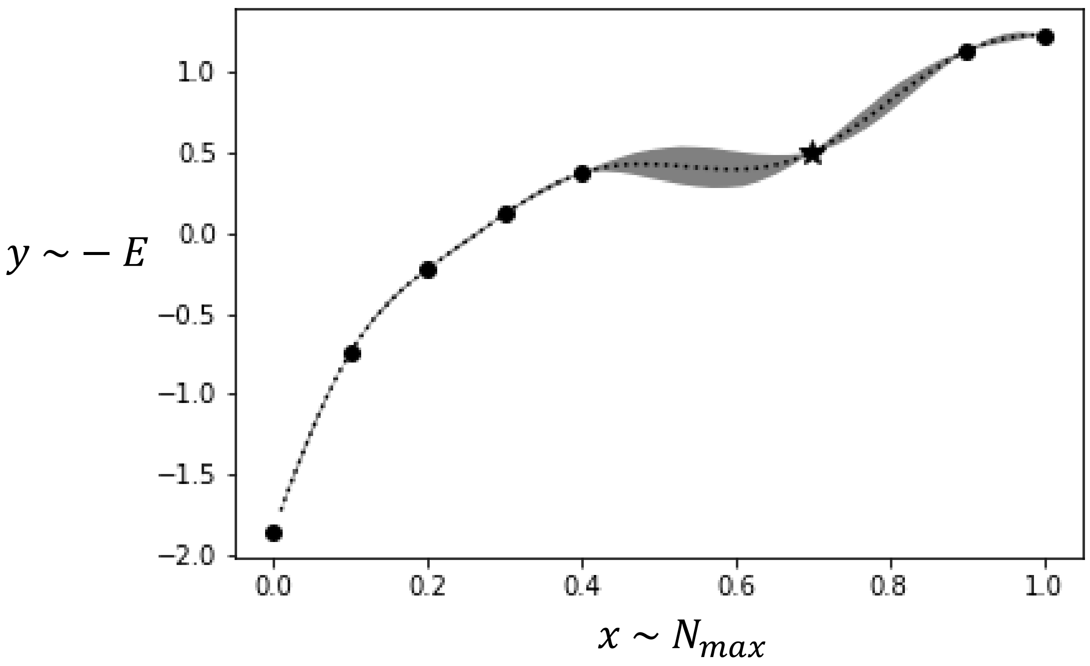
where

$$\mu_* = C_*^T C^{-1} y \qquad \Sigma_* = C_{**} - C_*^T C^{-1} C_*$$

Example of sampling

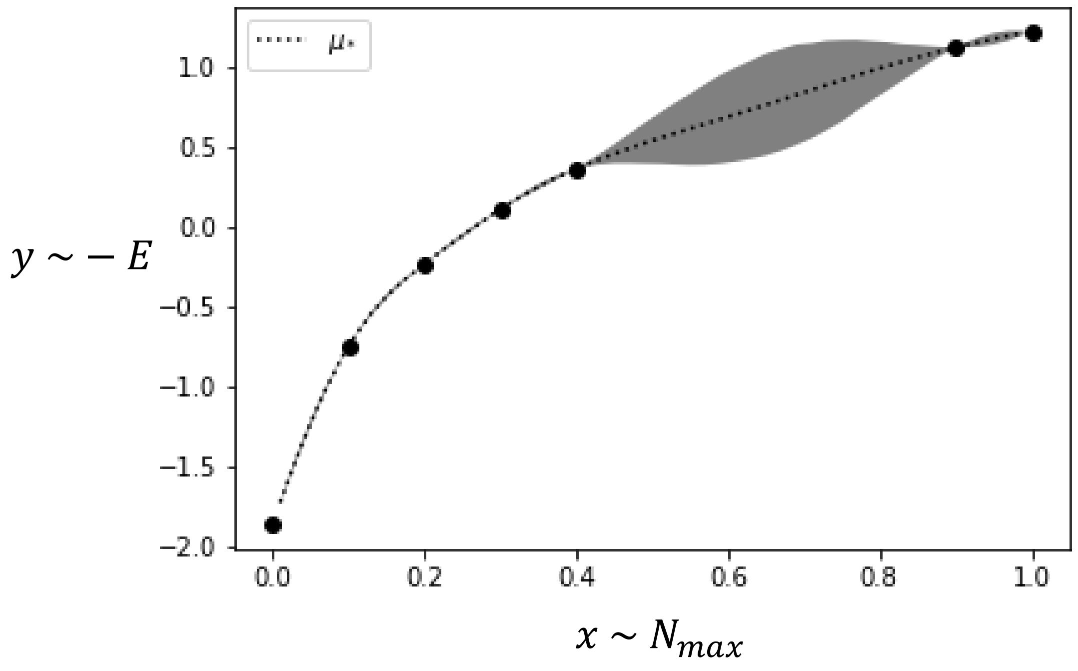
- New points are random samples from a Gaussian distribution
- Points can be sampled sequentially (depending on all the previous)

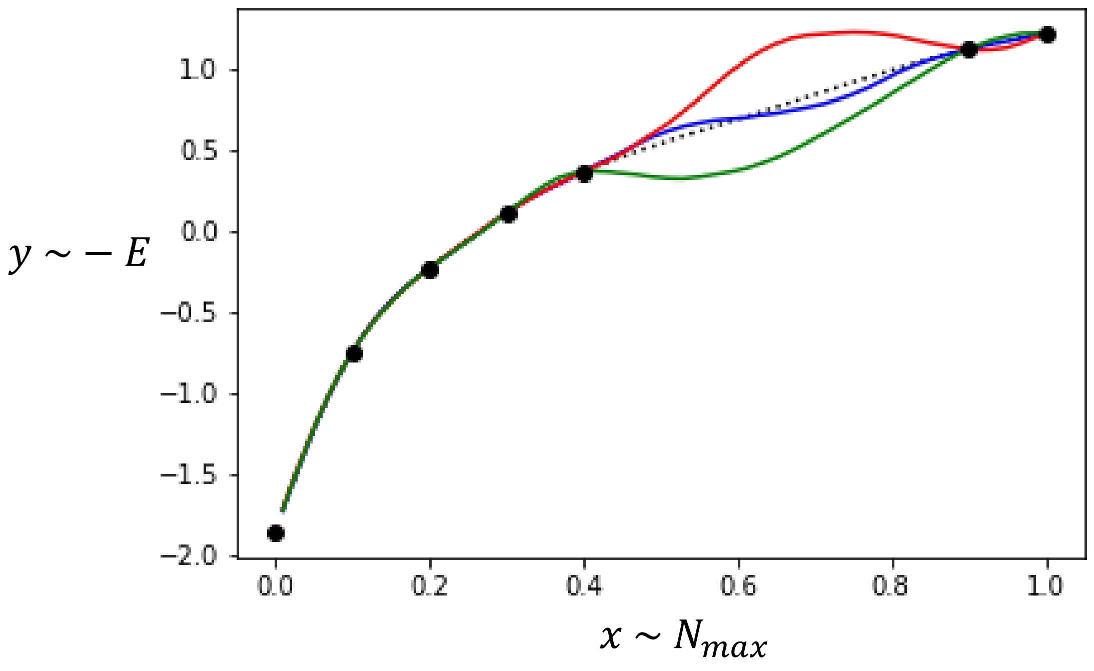




Example of sampling

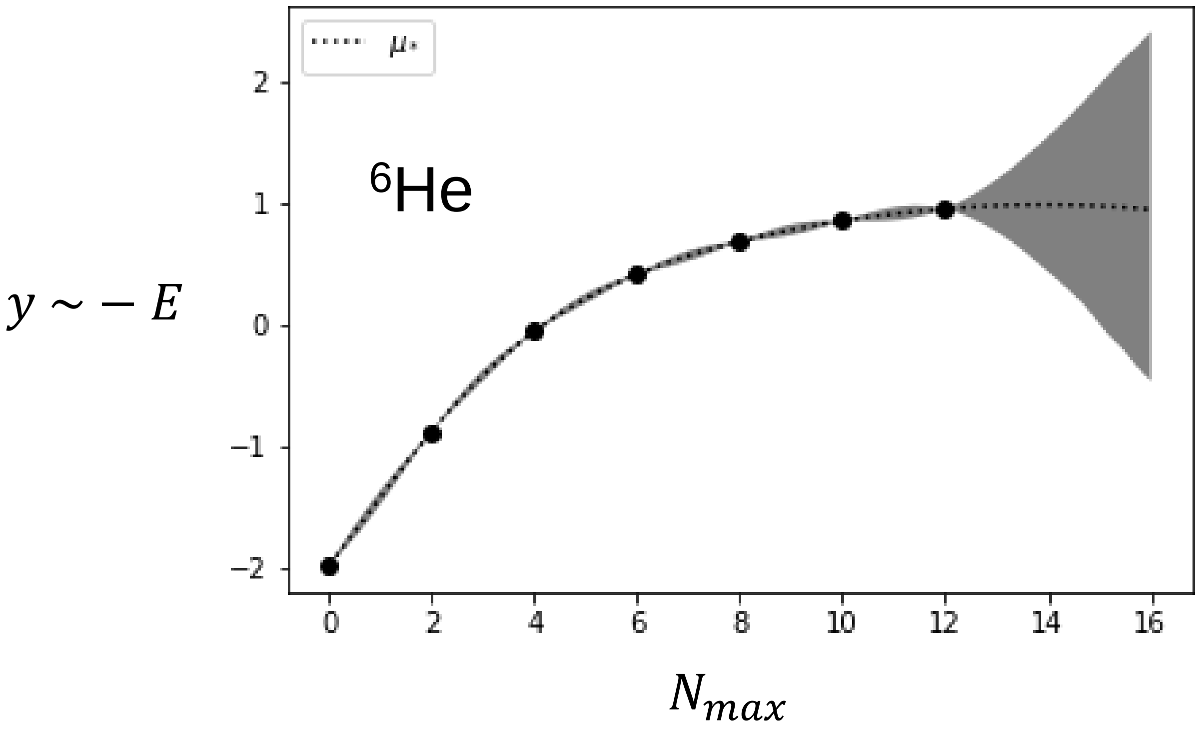
- New points are random samples from a Gaussian distribution
- Points can be sampled simultaneously (from higher-dimensional Gaussian)





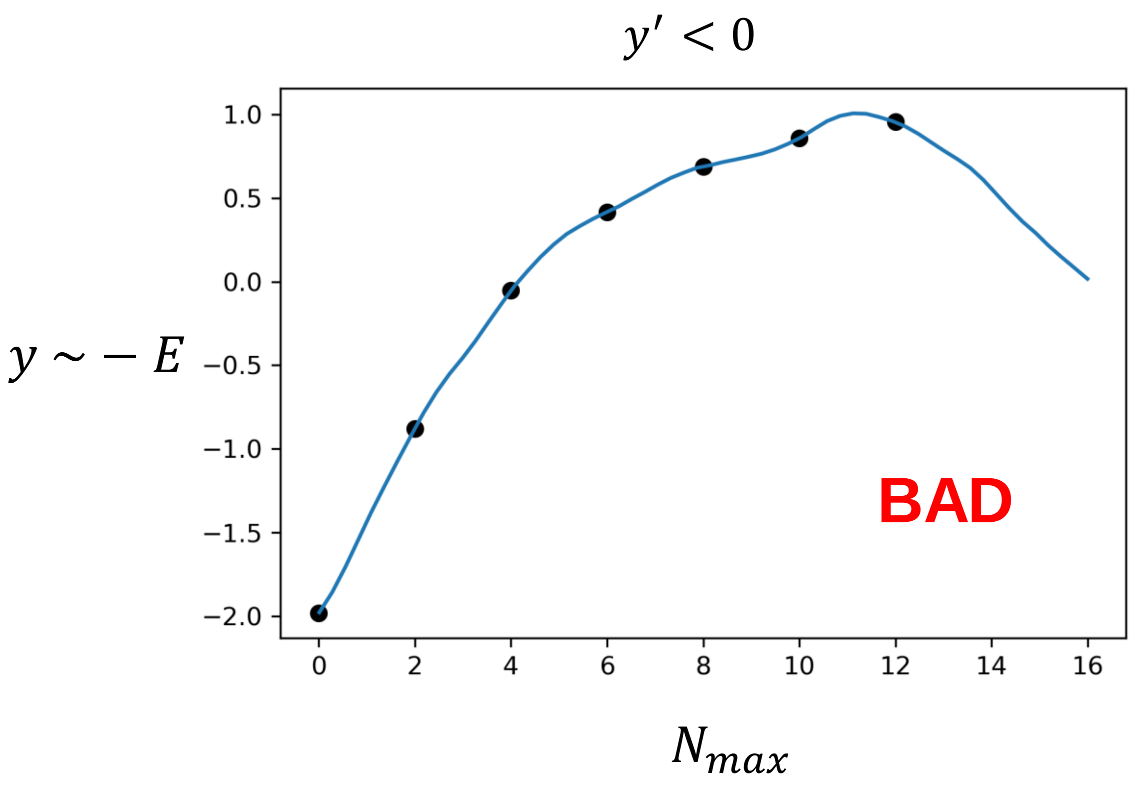
Extrapolation problem

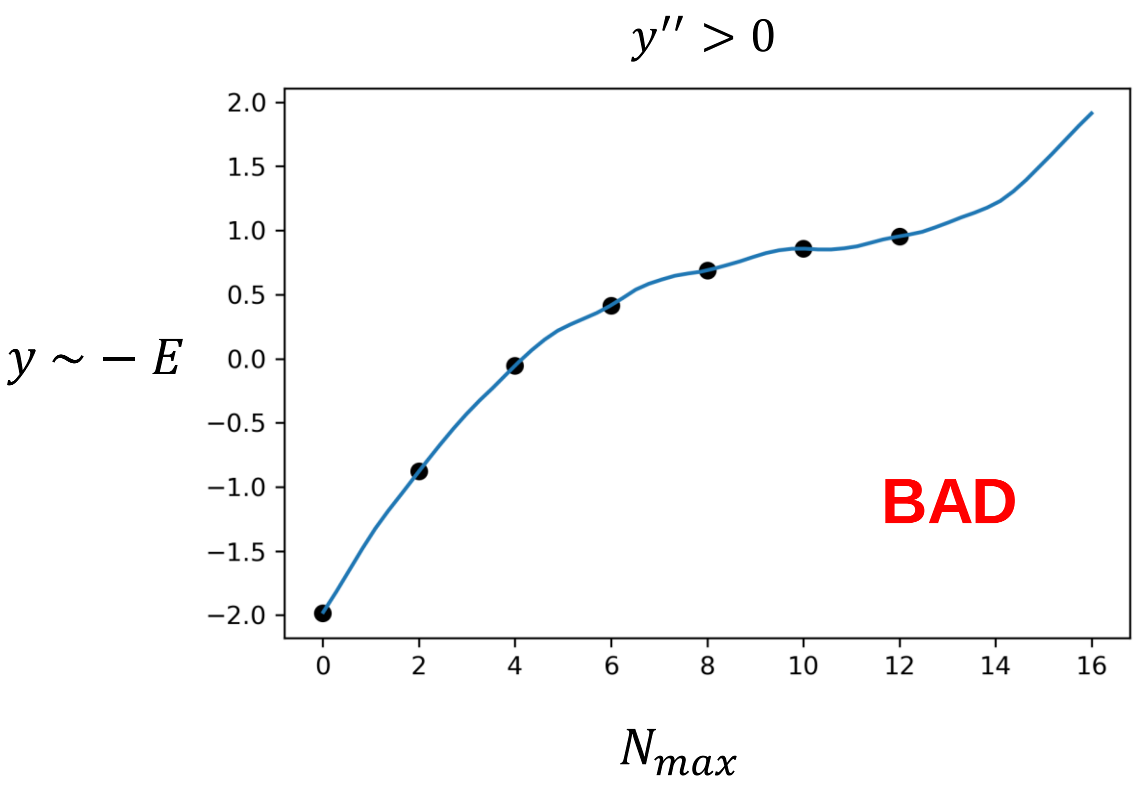
- While GP interpolates data as a requirement, error bars explode outside of the data range
- How do we deal with this radical behaviour?



Extrapolation problem

- We know what functional behaviour we need (and don’t want)
- Can we incorporate this information into the GP?





GPs with derivatives: Part 1

Constraining derivatives

- Define binary random variables representing sign of derivatives y_i' and y_j''

$$m(y_i') = \begin{cases} 1 & \text{if } y_i' > 0 \\ 0 & \text{otherwise} \end{cases} \qquad n(y_i'') = \begin{cases} 1 & \text{if } y_i'' < 0 \\ 0 & \text{otherwise} \end{cases}$$

- Weight probability of sample $p(y^*|y) \sim \mathcal{N}(\mu_*, \Sigma_*) \times m(y') \times n(y'')$ where

$$m(y') = \sum_i \left(m(y_i') = \begin{cases} 1 & \text{if } y_i' > 0 \\ 0 & \text{otherwise} \end{cases} \right) \qquad n(y'') = \sum_i \left(n(y_i'') = \begin{cases} 1 & \text{if } y_i'' < 0 \\ 0 & \text{otherwise} \end{cases} \right)$$

GPs with derivatives: Part 1

Constraining derivatives

How do we compute these?

- Define binary random variables representing sign of derivatives y_i' and y_j''

$$m(y_i') = \begin{cases} 1 & \text{if } y_i' > 0 \\ 0 & \text{otherwise} \end{cases} \qquad n(y_i'') = \begin{cases} 1 & \text{if } y_i'' < 0 \\ 0 & \text{otherwise} \end{cases}$$

- Weight probability of sample $p(y^*|y) \sim \mathcal{N}(\mu_*, \Sigma_*) \times m(y') \times n(y'')$ where

$$m(y') = \sum_i \left(m(y_i') = \begin{cases} 1 & \text{if } y_i' > 0 \\ 0 & \text{otherwise} \end{cases} \right) \qquad n(y'') = \sum_i \left(n(y_i'') = \begin{cases} 1 & \text{if } y_i'' < 0 \\ 0 & \text{otherwise} \end{cases} \right)$$

GPs with derivatives: Part 2

Computing derivatives

- The derivative of a GP is a GP
- $y_i' = \left.\frac{dy}{dx}\right|_{x=x_i'}$ and $y_i'' = \left.\frac{d^2y}{dx^2}\right|_{x=x_i''}$ also jointly Gaussian distributed with $\{y, y^*\}$
- Similarly draw derivative values at select x' and x'' points
- Can conveniently compute covariance of derivatives based on information from derivatives of kernel
- Calculate probability of all function behaviour given y

$$p\left(\begin{bmatrix} y^* \\ y' \\ y'' \end{bmatrix} \middle| \, y\right) = \mathcal{N}(\nu, \Sigma)$$

GPs with derivatives: Part 3

Obtaining distribution

- Want the posterior distribution $p\left(\begin{bmatrix} y^* \\ y' \\ y'' \end{bmatrix} \middle| y\right) \sim \mathcal{N}(\nu, \Sigma) \times m(y') \times n(y'')$

- GP method only generates the likelihood $p\left(\begin{bmatrix} y^* \\ y' \\ y'' \end{bmatrix} \middle| y\right) = \mathcal{N}(\nu, \Sigma)$

- Could generate samples (Monte-Carlo) from likelihood and accept/reject based on posterior distribution (inefficient)
- Perform Sequential Monte-Carlo (SMC) by adjusting distribution of samples over small constraint steps (efficient)

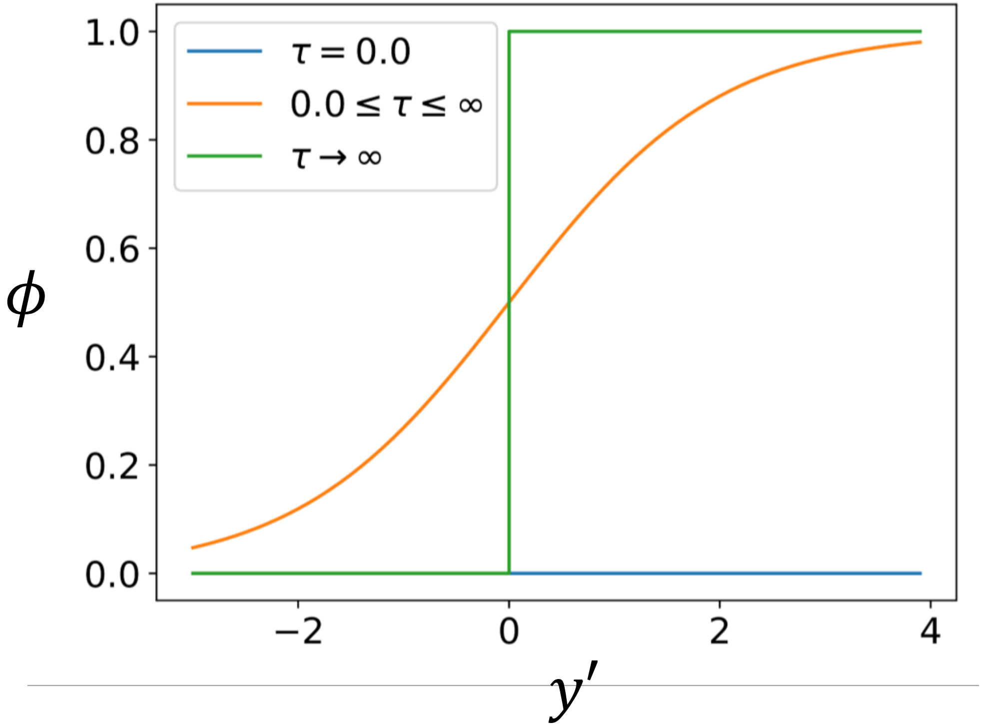
Sequential Monte-Carlo (SMC): Part 1

Parameterizing continuous constraints

- Apply constraint as function of new parameters τ_1 and τ_2
- Alter definition of variables representing sign of derivatives

$$m \sim \phi(\tau_1 y') \qquad n \sim \phi(\tau_2 y'')$$

- Construct discrete constraint schedule with small steps of constraint increase (simultaneous or asynchronous constraint application)



Sequential Monte-Carlo (SMC): Part 2

Setting up SMC

- Under constraints, y, y' and y'' are no longer GPs
- Inferences must be made using point-wise sampling, so algorithm is tailored to monotonic/convex function interpolation

SMC inputs

- Sequence of constraint parameters $\{\tau_1, \tau_2\}$
- Proposal distributions for GP parameters l and σ^2
- Particle number N

Sequential Monte-Carlo (SMC): Part 3

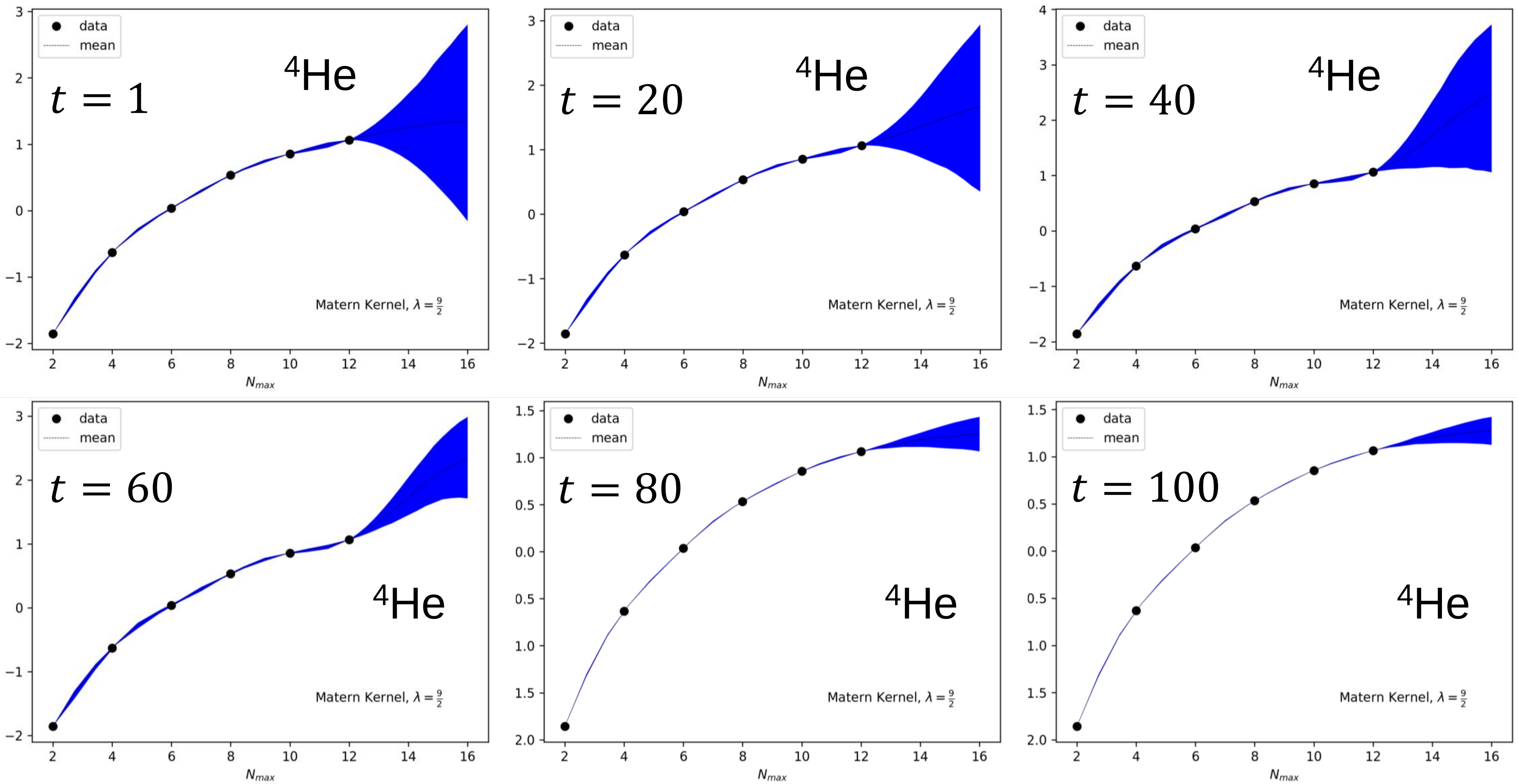
SMC algorithm and particle filter

- Draw N samples (particles with $[y^*, y', y'']^T$) from unconstrained GP
- For $\{\tau_1, \tau_2\}$ from 0 to ∞
  - for all particles $1{:}N$
    - $\rightarrow$ propose new l, σ^2 and particles $[y^*, y', y'']^T_{new}$ close to $[y^*, y', y'']^T$
    - $\rightarrow$ accept/reject new particles according to

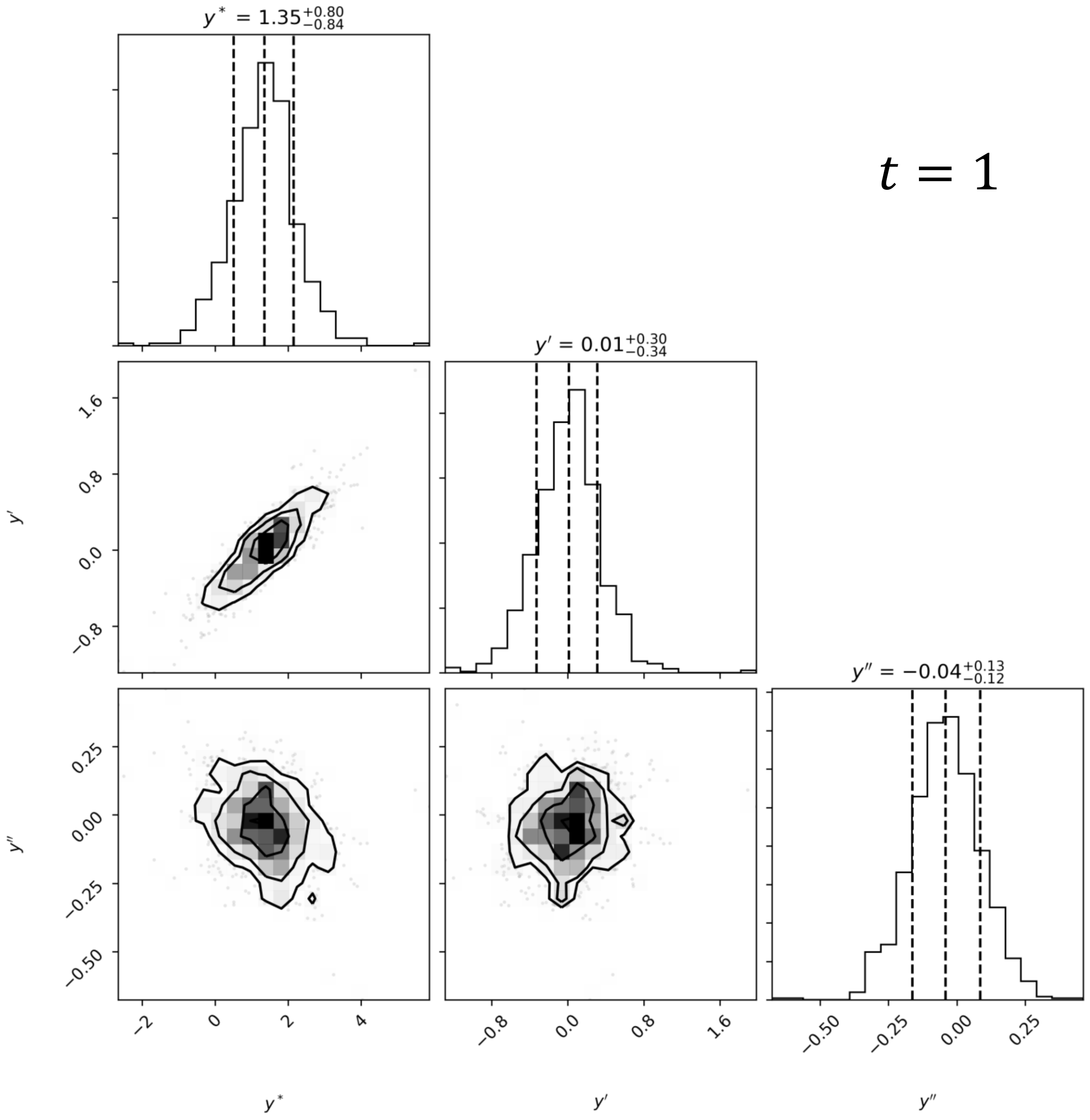
$$p\left(\begin{bmatrix} y^* \\ y' \\ y'' \end{bmatrix} \middle| y\right) \sim \mathcal{N}(\nu, \Sigma) \times m(y') \times n(y'')$$

  - resample: assign particle weights based on change under constraints
    - $\rightarrow$ throw away bad particles
    - $\rightarrow$ replace with copies of ‘better’ particles (weighted by constraints)

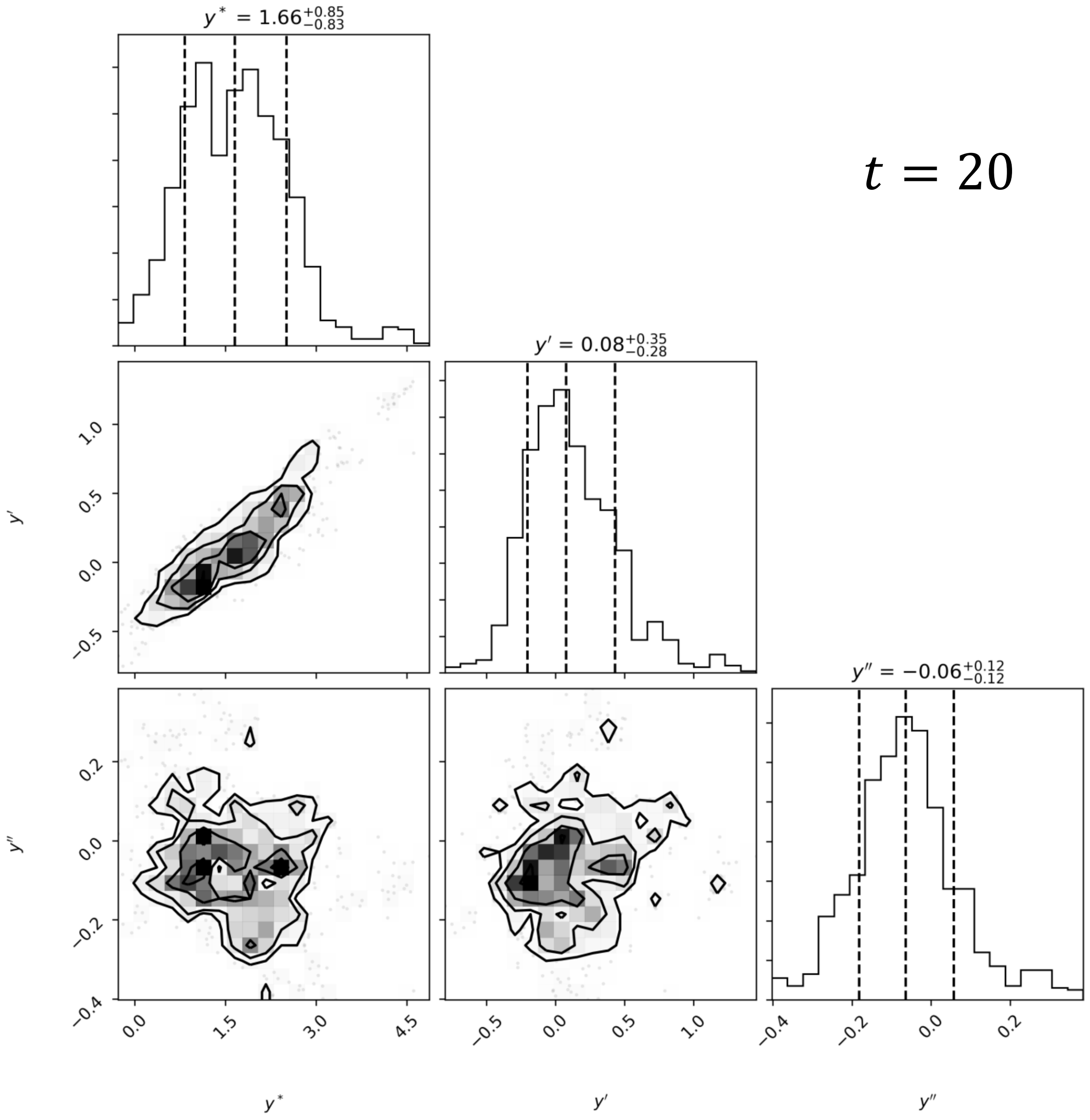
Results for ^{4}He at extrapolation value $N_{max} = 16$

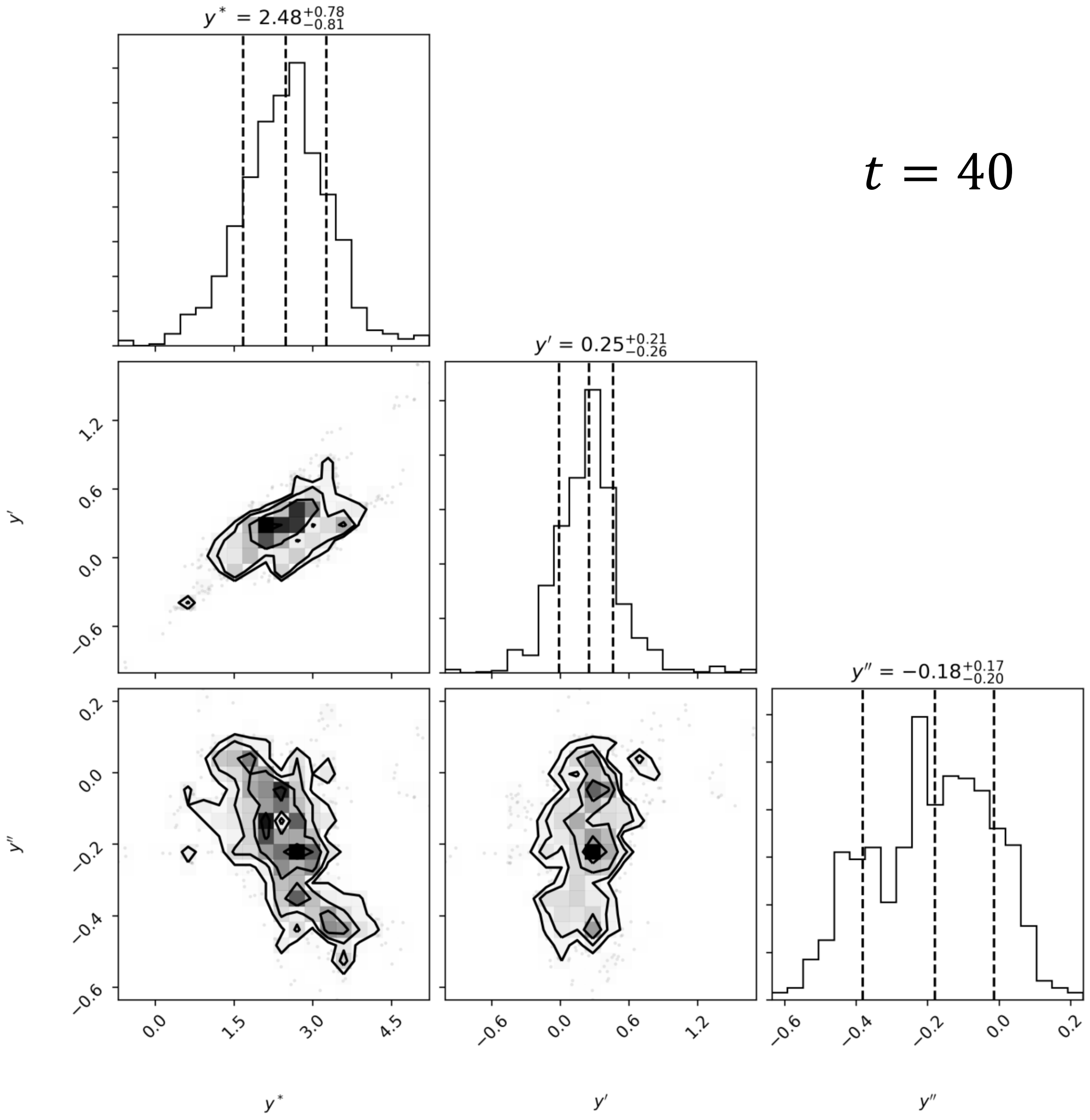


$t = 1$



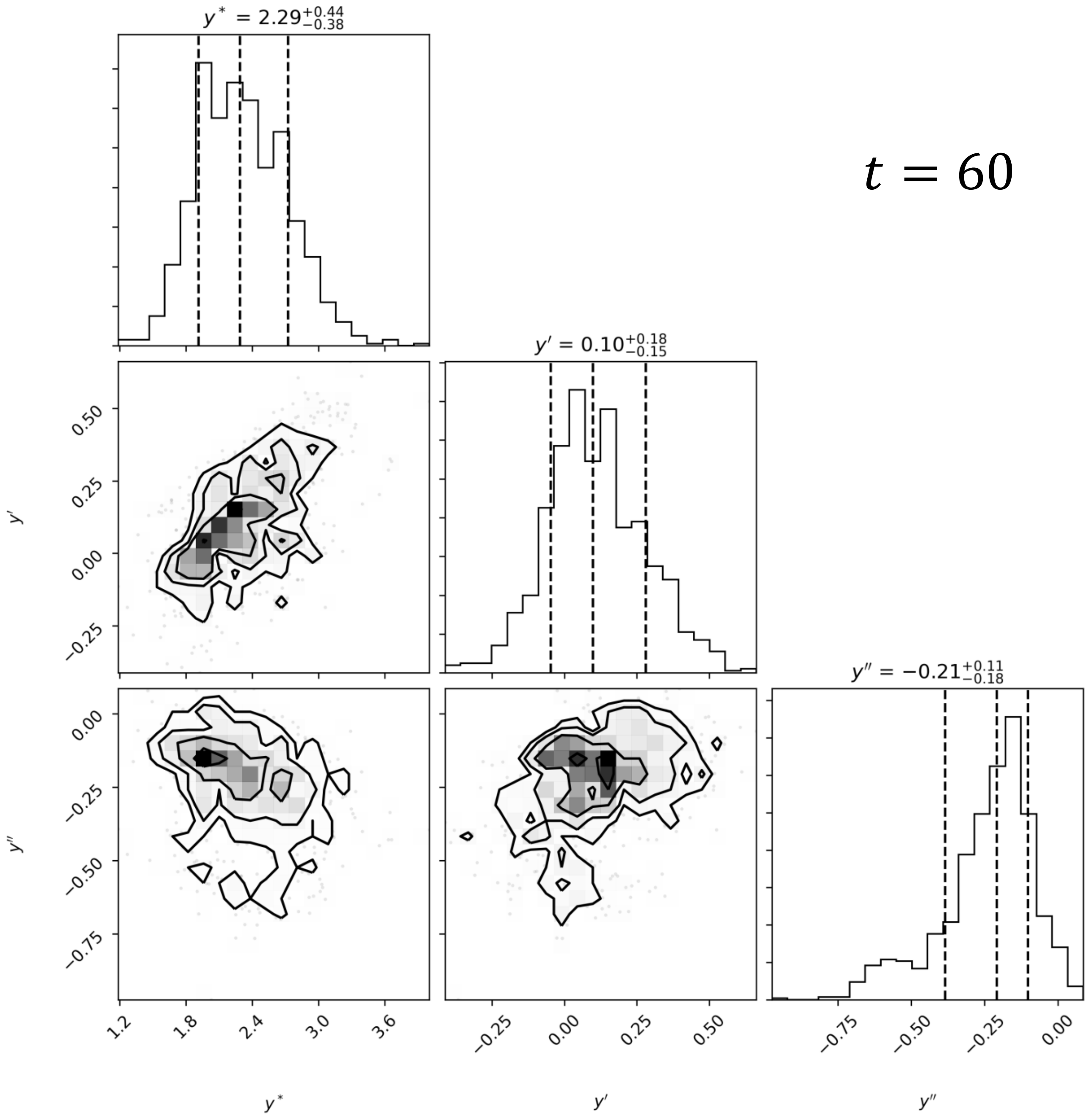
$t = 20$


y* = 1.66+0.85 −0.83
y′ = 0.08+0.35 −0.28
y″ = −0.06+0.12 −0.12
y′
y″
y*
y′
y″
0.0
1.5
3.0
4.5
−0.5
0.0
0.5
1.0
−0.4
−0.2
0.0
0.2

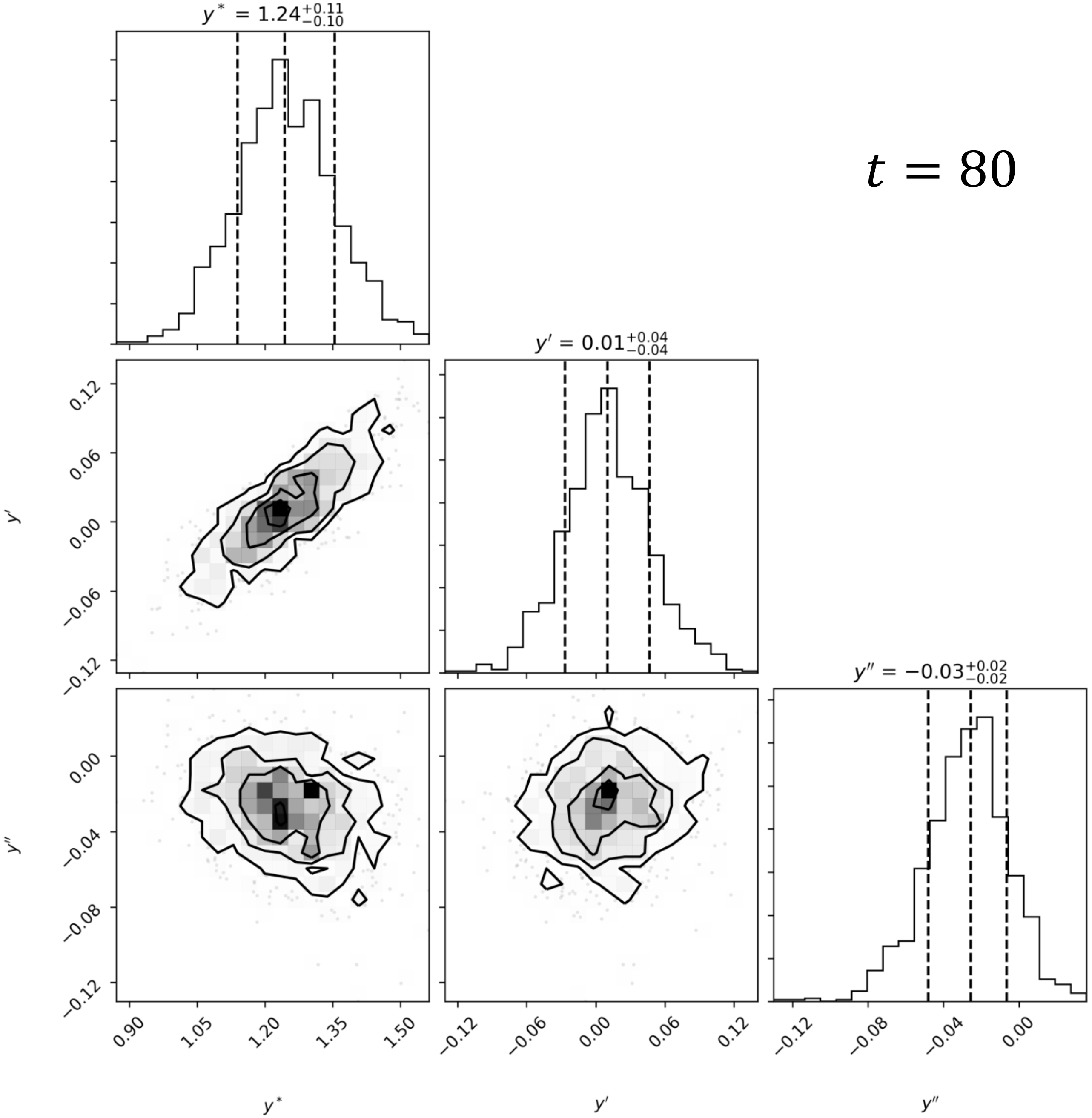


$t = 40$

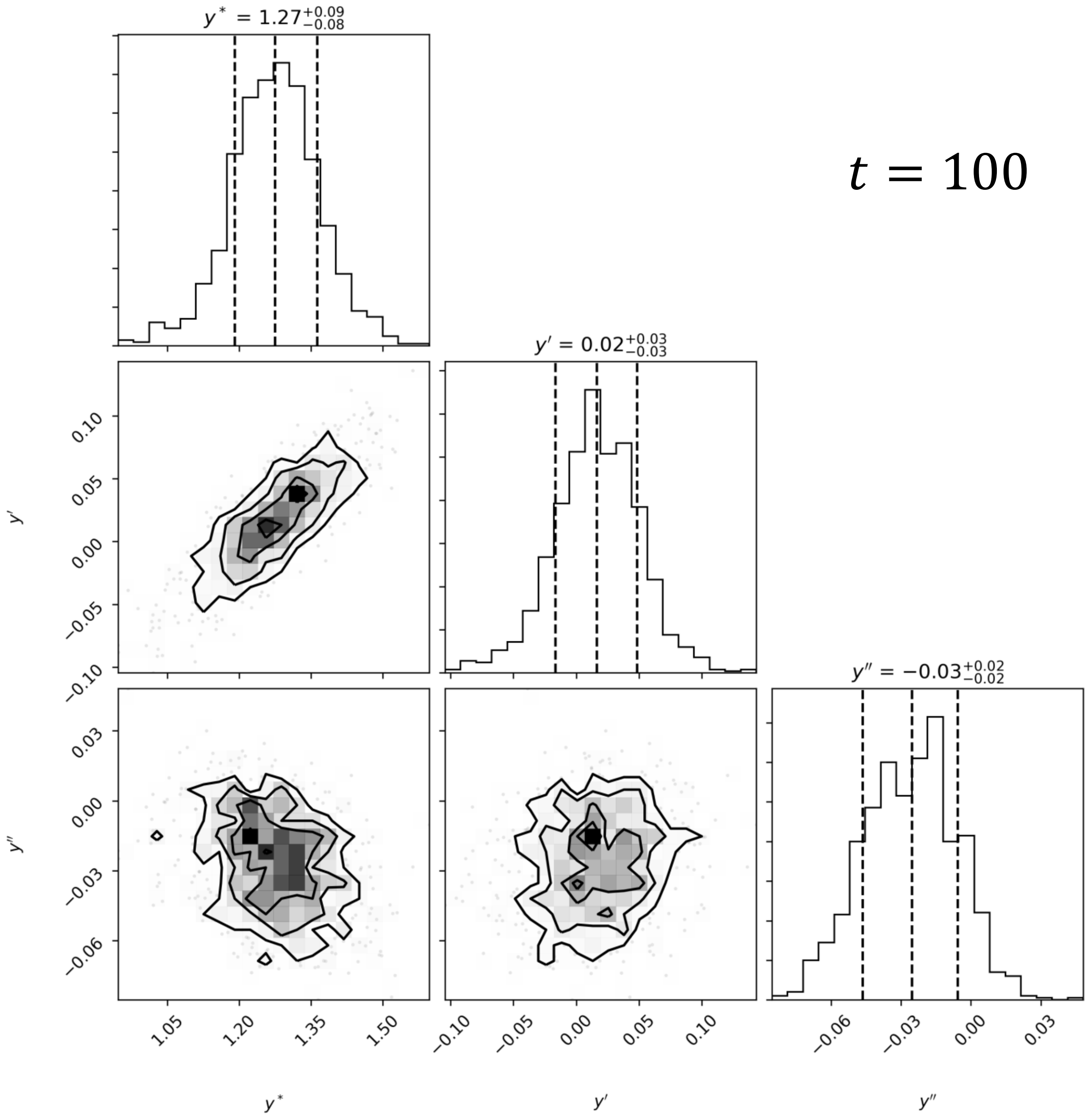
$t = 60$



$t = 80$

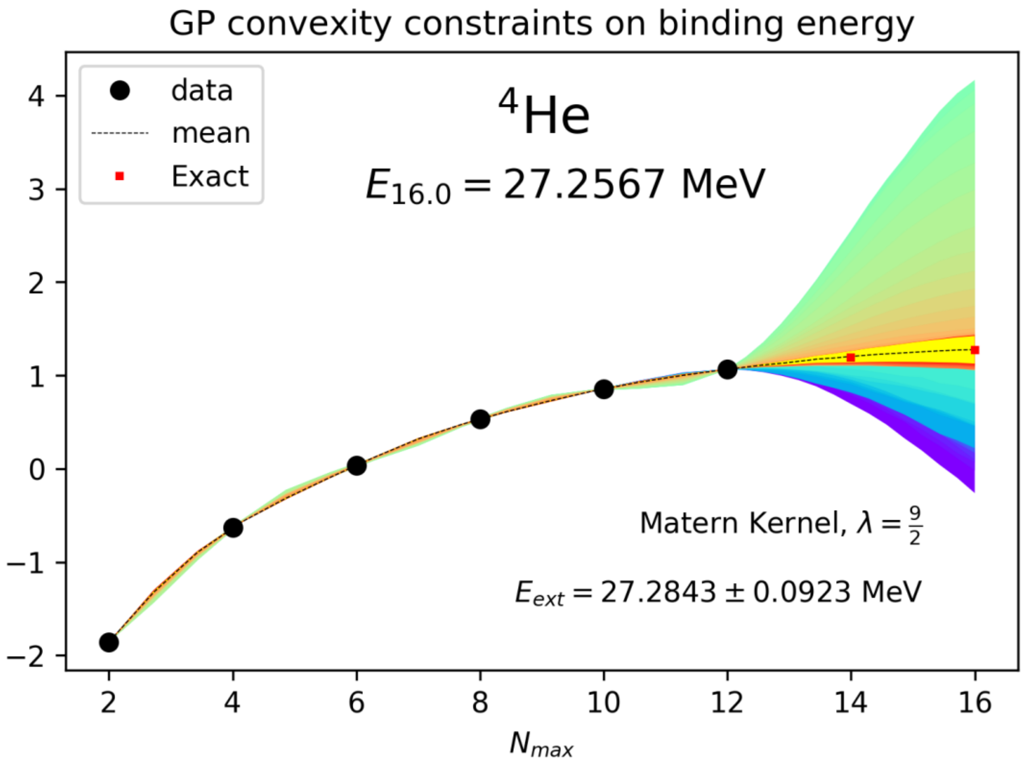


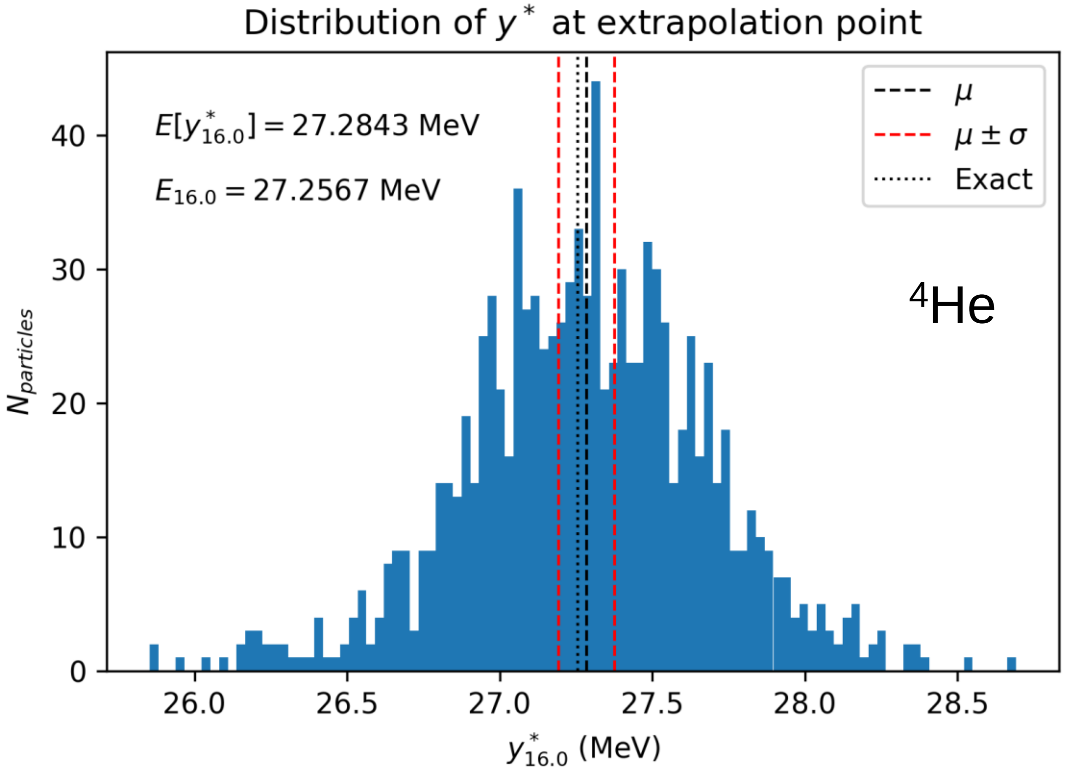
$y^* = 1.24^{+0.11}_{-0.10}$
$y' = 0.01^{+0.04}_{-0.04}$
$y'' = -0.03^{+0.02}_{-0.02}$
y'
y''
y^*
0.12
0.06
0.00
−0.06
−0.12
−0.04
−0.08
0.90
1.05
1.20
1.35
1.50



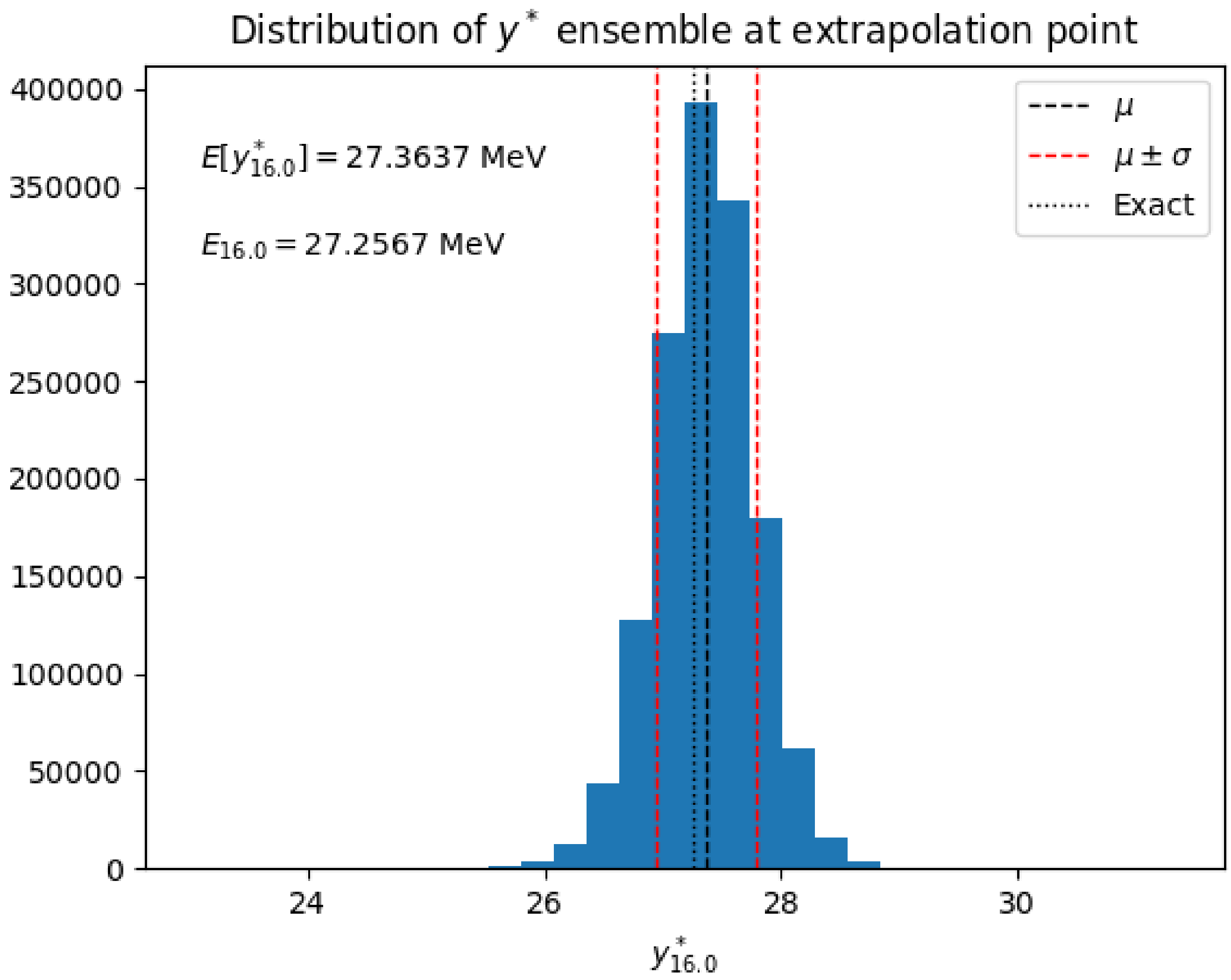
$$t = 100$$

Results for ^{4}He at extrapolation value $N_{max} = 16$





Ensemble results for ^{4}He at extrapolation value $N_{max} = 16$



Conclusions

Summary

- Application of derivative knowledge correctly constraining function space
- Demonstrated viability of extrapolation by constrained GPs

Outlook

- Can we push the extrapolations to much larger N_{max} values?
- Add additional harmonic oscillator basis parameter $\hbar\Omega$ to model

References

1. S. Golchi, D. R. Bingham, H. Chipman, and D. A. Campbell, Monotone emulation of computer experiments, SIAM/ASA J. Uncertain. Quantif., 3 (2015), pp. 370–392.

TRIUMF

Thank you
Merci

www.triumf.ca

Follow us **@TRIUMFLab**

Discovery,
accelerated

GPs with derivatives

$$p\left(\begin{bmatrix} y \\ y^* \\ y' \\ y'' \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \mu \\ \mu^* \\ \mu_1 \\ \mu_2 \end{bmatrix}, \begin{bmatrix} C & C_* & C_1 & C_2 \\ C_*^T & C_{**} & C_{1*} & C_{2*} \\ C_1^T & C_{*1} & C_{11} & C_{12} \\ C_2^T & C_{*2} & C_{21} & C_{22} \end{bmatrix}\right)$$

$$C_{*1}^{(ij)} = C[y_i^*, y_j'] = \frac{\partial}{\partial x_j} r(x_i^*, x_j')$$

$$\vdots$$

$$C_{22}^{(ij)} = C[y_i'', y_j''] = \frac{\partial^2}{\partial^2 x_i} \frac{\partial^2}{\partial^2 x_j} r(x_i'', x_j'')$$

$$p\left(\begin{bmatrix} y^* \\ y' \\ y'' \end{bmatrix} | y\right) = \mathcal{N}(\nu, \Sigma)$$

$$\nu = [C_*, C_1, C_2] C^{-1} y$$

$$\Sigma = \begin{bmatrix} C_{**} & C_{1*} & C_{2*} \\ C_{*1} & C_{11} & C_{12} \\ C_{*2} & C_{21} & C_{22} \end{bmatrix} - [C_*, C_1, C_2] C^{-1} \begin{bmatrix} C_* \\ C_1 \\ C_2 \end{bmatrix}$$