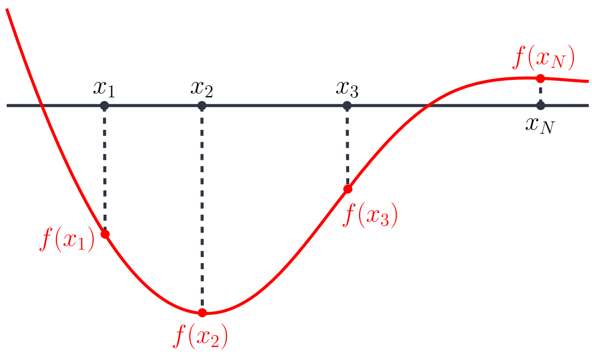
Numerical Integration

Cheng Soon Ong
Marc Peter Deisenroth

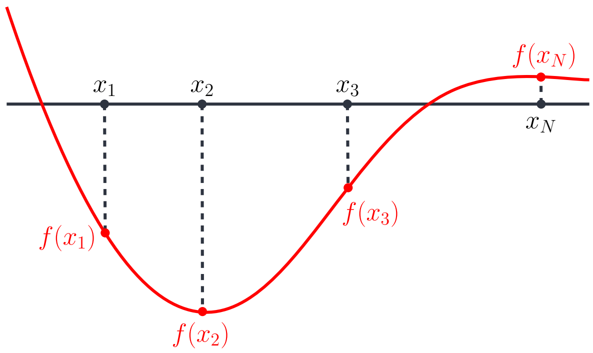
December 2020

Setting



- Approximate
$$\int_a^b f(x)dx \approx \sum_{n=1}^{N} w_n f(x_n), \quad x \in \mathbb{R}$$
- Nodes x_n and corresponding function values $f(x_n)$

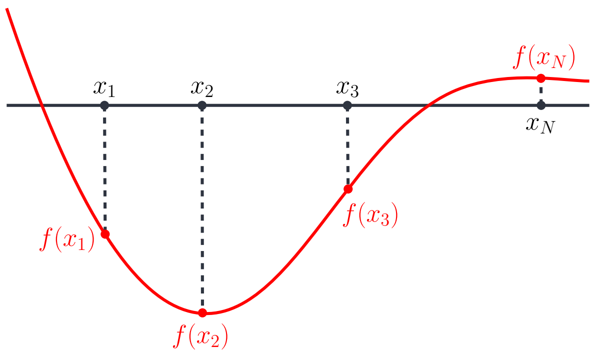
Numerical integration (quadrature)



Key idea

Approximate f using an interpolating function that is easy to integrate (e.g., polynomial)

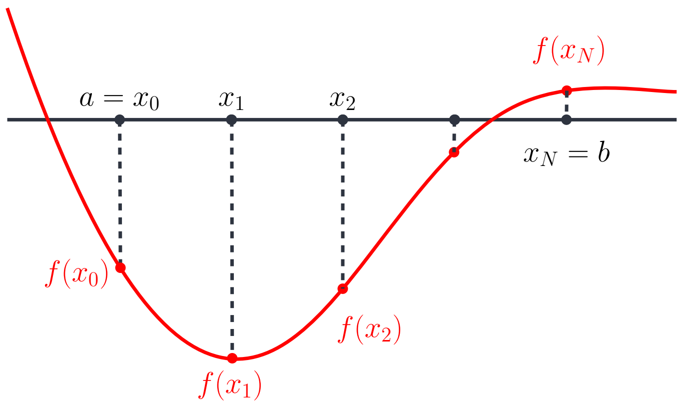
Quadrature approaches



Quadrature	Interpolant	Nodes
Newton–Cotes	low-degree polynomials	equidistant
Gaussian	orthogonal polynomials	roots of polynomial
Bayesian	Gaussian process	user defined

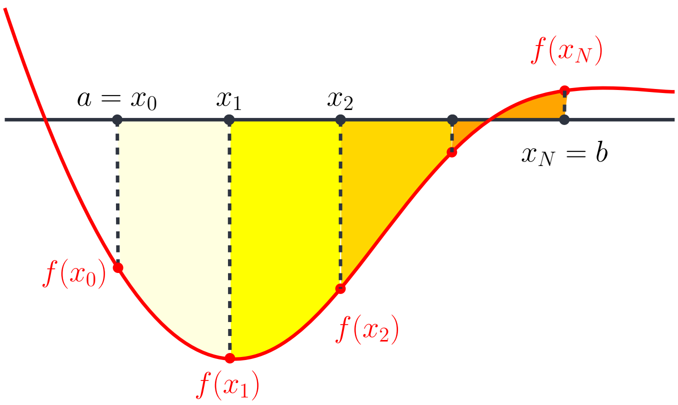
Newton–Cotes Quadrature

Overview



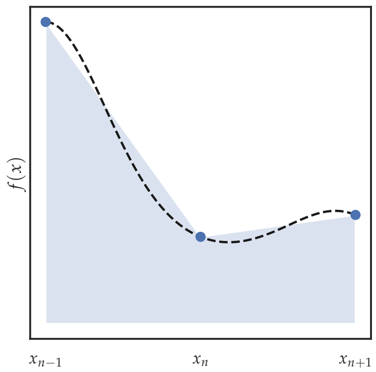
- Equidistant nodes $a = x_0, \ldots, x_N = b$ ⏩ Partition interval $[a, b]$
- Approximate f in each partition with a low-degree polynomial

Overview



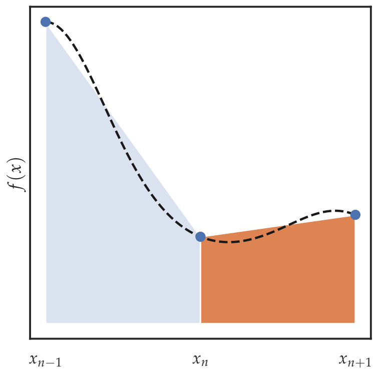
- Equidistant nodes $a = x_0, \dots, x_N = b$ ⏩ Partition interval $[a, b]$
- Approximate f in each partition with a low-degree polynomial
- Compute integral for each partition analytically and sum them up

Trapezoidal rule



- Partition $[a, b]$ into N segments with equidistant nodes x_n
- **Locally linear approximation** of f between nodes

Trapezoidal rule (2)

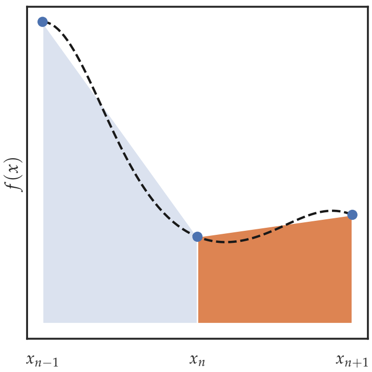


- Area of a trapezoid with corners $(x_n, x_{n+1}, f(x_{n+1}), f(x_n))$

$$\int_{x_n}^{x_{n+1}} f(x)dx \approx \frac{h}{2}\big(f(x_n) + f(x_{n+1})\big)$$

$h := |x_{n+1} - x_n|$ ▶▶▶Distance between nodes

Trapezoidal rule (2)



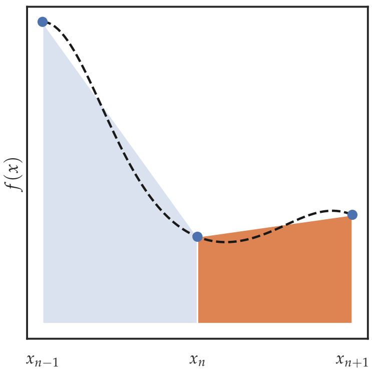
- Area of a trapezoid with corners $(x_n, x_{n+1}, f(x_{n+1}), f(x_n))$

$$\int_{x_n}^{x_{n+1}} f(x)dx \approx \frac{h}{2}\big(f(x_n) + f(x_{n+1})\big)$$

$h := |x_{n+1} - x_n|$ ⏵⏵⏵Distance between nodes

- Error $\mathcal{O}(h^2)$

Trapezoidal rule (2)



- Area of a trapezoid with corners $(x_n, x_{n+1}, f(x_{n+1}), f(x_n))$

$$\int_{x_n}^{x_{n+1}} f(x)dx \approx \frac{h}{2}\big(f(x_n) + f(x_{n+1})\big)$$

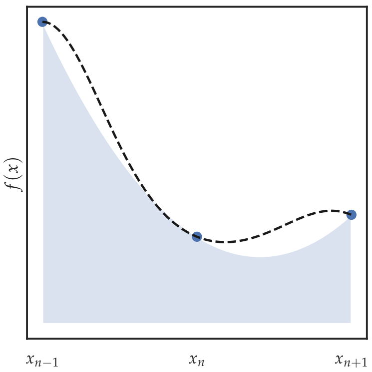
$h := |x_{n+1} - x_n|$ ⏩Distance between nodes

- Error $\mathcal{O}(h^2)$

- Full integral:

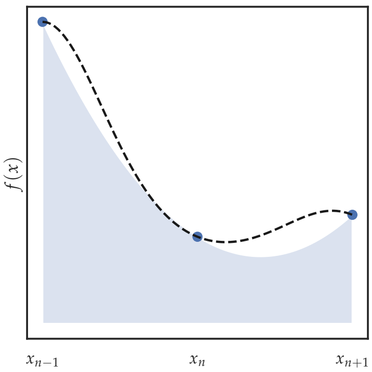
$$\int_a^b f(x)dx \approx \frac{h}{2}\big(f_0 + 2f_1 + \cdots + 2f_{N-1} + f_N\big), \quad f_n := f(x_n)$$

Simpson's rule



- Partition $[a, b]$ into N segments with equidistant nodes x_n
- **Locally quadratic approximation** of f connecting triplets $\big(f(x_{n-1}), f(x_n), f(x_{n+1})\big)$

Simpson's rule (2)

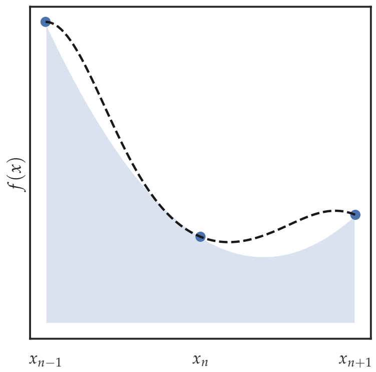


- Area of segment:

$$\int_{x_{n-1}}^{x_{n+1}} f(x)dx \approx \frac{h}{3}(f_{n-1} + 4f_n + f_{n+1})$$

$h := |x_{n+1} - x_n|$ ⏵⏵⏵Distance between nodes

Simpson's rule (2)



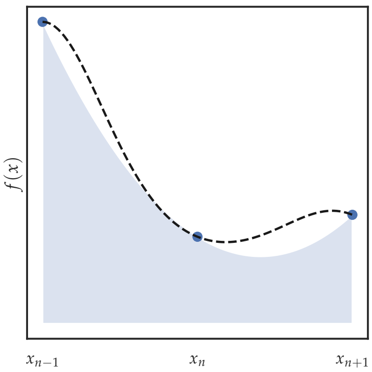
- Area of segment:

$$\int_{x_{n-1}}^{x_{n+1}} f(x)dx \approx \frac{h}{3}(f_{n-1} + 4f_n + f_{n+1})$$

$h := |x_{n+1} - x_n|$ ⏩Distance between nodes

- Error: $\mathcal{O}(h^4)$

Simpson's rule (2)



- Area of segment:

$$\int_{x_{n-1}}^{x_{n+1}} f(x)dx \approx \frac{h}{3}(f_{n-1} + 4f_n + f_{n+1})$$

$h := |x_{n+1} - x_n|$ ⏩Distance between nodes

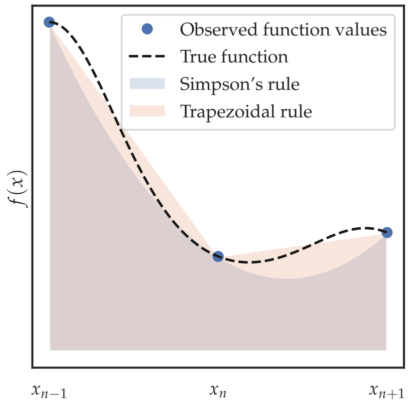
- Error: $\mathcal{O}(h^4)$

- Full integral:

$$\int_a^b f(x)dx \approx \frac{h}{3}(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 4f_{N-2} + 2f_{N-1} + f_N)$$

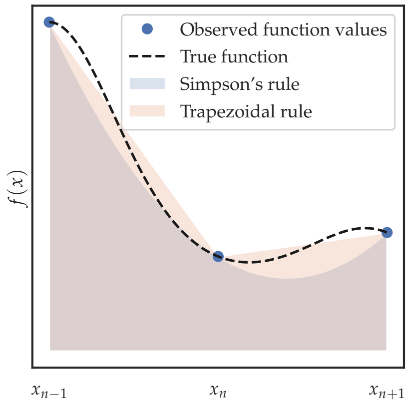
Example

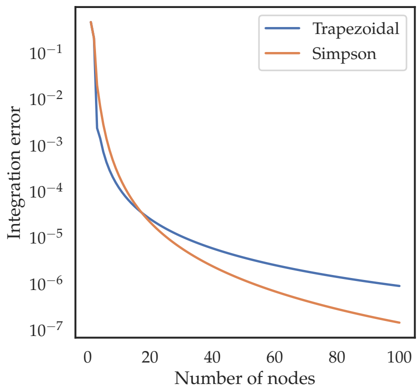
$$\int_0^1 \mathsf{exp}(-x^2 - \mathsf{sin}(3x)^2)dx$$



Example

$$\int_0^1 \exp(-x^2 - \sin(3x)^2)\,dx$$

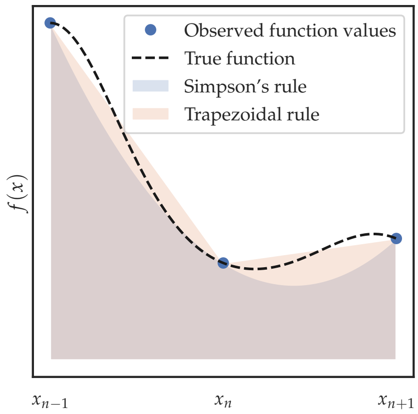




- Simpson's rule yields better approximations
- Very good approximations obtained fairly quickly

Summary: Newton–Cotes quadrature

- Approximate integrand between equidistant nodes with a low-degree polynomial (up to degree 6)
- Trapezoidal rule: linear interpolation
- Simpson's rule: quadratic interpolation
  ⏩ Better approximation and smaller integration error



Gaussian Quadrature

Gaussian quadrature

- Named after Carl Friedrich Gauß

Gaussian quadrature

- Named after Carl Friedrich Gauß
- Quadrature scheme that no longer relies on equidistant nodes ⏩ Higher accuracy

Gaussian quadrature

- Named after Carl Friedrich Gauß
- Quadrature scheme that no longer relies on equidistant nodes ⏩ Higher accuracy
- Central approximation

$$\int_a^b f(x)w(x)dx \approx \sum_{n=1}^{N} w_n f(x_n)$$

Gaussian quadrature

- Named after Carl Friedrich Gauß
- Quadrature scheme that no longer relies on equidistant nodes ⏩ Higher accuracy
- Central approximation

$$\int_a^b f(x)w(x)dx \approx \sum_{n=1}^{N} w_n f(x_n)$$

- Weight function $w(x) \geq 0$ (and some other integration-related properties, which are satisfied if $w(x)$ is a pdf)

Gaussian quadrature

- Named after Carl Friedrich Gauß
- Quadrature scheme that no longer relies on equidistant nodes ⏩ Higher accuracy
- Central approximation

$$\int_a^b f(x)w(x)dx \approx \sum_{n=1}^{N} w_n f(x_n)$$

- Weight function $w(x) \geq 0$ (and some other integration-related properties, which are satisfied if $w(x)$ is a pdf)
- Goal: Find nodes x_n and weights w_n, so that the approximation error is minimized

Central idea

- Quadrature nodes x_n are the roots of a family of **orthogonal polynomials**

Central idea

- Quadrature nodes x_n are the roots of a family of **orthogonal polynomials**
  - Nodes no longer equidistant

Central idea

- Quadrature nodes x_n are the roots of a family of **orthogonal polynomials**
  ⏵⏵⏵ Nodes no longer equidistant
- Exact if f is a polynomial of degree $\leq 2N-1$, i.e.,

$$\int_a^b f(x)w(x)dx = \sum_{n=1}^{N} w_n f(x_n)$$

  ⏵⏵⏵ Integral can be computed exactly by evaluating f N times at the optimal locations x_n (roots of an orthogonal polynomial) with corresponding optimal weights w_n

Central idea

- Quadrature nodes x_n are the roots of a family of **orthogonal polynomials**
  ⏵⏵⏵ Nodes no longer equidistant
- Exact if f is a polynomial of degree $\leq 2N-1$, i.e.,

$$\int_a^b f(x)w(x)dx = \sum_{n=1}^{N} w_n f(x_n)$$

⏵⏵⏵ Integral can be computed exactly by evaluating f N times at the optimal locations x_n (roots of an orthogonal polynomial) with corresponding optimal weights w_n
⏵⏵⏵ More accurate than Newton–Cotes for the same number of evaluations (with some memory overhead)

Example: Gauß–Hermite quadrature

- Solve

$$\int f(x)\,\underbrace{\exp(-x^2)}_{w(x)}\,dx = \int f(x)\sqrt{2\pi}\mathcal{N}\big(x\big|0,1\big)dx = \mathbb{E}_{x\sim\mathcal{N}(0,1)}[\sqrt{2\pi}f(x)]$$

Example: Gauß–Hermite quadrature

- Solve

$$\int f(x)\underbrace{\exp(-x^2)}_{w(x)}\,dx = \int f(x)\sqrt{2\pi}\mathcal{N}\big(x\big|0,1\big)dx = \mathbb{E}_{x\sim\mathcal{N}(0,1)}[\sqrt{2\pi}f(x)]$$

- With change-of-variables trick ⏩ Expectation w.r.t. a Gaussian measure

$$\mathbb{E}_{x\sim\mathcal{N}(\mu,\sigma^2)}[f(x)] \approx \frac{1}{\sqrt{\pi}}\sum_{n=1}^{N} w_n f(\sqrt{2}\sigma x_n + \mu).$$

Example: Gauß–Hermite quadrature (2)

- Follow general approximation scheme

$$\int f(x) \underbrace{\mathsf{exp}(-x^2)}_{w(x)} \, dx \approx \sum_{n=1}^{N} w_n f(x_n)$$

Example: Gauß–Hermite quadrature (2)

- Follow general approximation scheme

$$\int f(x) \underbrace{\exp(-x^2)}_{w(x)} dx \approx \sum_{n=1}^{N} w_n f(x_n)$$

- Nodes $x_1, \ldots, x_N$ are the roots of Hermite polynomial

$$H_N(x) := (-1)^n \exp\left(\frac{x^2}{2}\right) \frac{d^n}{dx^n} \exp(-x^2)$$

Example: Gauß–Hermite quadrature (2)

- Follow general approximation scheme

$$\int f(x) \underbrace{\exp(-x^2)}_{w(x)} dx \approx \sum_{n=1}^{N} w_n f(x_n)$$

- Nodes $x_1, \ldots, x_N$ are the roots of Hermite polynomial

$$H_N(x) := (-1)^n \exp\left(\frac{x^2}{2}\right) \frac{d^n}{dx^n} \exp(-x^2)$$

- Weights w_n are

$$w_n := \frac{2^{N-1} N! \sqrt{\pi}}{N^2 H_{N-1}^2(x_n)}$$

Overview (Stoer & Bulirsch, 2002)

$$\int_a^b w(x) f(x) dx \approx \sum_{n=1}^{N} w_n f(x_n)$$

$[a, b]$	$w(x)$	Orthogonal polynomial
$[-1, 1]$	1	Legendre polynomials
$[-1, 1]$	$(1 - x^2)^{-\frac{1}{2}}$	Chebychev polynomials
$[0, \infty]$	$\exp(-x)$	Laguerre polynomials
$[-\infty, \infty]$	$\exp(-x^2)$	Hermite polynomials

Application areas

- Probabilities for rectangular bivariate/trivariate Gaussian and t distributions (Genz, 2004)
- Integrating out (marginalizing) a few hyper-parameters in a latent-variable model (INLA; Rue et al., 2009)
- Predictions with a Gaussian process classifier (GPFlow; Matthews et al., 2017)

Summary: Gaussian quadrature

- Orthogonal polynomials to approximate f
- Nodes are the roots of the polynomial
- Higher accuracy than Newton–Cotes
- **Method of choice** for low-dimensional problems (1–3 dimensions)

Summary: Gaussian quadrature

- Orthogonal polynomials to approximate f
- Nodes are the roots of the polynomial
- Higher accuracy than Newton–Cotes
- **Method of choice** for low-dimensional problems (1–3 dimensions)
- Can't naturally deal with noisy observations
- Only works in low dimensions

Summary: Gaussian quadrature

- Orthogonal polynomials to approximate f
- Nodes are the roots of the polynomial
- Higher accuracy than Newton–Cotes
- **Method of choice** for low-dimensional problems (1–3 dimensions)
- Can't naturally deal with noisy observations
- Only works in low dimensions
- Approaches that scale better with dimensionality
  - **Bayesian quadrature** (up to ≈ 10 dimensions)
  - **Monte Carlo estimation** (high dimensions)

Bayesian Quadrature

Bayesian quadrature: Setting and key idea

$$Z := \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x}\sim p}[f(\boldsymbol{x})]$$

- Function f is expensive to evaluate
- Integration in moderate (≤ 10) dimensions
- Deal with noisy function observations

Bayesian quadrature: Setting and key idea

$$Z := \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x}\sim p}[f(\boldsymbol{x})]$$

- Function f is expensive to evaluate
- Integration in moderate (≤ 10) dimensions
- Deal with noisy function observations

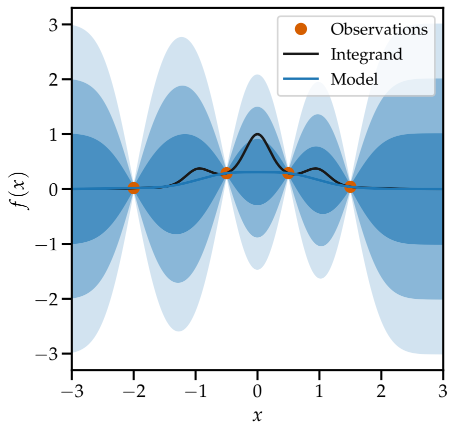
Key idea

- Phrase quadrature as a statistical inference problem
  ⏩ Probabilistic numerics (e.g., Hennig et al., 2015; Briol et al., 2015)
- Estimate distribution on Z using a dataset $\mathcal{D} := \{(\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_N, f(\boldsymbol{x}_N))\}$

Bayesian quadrature: How it works

$$Z := \int f(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x} \sim p}[f(\boldsymbol{x})]$$

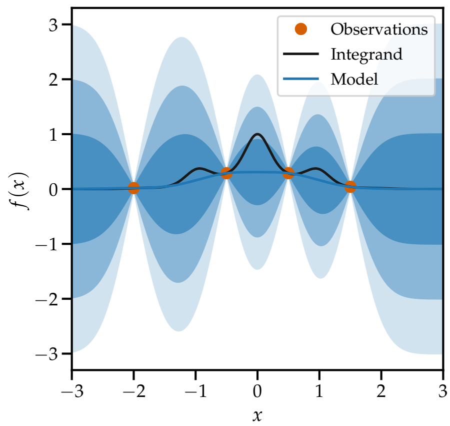
- Estimate distribution on Z using a dataset $\mathcal{D} := \{(\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_N, f(\boldsymbol{x}_N))\}$



Bayesian quadrature: How it works

$$Z := \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x}\sim p}[f(\boldsymbol{x})]$$

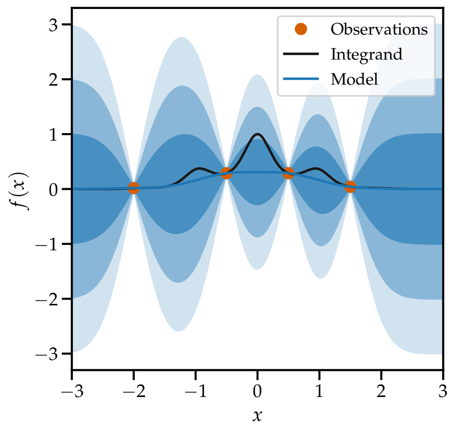
- Estimate distribution on Z using a dataset $\mathcal{D} := \{(\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \ldots, (\boldsymbol{x}_N, f(\boldsymbol{x}_N))\}$
- Place (Gaussian process) prior distribution on f and determine the posterior via Bayes' theorem (Diaconis 1988; O'Hagan 1991; Rasmussen & Ghahramani 2003)
  - Distribution on f induces a distribution on Z



Bayesian quadrature: How it works

$$Z := \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x}\sim p}[f(\boldsymbol{x})]$$

- Estimate distribution on Z using a dataset $\mathcal{D} := \{(\boldsymbol{x}_1, f(\boldsymbol{x}_1)), \dots, (\boldsymbol{x}_N, f(\boldsymbol{x}_N))\}$
- Place (Gaussian process) prior distribution on f and determine the posterior via Bayes' theorem (Diaconis 1988; O'Hagan 1991; Rasmussen & Ghahramani 2003)
  - Distribution on f induces a distribution on Z
- Generalizes to noisy function observations
$$y = f(\boldsymbol{x}) + \epsilon$$



Bayesian quadrature: Details

$$Z := \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}, \quad f \sim GP(0, k)$$

Bayesian quadrature: Details

$$Z := \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}, \quad f \sim GP(0, k)$$

- Exploit linearity of the integral (integral of a GP is another GP)

Bayesian quadrature: Details

$$Z := \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}, \quad f \sim GP(0, k)$$

- Exploit linearity of the integral (integral of a GP is another GP)

$$p(Z) = p\left(\int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}\right) = \mathcal{N}\big(Z \big| \mu_Z, \sigma_Z^2\big)$$

Bayesian quadrature: Details

$$Z := \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}, \quad f \sim GP(0, k)$$

- Exploit linearity of the integral (integral of a GP is another GP)

$$p(Z) = p\left(\int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}\right) = \mathcal{N}\big(Z \big| \mu_Z, \sigma_Z^2\big)$$

$$\mu_Z = \int \mu_{\text{post}}(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x}}[\mu_{\text{post}}(\boldsymbol{x})]$$

Bayesian quadrature: Details

$$Z := \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}, \quad f \sim GP(0, k)$$

- Exploit linearity of the integral (integral of a GP is another GP)

$$p(Z) = p\left(\int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}\right) = \mathcal{N}\left(Z \middle| \mu_Z, \sigma_Z^2\right)$$

$$\mu_Z = \int \mu_{\text{post}}(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x}}[\mu_{\text{post}}(\boldsymbol{x})]$$

$$\sigma_Z^2 = \iint k_{\text{post}}(\boldsymbol{x}, \boldsymbol{x}')p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' = \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}[k_{\text{post}}(\boldsymbol{x}, \boldsymbol{x}')]$$

Bayesian quadrature: Mean

$$\mathbb{E}_f[Z] = \mu_Z = \overbrace{\mathbb{E}_{\boldsymbol{x}\sim p}[\mu_{\text{post}}(\boldsymbol{x})]}^{\text{expected predictive mean}}$$

$$Z = \int f(\boldsymbol{x})p(\boldsymbol{x})d\boldsymbol{x}$$

$$f \sim GP(0, k)$$

$$p(Z) = \mathcal{N}\left(Z \middle| \mu_Z, \sigma_Z^2\right)$$

Training data: $\boldsymbol{X}, \boldsymbol{y}$

Bayesian quadrature: Mean

$$\mathbb{E}_f[Z] = \mu_Z = \overbrace{\mathbb{E}_{\boldsymbol{x}\sim p}[\mu_{\text{post}}(\boldsymbol{x})]}^{\text{expected predictive mean}}$$

$$\mu_{\text{post}}(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{X})\,\underbrace{\boldsymbol{K}^{-1}\boldsymbol{y}}_{=:\boldsymbol{\alpha}}, \quad \boldsymbol{K} := k(\boldsymbol{X}, \boldsymbol{X})$$

$$Z = \int f(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

$$f \sim GP(0, k)$$

$$p(Z) = \mathcal{N}\big(Z \big| \mu_Z, \sigma_Z^2\big)$$

Training data: $\boldsymbol{X}, \boldsymbol{y}$

Bayesian quadrature: Mean

$$\mathbb{E}_f[Z] = \mu_Z = \overbrace{\mathbb{E}_{\boldsymbol{x}\sim p}[\mu_{\text{post}}(\boldsymbol{x})]}^{\text{expected predictive mean}}$$

$$\mu_{\text{post}}(\boldsymbol{x}) = k(\boldsymbol{x}, \boldsymbol{X})\,\underbrace{\boldsymbol{K}^{-1}\boldsymbol{y}}_{=:\boldsymbol{\alpha}}, \quad \boldsymbol{K} := k(\boldsymbol{X}, \boldsymbol{X})$$

$$\mathbb{E}_f[Z] = \overbrace{\int k(\boldsymbol{x}, \boldsymbol{X}) p(\boldsymbol{x}) d\boldsymbol{x}}^{=:\boldsymbol{z}^\top}\,\boldsymbol{\alpha} = \boldsymbol{z}^\top \boldsymbol{\alpha}$$

$$z_n = \int k(\boldsymbol{x}, \boldsymbol{x}_n) p(\boldsymbol{x}) d\boldsymbol{x} = \mathbb{E}_{\boldsymbol{x}\sim p}[k(\boldsymbol{x}, \boldsymbol{x}_n)]$$

$$Z = \int f(\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{x}$$

$$f \sim GP(0, k)$$

$$p(Z) = \mathcal{N}\left(Z \middle| \mu_Z, \sigma_Z^2\right)$$

Training data: $\boldsymbol{X}, \boldsymbol{y}$

Bayesian quadrature: Variance

$$\mathbb{V}_f[Z] = \sigma_Z^2 = \overbrace{\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k_{\text{post}}(\boldsymbol{x},\boldsymbol{x}')]}^{\text{expected posterior covariance}}$$

Bayesian quadrature: Variance

$$\mathbb{V}_f[Z] = \sigma_Z^2 = \overbrace{\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k_{\text{post}}(\boldsymbol{x},\boldsymbol{x}')]}^{\text{expected posterior covariance}}$$
$$= \iint \underbrace{k(\boldsymbol{x},\boldsymbol{x}')}_{\text{prior covariance}} - \underbrace{k(\boldsymbol{x},\boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X},\boldsymbol{x}')}_{\text{information from training data}} p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}'$$

Bayesian quadrature: Variance

$$\mathbb{V}_f[Z] = \sigma_Z^2 = \overbrace{\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}' \sim p}[k_{\text{post}}(\boldsymbol{x},\boldsymbol{x}')]}^{\text{expected posterior covariance}}$$

$$= \iint \underbrace{k(\boldsymbol{x},\boldsymbol{x}')}_{\text{prior covariance}} - \underbrace{k(\boldsymbol{x},\boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X},\boldsymbol{x}')}_{\text{information from training data}} p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}'$$

$$= \iint k(\boldsymbol{x},\boldsymbol{x}')p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}'$$

$$= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}[k(\boldsymbol{x},\boldsymbol{x}')]$$

Bayesian quadrature: Variance

$$\mathbb{V}_f[Z] = \sigma_Z^2 = \overbrace{\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k_{\text{post}}(\boldsymbol{x},\boldsymbol{x}')]}^{\text{expected posterior covariance}}$$

$$= \iint \underbrace{k(\boldsymbol{x},\boldsymbol{x}')}_{\text{prior covariance}} - \underbrace{k(\boldsymbol{x},\boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X},\boldsymbol{x}')}_{\text{information from training data}} p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}'$$

$$= \iint k(\boldsymbol{x},\boldsymbol{x}')p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' - \underbrace{\int k(\boldsymbol{x},\boldsymbol{X})p(\boldsymbol{x})d\boldsymbol{x}}_{=\boldsymbol{z}^\top}$$

$$= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}[k(\boldsymbol{x},\boldsymbol{x}')] - \boldsymbol{z}^\top$$

Bayesian quadrature: Variance

$$\begin{aligned}
\mathbb{V}_f[Z] = \sigma_Z^2 = & \overbrace{\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k_{\text{post}}(\boldsymbol{x},\boldsymbol{x}')]}^{\text{expected posterior covariance}} \\
&= \iint \underbrace{k(\boldsymbol{x},\boldsymbol{x}')}_{\text{prior covariance}} - \underbrace{k(\boldsymbol{x},\boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X},\boldsymbol{x}')}_{\text{information from training data}} p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' \\
&= \iint k(\boldsymbol{x},\boldsymbol{x}')p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' - \underbrace{\int k(\boldsymbol{x},\boldsymbol{X})p(\boldsymbol{x})d\boldsymbol{x}}_{=\boldsymbol{z}^\top}\boldsymbol{K}^{-1} \\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}[k(\boldsymbol{x},\boldsymbol{x}')] - \boldsymbol{z}^\top\boldsymbol{K}^{-1}
\end{aligned}$$

Bayesian quadrature: Variance

$$\mathbb{V}_f[Z] = \sigma_Z^2 = \overbrace{\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k_{\text{post}}(\boldsymbol{x},\boldsymbol{x}')]}^{\text{expected posterior covariance}}$$

$$= \iint \underbrace{k(\boldsymbol{x},\boldsymbol{x}')}_{\text{prior covariance}} - \underbrace{k(\boldsymbol{x},\boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X},\boldsymbol{x}')}_{\text{information from training data}} p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}'$$

$$= \iint k(\boldsymbol{x},\boldsymbol{x}')p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' - \underbrace{\int k(\boldsymbol{x},\boldsymbol{X})p(\boldsymbol{x})d\boldsymbol{x}}_{=\boldsymbol{z}^\top}\boldsymbol{K}^{-1}\underbrace{\int k(\boldsymbol{X},\boldsymbol{x}')p(\boldsymbol{x}')d\boldsymbol{x}'}_{=\boldsymbol{z}'}$$

$$= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}[k(\boldsymbol{x},\boldsymbol{x}')] - \boldsymbol{z}^\top\boldsymbol{K}^{-1}\boldsymbol{z}'$$

Bayesian quadrature: Variance

$$
\begin{aligned}
\mathbb{V}_f[Z] = \sigma_Z^2 = &\overbrace{\mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k_{\text{post}}(\boldsymbol{x},\boldsymbol{x}')]}^{\text{expected posterior covariance}} \\
&= \iint \underbrace{k(\boldsymbol{x},\boldsymbol{x}')}_{\text{prior covariance}} - \underbrace{k(\boldsymbol{x},\boldsymbol{X})\boldsymbol{K}^{-1}k(\boldsymbol{X},\boldsymbol{x}')}_{\text{information from training data}} p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' \\
&= \iint k(\boldsymbol{x},\boldsymbol{x}')p(\boldsymbol{x})p(\boldsymbol{x}')d\boldsymbol{x}d\boldsymbol{x}' - \underbrace{\int k(\boldsymbol{x},\boldsymbol{X})p(\boldsymbol{x})d\boldsymbol{x}}_{=\boldsymbol{z}^\top}\boldsymbol{K}^{-1}\underbrace{\int k(\boldsymbol{X},\boldsymbol{x}')p(\boldsymbol{x}')d\boldsymbol{x}'}_{=\boldsymbol{z}'} \\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}[k(\boldsymbol{x},\boldsymbol{x}')] - \boldsymbol{z}^\top\boldsymbol{K}^{-1}\boldsymbol{z}' \\
&= \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'}[k(\boldsymbol{x},\boldsymbol{x}')] - \mathbb{E}_{\boldsymbol{x}}[k(\boldsymbol{x},\boldsymbol{X})]\boldsymbol{K}^{-1}\mathbb{E}_{\boldsymbol{x}'}[k(\boldsymbol{X},\boldsymbol{x}')]
\end{aligned}
$$

Computing kernel expectations

$$\mathbb{E}_{\boldsymbol{x}\sim p}[k(\boldsymbol{x}, \boldsymbol{X})], \quad \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k(\boldsymbol{x}, \boldsymbol{x}')]$$

- Solve a different (easier) integration problem

Computing kernel expectations

$$\mathbb{E}_{\boldsymbol{x}\sim p}[k(\boldsymbol{x}, \boldsymbol{X})], \quad \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k(\boldsymbol{x}, \boldsymbol{x}')]$$

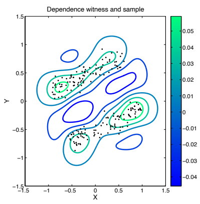
- Solve a different (easier) integration problem

	Input distribution p	
Kernel k	Gaussian	non-Gaussian
RBF/ polynomial/ trigonometric	analytical	analytical via importance-sampling trick
otherwise	Monte Carlo (numerical integration)	Monte Carlo (numerical integration)

Kernel expectations in other areas

$$\mathbb{E}_{\boldsymbol{x}\sim p}[k(\boldsymbol{x}, \boldsymbol{X})], \quad \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k(\boldsymbol{x}, \boldsymbol{x}')]$$

- Kernel MMD
  (e.g., Gretton et al., 2012)

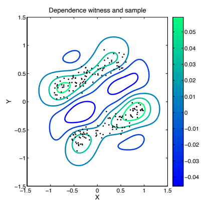


from Gretton et al. (2012)

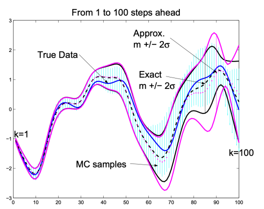
Kernel expectations in other areas

$$\mathbb{E}_{\boldsymbol{x}\sim p}[k(\boldsymbol{x}, \boldsymbol{X})], \quad \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k(\boldsymbol{x}, \boldsymbol{x}')]$$

- Kernel MMD
  (e.g., Gretton et al., 2012)
- Time-series analysis with Gaussian processes
  (e.g., Girard et al., 2003)



from Gretton et al. (2012)

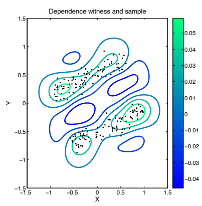


from Girard et al. (2003)

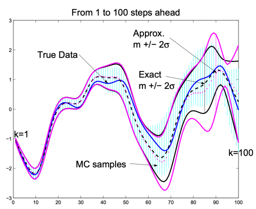
Kernel expectations in other areas

$$\mathbb{E}_{\boldsymbol{x}\sim p}[k(\boldsymbol{x}, \boldsymbol{X})], \quad \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k(\boldsymbol{x}, \boldsymbol{x}')]$$

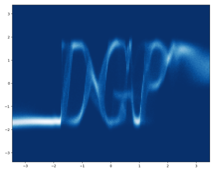
- Kernel MMD
  (e.g., Gretton et al., 2012)
- Time-series analysis with Gaussian processes
  (e.g., Girard et al., 2003)
- Deep Gaussian processes
  (e.g., Damianou & Lawrence, 2013)



from Gretton et al. (2012)



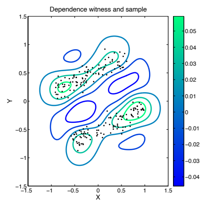
from Girard et al. (2003)


from Salimbeni et al. (2019)

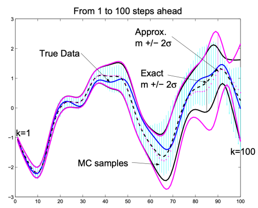
Kernel expectations in other areas

$$\mathbb{E}_{\boldsymbol{x}\sim p}[k(\boldsymbol{x},\boldsymbol{X})], \quad \mathbb{E}_{\boldsymbol{x},\boldsymbol{x}'\sim p}[k(\boldsymbol{x},\boldsymbol{x}')]$$

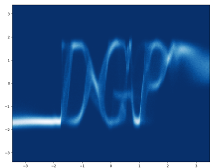
- Kernel MMD
  (e.g., Gretton et al., 2012)
- Time-series analysis with Gaussian processes
  (e.g., Girard et al., 2003)
- Deep Gaussian processes
  (e.g., Damianou & Lawrence, 2013)
- Model-based RL with Gaussian processes
  (e.g., Deisenroth & Rasmussen, 2011)



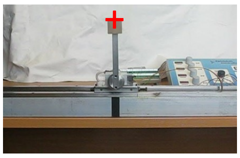
from Gretton et al. (2012)



from Girard et al. (2003)



from Salimbeni et al. (2019)



from Deisenroth & Rasmussen (2011)

Iterative procedure: Where to measure f next?

- Define an **acquisition function** (similar to Bayesian optimization)

Iterative procedure: Where to measure f next?

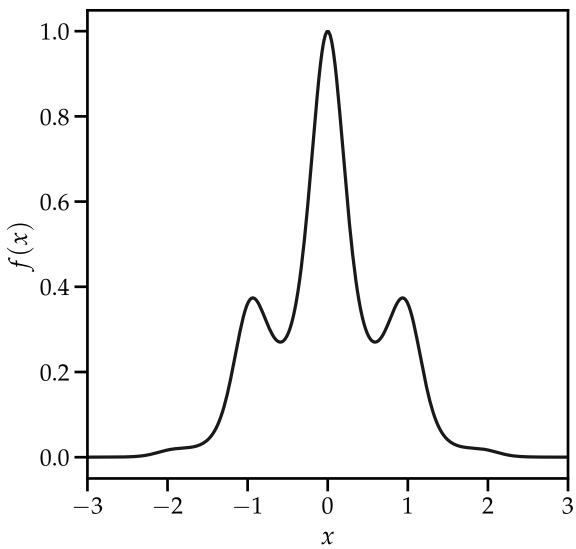
- Define an **acquisition function** (similar to Bayesian optimization)
- Example: Choose next node $\boldsymbol{x}_{n+1}$ so that the variance of the estimator is reduced maximally (e.g., O'Hagan, 1991; Gunter et al., 2014)

$$\boldsymbol{x}_{n+1} = \operatorname{argmax}_{\boldsymbol{x}_*} \overbrace{\mathbb{V}[Z|\mathcal{D}]}^{\text{current variance}} - \overbrace{\mathbb{E}_{y_*}\left[\mathbb{V}[Z|\mathcal{D}\cup\{(\boldsymbol{x}_*, y_*)\}]\right]}^{\text{new variance}}$$

Example with EmuKit (Paleyes et al., 2019)

Compute

$$Z = \int_{-3}^{3} e^{-x^2 - \sin^2(3x)} dx$$

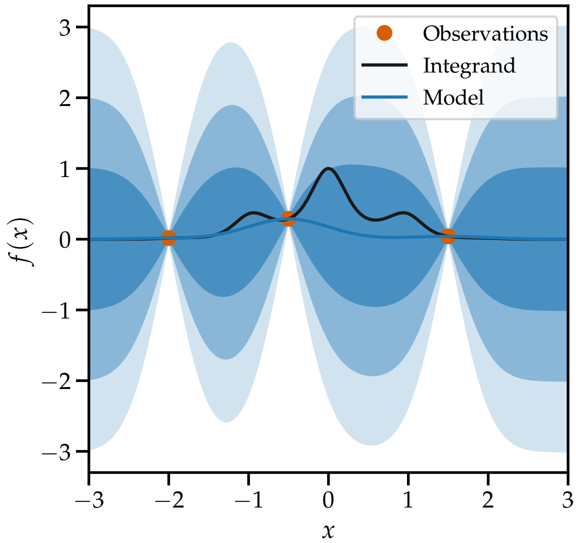


Example with EmuKit (Paleyes et al., 2019)

Compute

$$Z = \int_{-3}^{3} e^{-x^2 - \sin^2(3x)} dx$$

- Fit Gaussian process to observations $f(x_1), \ldots, f(x_n)$ at nodes $x_1, \ldots, x_n$

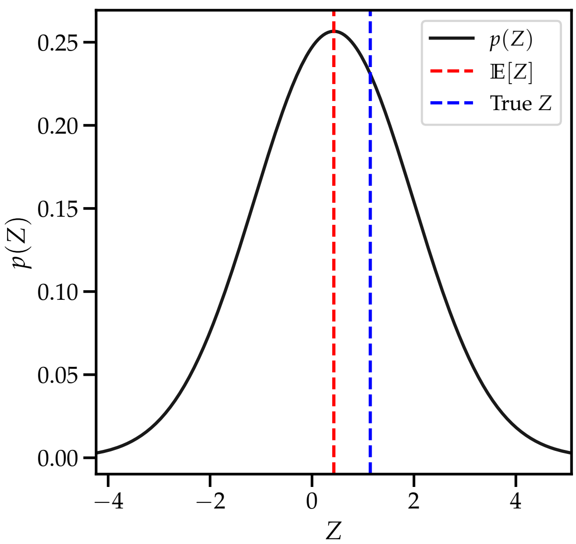


Example with EmuKit (Paleyes et al., 2019)

Compute

$$Z = \int_{-3}^{3} e^{-x^2 - \sin^2(3x)} dx$$

- Fit Gaussian process to observations $f(x_1), \ldots, f(x_n)$ at nodes $x_1, \ldots, x_n$
- Determine $p(Z)$

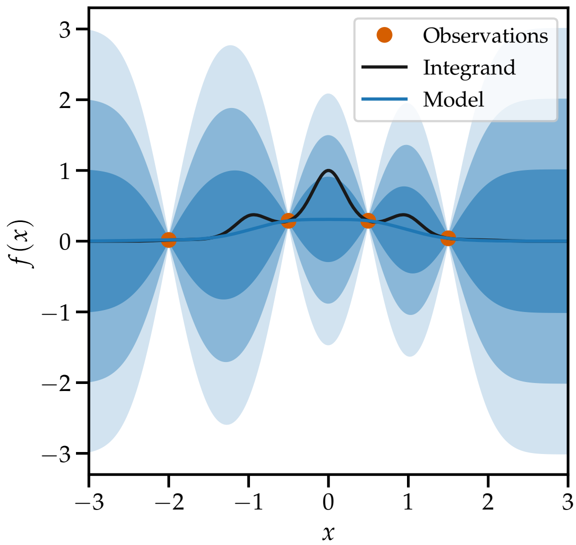


Example with EmuKit (Paleyes et al., 2019)

Compute

$$Z = \int_{-3}^{3} e^{-x^2 - \sin^2(3x)} dx$$

- Fit Gaussian process to observations $f(x_1), \dots, f(x_n)$ at nodes $x_1, \dots, x_n$
- Determine $p(Z)$
- Find and include new measurement
  1. Find optimal node x_{n+1} by maximizing an acquisition function
  2. Evaluate integrand at x_{n+1}
  3. Update GP with $\big(x_{n+1}, f(x_{n+1})\big)$

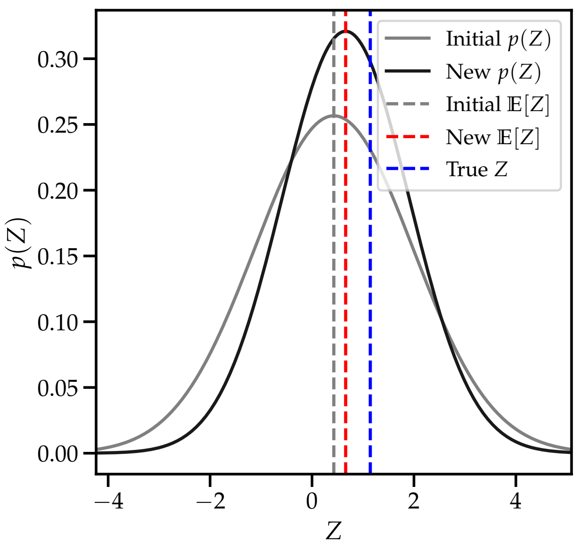


Example with EmuKit (Paleyes et al., 2019)

Compute

$$Z = \int_{-3}^{3} e^{-x^2 - \sin^2(3x)} dx$$

- Fit Gaussian process to observations $f(x_1), \ldots, f(x_n)$ at nodes $x_1, \ldots, x_n$
- Determine $p(Z)$
- Find and include new measurement
- Compute updated $p(Z)$

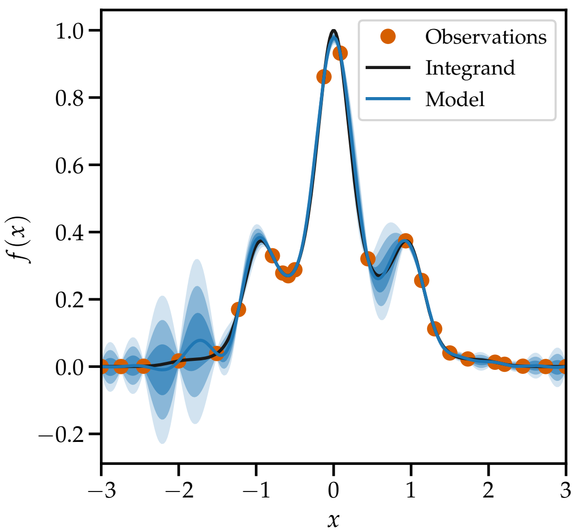


Example with EmuKit (Paleyes et al., 2019)

Compute

$$Z = \int_{-3}^{3} e^{-x^2 - \sin^2(3x)} dx$$

- Fit Gaussian process to observations $f(x_1), \ldots, f(x_n)$ at nodes $x_1, \ldots, x_n$
- Determine $p(Z)$
- Find and include new measurement
- Compute updated $p(Z)$
- Repeat

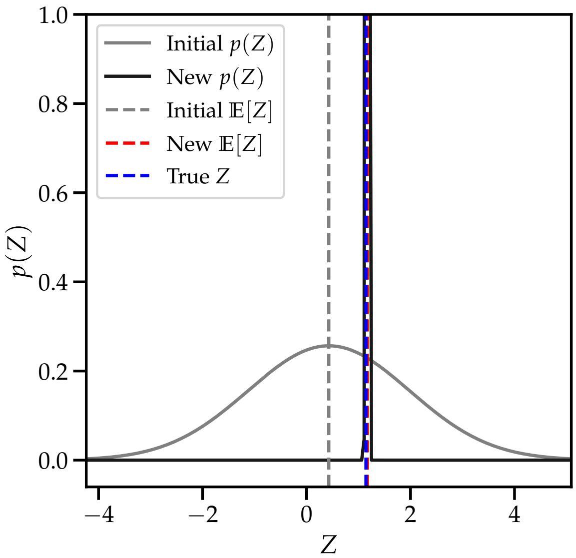


Example with EmuKit (Paleyes et al., 2019)

Compute

$$Z = \int_{-3}^{3} e^{-x^2 - \sin^2(3x)} dx$$

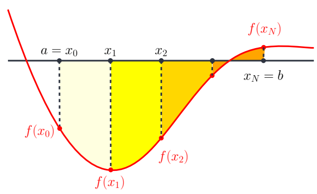
- Fit Gaussian process to observations $f(x_1), \ldots, f(x_n)$ at nodes $x_1, \ldots, x_n$
- Determine $p(Z)$
- Find and include new measurement
- Compute updated $p(Z)$
- Repeat



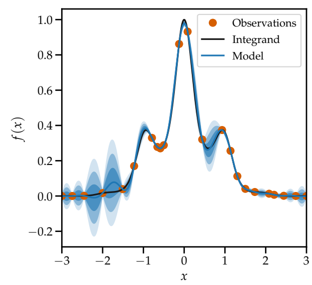
Summary

- Central approximation

$$\int f(\boldsymbol{x})d\boldsymbol{x} \approx \sum_{n=1}^{N} w_n f(\boldsymbol{x}_n)$$



- **Newton–Cotes:** Equidistant nodes $\boldsymbol{x}_n$, low-degree polynomial approximation of f
- **Gaussian quadrature:** Nodes $\boldsymbol{x}_n$ as the roots of interpolating orthogonal polynomials of f
- **Bayesian quadrature:** Integration as a statistical inference problem; Global approximation of f using a Gaussian process; scales to moderate dimensions



Numerical integration is a really good idea in low dimensions

References

Briol, F.-X., Oates, C., Girolami, M., and Osborne, M. A. (2015). Frank–Wolfe Bayesian Quadrature: Probabilistic Integration with Theoretical Guarantees. In *Advances in Neural Information Processing Systems*.

Cutler, M. and How, J. P. (2015). Efficient Reinforcement Learning for Robots using Informative Simulated Priors. In *Proceedings of the International Conference on Robotics and Automation*.

Damianou, A. and Lawrence, N. D. (2013). Deep Gaussian Processes. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*.

Deisenroth, M. P., Fox, D., and Rasmussen, C. E. (2015). Gaussian Processes for Data-Efficient Learning in Robotics and Control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(2):408–423.

Deisenroth, M. P. and Mohamed, S. (2012). Expectation Propagation in Gaussian Process Dynamical Systems. In *Advances in Neural Information Processing Systems*, pages 2618–2626.

References (cont.)

Deisenroth, M. P. and Rasmussen, C. E. (2011). PILCO: A Model-Based and Data-Efficient Approach to Policy Search. In *Proceedings of the International Conference on Machine Learning*.

Deisenroth, M. P., Turner, R., Huber, M., Hanebeck, U. D., and Rasmussen, C. E. (2012). Robust Filtering and Smoothing with Gaussian Processes. *IEEE Transactions on Automatic Control*, 57(7):1865–1871.

Diaconis, P. (1988). Bayesian Numerical Analysis. *Statistical Decision Theory and Related Topics IV*, 1:163–175.

Eleftheriadis, S., Nicholson, T. F. W., Deisenroth, M. P., and Hensman, J. (2017). Identification of Gaussian Process State Space Models. In *Advances in Neural Information Processing Systems*.

Genz, A. (2004). Numerical Computation of Rectangular Bivariate and Trivariate Normal and t Probabilities. *Statistics and Computing*, 14:251–260.

References (cont.)

Girard, A., Rasmussen, C. E., Quiñonero Candela, J., and Murray-Smith, R. (2003). Gaussian Process Priors with Uncertain Inputs—Application to Multiple-Step Ahead Time Series Forecasting. In *Advances in Neural Information Processing Systems*.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A Kernel Two-Sample Test. *Journal of Machine Learning Research*, 13(25):723–773.

Gunter, T., Osborne, M. A., Garnett, R., Hennig, P., and Roberts, S. J. (2014). Sampling for Inference in Probabilistic Models with Fast Bayesian Quadrature. In *Advances in Neural Information Processing Systems*.

Hennig, P., Osborne, M. A., and Girolami, M. (2015). Probabilistic numerics and uncertainty in computations. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 471:20150142.

Ko, J. and Fox, D. (2009). GP-BayesFilters: Bayesian Filtering using Gaussian Process Prediction and Observation Models. *Autonomous Robots*, 27(1):75–90.

References (cont.)

O'Hagan, A. (1991). Bayes-Hermite Quadrature. *Journal of Statistical Planning and Inference*, 29:245–260.

Paleyes, A., Pullin, M., Mahsereci, M., Lawrence, N., and González, J. (2019). Emulation of Physical Processes with Emukit. In *Second Workshop on Machine Learning and the Physical Sciences, NeurIPS*.

Salimbeni, H. and Deisenroth, M. P. (2017). Doubly Stochastic Variational Inference for Deep Gaussian Processes. In *Advances in Neural Information Processing Systems*.

Salimbeni, H., Dutordoir, V., Hensman, J., and Deisenroth, M. P. (2019). Deep Gaussian Processes with Importance-Weighted Variational Inference. In *Proceedings of the International Conference on Machine Learning*.

Stoer, J. and Bulirsch, R. (2002). *Introduction to Numerical Analysis*. Texts in Applied Mathematics. Springer-Verlag, 3rd edition.