# Bayesian Quadrature for Multiple Related Integrals

François-Xavier Briol

University of Warwick (Department of Statistics)
Imperial College London (Department of Mathematics)

PN Conference

# Collaborators

Xiaoyue Xi
(Warwick)

Mark Girolami
(Imperial & ATI)

Xi, X., Briol, F.-X., & Girolami, M. (2018). Bayesian Quadrature for Multiple Related Integrals. arXiv:1801.04153.

# Bayesian Quadrature

- The goal of numerical integration is to find an approximation of the integral of some function $f : \mathcal{X} \to \mathbb{R}$ ($\mathbb{R}^p, p \in \mathbb{N}$) against some measure $\Pi$.
- This is often done use a quadrature/cubature rule:

$$\int_{\mathcal{X}} f(x) \mathrm{d}\Pi(x) \approx \sum_{i=1}^{n} w_i f(x_i)$$

- Bayesian Quadrature provides a way of getting good weights. It is based on building a GP interpolant and looking at the push-forward through the integral operator.

# Bayesian Quadrature

- The goal of numerical integration is to find an approximation of the integral of some function $f : \mathcal{X} \to \mathbb{R}$ ($\mathbb{R}^p, p \in \mathbb{N}$) against some measure $\Pi$.
- This is often done use a quadrature/cubature rule:

$$\int_{\mathcal{X}} f(x) \mathrm{d}\Pi(x) \approx \sum_{i=1}^{n} w_i f(x_i)$$

- Bayesian Quadrature provides a way of getting good weights. It is based on building a GP interpolant and looking at the push-forward through the integral operator.

# Bayesian Quadrature

- The goal of numerical integration is to find an approximation of the integral of some function $f : \mathcal{X} \to \mathbb{R}$ ($\mathbb{R}^p, p \in \mathbb{N}$) against some measure $\Pi$.

- This is often done use a quadrature/cubature rule:

$$\int_{\mathcal{X}} f(x) \mathrm{d}\Pi(x) \approx \sum_{i=1}^{n} w_i f(x_i)$$

- Bayesian Quadrature provides a way of getting good weights. It is based on building a GP interpolant and looking at the push-forward through the integral operator.

# What About Multiple Integrals?

- But what if we instead have a sequence of such functions $f_1, f_2 \ldots, f_D$?? If we know something about how $f_1$ relates to $f_2$ (etc...) then we might be able to use that information in the design of a numerical integration method.
- It might make more sense to approximate the integral with a quadrature rule of the form:

$$\hat{\Pi}[f_d] = \sum_{d'=1}^{D} \sum_{i=1}^{N} (\boldsymbol{W}_i)_{dd'} f_{d'}(x_{d'i}).$$

# What About Multiple Integrals?

- But what if we instead have a sequence of such functions $f_1, f_2 \ldots, f_D$?? If we know something about how $f_1$ relates to $f_2$ (etc...) then we might be able to use that information in the design of a numerical integration method.
- It might make more sense to approximate the integral with a quadrature rule of the form:

$$\hat{\Pi}[f_d] \;=\; \sum_{d'=1}^{D} \sum_{i=1}^{N} (\boldsymbol{W}_i)_{dd'} f_{d'}(x_{d'i}).$$

# Bayesian Quadrature for Multiple Related Functions

- We can use the same type of results for Gaussian Processes on the extended space of vector-valued functions $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^D$ (rather than $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}$) where $\boldsymbol{f}(x) = (f_1(x), \ldots, f_D(x))$.

- This approach allow us to directly encode the relation between each function $f_i$ by specifying the kernel $K$.

- In this case the posterior distribution is a $\mathcal{GP}(\boldsymbol{m}_n, \boldsymbol{K}_n)$ with vector-valued mean $\boldsymbol{m}_n : \mathcal{X} \to \mathbb{R}^D$ and matrix-valued covariance $\boldsymbol{K}_n : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{D \times D}$:

$$
\begin{aligned}
\boldsymbol{m}_n(\boldsymbol{x}) &= \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{f}(\boldsymbol{X}) \\
\boldsymbol{K}_n(\boldsymbol{x}, \boldsymbol{x}') &= \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x}').
\end{aligned}
$$

The overall cost for computing this is $\mathcal{O}(n^3 D^3)$.

# Bayesian Quadrature for Multiple Related Functions

- We can use the same type of results for Gaussian Processes on the extended space of vector-valued functions $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^D$ (rather than $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}$) where $\boldsymbol{f}(x) = (f_1(x), \ldots, f_D(x))$.
- This approach allow us to directly encode the relation between each function $f_i$ by specifying the kernel $K$.
- In this case the posterior distribution is a $\mathcal{GP}(\boldsymbol{m}_n, \boldsymbol{K}_n)$ with vector-valued mean $\boldsymbol{m}_n : \mathcal{X} \to \mathbb{R}^D$ and matrix-valued covariance $\boldsymbol{K}_n : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{D \times D}$:

$$\begin{aligned} \boldsymbol{m}_n(\boldsymbol{x}) &= \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{f}(\boldsymbol{X}) \\ \boldsymbol{K}_n(\boldsymbol{x}, \boldsymbol{x}') &= \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x}'). \end{aligned}$$

The overall cost for computing this is $\mathcal{O}(n^3 D^3)$.

# Bayesian Quadrature for Multiple Related Functions

- We can use the same type of results for Gaussian Processes on the extended space of vector-valued functions $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^D$ (rather than $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}$) where $\boldsymbol{f}(x) = (f_1(x), \ldots, f_D(x))$.
- This approach allow us to directly encode the relation between each function $f_i$ by specifying the kernel $K$.
- In this case the posterior distribution is a $\mathcal{GP}(\boldsymbol{m}_n, \boldsymbol{K}_n)$ with vector-valued mean $\boldsymbol{m}_n : \mathcal{X} \to \mathbb{R}^D$ and matrix-valued covariance $\boldsymbol{K}_n : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^{D \times D}$:

$$\begin{aligned} \boldsymbol{m}_n(\boldsymbol{x}) &= \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{f}(\boldsymbol{X}) \\ \boldsymbol{K}_n(\boldsymbol{x}, \boldsymbol{x}') &= \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{K}(\boldsymbol{x}, \boldsymbol{X})\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{x}'). \end{aligned}$$

The overall cost for computing this is $\mathcal{O}(n^3 D^3)$.

- Consider multi-output Bayesian Quadrature with a $\mathcal{GP}(\mathbf{0}, \boldsymbol{K})$ prior on $\boldsymbol{f} = (f_1, \ldots, f_D)^\top$. The posterior distribution on $\Pi[\boldsymbol{f}]$ is a $D$-dimensional Gaussian with mean and covariance matrix:

$$\begin{aligned} \mathbb{E}_N[\Pi[\boldsymbol{f}]] &= \Pi[\boldsymbol{K}(\cdot, \boldsymbol{X})]\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{f}(\boldsymbol{X}) \\ \mathbb{V}_N[\Pi[\boldsymbol{f}]] &= \Pi\bar{\Pi}[\boldsymbol{K}] - \Pi[\boldsymbol{K}(\cdot, \boldsymbol{X})]\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\bar{\Pi}[\boldsymbol{K}(\boldsymbol{X}, \cdot)] \end{aligned}$$

- Kernel evaluations are now matrix-valued (i.e. in $\mathbb{R}^{D \times D}$) as opposed to scalar-valued. A simple example is the following separable kernel:

$$\boldsymbol{K}(x, x') = \boldsymbol{B}k(x, x')$$

  $B$ encodes the covariance between function, and $k$ the type of function in each of the components.

- This reduces the cost to $\mathcal{O}(n^3 + D^3)$. We also can show that the convergence rate will be the same as for the uni-output BQ rule based on $k$ (and we can extend the theory to the misspecied setting).

- Consider multi-output Bayesian Quadrature with a $\mathcal{GP}(\mathbf{0}, \boldsymbol{K})$ prior on $\boldsymbol{f} = (f_1, \ldots, f_D)^\top$. The posterior distribution on $\Pi[\boldsymbol{f}]$ is a $D$-dimensional Gaussian with mean and covariance matrix:

$$\begin{aligned}\mathbb{E}_N\left[\Pi[\boldsymbol{f}]\right] &= \Pi[\boldsymbol{K}(\cdot, \boldsymbol{X})]\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\boldsymbol{f}(\boldsymbol{X}) \\ \mathbb{V}_N\left[\Pi[\boldsymbol{f}]\right] &= \Pi\bar{\Pi}\left[\boldsymbol{K}\right] - \Pi[\boldsymbol{K}(\cdot, \boldsymbol{X})]\boldsymbol{K}(\boldsymbol{X}, \boldsymbol{X})^{-1}\bar{\Pi}[\boldsymbol{K}(\boldsymbol{X}, \cdot)]\end{aligned}$$

- Kernel evaluations are now matrix-valued (i.e. in $\mathbb{R}^{D\times D}$) as opposed to scalar-valued. A simple example is the following separable kernel:

$$\boldsymbol{K}(x, x') = \boldsymbol{B}k(x, x')$$

  $B$ encodes the covariance between function, and $k$ the type of function in each of the components.

- This reduces the cost to $\mathcal{O}(n^3 + D^3)$. We also can show that the convergence rate will be the same as for the uni-output BQ rule based on $k$ (and we can extend the theory to the misspecied setting).

- Consider multi-output Bayesian Quadrature with a $\mathcal{GP}(\mathbf{0}, \boldsymbol{K})$ prior on $\boldsymbol{f} = (f_1, \ldots, f_D)^\top$. The posterior distribution on $\Pi[\boldsymbol{f}]$ is a $D$-dimensional Gaussian with mean and covariance matrix:

$$\begin{aligned}\mathbb{E}_N\left[\Pi[\boldsymbol{f}]\right] &= \Pi[\boldsymbol{K}(\cdot,\boldsymbol{X})]\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})^{-1}\boldsymbol{f}(\boldsymbol{X}) \\ \mathbb{V}_N\left[\Pi[\boldsymbol{f}]\right] &= \Pi\bar{\Pi}\left[\boldsymbol{K}\right] - \Pi[\boldsymbol{K}(\cdot,\boldsymbol{X})]\boldsymbol{K}(\boldsymbol{X},\boldsymbol{X})^{-1}\bar{\Pi}[\boldsymbol{K}(\boldsymbol{X},\cdot)]\end{aligned}$$

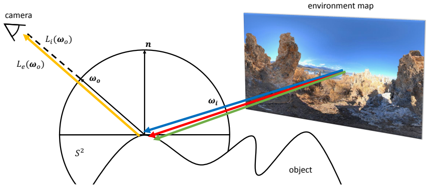
- Kernel evaluations are now matrix-valued (i.e. in $\mathbb{R}^{D\times D}$) as opposed to scalar-valued. A simple example is the following separable kernel:

$$\boldsymbol{K}(x,x') = \boldsymbol{B}k(x,x')$$

$B$ encodes the covariance between function, and $k$ the type of function in each of the components.

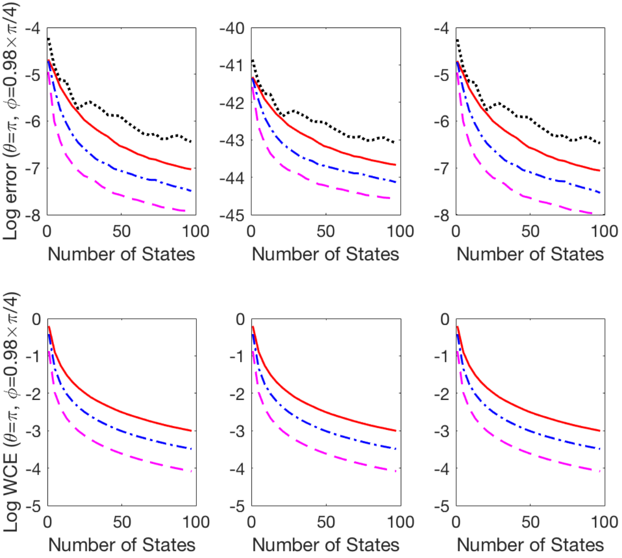
- This reduces the cost to $\mathcal{O}(n^3 + D^3)$. We also can show that the convergence rate will be the same as for the uni-output BQ rule based on $k$ (and we can extend the theory to the misspecied setting).

# Multi-output BQ for Global Illumination




- We compute integrals for different integrands based on various angles $\omega_0$ (akin to a camera moving).
- We pick a separable kernel $\boldsymbol{K}(x, x') = \boldsymbol{B}k(x, x')$ where $\boldsymbol{B}$ is chosen to represent the angle between integrands and $k(\boldsymbol{x}, \boldsymbol{x}') = \frac{8}{3} - \|\boldsymbol{x} - \boldsymbol{x}'\|_2$.
- We can prove that the worst-case integration error converges at a rate $\mathcal{O}(n^{-\frac{3}{4}})$ for each integrand. This is the same rate as uni-output BQ (but we usually improve on constants).

# Multi-output BQ for the Computer Graphics Example



# The Broader Picture

**Short summary:** The multi-output Bayesian Quadrature algorithm is an example of how joint-models can lead to significant improvements in performance.

**Interesting Question:** Can this idea, known as transfer learning in the ML literature, be useful in other areas of probabilistic numerics?

**Minor Question:** Does anyone know of any other literature on the topic of integrating multiple related functions?

Xi, X., Briol, F.-X., & Girolami, M. (2018). Bayesian Quadrature for Multiple Related Integrals. arXiv:1801.04153.

# The Broader Picture

**Short summary:** The multi-output Bayesian Quadrature algorithm is an example of how joint-models can lead to significant improvements in performance.

**Interesting Question:** Can this idea, known as transfer learning in the ML literature, be useful in other areas of probabilistic numerics?

**Minor Question:** Does anyone know of any other literature on the topic of integrating multiple related functions?

Xi, X., Briol, F.-X., & Girolami, M. (2018). Bayesian Quadrature for Multiple Related Integrals. arXiv:1801.04153.

# The Broader Picture

**Short summary:** The multi-output Bayesian Quadrature algorithm is an example of how joint-models can lead to significant improvements in performance.

**Interesting Question:** Can this idea, known as transfer learning in the ML literature, be useful in other areas of probabilistic numerics?

**Minor Question:** Does anyone know of any other literature on the topic of integrating multiple related functions?

Xi, X., Briol, F.-X., & Girolami, M. (2018). Bayesian Quadrature for Multiple Related Integrals. arXiv:1801.04153.

# The Broader Picture

**Short summary:** The multi-output Bayesian Quadrature algorithm is an example of how joint-models can lead to significant improvements in performance.

**Interesting Question:** Can this idea, known as transfer learning in the ML literature, be useful in other areas of probabilistic numerics?

**Minor Question:** Does anyone know of any other literature on the topic of integrating multiple related functions?

Xi, X., Briol, F.-X., & Girolami, M. (2018). Bayesian Quadrature for Multiple Related Integrals. arXiv:1801.04153.