Fast Gaussian Process Regression with Derivative Information

SIAM UQ 2024

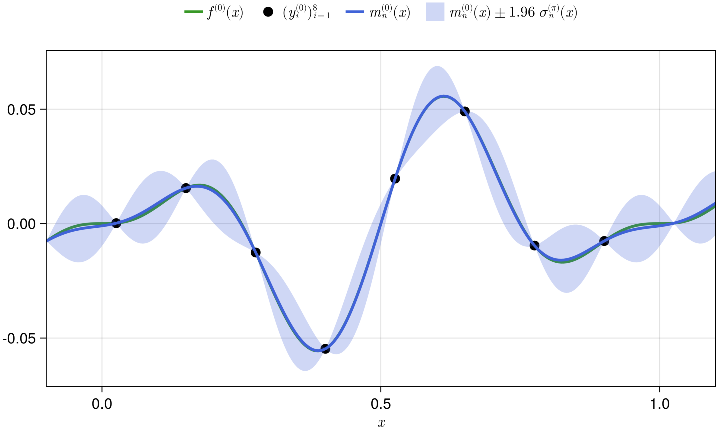
Aleksei G. Sorokin & Fred J. Hickernell

Illinois Institute of Technology, Department of Applied Mathematics

February 27, 2024

Why Gaussian Process Regression (GPR)?

- Encode simulation knowledge into model via kernel e.g. smoothness or periodicity
- Provides a distribution over simulations i.e. quantifies uncertainty in predictions



Motivation

Challenge: A GPR model costs $\mathcal{O}(n^3)$ to fit

- Applications often require GPR for $n > 10,000$ nodes which is very costly

Solution: Matching nodes and kernel reduces costs to $\mathcal{O}(n \log n)$

- ⋆ Require control over design of experiments

Observation: Derivative information can enhance GPR

- Derivatives available for free e.g. simulation is the numerical solution of a PDE
- Derivatives may be available at a nominal cost e.g. via automatic differentiation
- Derivatives may be the primary information source e.g. GPR for solving non-linear PDEs [Chen et al., 2021]

New Challenge: With m derivatives, can we improve the $\mathcal{O}(n^3 m^3)$ fitting cost?
New Solution: Exploit additional structure to reduce cost to $\mathcal{O}(m^2 n \log n + m^3 n)$

Outline and Related Work

1. **GPR**: follows book on GP for Machine Learning [Rasmussen et al., 2006]
2. **Fast GPR**: follows fast Bayesian cubature of Hickernell and Jagadeeswaran
   - Flavor #1: lattice sequence designs [Jagadeeswaran and Hickernell, 2019]
   - Flavor #2: digital sequence designs [Jagadeeswaran and Hickernell, 2022]
   - Unifying thesis of Jagadeeswaran Rathinavel [Rathinavel, 2019]
   - Adjacent work in a RKHS with lattice sequences [Kaarnioja et al., 2022]
   - Application to surrogate for PDE with random coefficients [Sorokin et al., 2023]
3. **GPR with derivative information**: incorporated gradients in [Solak et al., 2002]
4. **Fast GPR with derivative information**: our novel contribution!

Gaussian Process Regression

- Given *simulation* $f : [0,1]^s \to \mathbb{R}$
- Assume simulation an instance of a *Gaussian process*, $f \sim GP(0,K)$
  - Assume prior mean is zero (not necessary but simplifies presentation)
  - *Prior covariance kernel* $K : [0,1]^s \times [0,1]^s \to \mathbb{R}$ is symmetric positive definite

$$K(\boldsymbol{x},\boldsymbol{x}') = \mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')]$$

- *Sampling sequence* $\mathsf{X} = (\boldsymbol{x}_i)_{i=1}^n \in [0,1]^{n \times s}$
- *Observations* $\boldsymbol{y} = (y_i)_{i=1}^n = (f(\boldsymbol{x}_i) + \varepsilon_i)_{i=1}^n \in \mathbb{R}^{n \times 1}$ with *noise* $\varepsilon_1, \dots, \varepsilon_n \overset{\mathrm{IID}}{\sim} \mathcal{N}(0,\zeta)$
- *kernel (Gram) matrix* $\mathsf{K} = (K(\boldsymbol{x}_i,\boldsymbol{x}_j))_{i,j=1}^n \in \mathbb{R}^{n \times n}$
- *kernel vector* $\boldsymbol{k}_{\mathsf{X}}(\boldsymbol{x}) = (K(\boldsymbol{x},\boldsymbol{x}_i))_{i=1}^n \in \mathbb{R}^{n \times 1}$

$$\text{Posterior Mean: } m_n(\boldsymbol{x}) = \boldsymbol{k}_{\mathsf{X}}^\intercal(\boldsymbol{x})(\mathsf{K}+\zeta\mathsf{I})^{-1}\boldsymbol{y}$$

$$\text{Posterior Covariance: } K_n(\boldsymbol{x},\boldsymbol{x}') = K(\boldsymbol{x},\boldsymbol{x}') - \boldsymbol{k}_{\mathsf{X}}^\intercal(\boldsymbol{x})(\mathsf{K}+\zeta\mathsf{I})^{-1}\boldsymbol{k}_{\mathsf{X}}(\boldsymbol{x}')$$

$$\text{Posterior Mean: } m_n(\boldsymbol{x}) = \boldsymbol{k}_{\mathsf{X}}^{\intercal}(\boldsymbol{x})(\mathsf{K}+\zeta\mathsf{I})^{-1}\boldsymbol{y}$$

$$\text{Posterior Covariance: } K_n(\boldsymbol{x},\boldsymbol{x}') = K(\boldsymbol{x},\boldsymbol{x}') - \boldsymbol{k}_{\mathsf{X}}^{\intercal}(\boldsymbol{x})(\mathsf{K}+\zeta\mathsf{I})^{-1}\boldsymbol{k}_{\mathsf{X}}(\boldsymbol{x}')$$

Key is to solve systems of the form

$$(\mathsf{K}+\zeta\mathsf{I})\boldsymbol{a} = \boldsymbol{b}$$

for $\boldsymbol{a} \in \mathbb{C}^n$ where $\boldsymbol{b} \in \mathbb{R}^n$

- $(\mathsf{K}+\zeta\mathsf{I})^{-1}\boldsymbol{y}$ precomputed during fitting, typically costs $\mathcal{O}(n^3)$
- $(\mathsf{K}+\zeta\mathsf{I})^{-1}\boldsymbol{k}_{\mathsf{X}}(\boldsymbol{x}')$ computed when evaluating uncertainty, typically costs $\mathcal{O}(n^2)$ after precomputing factorization of $\mathsf{K}+\zeta\mathsf{I}$

Fast Gaussian Process Regression

What? Induce structure in $\mathsf{K}+\zeta\mathsf{I}$ so solving $(\mathsf{K}+\zeta\mathsf{I})\boldsymbol{a}=\boldsymbol{b}$ for $\boldsymbol{a}$ costs $\mathcal{O}(n\log n)$
How? Match quasi-random sequences with structured kernels [Rathinavel, 2019]

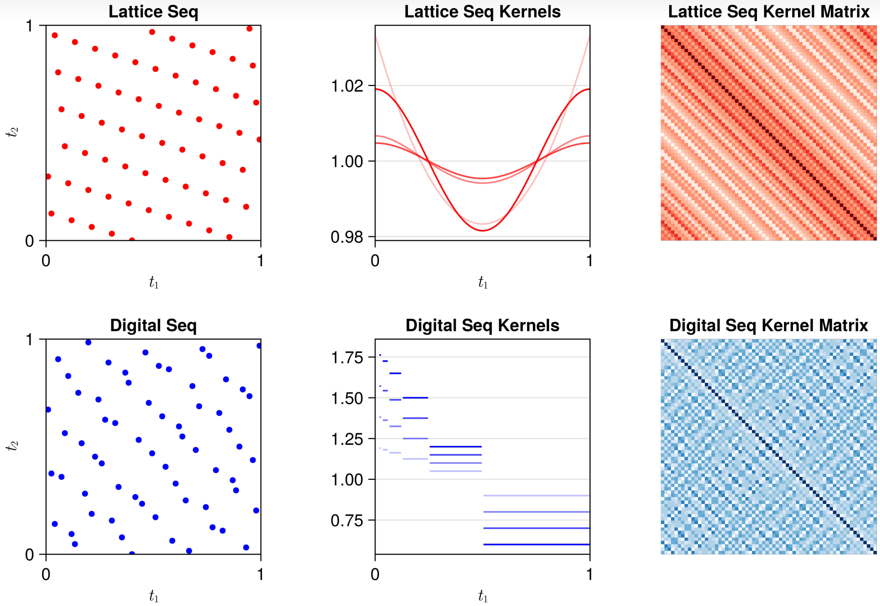
- K circulant with lattice sequence X and shift invariant kernel

$$K(\boldsymbol{x},\boldsymbol{x}')=K((\boldsymbol{x}-\boldsymbol{x}') \mod 1)$$

- K block-Toeplitz with digital sequence X and digitally shift invariant kernel

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

where $\ominus$ is XOR (exclusive or) of base 2 digits


Lattice Seq
Lattice Seq Kernels
Lattice Seq Kernel Matrix
Digital Seq
Digital Seq Kernels
Digital Seq Kernel Matrix
t_1
t_2
0
1
0.98
1.00
1.02
0.75
1.00
1.25
1.50
1.75

For circulant or block-Toeplitz K we have

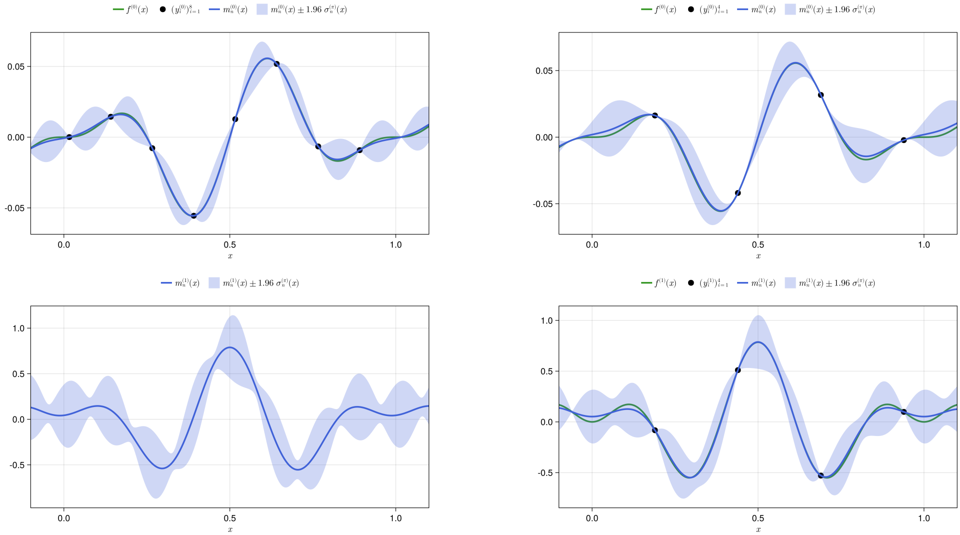
- $\mathsf{K}+\zeta\mathsf{I}$ inherits same structure as K
- Eigendecomposition $\mathsf{K}=\mathsf{V}\Lambda\mathsf{V}^\dagger$ with $\mathsf{V}^{-1}=\mathsf{V}^\dagger=$ Hermitian of V
- $\mathcal{F}(\boldsymbol{a}):=\mathsf{V}^\dagger\boldsymbol{a}$ and $\mathcal{F}^{-1}(\boldsymbol{b}):=\mathsf{V}\boldsymbol{b}$ can be computed in $\mathcal{O}(n\log n)$
  - Circulant K means $\mathcal{F}(\boldsymbol{a})$ is the fast Fourier transform of $\boldsymbol{a}$
  - Block-Toeplitz K means $\mathcal{F}(\boldsymbol{a})$ is the fast Walsh-Hadamard transform of $\boldsymbol{a}$
- First column of V is $\mathbf{1}/\sqrt{n}$

Solve $(\mathsf{K}+\zeta\mathsf{I})\boldsymbol{a}=\boldsymbol{b}$ for $\boldsymbol{a}$ at cost $\mathcal{O}(n\log n)$ with

$$\boldsymbol{a}=\mathcal{F}^{-1}\left(\frac{\mathcal{F}(\boldsymbol{b})}{\boldsymbol{\lambda}+\zeta}\right)$$

where $\boldsymbol{\lambda}=\mathrm{diag}(\Lambda)=\sqrt{n}\mathcal{F}(\boldsymbol{k}_{\mathsf{X}}(\boldsymbol{x}_1))$ and the division is done elementwise

Derivative Informed Gaussian Process Regression



Linear functional of a Gaussian process is still a Gaussian process

$$f^{(\boldsymbol{\beta})}(\boldsymbol{x}) := \frac{\partial^{|\boldsymbol{\beta}|}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}} f(\boldsymbol{x}) := \frac{\partial^{|\boldsymbol{\beta}|}}{\partial x_1^{\beta_1} \cdots \partial x_s^{\beta_s}} f(\boldsymbol{x})$$

$$\mathrm{Cov}[f^{(\boldsymbol{\beta})}(\boldsymbol{x}), f^{(\boldsymbol{\beta}')}(\boldsymbol{x}')] = \frac{\partial^{|\boldsymbol{\beta}|}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}}} \frac{\partial^{|\boldsymbol{\beta}'|}}{\partial \boldsymbol{x}^{\boldsymbol{\beta}'}} \mathrm{Cov}[f(\boldsymbol{x}), f(\boldsymbol{x}')] =: K^{(\boldsymbol{\beta},\boldsymbol{\beta}')}(\boldsymbol{x}, \boldsymbol{x}')$$

With m derivative multi-indices $\boldsymbol{\beta}_1, \dots, \boldsymbol{\beta}_m \in \mathbb{N}_0^s$ the kernel (Gram) matrix becomes

$$\mathsf{K} = \begin{pmatrix} \mathsf{K}^{(\boldsymbol{\beta}_1,\boldsymbol{\beta}_1)} & \dots & \mathsf{K}^{(\boldsymbol{\beta}_1,\boldsymbol{\beta}_m)} \\ \vdots & \ddots & \vdots \\ \mathsf{K}^{(\boldsymbol{\beta}_m,\boldsymbol{\beta}_1)} & \dots & \mathsf{K}^{(\boldsymbol{\beta}_m,\boldsymbol{\beta}_m)} \end{pmatrix} \in \mathbb{R}^{nm \times nm}, \quad \mathsf{K}^{(\boldsymbol{\beta}_k,\boldsymbol{\beta}_l)} = \left(K^{(\boldsymbol{\beta}_k,\boldsymbol{\beta}_l)}(\boldsymbol{x}_i, \boldsymbol{x}_j)\right)_{i,j=1}^n$$

so solving $(\mathsf{K} + \zeta \mathsf{I})\boldsymbol{a} = \boldsymbol{b}$ for $\boldsymbol{a} \in \mathbb{C}^{mn}$ where $\boldsymbol{b} \in \mathbb{R}^{mn}$ costs $\mathcal{O}(m^3 n^3)$ in general

Fast Gaussian Process Regression with Derivative Information

$\mathsf{K}^{(\boldsymbol{\beta}_k,\boldsymbol{\beta}_l)}$ retains structure of $\mathsf{K}^{(\mathbf{0},\mathbf{0})}$ e.g. circulant or block Toeplitz

$$\begin{pmatrix} \mathsf{K}^{(\boldsymbol{\beta}_1,\boldsymbol{\beta}_1)} & \dots & \mathsf{K}^{(\boldsymbol{\beta}_1,\boldsymbol{\beta}_m)} \\ \vdots & \ddots & \vdots \\ \mathsf{K}^{(\boldsymbol{\beta}_m,\boldsymbol{\beta}_1)} & \dots & \mathsf{K}^{(\boldsymbol{\beta}_m,\boldsymbol{\beta}_m)} \end{pmatrix} = \begin{pmatrix} \mathsf{V} & & \\ & \ddots & \\ & & \mathsf{V} \end{pmatrix} \underbrace{\begin{pmatrix} \mathsf{\Lambda}^{(\boldsymbol{\beta}_1,\boldsymbol{\beta}_1)} & \dots & \mathsf{\Lambda}^{(\boldsymbol{\beta}_1,\boldsymbol{\beta}_m)} \\ \vdots & \ddots & \vdots \\ \mathsf{\Lambda}^{(\boldsymbol{\beta}_m,\boldsymbol{\beta}_1)} & \dots & \mathsf{\Lambda}^{(\boldsymbol{\beta}_m,\boldsymbol{\beta}_m)} \end{pmatrix}}_{\mathsf{\Lambda} \in \mathbb{R}^{nm \times nm}} \begin{pmatrix} \mathsf{V}^\dagger & & \\ & \ddots & \\ & & \mathsf{V}^\dagger \end{pmatrix}$$

Let $\otimes$ be the Kronecker product so

$$\mathsf{K} + \zeta \mathsf{I} = (\mathsf{I} \otimes \mathsf{V})(\mathsf{\Lambda} + \zeta \mathsf{I})(\mathsf{I} \otimes \mathsf{V}^\dagger)$$

$$\mathsf{K} + \zeta \mathsf{I} = (\mathsf{I} \otimes \mathsf{V})(\Lambda + \zeta \mathsf{I})(\mathsf{I} \otimes \mathsf{V}^\dagger)$$

Since Λ is a diagonal block (striped) matrix, there is some permutation matrix P with

$$\mathsf{P}^\intercal(\Lambda + \zeta \mathsf{I})\mathsf{P} = \Upsilon + \zeta \mathsf{I}$$

where

$$\Upsilon = \begin{pmatrix} \Upsilon_1 & & \\ & \ddots & \\ & & \Upsilon_n \end{pmatrix}$$

is block diagonal with $\Upsilon_{i,kl} = \lambda_i^{(\boldsymbol{\beta}_k, \boldsymbol{\beta}_l)}$. Then

$$\mathsf{K} + \zeta \mathsf{I} = (\mathsf{I} \otimes \mathsf{V})\mathsf{P}(\Upsilon + \zeta \mathsf{I})\mathsf{P}^\intercal(\mathsf{I} \otimes \mathsf{V}^\dagger)$$

Cost of solving $(\mathsf{K}+\zeta\mathsf{I})\boldsymbol{a}=\boldsymbol{b}$ for $\boldsymbol{a}$ with structured $\mathsf{K}+\zeta\mathsf{I}$

$$\mathsf{K}+\zeta\mathsf{I}=(\mathsf{I}\otimes\mathsf{V})\mathsf{P}(\Upsilon+\zeta\mathsf{I})\mathsf{P}^\intercal(\mathsf{I}\otimes\mathsf{V}^\dagger)$$

Reduce cost from $\mathcal{O}(m^3n^3)$ to $\mathcal{O}(m^2n\log n+m^3n)$ with the following algorithm

1. Constructing Υ from eigenvalues $\boldsymbol{\lambda}^{(\boldsymbol{\beta}_k,\boldsymbol{\beta}_l)}=\mathcal{F}\left(\boldsymbol{k}_{\mathsf{X}}^{(\boldsymbol{\beta}_k,\boldsymbol{\beta}_l)}(\boldsymbol{x}_1)\right)$ costs $\mathcal{O}(m^2n\log n)$
2. $\check{\boldsymbol{b}}:=\mathsf{P}^\intercal(\mathsf{I}\otimes\mathsf{V}^\dagger)\boldsymbol{b}$ can be computed at cost $\mathcal{O}(mn\log n)$
3. $\check{\boldsymbol{a}}:=(\Upsilon+\zeta\mathsf{I})^{-1}\check{\boldsymbol{b}}$ can be computed at cost $\mathcal{O}(m^3n)$
4. $\boldsymbol{a}=(\mathsf{I}\otimes\mathsf{V})\mathsf{P}\check{\boldsymbol{a}}$ can be computed at cost $\mathcal{O}(mn\log n)$

Fast Kernel Parameter Optimization

K often depends on parameters $\boldsymbol{\theta}$ e.g. scaling factor, lengthscales, noise variance ζ

$\boldsymbol{\theta}$ which maximizes the marginal log likelihood is

$$\begin{aligned}\operatorname*{argmin}_{\boldsymbol{\theta}} L(\boldsymbol{\theta}|\boldsymbol{y}) &= \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\log\det(\mathsf{K}+\zeta\mathsf{I}) + \boldsymbol{y}^\intercal(\mathsf{K}+\zeta\mathsf{I})^{-1}\boldsymbol{y}\right] \\ &= \operatorname*{argmin}_{\boldsymbol{\theta}} \sum_{i=1}^{n} \left[\log\det(\Upsilon_i+\zeta\mathsf{I}) + \check{\boldsymbol{y}}_i^\dagger(\Upsilon_i+\zeta\mathsf{I})^{-1}\check{\boldsymbol{y}}_i\right]\end{aligned}$$

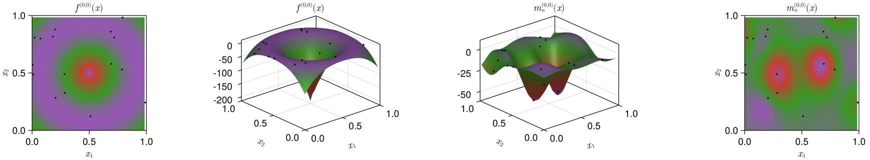
where

$$\check{\boldsymbol{y}} := \begin{pmatrix}\check{\boldsymbol{y}}_1 \\ \vdots \\ \check{\boldsymbol{y}}_n\end{pmatrix} := \mathsf{P}^\intercal(\mathsf{I}\otimes\mathsf{V}^\dagger)\boldsymbol{y}.$$

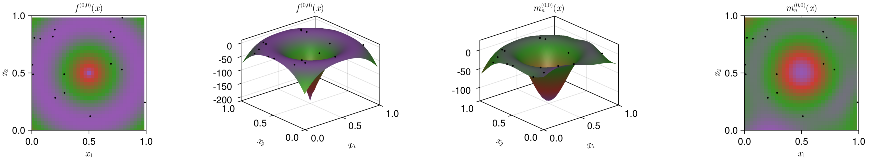
Both $L(\boldsymbol{\theta}|\boldsymbol{y})$ and $\partial_{\theta_j} L(\boldsymbol{\theta}|\boldsymbol{y})$ can still be computed in $\mathcal{O}(m^2 n\log n + m^3 n)$

Analytic Donut[1] in UMBridge [Seelinger et al., 2023]

IID Points, SE Kernel: No Gradient Information

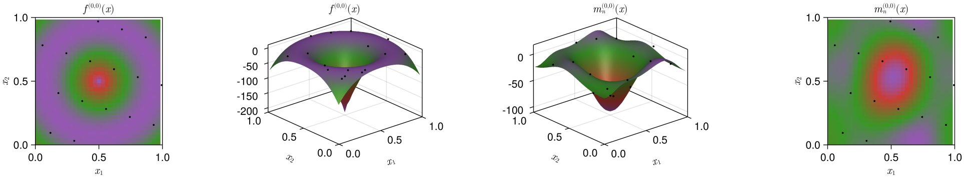


IID Points with SE Kernel: With Gradient Information

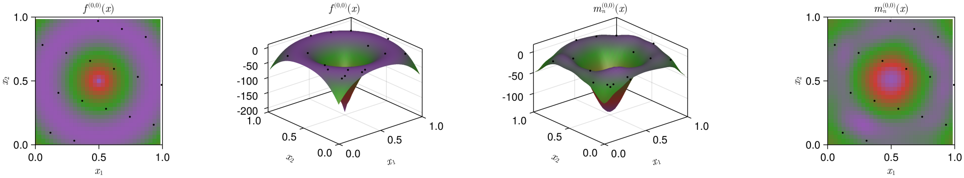


[1] https://um-bridge-benchmarks.readthedocs.io/en/docs/inverse-benchmarks/analytic-donut.html

Lattice Points, Matching Kernel: No Gradient Information



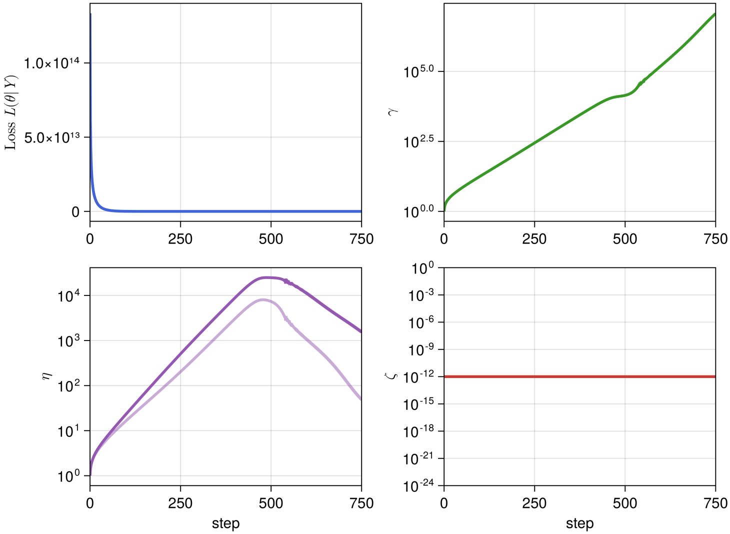
Lattice Points, Matching Kernel: With Gradient Information



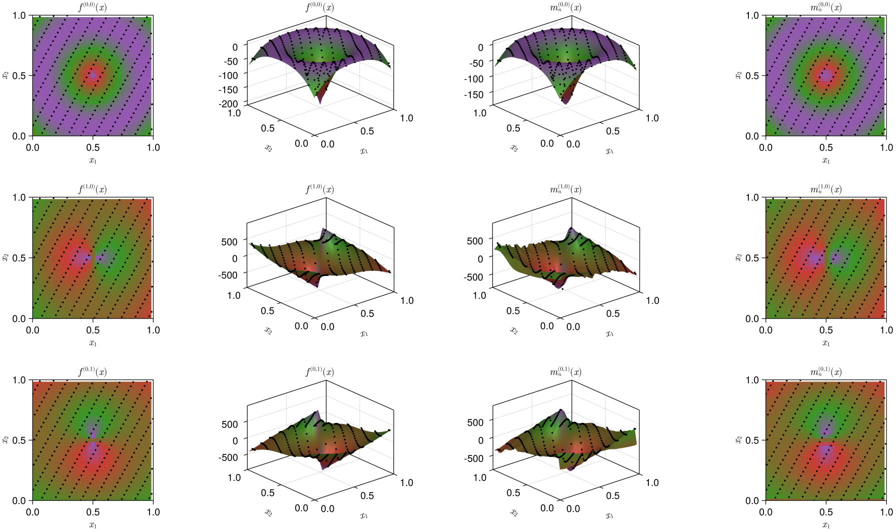
Lattice GP with Gradients for Donut Example

```julia
import FastGaussianProcesses; import QMCGenerators; import UMBridge

# docker run -it -p 4243:4243 linusseelinger/benchmark-analytic-donut
model = UMBridge.HTTPModel("posterior", "http://localhost:4243")
β = [0 0; 1 0; 0 1] # observe f^{(0,0)}, f^{(1,0)}, f^{(0,1)}

function f(x::Vector{Float64})
    z = [6*x[1]-3, 6*x[2]-3] # both componenets between -3 and 3
    [    UMBridge.evaluate(model,[z],Dict())[1][1], # f^{(0,0)}
        (UMBridge.gradient(model,0,0,[z],[1]) .* [6, 6])...] # f^{(1,0)}, f^{(0,1)}
end

seq = QMCGenerators.RandomShift(QMCGenerators.LatticeSeqB2(2),1,7) # s=2, seed=7
gp = FastGaussianProcesses.FastGaussianProcess(f,seq,2^8;β=β,optim_steps=750) # n=2^10
FastGaussianProcesses.plot_gp_optimization(gp,figpath=joinpath(@__DIR__,"optim.png"))
FastGaussianProcesses.plot_gp_2s(gp;f=f,β=β,figpath=joinpath(@__DIR__,"gp.png"))
```

Kernel Parameter Optimization



Gaussian Process and Gradient Visualization



Future Work

Theory

- Can we improve the $\mathcal{O}(m^2 n \log n + m^3 n)$ cost by relating $\boldsymbol{\lambda}^{(\boldsymbol{\beta},\boldsymbol{\kappa})}$ to $\boldsymbol{\lambda}^{(\boldsymbol{\beta}',\boldsymbol{\kappa}')}$?
- Link with RKHS setting
  - General GPR and RKHS kernel interpolation connections in [Kanagawa et al., 2018]
  - Optimize weights in [Kaarnioja et al., 2022] with GPR kernel parameter optimization
- Analogous developments for digital sequences

Practical Software

- `QMCGenerators.jl`[2]: Quasi-random sequence generators with randomizations
- `FastGaussianProcesses.jl`[3]: Fast GPR with derivatives (in development)
- `QMCPy`[4] [Choi et al., 2022]
  - Quasi-random sequence generators with randomizations
  - Fast GPR cubature [Rathinavel, 2019]

[2] https://github.com/alegresor/QMCGenerators.jl

[3] https://github.com/alegresor/FastGaussianProcesses.jl

[4] https://github.com/QMCSoftware/QMCSoftware

References I

Yifan Chen, Bamdad Hosseini, Houman Owhadi, and Andrew M. Stuart. Solving and learning nonlinear pdes with gaussian processes. *Journal of Computational Physics*, 447:110668, 2021. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2021.110668. URL https://www.sciencedirect.com/science/article/pii/S0021999121005635.

Sou-Cheng T. Choi, Fred J. Hickernell, Rathinavel Jagadeeswaran, Michael J. McCourt, and Aleksei G. Sorokin. Quasi-monte carlo software. In Alexander Keller, editor, *Monte Carlo and Quasi-Monte Carlo Methods*, pages 23–47, Cham, 2022. Springer International Publishing. ISBN 978-3-030-98319-2.

R. Jagadeeswaran and Fred J. Hickernell. Fast automatic bayesian cubature using lattice sampling. *Statistics and Computing*, 29(6):1215–1229, Sep 2019. ISSN 1573-1375. doi: 10.1007/s11222-019-09895-9. URL http://dx.doi.org/10.1007/s11222-019-09895-9.

References II

Rathinavel Jagadeeswaran and Fred J Hickernell. Fast automatic bayesian cubature using sobol sampling. In *Advances in Modeling and Simulation: Festschrift for Pierre L'Ecuyer*, pages 301–318. Springer, 2022.

Vesa Kaarnioja, Yoshihito Kazashi, Frances Y Kuo, Fabio Nobile, and Ian H Sloan. Fast approximation by periodic kernel-based lattice-point interpolation with application in uncertainty quantification. *Numerische Mathematik*, 150(1):33–77, 2022.

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur. Gaussian processes and kernel methods: A review on connections and equivalences. *arXiv preprint arXiv:1807.02582*, 2018.

Carl Edward Rasmussen, Christopher KI Williams, et al. *Gaussian processes for machine learning*, volume 1. Springer, 2006.

Jagadeeswaran Rathinavel. *Fast automatic Bayesian cubature using matching kernels and designs*. Illinois Institute of Technology, 2019.

References III

Linus Seelinger, Vivian Cheng-Seelinger, Andrew Davis, Matthew Parno, and Anne Reinarz. Um-bridge: Uncertainty quantification and modeling bridge. *Journal of Open Source Software*, 8(83):4748, 2023.

Ercan Solak, Roderick Murray-Smith, WE Leithead, D Leith, and Carl Rasmussen. Derivative observations in gaussian process models of dynamic systems. *Advances in neural information processing systems*, 15, 2002.

Aleksei G. Sorokin, Aleksandra Pachalieva, Daniel O'Malley, James M. Hyman, Fred J. Hickernell, and Nicolas W. Hengartner. Computationally efficient and error aware surrogate construction for numerical solutions of subsurface flow through porous media, 2023.