Robust Approximation of Sensitivity Indices in QMCPy

Aleksei G. Sorokin[1], Sou-Cheng T. Choi[1,2], Fred J. Hickernell[1], Michael J. McCourt[3], Jagadeeswaran Rathinavel[4]

[1]Illinois Institute of Technology, [2]Kamakura Corporation, [3]SigOpt, [4]Wi-Tronix

Thank you to NSF Grant DMS-1522687 and [3]SigOpt for supporting QMCPy

Sensitivity Indices

Functional Analysis of Variance Decomposition

$$f(\boldsymbol{x}) = \sum_{u\subseteq\{1,\dots,d\}} f_u(\boldsymbol{x}_u), \qquad \sigma^2 = \sum_{u\subseteq\{1,\dots,d\}} \sigma_u^2$$

Sensitivity Indices (SI) via $X, Z \overset{\text{IID}}{\sim} \mathcal{U}(0,1)^d$

$$\text{closed SI} \quad \underline{s}_u = \frac{\sum_{v\subseteq u}\sigma_v^2}{\sigma^2} = \overbrace{\frac{\mathbb{E}[f(X)(f(X_u,Z_{-u})-f(Z))]}{\underbrace{\mathbb{E}[f^2(X)]}_{\mu_2} - \underbrace{(\mathbb{E}[f(X)])^2}_{\mu_1}}}^{\text{closed Sobol' index: } \underline{\tau}_u}$$

$$\text{total SI} \quad \overline{s}_u = \frac{\sum_{v\cap u\neq\emptyset}\sigma_v^2}{\sigma^2} = \overbrace{\frac{\mathbb{E}[(f(Z)-f(X_u,Z_{-u}))^2/2]}{\underbrace{\mathbb{E}[f^2(X)]}_{\mu_2} - \underbrace{(\mathbb{E}[f(X)])^2}_{\mu_1}}}^{\text{total Sobol' index: } \overline{\tau}_u}$$

Vectorized Monte Carlo

$$\boldsymbol{\mu} = \mathbb{E}[f(X)] \approx \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{x}_i) = \hat{\boldsymbol{\mu}}_n \in \mathbb{R}^\rho, \qquad X \sim \mathcal{U}(0,1)^d$$

objective function $f : [0,1]^d \to \mathbb{R}^\rho$
discrete distribution $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots\} \sim \mathcal{U}(0,1)^d$ induced error from

- Full Grids: $\mathcal{O}(n^{-1/d})$
- IID (Monte Carlo): $\mathcal{O}(n^{-1/2})$
- Low Discrepancy (Quasi-Monte Carlo): $\mathcal{O}(n^{-1+\delta})$

Error Propagation for $\underline{s}_u$

Individual Bounds via (Q)MC

$$\boldsymbol{\mu} = \begin{pmatrix}\mu_1\\ \mu_2\\ \underline{\tau}_u\end{pmatrix} \in \left[\begin{pmatrix}\mu_1^-\\ \mu_2^-\\ \underline{\tau}_u^-\end{pmatrix}, \begin{pmatrix}\mu_1^+\\ \mu_2^+\\ \underline{\tau}_u^+\end{pmatrix}\right] \quad \left(\begin{array}{c}\text{guaranteed or}\\ \text{with high probability}\end{array}\right)$$

Combined Bounds

$\underline{s}_u \in [\underline{s}_u^-, \underline{s}_u^+]$ (guaranteed or with high probability) where

$$\underline{s}_u^- = \max\left(0,\ \min\left(\frac{\underline{\tau}_u^-}{\mu_2^+ - (\mu_1^-)^2}, \frac{\underline{\tau}_u^-}{\mu_2^+ - (\mu_1^+)^2}\right)\right)$$

$$\underline{s}_u^+ = \min\left(1,\ \max\left(\frac{\underline{\tau}_u^+}{\mu_2^- - (\mu_1^-)^2}, \frac{\underline{\tau}_u^+}{\mu_2^- - (\mu_1^+)^2}\right)\right)$$

Ishigami Function Example

$$g(T) = \left(1 + bT_3^4\right)\sin(T_1) + a\sin^2(T_2), \qquad T \sim \mathcal{U}(-\pi,\pi)^3$$
$$f(X) = g(\pi(2X-1)), \qquad X \sim \mathcal{U}(0,1)^3$$

```
>>> import qmcpy as qp; import numpy as np
>>> dnb2 = qp.DigitalNetB2(3,seed=7)
>>> ishigami = qp.Ishigami(dnb2,a=7,b=0.1)
>>> idxs = [[0],[1],[2],[0,1],[0,2],[1,2]]
>>> ishigami_si = qp.SensitivityIndices(ishigami,idxs)
>>> qmc_algo = qp.CubQMCNetG(ishigami_si,abs_tol=1e-3)
>>> solution,data = qmc_algo.integrate()
>>> print("Approx took %.1f sec and n = 2^(%d)"%
...     (data.time_integrate,np.log2(data.n_total)))
Approx took 1.1 sec and n = 2^(16)
```

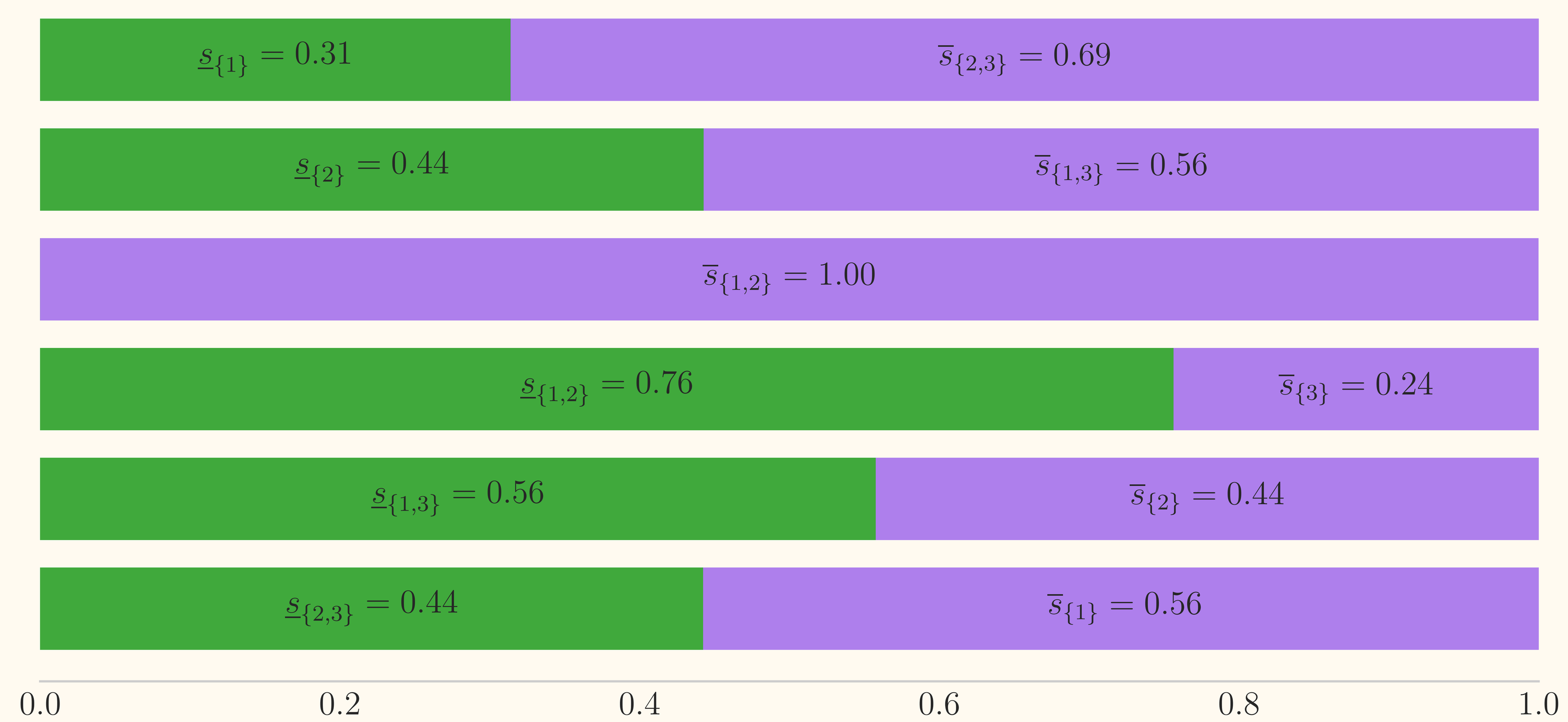
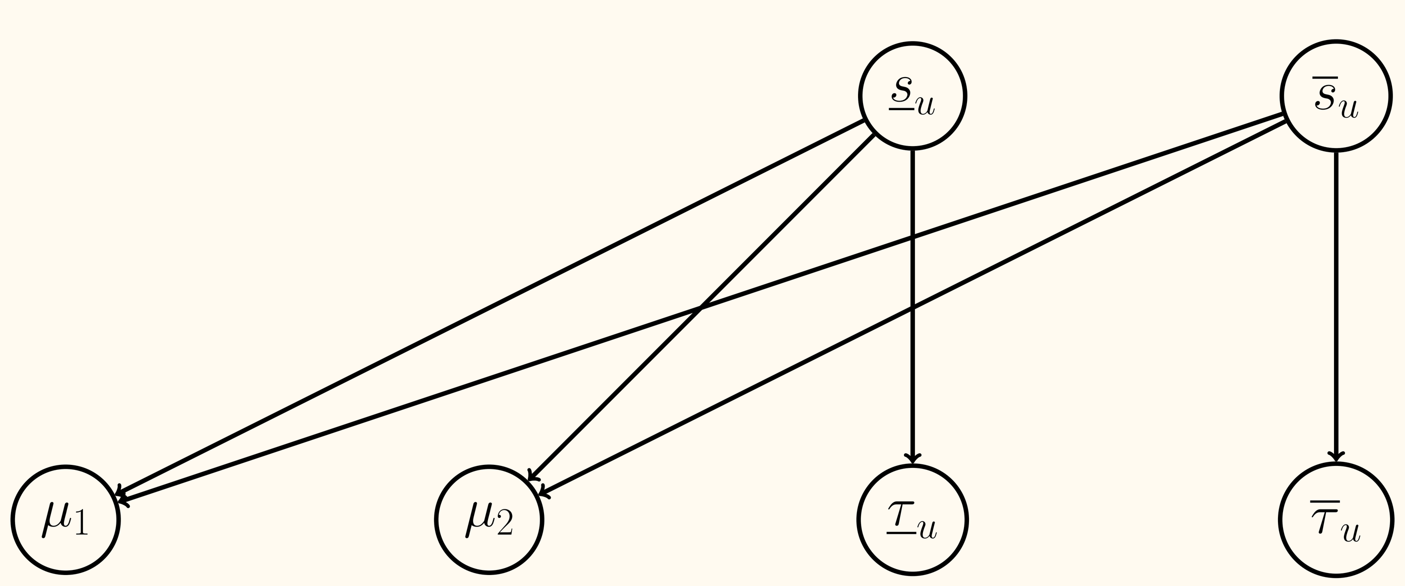


Figure 1: Sensitivity index approximations visualizing $\underline{s}_u + \overline{s}_{u^c} = 1$

Computation Dependency Structure

Propagate evaluation flags on combined solutions to flags on individual integrands



QMCPy Support Features

- Shared discrete distribution points
- Multi-dimensional individual and combined solutions
- Guaranteed error estimation
- Adaptive sampling to meet error tolerance
- Conservative function evaluation
- Parallel function evaluation

QMCPy Installation

PyPI: `pip install qmcpy`
GitHub: `git@github.com:QMCSoftware/QMCSoftware.git`

References

1. S.-C. T. Choi, F. J. Hickernell, R. Jagadeeswaran, M. J. McCourt, A. G. Sorokin, "Quasi-Monte Carlo Software," Monte Carlo and Quasi-Monte Carlo Methods 2020.
2. S.-C. T. Choi, F. J. Hickernell, R. Jagadeeswaran, M. J. McCourt, A. G. Sorokin, "QMCPy: A Quasi- Monte Carlo Python Library," www.qmcpy.org.
3. S.-C. T. Choi, Y. Ding, F. J. Hickernell, L. Jiang, D. Li, R. Jagadeeswaran, L.-A. Jimenez Rugama, X. Tong, K. Zhang, Y. Zhang, and X. Zhou, "GAIL: Guaranteed Automatic Integration Library," Version 2.3.1, MATLAB Software, 2020.
4. F. J. Hickernell, L.-A. Jimenez Rugama, and D. Li, "Adaptive Quasi-Monte Carlo Methods for Cubature," 2017, arXiv: 1702.01491 [math.NA].
5. R. Jagadeeswaran, F. J. Hickernell, "Fast automatic Bayesian cubature using lattice sampling," Statistics and Computing 29.6 (Sept. 2019), DOI 10.1007/s11222-019-09895-9.
6. T. Ishigami, T. Homma, "An importance quantification technique in uncertainty analysis for computer models", Uncertainty Modeling and Analysis, 1990.
7. L.-A. Jimenez Rugama, and L. Gilquin, "Reliable error estimation for Sobol' indices," Statistics and Computing 28.4 (July 2018), DOI 10.1007/s11222-017-9759-1.
8. A. B. Owen, "Monte Carlo theory, methods and examples," 2018.
9. M. Hofert and C. Lemieux, "qrng: (Randomized) Quasi-Random Number Generators," R package version 0.0-7, (2019).
10. A. B. Owen. "A randomized Halton algorithm in R2017." arXiv:1706.02808 [stat.CO].