# Instructions on Codes of SESK2018

SHEN Haihui
haihui.shen@my.cityu.edu.hk

May 3, 2018

### Disclaimer



WARNING: These codes are written only for the purpose of demonstration and verification. While the correctness has been carefully checked, the quality such as standardability, clarity, generality, and efficiency has not been well considered.

## 1 Introduction

The SESK2018 package contains the entire MATLAB implementations of all the numerical studies in Shen et al. (2018). The proposed stylized-model enhanced stochastic kriging (SESK) and compared ordinary stochastic kriging (OSK) are implemented using the open source code for stochastic kriging (stochastickriging.net). The codes are split into following folders.

- `\SKDec122009`, contains codes for stochastic kriging. It is originally downloaded from stochastickriging.net. A few lines of codes are added to calculate the measures for stylized model defined in Shen et al. (2018, §3).
- `printSK.m`, is a function, which prints out the parameters and measures of the kriging model.
- `\MM1_noiseless`, contains codes of the $M/M/1$ Queue in Shen et al. (2018, §2.2). Functions in `\SKDec122009` are called.
- `\MG1_exact_variance`, contains codes of the Illustrative Example – $M/G/1$ Queue in Shen et al. (2018, §3.4). Functions in `\SKDec122009`, and `printSK.m` are called.
- `\HUG_hospital`, contains codes of Case Study 1 – Patient Flow in a Hospital in Shen et al. (2018, §5.1). Functions in `\SKDec122009`, and `printSK.m` are called.

- `\Terminal_dock`, contains codes of Case Study 2 – Dock Allocation at an Air Cargo Terminal in Shen et al. (2018, §5.2). Functions in `\SKDec122009`, and `printSK.m` are called.

## 2 Installation

The codes were written and run in MATLAB R2015a, on Windows 7 Enterprise 64-bit OS, Intel i7-3770 CPU @ 3.40GHz, 8 GB RAM.

To install the MATLAB codes, just copy the entire folder `\SESK2018` into your MATLAB directory, or change the path of MATLAB to the folder `\SESK2018`.

## 3 Details on Numerical Studies

### 3.1 $M/M/1$ Queue in Shen et al. (2018, §2.2)

Get into folder `\MM1_noiseless`, and run script `MM1Queue.m`, which is self-explanatory. It will call functions in `\SKDec122009`.

### 3.2 $M/G/1$ Queue in Shen et al. (2018, §3.4)

Get into folder `\MG1_exact_variance`.

- To perform OSK and SESK, run script `MG1Queue.m`. The kriging model is specified through assigning different values to variable `kriging_type` in the beginning: `0` – OSK; `1` – SESK with $q^{(1)}$; `2` – SESK with $q^{(2)}$; `3` – SESK with $q^{(3)}$. It will call functions in `\SKDec122009`, and `printSK.m`.
- To perform stochastic kriging with gradient (SKG), run script `MG1Queue_SKG.m`. It will call functions in `\SKG`, and `printSK.m`. `\SKG` is implementation of SKG proposed by Chen et al. (2013). More instructions are available at simopt.github.io/SKG.

### 3.3 Patient Flow in a Hospital in Shen et al. (2018, §5.1)

Get into folder `\HUG_hospital`.

- Run script `OSK_SESK.m`. The kriging model is specified through assigning different values to variable `kriging_type` in the beginning: `0` – OSK; `1` – SESK. It will call functions in `\SKDec122009`, and `printSK.m`. The "true" response surface are evaluated in script `Generate_True_Sojourn_Time.m`.
- The simulation of the nine-station queueing network with finite capacity and BAS blocking, is accomplished by `HUG.m`, which will call `QN_GGsK_BAS_FIFO.m`. `HUG.m` also requires some third-party functions collected in `\third_party_function`. This implementation is a specialization of the Queueing Network Simulator (simopt.github.io/QNSim); see more instructions there.

- The stylized model is implemented in `QNA_Korporaal.m`, which is a modification of the decomposition method proposed in Korporaal et al. (2000); see the detailed description of the construction in Appendix B of Shen et al. (2018).

### 3.4 Dock Allocation at Terminal in Shen et al. (2018, §5.2)

Get into folder `\Terminal_dock`.

- To perform OSK and SESK on the 40 sets of design points, which are generated by some random search procedure, run script `OSK_SESK_all_designs.m`.

- The "true" response surface is evaluated in script `Generate_True_Waiting_Time.m`. For the sake of efficiency, we simulate each dock independently, and then compute the weighted average for the measure on the whole terminal (four docks). Originally, the simulation of each dock should be conducted using `Q_GGsK_FIFO_NS.m`, which is a simulator for $G_t/G/s$ queue. However the simulation is too slow since we set very long run length in order to get "true" response surface. In the end, the simulation for this part is carried out in Arena (which is much faster!) and the result is recorded and used to compute the final response in `Generate_True_Waiting_Time.m`. The Arena models (Version 14.00.00) of four docks are given in `/Terminal Dock_Arena` for reference. Notice that the simulation result from Arena is consistent to that from `Q_GGsK_FIFO_NS.m`.

- The 40 sets of design points and the observed response on the design points are generated in script `Generate_Design_Points_and_Obs.m`, which will call functions in `/Search_Optimal_Designs`.

- Here the optimal design is defined via maximizing the smallest pairwise distance between the design points. And the idea of algorithm is to consecutively generate different sets of pure random design points in the space, and keep record of the current best design so far. If the current best best design keeps unchanged for 1,000 iterations or the total iteration number reaches 10,000, the algorithm stops and outputs the current best design. The comparison between two sets of designs is accomplished by `mm.m` and `jd.m`, which are adopted from Forrester et al. (2008, §1.4.3).

- To generate observations on the design points, originally `Dock.m`, which is a simulator for the four-dock system, should be used. For the sake of efficiency, we again simulate each dock independently (using `Q_GGsK_FIFO_NS.m` in this case), and then compute the weighted average for the measure.

- To perform OSK and SESK on one typical set of design points, run script `OSK_SESK.m`.

- To perform discrete optimization via simulation (DOvS), get into `\DOvS` and run script `Comparision_GPS_SEGPS.m`. It will call the functions of Gaussian process-based search (GPS) algorithm proposed by Sun et al. (2014), and stylized-model enhanced GPS (SEGPS) algorithm proposed in Shen et al. (2018, §5.2.2). For implementation of GPS, more instructions are available at simopt.github.io/GPS. The implementation

of SEGPS requires some simple modifications to GPS, which are well described in Shen et al. (2018, §5.2.2).

## References


Chen, Xi, Bruce E. Ankenman, and Barry L. Nelson (2013). Enhancing stochastic kriging metamodels with gradient estimators. *Operations Research 61*(2), 512–528.

Forrester, Alexander I. J., András Sóbester, and Andy J. Keane (2008). *Engineering Design via Surrogate Modelling: A Practical Guide.* John Wiley & Sons, Inc.

Korporaal, R., A. Ridder, P. Kloprogge, and R. Dekker (2000). An analytic model for capacity planning of prisons in the Netherlands. *Journal of the Operational Research Society 51*(11), 1228–1237.

Shen, Haihui, L. Jeff Hong, and Xiaowei Zhang (2018). Enhancing Metamodeling for Queueing Simulation with Stylized Models. *IISE Transactions*, DOI: 10.1080/24725854.2018.1465242

Sun, Lihua, L. Jeff Hong, and Zhaolin Hu (2014). Balancing exploitation and exploration in discrete optimization via simulation through a Gaussian process-based search. *Operations Research 62*(6), 1416–1438.