NSQG Programming Structure

Shuotao Diao, sdiao@usc.edu

May 2019

1 Introduction

$$\min \ f(x,z) = c^T x + \mathbb{E}[h(x,W)|Z=z] \tag{1a}$$
$$\text{s.t } Ax \leq b \tag{1b}$$

$$h(x,W) = \min \ d^T y \tag{2a}$$
$$\text{s.t. } D_e y + C_e x = b_e(W) \tag{2b}$$
$$D_i y + C_i x \leq b_i(W) \tag{2c}$$

The non-parametric estimation of problem (1) is written as follows.

$$\min \ f(x,z) = c^T x + \sum_{i=1}^{N} v_{n,i}(z) h(x,W_i) \tag{3a}$$
$$\text{s.t } Ax \leq b \tag{3b}$$

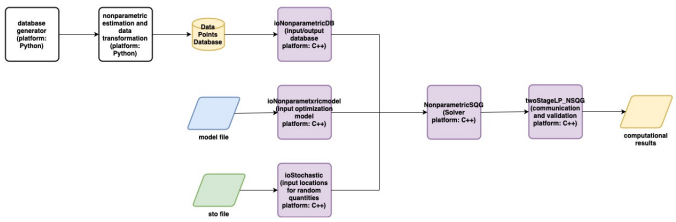

Figure 1: Programming structure of Nonparametric SQG

2 User Manual

A valid instance needs to contain model file, stochastic file and data file. All the files are written in a way which is quite similar to XML format. Model file

includes all the matrices in the model. Stochastic file shows the locations of random quantities in the model. Data file contains all the random quantities. There is a hierarchy in the data file, which will be given more details later. In short, a data file is a database which has several datasets. In the meanwhile, a dataset contains multiple data points.

2.1 Model File Setup

The name of model file needs to be "model.txt". Currently, all the matrices in the model need to be set up manually. A two stage shipment problem will be used as an example to demonstrate how to create those matrices.

$$\min f(x,z) = 5x_1 + 5x_2 + 5x_3 + 5x_4 + \sum_{i=1}^{N} v_{n,i}(z)h(x,W_i) \tag{4a}$$

$$\text{s.t. } x_1, ..., x_4 \geq 0 \tag{4b}$$

Here,

$$c = \begin{bmatrix} 5 & 5 & 5 & 5 \end{bmatrix} \tag{5a}$$

$$A = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \tag{5b}$$

$$b = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \tag{5c}$$

```
<c>
5,5,5,5;
</c>
<A>
-1,0,0,0;
0,-1,0,0;
0,0,-1,0;
0,0,0,-1;
</A>
<b>
0,0,0,0;
</b>
```

Figure 2: Example of c, A and b matrices in model file

The mathematical formulation of second stage problem is

$$
\begin{aligned}
h(x, W_i) = \min\ & 100y_1 + 100y_2 + 100y_3 + 100y_4 + && \text{(6a)}\\
& 1.5y_5 + 5.0026y_6 + 9.3408y_7 + 13.1240y_8 + && \text{(6b)}\\
& 16.0390y_9 + 17.8740y_{10} + 18.5000y_{11} + 17.8740y_{12} + && \text{(6c)}\\
& 16.0390y_{13} + 13.1240y_{14} + 9.3408y_{15} + 5.0026y_{16} + && \text{(6d)}\\
& 13.1240y_{17} + 9.3408y_{18} + 5.0026y_{19} + 1.5000y_{20} + 5.0026y_{21} + && \text{(6e)}\\
& 9.3408y_{22} + 13.1240y_{23} + 16.0390y_{24} + 17.8740y_{25} + 18.5000y_{26} + && \text{(6f)}\\
& 17.8740y_{27} + 16.0390y_{28} + 18.5000y_{29} + 17.8740y_{30} + 16.0390y_{31} + && \text{(6g)}\\
& 13.1240y_{32} + 9.3408y_{33} + 5.0026y_{34} + 1.5000y_{35} + 5.0026y_{36} + && \text{(6h)}\\
& 9.3408y_{37} + 13.1240y_{38} + 16.0390y_{39} + 17.8740y_{40} + && \text{(6i)}\\
& 13.1240y_{41} + 16.0390y_{42} + 17.8740y_{43} + 18.5000y_{44} + && \text{(6j)}\\
& 17.8740y_{45} + 16.0390y_{46} + 13.1240y_{47} + 9.3408y_{48} + 5.0026y_{49} + && \text{(6k)}\\
& 1.5000y_{50} + 5.0026y_{51} + 9.3408y_{52} && \text{(6l)}\\
\text{s.t. } & y_1, ..., y_{52} \geq 0 && \text{(6m)}\\
& -y_5 - y_{17} - y_{29} - y_{41} \leq b_{53}(W_i) && \text{(6n)}\\
& -y_6 - y_{18} - y_{30} - y_{42} \leq b_{54}(W_i) && \text{(6o)}\\
& ... && \text{(6p)}\\
& -y_{16} - y_{28} - y_{40} - y_{52} \leq b_{64}(W_i) && \text{(6q)}\\
& -y_1 + y_5 + ... + y_{16} - x_1 \leq 0 && \text{(6r)}\\
& -y_2 + y_{17} + ... + y_{28} - x_2 \leq 0 && \text{(6s)}\\
& -y_3 + y_{29} + ... + y_{40} - x_3 \leq 0 && \text{(6t)}\\
& -y_4 + y_{41} - ... - y_{52} - x_4 \leq 0 && \text{(6u)}
\end{aligned}
$$

2.2 Stochastic File Setup

The name of stochastic file must be "sto.txt". Currently, NSQG solver can only tackle the problem in which b_e and b_i are random. "*" is used to tell that one entry of b_e or b_i is random. According to equation 6, the dimension of b_i is 68 and random quantities appear at position 53, 54, ..., 64. The corresponding stochastic file is shown below.

```
<sto>
<be>
</be>
<bi>
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,*,*,*,*,*,*,*,*,*,*,*,*,*,*,0,0,0,0,0;
</bi>
</sto>
```

Figure 3: Example of stochastic file

2.3 Data File Setup

Each random matrix needs to be stored in separate data file. For instance, if b_i is random, then we have a data file called "bi_DB.txt" to store all the scenarios of b_i. In each scenario, there are predictor, response and weight, which is calculated by the choice of nonparametric method. The details of weight calculation will be illustrated later.

```
<database>
<dataset>
Predictor:
0.13539706621957123,-0.8274083386322941,0.21227826726934662;Response:-2.124845657827814,-0.0,-0.0,-0.0,-0.0,-1.8065278242360643,-5.79656447253248,-0.0,-0.2844040795720106,-0.7960767507616789
,-0.0,-0.0;Weight:0.05;
Predictor:
0.2591380898671876,-0.8597828072894333,0.22384255084466065;Response:-0.0,-0.749742980619789,-0.0,-0.0,-0.0,-2.0004697416034007,-0.0,-0.0,-2.3524117183378603,-0.0,-1.4476347169603503,-6.06254
1541874687;Weight:0.05;
Predictor:
0.16229487802854334,-0.891121406710837,0.032137349803448234;Response:-0.0,-0.0,-1.3733255449160222,-6.558537771077723,-0.0,-6.047186007789039,-0.0,-0.0,-0.0,-5.408504957278094,-0.0,-0.0;Weig
ht:0.05;
Predictor:-0.027678136551481797,-0.7960400866592859,0.25619966702434804;Response:-0.0,-0.6478048728232412,-2.11433790696746,-0.0,-0.0,-0.0,-0.0,-0.0,-6.9631457347955745,-0.0,-1.5254798529268
148,-1.3562327993994865;Weight:0.05;
Predictor:
0.2973604256025278,-0.6778859747176953,0.10829744731350724;Response:-0.22821251325558511,-0.0,-1.0970455741790215,-0.0,-0.0,-2.6771066291399537,-0.0,-0.9225683411000194,-3.3700420373138953,-
0.0,-3.6183476239334245,-0.0;Weight:0.05;
Predictor:-0.05687900841422319,-0.8670591535857037,0.246544831782489;Response:-0.7431188771691103,-0.5413617470901877,-0.0,-0.0,-0.0,-1.2905045910336648,-0.0,-0.0,-4.682745054701265,-1.75926
68483529628,-0.0,-0.0;Weight:0.05;
Predictor:0.3664726151766856,-0.679470368164898,0.20553388534184028;Response:-0.0,-0.0,-0.0,-3.5777467750244902,-0.0,-0.0,-0.7588173377686884,-0.0,-0.0,-0.0,-0.0,-0.17787181531595336;Weight:
0.05;
```

Figure 4: Screen shot of data file

2.4 Nonparametric SQG Solver Setup

There are two types of Nonparametirc SQG solver that the user can utilize. One is traditional Nonparametric SQG and another is Robust Nonparametric SQG. As for the traditional Nonparametric SQG, one needs to set up the directory path of target folder. The folder must contain model file, stochastic file and data files. One also needs to set up the maximum number of iterations, initial step size and the initial solution. Infeasible initial solution is allowed.

```
std::vector<double> twoStageLP_random_b_outputResults(const std::string& folder_path, int max_iterations,
    double initial_stepsize, std::vector<double> x_init);
```

Figure 5: Traditional Nonparametric SQG

As for the robust Nonparametric SQG, one needs to set up folder path, maximum number of iterations and initial solution, x_0. One also needs to provide $D_X = \max_{Ax \leq b} ||x - x_0||$, increment of number of the inner loops, which denoted by m, and the upper bound of the norm of the quasi-gradient. Currently, D_X

```
std::vector<double> robustTwoStageLP_random_b_outputResults(const std::string& folder_path, int
    max_iterations, std::vector<double> x_init, double Dx, int m, double M);
```

Figure 6: Robust Nonparametric SQG

can be estimated by the function below. One needs to provide initial solution, lower bound of first stage decision and the upper bound of first stage decision.

```
double Dx_estimateRectangle(const std::vector<double>&x, double lowerBound, double upperBound);
```

Figure 7: Estimation of D_x

2.5 Results Output

There are outer loops and inner loops in the robust Nonaparametric SQG. In one outer loop, there is multiple inner loops. In one inner loop, the estimated solution, estimated quasi-gradient, number of scenarios used and step size are recorded in the output file. After one outer loop is finished, the averaging estimated solution is recorded.

```
*******************************************
Robust Nonparametric Stochastic Quasi-Gradient Method
Fri Jun 14 08:16:31 2019 PDT
Initial solution: 2, 2, 2, 2
Max number of iterations: 5
Main loop
################################################
Outer Loop: Iteration 1
------------------------------------------------
Inner loop: Iteration  1
Number of scenarios: 20
All the subproblems are feasible in the current iteration
Stepsize: 0.536656
Quasigradient: -77.2708, -77.6907, -77.9298, -77.7046
New estimated solution in the subsequence: 43.4678, 43.6932, 43.8215, 43.7007
------------------------------------------------
------------------------------------------------
Inner loop: Iteration  2
Number of scenarios: 20
All the subproblems are feasible in the current iteration
Stepsize: 0.536656
Quasigradient: 5, 5, 5, 5
New estimated solution in the subsequence: 40.7846, 41.0099, 41.1382, 41.0174
------------------------------------------------
------------------------------------------------
Inner loop: Iteration  3
Number of scenarios: 20
All the subproblems are feasible in the current iteration
Stepsize: 0.536656
Quasigradient: 5, 5, 5, 5
New estimated solution in the subsequence: 38.1013, 38.3266, 38.455, 38.3341
------------------------------------------------
------------------------------------------------
Inner loop: Iteration  4
Number of scenarios: 20
All the subproblems are feasible in the current iteration
Stepsize: 0.536656
Quasigradient: 5, 5, 5, 5
New estimated solution in the subsequence: 35.418, 35.6433, 35.7717, 35.6508
------------------------------------------------
------------------------------------------------
Inner loop: Iteration  5
Number of scenarios: 20
All the subproblems are feasible in the current iteration
Stepsize: 0.536656
Quasigradient: 5, 5, 5, 5
New estimated solution in the subsequence: 32.7347, 32.9601, 33.0884, 32.9675
------------------------------------------------
New estimated solution by Robust SQG: 38.1013, 38.3266, 38.455, 38.3341
################################################
```

Figure 8: The output of robust nonparametric SQG

2.6 Validation Cost

Calculating validation cost allows the user to pick the best candidate solution. If validation set is available, one can use the function below to calculate the validation cost of the candidate solution.

```
double twoStageLP_validation_outputResults(const std::string& folder_path, const std::vector<double>&
    x_candidate);
```

Figure 9: Calculating validation cost

```
**************************************************
Estimating the quality of candidate solution
Fri Jun 14 08:18:01 2019 PDT
Candidate solution: 9.06608, 11.8481, 13.0487, 9.7975
Number of data points: 100000
Validation cost: 329.902
**************************************************
```

Figure 10: Output of validation cost

3 Generating Data File

4 Advanced Topics of Nonparametric SQG Solver

5 Numerical Experiment

5.1 Example 1

Here, we set that the target predictor is

$$z = \begin{bmatrix} 0.1702 & -0.8228 & 0.2142 \end{bmatrix} \tag{7}$$

Initial step size is 0.8 and the initial solution is

$$x_0 = \begin{bmatrix} 4 & 4 & 4 & 4 \end{bmatrix} \tag{8}$$

Initial dataset size is 500. Increase of the dataset size per iteration is 50. Total number of datasets is 100. $M = 30$ and $m = 5$. $k = 20$, $\beta = \frac{0.2}{3}$ and $C = 1$. The bandwidth is $h_N = CN^{-\beta}$.

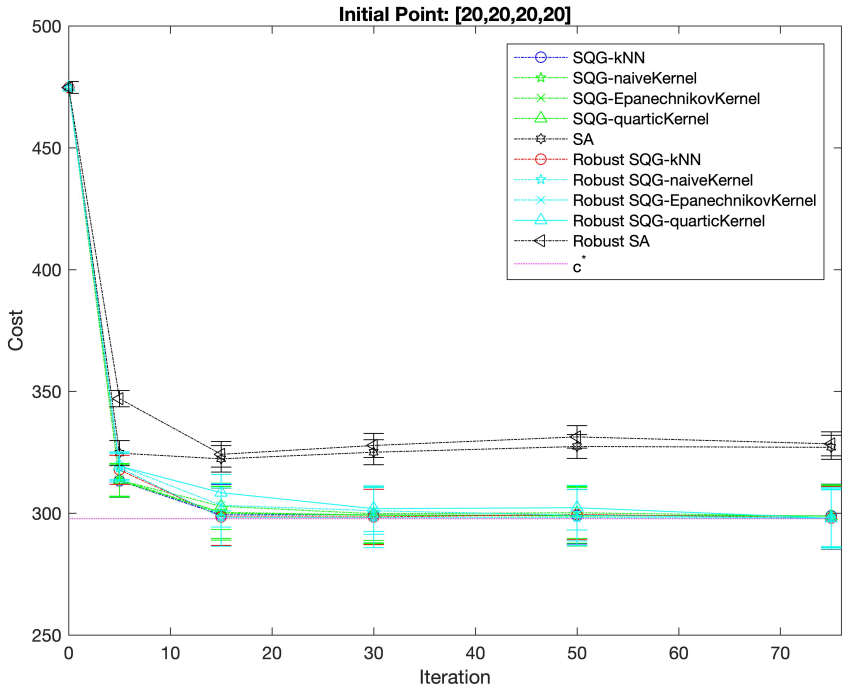

Figure 11: Validation cost of each method

5.2 Example 2

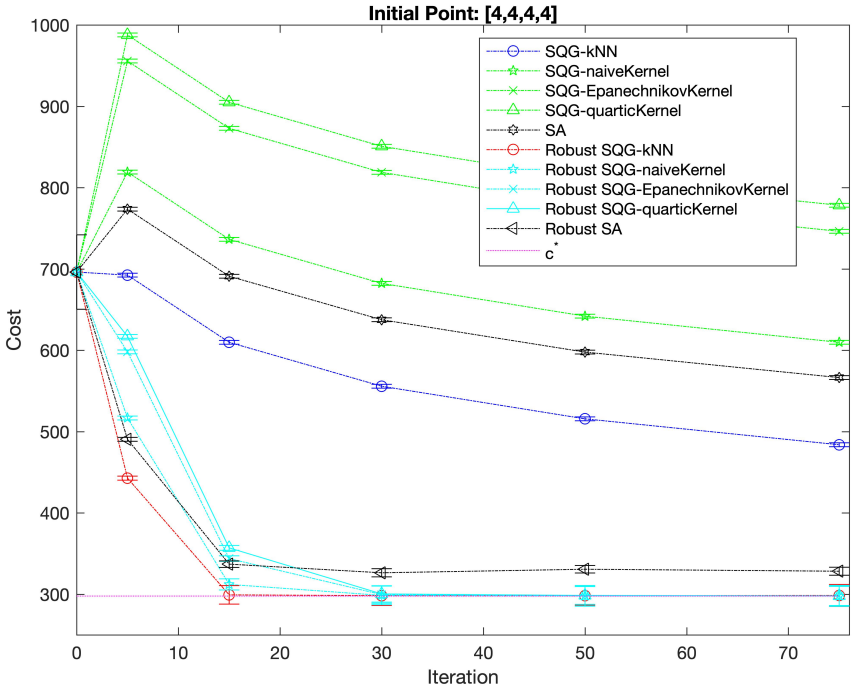

Figure 12: Validation cost of each method

6 Steps in the Numerical Experiment

Step 1: Use sample_generator.py to generate raw datasets. caseNumber is set from 1 to 30.
Step 2: Use dataAnalysisV2.py to do the nonparametric estimation on the raw datasets. Use signTransformation.py to convert the datasets generated in step 2 into solver-readable version.
Step 3: Use sample_generatorSAA.py to generate datasets for the iterative SAA. Use signTransformation_SAA.py to convert the datasets generated in this step into solver-readable version.
Step 4: Use sample_generatorSS.py to generate datasets for the stochastic subgradient method. Use signTransformation_SS.py to convert the datasets into solver-readable version.
Step 5: Use validationSet_generator.py to generate true validation set. Use signTransformation_validation.py to convert the dataset into solver readable

version.
Step 6: Use kNNvalidationSer_generator.py to generate validation set for kNN estimation.
Step 7: Use Nonparametric SQG solver to solve each instance.