Joint PDF Perturbation Robustness Code Documentation

Joseph Hart[1]

[1]Department of Mathematics, North Carolina State University, Raleigh, NC

1 Introduction

This document overviews the use of the codes contained in

Robustness_Codes/Joint_Perturbation

which execute the method described in [1]. The folder *Robustness_Codes/Joint_Perturbation* contains Matlab source code as well as C code generated and precompiled using Matlab Coder. Using the precompiled code is strongly recommended as the algorithm benefits significantly from executing the precompiled C code. The reader is direct to *Robustness_Codes/Manual.pdf* for more information about the precompiled code. Within the directory *Robustness_Codes/Joint_Perturbation* there are two subdirectories whose names are

src and *Linear_Example*

The user does not need to regard the content of *src* unless he/she would like to modify the algorithm. The subdirectory *Linear_Example* may be copied and edited by the user to generate new examples. Section 2 illustrates the use of the codes in *Linear_Example*. Output of the example is given in Section 3.

2 Example

2.1 Test Problem

Let $\mathbf{X} = (X_1, X_2, X_3)$ where each X_i, $i = 1, 2, 3$, is a independent beta random variable on $[0, 1]$ with shape parameters given in Table 1,

	X_1	X_2	X_3
a	1.2	1.4	1.6
b	1.2	1.4	1.6

Table 1: Beta distribution parameters for X_i, $i = 1, 2, 3$. Each X_i is assumed to be a independent beta random variable on $[0, 1]$ with unnormalized probability density function given by $x^{a-1}(1-x)^{b-1}$.

and $f : [0, 1]^3 \to \mathbb{R}$ be defined by

$$f(\mathbf{X}) = 1.5X_1 + 1.25X_2 + X_3. \quad (1)$$

This document demonstrates how to use the codes in *Robustness_Codes/Joint_Perturbation* to compute the total Sobol' indices of (1) and measure their robustness with respect to perturbation of the joint probability density function (PDF). The example is given in

Robustness_Codes/Joint_Perturbation/Linear_Example

The example folder *Robustness_Codes/Joint_Perturbation/Linear_Example* contains a file

Driver.m

which computes the total Sobol' indices and perturbed total Sobol' indices, and a directory

User_Functions

where the user must specifies a collection of Matlab functions. We begin by examining each file in *User_Functions* followed by describing the execution of *Driver.m*. To create a new example, the user is required to implement the functions in *User_Functions*.

2.2 Functions in *User_Functions*

There are six functions in *User_Functions*, two of which draw samples of $\mathbf{X}$, three of which evaluates the joint, marginal, and conditional PDFs of $\mathbf{X}$, and one which evaluations f. In what follows we use p to denote the number of inputs (3 in this example) and N to denote the number of samples used for Monte Carlo integration. The support of $\mathbf{X}$ is assumed to be $[0,1]^p$, see [1] for further discussion of this assumption. When developing new examples, the user should transform $\mathbf{X}$ to ensure that it is supported on $[0,1]^p$.

The file *f.m* defines the function f. It inputs a $N \times p$ matrix and returns a vector of length N corresponding to the evaluations of f at each row of the input matrix. If other input arguments are needed to define f, the user may edit *f.m* and must also edit lines 5 and 8 of the file *Sobol_Function_Evalution.m* (under *src*) accordingly.

The file *Sample_X.m* defines a function which draws samples of the random vector $\mathbf{X}$. It inputs a integer N, specifying the desired number of samples, and returns a $N \times p$ matrix of samples.

The file *Sample_X_Cond_not_k.m* defines a function which draws samples of X_k conditioned on $\mathbf{X}_{\sim k} = \{X_j | j = 1, 2, \dots, p, j \neq k\}$. It inputs a $N \times p$ matrix of samples and an integer k (indicating which variable is being sampled), and returns a $N \times p$ matrix of samples where the k^{th} column of the input matrix is replaced by samples from the conditional distribution of X_k given $\mathbf{X}_{\sim k}$.

The file *Eval_Phi.m* defines the PDF of $\mathbf{X}$. It inputs a $N \times p$ matrix and returns a vector of length N corresponding to the evaluations of the joint PDF at each row of the input matrix.

The file *Eval_Phi_Cond_not_k.m* defines the PDF of X_k conditioned on $\mathbf{X}_{\sim k}$. It inputs a $N \times p$ matrix and an integer k (specifying the conditional distribution of X_k given $\mathbf{X}_{\sim k}$), and returns a vector of length N corresponding to the evaluations of the conditional PDF at each row of the input matrix.

The file *Eval_Phi_Marg_not_k.m* defines the joint PDF of $\mathbf{X}_{\sim k}$, i.e. a $p-1$ dimensional marginal PDF. It inputs a $N \times p$ matrix and an integer k (specifying which variable is excluded) and returns a vector of length N corresponding to the evaluations of the $p-1$ dimensional marginal PDF at each row of the input matrix.

2.3 Execution of *Driver.m*

The script *Driver.m* is copied below for convenient referencing. This subsection explains the script.

```
1 restoredefaultpath
2 clear
3 clc
4 addpath(genpath('../src'))
5 addpath User_Functions/

7 rng(10123)

9 % Set constants
10 N = 5*10^3;
11 L = 50;
12 r = 60;
```

```
13 tau = 1.5;
14 m = 50;

16 % Compute indices, construct partition, precompute PDF evaluations
17 [A,C] = Sobol_Sampling(N);
18 [YA,YC] = Sobol_Function_Evaluation(A,C);
19 Sobol_Output = Sobol_Indices(A,C,YA,YC);
20 sd_nom = Sample_std_Estimate(m,Sobol_Output.YA,Sobol_Output.YC);
21 R = Regression_Tree_Partition(Sobol_Output.A,Sobol_Output.YA,L);
22 Phi_Data = Precompute_Phi_Data(Sobol_Output);
23 save('Preprocessed_Data.mat')

25 % Run perturbation analysis, this can be precompiled
26 % T_pert = Joint_Perturbation_Analysis(Sobol_Output,Phi_Data,sd_nom,R,d,m,tol);
27 T_pert = Joint_Perturbation_Analysis_mex(Sobol_Output,Phi_Data,sd_nom,R,d,m,tol);
28 save('Perturbed_Indices.mat')

30 % Largest_Perturbation(Sobol_Output.T,T_pert)
31 % Largest_Perturbation_Variable_i(Sobol_Output.T,T_pert,1)
```

Lines 1-3 are to ensure a clean workspace and file scope, and Lines 4-5 ensure that the correct functions are in scope. Line 7 sets the random number generator so that the example may be reproduced. These may be edited by the user if he/she desires.

Lines 10-13 are the parameters discussed as inputs of Algorithm 1 in [1]. Line 14 is a parameter specifying the number of random subsamples to generate when estimating the sample standard deviation of the total Sobol' indices.

Lines 17-22 compute the total Sobol' indices and prepare the data needed to perform robustness analysis. Line 17 generates the samples needed to estimate the Sobol' indices. Line 18 is a function which accepts these samples as inputs and evaluates f at each sample. For computationally intensive applications, *Sobol_Function_Evaluation* may be run offline with a computing cluster without effecting any of the robustness codes. Line 19 estimates the total Sobol' indices and Line 20 estimates the sample standard deviation of the estimator. Line 21 generates a partition of the support of $\mathbf{X}$ using a Regression Tree, see [1] for an additional discussion of this. The user may enter their own partition as long as they store R as a $M \times p \times 2$ array as it is generated by *Regression_Tree_Partition*, where M denotes the number of sets in the partition. Line 22 precomputes evaluations of the PDF and conditional PDF prior to the robustness analysis. This enables computational savings, and more importantly, enables the robustness codes to be precompiled independently of the specific example a user considers. Line 23 saves this data, this may be omitted if the user desires.

The user may freely edit anything in Lines 1-23 as long as the data

N, L, r, tau, m, A, C, YA, YC, Sobol_Output, sd_nom, R, Phi_Data

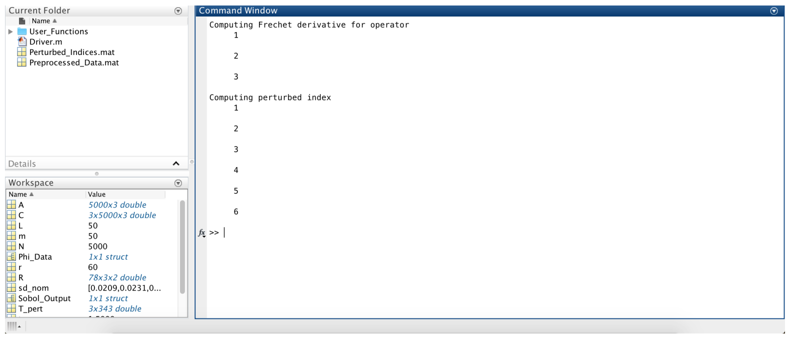
is stored in the same way as the example. The precompiled robustness codes expect the inputs to be formatted as in this example, but the precompiled codes do not depend on any of the functions in Lines 1-23 so the user is free to edit them.

Line 26 (commented out) executes robustness analysis using the basic Matlab codes and Line 27 executes the robustness analysis using the precompiled codes, which is more efficient. The only difference in the function calls is the additional "*_mex*" included in the function name in Line 27. Line 28 saves the results of the robustness analysis.

Line 30 (commented out) plots the largest absolute and relative perturbations of the total Sobol' indices, see [1]. Line 31 (commented out) plots the nominal total Sobol' indices and the perturbed total Sobol' indices with maximize and minimize T_i, where i is the third input argument of the function (1 in this example).

3 Example Output

Figure 1 displays the output when *Driver.m* in *Robustness_Codes/Joint_Perturbation* is executed. The robustness codes print its progress as it executes. Specifically, “Computing Frechet derivative for operator” indicates that the Fréchet derivatives are being computed, the numbers appearing below this line indicate which total Sobol’ index Fréchet derivative is currently being computed. The output “Computing perturbed index” indicates that the Fréchet derivatives have been computed and the code is now computing the perturbed total Sobol’ indices, the numbers below this line indicate which set of perturbed total Sobol’ indices are currently being computed. There may be up to $2p$ (6 since $p = 3$ in this example) sets of perturbed Sobol’ indices, but if multiple perturbations coincide there may be fewer.



Name	Value
A	5000x3 double
C	3x5000x3 double
L	50
m	50
N	5000
Phi_Data	1x1 struct
r	60
R	78x3x2 double
sd_nom	[0.0209,0.0231,0...
Sobol_Output	1x1 struct
T_pert	3x343 double

```
Computing Frechet derivative for operator
     1

     2

     3

Computing perturbed index
     1

     2

     3

     4

     5

     6

>> 
```

Figure 1: Matlab command line output when *Robustness_Codes/Joint_Perturbation/Driver.m* is executed.

Figure 2 displays the figures generated by executing

Largest_Perturbation(Sobol_Output.T,T_pert)

and

Largest_Perturbation_Variable_i(Sobol_Output.T,T_pert,1)

The function *Largest_Perturbation* searches all sets of perturbed total Sobol’ indices and plots the largest absolute perturbation and the largest relative perturbation, alongside the nominal total Sobol’ indices, see [1] for more information. The function *Largest_Perturbation_Variable_i* searches all sets of perturbed total Sobol’ indices and plots the perturbed total Sobol’ indices which maximize and minimize the total Sobol’ index specified by the third input argument (1 in this case), alongside the nominal total Sobol’ indices.

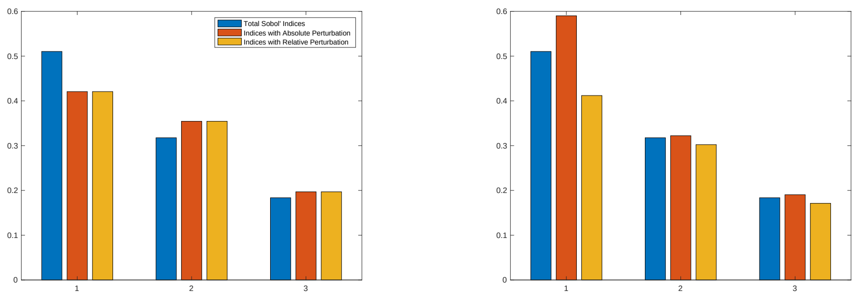


Figure 2: Left: nominal total Sobol’ indices (blue) and perturbed total Sobol’ indices corresponding to the largest absolute perturbation (orange) and the largest relative perturbation (gold). Right: nominal total Sobol’ indices (blue) and perturbed total Sobol’ indices which maximized (orange) and minimized (gold) T_1.

References

[1] Joseph Hart and Pierre Gremaud. Robustness of the Sobol' indices to distributional uncertainty. *https://arxiv.org/abs/1803.11249. Under review*, 2018.