**Instruction of Conducting Morris Sensitivity Analysis for a Single Model**

This file first explains the basics of Morris sensitivity analysis, and the readers can find the details of the Morris method in Chapter 3 of Saltelli et al. (2010). Consider a model, *Y*(**X**), with  parameters. If the parameter space is a unit hypercube, each parameter *Xi* takes values in the set {0, 1/(*p*-1), 2/(*p*-1), …,1}, where *p* is called levels. By using the *p* levels for each of the *X*s, the model parameter space is discretized into a *k*-dimensional *p*-level grid, denoted as Ω. Morris method is based on elementary effects. The elementary effect for the  parameter is defined as,



where , called a grid jump or sampling step, is a predetermined multiple of 1/(*p*-1). In other words,  can take any value in the set of . The selection of  needs to satisfy that  is still in the parameter space, Ω, where **e***i* is a vector of zeros but with a unit as its *i*-th component.

The finite distribution, *Fi*, of elementary effects, *di* ~ *Fi*, associated with the  parameter is obtained by randomly sampling different  from Ω. The number of elements of each *Fi* is . The sensitivity measures,  and  , proposed by Morris (1991) are respectively the estimates of the mean and the standard deviation of the distribution . The mean  assesses the overall influence of parameter *Xi* on model output *Y*. The standard deviation  estimates the ensemble of the parameter’s impacts on *Y* due to nonlinear of the model and/or interactions between *Xi* and other parameters.

Campolongo et al. (2007) proposed to replace **** with , the mean of the distribution, *Gi*, of the absolute values, |*di*|, of the elementary effects. Using  rather than **** reduces the error that may occur when positive and negative elements of the distribution  cancel each other out for computing . In this case, a low *μ* value may be produced even for an important parameter.

1. **Framework of Conductivity Morris Sensitivity Analysis**

The Morris sensitivity analysis can be conducted in the following three steps:

**Step 1**: Generate random trajectories in the parameter space of a model. The random trajectory generation depends on the model that the users are using. We give an example of generating random trajectories for the Sobol G function defined in Section 3.6 of Saltelli et al. (2010). This is implemented in the python file morris\_Sample.py, and the details of the random trajectory generation is described in Section 2 below.

**Step 2**: Run the model for the parameter values in the trajectories. The model run depends on the model that the users are using. We give an example of evaluating the Sobol G function defined in Section 3.6 of Saltelli et al. (2010), as shown in the python file sobol\_G.py.

**Step 3**: Calculate the elementary effects and their mean and standard deviation based on the results of model run in Step 2. This is implemented in the python file morris\_Analyze.py.

This framework is illustrated in the python code, example\_Run.py, for the Sobol G function. For a problem different from the example, the following adjustments are needed to revise the example\_Run.py:

1. Revise step 1 to define the parameter space for the user’s model by changing the parameter number, names, groups, bounds, and distributions. The Morris method working with groups is referred to Section 3.5 of Saltelli et al. (2010). The generation of random trajectories can be done by using function “sample”, as illustrated in the example\_Run.py.
2. Run the model outside of the python code. The model run should use the random parameters (the matrix param\_values in the example) generated in Step 1, and the model run outputs should be stored in vector Y. It should be noted that the elements of Y should correspond to the rows of matrix param\_values.
3. The calculation of the elementary effects and their mean and standard deviation can be done by using the “analyze” function of the example directly.

The scripts of the example are written in Python 3.7 within the Anaconda platform. Most of the codes are adopted form SALib (<https://salib.readthedocs.io/en/latest/>), but several functions were revised to support sampling form nonuniform distribution and any *p* levels.

**References**

Morris, M.D. (1991) Factorial Sampling Plans for Preliminary Computational Experiments. Technometrics 33(2), 161-174.

Saltelli, A., Ratto, M., Andres , T., Campolongo, F. and Cariboni, J. (2007) Global Sensitivity Analysis. The Primer, pp. 109-154.