**GSA++**

**Introduction**

GSA++ is a program in PEST++ suite of tools which performs global sensitivity analysis (GSA). It leverages previous PEST++ work and uses the existing PEST/PEST++ input file formats as well as the PEST++collection of run mangers. This makes it easy to use the powerful PEST++ parallelized run mangers to perform GSA analysis.

GSA techniques are more robust than traditional derivative based local sensitivity analysis methods because they capture sensitivities of model parameters over the entire input parameter domain rather than at a single point. There are many different global sensitivity methods. Some provide general information on the variability of the sensitivities and have relatively low computational requirements while others provide detailed information on interactions between parameters at the expense of larger computational requirements. For an in-depth introduction to GSA methods the reader is referred to Saltelli et el (2004) and (2008). GSA++ currently provides support for the Method of Morris Morris(1991) with extensions to this method proposed by Campolongo et al (2003) and Sin (2009) was well as Sobol’s method. Typically the Method of Morris of Morris is used in screening level analysis to find the most important parameters, and then Sobol’s method is performed on the most important parameters to analyze the effects of parameter interactions. Sobol’s Method is based on the decomposition of variance and can provide information on how parameter interactions affect the sensitivity but it has a high computational burden.

**Implementation of the Method of Morris**

The Method of Morris, Morris (1991) and Satelli (2004 and 2008) is a global sensitivity analysis method because it provides estimates of parameter sensitivities over the entire parameter domain. It is a “One at a Time” or OAT method as it varies one parameters at a time to compute the sensitivities. The method, originally proposed by Morris in his 1991 paper provides two measurements of parameter sensitivities: the mean (μ) and the standard deviation (σ). The mean measures the central tendency and captures the overall effect of a parameter where as the standard derivation represents the spread of the sensitivities around the mean and is an indicator of nonlinearities and parameter interactions. Compololongo et al. (2003) extended the method of Morris by adding the absolute mean (μ\*) as a metric. This provides a better estimate of the overall impact of parameters than the mean as it is not subject to canceling out of positive and negative values.

The method of Morris is based on elementary effects which are sensitivities to parameters rescaled to the interval [0,1]. Morris (1991), defines an elementary effect as:

Where k is the number of parameters and x is a k dimensional vector which contains the scaled parameters. Each member of x is drawn from the set 0, 1/(p-1), 2/(p-1),….,1} where p is a variable that defines the number of intervals used for each parameter and represents the size of the perturbation used on scaled parameters. Morris (1991) recommends choosing an even number for p and setting.

After a parameter perturbed, the method of Morris uses equation 1 to compute the elementary effects. A detailed account of how this is performed can be found in Morris’ original paper Morris (1991) as well as Saltelli (2004). Briefly the parameters are perturbed one at a time and to provide a trajectory through parameter space which can then be used to compute the sensitivity of a single observation with respect to each of the scaled parameter. The sample size, r, is then defined to be the number of trajectories used in the analysis and represents the number of times the sensitivities are computed for each parameter. Table XX summarizes the important variables in the method of Morris.

|  |  |  |
| --- | --- | --- |
| Variable | input/output | Description |
| p | input | Number of levels or the number of point each parameter is sampled at. |
|  | input | Size of the sampling step. This must be a multiple of and represent the size of the interval that will be used to calculate the sensitivities. |
| r | input | Sample size. The number of times the sensitivity will be computed for each parameter |
| μ | output | Mean sensitivity of the output with respect to each parameter. |
| μ\* | output | Mean sensitivity of the absolute value of the output with respect to each parameter. |
| σ | output | Standard deviation of the sensitivity of the output with respect to each parameter. |

The method of Morris can only compute the sensitivity of a single model output with respect to the input parameters. It is not designed to compute the sensitivities of multiple outputs. The method of Morris implementation in GSA++ uses the PEST/PEST++ objective function PHI for the single model output required by the method of Morris.

While the Method of Morris can provide valuable information, only being able to perform the analysis on a single model output can be a severe limitation. Because GAS++ is built on top of PEST++, it has access to the sensitivities of the individual observations used to construct the composite objective function. As this information, is readily available to GSA++ it makes sense to look for ways to extend the method of Morris to include all the sensitivities of all the individual observations in the analysis as this information is readily available and does not require any additional model run. Sin (2009) proposes a new metric the called Standardized Elementary Effects (SEE) to provide a Method of Morris like analysis for cases where it is desirable to include multiple model outputs in the analysis. As shown in equation 2 Sin proposes using the standard deviations of the parameters and model outputs to scale the sensitivities.

In equation 2, is the standardized elementary effect of parameter xi on model output yj , represents the sensitivity of model output yj with respect to parameter xi, represents the standard deviation of parameter xi, represents the standard deviation of model output yj.

While Equation 2 provides a means of scaling the sensitivities of the individual model outputs, it may not be ideal for analyzing a typical water resource problem as it rescales the responses of the individual model outputs and when analyzing a water resource problem one may be interest in knowing which model outputs are the most sensitive. A better solution would be to identify different types of model outputs. Say heads of flows and provide a method to scale these different types that allows for variability within the types.

NEED TO FINISH THIS AND TALK ABOUT POOLED VARIANCE BUT YOU PROBABLY DON”T NEED THIS OPTION

**Implementation of Sobol’s Method**

The Method

**Running GSA++**

GSA++ shares a common command line with PEST++ as well as the input control, template files and instruction file developed for PEST and used by PEST++. The command line specifies the name of the control file as well as the run manger. The control file (.pst) file specifies the parameters and observation to be used in the analysis and provides the information GSA++ needs make the forward runs. The command line for GSA++ as well as the \*.pst input file are briefly summarized below and the reader is referred to the PEST++ and PEST documentation for additional information. Variables specific to Global Sensitivity Analysis are stored in a new file with a .gsa suffix which is covered later in this section.

GSA++ provides a number of command line options to specify how to start a run using one of the supported run managers. Large problems (defined as having many parameters) often require parallel computing. GSA++ relies on run mangers to complete the forward model runs and the current version provides the following three options: 1) **Y**et **A**nother Run **M**anage**R** (**YAMR**), 2) GENIE and 3) serial run manager. YAMR and GENIE are sophisticated and capable of performing parallel runs on a single machine or over a TCP/IP-enabled network. YAMR duplicates the functionality of BEOPEST and is fully integrated in PEST++ and GSA++. When using the YAMR or GENIE parallel run managers GSA++ relies on PEST++ run in slave mode to perform model runs for YAMR and the external GMAN and GSLAVE programs to manage and perform the model runs for GENIE. The serial rule manager provides a simple alternative that duplicates the functionality currently in regular PEST. The command lines required to start GSA++ are summarized in the following table.

|  |  |
| --- | --- |
| **Run Manger / Mode** | **Command** |
| Serial Run Manager / Master | GSA++ control\_file.pst |
| YAMR / Master | GSA++ control\_file.pst /H :port |
| YAMR / Compute Node | pest++ /H hostname:port |
| Genie / Master | GSA++ control\_file.pst /G hostname:port |

In addition, GSA++ provides support for resuming a run that failed due to network or system problems through the /r command line option.

Information specific to the method of Morris must be specified in a file that has the same base name the pest control file, but with a “.gsa” extension. The variables in this file are listed below.

METHOD(MORRIS)

MORRIS\_R(4)

MORRIS\_P(4)

MORRIS\_DELTA(.666)

MORRIS\_POOLED\_OBS(FALSE)

PHI\_NORM(1)

SOBOL\_SAMPLES(4000)

General Options

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | Values | Description |
| METHOD | text | “MORRIS” | Specifies type of analysis to be performed. The Method of Morris is the only option currently supported. |
| RAND\_SEED | Unsigned integer |  | Seed for the random number generator |

Options Specific to Method of Morris

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | Values | Description |
| MORRIS\_R | integer | positive integer | Sample size. The number of times the sensitivity will be computed for each parameter |
| MORRIS\_P | integer | positive integer | Number of levels or the number of point each parameter is sampled at. |
| MORRIS\_DELTA | real | multiple of  where p=MORRIS\_P | Size of the sampling step. This must be a multiple of and represent the size of the interval that will be used to calculate the sensitivities. |
| MORRIS\_POOLED\_OBS | text | “TRUE” or “FALSE” |  |
| PHI\_NORM | integer | positive integer | Typically PEST++ computes the objective function from the observations using the L2 or Euclidian norm. This option can be used to specify a different norm. Sometimes it is useful to use this option to specify the L1 norm. |

Options Specific to Sobol’s MEthod

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | Values | Description |
| SOBOL\_SAMPLES | long integer | positive integer | Number of sample to be used for Sobol’s method |
| SOBOL\_PAR\_DIST | string | “NORM”  “UNIF” |  |

GSA++ Method of Morris Output Files

The GSA++ implementation produces the following output files.

1. .msn – Morris Sensitivity file

This is the primary output file which contains the metric associated with the method of Morris analysis. The output for an example problem is included below. The file contains a header line followed by a line with containing the metrics computed for each parameter.

*parameter\_name, sen\_mean, sen\_mean\_abs, sen\_std\_dev*

*X1, -16.4665, 108.885, 138.542*

*X2, 53.5115, 72.4633, 98.2834*

*.*

*.*

*.*

*X19, -2.32365, 5.56711, 6.42093*

*X20, -0.0338625, 2.83062, 3.88215*

The information contained in this file is described in header line and is pretty much self explanatory as shown below:

parameter\_name: the name of the parameter

sen\_mean: mean sensitivity (μ)

sen\_mean\_abs: mean absolute sensitivity (μ\*)

sen\_std\_dev: standard deviation of the sensitivity (σ)

1. .raw – raw sensitivity file

This file summarizes the raw outout that was used to compute the information stored in the .msn file. Each line stores a single sensitivity was computed from a pair of model runs.

*parameter\_name, phi\_0, phi\_1, par\_0, par\_1, sen*

*X1, 128.437, 84.7042, 0.999999, 0.333333, 65.5993*

*X2, 114.144, 128.437, 0.666666, 0, -21.4395*

*.*

*.*

*.*

The information contained in this file is described in header line and is pretty much self explanatory as shown below:

parameter\_name: the name of the parameter

phi\_0: phi value produced by the first model run

phi\_1: phi value produced by the second model run

par\_0: value of the adjustable parameter used to make the first model run

par\_1: value of the adjustable parameter used to make the second model run