**GSA++**

**Introduction**

GSA++ is a component of the PEST++ suite of tools designed to facilitate global sensitivity analysis (GSA). It leverages previous PEST++ work and uses the existing PEST/PEST++ input files and the PEST++collection of run mangers. This makes GSA analysis on a problem set-up for PEST and/or PEST++ trivial and it can also be extremely efficient if one of the parallel run managers is used.

GSA techniques are more robust than traditional derivative based local sensitivity analysis methods because they capture the sensitivities of model parameters over the entire input parameter space rather than just at a single point. There are many different global sensitivity methods. Some provide general information on the variability on the sensitivities and have relatively low computational requirements while others can provide detailed information about how interactions between parameters affect sensitivity and have much 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). There is also an effort underway to also incorporate the Sobol’s method into GSA++. Sobol’s Method, which uses the decomposition of variance, can provide information on how parameter interactions affect the sensitivity but it has much higher computational requirements than the method of Morris. 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.

**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 a single parameter 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. It provides a better estimate of the overall impact of parameters than the mean because 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 that have been rescaled to the interval [0,1]. Morris (1991), defines elementary effects as:

Where x is the k dimensional vector x is defined to contain the scaled parameters where each member of x is drawn from the set 0, 1/(p-1), 2/(p-1),….,1} where p is a variable that defines how many intervals should be used for each parameter and represents the size of the perturbation of the scaled parameters used in the analysis. Morris (1991) recommends choosing an even number for p and setting.

The method of Morris perturbs one parameter at a time and uses equation 1to compute the sensitivity. 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 be used to compute the sensitivity of a single observation with respect to each 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 sensitivity is 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 summaries the sensitivity of a single model output with respect to a number of input parameters and GSA++ uses the PEST/PEST++ objective function or phi as this output.

**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)

|  |  |  |  |
| --- | --- | --- | --- |
| Variable | Type | Values | Description |
| METHOD | text | “MORRIS” | Specifies type of analysis to be performed. The Method of Morris is the only option currently supported. |
| 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. As the method of Morris is not normally performed on a composite observation base on the sum of squares this option can be used to specify the L1 norm. |

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