Some Examples of `polychaosbasics` Use

J.P. Gauchi and A. Bouvier

MaIAGE, INRA, Université Paris-Saclay,
78350 Jouy-en-Josas,
France

March 10, 2016

Abstract

The **polychaosbasics** R package computes sensitivity indexes from polynomial chaos expansions. The functionalities and method are explained in the in-line manual of the package and in [Gauchi (2016)]). The necessary condition of the method is: the inputs must be uniformly and independently sampled. Since the inputs are uniformly distributed, the truncated PCE is built from the multivariate Legendre orthogonal polynomials.

This paper illustrates the way of using the package. Each example operates on one of the three possible kinds of datasets:

- a dataset simulated by using the Ishigami function ([Ishigami (1990)]),
- a dataset simulated by using the Sobol' function, ([Sudret (2008)]),
- a provided dataset. We use here the dataset FLORSYS1, which enables to build a submodel of the 3D light interception computer model exposed in [Gauchi (2016)].

Contents

1 Dataset simulated by the Ishigami function

Brief reminder of the Ishigami function

The Ishigami function has three inputs $\mathbf{X} = (X_1, X_2, X_3)$ that are linked to the output Y according to:

$$Y = \sin(X_1) + \theta_1[\sin(X_2)]^2 + \theta_2 X_3^4 \sin(X_1)$$

with $\theta_1 = 7$, and $\theta_2 = 0.1$, given in [Ishigami (1990)]. Each X_j is a uniform random variable on the interval $[-\pi; \pi]$.

The way of doing with the polychaosbasics package

We operate in three steps.

1. Build the PCE design by using the function `analyticsPolyLeg`.

```
> library("polychaosbasics", lib.loc="~/RLIBRARY/")
> degree <- 6 # polynomial degree
> nlhs <- 20000
> set.seed(42)
> pce <- analyticsPolyLeg(nlhs, degree, 'ishigami')
> print(pce)

Number of monomials:  83
Number of factors:  3
Polynomial degree:  6
Number of observations: 20000
```

2. Calculate the PCE Sensitivity Indexes by using the function `PCESI`.

```
> retour <- PCESI(pce)
```

To see the names of all the components included in the returned object, even those of the components that are hidden by default, we set the option `all` in the `print` command.

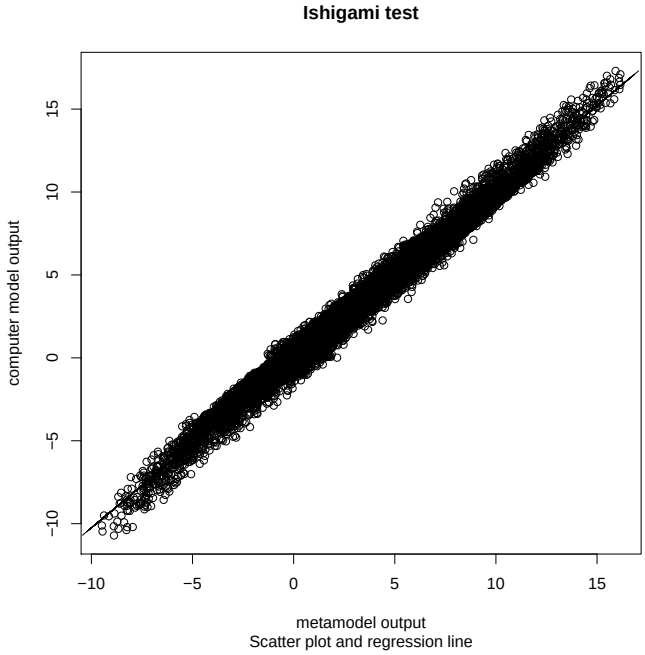
```
> print(retour, all=TRUE, digits=4)

PCE indexes:
         LE        PE    TPE
1 1.950e-01 3.198e-01 0.5581
2 1.646e-07 4.419e-01 0.4419
3 4.257e-06 1.117e-05 0.2383

PCE indexes (percentages):
  %LE   %PE  %TPE
1 100 41.99 45.07
2   0 58.01 35.69
3   0  0.00 19.24
```

```
PCE fit:
     R2   RMSEP
 0.9818  0.5059

Number of monomials:  83
Number of factors:  3
Polynomial degree:  6
Number of observations: 20000

Also included:
 * slot 'IMSI' (Individual Monomial Sensitivity Indexes). Length: 84
 * slot 'coef' (Regression coefficients). Length: 84
 * slot 'y.hat' (Metamodel output). Length: 20000
 * slot 'call.PCEpoly' (Design creation command).
```

The analytical values of the *PE* for the independent inputs are 0.3138, 0.4424 and 0, and the analytical values of the *TPE* are 0.5574, 0.4424 and 0.2436. Our results, 0.3198, 0.4418, 0.1e-6 for the *PE* and 0.5581, 0.4419, 0.2383 for the *TPE*, are very close to these values.

3. Plot the computer model output against the metamodel output:

```
> y.hat <- retour@y.hat
> y.obs <- pce[, "Y"]
> plot(y.hat, y.obs,
+       xlab="metamodel output", ylab="computer model output",
+       main="Ishigami test", sub="Scatter plot and regression line")
> # Add the regression line
> reg <- lm(y.hat ~ y.obs)
> lines(reg$fitted.values, y.obs)
```



2 Dataset simulated by the Sobol' function

Brief reminder of the Sobol' function

The Sobol' function has eight inputs that are linked to the output Y according to:

$$Y = \prod_{j=1}^{8} \frac{|4X_j - 2 + a_j|}{1 + a_j}$$

with $\mathbf{a} = (1, 2, 5, 10, 20, 50, 100, 500)$, given in [Sudret (2008)].

Each X_j is a uniform random variable on the interval $[0; 1]$.

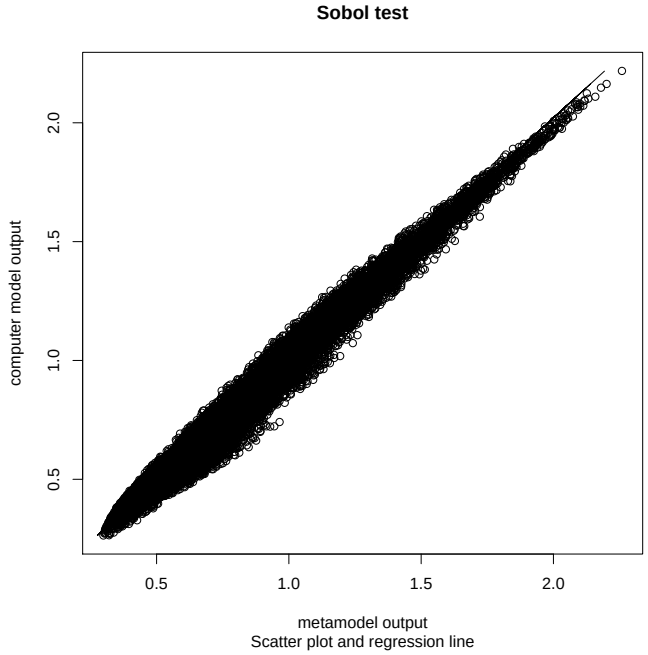
Since the last four `PE` and `PET` are close to zero (see [Gauchi (2016)]), the variables X_j, $j = 5, ..., 8$, were set at the value of $1/2$ (i.e., their mean value).

The way of doing with the polychaosbasics package

The process is similar to the previous one.

```
> # 1. Build the PCE design
> degree <- 5
> nlhs <- 20000
> set.seed(42)
> pce <- analyticsPolyLeg(nlhs, degree, 'sobol')
> print(pce)

Number of monomials:  125
Number of factors:  4
```

```
Polynomial degree:  5
Number of observations: 20000

> # 2. Calculate the PCE Sensitivity Indexes
> retour <- PCESI(pce)
> print(retour, digits=4)

PCE indexes:
         LE      PE     TPE
1 6.052e-07 0.60938 0.63592
2 5.779e-07 0.27229 0.29533
3 9.995e-08 0.06859 0.07592
4 1.447e-07 0.02012 0.02251

PCE indexes (percentages):
    %LE   %PE  %TPE
1 42.39 62.80 61.76
2 40.48 28.06 28.68
3  7.00  7.07  7.37
4 10.13  2.07  2.19

PCE fit:
     R2   RMSEP
0.98101 0.04742


> # 3. Plot the computer model output against the metamodel output
> y.hat <- retour@y.hat
> y.obs <- pce[, "Y"]
> plot(y.hat, y.obs,
+       xlab="metamodel output", ylab="computer model output",
+       main="Sobol test", sub="Scatter plot and regression line")
> # Add the regression line
> reg <- lm(y.hat ~ y.obs)
> lines(reg$fitted.values, y.obs)
```



The analytical values of the *PE* for the independent inputs are 0.6037, 0.2683, 0.0671 and 0.0200, and the analytical values of the *TPE* are 0.6342, 0.2945, 0.0756 and 0.0227. Once again, our results, 0.6093, 0.2723, 0.0686 and 0.0201, for the *PE* and 0.6359,0.2953, 0.0759, 0.0225, for the *TPE*, are very close to these values.

3 FLORSYS1 dataset

The dataset has 11 inputs and 9735 rows. It is stored in a file. The last column is the model outputs.

We first read the file.

```
> lhdnat <- read.table('FLORSYS1.txt')
```

The way of doing with the polychaosbasics package

We analyze the dataset by increasing progressively the polynomial degree from 2 until a value such as `R2` is close to 1 and `RMSEP` is low.

Polynomial degree 2

1. Build the PCE design by using the function `polyLeg`.

```
> nvx <- ncol(lhdnat) -1 # number of inputs
> Y <- lhdnat[,ncol(lhdnat)] # model outputs
> lhdnat <- lhdnat[, 1:nvx]
> degree <- 2
```

```
> pce <- polyLeg(lhdnat, Y, degree)
> print(pce)

Number of monomials:  77
Number of factors:  11
Polynomial degree:  2
Number of observations: 9735
```

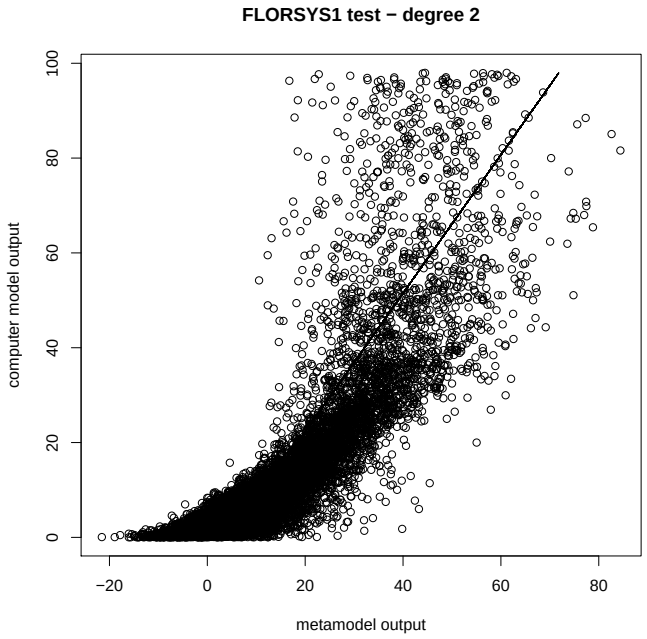
2. Calculate the PCE Sensitivity Indexes by using the function `PCESI`.

```
> retour <- PCESI(pce)
> print(retour, digits=4)

PCE indexes:
          LE        PE       TPE
1  6.930e-04 0.0119106 0.0127394
2  7.797e-07 0.0002733 0.0004329
3  8.983e-05 0.0003755 0.0007729
4  1.582e-04 0.0003393 0.0020790
5  1.802e-01 0.1822021 0.2629839
6  2.666e-01 0.3667424 0.4157418
7  2.099e-01 0.2605223 0.2985077
8  2.901e-02 0.0377647 0.0534424
9  2.609e-02 0.0318041 0.0465881
10 8.648e-04 0.0030035 0.0053359
11 1.836e-03 0.0018369 0.0046010

PCE indexes (percentages):
     %LE   %PE  %TPE
1   0.10  1.33  1.15
2   0.00  0.03  0.04
3   0.01  0.04  0.07
4   0.02  0.04  0.19
5  25.19 20.32 23.84
6  37.26 40.90 37.68
7  29.34 29.05 27.06
8   4.05  4.21  4.84
9   3.65  3.55  4.22
10  0.12  0.33  0.48
11  0.26  0.20  0.42

PCE fit:
       R2     RMSEP
   0.6849   10.5628
```

3. Plot the computer model output against the metamodel output:

```
> y.hat <- retour@y.hat
> y.obs <- pce[, "Y"]
> plot(y.hat, y.obs,
```

```
+       xlab="metamodel output", ylab="computer model output",
+       main="FLORSYS1 test", sub="Scatter plot and regression line - degree 2")
> # Add the regression line
> reg <- lm(y.hat ~ y.obs)
> lines(reg$fitted.values, y.obs)
```

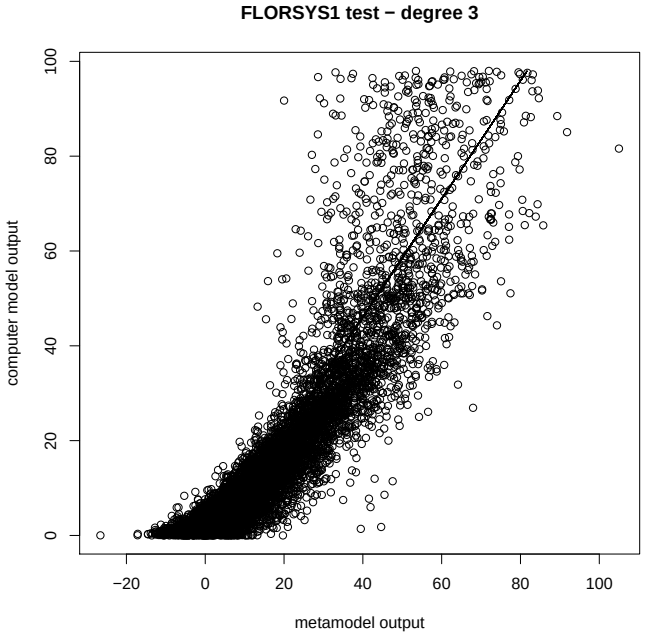


The values of `R2` (0.68) and `RMSEP` (10.56) are not satisfying. The plot reveals a bad fitting. So, we process again with a polynomial of degree 3.

Polynomial degree 3

```
> # 1. Build the PCE design
> degree <- 3
> pce <- polyLeg(lhdnat, Y, degree)
> print(pce)

Number of monomials:  363
Number of factors:  11
Polynomial degree:  3
Number of observations: 9735

> # 2. Calculate the PCE Sensitivity Indexes
> retour <- PCESI(pce, digits=4)
> print(retour)

PCE indexes:
          LE        PE      TPE
1  4.389e-04 0.0096359 0.032725
```

```
2  9.213e-08 0.0002611 0.008321
3  6.331e-05 0.0002969 0.002441
4  3.143e-04 0.0006102 0.002913
5  1.518e-01 0.1531346 0.249189
6  2.398e-01 0.3693097 0.449364
7  1.831e-01 0.2370309 0.302430
8  2.429e-02 0.0325594 0.061786
9  2.106e-02 0.0274092 0.053465
10 2.703e-04 0.0026024 0.009260
11 1.570e-03 0.0015882 0.007299

PCE indexes (percentages):
     %LE   %PE  %TPE
1   0.07  1.15  2.78
2   0.00  0.03  0.71
3   0.01  0.04  0.21
4   0.05  0.07  0.25
5  24.37 18.35 21.13
6  38.52 44.26 38.11
7  29.40 28.41 25.65
8   3.90  3.90  5.24
9   3.38  3.28  4.53
10  0.04  0.31  0.79
11  0.25  0.19  0.62

PCE fit:
      R2    RMSEP
  0.8036   8.6620


> # 3. Plot the computer model output against the metamodel output
> y.hat <- retour@y.hat
> y.obs <- pce[, "Y"]
> plot(y.hat, y.obs,
+       xlab="metamodel output", ylab="computer model output",
+       main="FLORSYS1 test", sub="Scatter plot and regression line - degree 3")
> # Add the regression line
> reg <- lm(y.hat ~ y.obs)
> lines(reg$fitted.values, y.obs)
```



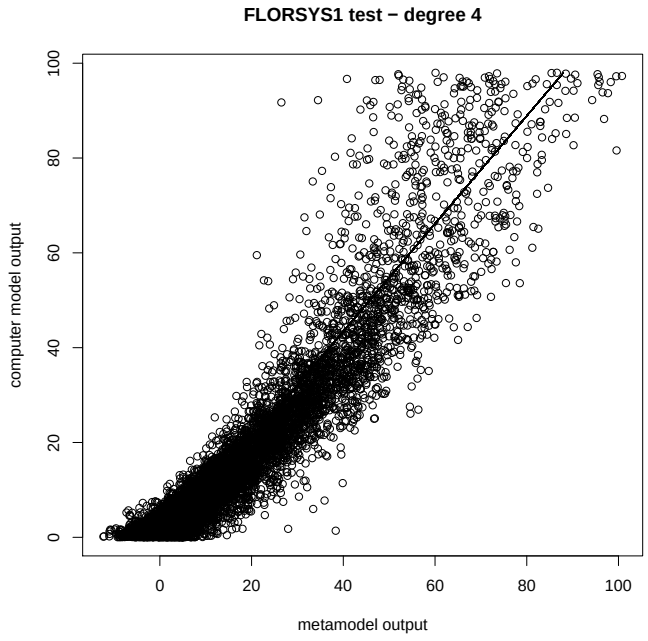
Once again, the results are rather bad. Try with a polynomial of degree 4.

Polynomial degree 4

```
> # 1. Build the PCE design
> degree <- 4
> pce <- polyLeg(lhdnat, Y, degree)
> # 2. Calculate the PCE Sensitivity Indexes
> retour <- PCESI(pce)
> print(retour, digits=4)

PCE indexes:
          LE        PE      TPE
1  4.327e-04 0.0091884 0.051652
2  4.256e-06 0.0003581 0.021360
3  8.407e-05 0.0001984 0.006524
4  1.191e-04 0.0003237 0.007115
5  1.342e-01 0.1355105 0.241383
6  2.232e-01 0.3707847 0.477355
7  1.607e-01 0.2107542 0.301198
8  2.254e-02 0.0303896 0.071592
9  1.939e-02 0.0247394 0.062887
10 3.688e-04 0.0022171 0.015224
11 1.554e-03 0.0015912 0.012042

PCE indexes (percentages):
     %LE   %PE  %TPE
1   0.08  1.17  4.07
```

```
2   0.00  0.05  1.68
3   0.01  0.03  0.51
4   0.02  0.04  0.56
5  23.86 17.24 19.03
6  39.68 47.17 37.64
7  28.56 26.81 23.75
8   4.01  3.87  5.64
9   3.45  3.15  4.96
10  0.07  0.28  1.20
11  0.28  0.20  0.95

PCE fit:
      R2    RMSEP
  0.8797   7.8725
```

```
> # 3. Plot the computer model output against the metamodel output
> y.hat <- retour@y.hat
> y.obs <- pce[, "Y"]
> plot(y.hat, y.obs,
+       xlab="metamodel output", ylab="computer model output",
+       main="FLORSYS1 test", sub="Scatter plot and regression line - degree 4")
> # Add the regression line
> reg <- lm(y.hat ~ y.obs)
> lines(reg$fitted.values, y.obs)
```

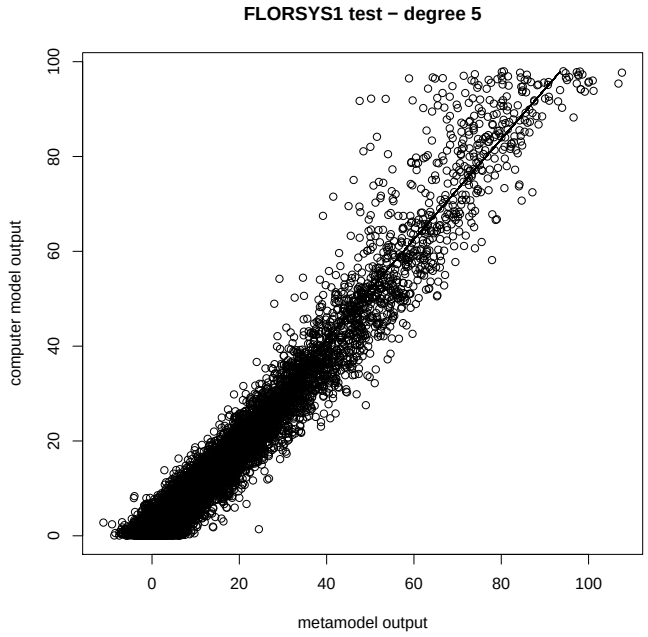


Once again, the results are rather bad. Try with a polynomial of degree 5.

Polynomial degree 5

```
> # 1. Build the PCE design
> degree <- 5
> pce <- polyLeg(lhdnat, Y, degree)
> # 2. Calculate the PCE Sensitivity Indexes
> retour <- PCESI(pce)
> print(retour, all=TRUE, digits=4)
PCE indexes:
          LE        PE     TPE
1  6.895e-04 8.143e-03 0.08813
2  2.095e-05 4.139e-04 0.05186
3  2.314e-05 7.091e-05 0.02849
4  2.153e-04 2.667e-04 0.03096
5  1.168e-01 1.181e-01 0.24826
6  2.032e-01 3.435e-01 0.48279
7  1.376e-01 1.804e-01 0.30926
8  1.876e-02 2.508e-02 0.09482
9  1.650e-02 2.227e-02 0.08685
10 1.810e-04 1.413e-03 0.03627
11 9.777e-04 1.117e-03 0.03612

PCE indexes (percentages):
     %LE   %PE  %TPE
1   0.14  1.16  5.90
2   0.00  0.06  3.47
3   0.00  0.01  1.91
4   0.04  0.04  2.07
5  23.59 16.85 16.62
6  41.05 49.02 32.32
7  27.81 25.74 20.70
8   3.79  3.58  6.35
9   3.33  3.18  5.81
10  0.04  0.20  2.43
11  0.20  0.16  2.42

PCE fit:
    R2  RMSEP
0.9468 9.7083

Number of monomials:  4367
Number of factors:  11
Polynomial degree:  5
Number of observations: 9735

Also included:
 * slot 'IMSI' (Individual Monomial Sensitivity Indexes). Length: 4368
 * slot 'coef' (Regression coefficients). Length: 4368
 * slot 'y.hat' (Metamodel output). Length: 9735
 * slot 'call.PCEpoly' (Design creation command).
```

```
> # 3. Plot the computer model output against the metamodel output
> y.hat <- retour@y.hat
> y.obs <- pce[, "Y"]
> plot(y.hat, y.obs,
+       xlab="metamodel output", ylab="computer model output",
+       main="FLORSYS1 test", sub="Scatter plot and regression line - degree 5")
> # Add the regression line
> reg <- lm(y.hat ~ y.obs)
> lines(reg$fitted.values, y.obs)
```



The results ($R2 = 0.947$ and $RMSEP = 957.882$ are now satisfying. We stop the process.

The execution time, proportional to the number of monomials, was about 4 hours on the computer whose characteristics are detailed in Section 5.

4 Display methods

Functions `polyLeg` and `analyticsPolyLeg` return an object from class `PCEpoly`. Function `PCESI` returns an object from class `PCEfit`. Methods are defined for these classes:

- `getNames`. Display the names, class and length of all the components.
- `print`. When its additional argument `all` is set to TRUE, display more about the content of the object.

- `show`. Same as `print` without any argument.

Example:

1. Build a PCEpoly object

```
> set.seed(42)
> pce <- analyticsPolyLeg(100, 2, 'ishigami')
```

- Standard display:

  ```
  > pce # same as 'show(pce)'

  Number of monomials:  9
  Number of factors:  3
  Polynomial degree:  2
  Number of observations: 100
  ```

- To know more about the polynome structure:

  ```
  > print(pce, all=TRUE)

  Polynomial expression:
  0 + 3 + 2 + 1 + 3*3 + 2*3 + 2*2 + 1*3 + 1*2 + 1*1
  Number of monomials of degree 1: 3
  Number of monomials of degree 2: 6
  Total number of monomials:  9
  Number of factors:  3
  Polynomial degree:  2
  Number of observations: 100
  Created by:
  analyticsPolyLeg(nlhs = 100, degree = 2, model.fun = "ishigami")
  ```

  The structure of the polynomial is expressed by using the inputs numbers. The zero is for the constant term.

- To get a description of all the 'slots' (i.e components) of the returned object:

  ```
  > getNames(pce)

   Slot: .Data. Class: "matrix". Dimension: (100, 11). Legendre polynomial
   Slot: design. Class: "PCEdesign". Dimension: (10, 3). The polynomial structure
   Slot: nvx. Class: "numeric". Length: (1). The number of inputs
   Slot: call. Class: "call". Length: (4). The command which creates the object
  ```

  The slot `.Data` is a matrix which contains the calculated values of the monomials in the Legendre polynomial. The slot `design` is an object of class `PCEdesign` which stores the polynomial structure.

2. Calculate the PCE sensitivity indexes. A `PCEfit` object is returned.

```
> retour <- PCESI(pce)
```

- Standard display:

```
> retour
```

```
PCE indexes:
          LE        PE       TPE
1 0.54971659 0.6306712 0.7583916
2 0.06816811 0.1076586 0.2020806
3 0.04833432 0.1119258 0.1892721

PCE indexes (percentages):
    %LE   %PE  %TPE
1 82.51 74.17 65.96
2 10.23 12.66 17.58
3  7.26 13.16 16.46

PCE fit:
       R2     RMSEP
0.2059645 3.6032743
```

- To see more:

```
> print(retour, all=TRUE)
```

```
PCE indexes:
          LE        PE       TPE
1 0.54971659 0.6306712 0.7583916
2 0.06816811 0.1076586 0.2020806
3 0.04833432 0.1119258 0.1892721

PCE indexes (percentages):
    %LE   %PE  %TPE
1 82.51 74.17 65.96
2 10.23 12.66 17.58
3  7.26 13.16 16.46

PCE fit:
       R2     RMSEP
0.2059645 3.6032743

Total number of monomials:  9
Number of factors:  3
Polynomial degree:  2
Number of observations: 100

Also included:
 * slot 'IMSI' (Individual Monomial Sensitivity Indexes). Length: 10
 * slot 'coef' (Regression coefficients). Length: 10
 * slot 'y.hat' (Metamodel output). Length: 100
 * slot 'call.PCEpoly' (Design creation command).
```

- Another way to see which are the components (or 'slots') stored in the object:

```
> getNames(retour)
```

```
Slot: indexes. Class: "matrix". Dimension:(3, 3). PCE indexes
Slot: indexes.percent. Class: "matrix". Dimension:(3, 3). Percentages of PCE indexes
Slot: fit. Class: "numeric". Length:(2). R2 and RMSEP Root Mean Square Error Prediction)
Slot: IMSI. Class: "numeric". Length:(10). Individual Monomial Sensitivity Indexes
Slot: coef. Class: "numeric". Length:(10). Regression coefficients
Slot: y.hat. Class: "numeric". Length:(100). Fitted values of the response
Slot: design. Class: "PCEdesign". Dimension:(10, 3). The polynomial structure
Slot: call.PCEpoly. Class: "call". Length:(4). The command which creates the input design
```

- To access a component whose values are not displayed by default, use the R notation arobas. For example, the regression coefficients are displayed by the command:

```
> print(retour@coef)
```

```
         0          3          2          1        3*3        2*3        2*2
 3.7139759  0.6472163 -0.7686211  2.1826850  0.9490032 -0.7488073 -0.7478496
       1*3        1*2        1*1
-1.2474699  1.3042017 -1.0707522
```

5 Acknowledgements

We are grateful to the INRA MIGALE bioinformatics platform (http://migale.jouy.inra.fr) for providing computational resources. All the examples have been run on a Linux platform, with 24 CPUs Intel(R) Xeon(R), CPU E7450, 2.40GHz and 128 Go de RAM.

6 References

[Gauchi (2016)] Gauchi, J.-P. and all. 2016. Metamodeling and global sensitivity analysis for computer models with correlated inputs: a practical approach tested with a 3D light interception computer model. In *Environmental Modelling & Software*, submitted.

[Ishigami (1990)] Ishigami, T., Homma, T., 1990. An importance quantification technique in uncertainty analysis for computer models. In *Proceedings of the First International Symposium on Uncertainty Modeling and Analysis*. IEEE, 398-403.

[Sobol (2003)] Sobol', I.M., 2003. Theorems and examples on high dimensional model representation. In *Reliability Engineering & System Safety* 79, 187-193.

[Sudret (2008)] Bruno Sudret. July 2008. Global sensitivity analysis using polynomial chaos expansions. In *Reliability Engineering and System Safety*, Vol. 93, Issue 7, pages 964-979.