MATLAB Parametric Empirical Kriging (MPErK) User's Guide

Gang Han and Thomas J. Santner (May 2008)
updated by Hyejung Moon (May 2009)

MPErK (a **M**ATLAB program for **P**arametric **E**mpi**r**ical **K**riging) can be used to fit regression plus stationary Gaussian Stochastic process models to data from a computer experiment, for predicting the output at untested sites, for calculating leave-one-out cross validated residuals for the training data, and for doing sensitivity analysis. More specifically, `mperk`

- Gives MLE or REML estimates of the correlation parameters for the Cubic, Gaussian, Power Exponential correlation functions based on user-specified linear regression and stationary Gaussian stochastic process models,
- Estimates the mean and process variance of the Gaussian Stochastic process,
- Optionally computes the Best Linear Unbiased Predictions and associated (process model) prediction errors at a requested set of new inputs
- Optionally performs cross validation estimation on the training data.
- Optionally computes sensitivity indices and creates main effect plots for the Gaussian and Cubic correlation family and a constant mean model (and only this model).

This manual provides instructions and examples for running the program. Section 1 introduces the Gaussian stochastic process models that are fit. Section 2 describes the MATLAB files used by MPErK and the inputs/outputs of the program. Section 3 illustrates the program with examples. Section 4 discusses some technical details of the program implementation.

1 Introduction

MPErK views the output from a computer experiment, $y(\boldsymbol{x})$, as a realization of a Gaussian stochastic process

$$Y(\boldsymbol{x}) = \boldsymbol{\beta}^\top \boldsymbol{f}(\boldsymbol{x}) + Z(\boldsymbol{x}), \tag{1}$$

where $\boldsymbol{x}$ is in the hyper rectangle $\prod_{i=1}^{d}[a_i, b_i]$, $\boldsymbol{f}(\boldsymbol{x})$ is a *known* $p \times 1$ vector of regression functions (usually the constant 1 and possibly other monomials of $\boldsymbol{x}$), $\boldsymbol{\beta}$ is an *unknown* vector of regression parameters, $Z(\boldsymbol{x})$ is a *zero-mean* covariance stationary Gaussian stochastic process with *unknown* variance σ^2 and parametric correlation function $R(\cdot|\boldsymbol{\xi})$ with *known* or *unknown* correlation parameters $\boldsymbol{\xi}$. Thus, for inputs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$,

$$Cov(Z(\boldsymbol{x}_1), Z(\boldsymbol{x}_2)) = \sigma^2 R(\boldsymbol{x}_1 - \boldsymbol{x}_2 \mid \boldsymbol{\xi}),$$

where $R(\cdot \mid \boldsymbol{\xi})$ is allowed to be one of the three correlation functions:

1. **Gaussian correlation**

$$R(\boldsymbol{h} \mid \boldsymbol{\xi}) = \prod_{i=1}^{d} \exp[-\theta_i (h_i)^2], \tag{2}$$

   where $\boldsymbol{\xi} = \boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$ and $\theta_i \geq 0$ for $i \in \{1, \ldots, d\}$.

2. **Product Power Exponential correlation**

$$R(\boldsymbol{h} \mid \boldsymbol{\xi}) = \prod_{i=1}^{d} \exp[-\theta_i \mid h_i|^{p_i}], \tag{3}$$

   where $\boldsymbol{\xi} = (\boldsymbol{\theta}, \boldsymbol{p}) = (\theta_1, \ldots, \theta_d, p_1, \ldots, p_d)$ with $\theta_i \geq 0$ and $0 < p_i \leq 2$.

3. **Cubic correlation**

$$R(\boldsymbol{h} \mid \boldsymbol{\xi}) = \prod_{i=1}^{d} R(h_i \mid \theta_i), \tag{4}$$

   where

$$R(h_i \mid \theta_i) = \begin{cases} 1 - 6(h_i/\theta_i)^2 + 6 \mid h_i|^3/\theta_i^3, & \text{if } |h_i| \leq \theta_i/2, \\ 2(1 - |h_i|/\theta_i)^3, & \text{if } \theta_i/2 \leq |h_i| \leq \theta_i, \\ 0, & \text{if } \theta_i \leq |h_i|, \end{cases} \tag{5}$$

   so that $\boldsymbol{\xi} = \boldsymbol{\theta} = (\theta_1, \ldots, \theta_d)$ and $\theta_i > 0$ and $R(h_i \mid 0) \equiv 0$.

The correlation parameters $\boldsymbol{\xi}$ can be estimated either by maximum likelihood estimation (MLE) or restricted maximum likelihood (REML) estimation. Optimizations of the likelihood or restricted likelihood to find the required $\boldsymbol{\xi}$ values are carried out using the MATLAB function `fmincon` based using multiple starting points. Some details of the implementation of `fmincon` are given in Section 4.

MPErK can predict the output $y(\cdot)$ at a new input site $\boldsymbol{x}_0$ given the training data $\boldsymbol{y}^n = (y(\boldsymbol{x}_1), \ldots, y(\boldsymbol{x}_n))$ as follows. First, assuming that $\boldsymbol{\xi}$ is known, we have, conditional on $(\boldsymbol{\beta}, \sigma^2)$,

$$\begin{pmatrix} Y(\boldsymbol{x}_0) \\ \boldsymbol{Y}^n \end{pmatrix} \sim N_{n+1} \left(\begin{pmatrix} \boldsymbol{f}_0^\top \\ \boldsymbol{F} \end{pmatrix} \boldsymbol{\beta}, \sigma^2 \begin{pmatrix} 1 & \boldsymbol{r}_0^\top \\ \boldsymbol{r}_0 & \boldsymbol{R} \end{pmatrix} \right), \tag{6}$$

where

- $Y(\boldsymbol{x}_0)$ and $\boldsymbol{Y}^n$ are the process analogs of $y(\boldsymbol{x}_0)$ and $\boldsymbol{y}^n$, respectively,
- $\boldsymbol{f}_0 = \boldsymbol{f}(\boldsymbol{x}_0)$ is the $p \times 1$ vector of regression functions of $Y(\boldsymbol{x}_0)$,
- $\boldsymbol{F} = (f_j(\boldsymbol{x}_i))$ is the $n \times p$ matrix of regression functions for the training data,
- $\boldsymbol{r}_0 = (R(\boldsymbol{x}_0 - \boldsymbol{x}_1 | \boldsymbol{\xi}), \ldots, R(\boldsymbol{x}_0 - \boldsymbol{x}_n \mid \boldsymbol{\xi}))^\top$ is the $n \times 1$ vector of correlation of $\boldsymbol{Y}^n$ with $Y(\boldsymbol{x}_0)$,
- $\boldsymbol{R} = (R(\boldsymbol{x}_i - \boldsymbol{x}_j \mid \boldsymbol{\xi}))$ is the $n \times n$ matrix of correlation among the $\boldsymbol{Y}^n$.

Assume that $(\boldsymbol{\beta}, \sigma^2)$ have the non-informative prior $\propto 1/\sigma^2$ and the parameters defining the correlation function are known. Then the conditional mean

$$\widehat{\boldsymbol{y}}(\boldsymbol{x}_0) = E\{Y(\boldsymbol{x}_0) \mid \boldsymbol{Y}^n\} = \left[\boldsymbol{f}_0^\top (\boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{F})^{-1} \boldsymbol{F}^\top \boldsymbol{R}^{-1} + \boldsymbol{r}_0^\top \boldsymbol{R}^{-1} (\boldsymbol{I}_n - \boldsymbol{F}(\boldsymbol{F}^\top \boldsymbol{R}^{-1} \boldsymbol{F})^{-1} \boldsymbol{F}^{-1} \boldsymbol{R}^{-1})\right] \boldsymbol{y}^n. \quad (7)$$

is the best linear unbiased predictor of $\boldsymbol{y}(\boldsymbol{x}_0)$ for this model.

MPErK assumes that the correlation parameters, $\boldsymbol{\xi}$, are unknown, and calculates an empirical version of equation (7) by plugging the estimated correlation parameters into (7) to form an empirical BLUP of $y(\boldsymbol{x}_0)$.

2 Inputs and Outputs

2.1 Introduction

The program `mperk.m` uses the inputs described in Section 2.2 and returns, as output, a MATLAB structure as described in Section 2.3. This section lists the various MATLAB (helper) programs that `mperk.m` calls.

- Programs used to estimate the correlation parameters
  1. `cormatexp.m` calculates the Gaussian or Power Exponential correlation function.
  2. `cubiccorrfn.m` calculates the cubic correlation function.
  3. `estgaussiancor.m` estimates the θ_i parameters for the Gaussian correlation (2).
  4. `estexponentcor.m` estimates the θ_i and p_i parameters for the product power exponential correlation function (3).
  5. `estcubiccor.m` estimates the θ_i parameters for the cubic correlation function (5).
  6. `calicatMLE.m` calls `llik1d.m` to calculate the likelihood.
  7. `calicatREML.m` calls `llik1d.m` to calculate the restricted likelihood.
  8. `llik1d.m` calculates the likelihood/the restricted likelihood.
- Programs used for prediction and cross validation
  1. `cormatexp.m` returns the product power exponential correlation matrix for the training data.
  2. `cubiccorrfn.m` computes the value of the cubic correlation function.
  3. `correxp.m` returns the product power exponential correlation matrix between the training data and the prediction inputs.
- Programs used to computate sensitivity coefficients
  1. `sensitivity.m` returns the sensitivity indices and main effect plots.
  2. `sgint.m` calculates an integration of the Gaussian correlation function.

3. `dbint.m` calculates a double integration of the Gaussian correlation function.
4. `mxint.m` calculates an integration of the product of two the Gaussian correlation functions.
5. `Cfunc.m` calculates the intermediate matrix result $\boldsymbol{C}$ by calling the functions `sgint.m` and `mxint.m` (see sensitivity analysis documentation for details).

2.2 MPErK Inputs

This subsection lists the possible values and default settings of all the inputs. Inputs are specified in *pairs*: the first item of each pair is a character string and the second is its value. See the job files in Section 3.1 for specific examples. *Note* `mperk.m` requires only the inputs `'X'` and `'Y'`; all other inputs are optional.

Required arguments.

- `'X' = x`, an $n \times p$ matrix of inputs for training input values (required) where $n =$ number of training data cases and $p =$ number of the computer code input for each experiment
- `'Y' = y`, an $n \times 1$ vector of real-valued outputs of computer code at the training data sites (required)

Optional Arguments: if not specified, the ***Default*** values are used.

- `'CorrelationEstimationMethod'` $\in \{$`'MLE','REML'`$\}$ where
  - `'MLE'` indicates the Maximum Likelihood (ML) estimation of $\boldsymbol{\xi}$ is to be determined
  - `'REML'` indicates the REstricted Maximum Likelihood (REML) estimation of $\boldsymbol{\xi}$ is to be determined
  - ***Default***: `'CorrelationEstimationMethod' = 'REML'`.
- `'CorrelationFamily'` $\in \{$`'PowerExponential', 'Gaussian', 'Cubic'`$\}$ where
  - `'PowerExponential'` is the product power exponential correlation function,
  - `'Gaussian'` is the Gaussian correlation function (the product power exponential family with powers equal to 2),
  - `'Cubic'` is the product cubic correlation function,
  - ***Default***: `'CorrelationFamily' = 'PowerExponential'`.
- `'XPred' = xpred`, an $n_{pred} \times p$ matrix of n_{pred} new inputs at which predictions are to be computed
  - n_{pred} is the number of the desired predictions,
  - ***Default***: no predictions are requested.
- `'RegressionModel'` $= \boldsymbol{F}$, an $n \times q$ matrix of regression functions describing the mean of the fitted GaSP, i.e., $E(\boldsymbol{Y}^n) = \boldsymbol{F} \times \boldsymbol{\beta}$, where
  - q is the number of regression parameters,

– ***Default***: an $n \times 1$ vector with elements 1.

- `'PredRegressionModel'` $= \boldsymbol{F}_{pred}$, an $n_{pred} \times q$ matrix of regression coefficients for prediction inputs `'Xpred'`.

  – ***Default***: an $n_{pred} \times 1$ vector with elements 1.

- `'CrossValidation'`: $\in$ {`'Yes'`, `'No'`}

  – `'Yes'` indicates that the program does cross validation of the training data and saves the results in the matrix `output.cv`,
  – `'No'` indicates that the program does NOT perform cross validation,
  – ***Default***: `'CrossValidation' = 'No'`.

- `'SensitivityAnalysis'`: $\in$ {`'Yes'`, `'No'`}

  – `'Yes'` indicates that the program produces sensitivity coefficients in `output.sens` and creates main effect plots (mainplot.jpg),
  Warning the sensitivity analysis will only be performed when Correlation family is 'Gaussian' or 'Cubic' and the mean model is the constant mean (the default mean model)
  – `'No'` indicates that the program does NOT compute sensitivity coefficients,
  – ***Default***: `'SensitivityAnalysis' = 'No'`.

- `'InitNumber' = I`, a positive integer number in $\{1, 2, \ldots, 200\}$ used to specify the number of the *starting points* in the search for the maximizer of the likelihood/restricted likelihood. The program generates $200 \times p$ starting points by creating a Latin Hypercube design with $200 \times p$ points, where p is the number (or dimension) of the correlation parameters. The program then computes the likelihood for each of the $200 \times p$ choices and then selects the 'InitNumber'$\times p$ of the correlation parameters having the *largest* likelihood values as starting values at which to run the optimization routine `fmincon`. After running `fmincon`, starting from each of these 'InitNumber'$\times p$ values, that fmincon solution which maximizes of the likelihood is taken to be the estimated correlation parameter used by the remainder of the program.

  – ***Default***: `I`$= 5$.

- `'UseCorModel' = pout`, a structure from a previous run of `mperk.m` containing the estimated correlation parameters to be used in this run. The program can directly predict the outputs and conduct cross validation

  – ***Default***: there is no input for `'UseCorModel'` and correlation parameters are computed during the current run.

2.3 MPErK Outputs

The output of MPErk is a MATLAB structure with five objects. The command of the program is of the form

```
output = mperk('X',x,'Y',...),
```

where `output` is a structure having five objects

1. `output.job`,
2. `output.corparms`,
3. `output.est`,
4. `output.preds` (optional),
5. `output.cv` (optional),
6. `output.sens` (optional)

The object `output.job` is a structure that describes the data, the model, and the estimation method. The elements of `output.job` are

1. `output.job.CorrelationFamily` returns the type of correlation function (`PowerExponential`/`Gaussian`/`Cubic`).
2. `output.job.CorrelationEstimationMethod` returns the estimation method, which is `'MLE'/'REML'`.
3. `output.job.X` is an $n \times p$ training data inputs.
4. `output.job.Y` is an $n \times 1$ training data outputs.
5. `output.job.XPred` is an $n_{pred} \times p$ prediction inputs.
6. `output.job.RegressionModel` $= \boldsymbol{F}$.
7. `output.job.PredRegressionModel` $= \boldsymbol{F}_{pred}$.

The object `output.corparms` is a structure that contains the estimated correlation parameters.

1. `output.corparms.theta` $= \boldsymbol{\theta}$ for the Gaussian/ Power Exponential/ Cubic correlation function.
2. `output.corparms.power` is $(p_1, \ldots, p_d)$ for the Power Exponential correlation; `output.corparms.power` is the vector $(2, 2, \ldots, 2)$ for the Gaussian correlation; `output.corparms.power` $= Inf$ for the Cubic correlation.

The object `output.est` is a structure that contains the following estimated quantities.

1. `output.est.loglik` is the maximized log likelihood/restricted likelihood value. For the log likelihood, `output.est.loglik` returns

$$-\frac{1}{2}\left(\log\left(\widehat{\sigma^2}(\boldsymbol{\xi})\right) + \log(\det(R(\boldsymbol{\xi}))) + n + n \times \log(2\pi)\right),$$

   where log denotes logarithm base e; this is the maximized log likelihood. When `CorrelationEstimationM` $=$ `REML`, `output.est.loglik` returns

$$-\frac{1}{2}\left((n-p)\log\left(\frac{n-1}{n}\widehat{\sigma^2}(\boldsymbol{\xi})\right) + \log(\det(R(\boldsymbol{\xi}))) + \log(\det(\boldsymbol{F}^\top R(\boldsymbol{\xi})^{-1}\boldsymbol{F}))\right),$$

   which is the maximized restricted log likelihood up to a constant.
2. `output.est.beta` is the $q \times 1$ vector of the MLE or REML of the regression parameters $\boldsymbol{\beta}$.
3. `output.est.invR` is an $n \times n$ estimated inverse correlation matrix $inv(\widehat{\boldsymbol{R}})$ corresponding to the training data inputs and the estimated correlation parameters.

4. `output.est.sigma2` is the MLE or REML estimate of $\boldsymbol{\sigma}$. When `CorrelationEstimationMethod` $=$ `MLE`, `output.est.sigma2` $= \frac{1}{n}(\boldsymbol{y}^n - \boldsymbol{F}\widehat{\boldsymbol{\beta}})^\top \boldsymbol{R}^{-1}(\boldsymbol{y}^n - \boldsymbol{F}\widehat{\boldsymbol{\beta}})$. When `CorrelationEstimationMethod` $=$ `REML`, `output.est.sigma2` $= \frac{1}{n-p}(\boldsymbol{y}^n - \boldsymbol{F}\widehat{\boldsymbol{\beta}})^\top \boldsymbol{R}^{-1}(\boldsymbol{y}^n - \boldsymbol{F}\widehat{\boldsymbol{\beta}})$.

The object `output.preds` is a MATLAB structure that contains the predictions and their estimated standard deviations.

Note `output.preds` is created only when `'XPred'` is specified at the input stage.

1. `output.preds.ypreds:` an $n_{pred} \times 1$ vector containing the predictions.
2. `output.preds.se:` an $n_{pred} \times 1$ vector containing the standard errors of the predictions.

The object `output.cv` is a structure that contains the leave-one-out cross validation predictions, standard errors, and residuals.

Note: `output.cv` is created only when `'CrossValidation' = 'Yes'` at the input stage.

1. `output.cv.ypreds:` an $n \times 1$ vector having the leave one out predictions.
2. `output.cv.se:` an $n \times 1$ vector having the standard errors of the predictions.
3. `output.cv.resids:` an $n \times 1$ vector having the cross validation residuals.

The object `output.sens` is a structure that gives the estimated sensitivity coefficients.

Note: `output.sens` is created only when `'SensitivityAnalysis' = 'Yes'` is specified at the input stage. Further more, this command produces estimated sensitivity indices and main effect plots (mainplot.jpg) only for the Gaussian or Cubic correlation family and the constant mean model are specified. In this case,

1. `output.sens.sme:` a $1 \times p$ vector having estimated main effect sensitivity indices.
2. `output.sens.ste:` a $1 \times p$ vector having estimated total effect sensitivity indices.
3. `output.sens.totalvar:` an estimated total variance.
4. `output.sens.ngrid:` the number of grids of input region for evaluating the main effect values.
5. `output.sens.maineffect:` an ngrid $\times p$ matrix having the main effect values evaluated at the grids of inputs which are used to create main effect plots.

For other correlation families or mean models, setting `'SensitivityAnalysis' = 'Yes'` results in `output.sens` being set equal to the warning message
`'MPERK only does sensitivity analysis for Gaussian or Cubic correlation family and constant mean.'`

3 Examples

This sections gives two examples demonstrating the use of MPErK.

3.1 Example 1

The training data for Example 1 is generated from the Branin function

$$y(x_1, x_2) = (x_2 - \frac{5.1}{4\pi^2}\, x_1^2 + \frac{5}{\pi}\, x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\, \cos(x_1) + 10,$$

for $(x_1, x_2) \in [-5, 10] \times [0, 15]$ in $\mathcal{R}^2$.
Goal' Predict $y(\cdot)$ for the Branin function at `xpred` based on the Gaussian process model with mean

$$E[Y(x_1, x_2)] = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_1 x_2$$

and cubic correlation function whose parameters are to be estimated by REML. Predictions are to be made at five new inputs.

The training data inputs are 21 points are obtained from a maximin Latin hypercube design in $[-5, 10] \times [0, 15]$.

```
x =[7.5000    6.0714
    1.0714    3.9286
    9.6429    8.2143
    4.6429    4.6429
    2.5000   14.6429
   -3.2143    2.5000
    3.2143    0.3571
   -4.6429    6.7857
   -3.9286   12.5000
    6.0714    1.7857
    8.2143   11.0714
    6.7857   13.9286
   -0.3571    1.0714
   -1.7857    5.3571
    0.3571   10.3571
    8.9286    3.2143
   -2.5000    9.6429
    5.3571    8.9286
    3.9286   11.7857
   -1.0714   13.2143
    1.7857    7.5000];
```

The training data outputs `Y`

```
y = [ 35.80951
      14.86287
      31.41880
      19.87899
      141.88566
      99.43335
      3.88973
      97.47380
      6.27060
      19.85914
      95.50587
      181.74214
      49.39445
      23.13762
      43.09524
      2.82392
      3.61474
      75.79100
      104.11175
      43.33586
      23.39797];
```

Predictions of $y(\cdot)$ are desired at (x_1, x_2) points in `xpred`

```
xpred = [  -4.5000    0.5000
           -4.5000   14.5001
            2.5000    7.5000
            9.5000    0.5000
            9.5000   14.5001];
```

The regression function for the training data is specified by.

```
F = [ones(21,1) x(:,1) x(:,2) diag(x(:,1)*(x(:,2))')];
```

The regression function for the prediction inputs is specified by.

```
Fpred = [ones(5,1) xpred(:,1) xpred(:,2) ...
        diag(xpred(:,1)*(xpred(:,2))')];
```

The MATLAB command to execute `mperk` is

```
branin_output = mperk('X',x, ...
                'Y',y,...
                'CorrelationEstimationMethod','REML',...
                'CorrelationFamily','Cubic',...
                'Xpred',xpred,...
                'RegressionModel',F,...
                'PredRegressionModel', Fpred);
```

The output is the structure `branin_output` which has components

• The original job specification

```
branin_output.job =
              CorrelationFamily: 'Cubic'
    CorrelationEstimationMethod: 'REML'
                              X: [21x2 double]
                              Y: [21x1 double]
                          XPred: [5x2 double]
                RegressionModel: [21x4 double]
            PredRegressionModel: [5x4 double]
```

• Estimated correlation parameters

```
branin_output.corparms =
         theta: [2x1 double]
         power: Inf
branin_output.corparms.theta =
   18.5006
   43.8566
```

• Miscellaneous estimated quantities

```
branin_output.est =
    loglik: -56.2986
      beta: [4x1 double]
     inv_R: [21x21 double]
    sigma2: 1.1362e+004
branin_output.est.beta =
  227.0857
  -24.3526
   -5.0816
    2.0273
```

• Predictions and their standard errors

```
[branin_output.preds.ypreds branin_output.preds.se] =
  214.6038   14.3067
    3.3244   10.8935
   23.8428    3.7069
  -19.0365   14.1905
  153.1061   15.7321
```

A second use of `mperk` requests the MLEs of the correlation parameters and the sensitivity analysis.

```
branin_outsen = mperk('X',x,...
                      'Y',y,...
                      'CorrelationEstimationMethod','MLE',...
                      'CorrelationFamily','Gaussian',...
                      'SensitivityAnalysis','Yes');
```

The output structure `branin_outsen` has components

• The original job specification

```
branin_outsen.job =
              CorrelationFamily: 'Gaussian'
    CorrelationEstimationMethod: 'MLE'
                              X: [21x2 double]
                              Y: [21x1 double]
                RegressionModel: [21x1 double]
            PredRegressionModel: 1
                           XPred: 'None'
```

• Estimated correlation parameters

```
branin_outsen.corparms =
         theta: [2x1 double]
         power: [2x1 double]
branin_outsen.corparms.theta =
    0.0345
    0.0022
```

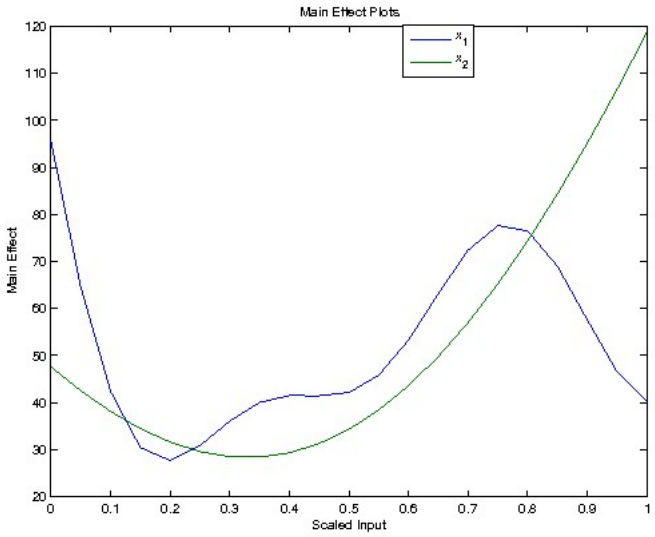
• Miscellaneous estimated quantities

```
branin_outsen.est =
    loglik: -94.8882
      beta: 196.4499
     inv_R: [21x21 double]
    sigma2: 2.2472e+04
```

- Sensitivity indices

```
branin_outsen.sens =
           sme: [0.1259 0.2966]
           ste: [0.7034 0.8741]
      totalvar: 2.1400e+03
         ngrid: 21
    maineffect: [21x2 double]
```

- Main effect plots (mainplot.jpg)



3.2 Example 2

Example 2 uses the piston slap data in Runze Li and Agus Sudjianto (2005). The *Goal* is to estimate the correlation for the piston slap data based on the Gaussian process model with constant mean power exponential correlation function whose parameters are estimated by MLE. Also construct cross validation estimates for all training data inputs.

The training data inputs are the 12 points

```
X=[71.0000   16.8000   21.0000    2.0000    1.0000    0.9800
   15.0000   15.6000   21.8000    1.0000    2.0000    1.3000
   29.0000   14.4000   25.0000    2.0000    1.0000    1.1400
   85.0000   14.4000   21.8000    2.0000    3.0000    0.6600
   29.0000   12.0000   21.0000    3.0000    2.0000    0.8200
   57.0000   12.0000   23.4000    1.0000    3.0000    0.9800
   85.0000   13.2000   24.2000    3.0000    2.0000    1.3000
   71.0000   18.0000   25.0000    1.0000    2.0000    0.8200
   43.0000   18.0000   22.6000    3.0000    3.0000    1.1400
   15.0000   16.8000   24.2000    2.0000    3.0000    0.5000
   43.0000   13.2000   22.6000    1.0000    1.0000    0.5000
   57.0000   15.6000   23.4000    3.0000    1.0000    0.6600];
```

The training data outputs `Y`:

```
Y=[56.7500
   57.6500
   53.9700
   58.7700
   56.3400
   56.8500
   56.6800
   58.4500
   55.5000
   52.7700
   57.3600
   59.6400];
```

The MATLAB command to find the maximum likelihood estimates of the correlation parameters for the power exponential family (3) with cross validated predictions is:

```
pistslap_out = mperk('X',X,...
               'Y',Y,...
               'CorrelationEstimationMethod','MLE',...
               'CorrelationFamily','PowerExponential',...
               'CrossValidation','Yes');
```

The output structure `pistslap_out` has components

```
pistslap_out =
         job: [1x1 struct]
    corparms: [1x1 struct]
         est: [1x1 struct]
          cv: [1x1 struct]
```

which give

• The original job specifications:

```
pistslap_out.job =
              CorrelationFamily: 'PowerExponential'
    CorrelationEstimationMethod: 'MLE'
                              X: [12x6 double]
                              Y: [12x1 double]
                RegressionModel: [12x1 double]
            PredRegressionModel: 1
                           Xpred: 'None'
```

- Estimated correlation parameters:

```
[pistslap_out.corparms.theta, pistslap_out.corparms.power] =
    0.0008    2.0000
    0.0000    2.0000
    0.0397    2.0000
    0.0000    2.0000
    0.0000    2.0000
    4.4468    2.0000
```

- Miscellaneous estimated quantities:

```
pistslap_out.est =
    loglik: -21.9834
      beta: 56.2509
     inv_R: [12x12 double]
    sigma2: 4.2716
```

- Leave-one-out cross validation predictions, their standard errors, and their residuals:

```
pistslap_out.cv =
    ypreds: [12x1 double]
        se: [12x1 double]
    resids: [12x1 double]

[pistslap_out.cv.ypreds, pistslap_out.cv.se,...
                         pistslap_out.cv.resids] =
   57.5857    1.2307   -0.8357
   55.3592    1.5600    2.2908
   55.9102    1.3402   -1.9402
   58.1166    1.5030    0.6534
   55.9018    1.5935    0.4382
   56.8350    0.6705    0.0150
   55.5453    1.9629    1.1347
   58.8664    1.0728   -0.4164
   55.6000    0.8836   -0.1000
   55.4814    1.6218   -2.7114
   57.8271    1.0015   -0.4671
   58.6507    0.7175    0.9893
```

4 Some Program Notes

For numerical stability, MPErK calculates correlation parameters to minimize the likelihood with correlations expressed in terms of standardized input values. For example, in the case of the Gaussian correlation function (2), MPErK calculates $\widehat{\theta}_1^\star, \ldots, \widehat{\theta}_d^\star$ to maximize the log-likelihood with $\boldsymbol{R} = (R_{ij})$ and

$$R_{ij} = \exp\left\{ -\sum_{\ell=1}^{d} \theta_\ell^\star \left(\frac{x_{i\ell} - x_{j\ell}}{r_\ell} \right)^2 \right\},$$

where $r_\ell = \max_q\{x_{q\ell}\} - \min_q\{x_{q\ell}\}$. The MLE (or REML) of $\theta_1, \ldots, \theta_d$ in (2) are obtained by `mperk` from the estimated $\{\theta_\ell^\star\}_\ell$ using $\widehat{\theta}_l = \widehat{\theta}_l^\star / r_l^2$, for $l = 1, \ldots, d$.

MPErK uses `fmincon` function to estimate the unknown correlation parameters based the largest value of the (restricted) likelihood that `fmincon` obtains starting with a user-specified number of starting points. The options for `fmincon` are set by the command.

```
options = optimset('Display','off',...
                   'LargeScale','on',...
                   'TolFun',.0001, ...
                   'Algorithm','active-set');
```

The default settings are used for all other `fmincon` options. Detailed explanations of the settings and all possible values of each option can be found in the MATLAB help file for functions `optimset` and `fmincon`.

We apply `fmincon` with lower and upper bounds all $\theta_i^\star$, $i = 1, \ldots, d$. For the product power exponential and Gaussian correlation functions, the lower bound for $\theta_i^\star$ is set so that the correlation between the two (hypothetical) points obtained by projecting the training input data into each dimension is less than or equal to 0.99 and the correlation between the two (hypothetical) points obtained by projecting the training input is greater than or equal to 0.01. More specifically, for the Gaussian correlation function let M_i denote the largest Euclidean (L_2) distance between two training data inputs in ith dimension, and m_i is the smallest non-zero L_2 distance between two training data inputs in ith dimension, $i = 1, \ldots, d$. Then we take the lower bound for $\theta_i^\star$ to be

$$-log(0.99^{1/d})/M_i$$

and the upper bound for θ_i to be

$$-log(0.01^{1/d})/m_i,$$

where d is the number of inputs. For the cubic correlation function, it is hard to set the bounds of $\theta_i^\star$ to control the overall correlations $Cov(Z(\boldsymbol{x}_i), Z(\boldsymbol{x}_i))$. We take the lower bound for $\theta_i^\star$ to be 0.00001 and the upper bound to be $100/d$.

References

T. J. Santner, B. J. Williams and W. I. Notz (2003). *The Design and Analysis of Computer Experiments*, Springer Verlag, New York.

Runze Li and Agus Sudjianto (2005). *Analysis of Computer Experiments Using Penalized Likelihood in Gaussian Kriging Models*, Technometrics, Volume **47**, number 2, pages 111-120.