1. Common Instructions

- Set the working directory to *Master* folder before running any R code.
- Parts of the code are set up to run in parallel, so after creating the clusters in *Main.R* and *Main_design_selection.R*, ensure each clusters knows the pathway to your R library through either *clusterCall()* or *clusterEvalQ()*, as mentioned in the comments of the code.
- The three R functions available in the main folder can be used to reproduce figures and tables shown in Example 1, Example 2 and in the supplementary material.
- The corresponding individual figures and tables are available in the following folder: *Results/Figures_and_Tables.*

2. Example 1:

2.1 Obtaining optimal designs

- The R code to obtain designs using three loss functions are available in the following folder: *R_codes/Example1.*
- You are required to install the R package *MaternEx1_1.0.tar.gz* to run this example.
- The designs for this example can be obtained by running the main R script called *Main_design_selection.R* inside the folder.
- Inside the file *Main_design_selection.R*, there are different options for users to run this code. They are:
  - *d_no* = number of design points in the design
  - *utility* = negative value of the loss function
  - *Dependence* = spatial dependency structure of the two responses

We have run this code with two different values for *d_no* (5 and 10) and three different values for *Dependence* (0.2, 0.5 and 0.8) based on the three loss functions. The resulting designs are saved in the following folder: *Results/Example1/Selected_designs*.

2.2 Design evaluation

- After designs have been determined for each loss function, they can be evaluated based on each design objective.
- For this purpose, the *Example1_design_evaluation.R* function available in the folder: *R_codes/Example1* can be used. Here, two parameters, *d_no* and *Dependence* should be set before running this code.
- Design evaluation results are saved in the folder *Results/Example1/Simulation_results/ Design_evaluation*.

2.3 Compare two approximations

- To compare the two approximations used in the paper for evaluating the prediction loss values, the function *Main.R* can be used. This file is location in folder: *R_codes/Example1.*

2.4 Simulation results

- To obtain the plots and tables shown for Example 1 we used the results obtained in Sections 1.2 and 1.3 above.
- The results obtained in Sections 1.2 and 1.3 are available in *Results/Example1/ Simulation_results*.
- *Example1_Results.R* in the *Master* folder can be used to plot the results for Example 1.

3. Example 2:

3.1 Obtaining optimal designs

- The R code to obtain designs using three loss functions are available in the following folder: *R_codes /Example2.*
- You are required to install the R package *AirQualityRcpp_1.0.tar.gz* to run this example.
- The designs for this example can be obtained by running the main script called *Main_design_selection.R* inside the folder.
- Inside the *Main_design_selection.R*, there are two options for users to run this code. They are,
    - *d_no* = number of design points in the design
    - *utility* = negative value of the loss function
- We have run this code with four different values for *d_no* (5, 7, 10 and 15) based on the three loss functions. The resulting designs are saved in the following folder: *Results/Example1/Selected_designs*.

3.2 Design evaluation

- After designs have been determined for each loss function, they can be evaluated based on each design objective.
- For this purpose, the function called *Example2_design_evaluation.R* available in the folder: *R_codes/Example2* can be used. Here, two parameters, *d_no* and *Dependence* should be set before running this code.
- Design evaluation results are saved in the folder: *Results/Example2/Simulation _results/ Design_evaluation*.

3.3 Compare two approximations

- To compare the two approximations used in the paper for evaluating the prediction loss values, the *Main.R* function in the following folder can be used: *R_codes/Example2.*

3.4 Simulation results

- To obtain the plots and tables shown for Example 2 we used the results obtained in Sections 2.2 and 2.3 above.
- The results obtained in Sections 2.2 and 2.3 are saved in *Results/Example2/ Simulation_results*.
- We use the function *Example2_Results.R* available in the *Master* folder to obtain the results related to Example 2.
- You are required to install the R package *mapping* to obtain Figure 1. This can be done either using the source file *mapping_0.1.tar.gz* available in the main folder or using the web link mentioned in the comments of the code.

4. Design comparison (Supplementary material):

- To obtain designs from cluster 3 in Example 2, we used the same set of code discussed in Section 2.1 above.
- Here, the function *Main_design_selection_C3.R* is used for design selection.

- To compare the efficiencies of the two approximations to the prediction loss function, the same set of R code discussed in Sections 1.3 and 2.3 can used. Here, the *Main_efficiency.R* functions available in the respective folders can be used.
- To compare the designs obtained from the estimation, prediction and dual purpose loss functions with designs suggested in the literature, posterior predictive variances and the posterior variances of the parameters were obtained. For this purpose, the function called *Main_common_design.R* available in the folder: *R_codes/Example1* can be used.
- The figures and tables available in the Supplementary material can be obtained using the function *Supplementary_Results.R* available in the *Master* folder.

5. User defined functions

5.1. Example 1 – functions in *functions_Ex1.R* script

Function	Inputs	Outputs
combine_ute()	d – design X – values for independent variables Y – data theta – parameter vector distM – distance matrix	mu_post – posterior mean Sigma_post – posterior covariance matrix det.out – dual purpose utility kld.out – estimation utility pred.out – prediction utility
combine_ute_both()	d – design X – values for independent variables Y – data theta – parameter vector distM – distance matrix	mu_post – posterior mean Sigma_post – posterior covariance matrix det.out – dual purpose utility kld.out – estimation utility pred.out – prediction utility ($\tilde{\lambda}$) pred.out2 – prediction utility ($\hat{\lambda}$) time1 – time taken to obtain $\tilde{\lambda}$ time2 – time taken to obtain $\hat{\lambda}$
exp.crit()	d – design B – number of prior predictive datasets	crit – B values of a given utility function.
exp.crit.all()	d – design B – number of prior predictive datasets	crit – a vector containing B values of three utility functions (i.e. estimation, dual-purpose and prediction)
exp.crit.both.P()	d – design B – number of prior predictive datasets	crit.dual – B values of the dual-purpose utility crit.kld – B values of the estimation utility crit.pred.App – B values of the prediction utility ($\tilde{\lambda}$) crit.pred – B values of the prediction utility ($\hat{\lambda}$)
exp.crit.var()	d – design B – number of prior predictive datasets	post.var – posterior variances of the parameters Y1.post – posterior prediction variances for response 1

		Y2.post – posterior prediction variances for response 2
Laplace_approx()	X – values for independent variables Y – data theta – parameter vector distM – distance matrix	lp.approx – a list containing the minimum negative log posterior value and its corresponding posterior mode and Hessian matrix
log_post()	X – values for independent variables Y – data theta – parameter vector distM – distance matrix	Neg_log_post – the negative log posterior value
Post_pred()	d – design X – values for independent variables Y – data theta – parameter vector distM – distance matrix	mu_post – posterior mean Sigma_post – posterior covariance matrix Y1.mat – posterior predictions for response 1 Y2.mat – posterior predictions for response 2
Pred_ute()	X_unsamp – values of independent variables at prediction locations distM – distance matrix post_samp1 – sample from a posterior distribution post_samp2 – sample from a posterior distribution r01 – partial sill r02 – range parameter	pred_ute – prediction utility obtained using the first approximation ($\hat{\lambda}$)
Pred_ute.App()	X_unsamp – values of independent variables at prediction locations distM – distance matrix post_samp1 – sample from a posterior distribution r01 – partial sill r02 – range parameter	pred_ute – prediction utility obtained using the second approximation ($\tilde{\lambda}$)
trace_mat()	M – square matrix	tr – trace of the matrix

5.2. Example 2 – functions in *functions_Ex2.R* script

Function	Inputs	Outputs
combine_ute()	d – design X – values for independent variables Y – data theta – parameter vector distM – distance matrix	mu_post – posterior mean Sigma_post – posterior covariance matrix det.out – dual purpose utility kld.out – estimation utility pred.out – prediction utility
combine_ute_all()	d – design	mu_post – posterior mean

	X – values for independent variables Y – data theta – parameter vector distM – distance matrix	Sigma_post – posterior covariance matrix det.out – dual purpose utility kld.out – estimation utility pred.out – prediction utility ($\tilde{\lambda}$) pred.out2 – prediction utility ($\hat{\lambda}$) time1 – time taken to obtain $\tilde{\lambda}$ time2 – time taken to obtain $\hat{\lambda}$
exp.crit()	d – design B – number of prior predictive datasets	crit – expected utility values from the three utility functions.
exp.crit.all()	d – design B – number of prior predictive datasets	crit – a vector containing B values of three utility functions (i.e. estimation, dual-purpose and prediction)
exp.crit.both.P()	d – design B – number of prior predictive datasets	crit – a vector containing B values of dual-purpose utility, estimation utility, prediction utility ($\tilde{\lambda}$), and prediction utility ($\hat{\lambda}$)
Laplace_approx()	X – values for independent variables Y – data theta – parameter vector distM – distance matrix	lp.approx – a list containing the minimum negative log posterior value and its corresponding parameter vector and the Hessian matrix
log_post()	X – values for independent variables Y – data theta – parameter vector distM – distance matrix	Neg_log_post – the negative log posterior value
Pred_ute()	X_unsamp – values of independent variables at prediction locations distM – distance matrix post_samp1 – sample from a posterior distribution post_samp2 – sample from a posterior distribution r01 – partial sill r02 – range parameter	pred_ute – prediction utility obtained using the first approximation ($\hat{\lambda}$)
Pred_ute.App()	X_unsamp – values of independent variables at prediction locations distM – distance matrix post_samp1 – sample from a posterior distribution r01 – partial sill r02 – range parameter	pred_ute – prediction utility obtained using the second approximation ($\tilde{\lambda}$)
trace_mat()	M – square matrix	tr – trace of the matrix

```
> sessionInfo()
R version 4.1.3 (2022-03-10)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 19044)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.1252
[2] LC_CTYPE=English_United States.1252
[3] LC_MONETARY=English_United States.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.1252

attached base packages:
[1] parallel  stats     graphics  grDevices
[5] utils     datasets  methods   base

other attached packages:
 [1] AirQualityRcpp_1.0 doParallel_1.0.17
 [3] iterators_1.0.14   foreach_1.5.2
 [5] acebayes_1.10      lhs_1.1.5
 [7] corpcor_1.6.10     nlme_3.1-155
 [9] matrixcalc_1.0-5   Matrix_1.4-0
[11] fields_14.0        viridis_0.6.2
[13] viridisLite_0.4.0  spam_2.9-0
[15] mvtnorm_1.1-3      MaternEx1_1.0
[17] ggplot2_3.3.6

loaded via a namespace (and not attached):
 [1] Rcpp_1.0.8.3      pillar_1.7.0
 [3] compiler_4.1.3    tools_4.1.3
 [5] dotCall64_1.0-1   compare_0.2-6
 [7] lattice_0.20-45   lifecycle_1.0.1
 [9] tibble_3.1.7      gtable_0.3.0
[11] pkgconfig_2.0.3   rlang_1.0.4
[13] DBI_1.1.2         cli_3.3.0
[15] rstudioapi_0.13   gridExtra_2.3
[17] withr_2.5.0       dplyr_1.0.9
[19] generics_0.1.2    vctrs_0.4.1
[21] maps_3.4.0        grid_4.1.3
[23] tidyselect_1.1.2  glue_1.6.2
[25] R6_2.5.1          randtoolbox_2.0.2
[27] fansi_1.0.3       purrr_0.3.4
[29] magrittr_2.0.3    codetools_0.2-18
[31] scales_1.2.0      ellipsis_0.3.2
[33] assertthat_0.2.1  colorspace_2.0-3
[35] utf8_1.2.2        rngWELL_0.10-7
[37] munsell_0.5.0     crayon_1.5.1
```