Experiments based on pre-calculated kernel matrix

Zhu Huanyan

1. Preparation

```r
library(hetGP)
library(ggplot2)
```

```r
# benchmark functions
# https://en.wikipedia.org/wiki/Test_functions_for_optimization
sphere_func <- function(x) {
    return(sum(x^2))
}

ackley_func <- function(x) {
    return(-20 * exp(-0.2 * sqrt(0.5 * sum(x^2))) -
        exp(0.5 * sum(cos(2 * pi * x))) + exp(1) + 20)
}

rastrigin_func <- function(x, A = 10) {
    n <- length(x)
    return(A * n + sum(x^2) - A * sum(cos(2 * pi * x)))
}

gaussian_pdf_func <- function(x) {
    f1 <- dmvnorm(x, mean = c(0.25, 0.75), sigma = diag(rep(0.01, 2)))
    f2 <- dmvnorm(x, mean = c(0.75, 0.25), sigma = diag(rep(0.01, 2)))

    return(f1 - f2)
}
```

```r
rastrigin_adj_func <- function(x) {
    return(sum(cos(8 * pi * x / (x + 0.2))) + sum(x^2))
}

# observations with constant noise N(0, 0.01)
y <- function(x, f) {
    return(f(x) + rnorm(1, 0, 0.01))
}
```

```r
mse <- function(
    X0, X.unique, mult,
    cov_type = "Gaussian", nu_hat = 1, theta = 0.1, noise = 0.01) {
    r <- nrow(X.unique)
    d <- ncol(X.unique)
    if (d > 1 && length(theta) == 1) theta <- rep(theta, d)

    KX <- nu_hat * cov_gen(
        X1 = X0, X2 = X.unique,
        theta = theta, type = cov_type
    )

    K <- nu_hat * cov_gen(X1 = X.unique, theta = theta, type = cov_type) +
        diag(rep(noise, r) / mult)

    L <- t(chol(K))
    v <- solve(L, t(KX))

    mse <- nu_hat - t(v) %*% v
    return(mse)
}
```

```r
# The exact difference = MSE(X0|X_1, ..., X_{N+1}) - MSE(X0|X_1, ..., X_N)
exact_mse_diff <- function(
    X0, X.new, X.unique, mult,
    cov_type = "Gaussian", nu_hat = 1, theta = 0.1, noise = 0.01) {
    r <- nrow(X.unique)
    d <- ncol(X.unique)

    if (d > 1 && length(theta) == 1) theta <- rep(theta, d)

    K1 <- nu_hat * cov_gen(X1 = X0, X2 = X.new, theta = theta, type = cov_type) # k(x, X.new
    KX <- nu_hat * cov_gen(X1 = X0, X2 = X.unique, theta = theta, type = cov_type) # kn(x)
    Sn <- nu_hat * cov_gen(X1 = X.unique, theta = theta, type = cov_type) +
        diag(rep(noise, r) / mult) # Sn
    Kn <- nu_hat * cov_gen(X1 = X.new, X2 = X.unique, theta = theta, type = cov_type) # kn(X

    L <- t(chol(Sn)) # K = L L^T
    u <- solve(L, t(Kn)) # L^(-1) kn(X.new)

    v0 <- solve(L, t(KX)) # L^(-1) kn(x)

    sigma2 <- nu_hat + noise - t(u) %*% u

    g <- K1 - t(u) %*% v0

    return(-1 / sigma2 * g^2)
}
```

```
d <- 1
N <- 10
cov_type <- "Gaussian"

X_input <- matrix(runif(N, 0, 1), ncol = 1)
X_input <- rbind(X_input, X_input)

f <- sphere_func
Y_input <- apply(X_input, 1, y, f = f)

tmp <- find_reps(X_input, Y_input)

X.unique <- tmp$X0
Y.avg <- tmp$Z0
mult <- tmp$mult
```

2. Verify MSE functions

The following experiment verified the recursive formula of MSE function:

$$exact_mse_diff = MSE(X0|X_1, ..., X_{N+1}) - MSE(X0|X_1, ..., X_N)$$

Let $X.new$ denote X_{N+1}.

```r
X.new <- matrix(runif(1, 0, 1))
print(X.new)
```

```
           [,1]
[1,] 0.08638314
```

```r
X_input_new <- rbind(X_input, X.new)
Y_input_new <- c(Y_input, y(X.new, f = f) + rnorm(1, 0, 0.1))

tmp_new <- find_reps(X_input_new, Y_input_new)

X.unique.new <- tmp_new$X0
mult.new <- tmp_new$mult

# Verify mse function and exact_mse_diff function at test points X0
for (X0 in seq(0, 1, length.out = 9)) {
    X0 <- matrix(X0)
    print(c(
        mse(X0, X.unique.new, mult.new) - mse(X0, X.unique, mult),
        exact_mse_diff(X0, X.new, X.unique, mult)
    ))
}
```

```
[1] -8.067659e-05 -8.067659e-05
[1] -0.003458342 -0.003458342
[1] -0.003414221 -0.003414221
[1] -0.00025492 -0.00025492
[1] -1.049263e-05 -1.049263e-05
[1] -5.887459e-09 -5.887459e-09
[1] -2.392875e-08 -2.392875e-08
[1] -1.393566e-07 -1.393566e-07
[1] -2.616261e-05 -2.616261e-05
```

3. Objective functions

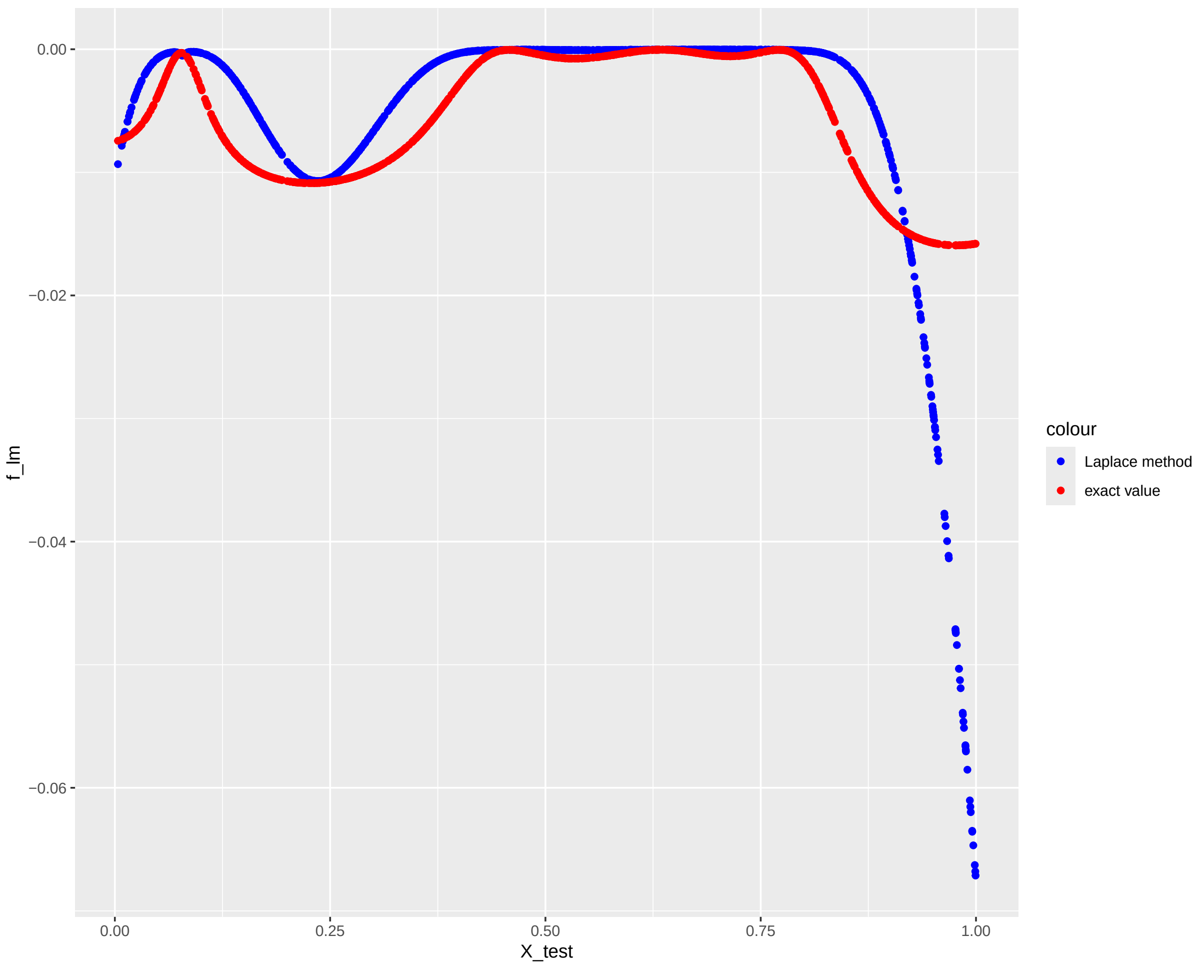
```r
# pdf of input covariate X
h <- function(x, dist) {
    if (dist == "unif") {
        return(1) # uniform(0, 1)
    }
    if (dist == "norm") {
        return(exp(-(x - 0.5)^2 / (2 * 0.1^2))) # norm(0.5, 0.1)
    }
    if (dist == "beta") {
        return((x^0.5) * ((1 - x)^3)) # beta(1.5, 4)
    }
    return(1)
}

# Approximation of objective function based on Laplace method
Laplace_method_mv <- function(
    par, X.unique, mult,
    dist = "unif", cov_type = "Gaussian", nu_hat = 1, theta = 0.1, noise = 0.01) {
    r <- nrow(X.unique)
    d <- ncol(X.unique)
    if (d > 1 && length(theta) == 1) theta <- rep(theta, d)

    X <- matrix(par, nrow = 1) # X
    X.new <- matrix(par, nrow = 1) # X.new

    K1 <- nu_hat * cov_gen(X1 = X, X2 = X.new, theta = theta, type = cov_type) # k(X, X.new)
    KX <- nu_hat * cov_gen(X1 = X, X2 = X.unique, theta = theta, type = cov_type) # kn(X)
    Sn <- nu_hat * cov_gen(X1 = X.unique, theta = theta, type = cov_type) +
        diag(rep(noise, r) / mult) # Sn
    Kn <- nu_hat * cov_gen(X1 = X.new, X2 = X.unique, theta = theta, type = cov_type) # kn(X

    L <- t(chol(Sn)) # Sn = L L^T
    u <- solve(L, t(Kn)) # L^(-1) kn(X.new)

    v0 <- solve(L, t(KX)) # L^(-1) kn(x)

    sigma2 <- nu_hat + noise - t(u) %*% u

    g <- K1 - t(u) %*% v0

    H <- matrix(rep(0, d^2), nrow = d)
    for (i in 1:d) {
        v1 <- solve(L, t((X[i] - X.unique[, i]) * KX)) # L^(-1) * diag(X[i] - X.unique[, i])
        gdi <- -2 / theta[i] * (X[i] - X.new[i]) * K1 + 2 / theta[i] * t(u) %*% v1
        for (j in i:d) {
            if (i == j) {
                v2 <- solve(L, t((X[i] - X.unique[, i])^2 * KX)) # L^(-1) diag(X[i] - X.uniq
                gdii <- -2 / theta[i] * K1 +
                    4 / theta[i]^2 * (X[i] - X.new[i])^2 * K1 +
                    2 / theta[i] * t(u) %*% v0 -
                    4 / theta[i]^2 * t(u) %*% v2

                H[i, j] <- -(gdi / g)^2 + gdii / g
            } else {
                gdj <- -2 / theta[j] * (X[j] - X.new[j]) * K1 +
                    2 / theta[j] * t(u) %*% v1

                v3 <- solve(L, t((X[i] - X.unique[, i]) * (X[j] - X.unique[, j]) * KX)) # L^

                gdij <- 4 / (theta[i] * theta[j]) * (X[i] - X.new[i]) *
                    (X[j] - X.new[j]) * K1 -
                    4 / (theta[i] * theta[j]) * t(u) %*% v3

                H[i, j] <- -gdi * gdj / g^2 + gdij / g
                H[j, i] <- H[i, j]
            }
        }
    }

    I <- -1 / sigma2 *
        pi^(d / 2) *
        prod(apply(X, 2, h, dist = dist)) *
        g^2 / sqrt(as.numeric(determinant(H, logarithm = FALSE)$modulus))
    return(I)
}
```

```
# exact objective function calculated by Wij function in hetGP package
exact_obj_unif <- function(
    par, X.unique, mult,
    dist = "unif", cov_type = "Gaussian", nu_hat = 1, theta = 0.1, noise = 0.01) {
    r <- nrow(X.unique)
    d <- ncol(X.unique)
    if (d > 1 && length(theta) == 1) theta <- rep(theta, d)

    X.new <- matrix(par, nrow = 1) # X.new
```

```r
    Sn <- nu_hat * cov_gen(X1 = X.unique, theta = theta, type = cov_type) +
        diag(rep(noise, r) / mult) # Sn
    Kn <- nu_hat * cov_gen(X1 = X.new, X2 = X.unique, theta = theta, type = cov_type) # kn(X

    L <- t(chol(Sn)) # Sn = L L^T
    u <- solve(L, t(Kn)) # L^(-1) kn(X.new)

    sigma2 <- nu_hat + noise - t(u) %*% u

    w0 <- Wij(mu1 = X.new, mu2 = X.new, theta = theta, type = cov_type)
    W <- Wij(mu1 = X.unique, mu2 = X.unique, theta = theta, type = cov_type)
    wn <- Wij(mu1 = X.new, mu2 = X.unique, theta = theta, type = cov_type)

    v <- solve(L, t(wn)) # L^(-1) wn

    # I1 <- -1 / sigma2 * (w0 + t(u) %*% solve(L) %*% W %*% t(solve(L)) %*% u - 2 * t(u) %*%
    # I2 <- -1 / sigma2 * (w0 + Kn %*% solve(Sn) %*% W %*% solve(Sn) %*% t(Kn) - 2 * Kn %*% s
    # print(c(I1, I2))

    return(-1 / sigma2 *
        (w0 + t(u) %*% solve(L) %*% W %*% t(solve(L)) %*% u - 2 * t(u) %*% v))
}
```

This is the plot of Laplace_method_mv function and exact_obj_unif, in order to visualize and compare.

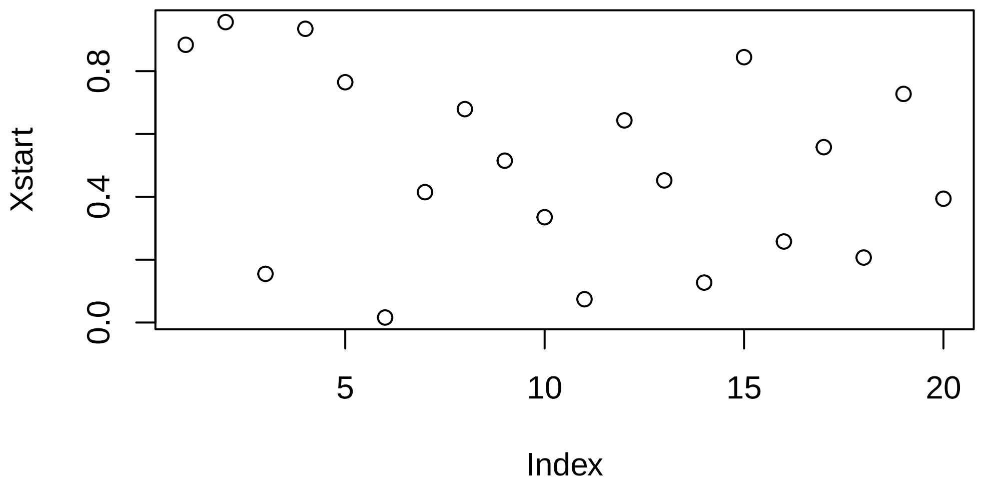
```r
X_test <- runif(1000, 0, 1)

f_lm <- as.numeric(lapply(X_test, Laplace_method_mv, X.unique = X.unique, mult = mult))
f_ex <- as.numeric(lapply(X_test, exact_obj_unif, X.unique = X.unique, mult = mult))

# plot of exact objective function and Laplace method approximation, for visualization
df <- data.frame(X_test, f_lm, f_ex)
ggplot(df) +
    geom_point(aes(x = X_test, y = f_lm, color = "blue")) +
    geom_point(aes(x = X_test, y = f_ex, color = "red")) +
    scale_color_manual(
        values = c("blue", "red"),
        labels = c("Laplace method", "exact value")
    )
```



From the above plot, although there are some gaps between the two functions, they show a similar pattern.

4. Training&Testing

```r
library(mvtnorm)
library(DiceDesign)

# multistart points for optimization
multi_start <- 20
Xstart <- maximinSA_LHS(lhsDesign(multi_start, d, seed = 2024)$design)$design
plot(Xstart)
```



```
library(optimx)

train_Laplace <- function(mod, f, step = 2, dist = "unif") {
    start <- Sys.time()

    d <- ncol(mod$X0)

    for (k in 1:step) {
        out <- multistart(
            Xstart,
            Laplace_method_mv,
            method = "L-BFGS-B",
            lower = rep(0, d), upper = rep(1, d),
            X.unique = mod$X0, mult = mod$mult,
            dist = dist, cov_type = cov_type,
            nu_hat = mod$nu_hat, theta = mod$theta, noise = mod$g
        )

        Xnew <- matrix(
            c(
                out$p1[which.min(out$value)],
                out$p2[which.min(out$value)]
            ),
            nrow = 1
        )
        Ynew <- y(Xnew, f = f)

        mod <- update(mod, Xnew = Xnew, Znew = Ynew, ginit = mod$g * 1.01)
```

```r
        # print(sprintf("iteration: %d, Xnew: (%f, %f), theta: (%f, %f), nu_hat: %f, noise: %
    }
    print(sprintf("Laplace train time: %f", Sys.time() - start))

    return(mod)
}
```

```r
train_exact_obj <- function(mod, f, step = 2, dist = "unif") {
    start <- Sys.time()

    d <- ncol(mod$X0)

    for (k in 1:step) {
        out <- multistart(
            Xstart,
            exact_obj_unif,
            method = "L-BFGS-B",
            lower = rep(0, d), upper = rep(1, d),
            X.unique = mod$X0, mult = mod$mult,
            dist = dist, cov_type = cov_type,
            nu_hat = mod$nu_hat, theta = mod$theta, noise = mod$g
        )

        Xnew <- matrix(
            c(
                out$p1[which.min(out$value)],
                out$p2[which.min(out$value)]
            ),
            nrow = 1
        )
        Ynew <- y(Xnew, f = f)

        mod <- update(mod, Xnew = Xnew, Znew = Ynew, ginit = mod$g * 1.01)
        # print(sprintf("iteration: %d, Xnew: (%f, %f), theta: (%f, %f), nu_hat: %f, noise: %
    }
    print(sprintf("Exact IMSE train time: %f", Sys.time() - start))

    return(mod)
}
```

```r
train_IMSPE <- function(mod, f, step = 2) {
    start <- Sys.time()

    for (k in 1:step) {
        out <- IMSPE_optim(mod, h = 1)

        Xnew <- matrix(out$par, nrow = 1)
        Ynew <- apply(Xnew, 1, y, f = f)

        mod <- update(mod, Xnew = Xnew, Znew = Ynew, ginit = mod$g * 1.01)

        # print(sprintf("iteration: %d, Xnew: (%f, %f), theta: (%f, %f), nu_hat: %f, noise: %
    }
    print(sprintf("IMSPE train time: %f", Sys.time() - start))

    return(mod)
}
```

4.1 1-D uniform experiment

This experiment is conducted on 1-D uniform data in [0, 1]. The input is 10 uniform points with 1 replication, 20 points in total. The benchmark function is the sphere_func. Three models were trained in 100 steps.

```r
# train HomGP model for one-dimensional data and sphere_func
mod <- mleHomGP(X = X_input, Z = Y_input, covtype = cov_type)

step <- 100

mod.laplace <- train_Laplace(mod, f, step)
```

```
[1] "Laplace train time: 14.886402"
```

```r
mod.exact <- train_exact_obj(mod, f, step)
```

```
[1] "Exact IMSE train time: 1.139440"
```

```r
mod.impse <- train_IMSPE(mod, f, step)
```

```
[1] "IMSPE train time: 12.538257"
```

```
xgrid <- matrix(seq(0, 1, length.out = 1001), ncol = 1)
ygrid <- apply(xgrid, 1, y, f = f)

predictions.laplace <- predict(x = xgrid, object = mod.laplace)$mean
predictions.exact <- predict(x = xgrid, object = mod.exact)$mean
predictions.imspe <- predict(x = xgrid, object = mod.imspe)$mean

sum(mean((predictions.laplace - ygrid)^2))
```

```
[1] 0.0001099098
```

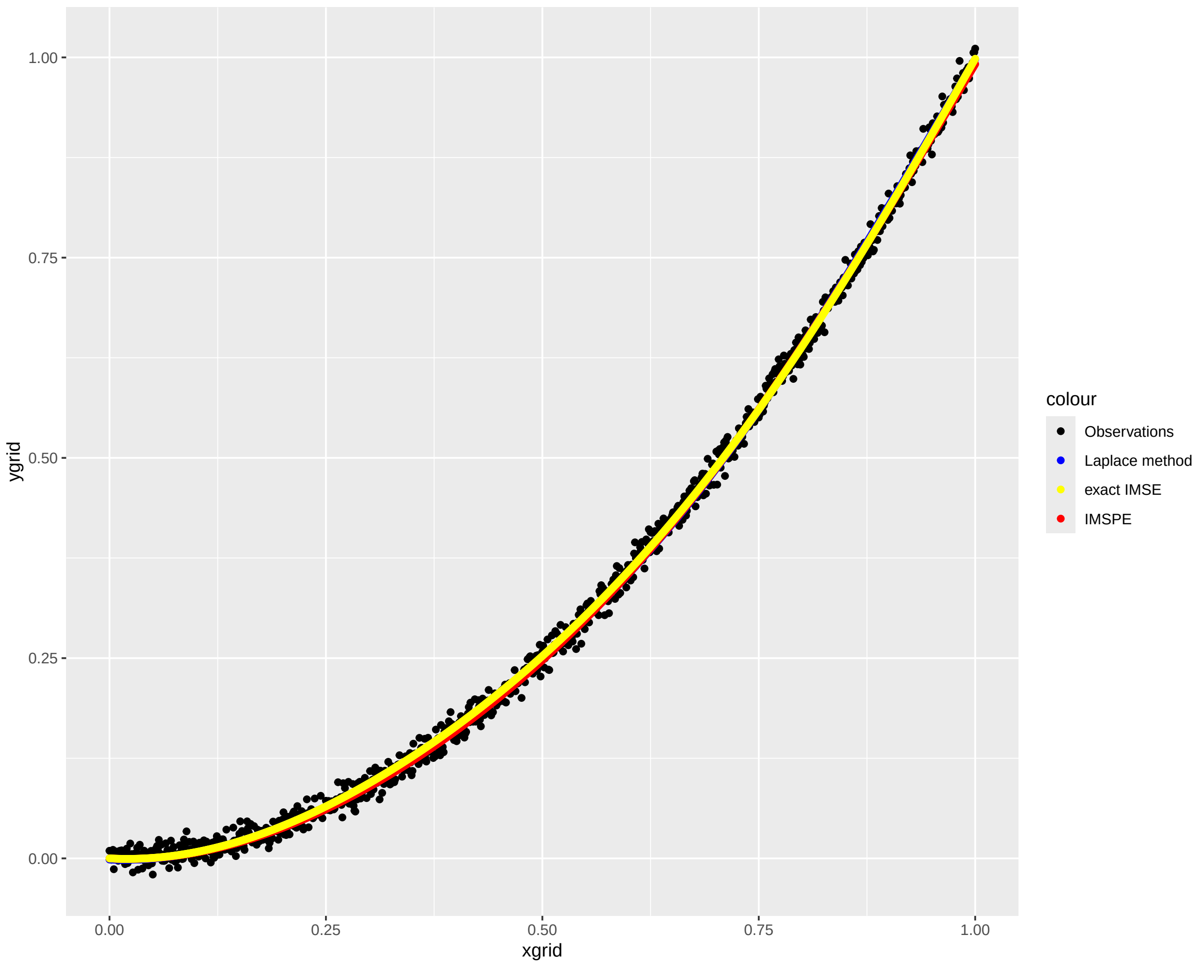
```
sum(mean((predictions.exact - ygrid)^2))
```

```
[1] 0.0001092186
```

```
sum(mean((predictions.imspe - ygrid)^2))
```

```
[1] 0.000107146
```

```
df <- data.frame(xgrid, ygrid, predictions.laplace, predictions.exact, predictions.imspe)
ggplot(df) +
    geom_point(aes(x = xgrid, y = ygrid, color = "black")) +
    geom_point(aes(x = xgrid, y = predictions.laplace, color = "blue")) +
    geom_point(aes(x = xgrid, y = predictions.exact, color = "yellow")) +
    geom_point(aes(x = xgrid, y = predictions.imspe, color = "red")) +
    scale_color_manual(
        values = c("black", "blue", "yellow", "red"),
        labels = c("Observations", "Laplace method", "exact IMSE", "IMSPE")
    )
```



The rmse values of all three models are very small, the predictions of all three models are close to underlying function, indicating a satisfactory fit.

4.2 2-D uniform experiment

The second experiment is still conducted on uniform data, while the benchmark functions are more complicated.

```r
library(gridExtra)

d <- 2
N <- 10
cov_type <- "Gaussian"
step <- 100
```

```r
X_input <- cbind(runif(N, 0, 1), runif(N, 0, 1))
X_input <- rbind(X_input, X_input)

Xstart <- maximinSA_LHS(lhsDesign(multi_start, d, seed = 2024)$design)$design

xgrid <- expand.grid(
    x1 = seq(0, 1, length.out = 200),
    x2 = seq(0, 1, length.out = 200)
)
xgrid <- cbind(xgrid[, 1], xgrid[, 2])
```

```r
f <- gaussian_pdf_func
Y_input <- apply(X_input, 1, y, f = f)
```

```r
mod <- mleHomGP(X = X_input, Z = Y_input, covtype = cov_type)

mod.laplace <- train_Laplace(mod, f, step)
```

```
[1] "Laplace train time: 4.286308"
```

```r
mod.exact <- train_exact_obj(mod, f, step)
```

```
[1] "Exact IMSE train time: 5.998980"
```

```r
mod.imspe <- train_IMSPE(mod, f, step)
```

```
[1] "IMSPE train time: 1.097560"
```

```r
ygrid <- apply(xgrid, 1, y, f = f)

predictions.laplace <- predict(x = xgrid, object = mod.laplace)$mean
predictions.exact <- predict(x = xgrid, object = mod.exact)$mean
predictions.imspe <- predict(x = xgrid, object = mod.imspe)$mean

sum(mean((predictions.laplace - ygrid)^2))
```

```
[1] 0.008002385
```

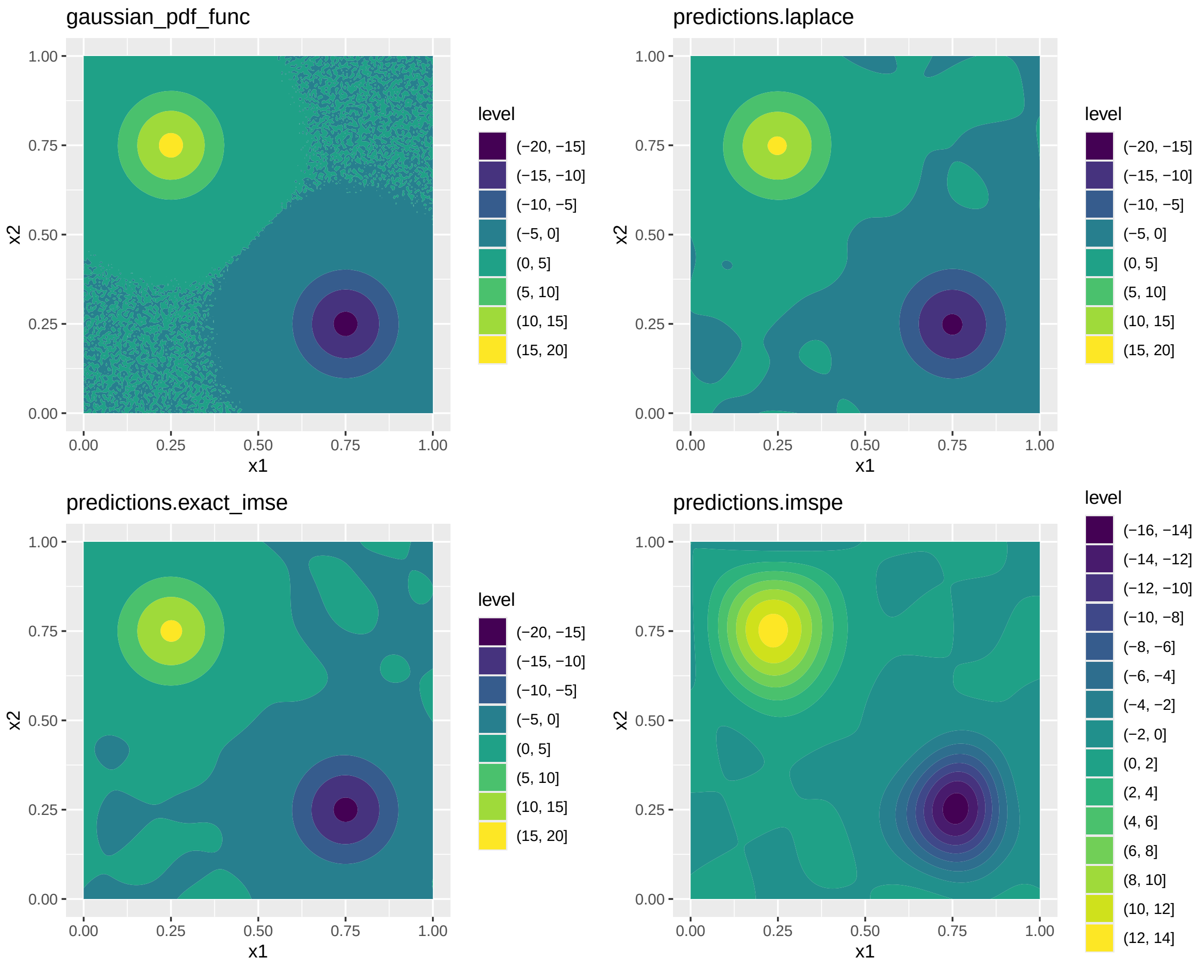
```
sum(mean((predictions.exact - ygrid)^2))
```

```
[1] 0.002700642
```

```
sum(mean((predictions.imspe - ygrid)^2))
```

```
[1] 0.2922203
```

```
df <- data.frame(
    x1 = xgrid[, 1], x2 = xgrid[, 2],
    ygrid,
    predictions.laplace, predictions.exact, predictions.imspe
)
g1 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = ygrid)) +
    ggtitle("gaussian_pdf_func")
g2 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.laplace)) +
    ggtitle("predictions.laplace")
g3 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.exact)) +
    ggtitle("predictions.exact_imse")
g4 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.imspe)) +
    ggtitle("predictions.imspe")

grid.arrange(g1, g2, g3, g4, nrow = 2)
```



For 2-D uniform input and gaussian_pdf_func function, both Laplace method and exact IMSE work well, while IMSPE method seems to fail to fit the peaks of the benchmark function. Considering that the input is uniform, it is evident that our new IMSE objective function is better than IMSPE.

Similar experiments were run on other benchmark functions.

```
f <- ackley_func
Y_input <- apply(X_input, 1, y, f = f)
```

```
mod <- mleHomGP(X = X_input, Z = Y_input, covtype = cov_type)

mod.laplace <- train_Laplace(mod, f, step)
```

```
[1] "Laplace train time: 1.724565"
```

```r
mod.exact <- train_exact_obj(mod, f, step)
```

```
[1] "Exact IMSE train time: 1.689753"
```

```r
mod.imspe <- train_IMSPE(mod, f, step)
```

```
[1] "IMSPE train time: 45.669064"
```

```r
ygrid <- apply(xgrid, 1, y, f = f)

predictions.laplace <- predict(x = xgrid, object = mod.laplace)$mean
predictions.exact <- predict(x = xgrid, object = mod.exact)$mean
predictions.imspe <- predict(x = xgrid, object = mod.imspe)$mean

sum(mean((predictions.laplace - ygrid)^2))
```

```
[1] 0.0002244792
```

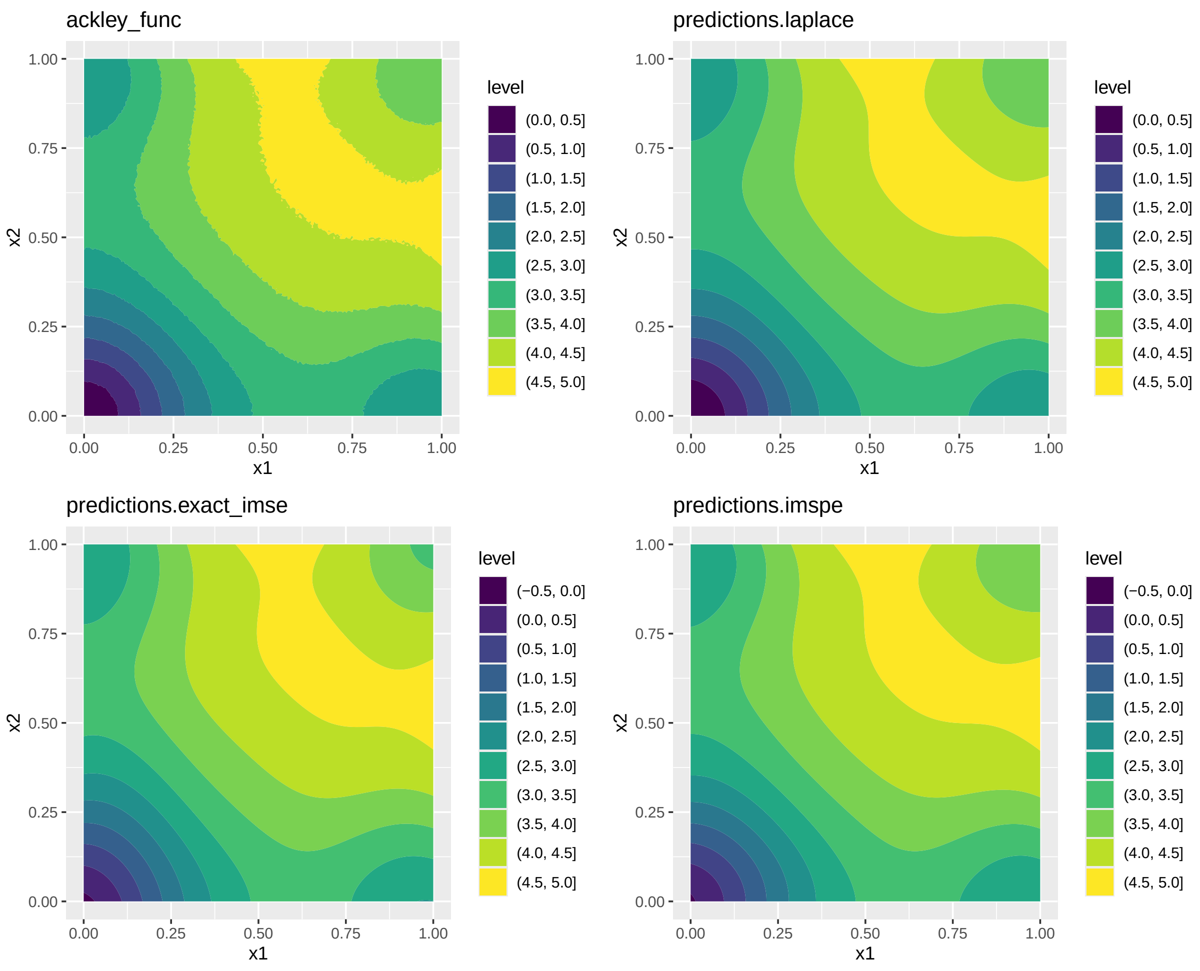
```r
sum(mean((predictions.exact - ygrid)^2))
```

```
[1] 0.0005396379
```

```r
sum(mean((predictions.imspe - ygrid)^2))
```

```
[1] 0.0002067063
```

```r
df <- data.frame(
    x1 = xgrid[, 1], x2 = xgrid[, 2],
    ygrid,
    predictions.laplace, predictions.exact, predictions.imspe
)
g1 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = ygrid)) +
    ggtitle("ackley_func")
g2 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.laplace)) +
    ggtitle("predictions.laplace")
g3 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.exact)) +
    ggtitle("predictions.exact_imse")
g4 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.imspe)) +
    ggtitle("predictions.imspe")

grid.arrange(g1, g2, g3, g4, nrow = 2)
```



```r
f <- rastrigin_func
Y_input <- apply(X_input, 1, y, f = f)
```

```r
mod <- mleHomGP(X = X_input, Z = Y_input, covtype = cov_type)

mod.laplace <- train_Laplace(mod, f, step)
```

```
[1] "Laplace train time: 2.289772"
```

```r
mod.exact <- train_exact_obj(mod, f, step)
```

```
[1] "Exact IMSE train time: 10.014296"
```

```r
mod.imspe <- train_IMSPE(mod, f, step)
```

```
[1] "IMSPE train time: 52.995250"
```

```r
ygrid <- apply(xgrid, 1, y, f = f)

predictions.laplace <- predict(x = xgrid, object = mod.laplace)$mean
predictions.exact <- predict(x = xgrid, object = mod.exact)$mean
predictions.imspe <- predict(x = xgrid, object = mod.imspe)$mean

sum(mean((predictions.laplace - ygrid)^2))
```

```
[1] 0.0001871513
```

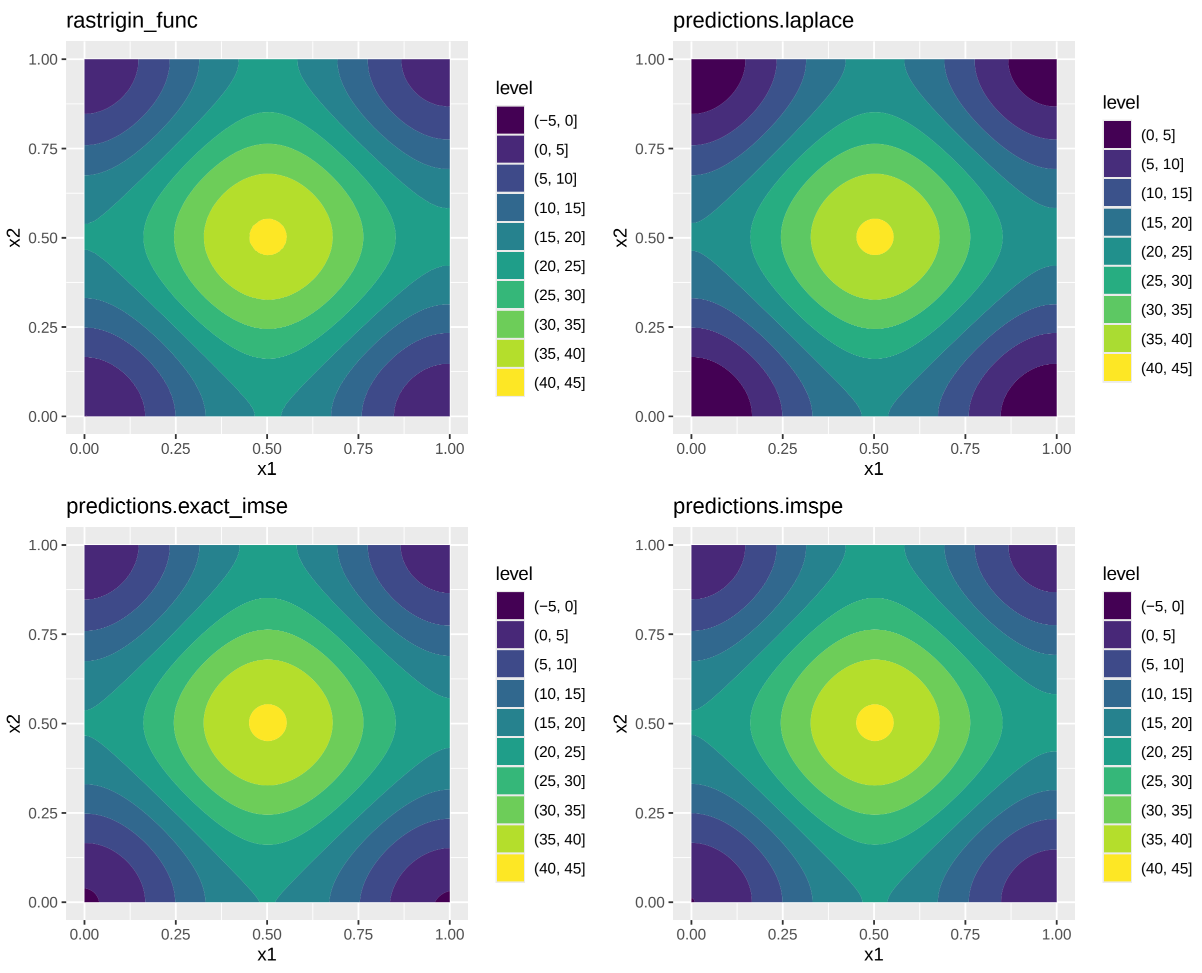
```r
sum(mean((predictions.exact - ygrid)^2))
```

```
[1] 0.01244109
```

```r
sum(mean((predictions.imspe - ygrid)^2))
```

```
[1] 0.0002055954
```

```r
df <- data.frame(
    x1 = xgrid[, 1], x2 = xgrid[, 2],
    ygrid,
    predictions.laplace, predictions.exact, predictions.imspe
)
g1 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = ygrid)) +
    ggtitle("rastrigin_func")
g2 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.laplace)) +
    ggtitle("predictions.laplace")
g3 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.exact)) +
    ggtitle("predictions.exact_imse")
g4 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.imspe)) +
    ggtitle("predictions.imspe")

grid.arrange(g1, g2, g3, g4, nrow = 2)
```



```
f <- rastrigin_adj_func
Y_input <- apply(X_input, 1, y, f = f)
```

```r
mod <- mleHomGP(X = X_input, Z = Y_input, covtype = cov_type)

step <- 200 # for this very complicated benchmark

mod.laplace <- train_Laplace(mod, f, step)
```

```
[1] "Laplace train time: 48.785497"
```

```r
mod.exact <- train_exact_obj(mod, f, step)
```

```
[1] "Exact IMSE train time: 1.258148"
```

```r
mod.imspe <- train_IMSPE(mod, f, step)
```

```
[1] "IMSPE train time: 2.473479"
```

```r
ygrid <- apply(xgrid, 1, y, f = f)

predictions.laplace <- predict(x = xgrid, object = mod.laplace)$mean
predictions.exact <- predict(x = xgrid, object = mod.exact)$mean
predictions.imspe <- predict(x = xgrid, object = mod.imspe)$mean

sum(mean((predictions.laplace - ygrid)^2))
```

```
[1] 0.3484994
```

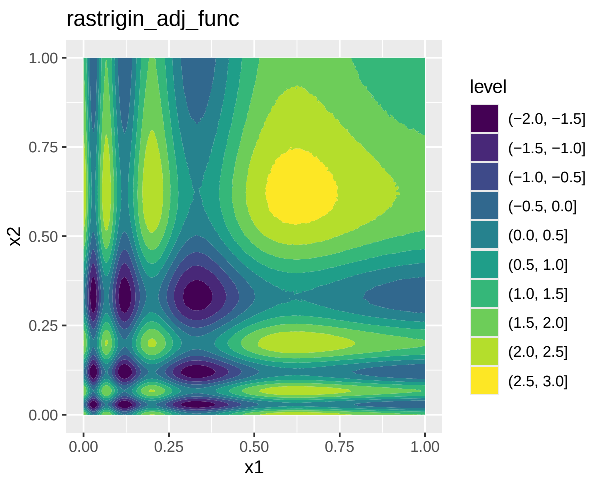
```r
sum(mean((predictions.exact - ygrid)^2))
```

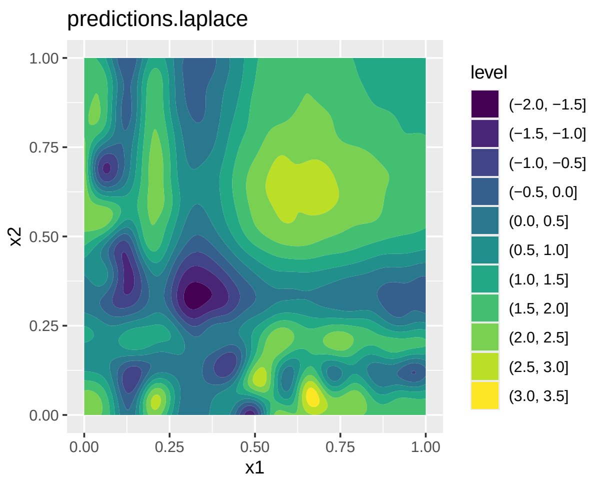
```
[1] 0.3261607
```

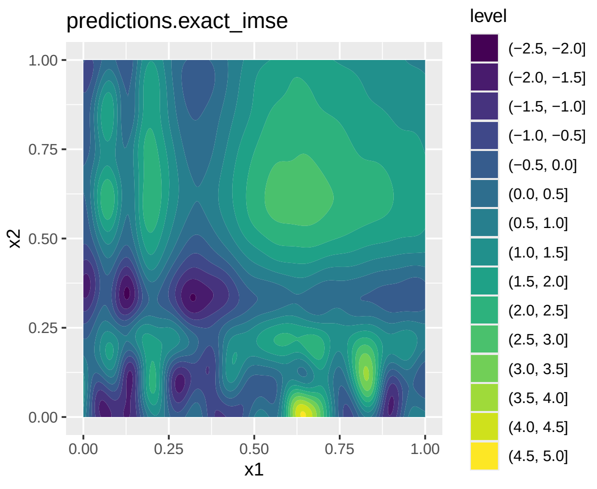
```r
sum(mean((predictions.imspe - ygrid)^2))
```

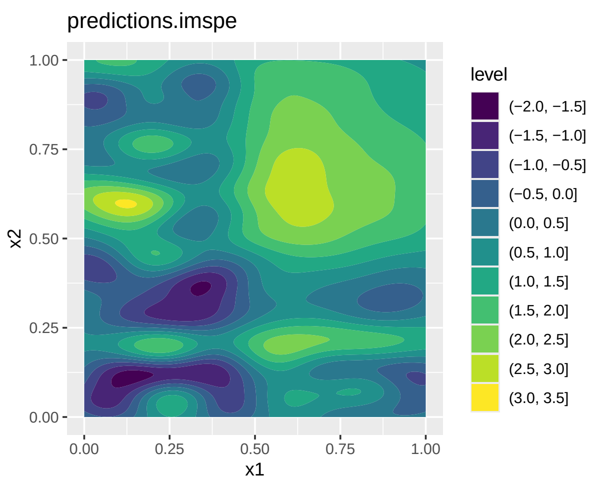
```
[1] 0.3699432
```

```r
df <- data.frame(
    x1 = xgrid[, 1], x2 = xgrid[, 2],
    ygrid,
    predictions.laplace, predictions.exact, predictions.imspe
)
g1 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = ygrid)) +
    ggtitle("rastrigin_adj_func")
g2 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.laplace)) +
    ggtitle("predictions.laplace")
g3 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.exact)) +
    ggtitle("predictions.exact_imse")
g4 <- ggplot(df) +
    geom_contour_filled(aes(x1, x2, z = predictions.imspe)) +
    ggtitle("predictions.imspe")

grid.arrange(g1, g2, g3, g4, nrow = 2)
```









In conclusion, for uniform input, the exact value of our IMSE objective function can be calculated by Wij function in hetGP package, our objective function works better than IMSE.