Simulation of Chen

Qi Wang

2023-04-14

In this section, we are going to reproduce the same result as Chen did in the deep Kriging paper. For 1d case, they simulated the z in this way:

$$z(s) = \mu + \nu(s) + \epsilon(s)$$

with $\mu = 1$, and $\nu(s)$ being a zero mean GP with an exponential covariance function:

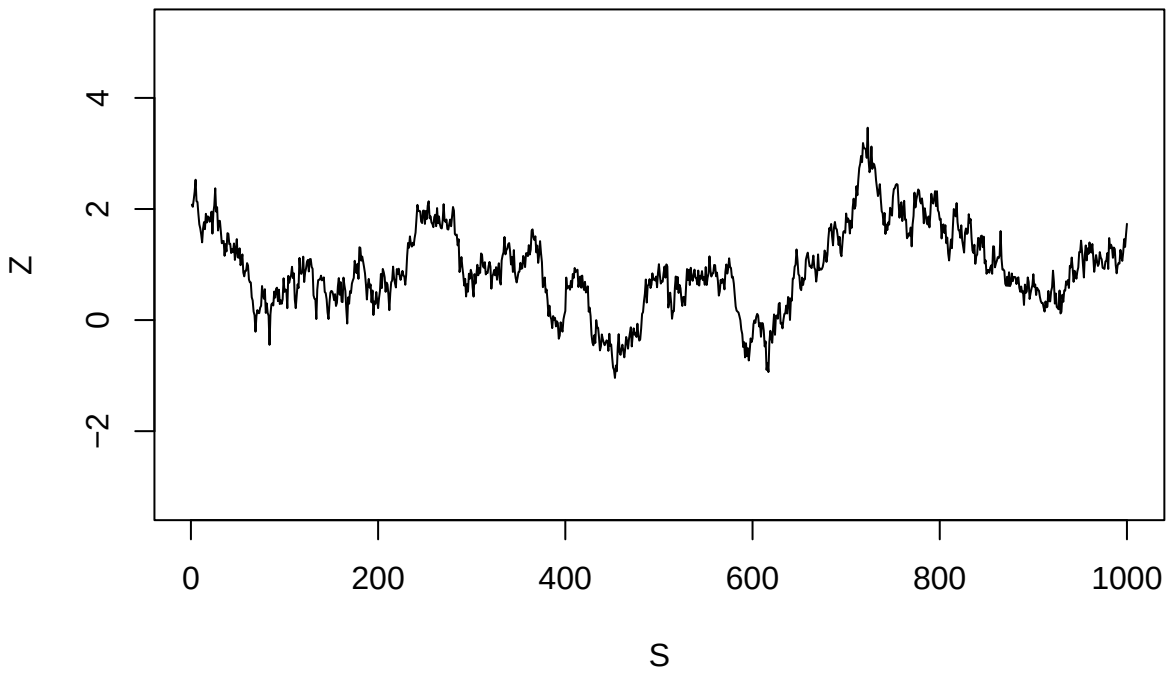
$$C(s, s') = \sigma^2 exp\{-|s - s'|/\rho\}$$

with variance $\sigma^2 = 1$ and the range parameter $\rho = 0.1$, and the ϵ is a Gaussian white noise with the nugget variance $\tau^2 = 0.01$.

Step 1

They generated 100 replicates for $z(s_1), z(s_2), ..., z(s_N)$ with $N = 1000$, and equally spaced locations over $[0, 1]$, with 800 locations randomly selected as training data and the rest are testing data. Here we have no observed covariates except for the intercept.

```
# Generate the GP

sim_all <- matrix(NA, nrow = 100, ncol = 1000)
s <- seq(from = 0, to = 1, length.out = 1000)

exp_cor <- function(d){
  return(exp(-abs(d)/0.1))
}
#
dis <- spDists(cbind(s, 1))
cov_mat <- exp_cor(dis)
# for (i in 1:100) {
#   sim_all[i,] <- mvtnorm::rmvnorm(1, mean = rep(1,1000), sigma = cov_mat) + rnorm(1000, mean = 0, sd =
#   print(i)
# }
# write.csv(sim_all, here::here("pm_small/sim_dat.csv"))

sim_dat <- read.csv(here::here("pm_small/sim_dat.csv"))[,-1]

plot(x = 1:1000, sim_dat[1,], type = 'l', ylim = c(min(sim_dat)-0.3, max(sim_dat)+0.3), ylab = "Z", xlab
```




```
# Create basis function for DK
  basis_1 <- seq(from = 0, to = 1, length.out = 10)
  basis_2 <- seq(from = 0, to = 1, length.out = 19)
  basis_3 <- seq(from = 0, to = 1, length.out = 37)
  basis_4 <- seq(from = 0, to = 1, length.out = 73)

  basis_fun_1 <- matrix(NA, nrow = 1000, ncol = length(basis_1))
  basis_fun_2 <- matrix(NA, nrow = 1000, ncol = length(basis_2))
  basis_fun_3 <- matrix(NA, nrow = 1000, ncol = length(basis_3))
  basis_fun_4 <- matrix(NA, nrow = 1000, ncol = length(basis_4))

  for (i in 1:1000) {

    for (j in 1:length(basis_1)) {
      basis_fun_1[i,j] <- nychka_fun(center_coords = cbind(1, basis_1)[j,], obs_coords = cbind(1,s)[i,]
                                     theta = diff(basis_1)[1])
    }
     for (j in 1:length(basis_2)) {
      basis_fun_2[i,j] <- nychka_fun(center_coords = cbind(1, basis_2)[j,], obs_coords = cbind(1,s)[i,]
                                     theta = diff(basis_2)[1])
     }
     for (j in 1:length(basis_3)) {
      basis_fun_3[i,j] <- nychka_fun(center_coords = cbind(1, basis_3)[j,], obs_coords = cbind(1,s)[i,]
                                     theta = diff(basis_3)[1])
     }
     for (j in 1:length(basis_4)) {
```

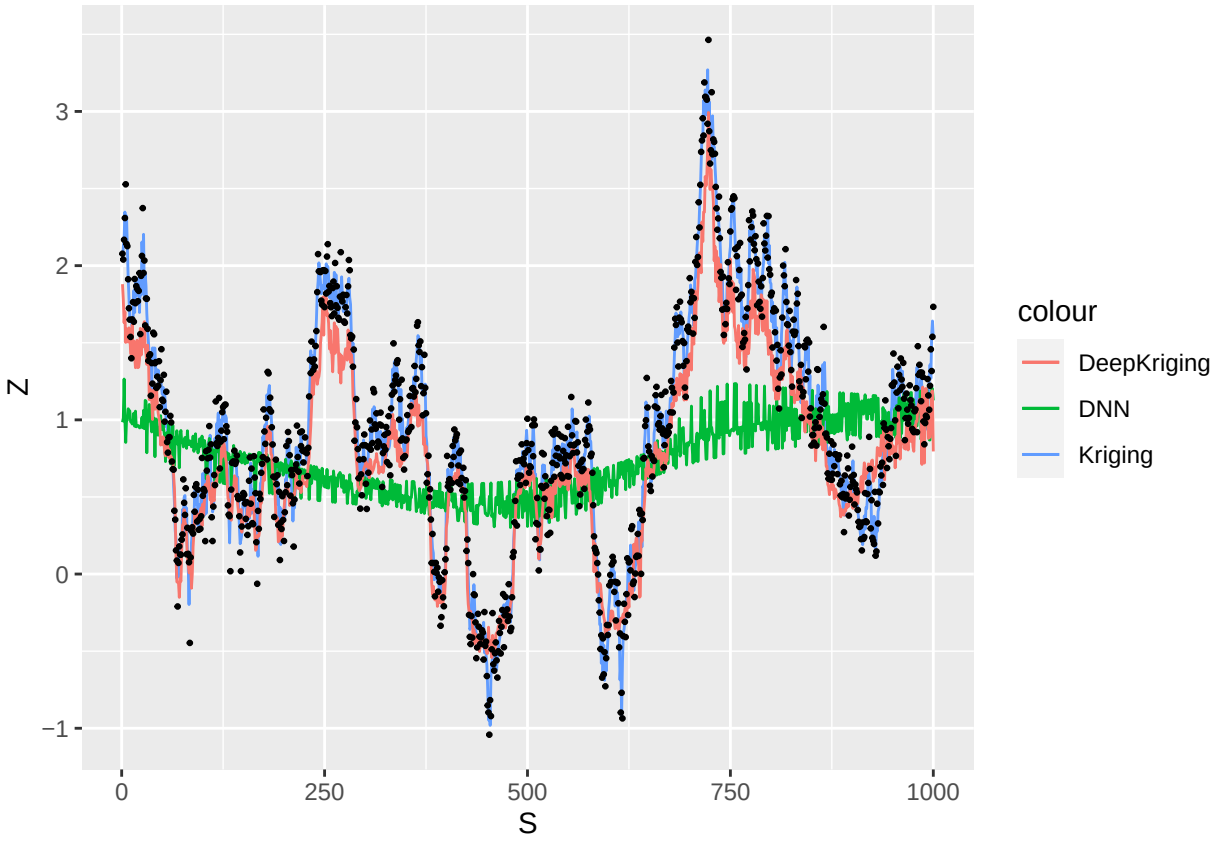
```
      basis_fun_4[i,j] <- nychka_fun(center_coords = cbind(1, basis_4)[j,], obs_coords = cbind(1,s)[i,]
                                     theta = diff(basis_4)[1])
    }

  }


  basis_fun_all <- cbind(basis_fun_1,basis_fun_2,basis_fun_3,basis_fun_4)
```

```
# Classical Kriging with true covariance function
rep_idx <- 1

train_all_index <- sample(1:10, 1000, replace = TRUE)

krig_mean_all <- rep(NA, 1000)
dkrig_mean_all <- rep(NA, 1000)
nn_mean_all <- rep(NA, 1000)

for (curr_index in 1:10) {

  train_index <- which(train_all_index != curr_index)
  train_x <- s[train_index]
  test_x <- s[-train_index]

  # Classical Kriging
  exp_sig_11 <- cov_mat[train_index, train_index]
  exp_sig_12 <- cov_mat[train_index, -train_index]
  exp_sig_21 <- t(exp_sig_12)
  exp_sig_22 <- cov_mat[-train_index, -train_index]
  krig_mean_all[-train_index] <- 1 + exp_sig_21 %*% solve(exp_sig_11) %*%
                                 matrix( as.numeric(sim_dat[rep_idx,train_index] - 1), ncol = 1 )


  # dnn_mean_all

  x_tr <- cbind(1, matrix(as.numeric(train_x), ncol = 1))
  x_te <- cbind(1, matrix(as.numeric(test_x), ncol = 1))
  x_tr <- array_reshape(x_tr, c(nrow(x_tr), 2))
  x_te <- array_reshape(x_te, c(nrow(x_te), 2))

  z_tr <- as.numeric(sim_dat[rep_idx,train_index])
  z_te <- as.numeric(sim_dat[rep_idx,-train_index])
  model_dnn <- keras_model_sequential()
  model_dnn %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(ncol(x_tr))) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 256, activation = 'relu') %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 256, activation = 'relu') %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 1, activation = 'linear')
```

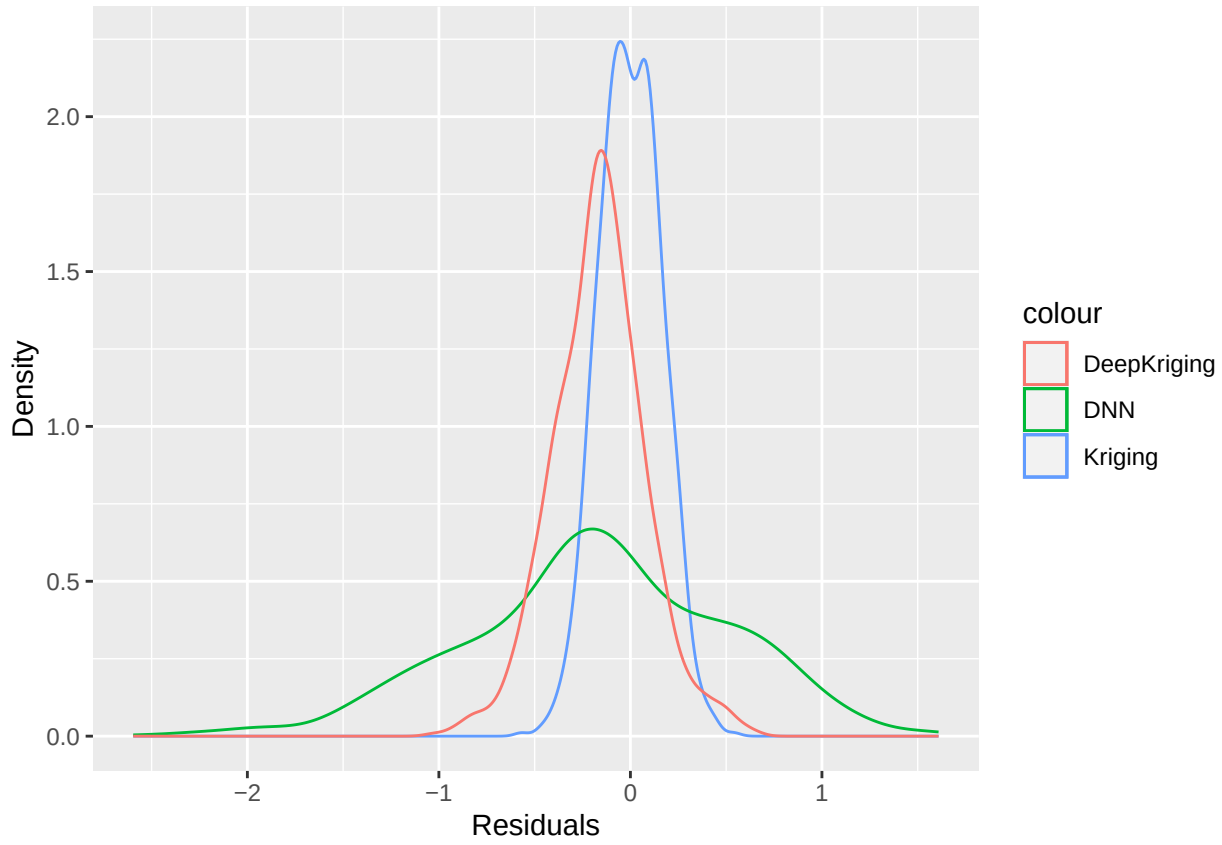
```
  model_dnn %>% compile(
    loss = "mse",
    optimizer = optimizer_adam(),
    metrics = list("mse")
  )

  mod_train_dnn <- model_dnn %>%
    fit(x = x_tr, y = z_tr, epochs = 30, batch_size = 128)

  nn_mean_all[-train_index] <- predict(model_dnn, x_te)

  # deep Kriging with Basis Function

  x_dk <- cbind(1, matrix(as.numeric(s), ncol = 1), basis_fun_all)

  train_x <- x_dk[train_index,]
  test_x <- x_dk[-train_index,]

  x_tr <- array_reshape(train_x, c(nrow(train_x), ncol(train_x)))
  x_te <- array_reshape(test_x, c(nrow(test_x), ncol(test_x)))

  z_tr <- as.numeric(sim_dat[rep_idx,train_index])
  z_te <- as.numeric(sim_dat[rep_idx,-train_index])

  model_dk <- keras_model_sequential()

  model_dk %>%
  layer_dense(units = 256, activation = 'relu', input_shape = c(ncol(x_tr))) %>%
  layer_dropout(rate = 0.4) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 128, activation = 'relu') %>%
  layer_dropout(rate = 0.3) %>%
  layer_dense(units = 1, activation = 'linear')

  model_dk %>% compile(
    loss = "mse",
    optimizer = optimizer_adam(),
    metrics = list("mse")
  )

  mod_train_dk <- model_dk %>%
    fit(x = x_tr, y = z_tr, epochs = 30, batch_size = 128, validation_split = 0.1)

  dkrig_mean_all[-train_index] <- predict(model_dk, x_te)

}
```

```
ggplot() +
  geom_path(aes(x = 1:1000, y = krig_mean_all, color = "Kriging")) +
  geom_path(aes(x = 1:1000, y = nn_mean_all, color = "DNN")) +
```

```
  geom_path(aes(x = 1:1000, y = dkrig_mean_all, color = "DeepKriging"))+
  geom_point( aes(x = 1:1000, y = as.numeric(sim_dat[rep_idx,])), size = 0.5) +
  labs(x = "S", y = "Z")
```




```
res_krig <- as.numeric(krig_mean_all - sim_dat[rep_idx,])
res_dnn <- as.numeric(nn_mean_all - sim_dat[rep_idx,])
res_dkrig <- as.numeric(dkrig_mean_all - sim_dat[rep_idx,])

ggplot() +
  geom_density(aes(x = as.numeric(res_krig), color = "Kriging")) +
  labs(x = "Residuals", y = "Density") +
  geom_density(aes(x = as.numeric(res_dnn), color = "DNN")) +
  geom_density(aes(x = as.numeric(res_dkrig), color = "DeepKriging"))
```




```
mean(res_krig^2)
```

```
## [1] 0.02577472
```

```
mean(res_dnn^2)
```

```
## [1] 0.4876624
```

```
mean(res_dkrig^2)
```

```
## [1] 0.08644099
```