Overview

In this vignette, I demonstrate the FCD sampler for the σ_k^2 parameters in the covariance model with dense eigenvectors. If we consider all of the other parameters in the model as fixed, then in this case, the individual FCDs are essentially like marginal posterior distributions, which we can determine exactly. The sampler does reproduce these known posterior distributions. Further, when repeated samples are taken from the data distribution, we see that the posterior mean is a good estimator, and 95% credible intervals often contain the true parameter value.

Model description

Data model: Data matrices Y_k are $P \times P$ Hermitian positive definite. n_k is the number of samples corresponding to data matrix Y_k. M_k is a known $P \times P$ Hermitian positive definite matrix.

$$Y_k|\sigma_k^2, M_k \sim \text{ComplexWishart}(n_k, \sigma_k^2 M_k)$$

Prior: The independence Jeffreys Prior is $p(\sigma_k^2) \propto 1/\sigma_k^2$.

FCD:

$$\begin{aligned} p(\sigma_k^2|Y_k) &\propto p(Y_k|\sigma_k^2, M_k)p(\sigma_k^2) \\ &\propto |Y_k|^{n_k-P}|\sigma_k^2 M_k|^{-n_k} etr[-(\sigma_k^2 M_k)^{-1}Y_k](\sigma_k^2)^{-1} \\ &\propto (\sigma_k^2)^{-Pn_k-1} \exp\left[\frac{1}{\sigma_k^2} tr(M_k^{-1}Y_k)\right] \\ &\Rightarrow \sigma_k^2|Y_k \sim \text{InvGamma}[Pn_k, tr(M_k^{-1}Y_k)] \end{aligned}$$

Based on properties of the Inverse Gamma distribution, we know the **posterior mean** is $E[\sigma_k^2|Y_k] = tr(M_k^{-1}Y_k)/(Pn_k - 1)$. If we consider the posterior mean as an estimator of σ_k^2, then due to the randomness in the data, the expectation of this estimator is:

$$\begin{aligned} E\left[tr(M_k^{-1}Y_k)/(Pn_k-1)\right] &= (Pn_k-1)^{-1} tr[E(M_k^{-1}Y_k)] \\ &= (Pn_k-1)^{-1} tr[M_k^{-1} n_k \sigma_k^2 M_k] \\ &= \frac{Pn_k}{Pn_k-1}\sigma_k^2. \end{aligned}$$

Thus, the posterior mean becomes less biased as n_k increases, and does not depend on M_k.

Compare sampler to known posterior

In this section, I will show that the sampler function does indeed sample from the correct distribution, which we know exactly. If we generate samples with the function, their empirical density matches the exact density that I expect.

Similar eigenvalues for signal portion

The matrix M_k is generated in a similar way to that in our larger model, where the parameter to the Complex Wishart distribution is $\sigma_k^2 M_k = \sigma_k^2(U_k \Lambda_k U_k^H + I_p)$. In this case, the entries of the $d \times d$ matrix Λ_k primarily affect the eigenvalues of M_k.

In this first example, the randomly generated Lambda values are low, all below 2. Note that for, say, the M_1 matrix, the "Summary of eigenvalues" results shows that the eigenvalues are all on a similar order of magnitude. The same is true for the M_2 and M_3 matrices.

In the demonstration, a data matrix Y_k is generated for each of the K groups. The function `sigmak2_gibbs_densCovar` is used to generate samples from the respective posteriors. After discarding half the generated samples as a burn-in, we can calculate the posterior mean. We can compare these to the exact posterior means of the known Inverse Gamma distributions, and also to the true σ^2_{k0} values.

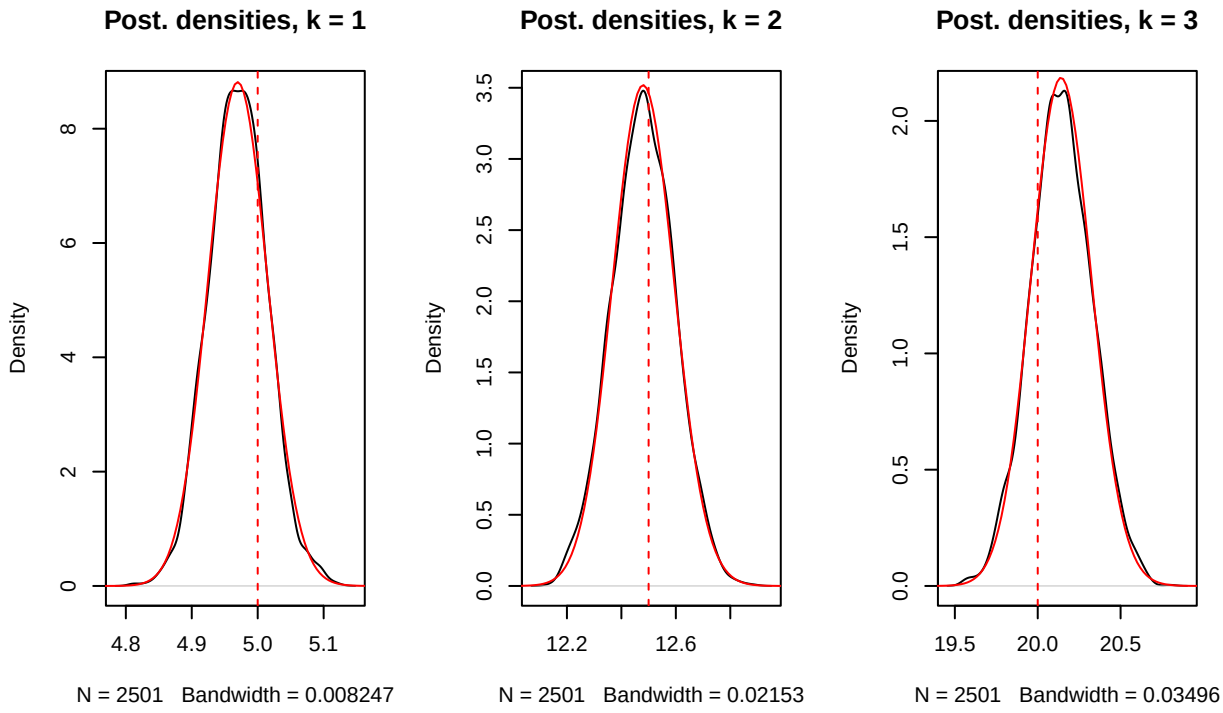
The exact and sampled posterior means are very close. They are all close to the true σ^2_{k0} values. The sampled densities are all very close to the true Inverse Gamma densities.

```
P <- 12
d <- 4
K <- 3
nk_scale <- 1000
gibbs_its <- 5000

set.seed(13032025)
Lambda_ks <- array(NA, c(d, K))
# These values lead to M_k matrices with eigenvalues from about 3 to 1.
for(k in 1:K) {
    Lambda_ks[, k] <- rgamma(d, 1, 1) |> sort(decreasing = TRUE)
}

# The generated Lambda values. Read down column k to see the values for group k.
print(Lambda_ks)
```

```
##           [,1]      [,2]      [,3]
## [1,] 0.8475202 1.1770745 1.6116179
## [2,] 0.4274250 1.0677248 0.5785050
## [3,] 0.3464576 0.5847943 0.5508674
## [4,] 0.0021038 0.5117572 0.3574090
```

```
demo_FCD(Lambda_ks)
```

```
## [1] "Summary of eigenvalues for M_1 matrix"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.135   1.088   1.848
## [1] "Summary of eigenvalues for M_2 matrix"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.278   1.530   2.177
## [1] "Summary of eigenvalues for M_3 matrix"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   1.258   1.406   2.612
## [1] "Exact posterior means"
## [1] 4.970416
## [1] 12.4819
## [1] 20.14396
## [1] "Sample posterior means"
## [1]  4.970409 12.480646 20.143584
## [1] "Known sigma_k2 values"
## [1]  5.0 12.5 20.0
```



Different magnitudes of signal eigenvalues

In this case, we generate the Lambda values in a different way. The result is that the M_k matrix eigenvalues are spread across different orders of magnitude.

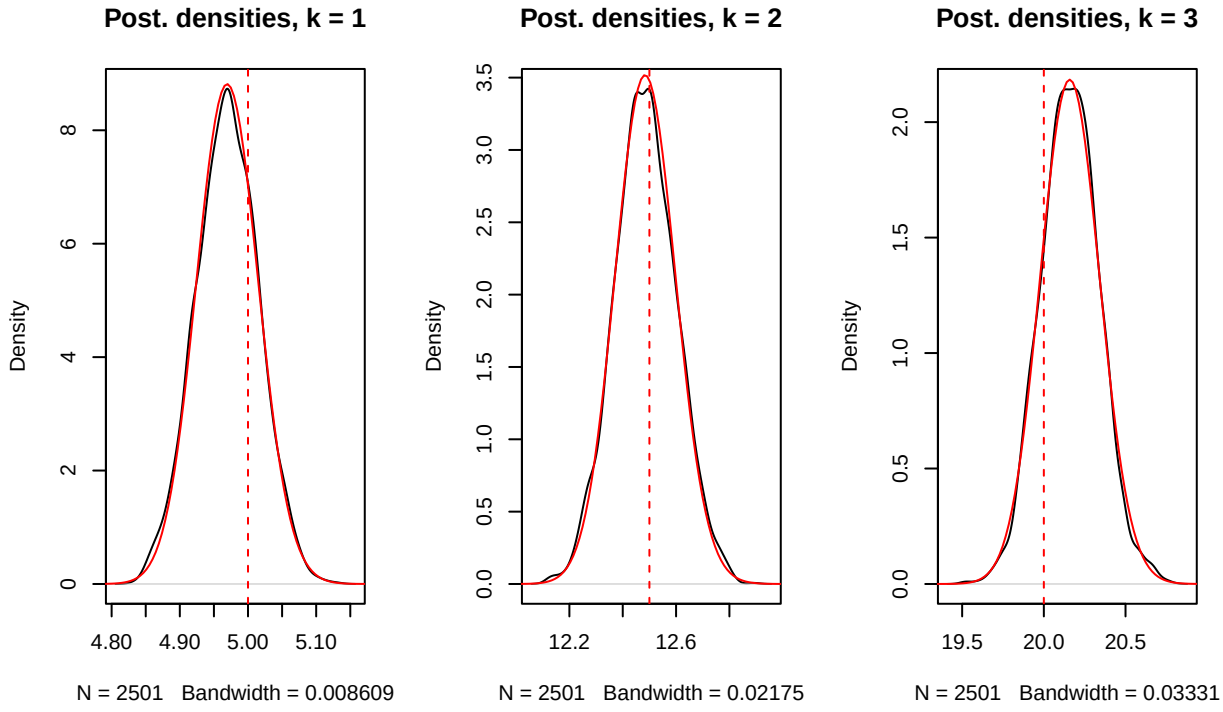
The exact and sample posterior means match, and the observed and exact densities match. Further, the posterior means and distributions are close to the true σ^2_{k0} values.

```
set.seed(13032025)
Lambda_ks <- array(NA, c(d, K))
# Different Lambda values are generated, to investigate effect on estimator
# quality. These values lead to eigenvalues on different scales.
for(k in 1:K) {
    Lambda_ks[, k] <- seq(5, 10^(k), length.out = d) |> sort(decreasing = TRUE)
}

# Read down column k to see the values for group k.
print(Lambda_ks)
```

```
##           [,1]      [,2]      [,3]
## [1,] 10.000000 100.00000 1000.0000
## [2,]  8.333333  68.33333  668.3333
## [3,]  6.666667  36.66667  336.6667
## [4,]  5.000000   5.00000    5.0000
```

```
demo_FCD(Lambda_ks)
```

```
## [1] "Summary of eigenvalues for M_1 matrix"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.000   3.500   6.417  11.000
## [1] "Summary of eigenvalues for M_2 matrix"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    1.00   18.50   13.92  101.00
## [1] "Summary of eigenvalues for M_3 matrix"
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    1.00    1.00  168.50   88.92 1001.00
## [1] "Exact posterior means"
## [1] 4.970209
## [1] 12.48563
## [1] 20.16141
## [1] "Sample posterior means"
## [1]  4.969599 12.485282 20.161459
## [1] "Known sigma_k2 values"
## [1]  5.0 12.5 20.0
```



Evaluating the distribution of the posterior mean

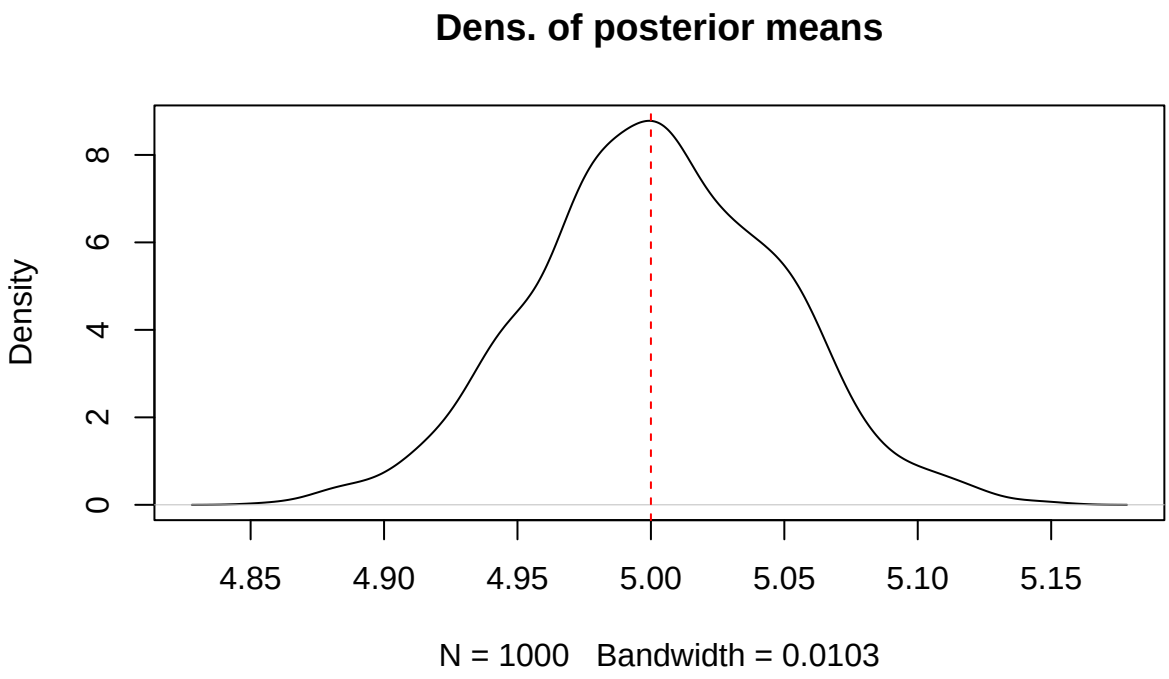
In this section, we consider whether the posterior distributions which result from randomly observed data will "recover" a known true σ_k^2 value. That is, if we start with a known σ_k^2 value and generate multiple observations of Y_k using that value, how do the resulting posterior distributions behave, due to the randomness in the data Y_k? Are the posterior means, $E[\sigma_k^2|Y_k] = tr(M_k^{-1}Y_k)/(Pn_k - 1)$, close to the true parameter value? Do posterior credible intervals often contain the true parameter value?

A case somewhat close

First, we look at a case where the entries of Λ_k are on different scales (1000 to 20). The true parameter, $\sigma_0^2 = 5$, is within the 95% credible intervals 94.9% of the time. The mean of the posterior means is 5.002, which matches the expectation that the posterior mean is a slightly positively biased estimator, i.e. since $\sigma_0^2(Pn)/(Pn-1) = 5.000415$ in this case.

```
# this demo has eigenvalues on different scales, and has decent coverage of the
# true sigma2 parameter.

# true parameter value
sigma_02 <- 5

n <- 1005 # df of the data generating CW distribution
ex_its <- 1000 # number of samples to draw
Mk_df <- 1000 # df of the CW dist. that generates the Mk parameter
hpd_round <- 11 # digits to round the Mk matrix, to ensure it is exactly hpd

# generate the parameter matrix for the data generating CW distribution
set.seed(14032025)
Mk <- rcomplex_wishart(Mk_df, P, diag(P))

# modify the eigenvalues, to have a similar scale
Mk <- eigen(Mk)$vectors %*%
    diag(seq(1000, 20, length.out = P)) %*%
    t(Conj(eigen(Mk)$vectors))

# a function that generates samples Pk, and calculates posterior quantities
demo_random_post(Mk, sigma_02, n, Mk_df, ex_its, hpd_round = hpd_round)
```

```
## [1] "Summary of eigenvalues for M_k matrix: "
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      20     265     510     510     755    1000
## [1] "True sigma2 is within 95% cred. int. 94.9% of the time"
## [1] "Median .025 posterior quantiles: 4.914"
## [1] "Median .975 posterior quantiles: 5.092"
## [1] "Mean of posterior means: 5.002"
## [1] "Median of posterior means: 5.002"
```



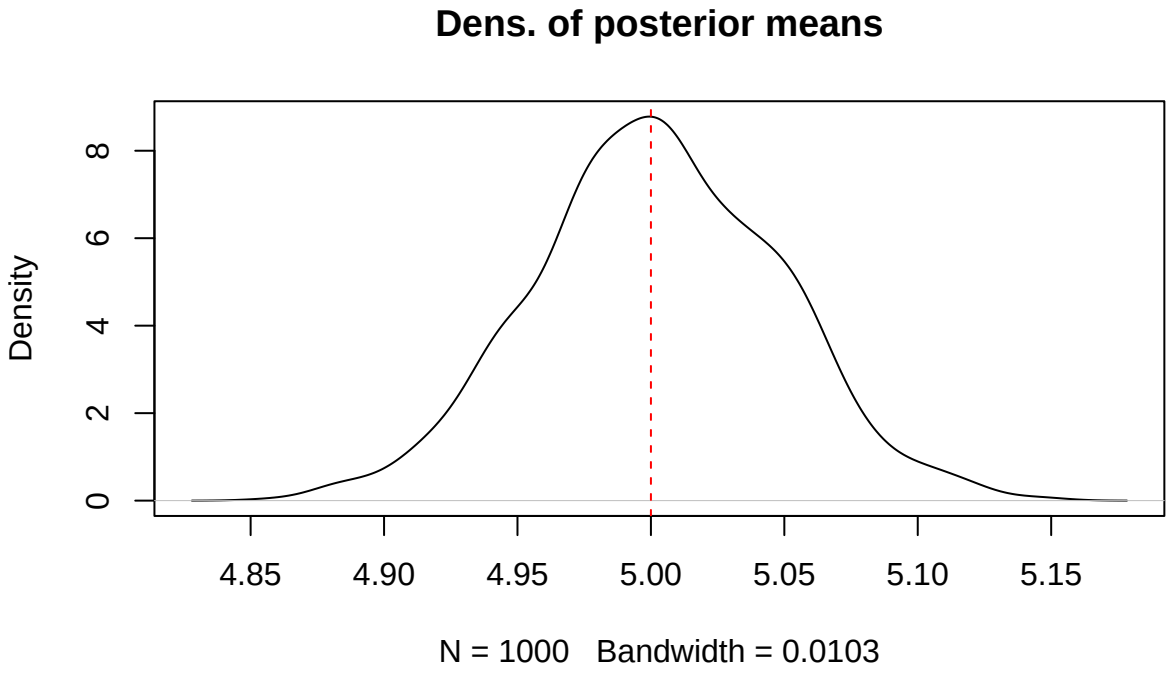
Eigenvalues similar to a previous example

Next, we investigate a similar setup, but with eigenvalues in the CW covariance matrix that are similar to those in the $k = 1$ group from the previous section. That is, the first $P - d = 8$ eigenvalues range from 1.848 to 1.0021, and the remaining $d = 4$ eigenvalues are 1.

Similar to the previous case, σ_0^2 is within the 95% credible intervals 94.9% of the time, and the mean of the posterior means is 5.002.

```
# this example recreates the eigenvalues from one of the previous examples, and
# similarly has good coverage and posterior means.

sigma_02 <- 5

n <- 1005
ex_its <- 1000
Mk_df <- 1000
hpd_round <- 11

set.seed(14032025)
Mk <- rcomplex_wishart(Mk_df, P, diag(P))

# set eigenvalues to be similar to those from a previous example, with the
# final d eigenvalues repeated.
Mk <- eigen(Mk)$vectors %*%
    diag(c(seq(1.848, 1.0021, length.out = P-d), rep(1, d))) %*%
    t(Conj(eigen(Mk)$vectors))

demo_random_post(Mk, sigma_02, n, Mk_df, ex_its, hpd_round = hpd_round)
```

```
## [1] "Summary of eigenvalues for M_k matrix: "
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.000   1.183   1.283   1.516   1.848
## [1] "True sigma2 is within 95% cred. int. 94.9% of the time"
## [1] "Median .025 posterior quantiles: 4.914"
## [1] "Median .975 posterior quantiles: 5.092"
## [1] "Mean of posterior means: 5.002"
## [1] "Median of posterior means: 5.002"
```



Appendix

```
#####
# sigmak2_gibbs_densCovar
#####

sigmak2_gibbs_densCovar
```

```
## function (data_list, param_list)
## {
##     result_sigmas <- vector(mode = "numeric", K)
##     IP <- diag(param_list$P)
##     for (k in 1:K) {
##         Pk <- data_list[[k]]
##         Uk <- param_list$U_ks[, , k]
##         Lambdak <- diag(param_list$Lambda_ks[, k])
##         Mk <- Uk %*% Lambdak %*% t(Conj(Uk)) + IP
##         ak <- .ak_sigmak2_densCovar(P, param_list$n_k[k])
##         bk <- .bk_sigmak2_densCovar(Mk, Pk)
##         result_sigmas[k] <- 1/rgamma(1, ak, rate = bk)
##     }
##     result_sigmas
## }
## <bytecode: 0x559b70867290>
```

```
#####
# .ak_sigmak2_densCovar
#####

.ak_sigmak2_densCovar
```

```
## function (P, nk)
## {
##     return(P * nk)
## }
## <bytecode: 0x559b708fa718>
```

```
#####
# .bk_sigmak2_densCovar
#####

.bk_sigmak2_densCovar
```

```
## function (Mk, Pk)
## {
##     return(Re(sum(diag((solve(Mk) %*% Pk)))))
## }
## <bytecode: 0x559b70a23b88>
```

```
#####
# Posterior demonstration function
#
# Generates true parameter values and data using those values.
# Draws samples from posterior using the sampling function to be validated.
# Calculates observed and exact posterior means, and plots observed and exact posterior
↪  densities.
#
#####
demo_FCD <- function(Lambda_ks, P = 12, d = 4, K = 3, nk_scale = 1000,
                     gibbs_its = 5000) {

    # degrees of freedom of the data generating CW distributions
    n_k <- nk_scale + 5*(1:K)

    # stores the K data matrices
    data_list <- list()

    # stores simulated U_k eigenvector matrices
    U_ks <- array(NA, c(P, d, K))

    # known sigma_k2 values, ranging from 5 to 20
    sigma_k20 <- seq(5, 20, length.out = K)

    # generate simulated parameters and data matrices
    for (k in 1:K) {
        # U_k matrices are semi-unitary
        U_ks[, , k] <- runitary(P, d)

        # temp matrix
        M_k <- diag(P) +
            U_ks[, , k] %*% diag(Lambda_ks[, k]) %*% t(Conj(U_ks[, , k]))
        M_k <- round(M_k, 9)

        # display a summary of the eigenvalues of the
        print(paste0("Summary of eigenvalues for M_",k," matrix"))
        print(summary(eigen(M_k)$values))

        data_list[[k]] <- rcomplex_wishart(n_k[k], P, sigma_k20[k] * M_k)
    }

    # sigma_k2, the parameters to be sampled
    sigma_k2s <- vector("numeric", K)

    # parameters list
    param_list <- list(P = P,
                       d = d,
                       K = K,
                       n_k = n_k,
                       U_ks = U_ks,
                       Lambda_ks = Lambda_ks,
                       sigma_k2s = sigma_k2s)

    # Test the function within a sampling loop
```

```
    # initialize an array to track the sigma_k2 values
    sigma_k2_S <- array(NA, c(K, gibbs_its))

    for (s in 1:gibbs_its) {
        sigma_k2s <- sigmak2_gibbs_densCovar(data_list, param_list)
        param_list$sigma_k2s <- sigma_k2s
        sigma_k2_S[, s] <- sigma_k2s
    }

    # in this case, we know that the distribution should be certain Inv Gammas
    # exact posterior mean
    print("Exact posterior means")
    for (k in 1:K) {
        # temp matrix
        M_k <- diag(P) +
            U_ks[, , k] %*% diag(Lambda_ks[, k]) %*% t(Conj(U_ks[, , k]))
        print(Re(sum(diag(solve(M_k) %*% data_list[[k]]))) / (P * n_k[k] - 1))
    }

    # discard the first half of samples
    post_samples <- round(gibbs_its/2):gibbs_its

    # Compare exact posterior means to the sample mean and known values
    # Sample posterior mean
    print("Sample posterior means")
    print(rowMeans(sigma_k2_S[, post_samples]))
    # Known sigmak2 values
    print("Known sigma_k2 values")
    print(sigma_k20)

    # plot observed and true densities
    par(mfrow = c(1, 3))
    for (k in 1:K) {
        xmin <- min(sigma_k2_S[k, post_samples]) * .8
        xmax <- max(sigma_k2_S[k, post_samples]) * 1.2
        plot(density(sigma_k2_S[k, post_samples]), type = "l",
             main = paste0("Post. densities, k = ",k))

        M_k <- diag(P) +
            U_ks[, , k] %*% diag(Lambda_ks[, k]) %*% t(Conj(U_ks[, , k]))

        # ak <- P*n_k[k]
        ak <- .ak_sigmak2_densCovar(P, n_k[k])
        # bk <- Re(sum(diag(solve(M_k) %*% data_list[[k]])))
        bk <- .bk_sigmak2_densCovar(M_k, data_list[[k]])
        # note the transformation required to plot inverse gamma, i.e. /x^2,
        curve(dgamma(1/x, shape = ak, rate = bk) / x^2,
          from = xmin, to = xmax, n = 501, add = TRUE,
          col = "red", ylab = "density")
        abline(v = sigma_k20[k], lty = 2, col = "red")
    }
    par(mfrow = c(1, 1))
}
```

```r
#####
# Distribution of random posteriors function
#
# demonstrates how the posterior distributions vary based on randomness of data.
#####

demo_random_post <- function(Mk, sigma_02, n, Mk_df, ex_its, hpd_round = 11) {

    P <- ncol(Mk)
    print("Summary of eigenvalues for M_k matrix: ")
    print(summary(eigen(Mk)$values))

    # parameter to the data generating CW distribution
    G0 <- round(sigma_02 * Mk, hpd_round)

    # pre-allocated arrays for posterior summaries
    post_means <- rep(NA, ex_its)
    post_CIs <- matrix(NA, ex_its, 2)

    # for different observed data matrices Pk, calculate posterior summaries
    for (i in 1:ex_its) {
        # generate a new data observation
        Pk <- rcomplex_wishart(n, P, G0)

        # calculate the Inverse Gamma parameters
        ak <- .ak_sigmak2_densCovar(P, n)
        bk <- .bk_sigmak2_densCovar(Mk, Pk)

        # calculate the exact Inverse Gamma mean and .025, .975 quantiles
        post_means[i] <- bk / (ak-1)
        post_CIs[i, 1] <- 1/qgamma(.975, shape = ak, rate = bk)
        post_CIs[i, 2] <- 1/qgamma(.025, shape = ak, rate = bk)
    }

    # Print summaries of the posteriors
    print(
      paste0("True sigma2 is within 95% cred. int. ",
             round(100*mean((post_CIs[, 1] <= sigma_02) & (sigma_02 <= post_CIs[, 2])),
             ↪  2),
             "% of the time"))

    print(paste0("Median .025 posterior quantiles: ",
             median(post_CIs[, 1]) |> round(3)))
    print(paste0("Median .975 posterior quantiles: ",
             median(post_CIs[, 2]) |> round(3)))
    print(paste0("Mean of posterior means: ",
             mean(post_means) |> round(3)))
    print(paste0("Median of posterior means: ",
             median(post_means) |> round(3)))

    # Plot the distribution of the observed posterior means
    plot(density(post_means),
         main = "Dens. of posterior means")
    abline(v = sigma_02, lty = 2, col = "red")
```

```
}
```