```
SimulateClaims <- function(v, m, s_squared, epsilon, NumberOfSimulations, K) {
  # simulate claims triangle; for k in K use lognormal, for k not in K use normal distribution
  n <- length(v)
  sigma_sqared <- log(epsilon^2 + 1)
  mu <- -sigma_sqared/2
  sigma <- sqrt(sigma_sqared)
  f <- matrix(1,nrow = n, ncol = NumberOfSimulations)
  for (i in 1:(n-1)) {
    f[i,] <- rlnorm(NumberOfSimulations, meanlog = mu[i], sdlog = sigma[i])
  }
  v_hat <- matrix(0,nrow = n, ncol = NumberOfSimulations)
  for (i in 1:(n-1)) {
    v_hat[i,] <- 1 / ( 1 / v[i] * apply(f[i:n,],2,'prod'))
  }
  v_hat[n,]=v[n]
  S <- array(0, dim = c(n,n,NumberOfSimulations))
  for (i in 1:n) {
    for (k in K) {
      sigma_sqared <- log(s_squared[k]/(v[i]*m[k]^2)+1)
      mu <- log(v[i]*m[k])-sigma_sqared/2
      sigma <- sqrt(sigma_sqared)
      S[i,k,] <- rlnorm(NumberOfSimulations, meanlog = mu, sdlog = sigma)
    }
    for (k in (1:n)[-K]) {
      S[i,k,] <- rnorm(NumberOfSimulations, mean = v[i] * m[k], sd = sqrt(v[i] * s_squared[k]))
    }
  }
  rslt <- list(volumes = v_hat, Triangle = S, m = m, s_squared = s_squared, epsilon = epsilon)
}




Generalized_LR_Method_fixed_g_h <- function(S, v, K, g, h, m, s_squared, UseAllEstimators = F) {
  #####################################################################################################
  # This calculated the predictions of the generalized loss ratio method with given weights g and h
  # Function is only used to check the calculation of the MSE^\bullet via simulation
  #####################################################################################################
  #
  # Required Parameters:
  # --------------------
  # S = incremental triange
  # v = vector of volume estimates
  # K = subset of {1,...,n-1} such that S_{i,k} is lognormally distributed for all (i,k) with k\in K
  # g = list containing weight vectors \hatg_i used
  # h = list containing weight vectors \hatg_i used
  # m = vector containing estimators for m_k (used in the function only to calculate the factors gamma)
  # s_squared = vector containing estimators for s_k^2 (used in the function only to calculate the factors gamma)
  #
  # Optional Parameters:
  # --------------------
  #
  # Output:
  # -------
  # Predictions = Full triangle including predictions
  # v_hat = vector containing resulting revised volume estimates
  # m_hat = vector containing resulting estimators for m_k


  n <- length(v)

  # calculate gammas
  gam <- matrix(0, ncol = n, nrow = n)
  for (i in 1:n) {
    for (k in K) {
      gam[i,k] <- 1+ s_squared[k]/(v[i]*m[k]^2)
    }
  }

  # Check and reduce K if necessary (if not all incremental claims are positive)
  K <- CheckK(S, K)
  # Matrix J[[i]] contains the set $J_i$
  J <- CreateJ(S, K, UseAllEstimators)
  # Matrix containing the set $J$, N[i] = $#J_i$
  J_all <- J[[1]]
  N <- nrow(J[[1]])
  for (i in 2:n) {
    J_all <- rbind(J_all,J[[i]])
    N <- c(N,nrow(J[[i]]))
  }
```

```
  # Calculate estimators r_hat
  r_hat <- list()
  for (i in 1:n) {
    r_hat[[i]] <- numeric()
    for (nu in 1:N[i]) {
      j <- J[[i]][nu,2]
      k <- J[[i]][nu,3]
      if (j==0) {
        Temp <- 1/v[i]
      } else {
        Temp <- 1/v[j] * S[j,k] / S[i,k] / gam[i,k]
      }
      r_hat[[i]] <- c(r_hat[[i]], Temp)
    }
  }


  v_hat <- numeric()
  for (i in 1:n) {
    Temp <- t(g[[i]]) %*% r_hat[[i]]
    v_hat <- c(v_hat, 1/Temp)
  }

  m_hat_per_cell <- S / v_hat
  m_hat_vec <- list()
  for (k in 1:n) {
    m_hat_vec[[k]] <- m_hat_per_cell[1:(n-k+1),k]
  }


  m_hat <- numeric()
  for (k in 1:n) {
    Temp <- t(h[[k]]) %*% m_hat_vec[[k]]
    m_hat <- c(m_hat, Temp)
  }




  Predictions <- S
  for (k in 2:n) {
    for (i in (n-k+2):n) {
      Predictions[i,k] <- v_hat[i] * m_hat[k]
    }
  }



  Rslt <- list(Predictions = Predictions, v_hat = v_hat, m_hat = m_hat)
}
```