```
library(expm)

simul_Hawkes_Erlang<-function(nb_pop, nb_neuron, intensity_function, c_vec, nu_vec = rep(1,n…
{
  # The function simulates the multiclass Hawkes process model of Ditlevsen and Locherbach (…
  # “nb_pop" is the number of sub-populations
  # "nb_neuron" is a vector of length nb_pop. Coordinate k is the number of neurons in popul…
  # "intensity_function" is a list of length nb_pop. Each coordinate is the function that ma…
  # c, nu and eta refer to the paper Ditlevsen and Locherbach (2016). CAREFUL : eta correspo…
  # "c_vec" is a vector of length nb_pop. Coordinate k is the multiplicative constant for th…
  # "nu_vec" is a vector of length nb_pop. Coordinate k is the time constant in the exponent…
  # "eta_vec" is a vector of length nb_pop. Coordinate k is the memory order for the interac…
  # "X_init" is a vector of length eta_1 x eta_2 x ... x eta_n. It gives the initial conditi…
  # The simulation stops when the dominating processes have produced "nb_points" spike.
  # Output : a list of length 3
  # Let N denote the total number of spikes simulated
  # $spike_train is a vector of length N with all the spiking times
  # $type is a vector of length N with the indices of the spiking neurons
  # $intensity is a matrix "nb_pop" x N with the intensity values for each population at eac…
  t = 0
  X = X_init
  lambda_bound = intensity_bound(X, nb_pop, eta_vec, intensity_function)
  points = rep(0, nb_points)
  type = rep(0, nb_points)
  intensity = matrix(0, nrow = nb_pop, ncol = nb_points)
  X_in_time = matrix(0, nrow = length(X_init), ncol = nb_points)
  A = linear_ODE_matrix(nu_vec, eta_vec)
  for (i in 1:nb_points)
  {
    dominant_ISI = rexp(nb_pop, nb_neuron*lambda_bound) # For each population we have the ne…
    spiking_pop = which.min(dominant_ISI)
    possible_ISI = dominant_ISI[spiking_pop]
    if ( possible_ISI==0 ){break} # To get out of a possible blow-up
    t = t + possible_ISI
    new_X = expAtv(A, X, possible_ISI)$eAtv # Update of X via the flow of the ODE
    # new_X = X + possible_ISI*(A%*%X)        Take only the linear term in the exponential (…
    new_lambda = intensity(X, nb_pop, eta_vec, intensity_function)
    if ( new_lambda[spiking_pop]==Inf ){break} # To get out of a possible blow-up
    test1 = rbinom(1,1,prob = new_lambda[spiking_pop]/lambda_bound[spiking_pop]) # If test =…
    if (test1)
    {
      neuron_number = sample(1:(nb_neuron[spiking_pop]), 1) # the spiking neuron is uniform …
      if (spiking_pop == 1) {influenced_pop = nb_pop} # Get the index of the population infl…
      else {influenced_pop = spiking_pop - 1}
      X = new_X
      X[ sum(eta_vec[0:influenced_pop]) ] = X[ sum(eta_vec[0:influenced_pop]) ] + c_vec[infl…
      lambda_bound = intensity_bound(X, nb_pop, eta_vec, intensity_function)
      # Store the values
      points[i] = t
      type[i] = sum(nb_neuron[0:(spiking_pop-1)]) + neuron_number # the index of the spiking…
      intensity[,i] = new_lambda
      X_in_time[,i] = X
    }
    else
    {
      X = new_X
      lambda_bound = intensity_bound(X, nb_pop, eta_vec, intensity_function)
    }
  }
  good_index=which(points!=0)
  return(list(spike_train = points[good_index], type = type[good_index], intensity = intensi…
}
```

```
intensity <- function(X, nb_pop, eta_vec, intensity_function)
  # Output : a vector of length "nb_pop". Coordinate k is the upper-bound of the intensity i…
  # If "potential" is positive then the bound is f("potential")
  # If "potential" is negative then the bound is f(0)
{
  value = rep(0, nb_pop)
  for (k in 1:nb_pop)
  {
    potential = X[sum(eta_vec[0:(k-1)]) + 1]
    value[k] = intensity_function[[k]](potential)
  }
  return(value)
}


intensity_bound <- function(X, nb_pop, eta_vec, intensity_function)
  # Output : a vector of length "nb_pop". Coordinate k is the upper-bound of the intensity i…
  # If "potential" is positive then the bound is f("potential")
  # If "potential" is negative then the bound is f(0)
{
  bound = rep(0, nb_pop)
  for (k in 1:nb_pop)
  {
    potential = X[sum(eta_vec[0:(k-1)]) + 1]
    bound[k] = intensity_function[[k]]( max(potential, 0) )
  }
  return(bound)
}


linear_ODE_matrix <- function(nu_vec, eta_vec)
  # Constructs the matrix of the linear ODE from nu_vec and eta_vec
{
  dimension = sum(eta_vec)
  A = matrix(0, nrow = dimension, ncol = dimension)
  nb_pop = length(nu_vec)
  D = c()
  for (k in 1:nb_pop)
  {
    D = c(D, rep(-nu_vec[k], eta_vec[k]))
  }
  diag(A) <- D
  upper = c()
  for (k in 1:(nb_pop-1))
  {
    upper = c(upper, rep(1, eta_vec[k]-1), 0)
  }
  upper = c(upper, rep(1, eta_vec[nb_pop]-1))
  indx <- 1:(dimension-1)
  A[cbind(indx,indx+1)] <- upper
  return(A)
}


### USAGE ###
## To reproduce Figure 1 in Eva and Susane's paper
nb_pop = 2
nb_neuron = rep(100, 2)
f1 <- function(x)
{
  if (x<log(20)) {return(10*exp(x))}
  else {return( 400/(1+400*exp(-2*x)) )}
```

```
}
f2 <- function(x)
{
  if (x<log(20)) {return(exp(x))}
  else {return( 40/(1+400*exp(-2*x)) )}
}
intensity_function = list(f1,f2)
c_vec = c(-1, 1)
nu_vec = rep(1, 2)
eta_vec = c(3, 2) + 1   # In comparison with the paper of Eva and Susanne, we take eta+1 ins…
X_init = rep(0, sum(eta_vec))

test = simul_Hawkes_Erlang(nb_pop, nb_neuron, intensity_function, c_vec, nu_vec, eta_vec, X_…

plot(test$spike_train, test$intensity[1,], type='l', ylim = c(0,40), col='blue')
lines(test$spike_train, test$intensity[2,],col='black')

plot(test$spike_train, test$X[1,], type = 'n', ylim = c(-40,10))
for (i in 2:4)
{
  lines(test$spike_train, test$X[i,], col='grey')
}
lines(test$spike_train, test$X[1,])
for (i in 6:7)
{
  lines(test$spike_train, test$X[i,], col='grey')
}
lines(test$spike_train, test$X[5,])


### My experiments
plot(test$X[1,], test$X[5,], type = "l")
```