Reliability of a system subjected to a cumulative shock model with a change point

Dheeraj Goyal and Arnab Hazra

Code for Figure 2(a):

```r
library(mcompanion)
library(matrixcalc)
library(expm)
```

```
## Loading required package: Matrix

##
## Attaching package: 'expm'

## The following object is masked from 'package:Matrix':
##
##     expm
```

```r
library(pracma)
```

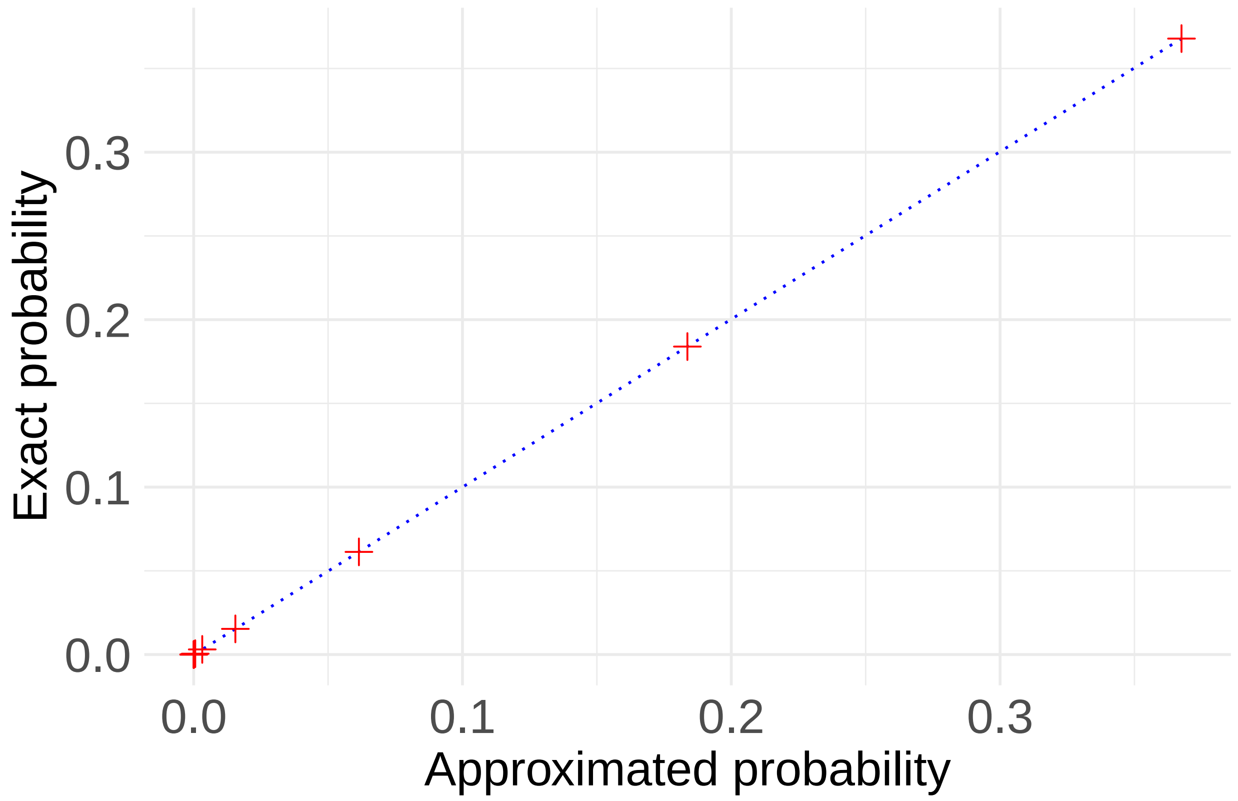
```
##
## Attaching package: 'pracma'

## The following objects are masked from 'package:expm':
##
##     expm, logm, sqrtm

## The following objects are masked from 'package:Matrix':
##
##     expm, lu, tril, triu
```

```r
library(doParallel)
```

```
## Loading required package: foreach

## Loading required package: iterators

## Loading required package: parallel
```

```r
a <- function(m)
{

  v = c()

  v[1] = 1

  for (i in 2:m) {

    v[i] = 0
  }

  return(t(v))

}

e <- function(m)
{

  v = c()

  v[1] = 1

  for (i in 2:m) {

    v[i] = 1

  }

  return(v)

}

G <- function(m)
{
  library(mcompanion)
  A  = m*(Jordan_matrix(m,m) - m*diag(m))-m*diag(m)
  return(A)
}

hpppmf <- function(t,l,n)
{

  x = exp((-l)*t) * ((l*t)^(n)/factorial(n))

  return(x)
}


pphppmf <- function(m,t,n,l)
{

  library(matrixcalc)

  x = l*t*diag(m) - G(m)

  xin = matrix.inverse(x)

  y = matrix.power(xin, n+1)

  z = (-1)*((l*t)^(n))*a(m)%*%y%*%G(m)%*%e(m)

  w = z[1]
```

```r
  return(w)

}

approxhpp <- function(t,l,n)
{
  z = 1
  m = 2

  while(z > 0.0000008)
  {

    x = pphppmf(m,t,n,l)

    z = abs(pphppmf(m+1,t,n,l) - pphppmf(m,t,n,l))

    m = m+1
  }

  return(x)

}


library(ggplot2)

# Define n, t, and l (lambda)
t <- 1
l <- 1

# Define the range for n
n_values <- seq(1, 10, by = 1)

# Calculate hpppmf and approxhpp for each n
data1 <- data.frame(
  n = n_values,
  hpppmf = sapply(n_values, function(n) hpppmf(t, l, n)),
  approxhpp = sapply(n_values, function(n) approxhpp(t, l, n))
)


p = ggplot(data1, aes(x = approxhpp, y = hpppmf)) +
  geom_line(linetype = "dotted", color = "blue") +
  geom_point(shape = 3, size = 3, color = "red") +
  labs(
    title = "",
    x = "Approximated probability",
    y = "Exact probability"
  ) +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 18),
    axis.title.y = element_text(size = 18),
    axis.text.x = element_text(size = 18),
    axis.text.y = element_text(size = 18)
  )

print(p)
```

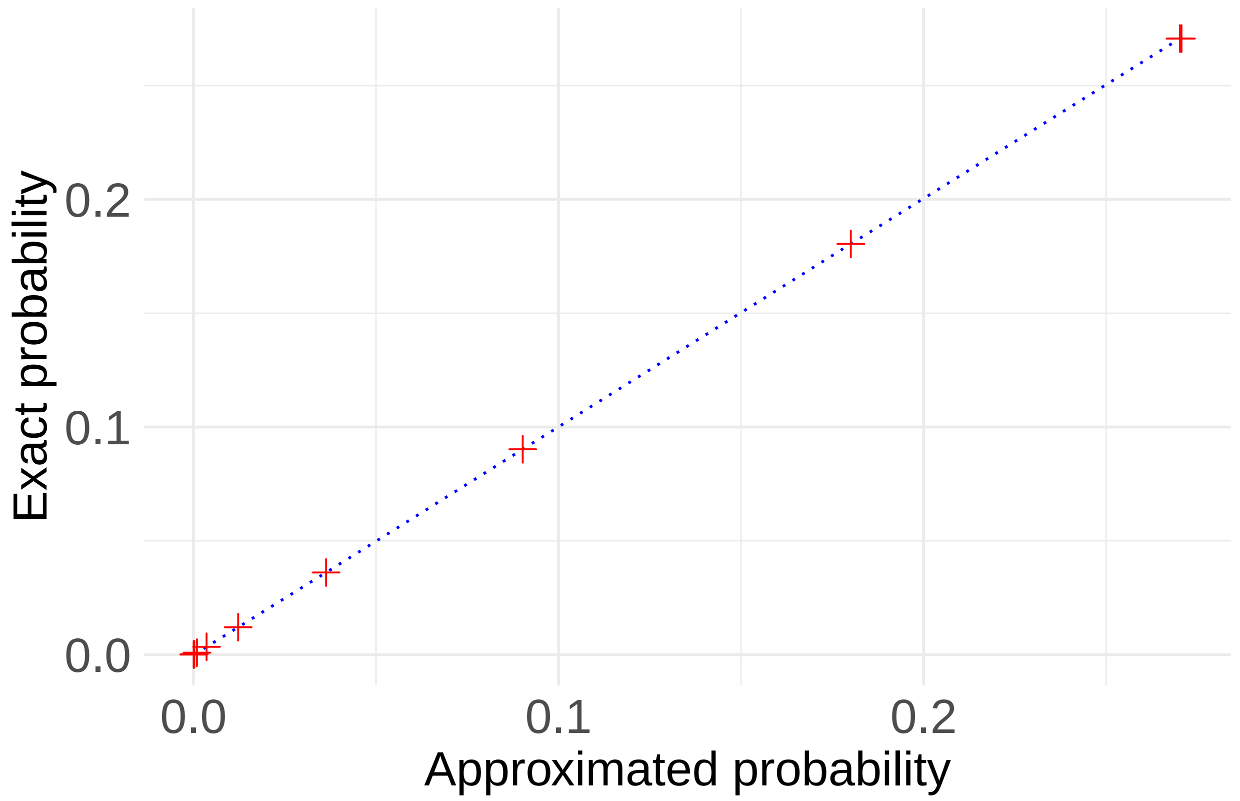


Code for Figure 2(b):

```
a <- function(m)
{

  v = c()

  v[1] = 1

  for (i in 2:m) {

    v[i] = 0

  }

  return(t(v))
```

```
}

e <- function(m)
{

  v = c()

  v[1] = 1

  for (i in 2:m) {

    v[i] = 1

  }

  return(v)

}

G <- function(m)
{
  library(mcompanion)
  A  = m*(Jordan_matrix(m,m) - m*diag(m))-m*diag(m)
  return(A)
}


hpppmf <- function(t,l,n)
{

  x = exp((-l)*t) * ((l*t)^(n)/factorial(n))

  return(x)
}


pphppmf <- function(m,t,n,l)
{

  library(matrixcalc)

  x = l*t*diag(m) - G(m)

  xin = matrix.inverse(x)

  y = matrix.power(xin, n+1)

  z = (-1)*((l*t)^(n))*a(m)%*%y%*%G(m)%*%e(m)

  w = z[1]

  return(w)
```

```r
}

approxhpp <- function(t,l,n)
{
  z = 1
  m = 2

  while(z > 0.0000008)
  {

    x = pphppmf(m,t,n,l)

    z = abs(pphppmf(m+1,t,n,l) - pphppmf(m,t,n,l))

    m = m+1
  }

  return(x)

}


library(ggplot2)

# Define n, t, and l (lambda)
t <- 2
l <- 1

# Define the range for n
n_values <- seq(1, 10, by = 1)

# Calculate hpppmf and approxhpp for each n
data1 <- data.frame(
  n = n_values,
  hpppmf = sapply(n_values, function(n) hpppmf(t, l, n)),
  approxhpp = sapply(n_values, function(n) approxhpp(t, l, n))
)




p = ggplot(data1, aes(x = approxhpp, y = hpppmf)) +
  geom_line(linetype = "dotted", color = "blue") +
  geom_point(shape = 3, size = 3, color = "red") +
  labs(
    title = "",
    x = "Approximated probability",
    y = "Exact probability"
  ) +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 18),
    axis.title.y = element_text(size = 18),
    axis.text.x = element_text(size = 18),
    axis.text.y = element_text(size = 18)
  )

print(p)
```



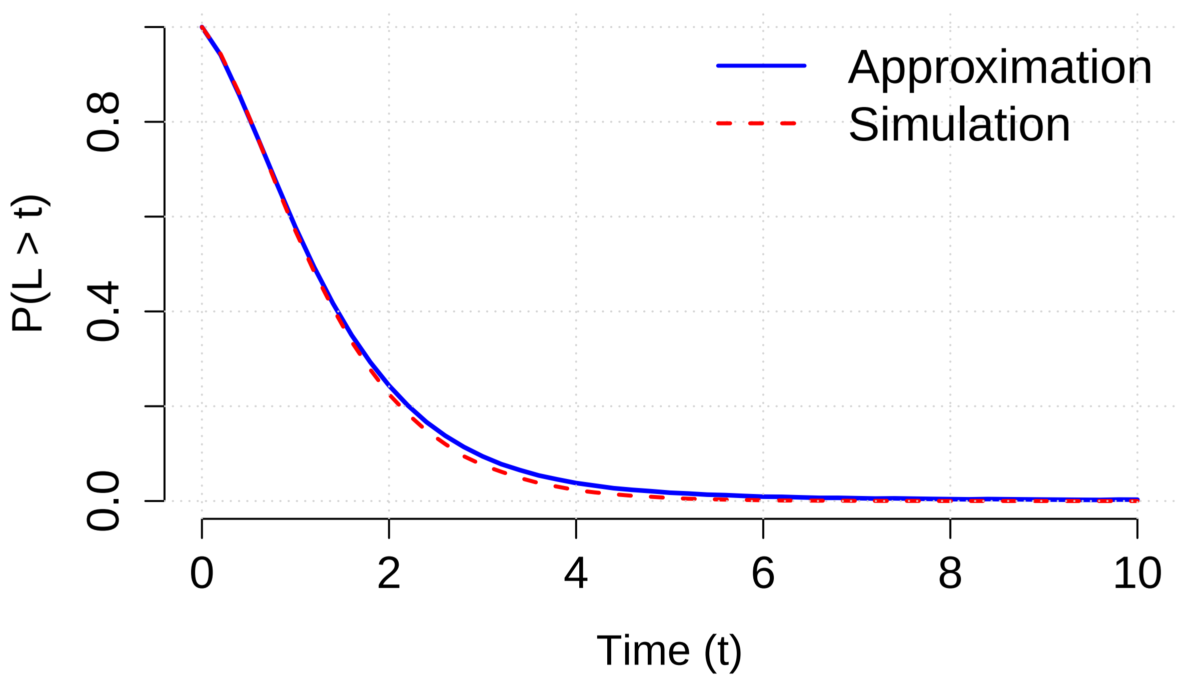
Code for Figure 3(a):

```
#Figure 3(a)

library(expm)
library(pracma)
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}
```

```r
alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}
```

```r
approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values <- parSapply(cl, t_values, approxrel)

stopCluster(cl)

rel <- function(t, H, lambda = 2, n = 1e5, x1 = 1/3, x2 = 1/2){

  set.seed(1)

  poisson_values <- rpois(n, lambda * t)
  change_value <- rexp(n, 1/x1)

  samps <- sapply(1:n, function(i){
    ifelse(t <= change_value[i],
           sum(rexp(poisson_values[i], 1 / x1)),
           sum(rexp(rpois(1, lambda * change_value[i]), 1 / x1)) +
             sum(rexp(rpois(1, lambda * (t - change_value[i])), 1 / x2)))})

  mean(samps < H)}


rel.grid <- sapply(seq(0, 10, 0.2), rel, H = 0.8)
```

```r
plot(t_values, rel_values, type = "l", col = "blue", lty = 1, lwd = 2.3,
     xlab = "Time (t)", ylab = "P(L > t)", main = "",
     cex.main = 1.5, cex.lab = 1.3, cex.axis = 1.4, bty = "n",
     col.lab = "black", col.axis = "black", font.main = 2)
lines(seq(0, 10, 0.2), rel.grid, col = "red", lwd = 2, lty = 2.3)
grid(col = "lightgray", lty = "dotted")
legend("topright", legend = c("Approximation", "Simulation"), col = c("blue", "red"),
       lty = c(1, 2), lwd = 2, bty = "n", cex = 1.5)
```



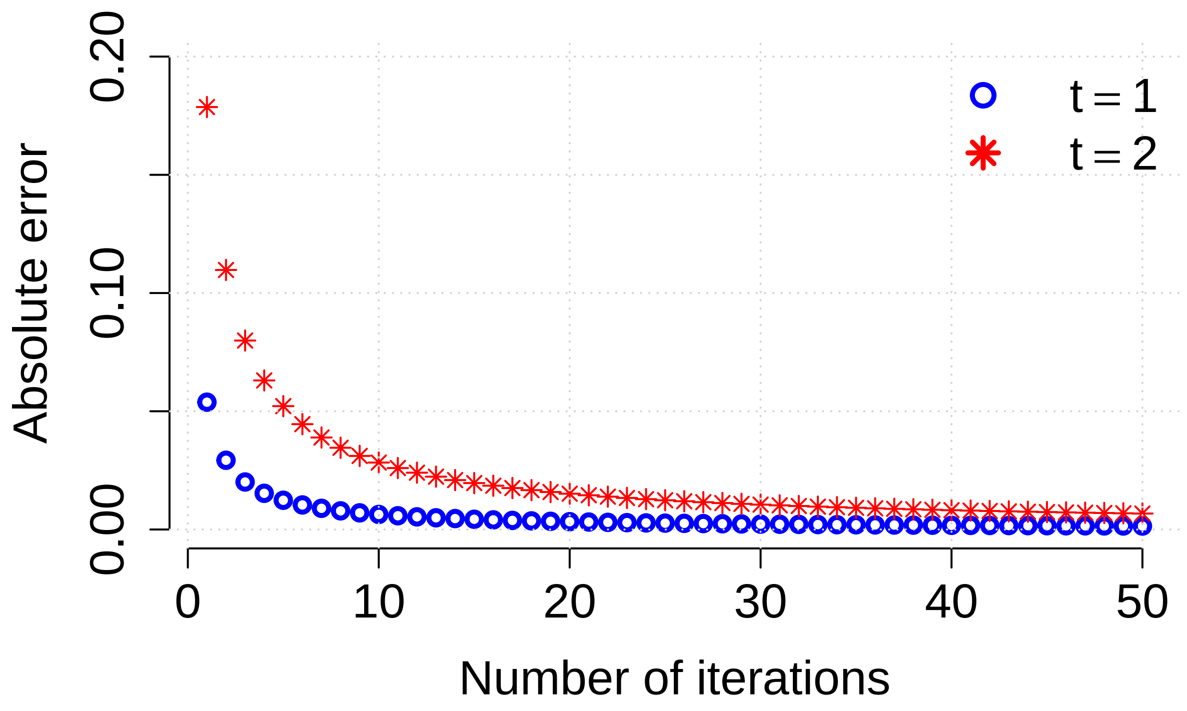
Code for Figure 3(b):

```r
#Figure 3(b)

library(expm)    # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result
```

```r
  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t,M){
  rel(t, M)}



M_values <- 1:50

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "M_values", "approxrel"))

rel_values_1 <- parSapply(cl, M_values, approxrel, t = 1)

stopCluster(cl)





rel <- function(t, H, lambda = 2, n = 1e5, x1 = 1/3, x2 = 1/2){

  set.seed(1)

  poisson_values <- rpois(n, lambda * t)
  change_value <- rexp(n, 1/x1)

  samps <- sapply(1:n, function(i){
    ifelse(t <= change_value[i],
           sum(rexp(poisson_values[i], 1 / x1)),
           sum(rexp(rpois(1, lambda * change_value[i]), 1 / x1)) +
             sum(rexp(rpois(1, lambda * (t - change_value[i])), 1 / x2)))})

  mean(samps < H)}

truerel_1 <- rel(1, H = 0.8)


err_rel_1 <- rel_values_1 - truerel_1
```

```r
library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}
  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t,M){
  rel(t, M)}


M_values <- 1:50

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "M_values", "approxrel"))

rel_values_2 <- parSapply(cl, M_values, approxrel, t = 2)

stopCluster(cl)


rel <- function(t, H, lambda = 2, n = 1e5, x1 = 1/3, x2 = 1/2){

  set.seed(1)

  poisson_values <- rpois(n, lambda * t)
  change_value <- rexp(n, 1/x1)

  samps <- sapply(1:n, function(i){
    ifelse(t <= change_value[i],
           sum(rexp(poisson_values[i], 1 / x1)),
           sum(rexp(rpois(1, lambda * change_value[i]), 1 / x1)) +
             sum(rexp(rpois(1, lambda * (t - change_value[i])), 1 / x2)))})

  mean(samps < H)}
```

```r
truerel_2 <- rel(2, H = 0.8)


err_rel_2 <- rel_values_2 - truerel_2


plot(M_values, err_rel_1, col = "blue", lty = 1, lwd = 2.5,
     xlab = "Number of iterations", ylab = "Absolute error", main = "",
     ylim = c(0, 0.20), cex.main = 1.5, cex.lab = 1.5, cex.axis = 1.5, bty = "n",
     col.lab = "black", col.axis = "black", font.main = 2, pch = 1)


points(M_values, err_rel_2, col = "red", pch = 8)


grid(col = "lightgray", lty = "dotted")
legend("topright", legend = c(expression(t == 1), expression(t == 2)),
       col = c("blue", "red"), pch = c(1, 8),
       lty = c(NA, NA),
       lwd = 2.5, bty = "n", cex = 1.5)
```

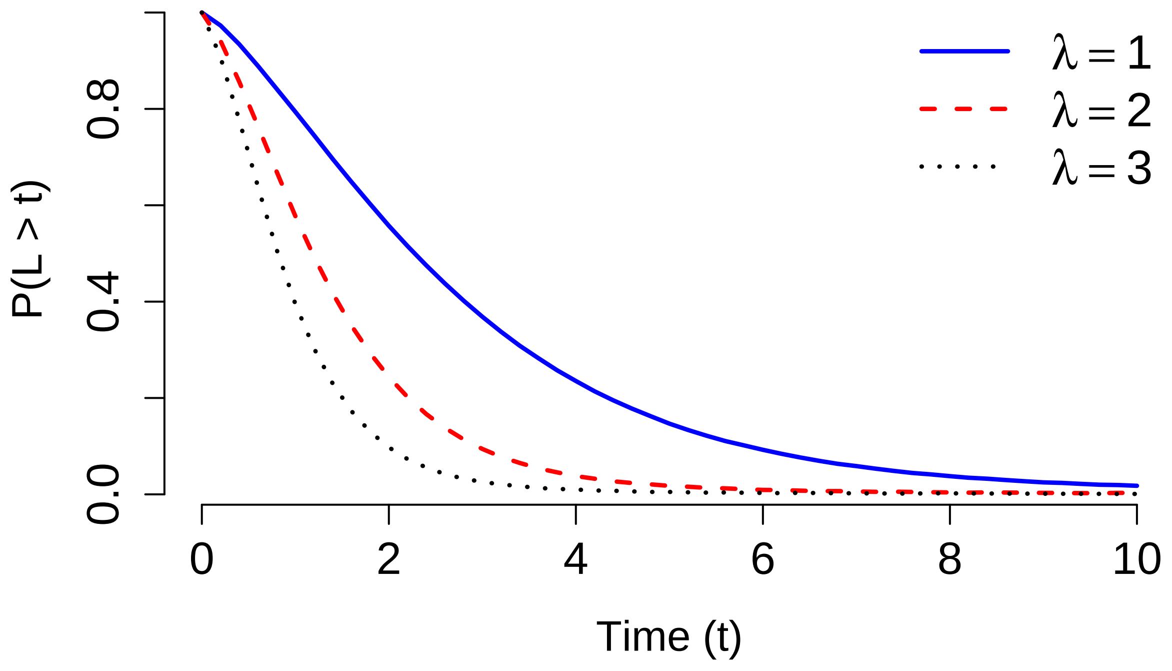


Code for Figure 4(a):

```r
#1

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}


rel <- function(t, m, H = 0.8, lambda.x = 3, l = 1,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}
  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values1 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)
```

```r
#2

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}
  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values2 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


#3
```

```r
library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 3, l = 3,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}
  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values3 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


plot(t_values, rel_values1, type = "l", col = "blue", lty = 1, lwd = 2.2,
     xlab = "Time (t)", ylab = "P(L > t)", main = "",
     cex.main = 1.5, cex.lab = 1.3, cex.axis = 1.4, bty = "n",
     col.lab = "black", col.axis = "black", font.main = 2)
lines(seq(0, 10, 0.2), rel_values2, col = "red", lwd = 2.2, lty = 2)
lines(seq(0, 10, 0.2), rel_values3, col = "black", lwd = 2.2, lty = 3)
#grid(col = "lightgray", lty = "dotted")
legend("topright", legend = c(expression(lambda == 1),
                              expression(lambda == 2),
                              expression(lambda == 3)),
       col = c("blue", "red", "black"), lty = c(1, 2, 3), lwd = 2.2, bty = "n", cex = 1.5)
```



Code for Figure 4(b):

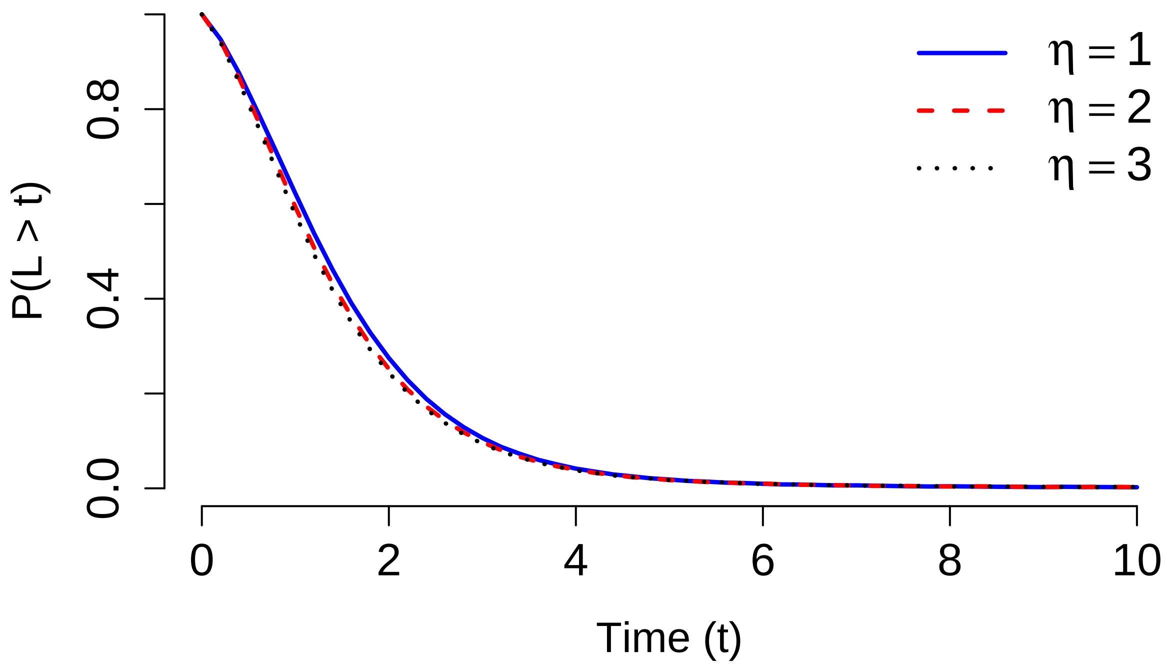
```
#1

library(expm)    # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
```

```r
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}


rel <- function(t, m, H = 0.8, lambda.x = 1, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result
  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values1 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


#2

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 2, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result
  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values2 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


#3

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}
```

```r
alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}
```

```r
approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values3 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)
```

```r
plot(t_values, rel_values1, type = "l", col = "blue", lty = 1, lwd = 2.2,
     xlab = "Time (t)", ylab = "P(L > t)", main = "",
     cex.main = 1.5, cex.lab = 1.3, cex.axis = 1.4, bty = "n",
     col.lab = "black", col.axis = "black", font.main = 2)
lines(seq(0, 10, 0.2), rel_values2, col = "red", lwd = 2.2, lty = 2)
lines(seq(0, 10, 0.2), rel_values3, col = "black", lwd = 2.2, lty = 3)
#grid(col = "lightgray", lty = "dotted")
legend("topright", legend = c(expression(eta == 1),
                              expression(eta == 2),
                              expression(eta == 3)),
       col = c("blue", "red", "black"), lty = c(1, 2, 3), lwd = 2.2, bty = "n", cex = 1.5)
```

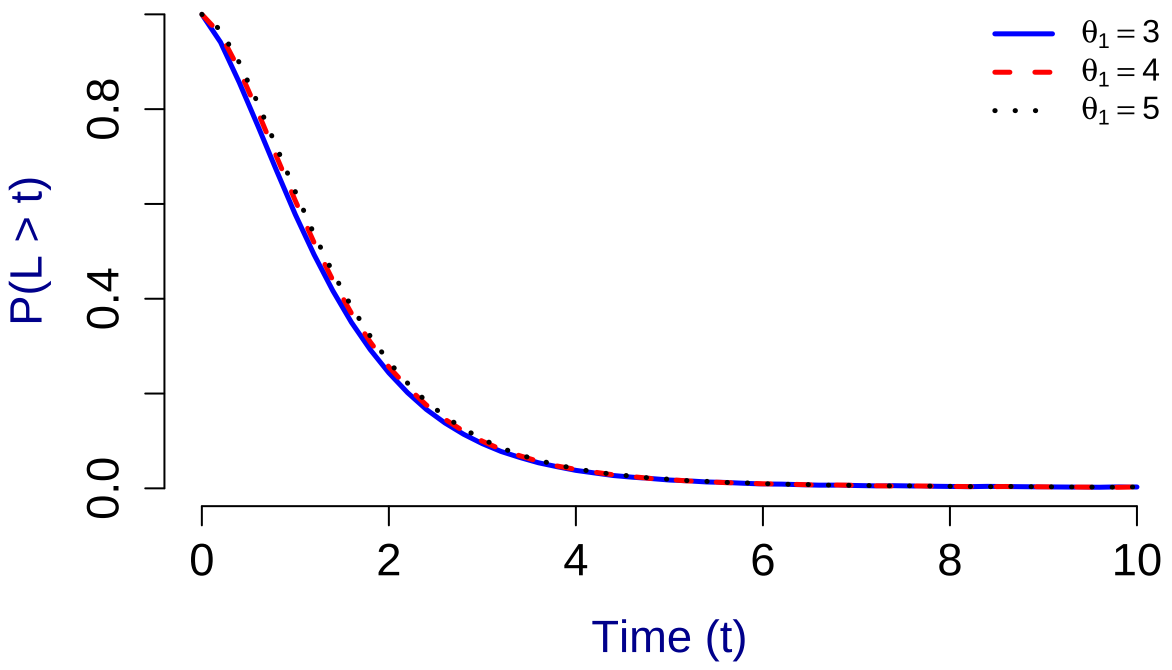


Code for Figure 4(c):

```
#1

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}
```

```r
rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values1 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


#2

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}
```

```r
rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-4, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values2 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


#3

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-5, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)
  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values3 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)

plot(t_values, rel_values1, type = "l", col = "blue", lty = 1, lwd = 2.5,
     xlab = "Time (t)", ylab = "P(L > t)", main = "",
     cex.main = 1.5, cex.lab = 1.4, cex.axis = 1.4, bty = "n",
     col.lab = "darkblue", col.axis = "black", font.main = 2)
lines(seq(0, 10, 0.2), rel_values2, col = "red", lwd = 2.5, lty = 2)
lines(seq(0, 10, 0.2), rel_values3, col = "black", lwd = 2.5, lty = 3)
legend("topright", legend = c(expression(theta[1] == 3),
                              expression(theta[1] == 4),
                              expression(theta[1] == 5)),
       col = c("blue", "red", "black"), lty = c(1, 2, 3), lwd = 2.5, bty = "n")
```



Code for Figure 4(d):

```
#1

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
```

```r
  out}


rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2.5, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2
  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values1 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


#2

library(expm)    # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}
```

```r
rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-2, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values2 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


#3

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 3, l = 2,
```

```r
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-3, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-1.5, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)
  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 10, by = 0.2)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values3 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


plot(t_values, rel_values1, type = "l", col = "blue", lty = 1, lwd = 2.5,
     xlab = "Time (t)", ylab = "P(L > t)", main = "",
     cex.main = 1.5, cex.lab = 1.4, cex.axis = 1.4, bty = "n",
     col.lab = "black", col.axis = "black", font.main = 2)
lines(seq(0, 10, 0.2), rel_values2, col = "red", lwd = 2.5, lty = 2)
lines(seq(0, 10, 0.2), rel_values3, col = "black", lwd = 2.5, lty = 3)
#grid(col = "lightgray", lty = "dotted")
legend("topright", legend = c(expression(theta[2] == 1.5),
                              expression(theta[2] == 2.0),
                              expression(theta[2] == 2.5)),
       col = c("blue", "red", "black"), lty = c(3, 2, 1), lwd = 2.5, bty = "n")
```

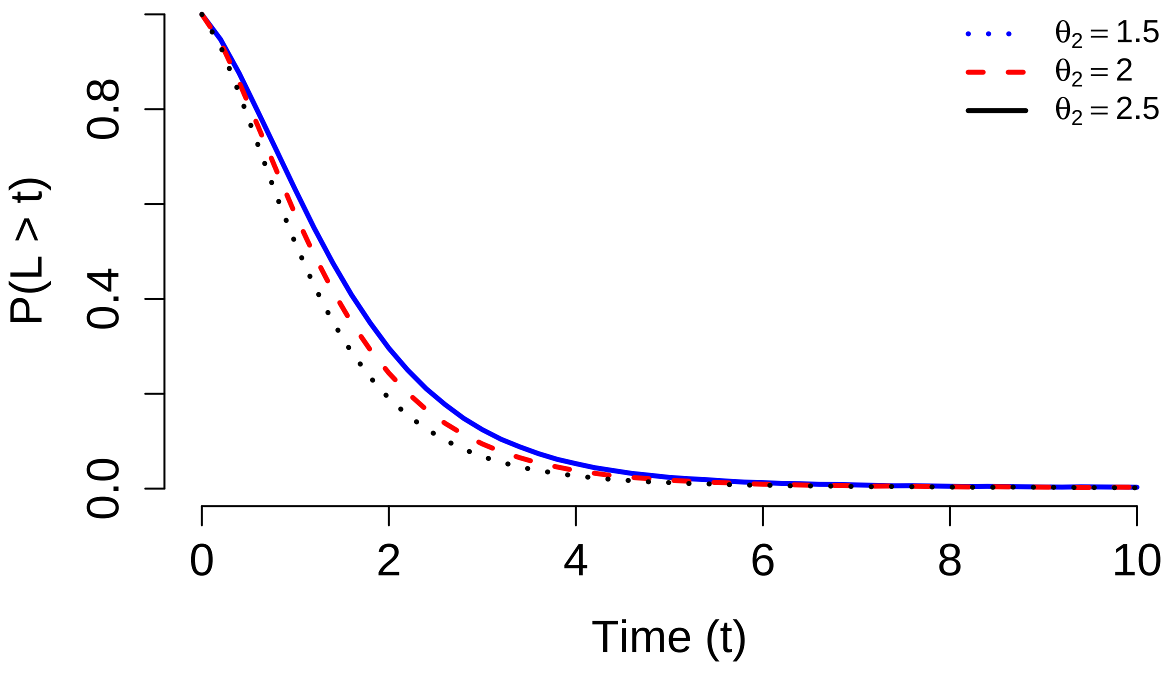
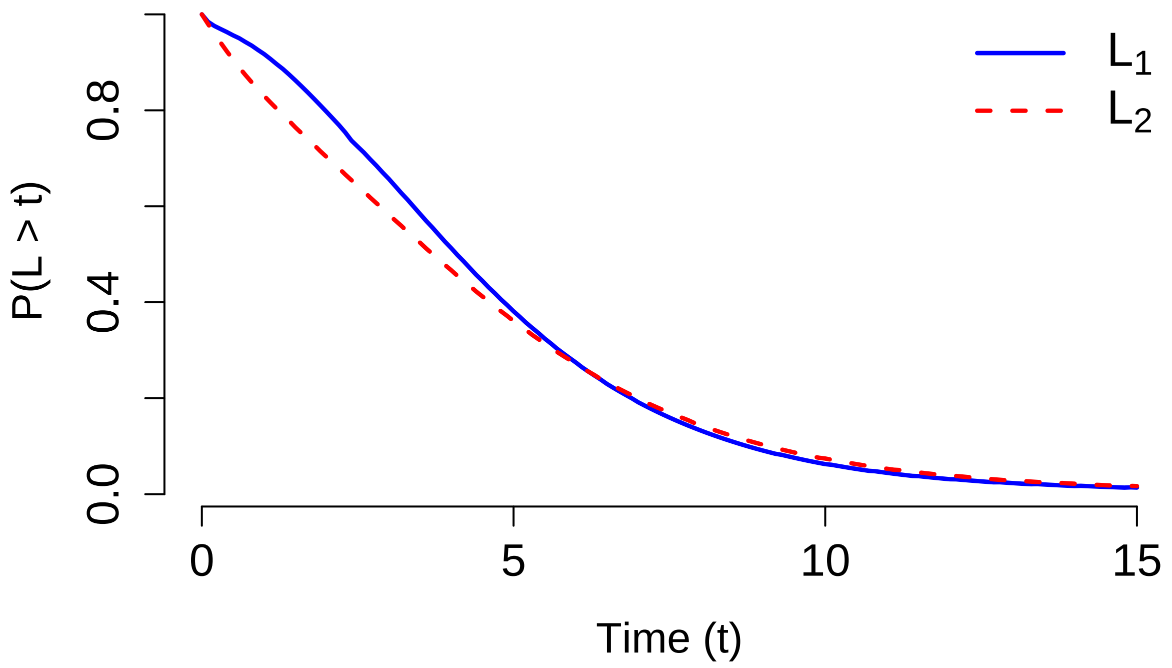


Figure 5:

```
#counter example



#1

library(expm)    # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 10, l = 1.1,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-2, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-5, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2

  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 15, by = 0.1)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values2 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


#2

library(expm)   # For matrix exponentiation
library(pracma) # For numerical integration
library(matrixcalc)

a <- function(m){matrix(c(1, rep(0, m - 1)), 1, m)}
e <- function(m){rep(1, m)} # Vector of ones
I <- function(m){diag(m)} # Identity matrix of size m

A <- function(m){
  out <- -m * diag(m)
  out[cbind(1:(m - 1), 2:m)] <- m
  out}

alpha_times_matinv <- function(m, t, l){
  matrix((m / (m + l * t))^(0:(m - 1)) / (1 + m / (l * t)), 1, m)}

matinv <- function(m, t, l){
  out <- (1 + m / (l * t))^(-1) * (m / (m + l * t))^abs(outer(1:m, 1:m, "-"))
  out[lower.tri(out)] <- 0
  out}

rel <- function(t, m, H = 0.8, lambda.x = 1, l = 1,
                size1 = 1, g1 = matrix(1, 1, 1), G1 = matrix(-2, 1, 1),
                size2 = 1, g2 = matrix(1, 1, 1), G2 = matrix(-5, 1, 1)){

  r1 <- kronecker(g1, alpha_times_matinv(m, t, l))
  R1 <- kronecker(G1, I(m)) +
    kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, t, l))

  r2 <- function(x){kronecker(g2, alpha_times_matinv(m, t - x, l))}

  R2 <- function(x){kronecker(G2, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G2)) %*% g2, matinv(m, t - x, l))}

  r3 <- function(x){kronecker(g1, alpha_times_matinv(m, x, l))}

  R3 <- function(x){kronecker(G1, I(m)) +
      kronecker(-1 * rowSums(as.matrix(G1)) %*% g1, matinv(m, x, l))}

  P_X_greater_than_t <- function(t){exp(-lambda.x * t)}

  f_X <- function(x){lambda.x * exp(-lambda.x * x)}

  q2 <- function(x){1 - sum(r3(x))}

  s <- function(x){cbind(r3(x), q2(x) * r2(x))}

  q3 <- function(x){-1 * rowSums(R3(x)) %*% r2(x)}

  zermat <- function(x){array(0, dim = dim(R3(x)))}

  S <- function(x){rbind(cbind(R3(x), q3(x)), cbind(zermat(x), R2(x)))}

  term_1 <- c(r1 %*% rowSums(expm::expm(R1 * H))) * P_X_greater_than_t(t)

  integrand <- function(x){c(s(x) %*% rowSums(expm::expm(S(x)*H))) * f_X(x)}

  integrand_vec <- Vectorize(integrand)

  result <- integral(integrand_vec, 0, t)

  updated_result <- term_1 + result

  sytem_rel <- 1 - updated_result

  sytem_rel}


approxrel <- function(t){
  z <- 1
  m <- 2
  x.old <- rel(t, m)
  x.new <- rel(t, m + 1)
  z <- abs(x.new - x.old)

  while(z > 1e-3){
    x.old <- x.new
    m <- m + 1
    x.new <- rel(t, m + 1)
    z <- abs(x.new - x.old)
  }
  x.old}

t_values <- seq(0, 15, by = 0.1)

library(parallel)
library(doParallel)

ncores <- 4

cl <- makeCluster(ncores)
registerDoParallel(cl)

clusterExport(cl, varlist = c("rel", "matinv", "alpha_times_matinv", "expm",
                              "integral","A", "a", "e", "I",
                              "t_values", "approxrel"))

rel_values3 <- parSapply(cl, t_values, approxrel)

stopCluster(cl)


plot(t_values, rel_values2, type = "l", col = "blue", lty = 1, lwd = 2.2,
     xlab = "Time (t)", ylab = "P(L > t)", main = "",
     cex.main = 1.5, cex.lab = 1.3, cex.axis = 1.4, bty = "n",
     col.lab = "black", col.axis = "black", font.main = 2)
lines(seq(0, 15, 0.1), rel_values3, col = "red", lwd = 2.2, lty = 2)
legend("topright", legend = c(expression(L[1]),
                              expression(L[2])),
       col = c("blue", "red"), lty = c(1, 2), lwd = 2.2, bty = "n", cex = 1.5)
```



P(L > t)
0.0
0.4
0.8
0
5
10
15
Time (t)
L_1
L_2