Functions used by the saddlepoint to compute a mixture of binomial densities

```
M0 <- function(t, p, n)
  {
    #MGF for binomial distribution with parameters n and p

    ((1-p) + p*exp(t))^n
  }


M1 <- function(t, p, n)
  {
    #First derivative of MGF for binomial distribution with
    #parameters n and p

    n*((1-p) + p*exp(t))^(n-1) * p*exp(t)
  }


M2 <- function(t, p, n)
  {
    #Second derivative of MGF for binomial distribution with
    #parameters n and p

    M1(t, p, n) + n*(n-1)*((1-p) + p*exp(t))^(n-2) * (p*exp(t))^2
  }


m0 <- function(t, n)
{
   #MGF for sum of cluster of binary observations
   #Uses Romberg integration with 10 iterations as an approximation
   #fp is the mixing distribution that must be externally defined
   maxiter <- 10
   a <- 0; b <- 1; h <- b - a
   R <- matrix(0, nrow = maxiter, ncol = maxiter)

   R[1, 1] <- (h * (M0(t, a, n) * fp(a) + M0(t, b, n) * fp(b)))/2

   for(i in 2:maxiter)
      {
         e1 <- M0(t, a + (1:2^(i - 2) - 0.5) * h, n)
         e2 <- fp(a + (1:2^(i - 2) - 0.5) * h)
         R[i, 1] <- 0.5 * (R[i - 1, 1] + h * sum(e1 * e2))
         for(j in 2:i)
            {
               R[i, j] <- (4^(j - 1) * R[i, j - 1] -
               R[i - 1, j - 1])/(4^(j - 1) - 1)
            }
         h <- h/2
      }
    as.numeric(R[maxiter, maxiter])
}
```

```
m1 <- function(t, n)
  {
   #First derivative of MGF for sum of cluster of binary observations
   #Uses Romberg integration with 10 iterations as an approximation
   #fp is the mixing distribution that must be externally defined
   maxiter <- 10
   a <- 0; b <- 1; h <- b - a
   R <- matrix(0, nrow = maxiter, ncol = maxiter)

   R[1, 1] <- (h * (M1(t, a, n) * fp(a) + M1(t, b, n) * fp(b)))/2

   for(i in 2:maxiter)
      {
         e1 <- M1(t, a + (1:2^(i - 2) - 0.5) * h, n)
         e2 <- fp(a + (1:2^(i - 2) - 0.5) * h)
         R[i, 1] <- 0.5 * (R[i - 1, 1] + h * sum(e1 * e2))
         for(j in 2:i)
            {
               R[i, j] <- (4^(j - 1) * R[i, j - 1] -
               R[i - 1, j - 1])/(4^(j - 1) - 1)
            }
         h <- h/2
      }
   as.numeric(R[maxiter, maxiter])
}


m2 <- function(t, n)
  {
   #Second derivative of MGF for sum of cluster of binary observations
   #Uses Romberg integration with 10 iterations as an approximation
   #fp is the mixing distribution that must be externally defined
   maxiter <- 10
   a <- 0; b <- 1; h <- b - a
   R <- matrix(0, nrow = maxiter, ncol = maxiter)

   R[1, 1] <- (h * (M2(t, a, n) * fp(a) + M2(t, b, n) * fp(b)))/2

   for(i in 2:maxiter)
      {
         e1 <- M2(t, a + (1:2^(i - 2) - 0.5) * h, n)
         e2 <- fp(a + (1:2^(i - 2) - 0.5) * h)
         R[i, 1] <- 0.5 * (R[i - 1, 1] + h * sum(e1 * e2))
         for(j in 2:i)
            {
               R[i, j] <- (4^(j - 1) * R[i, j - 1] -
               R[i - 1, j - 1])/(4^(j - 1) - 1)
            }
         h <- h/2
      }
   as.numeric(R[maxiter, maxiter])
}
```

```
K0 <- function(t, n)
  {
    #CGF for sum of cluster of binary observations
    #Uses Romberg integration with 10 iterations as an approximation

    log(m0(t, n))
  }


K1 <- function(t, n)
  {
    #First derivative of CGF for sum of cluster of binary observations
    #Uses Romberg integration with 10 iterations as an approximation

    m1(t, n)/m0(t, n)
  }


K2 <- function(t, n)
  {
    #Second derivative of CGF for sum of cluster of binary observations
    #Uses Romberg integration with 10 iterations as an approximation

    m2(t, n)/m0(t, n) - (m1(t, n)/m0(t, n))^2
  }
```

Functions used to compute the saddlepoint approximation

```
sp.eq <- function(t)
  {
    #The saddlepoint equation; sp.f finds the root of this function
    K0(t, n.sp) - t*y
  }


sp.f <- function(x, n.sp)
  {
    #Computes the saddlepoint density at a specific point x
    y <<- x+0.99
    n.sp <<- n.sp
    sp <- nlmin(sp.eq, 0.1)$x
    y <<- x
    ff <- exp((K0(sp, n.sp)-x*sp))/sqrt(2*pi*K2(sp, n.sp))
    rm(n.sp, y)
    as.numeric(ff)
  }


get.sp.distn <- function(nn)
  {
    #Computes the entire saddlepoint distribution for all
    #values from 0 to nn and then normalizes
    yy <- 0:nn
    fy <- sapply(yy, sp.f, n.sp=nn)
    fy[is.na(fy)] <- 0
    fy/sum(fy)
  }
```

Code used to compute power estimates shown in Section 4.2

```
Nsamp <- 20000
alpha <- 0.05

#User needs to supply:
#1) cluster sizes
    n.ctrl <- c(50, 57, 64, 70, 75, 75, 88, 97, 98, 160)
    n.trt <- c(40, 73, 107, 120, 122, 124, 126, 135, 182, 190)
#2) null hypothesis value
    p.ctrl <- 0.10
#3) alternative hypothesis value
    p.trt <- 0.05
#4) intra-cluster correlation for average-sized cluster
    rho <- 0.02


#Step 1: Computation of power assuming Beta(na, nb) mixture distribution
get.sp.sample <- function(i, n, a, b, Nsamp=10000)
  {
    #Computes saddlepoint distribution for cluster mean
    #and then samples from that distribution.
    cat("Deriving random sample for cluster ", i, "...\n", sep="")
    aa <<- n[i]*a
    bb <<- n[i]*b
    fp <<- function(p) dbeta(p, aa, bb)
    prob <- get.sp.distn(n[i])
    rm(fp, aa, bb)
    sample(0:n[i], Nsamp, replace=T, prob=prob)/n[i]
  }

#Parameters for Beta distribution
a.ctrl <- (p.ctrl * (1-rho)/rho)/mean(n.ctrl)
b.ctrl <- ((1-p.ctrl) * (1-rho)/rho)/mean(n.ctrl)
a.trt <- (p.trt * (1-rho)/rho)/mean(n.trt)
b.trt <- ((1-p.trt) * (1-rho)/rho)/mean(n.trt)

#Monte Carlo samples
data.ctrl <- sapply(1:n.clus, get.sp.sample, n=n.ctrl, a=a.ctrl,
                                             b=b.ctrl, Nsamp=Nsamp)
data.trt.null <- sapply(1:n.clus, get.sp.sample, n=n.trt, a=a.ctrl,
                                                  b=b.ctrl, Nsamp=Nsamp)
data.trt.alt <- sapply(1:n.clus, get.sp.sample, n=n.trt, a=a.trt,
                                                 b=b.trt, Nsamp=Nsamp)

#Finding critical value and power
trt <- matrix(c(rep(1/length(n.ctrl), length(n.ctrl)),
                rep(-1/length(n.trt), length(n.trt))), ncol=1, byrow=T)
null.distn <- cbind(data.ctrl, data.trt.null) %*% trt
alt.distn <- cbind(data.ctrl, data.trt.alt) %*% trt

crit.value <- sort(null.distn)[(1-alpha)*Nsamp]
tbpwr1 <- mean(alt.distn > crit.value)
```

```
#Step 2: Computation of power assuming Normal(a, b/n) mixture distribution
get.sp.sample <- function(i, n, a, b, Nsamp=10000)
  {
    #Computes saddlepoint distribution for cluster mean
    #and then samples from that distribution.
    cat("Deriving random sample for cluster ", i, "...\n", sep="")
    aa <<- a
    bb <<- sqrt(b/n[i])
    fp <<- function(p) {ok <- p>0 & p<1;
                        dnorm(logit(p), aa, bb)/(p*(1-p)+!ok)}
    prob <- get.sp.distn(n[i])
    rm(fp, aa, bb)
    sample(0:n[i], Nsamp, replace=T, prob=prob)/n[i]
  }

#Parameters for normal distribution
b.ctrl <- mean(n.ctrl)*rho/p.ctrl/(1-p.ctrl)
a.ctrl <- log(p.ctrl/(1-p.ctrl))
b.trt <- mean(n.trt)*rho/p.trt/(1-p.trt)
a.trt <- log(p.trt/(1-p.trt))

#Monte Carlo samples
data.ctrl <- sapply(1:n.clus, get.sp.sample, n=n.ctrl, a=a.ctrl,
                                             b=b.ctrl, Nsamp=Nsamp)
data.trt.null <- sapply(1:n.clus, get.sp.sample, n=n.trt, a=a.ctrl,
                                                  b=b.ctrl, Nsamp=Nsamp)
data.trt.alt <- sapply(1:n.clus, get.sp.sample, n=n.trt, a=a.trt,
                                                 b=b.trt, Nsamp=Nsamp)

#Finding critical value and power
trt <- matrix(c(rep(1/length(n.ctrl), length(n.ctrl)),
                rep(-1/length(n.trt), length(n.trt))), ncol=1, byrow=T)
null.distn <- cbind(data.ctrl, data.trt.null) %*% trt
alt.distn <- cbind(data.ctrl, data.trt.alt) %*% trt

crit.value <- sort(null.distn)[(1-alpha)*Nsamp]
tbpwr2 <- mean(alt.distn > crit.value)
```

```
#Step 3: Computation of power assuming a Gamma(an, b/n) mixture distribution
get.sp.sample <- function(i, n, a, b, Nsamp=10000)
  {
    #Computes saddlepoint distribution for cluster mean
    #and then samples from that distribution.
    cat("Deriving random sample for cluster ", i, "...\n", sep="")
    aa <<- n[i]*a
    bb <<- b/n[i]
    fp <<- function(p) {ok <- p>0 & p<1;
                        dgamma(-log(1-p*ok), aa, 1/bb)/(1-p*ok)}
    prob <- get.sp.distn(n[i])
    rm(fp, aa, bb)
    sample(0:n[i], Nsamp, replace=T, prob=prob)/n[i]
  }

#Parameters for gamma distribution
b.ctrl <- mean(n.ctrl)*rho*p.ctrl/(1-p.ctrl)/-log(1-p.ctrl)
a.ctrl <- -log(1-p.ctrl)/b.ctrl
b.trt <- mean(n.trt)*rho*p.trt/(1-p.trt)/-log(1-p.trt)
a.trt <- -log(1-p.trt)/b.trt

#Monte Carlo samples
data.ctrl <- sapply(1:n.clus, get.sp.sample, n=n.ctrl, a=a.ctrl,
                                             b=b.ctrl, Nsamp=Nsamp)
data.trt.null <- sapply(1:n.clus, get.sp.sample, n=n.trt, a=a.ctrl,
                                                  b=b.ctrl, Nsamp=Nsamp)
data.trt.alt <- sapply(1:n.clus, get.sp.sample, n=n.trt, a=a.trt,
                                                 b=b.trt, Nsamp=Nsamp)

#Finding critical value and power
trt <- matrix(c(rep(1/length(n.ctrl), length(n.ctrl)),
                rep(-1/length(n.trt), length(n.trt))), ncol=1, byrow=T)
null.distn <- cbind(data.ctrl, data.trt.null) %*% trt
alt.distn <- cbind(data.ctrl, data.trt.alt) %*% trt

crit.value <- sort(null.distn)[(1-alpha)*Nsamp]
tbpwr3 <- mean(alt.distn > crit.value)
```

```
#####Step 4: Compute power using GEE method of Pan/Shih#####
const <- 2*n.clus/sum(c(n.ctrl,n.trt)/(1+(c(n.ctrl, n.trt)-1)*rho))
vr <- 2*const*(1/(p.ctrl*(1-p.ctrl)) + 1/(p.trt*(1-p.trt)))
b <- log(p.trt/(1-p.trt)) - log(p.ctrl/(1-p.ctrl))
geepwr1 <- pnorm(qnorm(alpha) - b*sqrt(2*n.clus/vr))




#####Step 5: Compute power using GEE method of Liu/Liang#####
vif <- 1 + (mean(c(n.trt, n.ctrl))-1)*rho
num <- 2*n.clus*mean(c(n.trt, n.ctrl))*(p.trt-p.ctrl)^2/4
den <- vif/2*(p.ctrl*(1-p.ctrl) + p.trt*(1-p.trt))
geepwr2 <- pnorm(qnorm(alpha) + sqrt(num/den))




#####Step 6: Compute power using method of Hayes/Bennett#####
num <- (n.clus-1)*(p.ctrl-p.trt)^2
den <- p.ctrl*(1-p.ctrl)*mean(1/n.ctrl) + p.trt*(1-p.trt)*mean(1/n.trt)

k.ctrl1 <- mean(sqrt(b.ctrl/a.ctrl)/sqrt(a.ctrl*n.ctrl+b.ctrl*n.ctrl+1))
k.trt1 <- mean(sqrt(b.trt/a.trt)/sqrt(a.trt*n.trt+b.trt*n.trt+1))
den1 <- den + (k.ctrl1*p.ctrl)^2 + (k.trt1*p.trt)^2
hbpwr1 <- pnorm(qnorm(alpha) + sqrt(num/den1))

k.ctrl2 <- mean(sqrt(b.ctrl/n.ctrl)/(1+exp(a.ctrl)))
k.trt2 <- mean(sqrt(b.trt/n.trt)/(1+exp(a.trt)))
den2 <- den + (k.ctrl2*p.ctrl)^2 + (k.trt2*p.trt)^2
hbpwr2 <- pnorm(qnorm(alpha) + sqrt(num/den2))

k.ctrl3 <- mean(exp(-a.ctrl*b.ctrl)/
                    (1-exp(-a.ctrl*b.ctrl))*b.ctrl*sqrt(a.ctrl/n.ctrl))
k.trt3 <- mean(exp(-a.trt*b.trt)/(1-exp(-a.trt*b.trt))*b.trt*sqrt(a.trt/n.trt))
den3 <- den + (k.ctrl3*p.ctrl)^2 + (k.trt3*p.trt)^2
hbpwr3 <- pnorm(qnorm(alpha) + sqrt(num/den3))




#####Step 7: Combine results from all models#####
all.pwr <- rbind(all.pwr, c(n.clus, p.ctrl, p.trt, rho,
                            round(c(geepwr1, geepwr2, hbpwr1, hbpwr2, hbpwr3,
                                    tbpwr1, tbpwr2, tbpwr3),3)))
dimnames(all.pwr) <- list(1:nrow(all.pwr), c("M", "p.ctrl", "p.trt", "rho",
                            "G1", "G2", "Hb", "Hn", "Hg", "Bb", "Bn", "Bg"))
```