Unpublished document available at
http://www.res.otaru-uc.ac.jp/~emt-hogasa/,
https://barrel.repo.nii.ac.jp/.
January, 2021.

Supplement to the paper "A non-recursive formula for various moments of the multivariate normal distribution with sectional truncation"

Haruhiko Ogasawara
Otaru University of Commerce

This supplement gives the codes of the R-function "monsec" and the R-program using the function as an example, which yields the results in Table 8 of Ogasawara (2021). To run the program, the whole codes should be saved in a file with the name e.g., `monsecEX.R` in the working directory of the R process. Then, after the prompt of R, type `source('monsecEX.R')`. The results will be obtained in the same directory as the file `abc.u`.

It took about 106 seconds of "user" time using Intel Core i7-6700 CPU 3.40GHz. In monsec, the R-function "pmvnorm" in the R-package "mvtnorm", and the R-function "meanvarTMD" in the R-package MomTrunc are used. These two packages should be installed before monsec is used.

Reference

Ogasawara, H. (2021). A non-recursive formula for various moments of the multivariate normal distribution with sectional truncation. To appear in *Journal of Multivariate Analysis*.

```
#
# An example using function monsec to have partially absolute
# moments of the deviations of the bivariate normally distributed
# variables from arbitrary points with sectional truncation
#
#    <monsecEX.R>
#
#  Use as:   R prompt >source('file_name.R')
#
starttime=proc.time()
sink ('abc.u')
print(date(),quote=F)
library(MomTrunc)
library(mvtnorm)

############## The start of function monsec ############################
#
#           R-function [monsec]  version 2.0
#
#    The moments of the multivariate normal distribution
#               with sectional truncation
#                         2020
#                  Haruhiko Ogasawara
#       Otaru University of Commerce, Otaru, Japan
#
nCk=function(n,k) round(exp(lchoose(n,k))) # nCk=n!/{(n-k)!k!}

ldfac=function(i){ # log double factorial i.e., log(i!!)
if(i <= -2 || i %% 1 != 0){cat('!!!! invalid i=',i,'\n'); stop()}
if(i <= 1)ldfac=0
if(i >= 2)ldfac=sum(log(seq(i,1,by=-2)))
return(ldfac)
} # end of function ldfac

monsec=function(dtype,absol,k,A,B,mu,cov,d,
                eps=1e-6,mterm=300,dltuv=10){
############################ INPUT ###################################
#
# [dtype] 'r': the raw moment using vector X with d=0
#         'c': the central moment using vector X-E(X) with d=E(X)
#         'a': the moment using an arbitrary vector d
#              in the deviation X-d
# [absol] The n-vector of non-absolute (F) or absolute (T)
#         moment for each variable in a multivariate moment
# [k]     The n-vector of orders of a multivariate moment,
#         where 'n' is the mumber of normally distributed variables
# [A]     The n x R matrix of lower limits for selection,
#         where 'R' is the number of selected regions
```

```
# [B]     The n x R matrix of upper limits for selection
# [mu]    The n-vector of means of the multivariate normal
#         distribution without truncation
# [cov]   The n x n covariance matrix of the multivariate normal
#         distribution without truncation
# [d]     The n-vector of arbitrary values in the deviation X-d
#         when 'dtype' is 'a', otherwise any initial values for d
#         can be specified
# [eps]   The convergence criterion using the sum of current four
#         absolute terms divided by the absolute value of the moment
#         in a series, 1e-6(default)
# [mterm] The maximum order of terms for each variable in a series,
#         300(default)
# [dltuv] The increment of variables in a series after the first
#         four iterations, 10(default)
#
############################ OUTPUT ####################################
#
# The following are returned,when e.g.,
#   value=monsec(dtype,absol,k,A,B,mu,cov,d,eps,mterm,dltuv)
#
# [value$moment] The moment derived
# [value$nterms] The order of terms required to have convergence
#                for each variable in a series
# [value$rinc]   The final relative increment of variables
#                in a series, which should be smaller than
#                'eps' for convergence
# [value$alpha]  The value of 'alpha' or normalizer
#                in a truncated distribution

if(any(dim(A) != dim(B))){
cat('!!!!!! A,B= \n')
print(A); print(B)
stop()}

n=dim(A)[1]
R=dim(A)[2]
C=B-A
if(any(C <= 0))stop('!!!!!! b[i,j] <= a[i,j] \n')

if(any(absol == F & k %% 1 != 0)){
cat('!!!!! invalid absol=',absol,'k=',k=,'\n'); stop}

sig=sqrt(diag(cov))
cor=diag(1/sig) %*% cov %*% diag(1/sig)
tm=rep(0,n)
alpha=0
```

```
for (r in 1:R){
a=A[,r]
b=B[,r]
az=(a-mu)/sig
bz=(b-mu)/sig
alp=pmvnorm(lower=az,upper=bz,corr=cor)
alpha=alpha+alp
value=meanvarTMD(lower=a,upper=b,mu=mu,Sigma=cov,dist='normal')
tm=tm+alp*value$mean}

tm=tm/alpha
if(dtype == 'r')d=rep(0,n)
if(dtype == 'c')d=tm
Ad=A-matrix(d,n,R)
Bd=B-matrix(d,n,R)
md=mu-d
icov=solve(cov)

ak=rep(0,n)
for (i in 1:n)if(absol[i] == F)ak[i]=k[i]

mseq=NULL
dseq=NULL
ns=(n^2+n)/2

h0=1e100 # a large initial value

uv=-1
repeat{
if(uv <= 2)uv=uv+1
if(uv >= 3)uv=uv+dltuv  # to save computation time

muv=0

for (r in 1:R){
ad=Ad[,r]; bd=Bd[,r]

for (LL in 0:(2^n-1)){
L0=LL
L=NULL

for (i in 1:n){L=c(L0 %% 2,L); L0=L0 %/% 2} # the n-vector L is the
                                            # binary number of LL
cirdL=rep(0,n)
for (i in 1:n)cirdL[i]=ad[i]^L[i]*bd[i]^(1-L[i])

cabs=(-1)^sum(L)
for (i in 1:n)cabs=cabs*sign(cirdL[i])^(ak[i]+1)
```

```
nuv=rep(0,ns) # nuv will soon be used
for (i in 1:n)if( (icov %*% md)[i]*sign(cirdL[i]) != 0 )nuv[i]=1
ij=n
for (i in 1:(n-1)){
for (j in (i+1):n){
ij=ij+1

# avoid 0*Inf=NaN that cannot be compared with any numbers
if( -2*sign(cirdL[i])*sign(cirdL[j])*sign(icov[i,j]) != 0 )
 nuv[ij]=1}}

for (tuv in 0:((uv+1)^sum(nuv)-1)){ # (uv+1)^sum(nuv) is the number
                                    # of all the expanded terms
                                    # when uv is given
g=rep(0,ns)

tt=tuv
iuv=NULL

for (ii in 1:ns){
if(nuv[ii] == 0)iuv=c(iuv,0)  # not c(0,iuv)
else {iuv=c(iuv,tt %% (uv+1)); tt=tt %/% (uv+1)}
# not c(tt %% (uv+1),iuv)
}

# the ns-vector iuv is the 'reversed' base-(uv+1) number
# for variables u's and v's when uv is given
# (not 'sum(nuv)' but 'ns' in this case)

for (i in 1:n){
# avoid 0*Inf=NaN that cannot be compared with any numbers

aaa=(icov %*% md)[i]*sign(cirdL[i])
if(aaa == 0)abc=1
if(aaa != 0){
abc=exp( iuv[i]*log(abs( (icov %*% md)[i]*sqrt(2/icov[i,i]) ) )
         -lfactorial(iuv[i]) )
if(iuv[i] %% 2 == 1 && sign(aaa) == -1)
abc=-abc}

nt=iuv[i]
g1h1=n

for (g1 in 1:(n-1)){
for (h1 in (g1+1):n){
g1h1=g1h1+1
if(g1 == i || h1 == i)nt=nt+iuv[g1h1]}}
```

```
g[i]=sign(abc)*exp( lgamma( (k[i]+1+nt)/2 ) +
 log( pgamma( cirdL[i]^2*icov[i,i]/2,(k[i]+1+nt)/2 ) ) +log(abs(abc)) )

} # end of for (i in 1:n)

if(n >= 2){
ij=n
for (i in 1:(n-1)){
for (j in (i+1):n){
ij=ij+1
nuvij=iuv[ij]

scijc=sign(cirdL[i])*sign(cirdL[j])*sign(icov[i,j])
if( -2*scijc == 0 && nuvij == 0 )abc=1
if( -2*scijc == 0 && nuvij != 0 )abc=0
if( -2*scijc != 0){

abc=exp( nuvij*( log(abs(-2*icov[i,j]))
         -0.5*log(icov[i,i]*icov[j,j]) )
          -lfactorial(nuvij) )
if( nuvij %% 2 == 1 && sign(-2*scijc) == -1 )abc=-abc
} # end of if( -2*scijc != 0)
g[ij]=abc

}}
} # end of if(n >= 2)

srL=sign(cabs)
mrL=log(abs(cabs))
for (ij in 1:ns){
mrL=mrL+log( abs(g[ij]) )
srL=srL*sign(g[ij]) }

mrL=srL*exp(mrL)
muv=muv+mrL

} # end of for (tuv in 0:((uv+1)^sum(nuv)-1))
} # end of for (LL in 0:(2^n-1))
} # end of for (r in 1:R)

h1=muv
mseq=c(mseq,h1)
dseq=c(h1-h0,dseq)
h0=h1

msize=4
if(uv <= 2)msize=uv+1
```

```
inc=sum(abs(dseq[1:msize]))
rinc=inc/abs(muv)

# intermediate results when 'out1' is 1
out1=0
if(out1 == 1)cat('## uv=',uv,'muv=',muv,'rinc=',rinc,'\n')

# avoid NaN that cannot be compared with any numbers
if(is.nan(rinc))break

if(rinc < eps )break
if(uv > mterm){cat('!!!! no convergence, uv=',uv,'\n') ; break}
} # end of repeat

abc=1
for (i in 1:n)abc=abc*2^(k[i]/2)/icov[i,i]^((k[i]+1)/2)
abc=abc*exp( -t(md) %*% icov %*% md/2 )/
        ( alpha*(4*pi)^(n/2)*sqrt(det(cov)) )
muv=abc*muv

# the sequence of convergence when 'out2 is 1
out2=0
if(out2 == 1)cat('the sequnce of convergence \n',abc*mseq,'\n')

return(list(moment=muv,nterms=uv,rinc=rinc,alpha=alpha))

}
############ The end of function monsec ################################

ne1=0  # the start of the example-number
ne2=0  # the end of the example-number

xl=c('bivarite case of non-integer orders for absolute moments \n and
integer orders for non-absolute moments with sectional truncation \n')

n=2
R=2

eps=1e-6
mterm=300
dltuv=1

A=matrix(c(-1, 0,
           -1,-1),n,R,byrow=T)
B=matrix(c( 0, 1,
            1, 0),n,R,byrow=T)
mu=rep(0,n)
cov=matrix(c(1,0.5,0.5,1),n,n)
```

```
absol=c(T,F)

cat('\n\n\n')
cat('************************************************************ \n')
cat('************************************************************ \n')
cat('************************************************************ \n')
cat('\n')
cat('Partially-absolute multivariate moments under normality \n')

# the orders minus 1 of non-absolute moments
# the non-integer orders of absolute moments will be given later
# from mk
mk=c(-1:20,49,50,99,100,199,200,299,300,319,320,321)

for (dtype in c('r','c','a')){
d=rep(0,n)
cat('\n\n')
cat('************************************************************ \n')
cat('************************************************************ \n')
cat('Partially absolute multivariate')

if(dtype == 'r')cat(' raw moments \n')
if(dtype == 'c')cat(' central moments \n')
if(dtype == 'a'){
d=rep(0.3,n)
cat(' moments from the values \n')
cat('of the elements of vector d=',d,'\n')
}

for (ie in ne1:ne2){
cat('\n')
cat('************************************************************ \n')
cat('Example',ie,':',xl,'\n')

cat('n=',n, ' R=',R,'\n')
cat('A= ')
print(A,quote=F)
cat('B= ')
print(B,quote=F)

cat('mu=',mu,'\n')
cat('covariance matrix= \n')
print(cov,quote=F)
cat('eps=',eps,'\n')
cat('mterm=',mterm,'\n')
cat('dltuv=',dltuv,'\n')
```

```
for (k0 in mk){
k=c(k0+1/3,k0+1)    # a non-integer value and an integer
                    # for a bivariate moment
mmt=monsec(dtype,absol,k,A,B,mu,cov,d,eps,mterm,dltuv)

if(k0 == mk[1]){
cat('\n')
cat('k=the vector of orders of moment, ')
cat('ntm=the order of terms used \n')
cat('rinc=the relative increased value in a series \n')
cat('muv=the multivariate moment by <monsec> \n')

out=data.frame(k1=k[1],k2=k[2],ntm=mmt$nterms,rinc=mmt$rinc,
               muv=mmt$moment)
} # end of if(k0 == mk[1])

if(k0 != mk[1])out=
 rbind(out,c(k,mmt$nterms,mmt$rinc,mmt$moment))

}  # end of for (k0 in mk)-loop

cat('\n')
cat('alpha by <pmvnorm>=',mmt$alpha,'\n\n')
print(out)

}  # end of ie-loop
}  # end of dtype-loop

cat('\n')
print(date(),quote=F)
print(proc.time()-starttime,quote=F)

sink()
```