STAT 221 - ASSIGNMENT 1

GREG TAM

(1.1) For this question, we will apply Sylester's determinant theorem which states that if A and B are matrices of size $p \times n$ and $n \times p$ respectively,

$$\det(I_p + AB) = \det(I_n + BA)$$

We have the relationships

$$u_i = \frac{e^{x_i}}{1 + \sum_{j=1}^{d} e^{x_j}} \qquad x_i = \log\left(\frac{u_i}{u_{d+1}}\right) \qquad u_{d+1} = 1 - \sum_{j=1}^{d} u_j$$

We know that

$$f_X(\vec{x}) = (2\pi)^{-d/2} |\Sigma|^{-1/2} \exp\left(-\frac{1}{2}(\vec{x}-\mu)'\Sigma^{-1}(\vec{x}-\mu)\right)$$

and so by the multivariate transformation theorem we have

$$f_U(\vec{u}) = f_X(h_1(\vec{u}, \ldots, h_d(\vec{u}))|\det(J)|$$

where $h_i(\vec{u}) = \log\left(\frac{u_i}{u_{d+1}}\right)$ and J is a $d \times d$ matrix such that

$$J_{ij} = \frac{\partial h_i(\vec{u})}{\partial u_i}$$

These partial derivatives are given by

$$\begin{aligned}
\frac{\partial h_i(\vec{u})}{\partial u_i} &= \frac{u_{d+1}}{u_i}\left(\frac{u_{d+1}+u_i}{u_{d+1}^2}\right) \\
&= \frac{1}{u_i} + \frac{1}{u_{d+1}} \\
\frac{\partial h_i(\vec{u})}{\partial u_j} &= \frac{u_{d+1}}{u_i}\left(\frac{0+u_i}{u_{d+1}^2}\right) \\
&= \frac{1}{u_{d+1}}
\end{aligned}$$

and so

$$J = \begin{pmatrix}
\frac{1}{u_1} + \frac{1}{u_{d+1}} & \frac{1}{u_{d+1}} & \cdots & \frac{1}{u_{d+1}} \\
\frac{1}{u_{d+1}} & \frac{1}{u_2} + \frac{1}{u_{d+1}} & \cdots & \frac{1}{u_{d+1}} \\
\vdots & \vdots & \ddots & \vdots \\
\frac{1}{u_{d+1}} & \frac{1}{u_{d+1}} & \cdots & \frac{1}{u_d} + \frac{1}{u_{d+1}}
\end{pmatrix}$$

We can then use the fact that if we multiple any row by a constant, the determinant of the matrix is multipled by the same constant. So

$$\begin{aligned}
\det(J) &= \frac{1}{u_1 u_2 \cdots u_d} \begin{vmatrix} 1+\frac{u_1}{u_{d+1}} & \frac{u_1}{u_{d+1}} & \cdots & \frac{u_1}{u_{d+1}} \\ \frac{u_2}{u_{d+1}} & 1+\frac{u_2}{u_{d+1}} & \cdots & \frac{u_2}{u_{d+1}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{u_d}{u_{d+1}} & \frac{u_d}{u_{d+1}} & \cdots & 1+\frac{u_d}{u_{d+1}} \end{vmatrix} \\
&= \left(\prod_{i=1}^{d} u_i\right)^{-1} \det\left(I_d + \frac{1}{u_{d+1}} \begin{pmatrix} u_1 \\ \vdots \\ u_d \end{pmatrix} (1, \ldots, 1)\right) \\
&= \left(\prod_{i=1}^{d} u_i\right)^{-1} \det\left(1 + \frac{u_1 + \cdots + u_d}{u_{d+1}}\right) \\
&= \left(\prod_{i=1}^{d} u_i\right)^{-1} \frac{1}{u_{d+1}} \\
&= \left(\prod_{i=1}^{d} u_i\right)^{-1} \det\left(1 + \frac{1-u_{d+1}}{u_{d+1}}\right) \\
&= \left(\prod_{i=1}^{d+1} u_i\right)^{-1}
\end{aligned}$$

So the density of $\vec{u}$ is

$$f_U(\vec{u}) = (2\pi)^{-d/2}|\Sigma|^{-1/2} \exp\left(-\frac{1}{2}\{\log(u/u_{d+1}) - \mu\}'\Sigma^{-1}\{\log(u/u_{d+1}) - \mu\}\right) \qquad \text{for } u \in S^d$$

(1.2) Note that Σ is of the form

$$\Sigma = (\alpha+\beta)I_d + \beta U$$

where U is a matrix of ones. Σ^{-1} will be of the form

$$\Sigma^{-1} = c_1 I_d + c_2 U$$

for some constants c_1 and c_2. let us solve these by using the fact that

$$\{(\alpha+\beta)I_d - \beta U\}(c_1 I_d + c_2 U) = I_d$$

Expanding this gives

$$\begin{aligned}
(\alpha+\beta)c_1 I_d + (\alpha+\beta)c_2 U - \beta c_1 U - \beta c_2 d U &= I_d \\
(\alpha+\beta)c_1 I_d + \{(\alpha+\beta)c_2 - \beta(c_1 + c_2 d)\} &= I_d
\end{aligned}$$

Since the off-diagonal entries of the matrix I_d are 0, we know the off-diagonal entries of the left hand side are 0. Only the constant multiplied by U contributes to this, so

$$(\alpha+\beta)c_2 - \beta(c_1 + c_2 d) = 0 \Rightarrow c_2 = \frac{\beta c_1}{\alpha+\beta-\beta d}$$

Similarly because the diagonal entries are 1, we have

$$(\alpha+\beta)c_1 = 1 \Rightarrow c_1 = \frac{1}{\alpha+\beta}$$

Substituting this above, we have

$$c_2 = \frac{\beta}{(\alpha+\beta)(\alpha-(d-1)\beta)}$$

Using this, we can rewrite the density of U as

$$\begin{aligned}
f_U(\vec{u}) = \prod_{j=1}^{n} &|2\pi\{(\alpha+\beta)I_d - \beta U\}|^{-1/2} \left(\prod_{i=1}^{d+1} u_{j,i}\right)^{-1} \\
&\times \exp\left[-\frac{1}{2}\{\log(\vec{u}_j/u_{j,d+1}) - \mu\}'\left(\frac{1}{\alpha+\beta}I_d + \frac{\beta}{(\alpha+\beta)(\alpha-(d-1)\beta)}U\right)\{\log(\vec{u}_j/u_{j,d+1}) - \mu\}\right]
\end{aligned}$$

Differentiating with respect to μ, get that

$$\begin{aligned}
\frac{\partial}{\partial\mu} f_U(\vec{u}) &= \prod_{j=1}^{n} |2\pi\{(\alpha+\beta)I_d - \beta U\}|^{-1/2} \left(\prod_{i=1}^{d+1} u_{j,i}\right)^{-1} \\
&\quad \times \exp\left[-\frac{1}{2}\left\{-2\log(u_j/u_{j,d+1})'\Sigma^{-1} + 2\mu'\Sigma^{-1}\right\}\right] \\
&= \exp\left[\sum_{j=1}^{n} -\frac{1}{2}\left\{-2\log(u_j/u_{j,d+1})\Sigma^{-1} + 2\mu'\Sigma^{-1}\right\}\right] \\
&\quad \times \prod_{j=1}^{n} |2\pi\{(\alpha+\beta)I_d - \beta U\}|^{-1/2} \left(\prod_{i=1}^{d+1} u_{j,i}\right)^{-1} \\
&= \exp\left[\sum_{j=1}^{n} \log(u_j/u_{j,d+1})'\Sigma^{-1} - \mu'\Sigma^{-1}\right] \\
&\quad \times \prod_{j=1}^{n} |2\pi\{(\alpha+\beta)I_d - \beta U\}|^{-1/2} \left(\prod_{i=1}^{d+1} u_{j,i}\right)^{-1}
\end{aligned}$$

Setting this to 0 implies that

$$\sum_{j=1}^{n} \log(u_j/u_{j,d+1})\Sigma^{-1} = n\mu'\Sigma^{-1}$$

so we have

$$\hat{\mu} = \sum_{j=1}^{n} \frac{\log(u_j/u_{j,d+1})}{n}$$

When we find $\hat{\alpha}$, we differentiate the log-likelihood.

$$\begin{aligned}
\frac{\partial \log f_U(\vec{u})}{\partial\alpha} &= -\frac{1}{2}\left(\frac{d}{\alpha+\beta} + \frac{1}{\alpha-(d-1)\beta}\right) \\
&\quad + \frac{1}{2}\frac{1}{(\alpha+\beta)^2}\{\log(u/u_{d+1}) - \mu\}'\{\log(u/u_{d+1}) - \mu\} \\
&\quad - \frac{1}{2}\frac{(d-2)\beta - 2\alpha}{(\alpha+\beta)^2(\alpha-(d-1)\beta)^2}\{\log(u/u_{d+1}) - \mu\}'U\{\log(u/u_{d+1}) - \mu\}
\end{aligned}$$

again, where U is a matrix of ones. We set this to 0. Similarly, we differentiate the log-likelihood with respect to β to get

$$\begin{aligned}
\frac{\partial \log f_U(\vec{u})}{\partial\beta} &= -\frac{1}{2}\left(\frac{d}{\alpha+\beta} - \frac{d-1}{\alpha-(d-1)\beta}\right) \\
&\quad + \frac{1}{2}\frac{1}{(\alpha+\beta)^2}\{\log(u/u_{d+1}) - \mu\}'\{\log(u/u_{d+1}) - \mu\} \\
&\quad - \frac{1}{2}\frac{(\alpha+\beta)(\alpha-(d-1)\beta) + \alpha\beta(d-2) + 2(d-1)\beta^2}{(\alpha+\beta)^2(\alpha-(d-1)\beta)^2}\{\log(u/u_{d+1}) - \mu\}'U\{\log(u/u_{d+1}) - \mu\}
\end{aligned}$$

Setting both these equations, then solving for α and β will give you the MLEs.

(1.3)

```
dlogisticnorm = function(u,mu,alpha,beta)
{
  d = length(u)
  Sigma = matrix(-beta, nrow=d, ncol=d)
  for(i in 1:d)
    Sigma[i,i] = alpha
  x = log(u)
  ans = (2*pi)^(-d/2)*abs(det(Sigma))^(-1/2)*exp(-1/2*t(log(u/sum(u))-mu) %*% solve(Sigma) %*% (log(u/sum(u))-mu))
  as.numeric(ans)
}

logisticnorm.mle = function(U)
{
  wrapper = function(param)
  {
    tempMu = param[3:length(param)]
```

```
    d=length(tempMu)

    tempAlpha = abs(param[1])
    tempBeta = abs(param[2])

    loglikeli=0
    for(row in 1:nrow(U))
      loglikeli = loglikeli + log(dlogisticnorm(as.numeric(U[row,]),tempMu, tempAlpha, tempBeta))
    loglikeli
  }
  tempparam = c(1,1,rep(1,ncol(ldata)))
  tempoptim = optim(tempparam, wrapper, control=list(fnscale=-1))$par

  mu.hat = tempoptim[3:length(tempoptim)]
  alpha.hat = tempoptim[1]
  beta.hat = tempoptim[2]
  list("mu.hat" = mu.hat, "alpha.hat" = alpha.hat, "beta.hat" = beta.hat)
}
```

(1.4) Running the code written above, we get

```
ldata = read.table("dataLogisticNorm3D.txt", header = TRUE)

> logisticnorm.mle(ldata)
$mu.hat
[1]  3.865234  4.359615 -1.833477

$alpha.hat
[1] 2.513483

$beta.hat
[1] 1.282138
```

that is

$$\begin{aligned} \hat{\mu} &= (3.865234, 4.359615, -1.833477) \\ \hat{\alpha} &= 2.513483 \\ \hat{\beta} &= 1.282138 \end{aligned}$$

(2.1) Let $\ell = (l_1, \ldots, l_n)$. We have that

$$\begin{aligned} \mathbb{P}\Big(X_{n\times p} = \ell|\vec{\theta}\Big) &= \prod_{n=1}^{N} \mathbb{P}\Big(X_i^{(g)} = \ell_i\Big) \\ &= \prod_{n=1}^{N}\prod_{j=1}^{J} \mathbb{P}\Big(X_{ij}^{(g)} = \ell_{ij}|g=\gamma\Big)\ \mathrm{d}\mu(\gamma) \\ &= \prod_{n=1}^{N}\prod_{j=1}^{J} g_{L,i}\mathrm{Mult}(\vec{\theta}_{L,j}, 1) + g_{H,i}\mathrm{Mult}(\vec{\theta}_{H,j}, 1) \end{aligned}$$

Taking logs, we get that the log-likelihood is

$$\begin{aligned} \ell(\vec{\theta}) &= \sum_{n=1}^{N} \log\left(\prod_{j=1}^{J} g_{L,i}\mathrm{Mult}(\vec{\theta}_{L,j}, 1) + g_{H,i}\mathrm{Mult}(\vec{\theta}_{H,j}, 1)\right) \\ &= \sum_{n=1}^{N}\sum_{j=1}^{J} \log\Big\{g_{L,i}\mathrm{Mult}(\vec{\theta}_{L,j}, 1) + g_{H,i}\mathrm{Mult}(\vec{\theta}_{H,j}, 1)\Big\} \end{aligned}$$

(2.2)

```
ll = function(G, theta, X)
{
  #two make things easier to read, theta is a list of lists of vectors
  n = dim(G)[1]
  J = length(theta[[1]])
  likelihood_sum = 0
  for(i in 1:n) #for each observation
  {
    for(j in 1:J) #for each feature
    {
      #The theta are represented by a list of two lists
      #Each of the sublists represents a vector of probabilities for each feature
      #high
      theta_high = theta[[1]]
      #low
      theta_low = theta[[2]]

      theta_Lj = theta[[2]][[j]]
```

```
      theta_Hj = theta[[1]][[j]]

      val_L = theta_Lj[j]
      val_H = theta_Hj[j]
      likelihood_sum = likelihood_sum + G[n,1] * val_H + G[n,2] * val_L
    }
  }
  log(likelihood_sum)
}
```

(2.3)

```
gomMLE = function(X, G0, theta0)
{
  ll.G = function(param, G, index, theta, X)
  {
    N = length(G)/2
    P = length(theta)
    log_likelihood_sum = 0
    G[index] = param

    getHighVal = function(p,n)
    {
      theta[[p]]$high[Xmin[n,p]]
    }
    getLowVal = function(p,n)
    {
      theta[[p]]$low[Xmin[n,p]]
    }
    val_H = sapply(1:49, getHighVal,n=1:69)
    val_L = sapply(1:49, getLowVal,n=1:69)

    val_G = matrix(G[1:N], nrow=N,ncol=dim(val_H)[2])
    val_G2 = 1 - val_G

    mix = val_G * val_H + val_G2 * val_L
    sum(log(ifelse(mix>0, mix, mix+exp(1e-16))))
  }
  ll.thetaL = function(param, G, index, theta, X)
  {
    N = length(G)/2
    P = length(theta)
    log_likelihood_sum = 0
    #u_i
    scaled_param = exp(param)/(sum(exp(param)))
    theta[[index]]$low = scaled_param

    getHighVal = function(p,n)
    {
      theta[[p]]$high[Xmin[n,p]]
    }
    getLowVal = function(p,n)
    {
      theta[[p]]$low[Xmin[n,p]]
    }
    val_H = sapply(1:49, getHighVal,n=1:69)
    val_L = sapply(1:49, getLowVal,n=1:69)

    val_G = matrix(G[1:N], nrow=N,ncol=dim(val_H)[2])
    val_G2 = 1 - val_G

    mix = val_G * val_H + val_G2 * val_L
    sum(log(ifelse(mix>0, mix, mix+exp(1e-16))))
  }
  ll.thetaH = function(param, G, index, theta, X)
  {
    N = length(G)/2
    P = length(theta)
    log_likelihood_sum = 0
    #u_i
    scaled_param = exp(param)/(sum(exp(param)))
    theta[[index]]$high = scaled_param

    getHighVal = function(p,n)
    {
      theta[[p]]$high[Xmin[n,p]]
    }
    getLowVal = function(p,n)
    {
      theta[[p]]$low[Xmin[n,p]]
    }
    val_H = sapply(1:49, getHighVal,n=1:69)
    val_L = sapply(1:49, getLowVal,n=1:69)

    val_G = matrix(G[1:N], nrow=N,ncol=dim(val_H)[2])
    val_G2 = 1 - val_G

    mix = val_G * val_H + val_G2 * val_L
    sum(log(ifelse(mix>0, mix, mix+exp(1e-16))))
  }
```

```
  optTheta = theta0List
  optG = G
  likeli=c()

  for(counter in 1:11)
  {
    print(paste("Doing Count", counter, Sys.time()))
    #optimize G
    for(i in 1:69)
      optG[i] = optim(0.5, ll.G, control=list(fnscale=-1), method = "L-BFGS-B", lower=0, upper=1, theta = optTheta, X=X, G=optG, index=i)$par
    optG[,2] = 1-optG[,1]


    #optimize thetaL
    for(index in 1:length(theta))
    {
      #optimize thetaL
      init = rep(0,length(theta[[index]]$low))
      thetaL = optim(init, ll.thetaL, control=list(fnscale=-1), method = "L-BFGS-B", theta = optTheta,X=X,G=optG,index=index)$par
      optL = exp(thetaL)/(sum(exp(thetaL)))
      optL
      optTheta[[index]]$level = c(1: length(optL)-1)
      optTheta[[index]]$low = optL
    }
    for(index in 1:length(theta))
    {
      #optimize thetaH
      init = rep(0,length(theta[[index]]$high))
      thetaH = optim(init, ll.thetaH, control=list(fnscale=-1), method = "L-BFGS-B", theta = optTheta,X=X,G=optG,index=index)$par
      optH = exp(thetaH)/(sum(exp(thetaH)))
      optH
      optTheta[[index]]$high = optH
    }
    likeli[counter] = ll(optG, optTheta,X[-1,-1])
  }
  list(G.hat = optG, theta.hat = optTheta, maxlik=likeli[length(likeli)])
}

gomMLE(X,G,theta)
```

The final sequence of log-likelihoods is

```
 [1] -2191.750 -2111.485 -2018.332 -1983.130 -1972.673 -1969.443 -1967.806 -1966.313 -1965.264 -1964.829 -1964.449
[12] -1964.177
```

Note that I use log-likelihoods and not likelihoods here. (If you take e to the power of these log-likelihoods to get back the likelihood, `R` will round it down to 0). We see that this roughly converges to -1964.

This yields the MLE of (G, Θ) as given by `optG` and `optTheta` respectively in my code. These are given by

```
> optG
           [,1]       [,2]
 [1,] 0.2022443 0.79775569
 [2,] 0.5920600 0.40794004
 [3,] 0.5145629 0.48543714
 [4,] 0.7113350 0.28866502
 [5,] 0.5511282 0.44887182
 [6,] 0.7498088 0.25019124
 [7,] 1.0000000 0.00000000
 [8,] 0.3420241 0.65797589
 [9,] 0.1397390 0.86026101
[10,] 0.2445129 0.75548706
[11,] 1.0000000 0.00000000
[12,] 0.4204931 0.57950690
[13,] 0.8943193 0.10568072
[14,] 0.5663744 0.43362564
[15,] 0.4148697 0.58513026
[16,] 0.9774461 0.02255392
[17,] 0.4738269 0.52617306
[18,] 0.7284804 0.27151957
[19,] 0.4307313 0.56926872
[20,] 0.3880238 0.61197620
[21,] 0.3842778 0.61572217
[22,] 0.6702327 0.32976726
[23,] 0.6084578 0.39154222
[24,] 0.7136105 0.28638947
[25,] 0.4882562 0.51174379
[26,] 0.6075402 0.39245985
[27,] 0.4103282 0.58967178
[28,] 0.7163914 0.28360863
[29,] 0.5640968 0.43590324
[30,] 0.3695819 0.63041806
[31,] 0.7363522 0.26364776
[32,] 0.6598992 0.34010078
[33,] 1.0000000 0.00000000
[34,] 0.7152899 0.28471011
```

```
[35,] 0.7872013 0.21279870
[36,] 0.7878792 0.21212082
[37,] 0.5509311 0.44906891
[38,] 0.6989309 0.30106907
[39,] 0.5313829 0.46861714
[40,] 0.7051531 0.29484689
[41,] 0.6208766 0.37912339
[42,] 0.7792000 0.22080002
[43,] 0.7423650 0.25763504
[44,] 0.8569172 0.14308280
[45,] 0.6553577 0.34464233
[46,] 0.4599206 0.54007939
[47,] 0.6942858 0.30571422
[48,] 0.5529256 0.44707439
[49,] 0.7369606 0.26303939
[50,] 0.8354857 0.16451431
[51,] 0.8389240 0.16107604
[52,] 0.6653884 0.33461158
[53,] 0.5903246 0.40967542
[54,] 0.3072348 0.69276515
[55,] 0.7014028 0.29859723
[56,] 0.8649334 0.13506656
[57,] 0.5074752 0.49252479
[58,] 0.7187042 0.28129584
[59,] 0.2316812 0.76831883
[60,] 0.5796236 0.42037636
[61,] 0.2350011 0.76499888
[62,] 0.3088390 0.69116100
[63,] 0.2455026 0.75449745
[64,] 0.5211141 0.47888585
[65,] 0.5589738 0.44102620
[66,] 0.4073884 0.59261159
[67,] 0.4112768 0.58872316
[68,] 0.4810227 0.51897727
[69,] 1.0000000 0.00000000
> optTheta
[[1]]
  level       high          low
1     0 0.95122798 9.999995e-01
2     1 0.04877202 4.939291e-07

[[2]]
    level         high          low
116     0 3.372277e-01 1.749641e-01
117     1 6.627711e-01 8.250214e-01
118     2 1.231833e-06 1.452825e-05

[[3]]
   level      high          low
47     0 0.5724011 6.524463e-06
48     1 0.4275989 9.999935e-01

[[4]]
   level         high          low
49     0 9.341093e-01 4.318023e-01
50     1 6.588961e-02 5.681867e-01
51     2 1.098046e-06 1.103693e-05

[[5]]
   level         high          low
52     0 7.034463e-01 1.985720e-07
53     1 2.965461e-01 9.999997e-01
54     2 7.602929e-06 6.436465e-08

[[6]]
   level       high          low
11     0 0.46645800 5.250676e-01
12     1 0.34154653 4.294917e-01
13     2 0.17392625 8.198358e-07
14     3 0.01806922 4.543986e-02

[[7]]
   level         high          low
15     0 6.684080e-01 3.452745e-01
16     1 1.996255e-01 5.332791e-01
17     2 1.264059e-01 6.005227e-07
18     3 3.292714e-07 1.005449e-01
19     4 5.560261e-03 2.090090e-02

[[8]]
   level       high        low
8      0 0.55819282 0.32711048
9      1 0.37666883 0.63404349
10     2 0.06513836 0.03884603

[[9]]
   level       high       low
23     0 0.51186849 0.3452297
24     1 0.46447769 0.3993731
25     2 0.02365382 0.2553972
```

```
[[10]]
   level       high         low
26     0 0.56381847 0.002628358
27     1 0.38931326 0.796713081
28     2 0.04686827 0.200658561

[[11]]
   level         high          low
29     0 7.592923e-01 5.852873e-01
30     1 2.407076e-01 4.147119e-01
31     2 1.047939e-07 7.417663e-07

[[12]]
   level         high          low
32     0 7.331035e-01 9.543684e-01
33     1 2.668962e-01 4.563123e-02
34     2 3.009835e-07 3.494638e-07

[[13]]
   level         high          low
35     0 7.860295e-01 6.761523e-01
36     1 2.139604e-01 3.238475e-01
37     2 1.018396e-05 1.915744e-07

[[14]]
   level         high          low
20     0 7.415954e-02 6.108494e-08
21     1 9.258400e-01 9.999999e-01
22     2 4.258952e-07 8.529466e-08

[[15]]
   level         high          low
38     0 9.432546e-01 9.799661e-01
39     1 5.674519e-02 2.003374e-02
40     2 2.045636e-07 1.501653e-07

[[16]]
   level         high          low
41     0 8.518342e-01 9.999998e-01
42     1 4.938522e-02 9.698064e-08
43     2 4.940546e-02 4.205459e-08
44     3 4.937498e-02 5.071249e-08
45     4 6.296044e-08 2.983808e-08
46     5 6.296044e-08 2.983808e-08

[[17]]
   level       high        low
55     0 0.81458247 0.75032543
56     1 0.17355605 0.23158973
57     2 0.01186148 0.01808484

[[18]]
   level         high          low
58     0 1.330517e-01 1.671536e-01
59     1 8.669473e-01 8.328444e-01
60     2 9.652733e-07 1.976115e-06

[[19]]
   level         high          low
61     0 9.731017e-01 4.441416e-01
62     1 2.689808e-02 5.558579e-01
63     2 1.777464e-07 5.076253e-07

[[20]]
   level         high          low
64     0 9.999997e-01 4.463130e-01
65     1 2.129002e-07 5.536867e-01
66     2 3.831725e-08 2.595316e-07

[[21]]
   level         high          low
67     0 9.999998e-01 3.510484e-01
68     1 1.093960e-07 6.489514e-01
69     2 4.068797e-08 1.353015e-07

[[22]]
   level         high          low
70     0 9.999999e-01 9.999999e-01
71     1 5.065144e-08 6.899596e-08
72     2 5.065144e-08 6.899596e-08

[[23]]
   level       high        low
73     0 0.81608603 0.42700951
74     1 0.17459636 0.55581994
75     2 0.00931761 0.01717055

[[24]]
   level         high          low
```

```
76     0 9.999987e-01 6.738660e-01
77     1 1.212376e-06 3.261339e-01
78     2 6.392644e-08 1.250426e-07

[[25]]
   level         high          low
79     0 9.999999e-01 9.999999e-01
80     1 5.065401e-08 6.899694e-08
81     2 5.065401e-08 6.899694e-08

[[26]]
   level         high          low
82     0 8.059331e-01 3.255896e-01
83     1 1.940655e-01 6.744038e-01
84     2 1.410124e-06 6.566671e-06

[[27]]
   level         high          low
85     0 7.720332e-01 4.803198e-01
86     1 2.279667e-01 5.196801e-01
87     2 9.222411e-08 1.127543e-07

[[28]]
   level       high          low
88     0 0.46750152 8.616822e-02
89     1 0.50842670 9.138313e-01
90     2 0.02407178 4.702528e-07

[[29]]
   level       high          low
91     0 0.16978022 3.364485e-02
92     1 0.80585533 9.663548e-01
93     2 0.02436445 3.355641e-07

[[30]]
   level       high        low
94     0 0.60408308 0.79830631
95     1 0.33811966 0.03937354
96     2 0.05779726 0.16232016

[[31]]
    level       high          low
97      0 0.54414287 5.737808e-07
98      1 0.29786518 2.442359e-01
99      2 0.14057174 7.415752e-01
100     3 0.01742022 1.418832e-02

[[32]]
    level         high          low
101     0 9.999999e-01 7.388560e-01
102     1 8.694427e-08 2.611435e-01
103     2 5.205112e-08 5.368759e-07

[[33]]
    level         high          low
104     0 9.999998e-01 6.616942e-01
105     1 1.760996e-07 3.383050e-01
106     2 6.240362e-08 7.629321e-07

[[34]]
    level         high          low
107     0 9.937773e-01 9.639465e-01
108     1 6.222630e-03 3.605238e-02
109     2 1.030070e-07 1.107838e-06

[[35]]
    level         high          low
110     0 9.761541e-01 3.077380e-07
111     1 1.387839e-06 9.999996e-01
112     2 2.384448e-02 8.902263e-08

[[36]]
    level         high          low
113     0 9.750765e-01 2.505462e-01
114     1 2.202134e-06 7.494510e-01
115     2 2.492125e-02 2.807375e-06

[[37]]
    level       high          low
119     0 0.21881317 5.930624e-01
120     1 0.25270411 3.017993e-01
121     2 0.50388940 1.051347e-01
122     3 0.02459332 3.556599e-06

[[38]]
    level      high        low
123     0 0.5223827 0.91582944
124     1 0.4776173 0.08417056

[[39]]
```

```
    level      high        low
125     0 0.7896096 0.98815467
126     1 0.2103904 0.01184533

[[40]]
    level      high       low
127     0 0.8482605 0.2112293
128     1 0.1517395 0.7887707

[[41]]
    level         high          low
129     0 4.865334e-02 1.439933e-06
130     1 5.502268e-01 4.769374e-01
131     2 2.348304e-01 9.975219e-02
132     3 1.662893e-01 4.233090e-01
133     4 8.203881e-08 4.589863e-08

[[42]]
    level         high          low
134     0 2.709244e-01 9.999997e-01
135     1 7.290718e-01 2.116974e-07
136     2 3.850704e-06 7.185455e-08

[[43]]
    level      high          low
137     0 0.3045805 9.291060e-01
138     1 0.1253773 2.891591e-02
139     2 0.1973372 1.968937e-05
140     3 0.1233531 2.459023e-07
141     4 0.1480152 1.157781e-07
142     5 0.1013366 4.195800e-02

[[44]]
    level        high          low
143     0 0.049572205 2.351116e-01
144     1 0.048801085 2.940620e-07
145     2 0.645505079 6.081669e-01
146     3 0.252929435 1.342445e-01
147     4 0.003192197 2.247669e-02

[[45]]
    level         high          low
148     0 5.839674e-02 4.468540e-01
149     1 4.173608e-01 7.944629e-05
150     2 5.242385e-01 5.530501e-01
151     3 3.926355e-06 1.650303e-05

[[46]]
    level         high          low
152     0 2.486021e-06 4.303617e-01
153     1 5.616484e-01 1.380647e-01
154     2 2.017105e-01 4.194034e-07
155     3 9.818790e-02 1.219621e-01
156     4 1.384507e-01 3.096109e-01
157     5 1.742494e-08 3.224079e-08
158     6 1.742494e-08 3.224079e-08
159     7 1.742494e-08 3.224079e-08

[[47]]
    level         high         low
160     0 2.142164e-01 0.114642470
161     1 6.013124e-01 0.723675873
162     2 1.623300e-01 0.099593082
163     3 1.299026e-05 0.056527802
164     4 2.212819e-02 0.005560773

[[48]]
    level         high          low
165     0 8.035841e-01 9.648397e-01
166     1 1.964150e-01 3.774244e-07
167     2 9.237721e-07 3.515988e-02

[[49]]
  level         high          low
3     0 8.915348e-01 8.737094e-01
4     1 3.550409e-02 1.262881e-01
5     2 7.295939e-02 1.581563e-06
6     3 8.367568e-07 4.450489e-07
7     4 8.367568e-07 4.450489e-07
```