STA410HW1

Yuesheng Li 1002112064

03/10/2019

1 DCT to Denoise an Image

1.1 Matrix Transformation

Proof. Let $\hat{Z} = A_m Z A_m^T$, a $m \times n$ matrx be given. Let $\{A_n\}_n$ be a family of matrices satisfying $A_n^T A_n = D_n$, where D_n is diagonal.Consider the following :

$$\begin{aligned} D_m^{-1} A_m^T \hat{Z} A_n D_n^{-1} &= D_m^{-1} A_m^T A_m Z A_m^T A_n D_n^{-1} \\ &= D_m^{-1} (A_m^T A_m) Z (A_m^T A_n) D_n^{-1} \\ &= D_m^{-1} D_m Z D_n D_n^{-1} \\ &= Z \end{aligned} \tag{1}$$

□

1.2 Threshold Transformation

The following function are the `r` function for threshold transformation, `denoiceH` is the hard-thresholding and `denoiceS` stands for soft-thresholding:

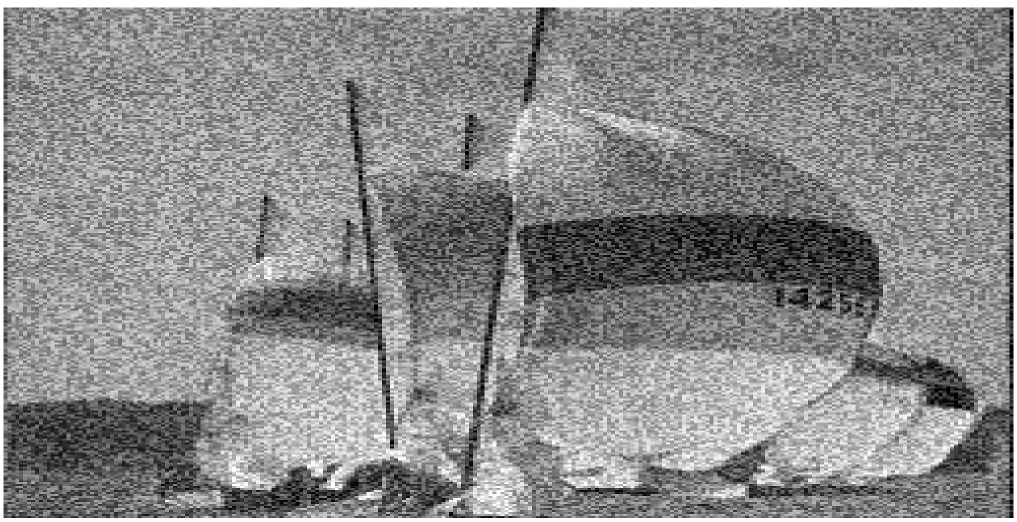
```
denoiseH <- function(dctmat,quant) {
  # Do the DCT on matrix
  dctmatT <- mvdct(dctmat)
  # if quant is missing, set it to the 0.8
  if(missing(quant)) {lambda <- quantile(abs(dctmatT),0.8)}
  else {lambda <- quantile(abs(dctmatT),quant)}
  # hard-thresholding
  a <- dctmatT[1,1]
  dctmat1 <- ifelse(abs(dctmatT)>lambda,dctmatT,0)
  dctmat1[1,1] <- a
  # inverse DCT to obtain denoised image "clean"
  clean <- mvdct(dctmat1,inverted=T)
  clean <- ifelse(clean<0,0,clean)
  clean <- ifelse(clean>1,1,clean)
  clean
}


denoiseS <- function(dctmat,lambda) {
  # Do the DCT on matrix
  dctmatT <- mvdct(dctmat)
  # if lambda is missing, set it to the 0.8 quantile of abs(dctmat)
  if(missing(lambda)) lambda <- quantile(abs(dctmatT),0.8)
  # soft-thresholding
  a <- dctmatT[1,1]
  dctmat1 <- sign(dctmatT)*pmax(abs(dctmatT)-lambda, 0)
  dctmat1[1,1] <- a
  # inverse DCT to obtain denoised image "clean"
```

```
  clean <- mvdct(dctmat1,inverted=T)
  clean <- ifelse(clean<0,0,clean)
  clean <- ifelse(clean>1,1,clean)
  clean
}
```

1.3 Denoise Methods and Results

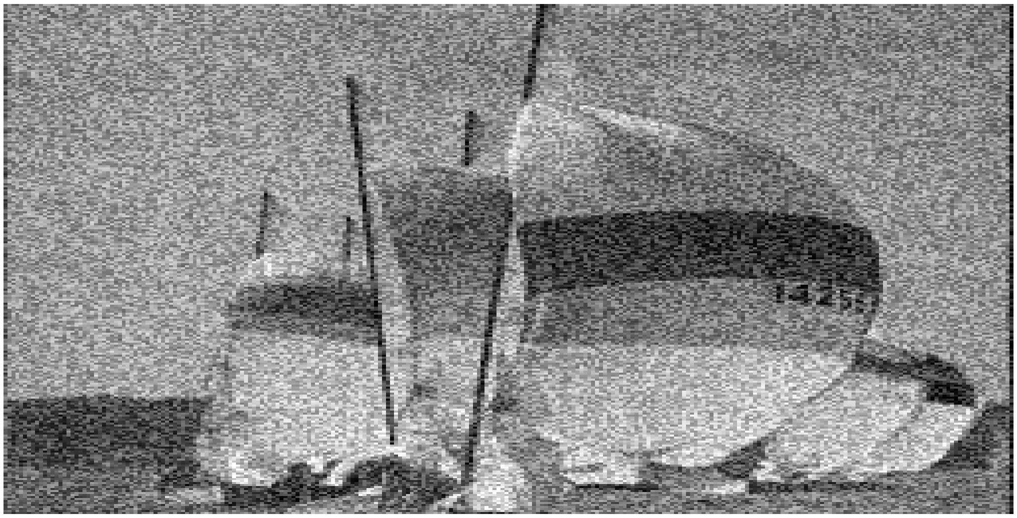
First of all, take a look at the default image.

```
image(boat, axes=F, col=grey(seq(0,1,length=256)))
```

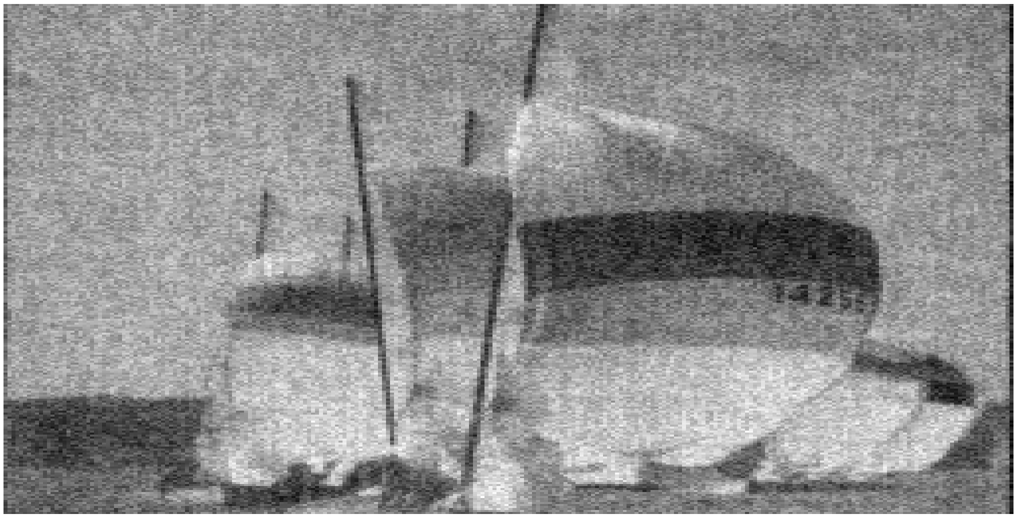


Now, try the ones we have above with different threshold λ

```
# Hard-threshold with quantile 0.4
K <- denoiseH(boat, 0.4)
image(K, axes=F, col=grey(seq(0,1,length=256)))
```



```r
# Soft-threshold with lambda = 12
K <- denoiseS(boat, 12)
image(K, axes=F, col=grey(seq(0,1,length=256)))
```



2 Hermite Distribution and FFT

2.1 PGF

Proof. Let U, V be independent Poisson r.v. with means λ_u and λ_v.Define $X = U + 2V$
Let $g_U(s)$, $g_V(s)$ and $g_X(s)$ denote the pgf of U, V, X respectively.

$$P(U = k) = \frac{\lambda_u^k e^{-\lambda}}{k!} \tag{2}$$

$$P(V = k) = \frac{\lambda_v^k e^{-\lambda}}{k!} \tag{3}$$

Now, using equation 2 and 3 together with taylor expansion of e

$$\begin{aligned} g_U(s) &= E(s^U) \\ &= \Sigma_{j=0}^{\infty} P(U = j) s^j \\ &= \Sigma_{j=0}^{\infty} \frac{\lambda_u^j e^{-\lambda_u}}{j!} s^j \\ &= e^{-\lambda_u} \Sigma_{j=0}^{\infty} \frac{(\lambda_u s)^j}{j!} \\ &= e^{-\lambda_u} \cdot e^{\lambda_u s} \\ &= e^{\lambda_u (s-1)} \end{aligned} \tag{4}$$

Similarly,

$$\begin{aligned}
g_{2V}(s) &= E(s^{2V}) \\
&= \Sigma_{j=0}^{\infty} P(V=j) s^{2j} \\
&= \Sigma_{j=0}^{\infty} \frac{\lambda_v^j e^{-\lambda_v}}{j!} s^{2j} \\
&= e^{-\lambda_v} \Sigma_{j=0}^{\infty} \frac{(\lambda_v s^2)^j}{j!} \\
&= e^{-\lambda_v} \cdot e^{\lambda_v s^2} \\
&= e^{\lambda_v (s^2-1)}
\end{aligned} \tag{5}$$

Now we put everything together, and the fact that U and V are independent.

$$\begin{aligned}
g_X(s) &= E(s^X) \\
&= E(s^{(U+2V)}) \\
&= E(s^U) \cdot E(s^{2V}) \\
&= e^{\lambda_u (s-1)} \cdot e^{\lambda_v (s^2-1)} \\
&= e^{\lambda_u (s-1) + \lambda_v (s^2-1)}
\end{aligned} \tag{6}$$

□

2.2 Finding M

Proof. Fix some $\epsilon > 0$. By *Markov's Inequality*, we have

$$P(X \geq M) = P(s^X \geq s^M) \leq \frac{E(s^X)}{s^M} = \frac{\exp[\lambda_u(s-1) + \lambda_v(s^2-1)]}{s^M} \tag{7}$$

Now, to ensure $P(X \geq M) \leq \epsilon$, we can first determine a M^* for each $s > 1$ such that,

$$\begin{aligned}
\epsilon &= \frac{E(s^X)}{s^{M^*}} \geq P(X \geq M^*) \\
\epsilon &= \frac{\exp(\lambda_u(s-1) + \lambda_v(s^2-1))}{s^{M^*}}
\end{aligned} \tag{8}$$

$$\begin{aligned}
\Rightarrow s^{M^*} &= \frac{\exp(\lambda_u(s-1) + \lambda_v(s^2-1))}{\epsilon} \\
\Rightarrow M^* \ln(s) &= \lambda_u(s-1) + \lambda_v(s^2-1) - \ln(\epsilon) \\
\Rightarrow M^* &= \frac{\lambda_u(s-1) + \lambda_v(s^2-1) - \ln(\epsilon)}{\ln(s)}
\end{aligned} \tag{9}$$

In fact, we can view M^* as a function of s, ie. $M^*(s)$. Then we take

$$M = \inf_{s>1} M^*(s) = \inf_{s>1} \frac{\lambda_u(s-1) + \lambda_v(s^2-1) - \ln(\epsilon)}{\ln(s)} \tag{10}$$

To complete the proof, fix $\delta > 0$, there exists an $s > 1$ for $M^*(s)$ such that

$$M + \delta > M^*(s) \tag{11}$$

Then we have:

$$\frac{\exp(\lambda_u(s-1)+\lambda_v(s^2-1))}{s^{M+\delta}} < \frac{\exp(\lambda_u(s-1)+\lambda_v(s^2-1))}{s^{M^*(x)}} \tag{12}$$

$$\Rightarrow \frac{\exp(\lambda_u(s-1)+\lambda_v(s^2-1))}{s^{M+\delta}} < \epsilon \tag{13}$$

Now shrink $\delta \to 0$, we can conclude that

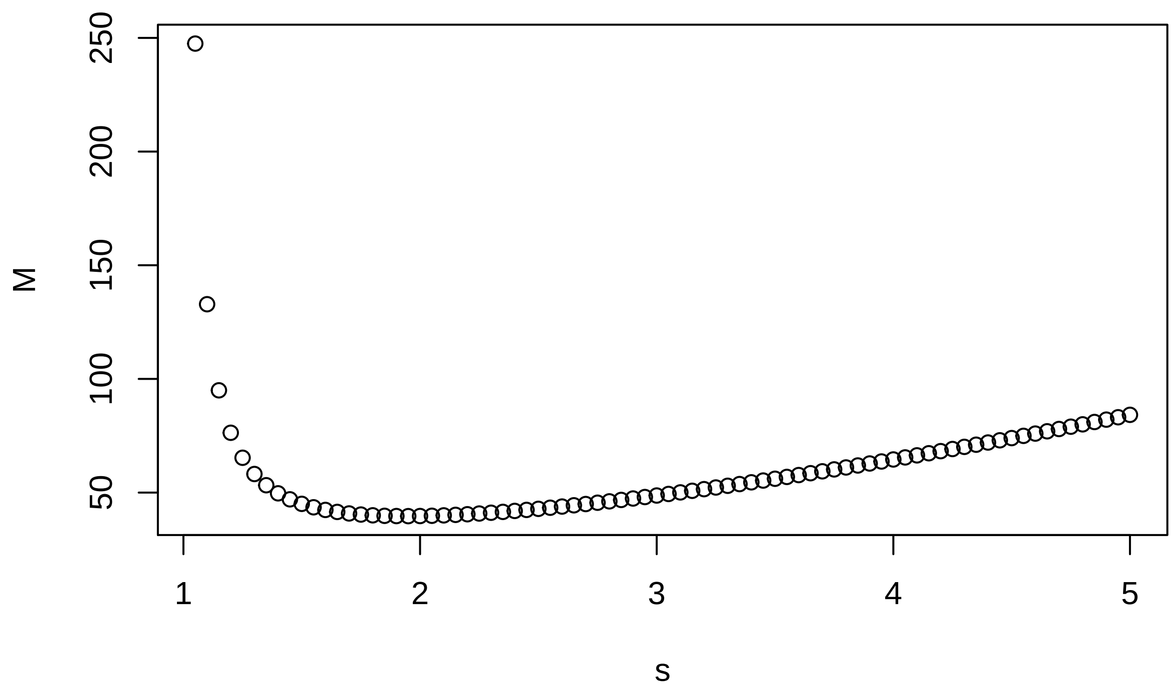
$$P(X \geq M) \leq \frac{E(s^X)}{s^M} = \frac{\exp(\lambda_u(s-1)+\lambda_v(s^2-1))}{s^M} \leq \epsilon \tag{14}$$

□

2.3 FFT for Distribution

Before we determine value of M, we first make some plots of $M vs. s$ with fixed $\epsilon, \lambda_u, \lambda_v$

```
Mplot(20, 1, 5)
```



We can observe that the min of M is reached when s is approxmately 2 Now we evalute the value M and do FFT.

```
M <- Mval(20, 1, 5)
M
```

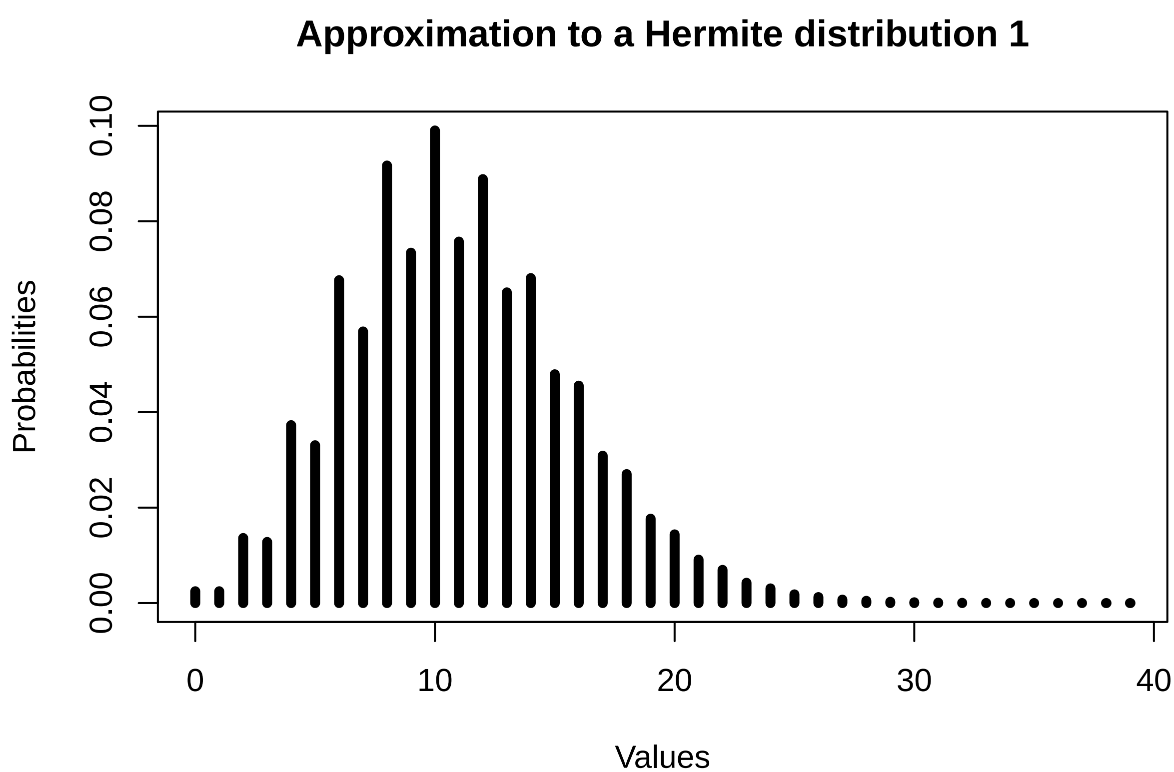
```
## [1] 40
```

```r
p1 <- eval(M, 1, 5)
p1
```

```
## $x
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
## [24] 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
##
## $prob
##  [1] 2.479113e-03 2.478928e-03 1.363323e-02 1.280693e-02 3.728459e-02
##  [6] 3.307070e-02 6.765273e-02 5.690851e-02 9.167947e-02 7.341829e-02
## [11] 9.902129e-02 7.574583e-02 8.882990e-02 6.509909e-02 6.809986e-02
## [16] 4.793939e-02 4.555863e-02 3.087956e-02 2.702588e-02 1.767481e-02
## [21] 1.439668e-02 9.102134e-03 6.957679e-03 4.259957e-03 3.076531e-03
## [26] 1.827044e-03 1.253552e-03 7.231109e-04 4.735226e-04 2.656769e-04
## [31] 1.666968e-04 9.107956e-05 5.493897e-05 2.926468e-05 1.701925e-05
## [36] 8.847602e-06 4.973335e-06 2.525658e-06 1.375237e-06 6.828672e-07
```

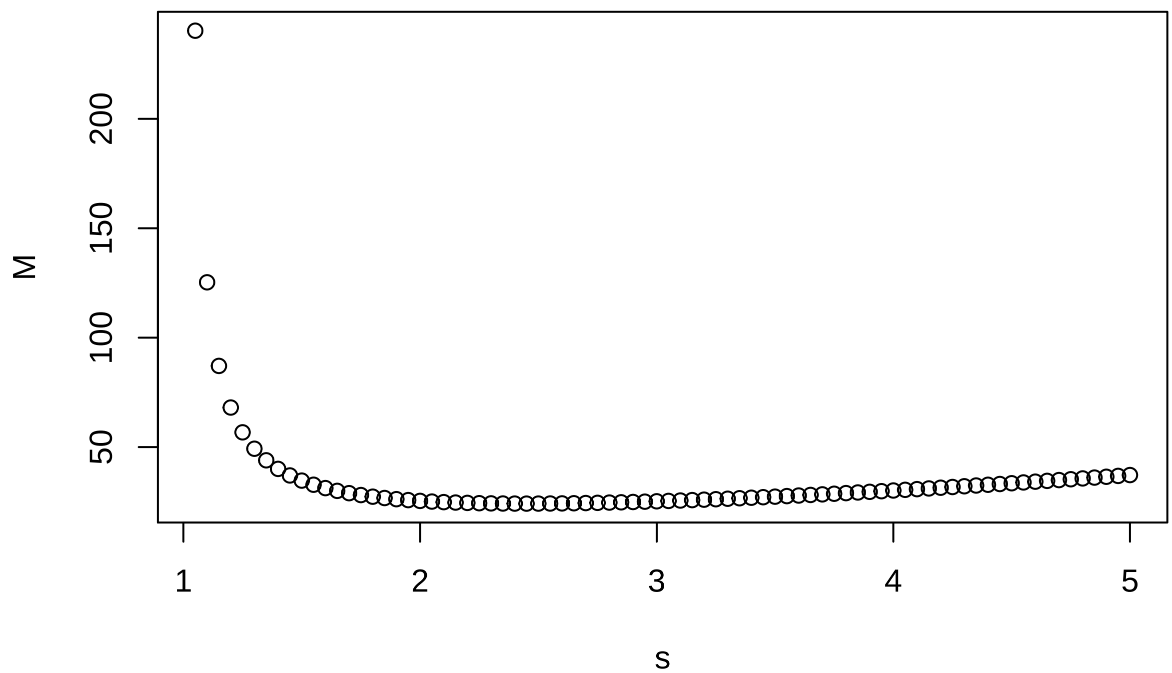
To clearly illistrate this result, let's have a plot.

```r
plot(p1$x, p1$prob, type='h', lwd=5,xlab= 'Values',
     ylab = 'Probabilities', main ='Approximation to a Hermite distribution 1')
```



Now we do the same thing for part (ii)

```r
Mplot(20, 0.1, 2)
```



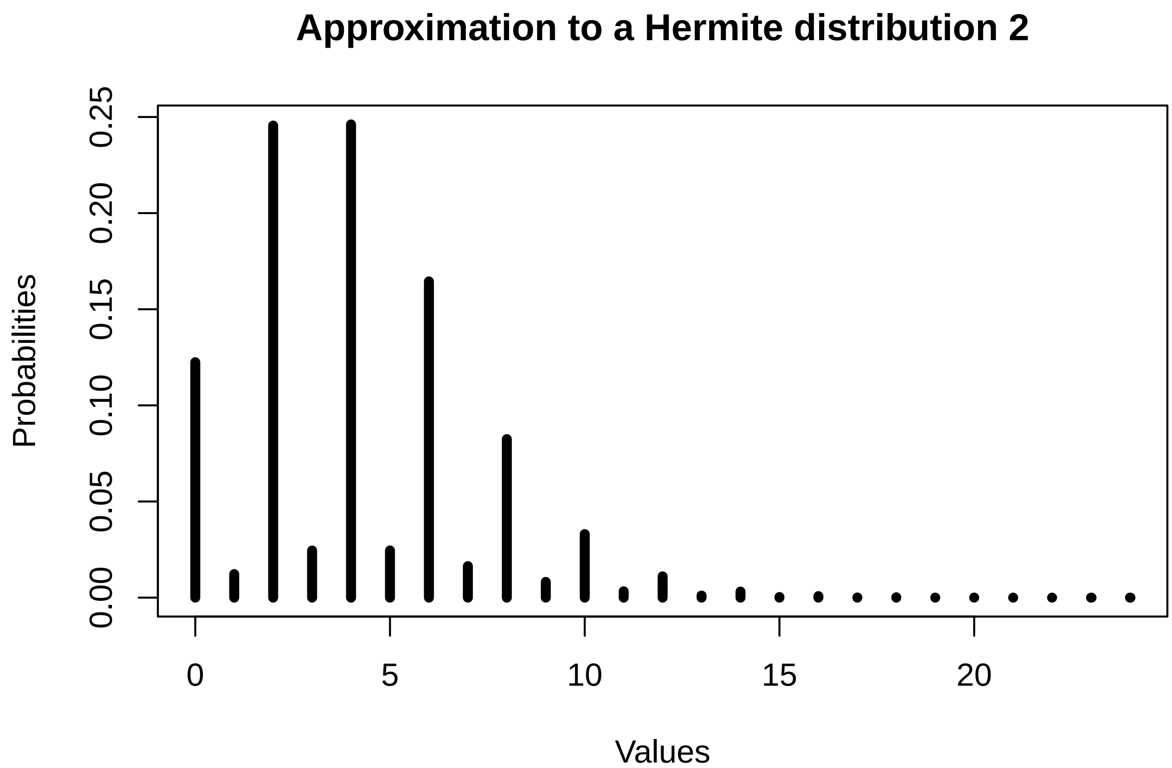

```r
M <- Mval(20, 0.1, 2)
M
```

```
## [1] 25
```

```r
p2 <- eval(M, 0.1, 2)
p2
```

```
## $x
##  [1]  0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22
## [24] 23 24
##
## $prob
##  [1] 1.224565e-01 1.224581e-02 2.455252e-01 2.451172e-02 2.461379e-01
##  [6] 2.453212e-02 1.645008e-01 1.636836e-02 8.245502e-02 8.190995e-03
## [11] 3.306392e-02 3.279125e-03 1.104863e-02 1.093951e-03 3.164566e-03
## [16] 3.128173e-04 7.930966e-04 7.826936e-05 1.766785e-04 1.740765e-05
## [21] 3.542274e-05 3.484422e-06 6.456336e-06 6.340574e-07 1.078698e-06
```

```r
plot(p2$x, p2$prob, type='h', lwd=5, xlab= 'Values',
     ylab = 'Probabilities', main ='Approximation to a Hermite distribution 2')
```


Approximation to a Hermite distribution 2
Probabilities
0.00
0.05
0.10
0.15
0.20
0.25
0
5
10
15
20
Values