University of Toronto

Coursework & Homework demo

YiFei Gu

1

a. With a definition of a transform of Z as $\hat{Z} = A_m Z A_n^T$, want to show that $Z = D_m^{-1} A_m^T \hat{Z} A_n D_n^{-1}$:

Note that given $\hat{Z} = A_m Z A_n^T$, then we have:

$$D_m^{-1} A_m^T \hat{Z} A_n D_n^{-1} = D_m^{-1} A_m^T A_m Z A_n^T A_n D_n^{-1}$$

And since $\{A_n\}$ families of matrices satisfying $A_n^T A_n = D_n$ where D_n diagonal matrix, then:

$$D_m^{-1} A_m^T A_m Z A_n^T A_n D_n^{-1} = D_m^{-1} D_m Z D_n D_n^{-1} = Z$$

b. & c. Hard threshold R function:

```
dctmat1 <- ifelse(abs(dctmat)>lambda,dctmat,0)
```

```
    > denoise_hard <- function(dctmat,lambda) {
+     if(missing(lambda)) lambda <- quantile(abs(dct),0.8)
+     # hard-thresholding
+     a <- dctmat[1,1]
+     dctmat1 <- ifelse(abs(dctmat)>lambda,dctmat,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
+ }
> b10 <- denoise_hard(boats, 10)
> image(b10, axes=F, col=grey(seq(0,1,length=256)))
> b25 <- denoise_hard(boats, 25)
> image(b25, axes=F, col=grey(seq(0,1,length=256)))
> b35 <- denoise_hard(boats, 35)
> image(b35, axes=F, col=grey(seq(0,1,length=256)))
```

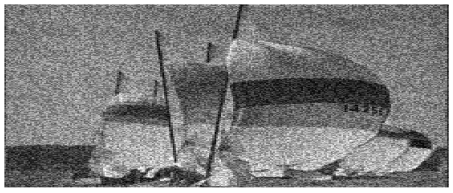


Figure 1: hard threshold at $\lambda = 10$

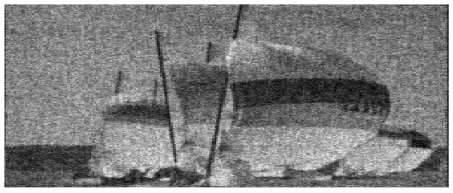


Figure 2: hard threshold at $\lambda = 25$

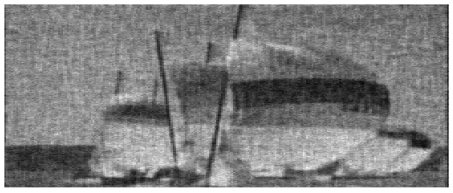


Figure 3: hard threshold at $\lambda = 35$

Soft threshold R funtion:

```
dctmat1 <- sign(dctmat)*pmax(abs(dctmat)-lambda, 0)
```

```
> denoise_soft <- function(dctmat,lambda) {
+
+     if(missing(lambda)) lambda <- quantile(abs(dct),0.8)
+     # soft-thresholding
+     a <- dctmat[1,1]
+     dctmat1 <- sign(dctmat)*pmax(abs(dctmat) - 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
```

```
+ }
> b10s <- denoise_soft(boats, 10)
> image(b10s, axes=F, col=grey(seq(0,1,length=256)))
> b25s <- denoise_soft(boats, 25)
> image(b25s, axes=F, col=grey(seq(0,1,length=256)))
> b35s <- denoise_soft(boats, 35)
> image(b35s, axes=F, col=grey(seq(0,1,length=256)))
```

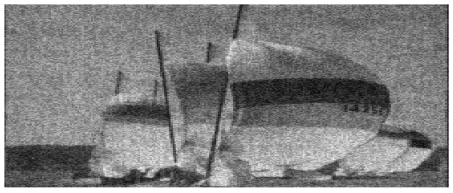


Figure 4: soft threshold at $\lambda = 10$

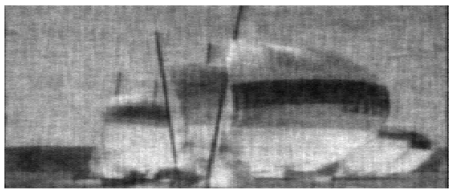


Figure 5: soft threshold at $\lambda = 25$

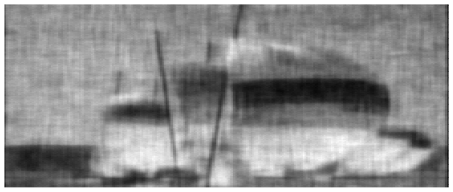


Figure 6: soft threshold at $\lambda = 35$

2

a. Given $X = U + 2V$, then:

$$g(s) = E(s^x) = E(s)^{u+2v} = E(s)^u E(s)^{2v}$$

Note that:

$$E(s^u) = \exp(-\lambda_u) \sum_{n=0}^{\infty} s^n \frac{\lambda_u^n}{n!} = \exp(-\lambda) \exp(\lambda_u s) = \exp(\lambda_u (s-1))$$

And:

$$E(s^{2v}) = \exp(-\lambda_v) \sum_{n=0}^{\infty} (s^2)^n \frac{\lambda_v^n}{n!} = \exp(-\lambda_v) \exp(s^2 \lambda_v) = \exp(\lambda_v (s^2 - 1))$$

Thus:

$$E(s^u)E(s^{2v}) = \exp[\lambda_u(s-1) + \lambda_v(s^2-1)], \text{ as required.}$$

b. For $s > 1$:

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

Take ln on both side gives:

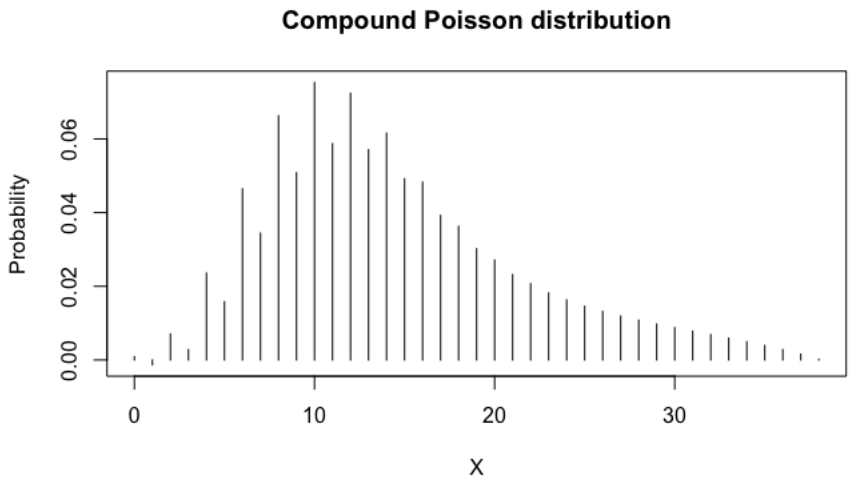
$$\begin{aligned}
\frac{[\lambda_u(s-1) + \lambda_v(s^2-1)]}{\ln(s^M)} &= \ln(\varepsilon) \\
[\lambda_u(s-1) + \lambda_v(s^2-1)] - \ln(s^M) &= \ln(\varepsilon) \\
[\lambda_u(s-1) + \lambda_v(s^2-1)] - \ln(\varepsilon) &= \ln(s^M) = M \ln(s) \\
\frac{[\lambda_u(s-1) + \lambda_v(s^2-1)] - \ln(\varepsilon)}{\ln(s)} &= M
\end{aligned}$$

Thus, for $P(X \geq M) \leq \varepsilon$, one can take

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

c. for $\lambda_u = 1$ and $\lambda_v = 5$:

```
> s = c(1001: 10000)/1000
> M <- min((1*(s - 1) + 5*(s^2 - 1) - log(10^-5))/log(s))
> M
[1] 39.64294
> s <- exp(-2*pi*1i*c(0:(M-1))/M)
> gs <- exp(1*(s - 1) + 5*(s^2 - 1))
> pf <- Re(fft(gs, inverse = T))/M
> r = c(0:(M-1))
> plot(r, pf, type = "h", main = "Compound Poisson distribution",
xlab = "X",ylab = "Probability")
```



for $\lambda_u = 0.1$ and $\lambda_v = 2$:

```
> s = c(1001: 10000)/1000
> M <- min((0.1*(s - 1) + 2*(s^2 - 1) - log(10^-5))/log(s))
> s <- exp(-2*pi*1i*c(0:(M-1))/M)
> gs <- exp(0.1*(s - 1) + 2*(s^2 - 1))
> pf <- Re(fft(gs, inverse = T))/M
> r = c(0:(M-1))
> plot(r, pf, type = "h", main = "Compound Poisson distribution",
xlab = "X",ylab = "Probability")
```