Take Home Final - Analysis of High Dimensional Data

1a. Let $f(x) = \|x\|_1$ with domain of $f = \{x : \|x\|_\infty \leq 1\}$. Note that

$$\begin{aligned}
[\text{prox}_f(x)]_i &= \arg\min_u [(\|u\|_1 + \frac{1}{2}\|u - x\|_2^2)]_i \\
&= \begin{cases} x_i - 1 & \text{if } x_i > 1 \\ 0 & \text{if } |x_i| \leq 1 \\ x_i + 1 & \text{if } x_i < -1 \end{cases}
\end{aligned}$$

Now for $\text{prox}_f(x)$ such that $\{u : \|u\|_\infty \leq 1\}$ we will have

$$[\text{prox}_f(x)]_i = \begin{cases} 1 & \text{if } x_i > 2 \\ x_i - 1 & \text{if } 1 < x_i \leq 2 \\ 0 & \text{if } |x_i| \leq 1 \\ x_i + 1 & \text{if } -2 \leq x_i < -1 \\ -1 & \text{if } x_i < -2 \end{cases}$$

1b. Let $f(x) = \max_{k=1,\ldots,n} x_k = \|x\|_\infty$. Thus

$$\begin{aligned}
\text{prox}_f(x) &= \arg\min_u (\|u\|_\infty + \frac{1}{2}\|u - x\|_2^2) \\
&= x - P_C(x)
\end{aligned}$$

where $C = \{x | \|x\|_1 \leq 1\}$. Note that

$$[P_C(x)]_i = = \begin{cases} x_i - \lambda & \text{if } x_i > \lambda \\ 0 & \text{if } |x_i| \leq \lambda \\ x_i + \lambda & \text{if } x_i < -\lambda \end{cases}$$

where $\lambda = 0$ if $\|x\|_1 \leq 1$ and otherwise is the solution of the equation

$$\sum_{i=1}^n \max\{|x_i| - \lambda, 0\} = 1.$$

1c. Let $f(x) = \|Ax - b\|_1$ where $AA' = D = \text{diag}(d_{11}, ..., d_{pp})$. Thus

$$\begin{aligned}\text{prox}_f(x) &= \arg\min_u(\|Au - b\|_1 + \frac{1}{2}\|u - x\|_2^2)\\ &= \arg\min_{y,u}(\|y\|_1 + \frac{1}{2}\|u - x\|_2^2) \text{ subject to } Au - b = y\end{aligned}$$

Note that as $AA' = D$,

$$Av = \begin{cases} d & \text{if } v \in \mathcal{C}(A') \\ 0 & \text{if } v \notin \mathcal{C}(A') \end{cases}$$

Thus for $u - x \in \mathcal{C}(A')$

$$\begin{aligned} u - x &= A'(AA')^{-1}A(u - x) \\ &= A'(AA')^{-1}(y + b - Ax) \\ \implies \text{prox}_f(y) &= \arg\min_y(\|y\|_1 + \frac{1}{2}\|A'(AA')^{-1}(y + b - Ax)\|_2^2) \\ &= \arg\min_y(\|y\|_1 + \frac{1}{2}\|D^{-\frac{1}{2}}(y + b - Ax)\|_2^2) \\ &= \arg\min_y(\sum_{i=1}^p |y_i| + \frac{1}{2d_{ii}}(y_i + b_i - a_i'x)^2) \\ &= \arg\min_y(\sum_{i=1}^p d_{ii}|y_i| + \frac{1}{2}(y_i + b_i - a_i'x)^2) \\ \implies [\text{prox}_f(y)]_i &= \begin{cases} -(b_i - a_i'x) - d_{ii} & \text{if } -(b_i - a_i'x) > d_{ii} \\ 0 & \text{if } |-(b_i - a_i'x)| \le d_{ii} \\ -(b_i - a_i'x) + d_{ii} & \text{if } -(b_i - a_i'x) < -d_{ii} \end{cases} \end{aligned}$$

1d. For this problem, for $\Sigma = PDP' \in S_{++}^n$, $\Sigma^{\frac{1}{2}} = PD^{\frac{1}{2}}P'$.
Let $f(X) = -\log\det(X), X \in S_+^n$ and domain$(f) = S_{++}^n$.

$$\begin{aligned}\text{prox}_f(X) &= \arg\min_U(-\log\det(U) + \frac{1}{2}\|U - X\|_F^2) \\ &= \arg\min_U(-\log\det(U) + \frac{1}{2}\text{trace}(U - X)(U - X)')\end{aligned}$$

Note that

$$\begin{aligned}
\frac{d}{dU}(\log\det(U)) &= \frac{1}{\det(U)}\frac{d}{dU}\det(U) \\
&= \frac{1}{\det(U)}\det(U)(U^{-1}) \\
&= U^{-1}
\end{aligned}$$

and

$$\frac{d}{dU}\text{trace}(U-X)(U-X)') = 2U-2X$$

which implies that

$$\begin{aligned}
\frac{d}{dU}(-\log\det(U)+\frac{1}{2}\text{trace}(U-X)(U-X)') = -U^{-1}+U-X = 0 \\
\iff -I+U^2-UX = 0 \\
\iff U^2-UX = I \\
\iff U^2-2U(\frac{1}{2}X)+(\frac{1}{2}X)^2 = I+(\frac{1}{2}X)^2 \\
\iff (U-(\frac{1}{2}X))^2 = I+(\frac{1}{2}X)^2 \\
\iff U-(\frac{1}{2}X) = (I+(\frac{1}{2}X)^2)^{\frac{1}{2}} \\
\iff U = (I+(\frac{1}{2}X)^2)^{\frac{1}{2}}+(\frac{1}{2}X) \succ 0.
\end{aligned}$$

Thus

$$\text{prox}_f(X) = (I+(\frac{1}{2}X)^2)^{\frac{1}{2}}+(\frac{1}{2}X).$$

2 We want to solve the following problem:

Primal: $\min(x_1-x_2)$

subject to $x \in X = \{x : x_1 \geq 0, x_2 \geq 0\}, x_1+1 \leq 0, 1-x_1-x_2 \leq 0$

Now the Lagrangian is

$$\begin{aligned}
\mathcal{L}(x,\lambda) &= (x_1 - x_2) + \lambda_1(-x_1) + \lambda_2(-x_2) \\
&\quad + \lambda_3(x_1+1) + \lambda_4(1 - x_1 - x_2) \\
&= x_1(1 - \lambda_1 + \lambda_3 - \lambda_4) + x_2(-1 - \lambda_2 - \lambda_4) + \lambda_3 + \lambda_4 \\
\implies g(\lambda_1,\lambda_2,\lambda_3,\lambda_4) &= \inf_x \mathcal{L}(x,\lambda) \\
&= \begin{cases} -\infty & \text{if } 1 - \lambda_1 + \lambda_3 - \lambda_4 \neq 0 \text{ or } -1 - \lambda_2 - \lambda_4 \neq 0 \\ \lambda_3 + \lambda_4 & \text{else} \end{cases}
\end{aligned}$$

Thus the dual problem is

$$\begin{aligned}
\textbf{Dual:} \max_{\lambda_i \geq 0} &\ \lambda_3 + \lambda_4 \\
&\text{subject to } 1 - \lambda_1 + \lambda_3 - \lambda_4 = 0 \\
&\text{and } -1 - \lambda_2 - \lambda_4 = 0
\end{aligned}$$

The constraints have infinitely many solutions. Hence this is clearly maximized when $\lambda_i = \infty, i = 3,4$ which implies that the dual is unbounded.

3a. Let $A_j \in \mathbb{R}^{m\times n}, y_j \in \mathbb{R}$. We want to solve the following problem:

$$\begin{aligned}
\textbf{Primal:} \min_{X\in\mathbb{R}^{m\times n}} &\sum_{j=1}^{J}(y_j - \text{trace}(X'A_j))^2 \\
&\text{subject to } \|X\|_* \leq t
\end{aligned}$$

or equivalently,

$$\min_{X\in\mathbb{R}^{m\times n}} \sum_{j=1}^{J}(y_j - \text{trace}(X'A_j))^2 + \lambda\|X\|_*$$

For our simulations we used, $m = 10, n = 5$ and $J = 5$. In addition, A_j are matrices with elements randomly generated from $\mathcal{U}(-5,5)$. To generate the true X we generated a matrix with elements from $\mathcal{U}(-5,5)$ took the SVD as $U\Sigma V'$ and set $X = UV'$. We then calculated $y_j = \text{trace}(X'A_j) + \epsilon_j$ where $\epsilon_j \sim \mathcal{N}(0,1)$. Note that $\frac{d}{dX}\sum_{j=1}^{J}(y_j - \text{trace}(X'A_j))^2 = -2\sum_{j=1}^{J}(y_j -$

$\text{trace}(X'A_j))A_j$. Thus a proximal gradient method would be

$$X^{k+1} = X^k + t(2\sum_{j=1}^{J}(y_j - \text{trace}((X^k)'A_j))A_j) \tag{1}$$

$$X^{k+1} = S_{t\lambda}(X^{k+1}) \tag{2}$$

where t is the stepsize and $S_{t\lambda}(X^{k+1}) = US_{t\lambda}(\Sigma)V'$. Here,

$$\begin{aligned} X &= U\Sigma V' \\ &= U\text{diag}(\sigma_1, ..., \sigma_p)V' \end{aligned}$$

is the SVD of X and $S_{t\lambda}(\Sigma) = \text{diag}(S_{t\lambda}(\sigma_1), ..., S_{t\lambda}(\sigma_p))$ is the soft thresholding operator applied elementwise to the diagonal of Σ. Using a cross-validation procedure we find that $\lambda \approx 570$ minimizes the objective function as shown in Figure 1. Doing 10 replications using this value of λ converges in about $200 - 300$ iterations and gives us an average Frobenius norm loss of

$$F = \sum_{r=1}^{10} \frac{\|\hat{X}_r - X_r\|_F}{\|X_r\|_F} = 0.9858716.$$

We define relative error as

$$RE = \frac{\|\hat{X}^k - X_{true}\|_F}{\|X_{true}\|_F}$$

This is shown in Figure 2 for $\lambda = 570$.

In general as m, n and J increase we need to reduce the stepsize to achieve convergence. For $m = 50, n = 30, J = 100$ and $\lambda = 100$ with a stepsize of 10^{-7} we achieved convergence in $200 - 300$ iterations with an average Frobenius norm loss of

$$F = \sum_{r=1}^{10} \frac{\|\hat{X}_r - X_r\|_F}{\|X_r\|_F} = 1.001247.$$

4. We propose two methods to solve this problem. We want to solve the following problem for $i = 1, ..., p$:

$$\begin{aligned} \textbf{Method 1:} \min &\|\beta^1\|_1 + \|\beta^2\|_1 + \|\beta^2 - \beta^1\|_1 \\ &\text{subject to } \|S\beta^1 - e_i\|_\infty \le t_1, \|S\beta^2 - e_i\|_\infty \le t_2 \end{aligned}$$

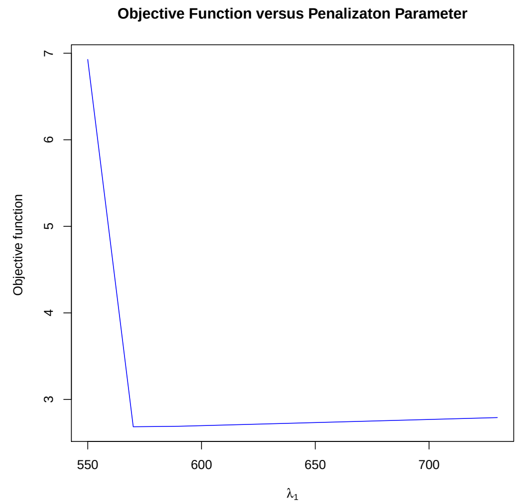

Figure 1: Value of objective function for predicted X_λ versus λ for trace norm constrained optimization problem

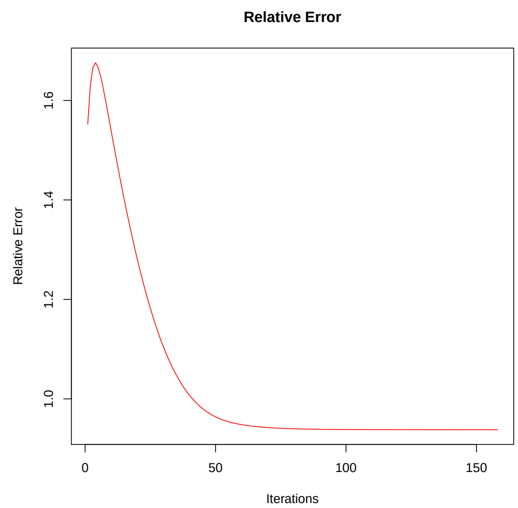

Figure 2: Relative Error plot for trace norm constrained optimization problem

Note that this is equivalent to

$$\begin{aligned}
&\min \|\beta^1\|_1 + \|\beta^2\|_1 + \|\beta^2 - \beta^1\|_1 \\
&\text{subject to } -t_1 1 \le S\beta^1 - e_i \le t_1 1 \\
&\text{and } -t_2 1 \le S\beta^2 - e_i \le t_2 1
\end{aligned}$$

Now reparametrize by letting $u_j^1 = |\beta_j^1|_1$, $u_j^2 = |\beta_j^2|_1$ and $z_j = |\beta_j^2 - \beta_j^1|$. Then the problem is

$$\begin{aligned}
&\min \sum_{j=1}^{p} u_j^1 + \sum_{j=1}^{p} u_j^2 + \sum_{j=1}^{p} z_j \\
&\text{subject to } -\beta^1 \le u^1 \le \beta^1, -\beta^2 \le u^2 \le \beta^2 \\
&\text{and } \beta^1 - \beta^2 \le z \le \beta^2 - \beta^1 \\
&\text{and } -t_1 1 \le S\beta^1 - e_i \le t_1 1 \\
&\text{and } -t_2 1 \le S\beta^2 - e_i \le t_2 1
\end{aligned}$$

Hence we can write the above problem as a linear programming problem:

$$\begin{aligned}
&\min c^T x \\
&\text{subject to } Ax \le b \\
&\text{and } x \ge 0
\end{aligned}$$

where

$$x = \begin{pmatrix} u^1 \\ u^2 \\ \beta^1 + u^1 \\ \beta^2 + u^2 \\ z \end{pmatrix},$$

$$c^T = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 \end{pmatrix},$$

$$A = \begin{pmatrix} -2I & 0 & I & 0 & 0 \\ 0 & 0 & -I & 0 & 0 \\ 0 & -2I & 0 & I & 0 \\ 0 & 0 & 0 & -I & 0 \\ -S & 0 & S & 0 & 0 \\ S & 0 & -S & 0 & 0 \\ 0 & -S & 0 & S & 0 \\ 0 & S & 0 & -S & 0 \\ I & -I & -I & I & -I \\ I & -I & -I & I & I \end{pmatrix},$$

and

$$b^T = \begin{pmatrix} 0 & 0 & 0 & 0 & t_1 1 + e_i & t_1 1 - e_i & t_2 1 + e_i & t_2 1 - e_i & 0 & 0 \end{pmatrix}.$$

For the second method we want to solve the following problem for $i = 1, ..., p$:

$$\begin{aligned} \textbf{Method 2:} \min & \|\beta^1\|_1 + \|\beta^2\|_1 \\ \text{subject to } & \|S\beta^1 - e_i\|_\infty \le t_1, \|S\beta^2 - e_i\|_\infty \le t_2 \\ \text{and } & \|\beta^2 - \beta^1\|_\infty \le t_3 \end{aligned}$$

Note that this is equivalent to

$$\begin{aligned} & \min \|\beta^1\|_1 + \|\beta^2\|_1 \\ & \text{subject to } -t_1 1 \le S\beta^1 - e_i \le t_1 1 \\ & \text{and } -t_2 1 \le S\beta^2 - e_i \le t_2 1 \\ & \text{and } -t_3 1 \le \beta^2 - \beta^1 \le t_3 1 \end{aligned}$$

Now reparametrize by letting $u^1 = |\beta^1|_1$ and $u^2 = |\beta^2|_1$. Then the problem is

$$\begin{aligned} & \min \sum_{j=1}^p u_j^1 + \sum_{j=1}^p u_j^2 \\ & \text{subject to } -\beta^1 \le u^1 \le \beta^1, -\beta^2 \le u^2 \le \beta^2 \\ & \text{and } -t_1 1 \le S\beta^1 - e_i \le t_1 1 \\ & \text{and } -t_2 1 \le S\beta^2 - e_i \le t_2 1 \\ & \text{and } -t_3 1 \le \beta^2 - \beta^1 \le t_3 1 \end{aligned}$$

or equivalently,

$$\begin{aligned}
&\min \sum_{j=1}^{p} u_j^1 + \sum_{j=1}^{p} u_j^2 \\
&\text{subject to } 0 \le 2u^1 + (\beta^1 - u^1) \le 2\beta^1, 0 \le 2u^2 + (\beta^2 - u^2) \le 2\beta^2 \\
&\text{ and } S(\beta^1 + u^1) - Su^1 \le e_i + t_1 1 \\
&\text{ and } -(S(\beta^1 + u^1) - Su^1) \le t_1 1 - e_i \\
&\text{ and } S(\beta^2 + u^2) - Su^2 \le e_i + t_2 1 \\
&\text{ and } -(S(\beta^2 + u^2) - Su^2) \le t_2 1 - e_i \\
&\text{ and } (\beta^1 - u^1) - (\beta^2 - u^2) + u^1 - u^2 \le -t_3 1 \\
&\text{ and } (\beta^2 - u^2) - (\beta^1 - u^1) - u^1 + u^2 \le t_3 1
\end{aligned}$$

Hence we can write the above problem as a linear programming problem:

$$\begin{aligned}
&\min c^T x \\
&\text{subject to } Ax \le b \\
&\text{and } x \ge 0
\end{aligned}$$

where

$$x = \begin{pmatrix} u^1 \\ u^2 \\ \beta^1 + u^1 \\ \beta^2 + u^2 \end{pmatrix},$$

$$c^T = \begin{pmatrix} 1 & 1 & 0 & 0 \end{pmatrix},$$

$$A = \begin{pmatrix} -2I & 0 & I & 0 \\ 0 & 0 & -I & 0 \\ 0 & -2I & 0 & I \\ 0 & 0 & 0 & -I \\ -S & 0 & S & 0 \\ S & 0 & -S & 0 \\ 0 & -S & 0 & S \\ 0 & S & 0 & -S \\ I & -I & -I & I \\ -I & I & I & -I \end{pmatrix},$$

and

$$b^T = \begin{pmatrix} 0 & 0 & 0 & 0 & t_1 1 + e_i & t_1 1 - e_i & t_2 1 + e_i & t_2 1 - e_i & t_3 1 & -t_3 1 \end{pmatrix}.$$

For both methods set $\hat{\beta}^1 = (\hat{\beta}^1 + \hat{u}^1) - \hat{u}^1$ and $\hat{\beta}^2 = (\hat{\beta}^2 + \hat{u}^2) - \hat{u}^2$ and let the solutions to the i-th problem be the i-th columns of $\hat{\Omega}^1$ and $\hat{\Omega}^2$. Finally we symmetrized $\hat{\Omega}^k$ by taking $\hat{\Omega}^k_{ij} = \min(\hat{\Omega}^k_{ij}, \hat{\Omega}^k_{ji})$ as reccomended by Cai et. al (2011). For method 1, we tried values of $t_1 = t_2$ from a grid between 0.05 and 0.95 and settled on 0.1 as that minimized the distance between the false positive and false negative rates for the values we tried. We tried a similar approach for method 2, except that we first searched for the optimal $t1 = t2$ and then for $t3$.

For both Ω^1 and Ω^2 we calculate the following statistics.

$$\begin{aligned} F(\hat{\Omega}) &= \frac{1}{20}\sum_{k=1}^{20} \frac{\|\Omega^{(k)} - \hat{\Omega^{(k)}}\|_F^2}{\|\Omega^{(k)}\|_F^2} \\ F_{mle}(\hat{\Omega}) &= \frac{1}{20}\sum_{k=1}^{20} \frac{\|\Omega^{(k)}_{mle} - \hat{\Omega^{(k)}}\|_F^2}{\|\Omega^{(k)}\|_F^2} \\ FP(\hat{\Omega}) &= \frac{1}{20}\sum_{k=1}^{20} \frac{\sum_{i,j} I(\omega^{(k)}_{ij} = 0, \hat{\omega}^{(k)}_{ij} \neq 0)}{\sum_{i,j} I(\omega^{(k)}_{ij} = 0)} \\ FN(\hat{\Omega}) &= \frac{1}{20}\sum_{k=1}^{20} \frac{\sum_{i,j} I(\omega^{(k)}_{ij} \neq 0, \hat{\omega}^{(k)}_{ij} = 0)}{\sum_{i,j} I(\omega^{(k)}_{ij} \neq 0)} \end{aligned}$$

Note that F_{mle} was calculated by finding the corresponding graph first and then solving the following problem:

$$\begin{aligned} &\max_{\Omega \succ 0} \log|\Omega| - tr(\Omega S) \\ &\text{subject to } (i,j) \notin E \implies \omega_{ij} = 0. \end{aligned} \tag{3}$$

In the tables below we report the average for Ω^1 and Ω^2 for each case.

	Chain($t_i = .1$)	Nearest Neighbor($t_i = .1$)	Scale-free($t_i = .1$)
F	0.05329148	0.05972657	0.06793882
F_{mle}	0.1681349	0.1722356	0.1898157
FP	0.2646804	0.2794095	0.2772885
FN	0.7228644	0.6645634	0.695503

Table 1: A comparison of the Dantzig Selector Joint estimation method for Chain, Nearest Neighbor network and Scale-free network graphs with $\rho = \frac{1}{4}$ using Method 1. Estimation for all types of graphs is fairly similar as the values of F, FN and FP indicate. Method 1 is better if we are more interested in FP than FN.

	Chain $t_1/t_2 = .25, t_3 = .005$	Nearest Neighbor $t_1/t_2 = .25, t_3 = .025$	Scale-free $t_1/t_2 = .15, t_3 = .02$
F	0.04923219	0.06669798	0.03500943
F_{mle}	0.165453	0.1761309	0.1970076
FP	0.6177833	0.6310923	0.73976
FN	0.3400952	0.3453805	0.279942

Table 2: A comparison of the Dantzig Selector Joint estimation method for Chain, Nearest Neighbor network and Scale-free network graphs with $\rho = \frac{1}{4}$ using Method 2. Estimation for all types of graphs is fairly similar as the values of F, FN and FP indicate. Method 2 is better if we are more interested in FN than FP.

Appendix 1: R code for problem 3

```
proxmialgradient3 <- function
(y,A,lambda,stepsize,initialX,maxiter,tol){
oldX = initialX
iter = 1
eps = 1

while(eps>tol&iter<maxiter){
(cat('Iter:', iter, 'Eps:', eps, '\n'))
(newX = oldX - stepsize*grad(y,oldX,A))
(svdnewX = svd(newX))
(threshd = sign(svdnewX$d)*max(abs(svdnewX$d)-stepsize*lambda,0))
(newX = svdnewX$u%*%diag(threshd)%*%t(svdnewX$v))
(eps = norm(newX-oldX,type=c('F'))/norm(oldX,type=c('F')))
(iter = iter + 1)
(oldX = newX)
}
return(list(X = newX, iter=iter, eps = eps))
}
```

Appendix 2: R code for problem 4

```
###method 2 for scale-free network
scaleestimatem1 <- function(p,n,t1,t2,t3){
Scale = scalefree2(p)
Sigma = round(Scale$sigma1,10)
Omega = round(Scale$omega1,10)
mu = rep(0,p)
Y1 = mvrnorm(n, mu, Sigma, tol = 1e-6,
empirical = FALSE, EISPACK = FALSE)
S1 = (1/n)*t(Y1)%*%Y1
Sigma2 = round(Scale$sigma2,10)
Omega2 = round(Scale$omega2,10)
Y2 = mvrnorm(n, mu, Sigma2, tol = 1e-6,
empirical = FALSE, EISPACK = FALSE)
S2 = (1/n)*t(Y2)%*%Y2
A = makeA(S1,S2,p)
c1 = matrix(1,ncol = 1,nrow = 2*p)
c2 = matrix(0,ncol = 1,nrow = 2*p)
(c = rbind(c1,c2))
(f.dir = rep("<=",dim(A)[1]))
omegahat1 = matrix(0,nrow=p,ncol=p)
omegahat2 = matrix(0,nrow=p,ncol=p)
for(i in 1:p){
    #i=1
    (b = makeb(p,i,t1,t2,t3))
    linearprog = lp("min", t(c), A, f.dir, b)
    x = linearprog$solution
    u1 = x[1:p]
    u2 = x[(p+1):(2*p)]
    b1pu1 = x[(2*p+1):(3*p)]
    b2pu2 = x[(3*p+1):(4*p)]
    omegahat1[,i] = b1pu1-u1
    omegahat2[,i] = b2pu2-u2
}
omegahat1 =  makeSymmetricMin(omegahat1)
omegahat2 =  makeSymmetricMin(omegahat2)
```

```
return(list(Omega1 = Omega,Omega2 = Omega2,
omegahat1 = omegahat1, omegahat2 = omegahat2,
S1 = S1, S2= S2))
}

###method 1 for scale-free network
scalestimate <- function(p,n,t1,t2){
Scale = scalefree2(p)
Sigma = round(Scale$sigma1,10)
Omega = round(Scale$omega1,10)
mu = rep(0,p)
Y1 = mvrnorm(n, mu, Sigma, tol = 1e-6,
empirical = FALSE, EISPACK = FALSE)
S1 = (1/n)*t(Y1)%*%Y1
Sigma2 = round(Scale$sigma2,10)
Omega2 = round(Scale$omega2,10)
Y2 = mvrnorm(n, mu, Sigma2, tol = 1e-6,
empirical = FALSE, EISPACK = FALSE)
S2 = (1/n)*t(Y2)%*%Y2
A = makeA(S1,S2,p)
c1 = matrix(1,ncol = 1,nrow = p)
c2 = matrix(1,ncol = 1,nrow = p)
c3 = matrix(0,ncol = 1,nrow = p)
c4 = matrix(0,ncol = 1,nrow = p)
c5 = matrix(1,ncol = 1,nrow = p)
(c = rbind(c1,c2,c3,c4,c5))
(f.dir = rep("<=",dim(A)[1]))
omegahat1 = matrix(0,nrow=p,ncol=p)
omegahat2 = matrix(0,nrow=p,ncol=p)
for(i in 1:p){
    #i=1
    (b = makeb(p,i,t1,t2))
    linearprog = lp("min", t(c), A, f.dir, b)
    x = linearprog$solution
    u1 = x[1:p]
    u2 = x[(p+1):(2*p)]
    b1pu1 = x[(2*p+1):(3*p)]
    b2pu2 = x[(3*p+1):(4*p)]
```

```
    omegahat1[,i] = b1pu1-u1
    omegahat2[,i] = b2pu2-u2
}
omegahat1 =  makeSymmetricMin(omegahat1)
omegahat2 =  makeSymmetricMin(omegahat2)
return(list(Omega1 = Omega,Omega2 = Omega2,
omegahat1 = omegahat1,
omegahat2 = omegahat2,S1 = S1, S2= S2))
}
```