```r
sylvester=function(A,B,C){
  I_A=diag(1,dim(A)[1],dim(A)[2])
  I_B=diag(1,dim(B)[1],dim(B)[2])
  M=kronecker(I_B,A) + kronecker(t(B),I_A)
  L=matrix(c(C),dim(A)[1]*dim(B)[1],1)
  cx=ginv(M) %*% L
  X=matrix(cx,dim(A)[1],dim(B)[1])
  return(X)
}

L_num=function(n){
  library(tidyr)
  L=matrix(0,n,n)
  for(i in 1:n){
    for(j in 1:n){
      L[i,j] = paste(i,',',j)
    }
  }
  l=c(L[upper.tri(L)])
  ll=data.frame(l)

  LL=ll %>% separate(l,c('l1','l2'),sep=',')
  LL=data.frame(l1=as.numeric(LL[,'l1']),l2=as.numeric(LL[,'l2']))
  return(LL)
}

update_A = function(X, nu1, nu2, lambda_1l, lambda_2k, v, z){
  library(MASS)
  n=dim(X)[1]; p=dim(X)[2]
  eplison_n=L_num(n)
  eplison_p=L_num(p)

  En=matrix(0,n,n)
  Ep=matrix(0,p,p)

  for(i in 1:dim(eplison_n)[1]){
    l1=eplison_n[i,'l1']
    l2=eplison_n[i,'l2']
    el1=matrix(rep(0,n),n,1)
    el2=matrix(rep(0,n),n,1)
    el1[l1,1]=1;el2[l2,1]=1
    En=En+(el1-el2) %*% t(el1-el2)
  }

  for(i in 1:dim(eplison_p)[1]){
    l1=eplison_p[n,'l1']
    l2=eplison_p[n,'l2']
    el1=matrix(rep(0,p),p,1)
    el2=matrix(rep(0,p),p,1)
    el1[l1,1]=1;el2[l2,1]=1
    Ep=Ep+(el1-el2) %*% t(el1-el2)
  }
```

```
  M = diag(1,n,n) + nu1 * En

  N = nu2 * Ep

  C2 = matrix(0,n,p)

  for(i in 1:dim(eplison_n)[1]){
    l1=eplison_n[i,'l1']
    l2=eplison_n[i,'l2']
    el1=matrix(rep(0,n),n,1)
    el2=matrix(rep(0,n),n,1)
    el1[l1,1]=1;el2[l2,1]=1
    C2 = C2 + (el1-el2) %*% t(lambda_1[,i] + nu1 * v[,i])
  }

  C3 = matrix(0,n,p)

  for(i in 1:dim(eplison_p)[1]){
    l1=eplison_p[i,'l1']
    l2=eplison_p[i,'l2']
    el1=matrix(rep(0,p),p,1)
    el2=matrix(rep(0,p),p,1)
    el1[l1,1]=1;el2[l2,1]=1
    C3 = C3+(lambda_2[,i] + nu2 * z[,i]) %*% t(el1-el2)
  }

  C = X +  C2 + C3

  A = sylvester(M,N,C)

  return(A)
}

update_lambda=function(X, A, nu1, nu2,v, z){
  n = dim(X)[1]; p = dim(X)[2]
  eplison_p = L_num(p)
  eplison_n = L_num(n)
  for(i in 1:dim(eplison_n)[1]){
    l1=eplison_n[i,'l1']
    l2=eplison_n[i,'l2']
    a_l1 = matrix(A[l1,],p,1)
    a_l2 = matrix(A[l2,],p,1)
    lambda_1[,i] = lambda_1[,i] + nu1 * (v[,i] - a_l1 + a_l2)
  }
  for(i in 1:dim(eplison_p)[1]){
    l1=eplison_n[i,'l1']
    l2=eplison_n[i,'l2']
    a_k1 = matrix(A[,l1],n,1)
    a_k2 = matrix(A[,l2],n,1)
    lambda_2[,i] = lambda_2[,i] + nu2 * (z[,i] - a_k1 + a_k2)
  }
  return(lambda=list(lambda_1,lambda_2))
}
```

```
prox = function(v,sigma,n=2){
  if(n == 2){
    return(max(1-sigma/sum(v*v),0) %*% v)
  }
}

update_vz = function(X, A, lambda_1, lambda_2, gamma_1, gamma_2, nu1, nu2){

  n=dim(X)[1]; p=dim(X)[2]
  eplison_n=L_num(n)
  eplison_p=L_num(p)

  w_l = rep(0,dim(eplison_n)[1])
  u_k = rep(0,dim(eplison_p)[1])

  for(i in 1:dim(eplison_n)[1]){
    l1=eplison_n[i,'l1']
    l2=eplison_n[i,'l2']
    w_l[i] = exp(-0.5 * (t(X[l1,] - X[l2,]) %*% (X[l1,] - X[l2,])))
  }

  for(i in 1:dim(eplison_p)[1]){
    l1=eplison_n[i,'l1']
    l2=eplison_n[i,'l2']
    u_k[i] = exp(-0.5 * (t(X[,l1] - X[,l2]) %*% (X[,l1] - X[,l2])))
  }

  w_l = w_l / sum(w_l) * 1/sqrt(p)
  u_k = u_k / sum(u_k) * 1/sqrt(n)

  for(i in 1:dim(eplison_n)[1]){
    l1=eplison_n[i,'l1']
    l2=eplison_n[i,'l2']
    a_l1 = A[l1,]; a_l2 = A[l2,]
    v_temp = a_l1 - a_l2 + 1/nu1 * lambda_1[,i]
    sigma_1l = gamma_1 * w_l[i]/nu1
    v[,i] = prox(v_temp,sigma_1l)
  }
  for(i in 1:dim(eplison_p)[1]){
    l1=eplison_p[i,'l1']
    l2=eplison_p[i,'l2']
    a_l1 = A[,l1]; a_l2 = A[,l2]
    v_temp = a_l1 - a_l2 + 1/nu2 * lambda_2[,i]
    #u_k = exp(-0.5 * (t(X[,l1] - X[,l2]) %*% (X[,l1] - X[,l2])))
    sigma_2k = gamma_2 * u_k[i]/nu2
    z[,i] = prox(v_temp,sigma_2k)
  }

  return(list(v = v, z = z))
}

Bi_ADMM = function(X, nu1, nu2, lambda_1, lambda_2, v, z, gamma_1, gamma_2){
  A = 0;
```

```
  for(iter in 1: 100){
    A_old = A
    A = update_A(X, nu1, nu2, lambda_1, lambda_2, v, z)

    v_old = v; z_old = z
    vz = update_vz(X, A, lambda_1, lambda_2, gamma_1, gamma_2, nu1, nu2)
    v = vz[[1]]
    z = vz[[2]]

    lambda = update_lambda(X, A, nu1, nu2,v, z)
    lambda_1_old = lambda_1; lambda_2_old = lambda_2
    lambda_1 = lambda[[1]]
    lambda_2 = lambda[[2]]

    if(sum(abs(A - A_old)) < e &
       sum(abs(v - v_old)) < e &
       sum(abs(z - z_old)) < e &
       sum(abs(lambda_1 - lambda_1_old)) < e &
       sum(abs(lambda_2 - lambda_2_old)) < e){
      return(list(A = A))
      break
    }
  }
  if(iter == 100){print('not converage within 100 iters')}
}

Bi_ADMM = function(X, nu1, nu2, lambda_1, lambda_2, v, z, gamma_1, gamma_2, u_k){
   A = 0;
   for(iter in 1){
     print(iter)
     A_old = A
     A = update_A(X, nu1, nu2, lambda_1, lambda_2, v, z)

     v_old = v; z_old = z
     vz = update_vz(X, A, lambda_1, lambda_2, gamma_1, gamma_2, nu1, nu2)
     v = vz[[1]]
     z = vz[[2]]

     lambda = update_lambda(X, A, nu1, nu2,v, z)
     lambda_1_old = lambda_1; lambda_2_old = lambda_2
     lambda_1 = lambda[[1]]
     lambda_2 = lambda[[2]]

     if(sum(abs(A - A_old)) < e &
        sum(abs(v - v_old)) < e &
        sum(abs(z - z_old)) < e &
        sum(abs(lambda_1 - lambda_1_old)) < e &
        sum(abs(lambda_2 - lambda_2_old)) < e){
       return(list(A = A))
       break
     }
   }
```

```
  if(iter == 100){print('not converage within 100 iters')}
}



A = B =matrix(1:9, 3,3)
X = matrix(9:1 , 3, 3)
C = A%*% X + X%*%B

library(Matrix)
A1 = Schur(A)
Q1 = A1$Q; R1 = A1$T

A2 = Schur(B)
Q2 = A2$Q; R2 = A2$T

C = -C
C = t(Q1) %*% C %*% Q2
Rsq = R1 * R1
I = diag(dim(A)[1])

b = - C[,1]
X = matrix(0, dim(A)[1], dim(B)[1])
X[,1] = (solve(R1 +  R2[1,1] * I) %*% b)

n = dim(A)[1]
j=2
while(j < n){
  if(j<n & abs(R2[j+1,j]<0.01)){
      left = R1 + R2[j,j] * I
      if(j == 2){
        right = - C[,j] - X[, 1:(j-1)] * R2[1: (j-1), j]
      }else{
        right = - C[,j] - X[, 1:(j-1)] %*% R2[1: (j-1), j]
      }
      j = j+1
      X[,j] = qr.solve(left) %*% right
  }else{
    r11 = R2[j,j];r12 = R2[j, j+1]
    r21 = R2[j+1,j]; r22 = R2[j+1, j+1]
    b = -C[,j:(j+1)] - X[,1: (j-1)] %*% R2[1:(j-1), j:(j+1)]

    b = rbind(R1 %*% b[,1] + r22 * b[,1] - r21 * b[,2],R1 %*% b[,2] + r11 * b[,2] - r12 * bp,1)
    X[, j:(j+1)] = qr.solve(Rsq + (r11 + r22)%*% R1 + (r11 * r22 - r12 * r21) * I) %*% b
    j = j+2
  }
}
```