MATLAB Script of the Generalized Water Filling (GWF) Algorithm

Yuan Yu and Yingbo Hua

Posted on January 18, 2010

Reference: Y. Yu and Y. Hua, “Power allocation for a MIMO relay system with multiple-antenna users,” IEEE Transactions on Signal Processing, to appear.

```
% Function Inputs:
%     J:   The total number of power constraints.
%     H:   M x N channel matrix.
%     B:   M x N x J matrix for the power constraints, B(:,:,k) is the M x N weight matrix for
%            the k-th power constraint.
%     P:   J x 1 vector of the values for the power constraints.
% Function Outputs:
%      capacity:    log|I + HQH'|.
%      Q:      The N x N covariance matrix.
%      mu:     J x 1 vector of the Lagrangian multipliers.
% With the understanding of the above parameters, this Matlab code is ready to run to
%      solve the problem in (1). If more information about other notations used in this code
%      is needed, please contact the authors.
function [capacity, Q, mu] = op_2c_barrier(H, B, P)
[M,N,J] = size(B);
mu = 1e-2*rand(1,J);  % initialization of \mu.
stop_cre=1e-4;
t_b =2;
del = 2;
epsi = 1e-4;
out_count=0;
MM=[];
capacity=[];
while J/t_b>stop_cre
   out_count=out_count+1;
   t_b = t_b*del;
   flag = 1;
   if min(real(mu))<0
      [mi_mu ind]=min(real(mu));
      mu(ind)=0;
   else
   end
   while flag==1
      R = zeros(N,N);
      if min(real(mu))<1e-5
         mu=mu+0.0002*rand(1,J);
      else
```

```
end
for j = 1:J
   R = R+mu(j)*B(:,:,j)'*B(:,:,j);
end
a=svd(R);
if min(a)<1e-5
   R=R+0.001*diag(rand(N,1));  % A random perturbation is added here to ensure a better
                               % numerical property.
else
end
[E Fi] = svd(R);
R_inv = E*Fi^(-0.5)*E';
temp_IRH=R_inv*H'*H*R_inv;
[V D] = svd(temp_IRH);
clear temp_IRH
N_D = sum(diag(D>1));
temp_D=diag(D);
temp_D(N_D+1:N)=1;
Q = R_inv*V(:,1:N_D)*(eye(N_D)-inv(diag(temp_D(1:N_D))))*V(:,1:N_D)'*R_inv;
p_fi=[]; p_e=[];
for j=1:J
   for n=1:N
      p_fi(n,j)=E(:,n)'*B(:,:,j)'*B(:,:,j)*E(:,n);
      Fi_inv = (Fi-Fi(n,n)*eye(N));
      Fi_inv(n,n)=1;
      Fi_inv = inv(Fi_inv);
      Fi_inv(n,n)=0;
      p_e(:,n,j) = -E*Fi_inv*E'*B(:,:,j)'*B(:,:,j)*E(:,n);
   end
end
p_R=[];
for j=1:J
   temp = p_e(:,:,j)*Fi^(-0.5)*E';
   p_R(:,:,j) = temp+temp'-1/2*E*Fi^(-1.5)*diag(p_fi(:,j))*E';
end
clear temp;
p_RH=[]; p_D=[]; p_v=[];
for j =1:J
   temp = p_R(:,:,j)*H'*H*R_inv;
   p_RH(:,:,j) = temp+temp';
   for n=1:N
      p_D(n,j)=V(:,n)'*(p_RH(:,:,j))*V(:,n);
      D_inv = D-D(n,n)*eye(N);
      D_inv(n,n) = 1;
      D_inv = inv(D_inv);
      D_inv(n,n) = 0;%  D_inv = (D_inv<1).*D_inv;
```

```
        p_v(:,n,j) = - V*D_inv*V'*p_RH(:,:,j)*V(:,n);
    end
end
p_D(N_D+1:N,:)=0;
p_v(:,N_D+1:N,:)=0;
clear temp
dd=diag(D);
D_D=diag(D);
dd(N_D+1:N)=1;
D_1=diag(dd);
D_D=dd.^(-1);  % for I-inv(D_1) with N_D elements
D_D(N_D+1:N)=0;
D_2=diag(D_D);     % for inv(D_1) with N_D elements.
p_vd=[]; p_Q=[];
for j=1:J
    temp_1 = p_v(:,:,j)*(eye(N)-inv(D_1))*V';
    p_vd(:,:,j) =temp_1+temp_1'+V*D_2^(2)*diag(p_D(:,j))*V';
    temp_2 = p_R(:,:,j)*V*(eye(N)-inv(D_1))*V'*R_inv;
    p_Q(:,:,j)=temp_2+temp_2'+R_inv*p_vd(:,:,j)*R_inv;
end
clear temp_1;
clear temp_2;
p_J=[];
for j=1:J
    tr_c = 0;
    for i=1:J
        tr_c = mu(i)*trace(B(:,:,i)*p_Q(:,:,j)*B(:,:,i)')+tr_c;
    end
    p_J(j) = (-trace(inv(eye(M)+H*Q*H')*H*p_Q(:,:,j)*H')+trace(B(:,:,j)*Q*B(:,:,j)')-...
                P(j)+tr_c)+1/mu(j)/t_b;
end
p2_fi=[]; p2_e=[];
for n =1:N
    for i=1:J
        for j=1:J
            p2_fi(n,i,j)=E(:,n)'*((B(:,:,i)'*B(:,:,i)-p_fi(n,i)*eye(N))*p_e(:,n,j)+...
                (B(:,:,j)'*B(:,:,j)-p_fi(n,j)*eye(N))*p_e(:,n,i));
            Fi_inv = (Fi-Fi(n,n)*eye(N));
            Fi_inv(n,n)=1;
            Fi_inv = inv(Fi_inv);
            Fi_inv(n,n)=0;
            p2_e(:,n,i,j)=E*Fi_inv*E'*(p_fi(n,i)*p_e(:,n,j)+p_fi(n,j)*p_e(:,n,i)-...
                B(:,:,i)'*B(:,:,i)*p_e(:,n,j)-B(:,:,j)'*B(:,:,j)*p_e(:,n,i));
        end
    end
end
```

```
temp=zeros(N,N); p2_R=[]; p2_RH=[];
for i=1:J
   for j=1:J
      temp = p2_e(:,:,i,j)*Fi^(-0.5)*E'-0.5*p_e(:,:,i)*Fi^(-1.5)*diag(p_fi(:,j))*E'+...
           p_e(:,:,i)*Fi^(-0.5)*p_e(:,:,j)'-0.5*p_e(:,:,j)*Fi^(-1.5)*diag(p_fi(:,i))*E';
      p2_R(:,:,i,j)=temp+temp'+E*(3/4*Fi^(-2.5)*diag(p_fi(:,j))*diag(p_fi(:,i))-...
           0.5*Fi^(-1.5)*diag(p2_fi(:,i,j)))*E';
      temp_r=p2_R(:,:,i,j)*H'*H*R_inv+p_R(:,:,i)*H'*H*p_R(:,:,j);
      p2_RH(:,:,i,j)=temp_r+temp_r';
   end
end
clear temp;clear temp_r
p2_D=[]; p2_v=[];
for n=1:N
   for i=1:J
      for j=1:J
         p2_D(n,i,j)=V(:,n)'*p2_RH(:,:,i,j)*V(:,n)+V(:,n)'*(p_RH(:,:,i)-...
              p_D(n,i)*eye(N))*p_v(:,n,j)+V(:,n)'*(p_RH(:,:,j)-p_D(n,j)*eye(N))*p_v(:,n,i);
         D_inv = D-D(n,n)*eye(N);
         D_inv(n,n) = 1;
         D_inv = inv(D_inv);
         D_inv(n,n) = 0;
         % D_inv = (D_inv<1).*D_inv;
         p2_v(:,n,i,j)= V*D_inv*V'*(-p2_RH(:,:,i,j)*V(:,n)+p_D(n,i)*p_v(:,n,j)-...
            p_RH(:,:,i)*p_v(:,n,j)+p_D(n,j)*p_v(:,n,i)-p_RH(:,:,j)*p_v(:,n,i));
      end
   end
end
p2_D(N_D+1:N,:,:)=0;
p2_v(:,N_D+1:N,:,:)=0;
temp_vd=zeros(N,N);
temp_q=zeros(N,N);
p2_vd=[]; p2_Q=[];
for i=1:J
   for j=1:J
      temp_vd=p2_v(:,:,i,j)*(eye(N)-inv(D_1))*V'+p_v(:,:,j)*D_2^2*diag(p_D(:,i))*V'+...
         +p_v(:,:,j)*(eye(N)-inv(D_1))*p_v(:,:,i)'+p_v(:,:,i)*D_2^2*diag(p_D(:,j))*V';
      p2_vd(:,:,i,j)=temp_vd+temp_vd'-2*V*D_2^3*diag(p_D(:,i))*diag(p_D(:,j))*V'+...
         V*D_2^2*diag(p2_D(:,i,j))*V';
      temp_q=p2_R(:,:,i,j)*V*(eye(N)-inv(D_1))*V'*R_inv+p_R(:,:,j)*p_vd(:,:,i)*R_inv+...
         p_R(:,:,j)*V*(eye(N)-inv(D_1))*V'*p_R(:,:,i)+p_R(:,:,i)*p_vd(:,:,j)*R_inv;
      p2_Q(:,:,i,j)=temp_q+temp_q'+R_inv*p2_vd(:,:,i,j)*R_inv;
   end
end
clear temp_vd;clear temp_q;
for i = 1:J
```

```
            for j = 1:J
               tr_l=0;
               for t =1:J
                  tr_l = tr_l+mu(t)*trace(B(:,:,t)*p2_Q(:,:,i,j)*B(:,:,t)');
               end
               p2_L(i,j) = trace(inv(eye(M)+H*Q*H')*H*p_Q(:,:,i)*H'*inv(eye(M)+...
                     H*Q*H')*H*p_Q(:,:,j)*H')-...

trace(inv(eye(M)+H*Q*H')*H*p2_Q(:,:,i,j)*H')+trace(B(:,:,j)*p_Q(:,:,i)*B(:,:,j)')+...
                     trace(B(:,:,i)*p_Q(:,:,j)*B(:,:,i)')+tr_l;
            end
         end
         p2_L=p2_L-diag(mu.^-2)/t_b;
         mu = mu - p_J*inv(p2_L);
         if (min(real(mu)))<0
            break
         else
         end
         condition=p_J*inv(p2_L)*p_J';
         if abs(condition)< epsi
            flag =0;
            R = zeros(N,N);
            for j = 1:J
               R = R+mu(j)*B(:,:,j)'*B(:,:,j);
            end
            [E Fi] = svd(R);
            R_inv = E*Fi^(-0.5)*E';
            [V D] = svd(R_inv*H'*H*R_inv);
            N_D = sum(diag(D>1));
            %I = [ones(N_D,1); zeros(N-N_D,1)];
            Q = R_inv*V(:,1:N_D)*(eye(N_D)-inv(diag(temp_D(1:N_D))))*V(:,1:N_D)'*R_inv;
         else
         end
      end
      MM(out_count,:)=mu;
      capacity(out_count)=log2(det(eye(M)+H*Q*H'));
   end
   plot(real(capacity))  % plot the capacity versus the barrier constant t
   figure
   plot(real(MM(:,1)));  % plot \mu_1 versus t
   hold on
   plot(real(MM(:,2)),'r')  % plot \mu_2 versus t
```