```
end

Km = (p+q)*nu*K;
fprintf('K  =  %.15f ',K)
fprintf('    (p+q)*nu*Ks/2 =  %.15f \n',Km/2)
fprintf('sum(lambda) =  %.15f ',sum(lamb))
fprintf('    sum(mu) =  %.15f \n',sum(mu))

if (numsvl1 > 0 || numsvm1 > 0) && badnu == 0
   if eta < 10^(-9)
         fprintf('** Warning, eta too small or negative ** \n')
         eta = 0;
   end
   delta = eta/nw;
   fprintf('delta =  %.15f \n',delta)
   tolxi = 10^(-10);
   % tols < lambda_i < K - tolh or K - tolh <= lambda_i and epsilon_i < tolxi
   [lamsv,psf,epsilon] = findsvl2(lamb,w,b,u,eta,K,tols,tolh,tolxi);
    % tols < mu_i < K - tolh or K - tolh <= mu_i and xi_i < tolxi
   [musv,qsf,xi] = findsvm2(mu,w,b,v,eta,K,tols,tolh,tolxi);
   fprintf('psf =  %d ',psf)
   fprintf('   qsf =  %d \n',qsf)
   fprintf('pf - psf =  %d ',pf - psf)
   fprintf('   qf - qsf =  %d \n',qf - qsf)

   % computes eta from the duality gap
   errterm = w'*(sKv - sKu) + (pf - qf)*b;
   Pterm = (1/K)*(lam'*P2*lam);
   denom = (p+q)*nu - pf -qf;
   fprintf('denom =  %.15f \n',denom)
   if denom > 0
      eta1 = (errterm + Pterm)/denom;
      fprintf('eta1 =  %.15f \n',eta1)
   end
end
end
```

The constraint matrix and the matrices defining the quadratic program are constructed by the function `buildSVMs2pb`.

```
function [A,c,X,Pa,q] = buildSVMs2pb(nu,u,v,K)
%  builds the matrix of constraints A for
%  soft margin nu-SVM s2'
```