```
%
%   First builds the matrices for the dual program
%    K is a scale factor
%
p = size(u,2); q = size(v,2); n = size(u,1);
[A,c,X,Pa,qa] = buildSVMs2pb(nu,u,v,K);
%
%  Runs quadratic solver
%
tolr = 10^(-10); tols = 10^(-10); iternum = 80000;
[x,U,nr,ns,kk] = qsolve1(Pa, qa, A, c, rho, tolr, tols, iternum);
fprintf('nr =  %d ',nr)
fprintf('   ns =  %d \n',ns)
fprintf('kk =  %d \n',kk)
noconv = 0;
if kk > iternum
   noconv = 1;
   fprintf('** qsolve did not converge. Problem not solvable ** \n')
end
lam = x(1:(p+q),1);
alpha = x((p+q+1):2*p+q,1);
beta = x(2*p+q+1:2*(p+q),1);
w = -X*lam;
nw = sqrt(w'*w);   % norm of w
fprintf('nw =  %d \n',nw)
%
lamb = x(1:p,1);
mu = x(p+1:p+q,1);
tols = 10^(-10); tolh = 10^(-9);
% tols < lambda_i < K - tolh
[lambnz,numsvl1] = findpsv2(lamb,K,tols,tolh);
% tols < mu_i < K - tolh
[munz,numsvm1] = findpsv2(mu,K,tols,tolh);
fprintf('numsvl1 = %d ',numsvl1)
fprintf('   numsvm1 = %d \n',numsvm1)
%  lambda_i >= K - tolh
[lamK,pf] = countumf2(lamb,K,tolh);  % number of blue margin failures
%  mu_j >= K - tolh
[muK,qf] = countvmf2(mu,K,tolh);  % number of red margin failures
fprintf('pf =  %d ',pf)
fprintf('   qf =  %d \n',qf)
[~,pm] = countmlu2(lamb,tols);  % number of  points such that lambda_i > tols
```