```
%   (Without the variable gamma)
%
%  Input: an m x n matrix of data points represented as
%  as the rows of X, and y a vector in R^n
%
%   First builds the matrices for the dual program
%    C is a scale factor
%
m = size(X,1); n = size(X,2);
[A,c,P,Pa,qa] = buildnuregb(nu,X,y,C);
%
%  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('nr =  %d',nr)
fprintf('  ns =  %d',ns)
fprintf('  kk =  %d \n',kk)
noconv = 0;
if kk > iternum
   noconv = 1;
   fprintf('** qsolve did not converge. Problem not solvable ** \n')
end
lamb = x(1:m,1);
mu = x(m+1:2*m,1);
alpha = x((2*m+1):3*m,1);
beta = x(3*m+1:4*m,1);
w = X'*(mu - lamb);
%
b = 0; epsilon = 0;
tols = 10^(-10); tolh = 10^(-9);
% tols < lambda_i < C/m - tolh
[lambnz,numsvl1] = findpsv2(lamb,C/m,tols,tolh);
% tols < mu_i < C/m - tolh
[munz,numsvm1] = findpsv2(mu,C/m,tols,tolh);
fprintf('numsvl1 = %d',numsvl1)
fprintf('   numsvm1 = %d \n',numsvm1)
%  lambda_i >= C/m - tolh
[lamK,pf] = countumf2(lamb,C/m,tolh);  % number of blue margin failures
%  mu_j >= C/m - tolh
```