Stanford University, Management Science and Engineering (and ICME)

MS&E 318 (CME 338) Large-Scale Numerical Optimization

Student: Adele Kuzmiakova Spring 2018

Homework 3, Due Wednesday May 2

`http://stanford.edu/class/msande318/homework.html`

1. Please see the previous scans attached.

2. (a) Figure 1 compares the log base 10 norm of residuals $\|r_k\|$ for each iteration k using 4 solvers: PCG, MINRES, SYMMLQ, and LSQR. Additionally, Figure 2 summarizes the results in terms of number of total iterations, relative residuals, and error term for each of these solvers.

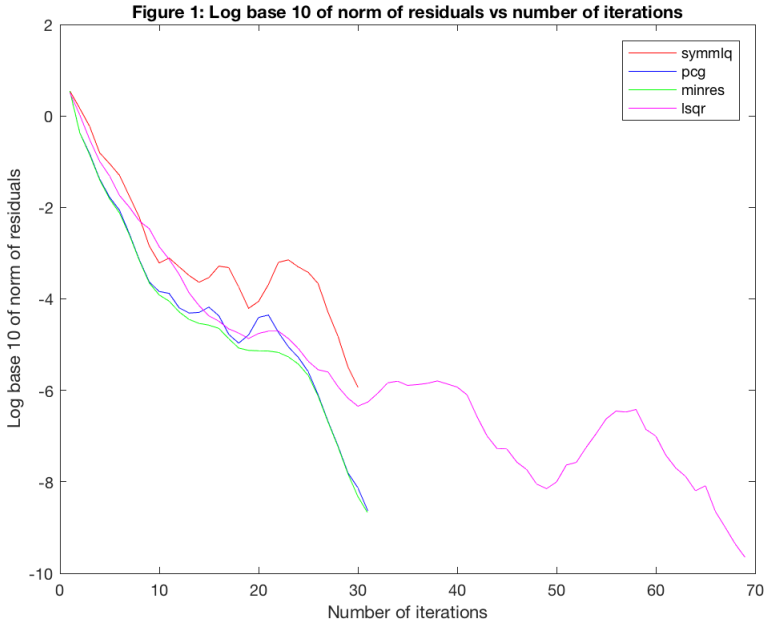

Figure 1: Log base 10 norm of residuals over k iterations.

```
POS-DEFINITE B = A + sigma1*I,  sigma1 = 2.70,  condest(B) =  2.4e+03

                 flag  iter   relres    error
CG       Bx = b   0     30  6.6e-10  1.6e-10  b
SYMMLQ   Bx = b   0     29  4.0e-10  8.7e-11  r
MINRES   Bx = b   0     30  5.9e-10  2.3e-10  g
LSQR     Bx = b   0     68  8.0e-10  3.5e-10  m
```

Figure 2: A comparison between 4 solvers in terms of total iterations, relative residuals, and the error term at convergence.

(b) All four solvers (PCG, MINRES, SYMMLQ, and LSQR) converged within a specified tolerance set to 1e-9 and hence their relative residuals in Figure 2 are on the same order of magnitude (order of 1e-10). In terms of log base 10 norm of residuals, $\|r_k\|$, we notice that LSQR performs

the best while SYMMLQ performs the worst. The difference between these two methods is approximately four orders of magnitude, which is significant. In terms of the number of iterations needed to reach the convergence, solvers PCG, MINRES, and SYMMLQ used 29-30 iterations while LSQR needed 68 iterations (almost twice as much). There are a couple of properties we can check on A and B in order to explain these differences:

(i) A is symmetric (checked by `issymmetric(A)`), which favors PCG, MINRES, and SYMMLQ solvers in terms of the number of iterations since they work well for symmetric case $Ax = b$. LSQR is typically applied on unsymmetric square or rectangular A.

(ii) A is ill-conditioned since its condition number is 1.1441e+07. This situation favors LSQR, which is consistent with Question 3 where C is also ill-conditioned and LSQR performs the best.

(iii) B is reasonably well-conditioned since its condition number is 2353. Therefore small differences between 3 solvers (PCG, MINRES, and SYMMLQ) should not be affected by cond(B).

Generally, it is reasonable to expect that PCG, MINRES, and SYMMLQ converge to the specified tolerance within roughly the same number of iterations. All four solvers seek to minimize $\|r_k\|$ where $r_k = V_{k+1}t_{k+1}$ and $t_{k+1} \equiv \beta_1 e_1 - H_k y_k$. Specifically, PCG, MINRES, and SYMMLQ utilize Lanczos process while LSQR utilizes Golub-Kahan process. Therefore, each solver deals with a different subproblem:

(i) CG: makes $t_{k+1} = 0$ everywhere except its last element

(ii) MINRES: minimizes $\|r_k\|$ such that $x_k = Vky_k$

(iii) SYMMLQ: makes $t_{k+1} = 0$ everywhere except its last two elements while keeping $\|y_k\|$ as small as possible

(iv) LSQR: similar as MINRES, minimizes $\|r_k\|$ such that $x_k = Vky_k$

(c) The following stopping rules are implemented:

(i) PCG: uses the condition `(normr <= tolb || stag >= maxstagsteps || moresteps)`, where `normr` $= \|r_k\|$, `tolb = tol` $\|b\|$, `maxstagsteps` is set to 3 and refers to a stagnation where two consecutive iterations produce roughly the same results, and `moresteps` is an indicator for exceeding the maximum number of steps.

(ii) MINRES: uses the condition `(normr <= tolb || stag >= maxstagsteps || moresteps)`, with parameters being the same as above.

(iii) SYMMLQ: uses the condition `(normr <= tolb || stag >= maxstagsteps || moresteps)`, with parameters being the same as above.

(iii) LSQR: uses the condition `(normr <= tolb)`, with parameters being the same as above. Additionally, LSQR checks for convergence in $min\ |b - Ax|$

As a result, PCG, MINRES, and SYMMLQ use the same stopping criteria consisting of $\|r_k\|$ being below some treshold, number of iterations not exceeding the maximum limit, and stagnation not exceeding 3 steps. LSQR

also utilizes $\|r_k\|$ being below some treshold and considers convergence in $min\ |b - Ax|$. This explains why PCG, MINRES, and SYMMLQ used roughly the same number of iterations whereas the additional criterion in LSQR might have led to additional iterations with better performance.

3. (a) Figure 3 compares the log base 10 norm of residuals $\|r_k\|$ for each iteration k for case when $Cx = b$ with C being indefinite. Additionally, Figure 4 summarizes the results in terms of number of total iterations, relative residuals, and error term.

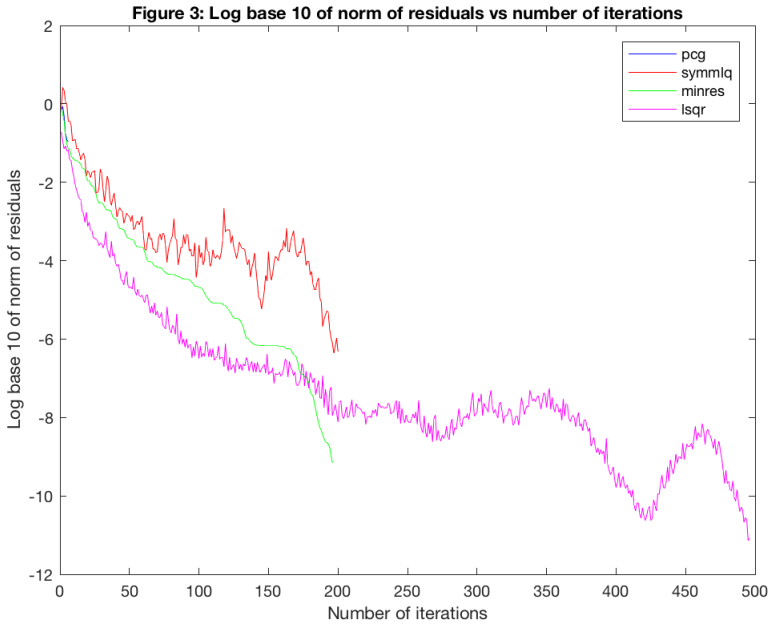

Figure 3: Log base 10 norm of residuals over k iterations for case $Cx = b$.

```
INDEFINITE C = A + sigma2*I,  sigma2 = 0.50,  condest(C) =  8.3e+04

                flag  iter   relres    error
CG       Cx = b   4     5  1.5e-01  1.6e-01  b
SYMMLQ   Cx = b   0   199  3.7e-10  2.2e-10  r
MINRES   Cx = b   0   195  9.6e-10  8.0e-10  g
LSQR     Cx = b   0   495  9.8e-10  5.4e-10  m
```

Figure 4: A comparison between 4 solvers in terms of total iterations, relative residuals, and the error term at convergence for case $Cx = b$.

(b) Since C is not a positive-definite matrix, the PCG solver terminated after 5 iterations. In Figure 4, we see that `flag = 4` is being raised to explain the early stopping. The flag refers to the condition when `one of the scalar quantities calculated during PCG became too small or too`

`large to continue computing`. Hence the solver stopped the process once it was realized that C is ill-conditioned. In situations when C is positive-definite, each T_k is theoretically positive-definite and PCG can obtain Cholesky factors $T_k = L_k D_k {T_k}^T$. In situations when C is not positive-definite, some T_k may be singular but T_k cannot be singular twice in a row. In this case, it was probably the second scenario leading to the breakdown.

(c) Consistent with the problem setup in Question 2, here C is also ill-conditioned (its condition number is 8.2837e+04). Similarly, LSQR performs the best and SYMMLQ performs the worst. The difference between the two methods is approximately 5 orders of magnitude, which is significant. In terms of the number of iterations to reach the convergence, MINRES and SYMMLQ utilized roughly the same number of iterations (195 and 199). This is expected due to the same convergence criteria summarized in Question 2. Again LSQR needs more than twice the number of iterations to reach the convergence but leads to a better performance since the system is ill-conditioned.

4. Figures 5 and 6 show the distribution of eigenvalues for case $Cx = b$. $\lambda(C)$ lie on the interval [-2.1830, 7.3331], however, some of them are repeated – for instance, $\lambda(C) = 0.5032$ is repeated three times. Additionally, from Figure 6 we can deduce that a significant majority of eigenvalues are concentrated on the interval [0.5, 0.6]. More precisely, roughly 280 eigenvalues are concentrated around the value of 0.5. The repeated values of $\lambda(C)$ suggest that C could be viewed as a matrix with fewer than n distinct eigenvalues either because some of them repeated or some of them are very similar to each other to the extent of being regarded as essentially the same. In previous problem set, we established that if A has $m < n$ distinct eigenvalues, the Lanczos process will use at most m iterations. Similarly, in our case the number of distinct eigenvalues is also less than n, and hence the solvers would require significantly fewer than n iterations.

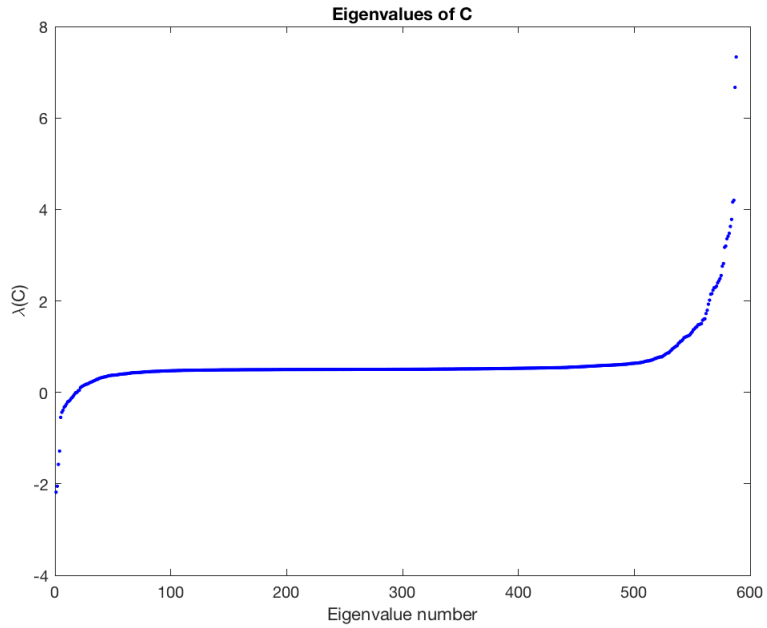

Figure 5: Distribution of eigenvalues for case $Cx = b$..

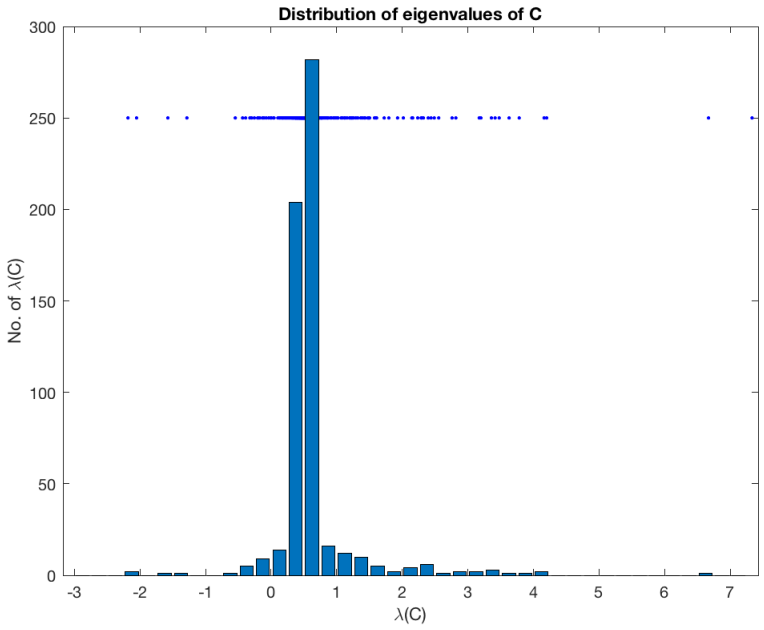

Figure 6: Distribution of eigenvalues for case $Cx = b$.

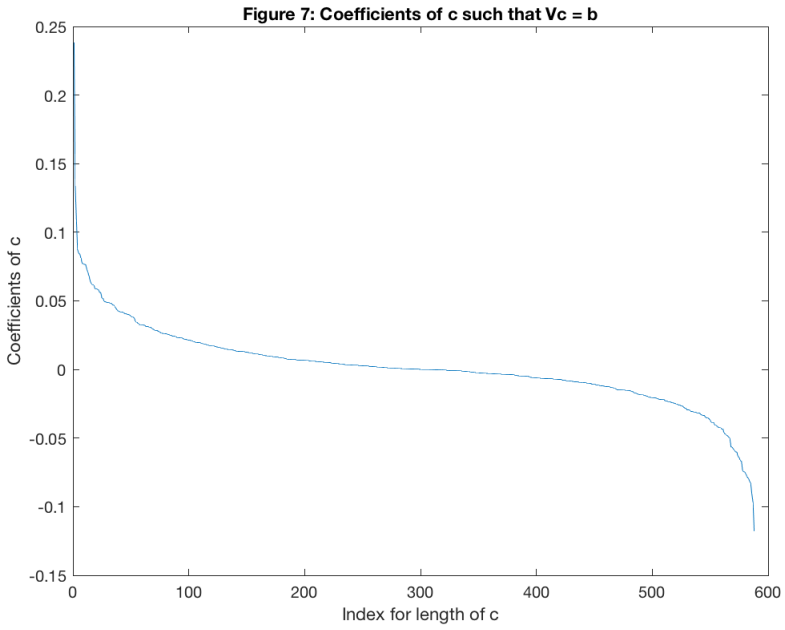

Figure 7: Coefficients of c such that $Vc = b$.

We can also obtain the full eigendecomposition of C and write the relationship $Vc = b$, where V represents an orthogonal matrix whose columns are corresponding eigenvectors of C and c represents a vector whose coefficients satisfy the following equalities:

(i) $Vc = b$

(ii) $c = V^T b$

Hence, c is coefficient vector, which represents b in terms of the eigenvector basis of C. Figure 7 reveals that a majority of coefficients are near-zero values though they never actually become zero. Therefore, it may be plausible that the Lanczos process discards some of these near-zero coefficient eigenvectors, leading to an additional reduction in the number of iterations necessary to solve $Cx = b$.

In conclusion, the combination of repeated or near-equivalent eigenvalues of C and near-zero coefficients for eigenvectors of C could explain why the symmetric solvers require significantly fewer than n iterations.

Please find the code used to produce the graphs below.

```
% CGtest8.m is a script for comparing {cg, symmlq, minres}
% on sparse matrix Boeing/bcsstm34, n=588, nnz=24270).
% See http://faculty.cse.tamu.edu/davis/welcome.html (Tim Davis).
% The matrix A is from a structural problem.
% It is symmetric indefinite with lambda_min = -2.6830.
%
% 09 Apr 2017: Problem Boeing/bcsstm34 used for Homework 3.
% 22 Apr 2018: Removed pcg and minres on C^2x = Cb.
%-------------------------------------------------------

load bcsstm34.mat;   % lambda(min) = -2.6830, lambda(max) = 6.8331
A     = Problem.A;   % Save original matrix A
condest(A)
[n,n] = size(A);
x     = 1./(1:n)';

%-------------------------------------------------------
sigma1 = 2.7;
B      = A + sigma1*speye(n);   condB = condest(B);
b      = B*x;
tol    = 1e-9;   % Not highly accurate
maxit  = 1000;

[xC,flagC,relresC,iterC,resvecC]       = pcg   (B,b,tol,maxit);
[xL,flagL,relresL,iterL,resvecL]       = symmlq(B,b,tol,maxit);
[xM,flagM,relresM,iterM,resvecM]       = minres(B,b,tol,maxit);
[xS,flagS,relresS,iterS,resvecS,lsvec] = lsqr  (B,b,tol,maxit);

errC  = norm(xC-x,inf);   % The inf-norm is best for large vectors
errL  = norm(xL-x,inf);
errM  = norm(xM-x,inf);
errS  = norm(xS-x,inf);

fprintf('\nPOS-DEFINITE B = A + sigma1*I,')
fprintf('  sigma1 =%5.2f,  condest(B) = %8.1e\n\n', sigma1, condB)
fprintf('                flag  iter   relres    error\n')
fprintf(' CG       Bx = b%4g %5g %8.1e %8.1e  b\n', flagC,iterC,relresC,errC)
fprintf(' SYMMLQ   Bx = b%4g %5g %8.1e %8.1e  r\n', flagL,iterL,relresL,errL)
fprintf(' MINRES   Bx = b%4g %5g %8.1e %8.1e  g\n', flagM,iterM,relresM,errM)
fprintf(' LSQR     Bx = b%4g %5g %8.1e %8.1e  m\n', flagS,iterS,relresS,errS)

figure(1)
hold off;  plot(log10(resvecL),'r-')
hold  on;  plot(log10(resvecC),'b-')
hold  on;  plot(log10(resvecM),'g-')
hold  on;  plot(log10(lsvec)  ,'m-')
xlabel('Number of iterations') % x-axis label
ylabel('Log base 10 of norm of residuals') % y-axis label
legend('symmlq','pcg', 'minres', 'lsqr')
title('Figure 1: Log base 10 of norm of residuals vs number of iterations')

%-------------------------------------------------------
sigma2 = 0.5;
C      = A + sigma2*speye(n);   condC  = condest(C);
```

```
b      = C*x;    Cfun  = @(x) C*x;       % Treat  C  as a function
%b2    = C*b;    Cfun2 = @(x) C*(C*x);   % Treat C*C as a function

[xC,flagC,relresC,iterC,resvecC]       = pcg   (Cfun ,b ,tol,maxit);
[xL,flagL,relresL,iterL,resvecL]       = symmlq(Cfun ,b, tol,maxit);
[xM,flagM,relresM,iterM,resvecM]       = minres(Cfun ,b ,tol,maxit);
[xS,flagS,relresS,iterS,resvecS,lsvec] = lsqr  (C    ,b ,tol,maxit);

errC  = norm(xC-x,inf);
errL  = norm(xL-x,inf);
errM  = norm(xM-x,inf);
errS  = norm(xS-x,inf);

fprintf('\n  INDEFINITE C = A + sigma2*I,')
fprintf('  sigma2 =%5.2f,  condest(C) = %8.1e\n\n', sigma2, condC)
fprintf('                flag  iter   relres    error\n')
fprintf(' CG       Cx = b%4g %5g %8.1e %8.1e  b\n', flagC,iterC,relresC,errC)
fprintf(' SYMMLQ   Cx = b%4g %5g %8.1e %8.1e  r\n', flagL,iterL,relresL,errL)
fprintf(' MINRES   Cx = b%4g %5g %8.1e %8.1e  g\n', flagM,iterM,relresM,errM)
fprintf(' LSQR     Cx = b%4g %5g %8.1e %8.1e  m\n', flagS,iterS,relresS,errS)

figure(2)
hold off;  plot(log10(resvecC),'b-')
hold  on;  plot(log10(resvecL),'r-')
hold  on;  plot(log10(resvecM),'g-')
hold  on;  plot(log10(lsvec)  ,'m-')

xlabel('Number of iterations') % x-axis label
ylabel('Log base 10 of norm of residuals') % y-axis label
legend('pcg', 'symmlq', 'minres', 'lsqr')
title('Figure 3: Log base 10 of norm of residuals vs number of iterations')

%-------------------------------------------------------------
% Plot the eigenvalues of C.
%-------------------------------------------------------------
lambda = eig(full(C));
figure(3)
hold off;    plot(lambda,'b.')
xlabel('Eigenvalue number');   ylabel('\lambda(C)');
title('Eigenvalues of C');

% Show if the eigenvalues are clustered.
figure(4)
hold off;    plot(lambda,250*ones(n,1),'b.')
hold on

y1    = -3;      yn    =  7;
step  =  0.25;   nbar  = (yn - y1)/step + 1;
y     = zeros(nbar,1);
nlam  = zeros(nbar,1);

for i = 1:nbar
  y2      = y1 + step;
  nlam(i) = length( find(lambda>y1 & lambda<=y2) );
```

```
  y(i)    = y1 + 0.5*step;
  y1      = y2;
end

bar( y, nlam )
xlabel('\lambda(C)');   ylabel('No. of \lambda(C)');
title('Distribution of eigenvalues of C');

[V, D] = eig(full(C));
c = V\b;
figure(5)
plot(sort(c, 1, 'descend'))
xlabel('Index for length of c') % x-axis label
ylabel('Coefficients of c') % y-axis label
title('Figure 7: Coefficients of c such that Vc = b')
```