CSC 2305 Numerical Methods for Optimization Problems (Winter 2016)

Assignment # 5

1002119049 Hao Wang (UTORid: `wangh110`)

March 28, 2016

1 Problem 1

1.1 Output:

```
n         k         true error        condition number
=================================================
5         6         1.56282e-07        476607
8         19        0.0131867          1.52576e+10
12        38        3.98753e+08        1.63145e+16
20        73        6.00139e+14        2.50718e+18
```

1.2 Comments

We can see the number of iterations becomes larger when the dimension of system gets larger. This is because the condition number of the Hilbert matrix is very large and becomes larger when dimension increases. Large condition number means Hilbert is nearly singular, or we can say it is "numerically" singular. Given a right hand side b, it may not belong to the range of Hilbert matrix. Hence it is hard to solve.

1.3 Source code list

```
function a5_1()
    TOL = 1e-6;
    n_list = [5 8 12 20];
    disp([sprintf('n\t'), sprintf('k\t'),...
        sprintf('true error\t'), 'condition number']);
    disp('=================================================');
    for n = n_list
        CG(n, zeros(n,1), TOL);
    end

end
```

```
function [ ] = CG( n, x, tol )
    A = hilb(n);
    b = ones(length(x) , 1);
```

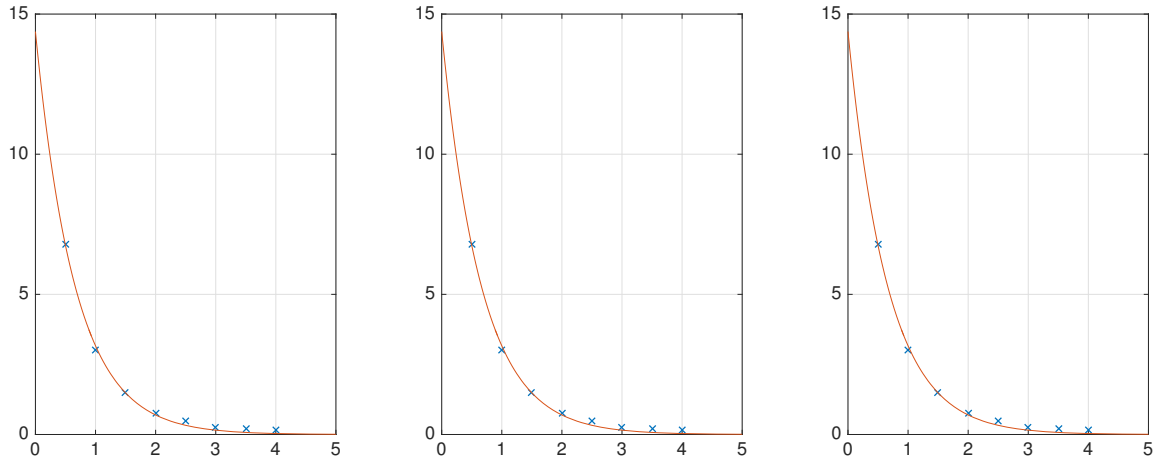
```
    r = A*x - b;
    p = -r;
    k = 0;

    while norm(r) > tol
        alpha = (r'*r) / (p'*A*p);
        x = x + alpha*p;
        old_r = r;
        r = r + alpha*A*p;
        beta = (r'*r) / (old_r' * old_r);
        p = -r + beta*p;
        k = k + 1;
    end

    % true error
    err = norm(invhilb(n)*b - x);
    disp([sprintf('%g\t', n), sprintf('%g\t', k),...
        sprintf('%g\t', err), sprintf('%g', cond(A))]);
end
```

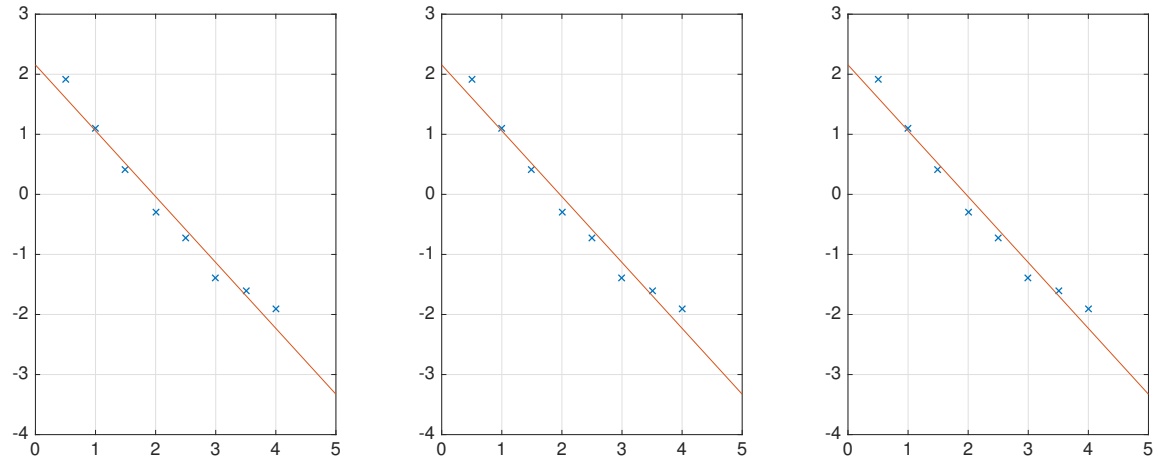
2 Problem 2

2.1 Plots



(a) Normal equations. (b) QR. (c) SVD.

Figure 1: With model function as $y = f(t, x) = x_1 e^{x_2 t}$.



(a) Normal equations. (b) QR. (c) SVD.

Figure 2: With model function as $y = f(t, x) = \log(x_1) + x_2 t$.

2.2 Comments

(a) From the plots and outputs, we can observe the solution process is not sensitive to different solution methods, such as Normal equations, QR and SVD.

(b) The values obtained in this model does not agree with values in (a). The $\mathbf{x}$ leads to a larger $f(\mathbf{x})$ compared to the $\mathbf{x}$ in (a). The reason why this happens is that in (b) we use a linear least

squares method to approximate the non-linear model, which introduces more errors compared to the non-linear least squares method – Gauss-Newton Method.

2.3 Output:

2.3.1 (a)

The outputs of Normal equations, QR, SVD are the same.

```
n         x1                  x2                  norm(gy)         f(x)
=================================================================================
1         12.749097         -1.736746          9.153186         7.851829
2         14.252544         -1.447724          6.881350         1.644517
3         14.306967         -1.505248          1.745266         0.094809
4         14.370120         -1.513220          0.069292         0.048154
5         14.376136         -1.513862          0.003133         0.047956
6         14.376591         -1.513912          0.000257         0.047955
7         14.376626         -1.513915          0.000020         0.047955
8         14.376629         -1.513916          0.000002         0.047955
9         14.376629         -1.513916          0.000000         0.047955
```

2.3.2 (b)

The outputs of Normal equations, QR, SVD are the same.

```
n         log(x1)           x2                norm(gy)         f(x)
=================================================================================
0         2.155818         -1.096669         2.331118         1.855976
```

2.4 Source code list

```
function a5_2()
    %     func = @objFunc;
    func = @objFunc2;
    TOL = 1e-6;
    x0 = [2, -1]';
    x = x0;
    [~, ~, ~, gy] = func(x(1), x(2));
    n = 0;

    disp([sprintf('n\t'), sprintf('x1\t\t'), sprintf('x2\t\t'),...
        sprintf('norm(gy)\t'), 'f(x)']);
    disp(['=================================', ...
        '=================================']);

    while norm(gy) > TOL
```

```
16
17          [r, y, Jr, gy] = func(x(1), x(2));
18          %% normal equations
19  %         p = - inv(Jr'*Jr)*Jr'*r;
20
21          %% QR
22  %         [Q,R,E] = qr(Jr);
23  %         p = -E/R*Q'*r;
24
25          %% SVD
26          [U,S,V] = svd(Jr, 0);
27          p = -V/S*U'*r;
28
29          alpha = backtracking(func, p, x, 0.5, 0.1);
30          x = x + alpha*p;
31          n = n + 1;
32
33          disp([sprintf('%g\t', n), sprintf('%f\t', x(1)), ...
34              sprintf('%f\t', x(2)), sprintf('%f\t', norm(gy)), ...
35              sprintf('%f\t', y)]);
36      end
37      %     disp(x);
38
39      %% plot
40      t = [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0];
41      y = [6.80, 3.00, 1.50, 0.75, 0.48, 0.25, 0.20, 0.15];
42
43      figure;
44      hold on;
45      box on;
46      grid on;
47      scatter(t, y, 'x');
48      set(gcf, 'Position', [0 0 300 400]);
49      set(gca,'FontSize',14);
50      fplot(@(a) x(1)*exp(x(2)*a), [0 5]);
51  end
```

```
1  function [ r, y, Jr, gy] = objFunc( x1, x2 )
2      r = [34/5 - x1*exp(x2/2), 3 - x1*exp(x2), ...
3          3/2 - x1*exp((3*x2)/2), 3/4 - x1*exp(2*x2), ...
4          12/25 - x1*exp((5*x2)/2), 1/4 - x1*exp(3*x2), ...
5          1/5 - x1*exp((7*x2)/2), 3/20 - x1*exp(4*x2)]';
6
7      y = 1/2*((x1*exp(x2) - 3)^2 + (x1*exp(3*x2) - 1/4)^2 ...
8          + (x1*exp(2*x2) - 3/4)^2 + (x1*exp((3*x2)/2) - 3/2)^2 ...
9          + (x1*exp((7*x2)/2) - 1/5)^2 + (x1*exp(4*x2) - 3/20)^2 ...
10         + (x1*exp(x2/2) - 34/5)^2 + (x1*exp((5*x2)/2) - 12/25)^2);
```

```
11
12      gy = [exp(2*x2)*(x1*exp(2*x2) - 3/4) + ...
13          exp(3*x2)*(x1*exp(3*x2) - 1/4) ...
14          + exp((3*x2)/2)*(x1*exp((3*x2)/2) - 3/2) ...
15          + exp((7*x2)/2)*(x1*exp((7*x2)/2) - 1/5) ...
16          + exp(4*x2)*(x1*exp(4*x2) - 3/20) ...
17          + exp(x2/2)*(x1*exp(x2/2) - 34/5) ...
18          + exp((5*x2)/2)*(x1*exp((5*x2)/2) - 12/25) ...
19          + exp(x2)*(x1*exp(x2) - 3); ...
20          2*x1*exp(2*x2)*(x1*exp(2*x2) - 3/4) ...
21          + 3*x1*exp(3*x2)*(x1*exp(3*x2) - 1/4) ...
22          + (3*x1*exp((3*x2)/2)*(x1*exp((3*x2)/2) - 3/2))/2 ...
23          + (7*x1*exp((7*x2)/2)*(x1*exp((7*x2)/2) - 1/5))/2 ...
24          + 4*x1*exp(4*x2)*(x1*exp(4*x2) - 3/20) ...
25          + (x1*exp(x2/2)*(x1*exp(x2/2) - 34/5))/2 ...
26          + (5*x1*exp((5*x2)/2)*(x1*exp((5*x2)/2) - 12/25))/2 ...
27          + x1*exp(x2)*(x1*exp(x2) - 3)];
28
29      Jr = [-exp(conj(x2)/2),        -(exp(conj(x2)/2)*conj(x1))/2;
30          -exp(conj(x2)),              -exp(conj(x2))*conj(x1);
31          -exp((3*conj(x2))/2), -(3*exp((3*conj(x2))/2)*conj(x1))/2;
32          -exp(2*conj(x2)),          -2*exp(2*conj(x2))*conj(x1);
33          -exp((5*conj(x2))/2), -(5*exp((5*conj(x2))/2)*conj(x1))/2;
34          -exp(3*conj(x2)),          -3*exp(3*conj(x2))*conj(x1);
35          -exp((7*conj(x2))/2), -(7*exp((7*conj(x2))/2)*conj(x1))/2;
36          -exp(4*conj(x2)),          -4*exp(4*conj(x2))*conj(x1)];
37  end
```

```
1   function [ r, y, Jr, gy] = objFunc2( x1, x2 )
2
3       r = [ 1079131495240151/562949953421312 - x2/2 - x1, ...
4           2473854946935173/2251799813685248 - x2 - x1, ...
5           1826052509787667/4503599627370496 - (3*x2)/2 - x1, ...
6           - x1 - 2*x2 - 2591209748590025/9007199254740992, ...
7           - x1 - (5*x2)/2 - 3305503303392621/4503599627370496, ...
8           - x1 - 3*x2 - 6243314768165359/4503599627370496, ...
9           - x1 - (7*x2)/2 - 7248263982714163/4503599627370496, ...
10          - x1 - 4*x2 - 1067983607126147/562949953421312]';
11
12      y = (abs(x1 + x2 - 2473854946935173/2251799813685248)^2 ...
13          + abs(x1 + x2/2 - 1079131495240151/562949953421312)^2 ...
14          + abs(x1 + (7*x2)/2 + 7248263982714163/4503599627370496)^2 ...
15          + abs(x1 + (3*x2)/2 - 1826052509787667/4503599627370496)^2 ...
16          + abs(x1 + 3*x2 + 6243314768165359/4503599627370496)^2 ...
17          + abs(x1 + 2*x2 + 2591209748590025/9007199254740992)^2 ...
18          + abs(x1 + 4*x2 + 1067983607126147/562949953421312)^2 ...
19          + abs(x1 + (5*x2)/2 ...
```

```
        + 3305503303392621/4503599627370496)^2)^(1/2)/2;

    gy = [(2*abs(x1 + 4*x2 ...
        + 1067983607126147/562949953421312)*sign(x1 ...
        + 4*x2 + 1067983607126147/562949953421312) ...
        + 2*sign(x1 + x2 - 2473854946935173/2251799813685248)*abs(x1 ...
        + x2 - 2473854946935173/2251799813685248) ...
        + 2*sign(x1 + x2/2 - 1079131495240151/562949953421312)*abs(x1 ...
        + x2/2 - 1079131495240151/562949953421312) + 2*abs(x1 ...
        + (5*x2)/2 + 3305503303392621/4503599627370496)*sign(x1 ...
        + (5*x2)/2 + 3305503303392621/4503599627370496) ...
        + 2*abs(x1 + (7*x2)/2 + 7248263982714163/4503599627370496) ...
        *sign(x1 + (7*x2)/2 + 7248263982714163/4503599627370496) ...
        + 2*abs(x1 + (3*x2)/2 - 1826052509787667/4503599627370496) ...
        *sign(x1 + (3*x2)/2 - 1826052509787667/4503599627370496) ...
        + 2*abs(x1 + 3*x2 + 6243314768165359/4503599627370496) ...
        *sign(x1 + 3*x2 + 6243314768165359/4503599627370496) ...
        + 2*abs(x1 + 2*x2 + 2591209748590025/9007199254740992) ...
        *sign(x1 + 2*x2 + 2591209748590025/9007199254740992))/(4*(abs(x1 ...
        + x2 - 2473854946935173/2251799813685248)^2 + abs(x1 ...
        + x2/2 - 1079131495240151/562949953421312)^2 + abs(x1 ...
        + (7*x2)/2 + 7248263982714163/4503599627370496)^2 ...
        + abs(x1 + (3*x2)/2 - 1826052509787667/4503599627370496)^2 ...
        + abs(x1 + 3*x2 + 6243314768165359/4503599627370496)^2 ...
        + abs(x1 + 2*x2 + 2591209748590025/9007199254740992)^2 ...
        + abs(x1 + 4*x2 + 1067983607126147/562949953421312)^2 ...
        + abs(x1 + (5*x2)/2 + 3305503303392621/4503599627370496)^2)^(1/2));
        (8*abs(x1 + 4*x2 + 1067983607126147/562949953421312) ...
        *sign(x1 + 4*x2 + 1067983607126147/562949953421312) ...
        + 2*sign(x1 + x2 - 2473854946935173/2251799813685248) ...
        *abs(x1 + x2 - 2473854946935173/2251799813685248) ...
        + sign(x1 + x2/2 - 1079131495240151/562949953421312) ...
        *abs(x1 + x2/2 - 1079131495240151/562949953421312) ...
        + 5*abs(x1 + (5*x2)/2 + 3305503303392621/4503599627370496) ...
        *sign(x1 + (5*x2)/2 + 3305503303392621/4503599627370496) ...
        + 7*abs(x1 + (7*x2)/2 + 7248263982714163/4503599627370496) ...
        *sign(x1 + (7*x2)/2 + 7248263982714163/4503599627370496) ...
        + 3*abs(x1 + (3*x2)/2 - 1826052509787667/4503599627370496) ...
        *sign(x1 + (3*x2)/2 - 1826052509787667/4503599627370496) ...
        + 6*abs(x1 + 3*x2 + 6243314768165359/4503599627370496) ...
        *sign(x1 + 3*x2 + 6243314768165359/4503599627370496) ...
        + 4*abs(x1 + 2*x2 + 2591209748590025/9007199254740992) ...
        *sign(x1 + 2*x2 + 2591209748590025/9007199254740992)) ...
        /(4*(abs(x1 + x2 - 2473854946935173/2251799813685248)^2 ...
        + abs(x1 + x2/2 - 1079131495240151/562949953421312)^2 ...
        + abs(x1 + (7*x2)/2 + 7248263982714163/4503599627370496)^2 ...
        + abs(x1 + (3*x2)/2 - 1826052509787667/4503599627370496)^2 ...
        + abs(x1 + 3*x2 + 6243314768165359/4503599627370496)^2 ...
```

```
        + abs(x1 + 2*x2 + 2591209748590025/9007199254740992)^2 ...
        + abs(x1 + 4*x2 + 1067983607126147/5629499534213l2)^2 ...
        + abs(x1 + (5*x2)/2 ...
        + 3305503303392621/4503599627370496)^2)^(1/2))];

    Jr = [-1, -1/2;
          -1,   -1;
          -1, -3/2;
          -1,   -2;
          -1, -5/2;
          -1,   -3;
          -1, -7/2;
          -1,   -4];
end
```

```
function [r, f, gf, Jr] = a5 ()

    t = [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0];
    y = [6.80, 3.00, 1.50, 0.75, 0.48, 0.25, 0.20, 0.15];

    syms x1 x2;
    r = log(y) - x1 - x2*t;
%     r = y - x1*exp(x2*t);
    f = 1/2*norm(r);
    gf = gradient(f);
    Jr = jacobian(r);

end
```

3 Problem 3

3.1 Comments

It reaches the convergence at iteration number $k = 180$. In the Section 3.2, we can verify that when $k < n$, the two inequality-equations are satisfied. From the output, we can see when the iteration begins, the equation (5.6) has provided a tight upper bound and the Theorem 5.5 has provided a tight upper bound when k approaches n:

$$k = 0,\ \|x_k - x^*\|_A = 29.577 \leq 2\Big(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\Big)^k \|x_0 - x^*\|_A = 59.2$$

$$k = 97,\ \|x_{k+1} - x^*\|_A^2 = 0.001 \leq \Big(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\Big)^2 \|x_0 - x^*\|_A^2 = 0.138$$

3.2 Output

`left1` denotes $\|x_{k+1} - x^*\|_A^2$,
`right1` denotes $\left(\frac{\lambda_{n-k}-\lambda_1}{\lambda_{n-k}+\lambda_1}\right)^2 \|x_0 - x^*\|_A^2$,
`left2` denotes $\|x_k - x^*\|_A$,
`right2` denotes $2\Big(\frac{\sqrt{\kappa(A)}-1}{\sqrt{\kappa(A)}+1}\Big)^k \|x_0 - x^*\|_A$,
`ineq1` denotes the equation from Theorem 5.5 and `ineq2` denotes the equation (5.36). Once the equation is satisfied, the output will be `T`, which means "True", and `F` means "False".

k	true error	left1	right1	left2	right2	ineq1	ineq2
0	9.94	852.896	5.72e+09	29.577	59.2	T	T
1	6.77	139.560	8.22e+05	29.204	59.1	T	T
2	5.45	40.349	7.2e+05	11.814	59.1	T	T
3	4.87	20.719	6.66e+05	6.352	59.1	T	T
4	4.46	13.673	6.17e+05	4.552	59	T	T
5	3.97	8.742	5.99e+05	3.698	59	T	T
6	3.54	5.875	5.39e+05	2.957	59	T	T
7	3.33	4.838	5.13e+05	2.424	58.9	T	T
8	3.15	3.745	5.01e+05	2.199	58.9	T	T
9	2.89	2.558	4.8e+05	1.935	58.9	T	T
10	2.68	1.887	4.42e+05	1.599	58.9	T	T
11	2.48	1.382	4.1e+05	1.374	58.8	T	T
12	2.31	1.045	3.46e+05	1.176	58.8	T	T
13	2.16	0.776	3.35e+05	1.022	58.8	T	T
14	2.07	0.636	3.25e+05	0.881	58.7	T	T
15	2.02	0.558	2.9e+05	0.797	58.7	T	T
16	1.91	0.377	2.77e+05	0.747	58.7	T	T
17	1.85	0.304	2.65e+05	0.614	58.6	T	T
18	1.79	0.238	2.46e+05	0.551	58.6	T	T
19	1.75	0.195	2.41e+05	0.488	58.6	T	T
20	1.71	0.167	2.27e+05	0.442	58.6	T	T

21	1.71	0.166	2.08e+05	0.409	58.5	T	T
22	1.67	0.144	2e+05	0.408	58.5	T	T
23	1.62	0.123	1.9e+05	0.379	58.5	T	T
24	1.58	0.107	1.85e+05	0.350	58.4	T	T
25	1.55	0.097	1.72e+05	0.327	58.4	T	T
26	1.51	0.087	1.61e+05	0.312	58.4	T	T
27	1.47	0.080	1.45e+05	0.296	58.4	T	T
28	1.47	0.079	1.41e+05	0.282	58.3	T	T
29	1.43	0.073	1.23e+05	0.281	58.3	T	T
30	1.4	0.067	1.19e+05	0.270	58.3	T	T
31	1.36	0.062	1.14e+05	0.259	58.2	T	T
32	1.31	0.057	1.01e+05	0.249	58.2	T	T
33	1.27	0.052	9.52e+04	0.238	58.2	T	T
34	1.25	0.050	8.88e+04	0.227	58.1	T	T
35	1.2	0.045	8.32e+04	0.224	58.1	T	T
36	1.14	0.039	7.42e+04	0.213	58.1	T	T
37	1.09	0.034	7.17e+04	0.198	58.1	T	T
38	1.03	0.029	6.66e+04	0.185	58	T	T
39	0.996	0.025	6.28e+04	0.170	58	T	T
40	0.964	0.022	6.14e+04	0.158	58	T	T
41	0.964	0.022	5.17e+04	0.148	57.9	T	T
42	0.928	0.019	5.1e+04	0.148	57.9	T	T
43	0.904	0.016	4.36e+04	0.136	57.9	T	T
44	0.87	0.013	4.1e+04	0.128	57.9	T	T
45	0.852	0.012	3.86e+04	0.115	57.8	T	T
46	0.838	0.010	3.59e+04	0.107	57.8	T	T
47	0.837	0.010	3.32e+04	0.101	57.8	T	T
48	0.822	0.009	2.99e+04	0.101	57.7	T	T
49	0.813	0.008	2.78e+04	0.095	57.7	T	T
50	0.806	0.008	2.7e+04	0.091	57.7	T	T
51	0.799	0.007	2.59e+04	0.088	57.6	T	T
52	0.791	0.007	2.19e+04	0.086	57.6	T	T
53	0.78	0.006	1.65e+04	0.083	57.6	T	T
54	0.779	0.006	1.49e+04	0.080	57.6	T	T
55	0.765	0.006	1.45e+04	0.080	57.5	T	T
56	0.75	0.005	1.41e+04	0.075	57.5	T	T
57	0.741	0.005	1.32e+04	0.071	57.5	T	T
58	0.729	0.004	1.15e+04	0.069	57.4	T	T
59	0.721	0.004	1.12e+04	0.065	57.4	T	T
60	0.718	0.004	1e+04	0.063	57.4	T	T
61	0.713	0.004	9.91e+03	0.062	57.4	T	T
62	0.701	0.003	8.72e+03	0.061	57.3	T	T
63	0.693	0.003	7.53e+03	0.057	57.3	T	T
64	0.685	0.003	6.65e+03	0.055	57.3	T	T
65	0.675	0.002	5.99e+03	0.052	57.2	T	T
66	0.671	0.002	5.35e+03	0.049	57.2	T	T
67	0.671	0.002	4.63e+03	0.047	57.2	T	T
68	0.668	0.002	3.99e+03	0.047	57.2	T	T

69	0.666	0.002	3.29e+03	0.046	57.1	T	T
70	0.662	0.002	2.74e+03	0.046	57.1	T	T
71	0.66	0.002	2.61e+03	0.045	57.1	T	T
72	0.656	0.002	2.46e+03	0.045	57	T	T
73	0.655	0.002	1.88e+03	0.044	57	T	T
74	0.652	0.002	1.63e+03	0.044	57	T	T
75	0.646	0.002	1.25e+03	0.044	57	T	T
76	0.634	0.002	1.07e+03	0.043	56.9	T	T
77	0.62	0.002	911	0.042	56.9	T	T
78	0.612	0.002	875	0.041	56.9	T	T
79	0.603	0.002	685	0.040	56.8	T	T
80	0.602	0.002	573	0.039	56.8	T	T
81	0.587	0.001	429	0.039	56.8	T	T
82	0.577	0.001	369	0.037	56.8	T	T
83	0.571	0.001	299	0.036	56.7	T	T
84	0.562	0.001	213	0.036	56.7	T	T
85	0.557	0.001	126	0.035	56.7	T	T
86	0.545	0.001	120	0.034	56.6	T	T
87	0.544	0.001	117	0.033	56.6	T	T
88	0.531	0.001	79.5	0.033	56.6	T	T
89	0.523	0.001	60.8	0.031	56.6	T	T
90	0.513	0.001	50.3	0.030	56.5	T	T
91	0.506	0.001	26.6	0.029	56.5	T	T
92	0.501	0.001	19	0.028	56.5	T	T
93	0.501	0.001	7.67	0.027	56.4	T	T
94	0.497	0.001	4.79	0.027	56.4	T	T
95	0.493	0.001	1.37	0.026	56.4	T	T
96	0.489	0.001	1.08	0.026	56.4	T	T
97	0.482	0.001	0.138	0.025	56.3	T	T
98	0.476	0.001	0.243	0.024	56.3	T	T
99	0.469	0.000	874	0.023	56.3	T	T
100	0.467				F	F	
101	0.462				F	F	
102	0.458				F	F	
103	0.456				F	F	
104	0.454				F	F	
105	0.450				F	F	
...							
172	0.002				F	F	
173	0.000				F	F	
174	0.000				F	F	
175	0.000				F	F	
176	0.000				F	F	
177	0.000				F	F	
178	0.000				F	F	
179	0.000				F	F	

3.3 Source code list

```
function a5_3()
    TOL = 1e-6;

    disp([sprintf('k  '), sprintf('true error\t'), ...
        sprintf('left1\t'), sprintf('right1\t\t'), ...
        sprintf('left2\t'), sprintf('right2\t'), ...
        sprintf('ineq1  '), sprintf('ineq2')]);
    disp('================================================');

    n = 100;
    M = rand(n, n);
    A = M'*M;
    x_opt = repmat([1, -1], 1, 50)';
    b = A*x_opt;

    x0 = zeros(n, 1);
    x = x0;
    r = A*x - b;
    p = -r;
    k = 0;
    e = eig(A);

    while norm(r) > TOL
        ineq1 = 'F';
        ineq2 = 'F';
        alpha = (r'*r) / (p'*A*p);
        old_x = x;
        x = x + alpha*p;
        old_r = r;
        r = r + alpha*A*p;
        beta = (r'*r) / (old_r' * old_r);
        p = -r + beta*p;

        err = norm(x_opt - x);

        if k >= n
            disp([sprintf('%g  ', k),...
                sprintf('%.3f\t', err), ineq1, sprintf('  '), ineq2]);
        else
            left1 = (x - x_opt)'*A*(x - x_opt);
            right1 = (e(n-k) - e(1) / e(n-k) + e(1))^2 ...
                * (x0 - x_opt)'*A*(x0 - x_opt);

            left2 = (old_x - x_opt)'*A*(old_x - x_opt);
            right2 = ( (sqrt(e(1)/e(n))-1)/(sqrt((e(1)/e(n)))+1) )^(2*k) ...
```

```
46                  * (x0-x_opt)'*A*(x0-x_opt);
47
48              if left1 <= right1
49                  ineq1 = 'T';
50              end
51
52              if left2 <= right2
53                  ineq2 = 'T';
54              end
55
56              disp([sprintf('%g  ', k),...
57                  sprintf('%6.3g\t', err), sprintf('%3.3f\t', left1), ...
58                  sprintf('%8.3g\t ', right1), sprintf('%4.3f\t', left2), ...
59                  sprintf('%.3g  ', right2), ineq1, sprintf('  '), ineq2]);
60          end
61          k = k + 1;
62      end
63  end
```