

Math 426/CISC 410 08F, All Sections

R.J. Braun

Homework 6 Solutions, Hints and Answers

Problem 2.7.1. We assume that the relative change in b is eps ; that is, $\|\Delta b\|/\|b\| = \text{eps}$. This script will do the job.

```
% Script: Prob2_7_1.m
% Problem 2.7.1: verify relative error bound using 2-norm

format compact;
format short e;
n = [10:10:70];
relerr = []; condA = []
for k = 1:length(n)
    A = gallery('prolate',n(k),0.4);
    x_exact = [1:n(k)]'/n(k);
    b = A*x_exact;
    x = A\b;
    relerr = [relerr norm(x-x_exact)/norm(x)];
    condA = [condA cond(A)];
end

disp(' ') % a blank line
disp(' Problem 2.7.1 ')
disp(sprintf(' n      rel err      kappa(A)      rhs      '))
disp('-----')
for k=1:length(n)
    disp(sprintf('%2.0f   %6.4e   %6.4e   %6.4e',...
                n(k),relerr(k),condA(k),condA(k)*eps))
end
```

The bound works in every case as shown by the results below. Here `rel err` denotes the relative error in the solution, `kappa(A)` is the 2-norm of the A , and `rhs` denotes the right hand side of the error bound.

```
>> Prob2_7_1
condA =
     []
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 2.184501e-016.
> In Prob2_7_1 at 12
Warning: Matrix is close to singular or badly scaled.
Results may be inaccurate. RCOND = 1.434659e-017.
> In Prob2_7_1 at 12
```

```

Problem 2.7.1
n      rel err      kappa(A)      rhs
-----
10    4.9190e-016    5.6612e+001    1.2570e-014
20    3.1729e-015    2.0787e+004    4.6157e-012
30    1.4464e-010    9.8017e+006    2.1764e-009
40    1.1723e-007    4.9834e+009    1.1065e-006
50    5.6409e-005    2.6328e+012    5.8460e-004
60    7.9595e-002    1.0663e+015    2.3677e-001
70    7.8108e-001    8.5296e+015    1.8939e+000
>>

```

Problem 2.7.4. The contents of this script file shown below will solve this problem. In the approach below, A is constructed with a `toeplitz` command, but then the last column is changed. Using this approach emphasizes the structure of the matrix much more than typing it in. Also, a systematic increase in size of the matrix makes for a more convincing comparison.

```

% Script: Prob2_7_4.m
% Problem 2.7.4: verify growth of condition numbers with increasing
% size of the pathological matrix for partial pivoting and LU factorization

format compact;
format short e;
n = [5:5:50]
condL_theory = n.*2.^(n-1);
condU_theory = 2.^n - 1;
condA = []; condL = []; condU = [];
for k = 1:length(n)
    A = toeplitz([1 -ones(1,n(k)-1)], [1 zeros(1,n(k)-1)]); A(:,n(k))=1;
    [L,U] = lu(A);
    condA = [condA cond(A,1)];
    condL = [condL cond(L,1)];
    condU = [condU cond(U,1)];
end

disp(' ') % a blank line
disp(' Problem 2.7.4 ')
disp(sprintf(' n      cond(A,1)      L,theory      cond(L,1)      U_theory      cond(U,1)  ')
disp('-----'))
for k=1:length(n)
    disp(sprintf('%2.0f      %5.3e      %5.3e      %5.3e      %5.3e      %5.3e',...
    n(k),condA(k),condL_theory(k),condL(k),condU_theory(k),condU(k)))
end

```

The output below shows that the bounds really work, even when the conditioning is poor.

The warning occurs at $n = 50$, when the conditioning is so bad that the matrix is effectively singular.

```
>> Prob2_7_4
n =
     5     10     15     20     25     30     35     40     45     50
Warning: Matrix is close to singular or badly scaled.
       Results may be inaccurate. RCOND = 3.552714e-017.
> In cond at 48
   In Prob2_7_4 at 14
```

```
Problem 2.7.4
n      cond(A,1)      L,theory      cond(L,1)      U_theory      cond(U,1)
-----
  5      5.000e+000      8.000e+001      8.000e+001      3.100e+001      3.100e+001
 10      1.000e+001      5.120e+003      5.120e+003      1.023e+003      1.023e+003
 15      1.500e+001      2.458e+005      2.458e+005      3.277e+004      3.277e+004
 20      2.000e+001      1.049e+007      1.049e+007      1.049e+006      1.049e+006
 25      2.500e+001      4.194e+008      4.194e+008      3.355e+007      3.355e+007
 30      3.000e+001      1.611e+010      1.611e+010      1.074e+009      1.074e+009
 35      3.500e+001      6.013e+011      6.013e+011      3.436e+010      3.436e+010
 40      4.000e+001      2.199e+013      2.199e+013      1.100e+012      1.100e+012
 45      4.500e+001      7.916e+014      7.916e+014      3.518e+013      3.518e+013
 50      5.000e+001      2.815e+016      2.815e+016      1.126e+015      1.126e+015
>>
```

Problem 2.8.1. (a) Yes. (b) No (rows 2 and 3). (c) No (row 5).

Problem 2.8.2. >> % Computational approaches

```
>> A = [1 0 -1; 0 4 5; -1 5 10]
A =
     1     0     -1
     0     4     5
    -1     5    10
>> R = chol(A)
R =
 1.0000e+000         0 -1.0000e+000
         0 2.0000e+000 2.5000e+000
         0         0 1.6583e+000
>> % A is symmetric, chol(A) works, so A is SPD.
>> R'*R
ans =
     1     0     -1
     0     4     5
    -1     5    10
>> B = [1 0 1; 0 4 5; -1 5 10]
```

```

B =
     1     0     1
     0     4     5
    -1     5    10
>> B'==B
ans =
     1     1     0
     1     1     1
     0     1     1
>> B'-B
ans =
     0     0    -2
     0     0     0
     2     0     0
>> % working hard at this, either of the above shows that B is not symmetric
>> % so it can't be SPD.
>> C = [1 0 1; 0 4 5; 1 5 1]
C =
     1     0     1
     0     4     5
     1     5     1
>> R = chol(C)
??? Error using ==> chol
Matrix must be positive definite.

>> % C is symmetric, but not SPD; Matlab did the test for us.
Here's one by hand. Working with A, we can make a non-negative quadratic form as follows.
Let  $x = [x_1 \ x_2 \ x_3]^T$ .

```

$$\begin{aligned}
x^T Ax &= x^T Ax \\
&= [x_1 \ x_2 \ x_3] \begin{bmatrix} x_1 - x_3 \\ 4x_2 + 5x_3 \\ -x_1 + 5x_2 + 10x_3 \end{bmatrix} \\
&= x_1^2 - x_1x_3 + 4x_2^2 + 5x_2x_3 - x_1x_3 + 5x_2x_3 + 10x_3^2 \\
&= (x_1^2 - 2x_1x_3 + x_3^2) + 4x_2^2 + 10x_2x_3 + 9x_3^2 \\
&= (x_1 - x_3)^2 + (x_2^2 + 4x_2x_3 + 4x_3^2) + 2x_3^3 + (3x_2^2 + 6x_2x_3 + 3x_3^2) \\
&= (x_1 - x_3)^2 + (x_2 + 2x_3)^2 + 2x_3^3 + 3(x_2 + x_3)^2.
\end{aligned}$$

Problem 3.1.3. The following script does the comparison with Problem 2.3.3 and a fit of the linearization of $y \approx ae^{bt}$. To make this into a linear least squares problem, take the log (natural) of both sides: $\log(y) = \log(a) + bt$; then, let $c_1 = \log(a)$ and $c_2 = b$ and solve for the c_i .

```

% Script: Problem 3.1.3
% try different fit on census data

```

```

% linearize the fit  $y = a \exp(bt)$  to
%  $\ln y = \ln a + b t$  and then do least squares

% linearized fit
load census;
t = cdate - 1790; y = log(pop);
A = zeros(length(t),2);
for j=1:2
    A(:,j) = t.^(j-1);
end
c = (A'*A)\(A'*y);
a = exp(c(1)), b = c(2)
tt = linspace(cdate(1),cdate(end),101);
y = a*exp(b*(tt-1790));

% original fit
p = 3;
A = ones(length(t),p+1);
for j=2:p+1
    A(:,j) = t.^(j-1);
end
c_orig = (A'*A)\(A'*pop);
y_orig = A*c_orig;

plot(tt,y,'-',cdate,pop,'o',cdate,y_orig,'--')
xlabel('year'); ylabel('Population (millions)');
legend('a exp(bt)', 'data', '3rd deg poly', 'Location', 'NorthWest');
title('Problem 3.1.3 -- Fit of Population Data');

```

The following plot shows the results in the original variables y and t . The cubic fit is better than the linearized exponential fit.

Problem 3.1.4. The following script will do the job; the published version from Matlab is on the course web page.

```

%% Problem 3.1.4
% Try different fits on periodic function
%  $y = \exp(\sin(t-1))$ 
%% Part (a), polynomial fit
%Use a polynomial fit to the data.
clear all; format compact;
t = linspace(0,2*pi,200)'; b = exp(sin(t-1));
p = 7; % degree of polynomial fit is p-1
A = zeros(length(t),p+1);
for j=1:p+1
    A(:,j) = t.^(j-1);

```

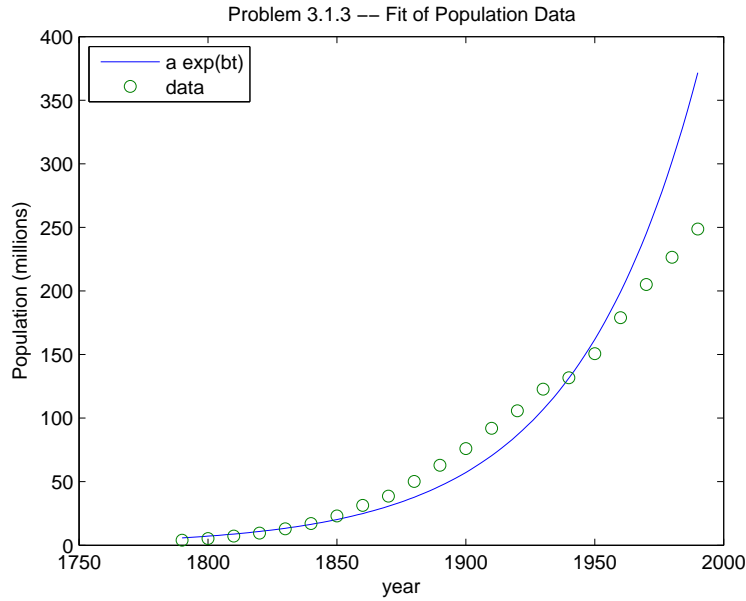


Figure 1: The linearized fit with ae^{bt} is not as good as the cubic fit.

```

end
disp(' ')
disp('Part (a) coefficients: ')
c = (A'*A)\(A'*b)
p = A*c; % evaluate polynomial at all t values in one line!

%% Part (b), trig function fit
% Trigonometric function fit uses periodic functions which have that
% property in common with the original data.
A = ones(length(t),5);
A(:,2) = cos(t);
A(:,3) = sin(t);
A(:,4) = cos(2*t);
A(:,5) = sin(2*t);

disp(' ')
disp('Part (b) coefficients: ')
d = (A'*A)\(A'*b)
trig = A*d; % again, compute the fits at all t values in one line!

%% Part (c)
% Plot the results for comparison.

plot(t,p,'-',t,trig,'--',t,b,'-.'.')

```

```

axis([0 2*pi 0 3])
xlabel('t');
legend('6th degree poly','trig fit','actual','Location','NorthEast');
title('Problem 3.1.4 -- Fit of Periodic Function');

%% Discussion
% The fit with the trigonometric functions is better near the ends
% because those functions for the interpolation are periodic, like the
% original function is. The polynomial does pretty well, but to achieve a
% specified level of error we expect that using a smaller number of trig
% functions should do the job.

```

The figure it produces shows that the periodic trig functions do a better job of fitting with less functions than the polynomial.

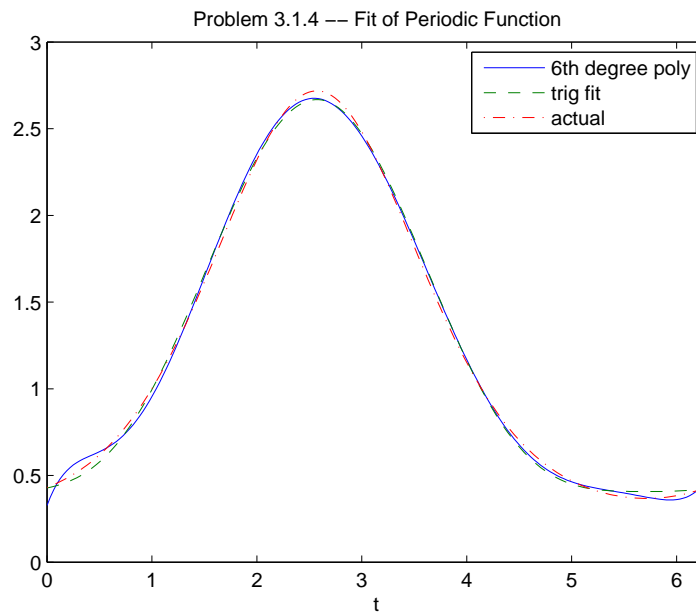


Figure 2: The periodic trig function fit is better.