Solution of Week 8 Lab (Prepared by Yuan Yin)

December 22, 2019

1 Exercise 1: Condition Numbers:

1.1 Part a:

Note that the normwise condition number of a square nonsingular matrix is $\kappa(M) = ||M|| \times ||M^{-1}||$.

(a).

$$A = \begin{bmatrix} 1001 & 1000 \\ 1 & 1 \end{bmatrix} \Rightarrow A^{-1} = \begin{bmatrix} 1 & -1000 \\ -1 & 1001 \end{bmatrix}$$

$\Rightarrow \kappa_1(A) = ||A||_1 \times ||A^{-1}||_1 = (1001 + 1) \times (|-1000| + 1001) = 1002 \times 2001;$

$\Rightarrow \kappa_\infty(A) = ||A||_\infty \times ||A^{-1}||_\infty = (1001 + 1000) \times (|-1| + 1001) = 2001 \times 1002;$

Find a nearby matrix, $A + \Delta A$, which is singular. In this case,

$$\Delta A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}$$

$\Rightarrow \frac{||\Delta A||_1}{||A||_1} = \frac{1}{1002}$ and $\frac{||\Delta A||_\infty}{||A||_\infty} = \frac{1}{2001}$

It's now clear that $\frac{||\Delta A||_1}{||A||_1} > \frac{1}{\kappa_1(A)}$ and $\frac{||\Delta A||_\infty}{||A||_\infty} > \frac{1}{\kappa_\infty(A)}$.

(b).

$$B = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix} \Rightarrow B^{-1} = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix}$$

$\Rightarrow \kappa_1(B) = ||B||_1 \times ||B^{-1}||_1 = 0.1 \times 10 = 1;$

$\Rightarrow \kappa_\infty(B) = ||B||_\infty \times ||B^{-1}||_\infty = 0.1 \times 10 = 1;$

Find a nearby matrix, $B + \Delta A$, which is singular. In this case,

$$\Delta B = \begin{bmatrix} -0.1 & 0 \\ 0 & 0 \end{bmatrix}$$

$\Rightarrow \frac{||\Delta B||_1}{||B||_1} = \frac{0.1}{0.1}$ and $\frac{||\Delta B||_\infty}{||B||_\infty} = \frac{0.1}{0.1}$

It's now clear that $\frac{||\Delta B||_1}{||B||_1} \geq \frac{1}{\kappa_1(B)}$ and $\frac{||\Delta B||_\infty}{||B||_\infty} \geq \frac{1}{\kappa_\infty(B)}$.

1.2 Part b:

Run'CondEgs.m' and 'ErrChol.m'.

1.3 Part c:

[1]:
```
%%file AdaptedCondEgs.m

function AdaptedCondEgs

for n = [4 8 12 16]

   A = pascal(n);
   fprintf('cond(pascal(%2d)) = %8.4e\n',n,cond(A));

   disp('True solution is vector of ones. Computed solution =')

   xTrue = ones(n,1);
   b = A * xTrue;
   format long
%    [L,U,P] = lu(A);
%    x = U\(L\(P*b))
   [Q,R] = qr(A);
   x = R\(Q'*b)

   format short
   relerr = norm(x - xTrue)/norm(xTrue);
   fprintf('Relative error = %8.4e\n',relerr);

   bound = eps * cond(A);
   fprintf('Predicted value = eps * cond(A) = %8.4e\n',bound);

   r = b - A * x;
   residual  = norm(r) / (norm(A) * norm(x));
   fprintf('Relative Residual = %8.4e\n',residual);

   % pause
   %  input('Strike Any Key to Continue.');
end

end
```



```
Created file '/Users/RebeccaYinYuan/MAST30028 Tutorial Answers Yuan Yin/WEEK
8/AdaptedCondEgs.m'.
```

[2]:
```
AdaptedCondEgs
```

```
cond(pascal( 4)) = 6.9194e+02
```

```
True solution is vector of ones. Computed solution =

x =

   1.000000000000002
   0.999999999999996
   1.000000000000002
   1.000000000000000

Relative error = 2.7744e-15
Predicted value = eps * cond(A) = 1.5364e-13
Relative Residual = 1.6883e-17
cond(pascal( 8)) = 2.0645e+07
True solution is vector of ones. Computed solution =

x =

   1.000000000002498
   0.999999999955526
   1.000000000158517
   0.999999999720579
   1.000000000284056
   0.999999999829995
   1.000000000055926
   0.999999999992166

Relative error = 1.6506e-10
Predicted value = eps * cond(A) = 4.5841e-09
Relative Residual = 1.0911e-16
cond(pascal(12)) = 8.7639e+11
True solution is vector of ones. Computed solution =

x =

   1.000000022911023
   0.999999764375295
   1.000001117062836
   0.999996810616196
   1.000006082464622
   0.999991870102494
   1.000007769343390
   0.999994691802608
   1.000002540985069
   0.999999188342651
   1.000000155709023
   0.999999986408738

Relative error = 4.1858e-06
```

```
Predicted value = eps * cond(A) = 1.9460e-04
Relative Residual = 5.4595e-17
cond(pascal(16)) = 4.2464e+16
True solution is vector of ones. Computed solution =

x =

   0.999822183060471
   1.002484348340855
   0.983690501395390
   1.066611355944322
   0.810918697179950
   1.394877212467325
   0.373497990205333
   1.768701852395546
   0.264781479577272
   1.548031849592967
   0.684290211221038
   1.138014933908382
   0.955686478482983
   1.009864259838801
   0.998638901914901
   1.000087751112071

Relative error = 3.6584e-01
Predicted value = eps * cond(A) = 9.4289e+00
Relative Residual = 5.9707e-17
```


[3]:

```
%%file AdaptedErrChol.m

function AdaptedErrChol

clc
disp('  n      cond(A)        relerr        relresidual ')
disp('-----------------------------------------------')

for n = 2 : 12

   A = hilb(n);
   b = randn(n,1);

   R = chol(A);
   x = R\(R'\b);

   condA = cond(A);

   xTrue = invhilb(n) * b;
```

```
    relerr = norm(x - xTrue) / norm(xTrue);
    r = b - A * x;
    residual  = norm(r) / (norm(A) * norm(x));

    fprintf(' %2.0f   %10.3e    %10.3e    %10.3e\n', n, condA, relerr, residual);

end

end
```

Created file '/Users/RebeccaYinYuan/MAST30028 Tutorial Answers Yuan Yin/WEEK 8/AdaptedErrChol.m'.

[4]:
```
AdaptedErrChol
```

```
  n      cond(A)        relerr        relresidual
-----------------------------------------------
  2    1.928e+01     1.913e-16     4.218e-18
  3    5.241e+02     1.520e-15     6.220e-18
  4    1.551e+04     8.748e-14     6.629e-18
  5    4.766e+05     1.044e-12     1.309e-17
  6    1.495e+07     8.310e-11     6.513e-18
  7    4.754e+08     4.042e-09     9.279e-18
  8    1.526e+10     1.175e-07     7.131e-18
  9    4.932e+11     1.914e-06     9.089e-18
 10    1.603e+13     9.551e-05     6.802e-18
 11    5.220e+14     3.072e-03     8.725e-18
 12    1.621e+16     9.489e-02     1.001e-17
```


2 Exercise 2: Data Fitting:

2.1 Linear Models:

(a).

This is a linear model in c_1 and c_2.

(b).

This is a separable model: c_1 is linear, c_4 is nonlinear, while c_2 appears both linearly and nonlinearly.

(c).

This model is linear in c_1 and nonlinear in c_2. However, this model can be transformed into a linear one:

$$y = c_1 e^{c_2 x} \Rightarrow log(y) = log(c_1) + c_2 x$$

By setting $c_3 = log(c_1)$, we have $log(y) = c_3 + c_2 x$, which is now a linear model in c_3 and c_2!

(d).

This model is linear in c_1 and nonlinear in c_2. However, using the similar method as we did in part (c), we have:

$$log(y) = log(c_1) + c_2 log(x)$$

By setting $c_3 = log(c_1)$, the model has been transformed into a linear model in c_2 and c_3.

(e).

This is a nonlinear model in c_1 and c_2.

(f).

This is a linear model in c_1, c_2, and c_3.

2.2 Curve Fitting:

(a).

- 'Pchip' and 'Spline' appear to be interpolating the data while 'Polynomial' and 'Exponential' are fitting the data;
- If you click on the 'error estimate' button, you will see that $Polynomial < Exponential < Pchip < Spline$.

(b).

Before changing the data point, the predicted values for 2020 using different fitting models are:

Polynomial: 341.125; Pchip: 331.268; Spline: 311.820; Exponential: 363.607.

After creating an outlier, the predicted values are:

Polynomial: 349.206; Pchip: 331.268; Spline: 306.848; Exponential: 339.778.

As one can see—— 'Pchip': not sensitive; 'Exponential': sensitive.

2.3 Do It Yourself:

[5]:
```
%%file Tute8CurveFitting.m

function Tute8CurveFitting

clc;

format long

% Plot the Data Points:
t = 1 : 0.1 : 3;
y = 1 + 2 * sin(3 * t) + 0.5 * rand(size(t));
plot(t, y, 'o', 'MarkerSize',8);
hold on;

% Using Least Square to fit the data to the given model:
n = length(t);
A = [ones(n, 1), (sin(3 * t))'];
```

```
x = A \ y';
c1 = x(1)
c2 = x(2)
curve_fitting = c1 + c2.* sin(3 * t);
plot(t, curve_fitting, 'r', 'LineWidth',2);
hold on;


% Try 'lsqcurvefit':
FUN = @(d, t) d(1) + d(2).* sin(3 * t);
X0 = [1, 4];
X = lsqcurvefit(FUN,X0,t,y);
X(1)
X(2)
lsq_curve = X(1) + X(2) * sin(3 * t);
plot(t, lsq_curve, 'b--', 'LineWidth',2);


end
```



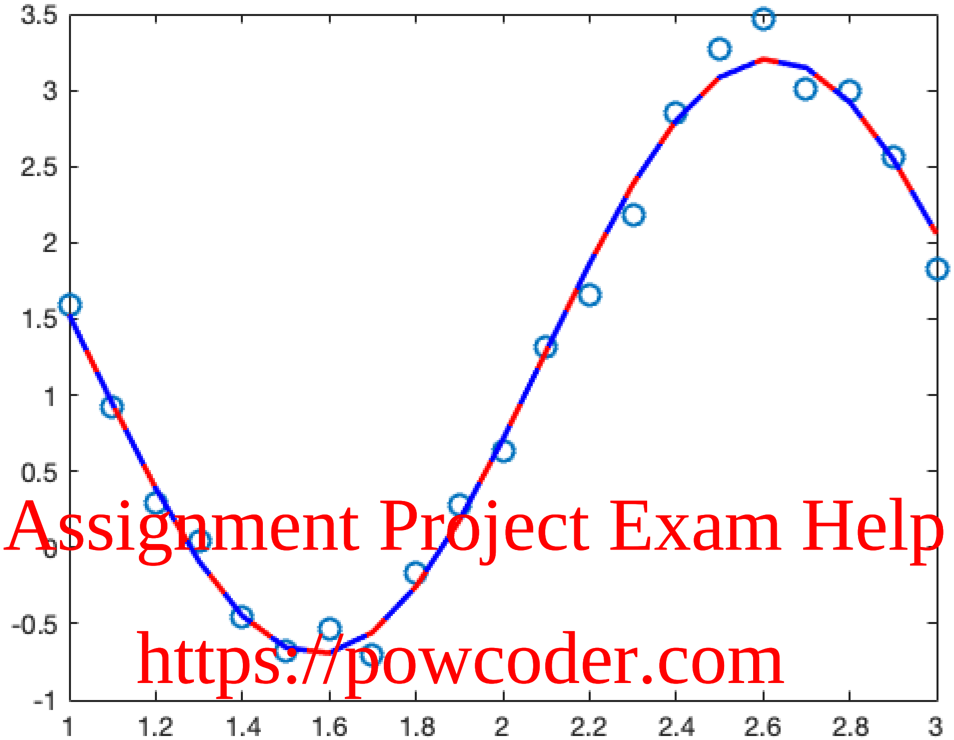
```
Created file '/Users/RebeccaYinYuan/MAST30028 Tutorial Answers Yuan Yin/WEEK
8/Tute8CurveFitting.m'.
```

[6]:
```
Tute8CurveFitting
```



```
c1 =

   1.252959720631116


c2 =

   1.953339998219989


Local minimum found.

Optimization completed because the size of the gradient is less than
the value of the optimality tolerance.


ans =

   1.252959720255180


ans =
```

```
1.953339993089838
```



Since $y = 1 + 2 \times sin(3t) + 0.5 \times randn(size(t))$, we can see that the underlying model is $y = 1 + 2sint(3t)$ with perturbation $0.5 \times randn(size(t))$. I.e., our underlying values for c_1 and c_2 are 1, 2 respectively. One can see that our calculated coefficients $c_{1calc} = 1.2628$ and $c_{2calc} = 1.9883$ are relatively close to c_1 and c_2.

Note that I also use 'lsqcurvefit' command to fit the data. You can check how to use it and compare which method is better. \Also, this file shows how to set 'MarkerSize' and 'LineWidth' using 'plot' command. Type 'help plot' for more information!