Weimin Gao

9/22/2017

CSc 301

HW#1

Read Section 1.3.2 Random Processes (p. 34 – 37) Do Problem P1.3.2 (p. 40)

Read Section 1.5.2 Numerical Differentiation (p.54 – 57) Do Problem P1.5.1 (p. 57)

Read Section 1.6.1 Three-digit Arithmetic (p. 61 – 62) Do Problem P1.6.1 (p. 63)

Problem P1.3.2

My script file:

```
%Script File: Probability of complex roots

close all
rand('seed',.12345);
n = 100:100:800;
Pro1 = zeros(8,1);
Pro2 = zeros(8,1);
dispN = zeros(8,1);
for k=1:8
%P1: The coefficients are random variables with uniform(0,1).
    complexRoots1 = 0;
    a1=rand(n(k),1);
    b1=rand(n(k),1);
    c1=rand(n(k),1);
    ans1=(-b1+sqrt( b1.^2-4.*a1.*c1))./2.*a1;%First root of equation.
    root1=imag(ans1);%Find the imaginary part of the first root.
    ans2=(-b1-sqrt( b1.^2-4.*a1.*c1))./2.*a1;%second root of equation.
    root2=imag(ans2);%Find the imaginary part of the second root.
%P2: The coefficients are random variables with normal(0,1).
    complexRoots2 = 0;
    a2=rand(n(k),1);
    b2=rand(n(k),1);
    c2=rand(n(k),1);
    ans3=(-b2+sqrt( b2.^2-4.*a2.*c2))./2.*a2;
    root3=imag(ans3);
    ans4=(-b2-sqrt( b2.^2-4.*a2.*c2))./2.*a2;
    root4=imag(ans3);
    for i=1:n(k)
        if(root1(i)~=0) complexRoots1=complexRoots1+1; end
        %Check: if the imaginary part is not equal to 0, the root is
complex.
        if(root2(i)~=0) complexRoots1=complexRoots1+1; end
        %also check the imaginary part of second root.
        if(root3(i)~=0) complexRoots2=complexRoots2+1; end
        if(root4(i)~=0) complexRoots2=complexRoots2+1; end
```

```
    end
    dispN(k)=n(k);
    Pro1(k)=complexRoots1/(n(k)*2)*100; %calculate probability.
    Pro2(k)=complexRoots2/(n(k)*2)*100;
end
fprintf('The probability P1(%i): %5.3f%%     The probability
P2(%i): %5.3f%%\n',[dispN, Pro1,dispN, Pro2]')
```

The output is: (*seed* is 0.12345)

```
The probability P1(100): 71.000%     The probability P2(100): 81.000%
The probability P1(200): 70.500%     The probability P2(200): 74.000%
The probability P1(300): 75.333%     The probability P2(300): 75.333%
The probability P1(400): 75.500%     The probability P2(400): 76.000%
The probability P1(500): 75.800%     The probability P2(500): 77.600%
The probability P1(600): 77.500%     The probability P2(600): 72.500%
The probability P1(700): 71.429%     The probability P2(700): 74.286%
The probability P1(800): 75.375%     The probability P2(800): 76.375%
```

If I change *seed* from 0.12345 to 123, the output is:

```
The probability P1(100): 75.000%     The probability P2(100): 70.000%
The probability P1(200): 77.500%     The probability P2(200): 79.000%
The probability P1(300): 72.000%     The probability P2(300): 77.000%
The probability P1(400): 75.500%     The probability P2(400): 71.500%
The probability P1(500): 67.000%     The probability P2(500): 72.200%
The probability P1(600): 74.167%     The probability P2(600): 75.833%
The probability P1(700): 73.429%     The probability P2(700): 75.857%
The probability P1(800): 77.750%     The probability P2(800): 73.875%
```

Problem P1.5.1

Weimin Gao

Problem P1.5.1:

$$f(x) = f(x) + f'(x)(x-a) + \frac{f''(x)}{2!}(x-a)^2 + \frac{f'''(\eta)}{3!}(x-a)^3$$

$$f(a+h) = f(x) + f'(x)h + \frac{f''(x)}{2!}h^2 + \frac{f'''(\eta)}{3!}h^3$$

$$f(a-h) = f(x) - f'(x)h + \frac{f''(x)}{2!}h^2 - \frac{f'''(\eta)}{3!}h^3$$

$$f(a+h) - f(a-h) = 2f'(x)h + 2\frac{f'''(\eta)}{3!}h^3$$

$$\frac{f(a+h) - f(a-h)}{2h} = f'(x) + \frac{f'''(\eta)}{3!}h^3$$

$$\boxed{f'(x) \approx \frac{f(a+h) - f(a-h)}{2h}} + \frac{f'''(\eta)}{3!}h^2$$

$$\Rightarrow \quad C(h) = \frac{f(a+h) - f(a-h)}{2h}$$

$$\boxed{f'(x) \approx \frac{f(a+h) - f(a-h)}{2h}} + \frac{f'''(\eta)}{3!}h^2$$

$$\Rightarrow \quad C(h) = \frac{f(a+h) - f(a-h)}{2h}$$

$$|C(h) - f'(a)| < \frac{h^2}{6}|f'''(\eta)|$$

$$|C(h) - f'(a)| \approx \frac{h^2}{6}|f'''(\eta)| + \boxed{\frac{2\varepsilon}{h}}$$

The right-hand side incorporates the "truncation error" due to calculus and the computation error due to roundoff.

- If we have a upper bound on the third derivative of the form $|f'''(x)| \le M_3$, then:

$$|C(h) - f'(a)| \le \frac{M_3}{6}h^2$$

- If the absolute error in a computed function evalution is bounded by δ, then:

$$err\, C(h) = M_3\frac{h^2}{6} + \frac{2\delta}{h}$$

$$err\,C(h) = M_3\frac{h^2}{6} + \frac{2\delta}{h}$$

is a reasonable model for total error. This quantity is minimized if:

$$|f'''(x)| \le M_3$$

$$F(h) = C\cdot\frac{h^2}{6} + \frac{2\delta}{h} \qquad *(C = M_3)$$

$$\text{argument}\left(C\cdot\frac{h^2}{6} + \frac{2\delta}{h}\right)$$

$$C = -\frac{2\delta}{h}\cdot\frac{6}{h^2}$$

$$C = -\frac{12\delta}{h^3}$$

$$h^3 = -\frac{12\delta}{C}$$

$$h(opt) = \left(-\frac{12\delta}{C}\right)^{\frac{1}{3}} \text{ or } \sqrt[3]{-\frac{12\delta}{C}}$$

$$\Rightarrow h(opt) = \left(-\frac{12\delta}{M_3}\right)^{\frac{1}{3}} \text{ or } \sqrt[3]{-\frac{12\delta}{M_3}}$$

$$h^3 = -\frac{12\delta}{C}$$

$$h(opt) = \left(-\frac{12\delta}{C}\right)^{\frac{1}{3}} \text{ or } \sqrt[3]{-\frac{12\delta}{C}}$$

$$\Rightarrow h(opt) = \left(-\frac{12\delta}{M_3}\right)^{\frac{1}{3}} \text{ or } \sqrt[3]{-\frac{12\delta}{M_3}}$$

$$err\,C(h(opt)) = \left(M_3\frac{\left(-\frac{12\delta}{M_3}\right)^{\frac{2}{3}}}{6} + \frac{2\delta}{\left(\frac{-12\delta}{M_3}\right)^{\frac{1}{3}}}\right)$$

$$\left(\frac{-12\delta}{M_3}\right)^{\frac{1}{3}} err\,C(h(opt)) = \left(M_3\frac{\left(-\frac{12\delta}{M_3}\right)^{\frac{2}{3}}}{6} + \frac{2\delta}{\left(\frac{-12\delta}{M_3}\right)^{\frac{1}{3}}}\right)\cdot\left(\frac{-12\delta}{M_3}\right)^{\frac{1}{3}}$$

$$\left(\frac{-12\delta}{M_3}\right)^{\frac{1}{3}} err\,C(h(opt)) = \cancel{M_3}\frac{\left(-\frac{12\delta}{\cancel{M_3}}\right)}{6} + 2\delta$$

$$= -2\delta + 2\delta$$

$$\left(\frac{-12\delta}{M_3}\right)^{\frac{1}{3}} err\,C(h(opt)) = 0$$

$$\Rightarrow \boxed{err\,C(h(opt)) = 0}$$

Then I code a Derivative function for M3:

```
%Problem 2, function [d,err] = Derivative(fname,a,delta,M3)
function [d,err] = Derivative(fname,a,delta,M3)

if nargin <= 4
   % No derivative bound supplied, so assume the
   % third derivative bound is 1.
   M3 = 1;
end
if nargin == 2
   % No function evaluation error supplied, so
   % set delta to eps.
   delta = eps;
end
% Compute optimum h and divided difference
hopt = (-12*delta/M3)^1/3;
d   = (feval(fname,a+hopt) - feval(fname,a-hopt))/(2*hopt); %new
approximations.
err = 0; %the error of approximations C(h).
```

Problem P1.6.1

Function of Represent, Convert and Float show in next few pages.

This is my dot3 function:

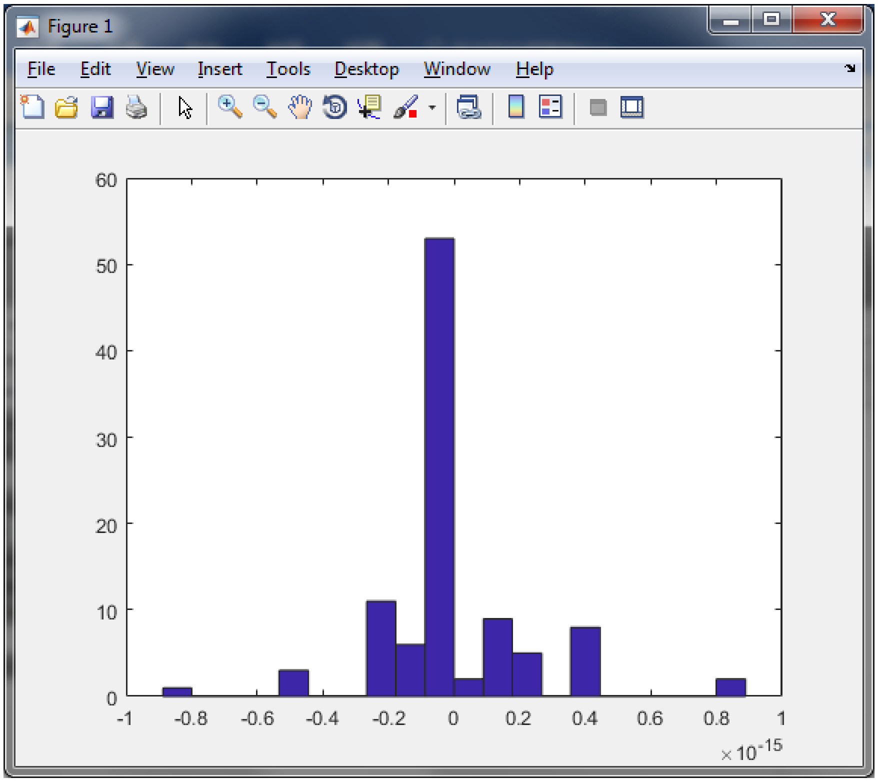
```
function s = dot3(x,y)
   for k=1:5
     xay=Represent(x(k));
     yay=Represent(y(k));
     s=Float(xay,yay,'*');
     s=Convert(s);
   end
```

I convert s inside of the function, so in script file, I don’t convert dot3 again.

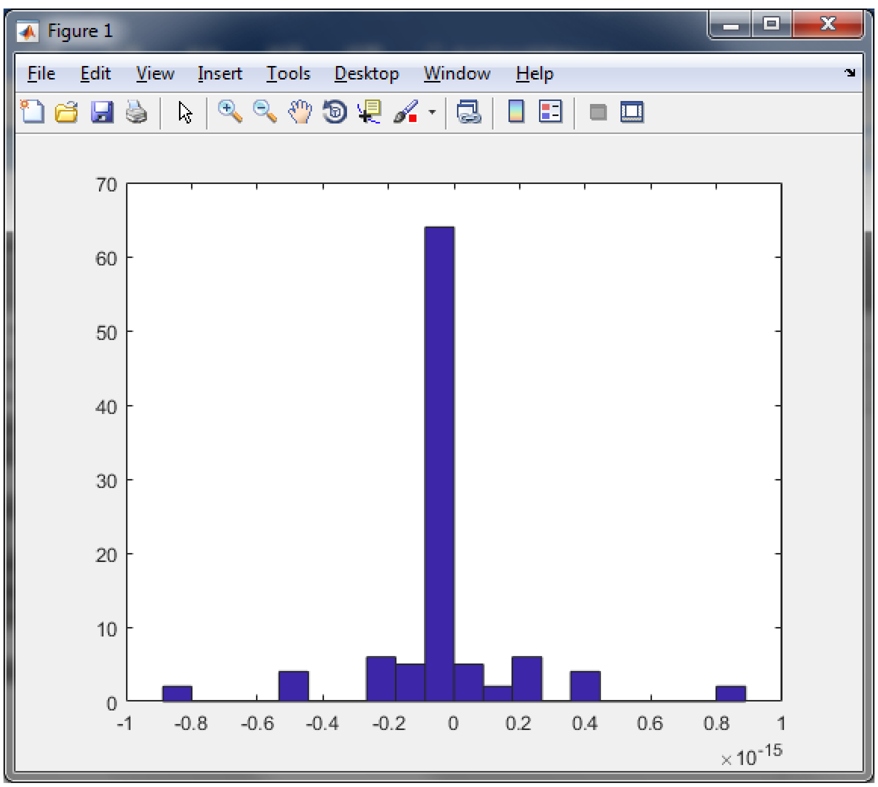
Below is my script file:

```
%Script File: Histograms of the error (dot3)
%Histograms of randn(5,1) with 20 bins
close all
err = zeros(100:1);
for k=1:100
    x = randn(5,1);
    y = randn(5,1);
    err(k) = x'*y-dot(x,y);
end
hist (err,20)
```

First time outcome:



Second time outcome:



Function for Problem P1.6.1:

Represent

```
function f = Represent(x)
% f = Represent(x)
% Yields a 3-digit mantissa floating point representation of f:
%
%    f.mSignBit   mantissa sign bit (0 if x>=0, 1 otherwise)
%    f.m          mantissa (= f.m(1)/10 + f.m(2)/100 + f.m(3)/1000)
%    f.eSignBit   the exponent sign bit (0 if exponent nonnegative, 1
otherwise)
%    f.e          the exponent (-9<=f.e<=9)
%
% If x is out side of [-.999*10^9,.999*10^9], f.m is set to inf.
% If x is in the range [-.100*10^-9,.100*10^-9] f is the representation
of zero
% in which both sign bits are 0, e is zero, and m = [0 0 0].

f = struct('mSignBit',0,'m',[0 0 0],'eSignBit',0,'e',0);

if abs(x)<.100*10^-9
   % Underflow. Return 0
   return
end

if x>.999*10^9
   % Overflow. Return inf
   f.m = inf;
   return
end
if x<-.999*10^9
   % Overflow. Return -inf
   f.mSignBit = 1;
   f.m = inf;
   return
end

% Set the mantissa sign bit
if x>0
   f.mSignBit = 0;
else
   f.mSignBit = 1;
end

% Determine m and e so .1 <= m < 1 and abs(x) = m*10^e
m = abs(x); e = 0;
while m >= 1,  m = m/10; e = e+1; end
while m < 1/10,m = 10*m; e = e-1; end

% Determine nearest integer to 1000m
z = round(1000*m);
% Set the mantissa
f.m(1) = floor(z/100);
f.m(2) = floor((z - f.m(1)*100)/10);
```

```
f.m(3) = z - f.m(1)*100 - f.m(2)*10;
% Set the exponent and its sign bit.
if e>=0
   f.eSignBit = 0;
   f.e = e;
else
   f.eSignBit = 1;
   f.e = -e;
end
```

<u>Convert</u>

```
  function x = Convert(f)
% x = Convert(f)
% f is a is a representation of a 3-digit floating point number. (For
details
% type help represent. x is the value of v.

% Overflow situations
if (f.m == inf) & (f.mSignBit==0)
   x = inf;
   return
end
if (f.m == inf) & (f.mSignBit==1)
   x = -inf;
   return
end

% Mantissa value
mValue = (100*f.m(1) + 10*f.m(2) + f.m(3))/1000;
if f.mSignBit==1
   mValue = -mValue;
end

% Exponent value
eValue = f.e;
if f.eSignBit==1
   eValue = -eValue;
end

x = mValue * 10^eValue;
```

Float

```
function z = Float(x,y,op)
% z = Float(x,y,op)
% x and y are representations of a 3-digit floating point number. (For
details
% type help represent.
% op is one of the strings '+', '-', '*', or '/'.
% z is the 3-digit floating point representation of x op y.

sx = num2str(Convert(x));
sy = num2str(Convert(y));
z = Represent(eval(['(' sx ')' op '(' sy ')' ]));
```