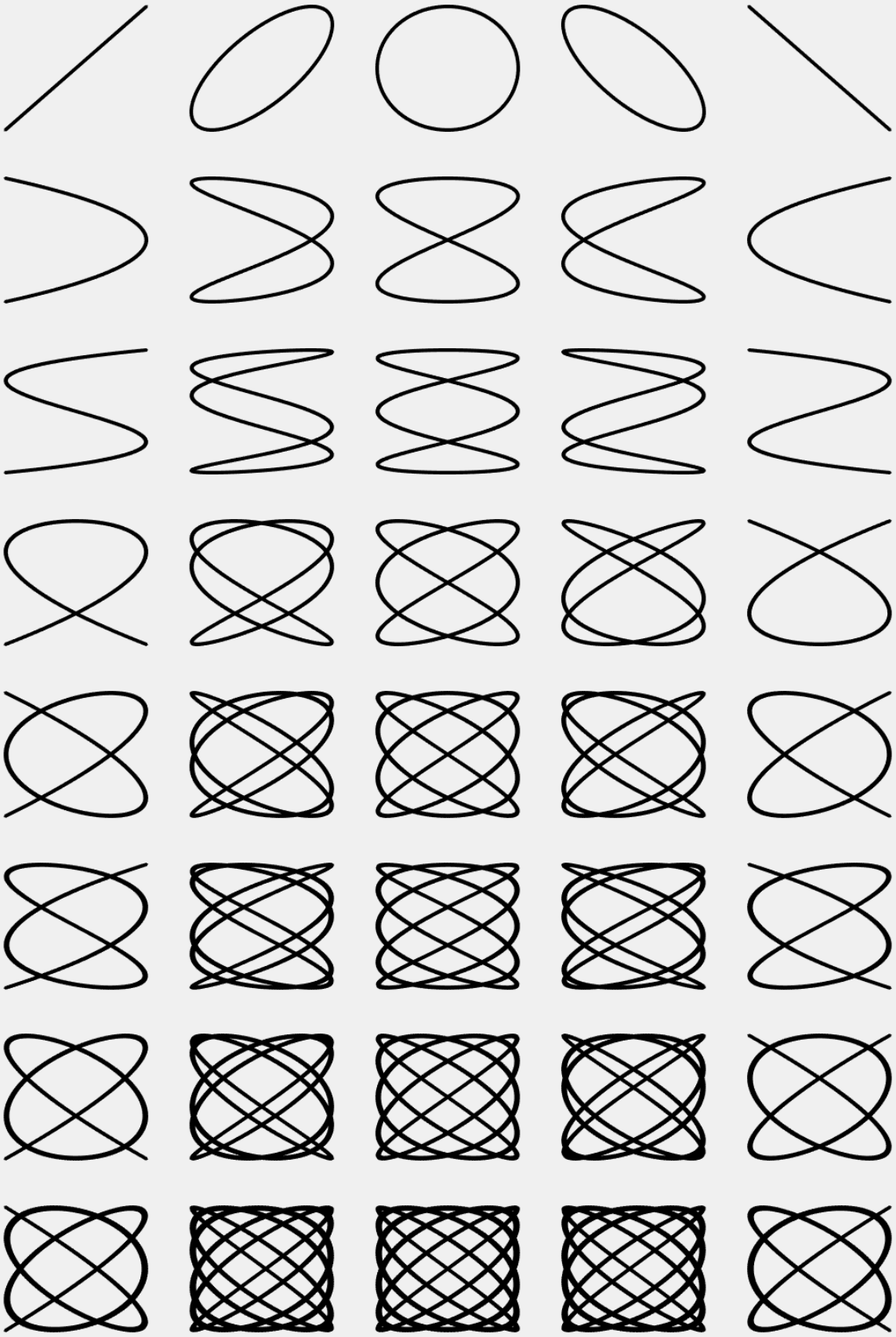
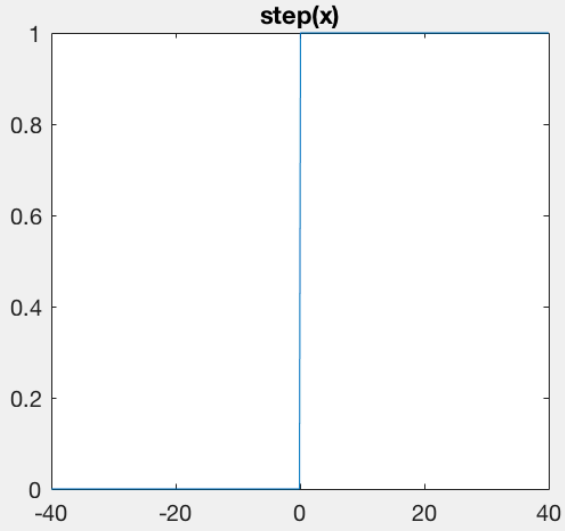
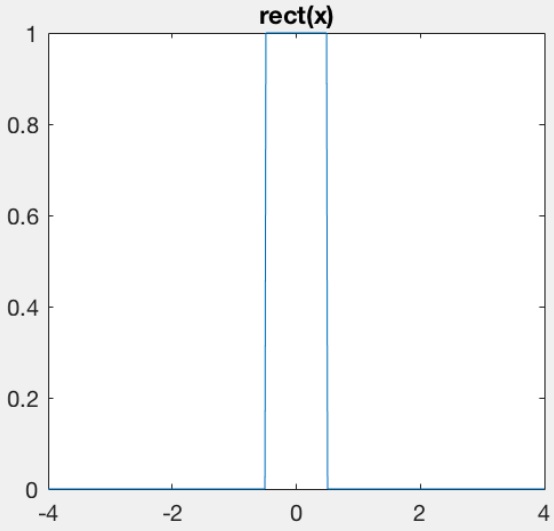
```
% ### EXlissajou2B.m ###      2018.02.14      C. Bergevin

% Goal: To make array of lissajou figs (consistent w/ "2018.02 lissajou.pdf")

% Notes:
% - comparing to "2018.02 lissajou.pdf", this code seems to indicate the
% phase values used for the columns were not consistent for the different
% ratios (presumably to help w/ artistic license....)
% - ratio "b/a" would be In.freqs(2)/In.freqs(1)
% - syntax below also shows how to change fonts (type "listfonts" to see
% what is available

clear
% ====================================================
In.tMax= 100;   % max. time to compute out too {100}
In.tS= 1000;    % total # of time steps {1000}
In.delta= pi*[0 0.25 0.5 0.75 1;  % phase diff. vals
              0.75 0.8750 1 1.125 1.25;
              1 0.25 0.5 0.75 0;
              0.5 1.25 1 0.75 1.5;
              0.8750 0.8125 0.75 0.6875 0.625;
              1 0.25 0.5 0.75 0;
              1.5 0.75 1 1.25 0.5;
              0.25 0.515 0.5 0.485 0.75];
In.freqs= [1 1;1 2;1 3;2 3;3 4;3 5;4 5;5 6];    % freqs (a and b) for two
sinusoids
In.mag= [1 1];  % mags. (A and B) for two sinusoids {[1 1]}
In.flip= [1 -1 -1 1 1 1 1 1];  % flip for x re "2018.02 lissajou.pdf"
% ====================================================
%0.25 0.375 0.5 0.625 0.75
% ---
t= linspace(0,In.tMax,In.tS);
figure(1); clf; hold on; title('Sound Vibrations')
nH= size(In.delta,2);  % # of phases to compute/plot
nV= size(In.freqs,1);  % # of ratios to compute/plot
indx= 1;    % dummy indexer for subplot
% ---
for nn=1:nV
    for mm=1:nH
        x= In.mag(1)*sin(In.freqs(nn,1)*t + In.delta(nn,mm));
        y= In.mag(2)*sin(In.freqs(nn,2)*t);
        subplot(nV,nH,indx);
        plot(y,x,'k-','LineWidth',2);
        indx= indx+ 1;
        ax = gca; ax.Visible = 'off';
    end

end
% --- snippet below is for the title (borrowed some bits from mathworks.com)
ha= axes('Position',[-0.135 -0.02 1 1],'Xlim',[0 1],'Ylim',[0
1],'Box','off','Visible','off','Units',...
    'normalized','clipping','off');
text(0.5,0.98,'Sound Vibrations','FontSize',
24,'FontWeight','bold','FontName','Monotype Corsiva');
```

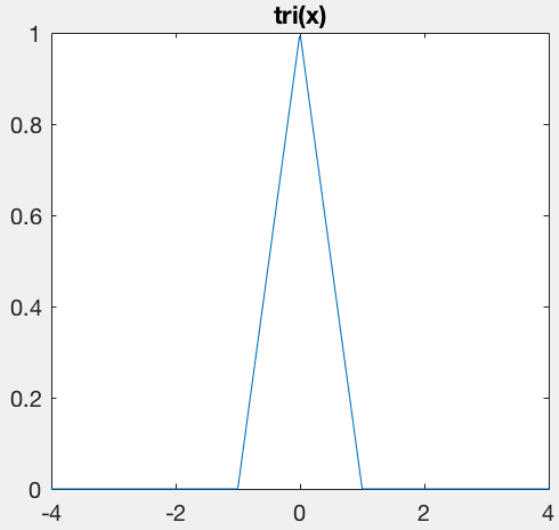
Sound Vibrations

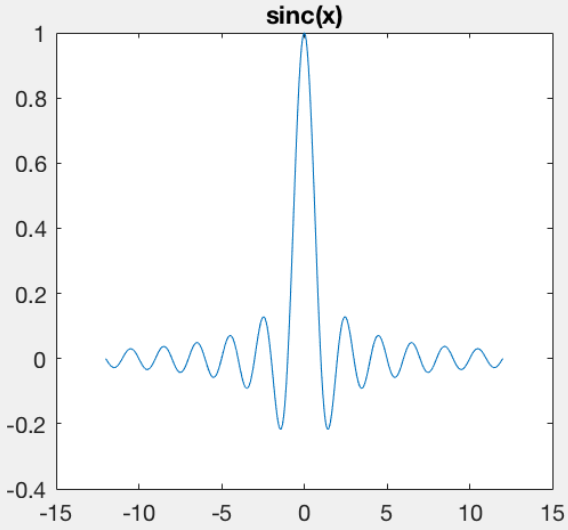


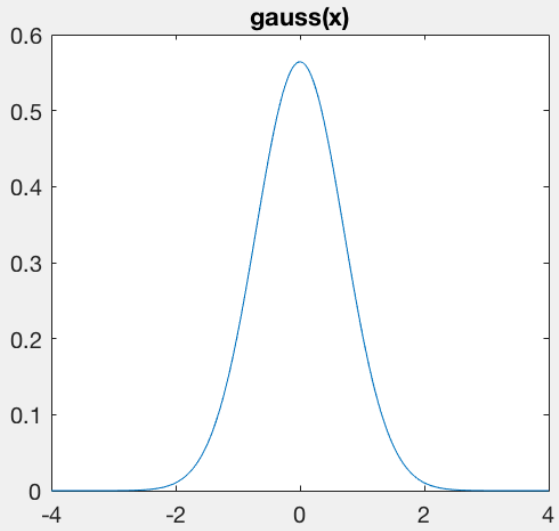
```
% ### EXcreateSPcurves.m ###      2017.01.30

% code to plot the curves shown in Fig.4.1 of Buzug (2008) re "fundamental
% functions of signal processing"

% ============================================================
P.N= 500;       % # of points for each plot
% ============================================================
% --- [set up relevant x-values]
x1= linspace(-40,40,P.N);
x2= linspace(-4,4,P.N);
x3= linspace(-12,12,P.N);
% --- [create the functions]
y1= x1>=0;  % step function via a conditional
y2= (x2>=-0.5 & x2<=0.5);  % rectangle function via a conditional
y3= 1-abs(x2);
y3= y3.*(x2>=-1 & x2<=1);  % triangle function via a conditional, then
multiplication
y4= sin(pi*x3)./(pi*x3);
y5= exp(-x2.^2)./sqrt(pi);
temp= find(x1>0);
y6= zeros(P.N,1); y6(temp(1)-1)= 100000000000000;
% --- [plot]
h0= figure(1); clf;
h0.Name= 'Reproduction of Fig.4.1 from Buzug (2008)';
subplot(321); plot(x1,y1); title('step(x)')
subplot(322); plot(x2,y2); title('rect(x)')
subplot(323); plot(x2,y3); title('tri(x)')
subplot(324); plot(x3,y4); title('sinc(x)'); ylim([-0.4 1]);
subplot(325); plot(x2,y5); title('gauss(x)')
subplot(326); plot(x1,y6); title('\delta(x)')
```

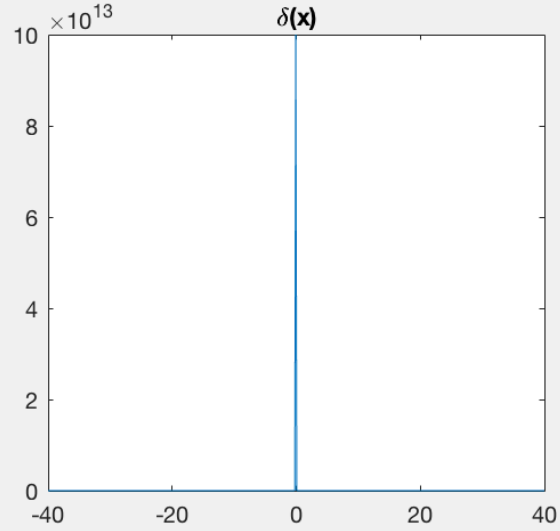

step(x)
1
0.8
0.6
0.4
0.2
0
-40
-20
0
20
40


rect(x)
1
0.8
0.6
0.4
0.2
0
-4
-2
0
2
4


tri(x)
1
0.8
0.6
0.4
0.2
0
-4
-2
0
2
4

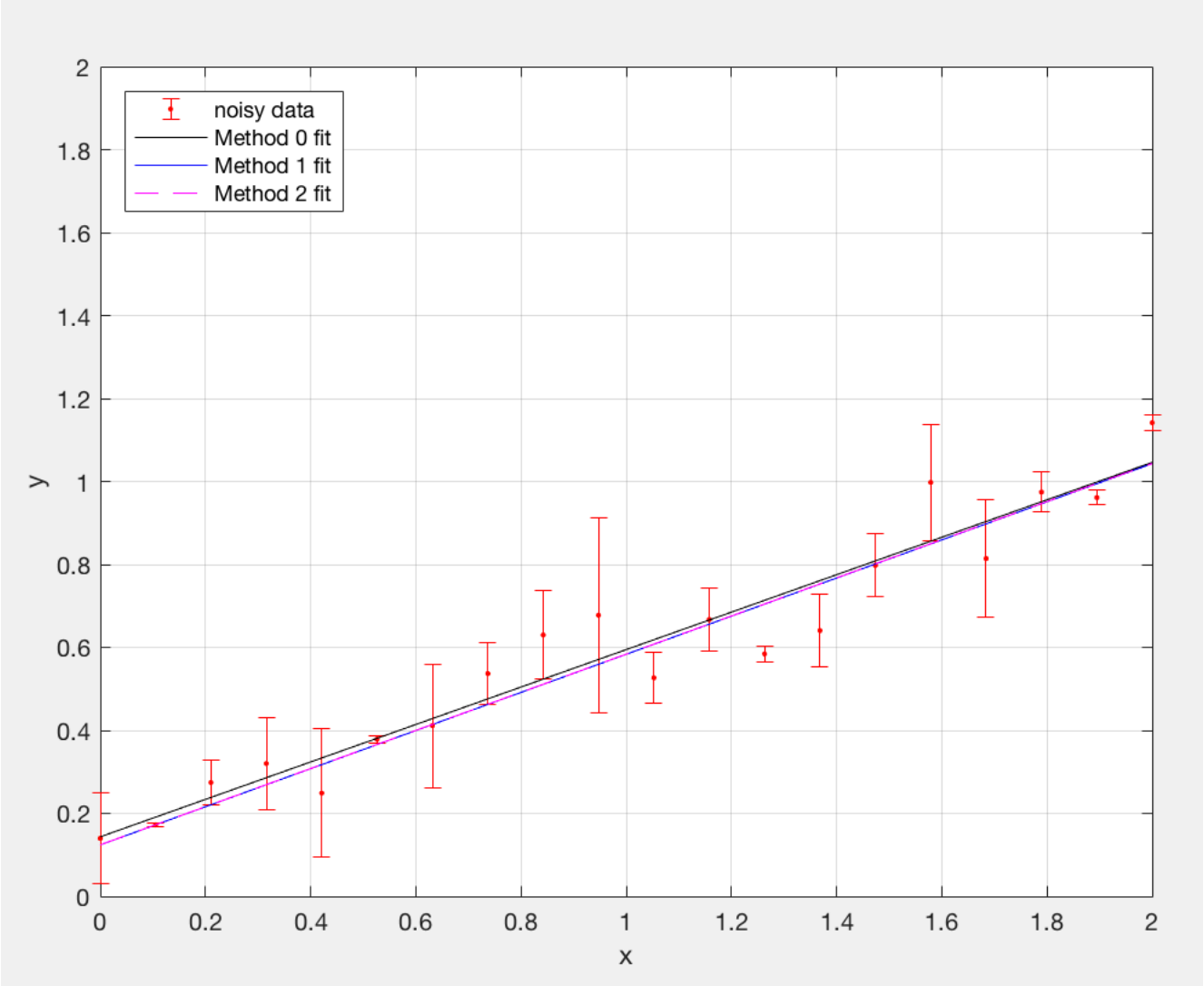

sinc(x)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-15
-10
-5
0
5
10
15


gauss(x)
0.6
0.5
0.4
0.3
0.2
0.1
0
-4
-2
0
2
4


×10^13
δ(x)
10
8
6
4
2
0
-40
-20
0
20
40

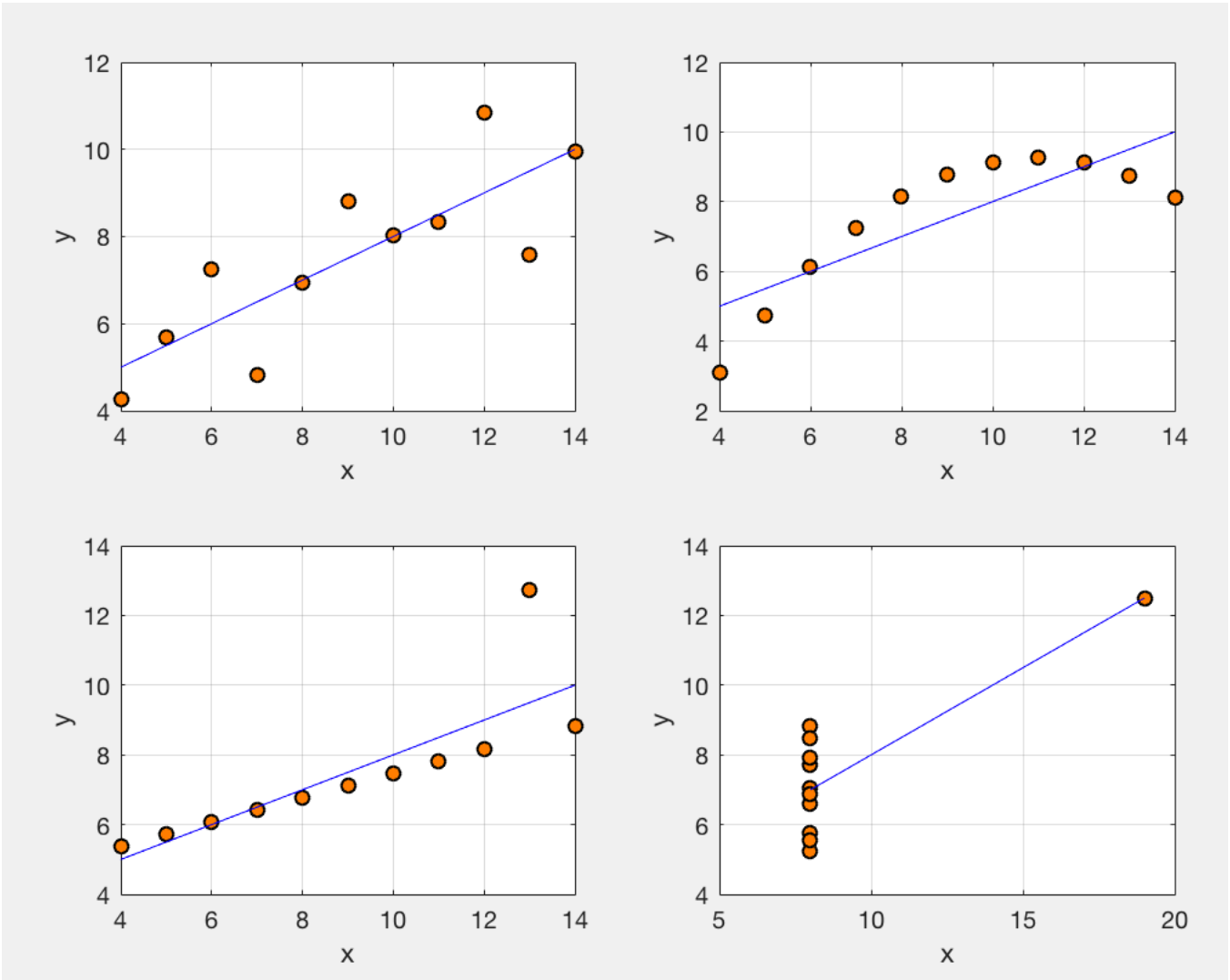
```
% ### EXregressStraightLine2.m ###     10.04.14
% Fit a straight line: y(x) = a + b*x (i.e., find optimal values for a and
% b for a linear function to a given set of data) using three methods:
% Method 1. Matlab's built-in blackbox function polyfit.m
% Method 2. exact analytic solution (via Bevington)
% Method 3. brute force minimizing chi-squared via a grid-search method (user
specifies grid)
% --> *** assumes random standard deviation for each point ***

clear; figure(1); clf;
% -------------------------------------
% specify range of x-values
xMin= 0;    % min.
xMax= 2;    % max.
xNum= 20;  % number of 'data' points
%  choose parameters of linear function (y=aD+bD*x+noise)
aD= 0.15;   % intercept
bD= 0.44;   % slope
noiseF= 0.1;   % noise factor {0.1}
noiseSD= 0.1;  % noise factor for standard deviation for each point
% define 'grid' range for Method 2 (i.e., one needs to 'guess' here!)
gridA= linspace(0.1,0.4,100);   % intercept
gridB= linspace(0.1,0.5,100);   % slope

% -------------------------------------
data.x= linspace(xMin,xMax,xNum);
N= numel(data.x);   % total number of points
data.y= aD+ bD*data.x + noiseF*randn(N,1)'; % determine noisy (straight) line
data.SD= noiseSD* randn(N,1)';
%data.SD= ones(N,1)';
XX= data.SD.^2; % dummy variable to save on coding syntax
disp(['==================================']);
disp(['Base function: y= a+b*x= ',num2str(aD),' + ',num2str(bD),'*x (+
Gaussian noise)']);
% ------------
% Method 0: Exact linear regression *IGNORING UNCERTAINTY* (see ch.6 from
Bevington, eqns.6.13 and 6.23)
Delta= N*sum(data.x.^2)-sum(data.x)^2;
a0= 1/Delta*(sum(data.x.^2)*sum(data.y)- sum(data.x)*sum(data.x.*data.y));
b0= 1/Delta*(N*sum(data.x.*data.y)- sum(data.x)*sum(data.y));
sigmaA0= sqrt(1/Delta*sum(data.x.^2));
sigmaB0= sqrt(N/Delta);
% also calculate r^2 (coefficient of determination) via
% http://en.wikipedia.org/wiki/Coefficient_of_determination
meanY= sum(data.y)/N;
bottom0= sum((data.y-meanY).^2);
top0= sum((data.y-a0-b0*data.x).^2);
RS0= 1- top0/bottom0;
disp(['Method 0 (exact, sans uncertainty): y= ',num2str(a0),' +
',num2str(b0),'*x (r^2 = ',num2str(RS0),')']);
% ------------
% Method 1: Exact linear regression accounting for uncertainty (see ch.6 from
Bevington, eqns.6.13 and 6.23)
Delta= sum(1./XX)*sum((data.x.^2)./XX)-(sum(data.x./XX))^2;
a1= 1/Delta*(sum((data.x.^2)./XX)*sum((data.y./XX))- sum((data.x./
```

```
XX))*sum((data.x.*data.y)./XX));
b1= 1/Delta*(sum(1./XX)*sum((data.x.*data.y)./XX)- sum((data.x./
XX))*sum(data.y./XX));
sigmaA1= sqrt(1/Delta*sum((data.x.^2)/XX));
sigmaB1= sqrt(sum(1./XX)/Delta);
% also calculate r^2 (coefficient of determination) via
% http://en.wikipedia.org/wiki/Coefficient_of_determination
meanY= sum(data.y)/N;
bottom1= sum((data.y-meanY).^2);
top1= sum((data.y-a1-b1*data.x).^2);
RS1= 1- top1/bottom1;
disp(['Method 1 (exact, factors in uncertainty): y= ',num2str(a1),' +
',num2str(b1),'*x (r^2 = ',num2str(RS1),')']);
%disp(['Method 1 (exact) standard deviations: SD.a= ',num2str(sigmaA1),',
SD.b= ',num2str(sigmaB1)]);
% ------------
% Method 2: Brute-force minimize chi-squared via grid search
for n=1:numel(gridA)
    for m=1:numel(gridB)
        aT= gridA(n);   % extract 'test' parameters
        bT= gridB(m);
        chiS(n,m)= sum(((data.y-aT-bT*data.x)./data.SD).^2); % determine chi-
squared and store away
    end
end
% determines coords. of global minimum in chi-squared array (two lines
% below are a quick/dirty way to find the desired coords.)
[junk,indxT] = min(chiS(:));
[p,q] = ind2sub(size(chiS),indxT);
a2= gridA(p);   b2= gridB(q);   % extract the associated values
top2= sum((data.y-a2-b2*data.x).^2);
RS2= 1- top2/bottom1;
disp(['Method 2 (grid-search): y= ',num2str(a2),' + ',num2str(b2),'*x (r^2 =
',num2str(RS2),')']);
% ------------
% visualize
errorbar(data.x,data.y,data.SD,'r.');   % original 'data' points
hold on; grid on; xlabel('x'); ylabel('y'); axis([xMin xMax
floor(min(data.y)) ceil(max(data.y))]);
%plot(data.x,aD+bD*data.x,'g-','LineWidth',2); % base function (sans noise)
plot(data.x,a0+b0*data.x,'k-'); % Method 0 fit
plot(data.x,a1+b1*data.x,'b-'); % Method 1 fit
plot(data.x,a2+b2*data.x,'m--'); % Method 2 fit
legend('noisy data','Method 0 fit','Method 1 fit','Method 2
fit','Location','NorthWest');
```


noisy data
Method 0 fit
Method 1 fit
Method 2 fit
y
x
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2

```
==================================
Base function: y= a+b*x= 0.15 + 0.44*x (+ Gaussian noise)
Method 0 (exact, sans uncertainty): y= 0.14349 + 0.45133*x (r^2 = 0.92844)
Method 1 (exact, factors in uncertainty): y= 0.12407 + 0.45948*x (r^2 = 0.92657)
Method 2 (grid-search): y= 0.12424 + 0.4596*x (r^2 = 0.92664)
```

```
% ### EXregressAnscombe.m ###     2017.02.23 C.Bergevin

% Code to reproduce Anscombe's quartet fits
% (as per https://en.wikipedia.org/wiki/Anscombe's_quartet#/media/
File:Anscombe%27s_quartet_3.svg

% Notes
% o Uses Matlab's built-in polyfit.m function (see also EXregressLinear1.m
% if you wanted to do this manually)
% o Values obtained from http://hywelowen.org/anscombes-quartet/
% o First set of data (x1,y1) would be x1=A(:,1), y1=A(:,2). Similarly for
% the others
% o Ref: https://en.wikipedia.org/wiki/Anscombe's_quartet and
% https://en.wikipedia.org/wiki/Coefficient_of_determination

clear
% ========================================================

% --- specify the associated data
A=[10   8.04    10  9.14    10  7.46    8   6.58;
8   6.95    8   8.14    8   6.77    8   5.76;
13  7.58    13  8.74    13  12.74   8   7.71;
9   8.81    9   8.77    9   7.11    8   8.84;
11  8.33    11  9.26    11  7.81    8   8.47;
14  9.96    14  8.1     14  8.84    8   7.04;
6   7.24    6   6.13    6   6.08    8   5.25;
4   4.26    4   3.1     4   5.39    19  12.5;
12  10.84   12  9.13    12  8.15    8   5.56;
7   4.82    7   7.26    7   6.42    8   7.91;
5   5.68    5   4.74    5   5.73    8   6.89];

% --- loop through for each of the four data sets and do the analysis
cnt= 1; figure(1); clf;
for nn=1:4
    tempX= A(:,cnt); tempY=A(:,cnt+1);  % extract relevant #s from A
    [junk,indx]= sort(tempX);   % reorder (not necessary)
    x= tempX(indx); y= tempY(indx);
    p(nn,:)= polyfit(x,y,1);
    f= p(nn,1)*x + p(nn,2);
    % --- determine R ("Coefficient of determination")
    yM= sum(y)/numel(y);
    SStot= sum((y-yM).^2); SSreg= sum((f-yM).^2); SSres= sum((y-f).^2);
    Rsquared= 1- SSres/SStot;
    % --- visualize
    subplot(2,2,nn); h= plot(x,y,'o','LineWidth',1); xlabel('x');
ylabel('y');
    set(h(1),'MarkerEdgeColor','k','MarkerFaceColor',[1 .5 0]); hold on; grid
on;
    h2= plot(x,f,'b-');
    % --- indicate best fit params
    disp(['Set ',num2str(nn),': best fit is y=',num2str(p(nn,
1)),'x+',num2str(p(nn,2)),...
        ' --> R^2=',num2str(Rsquared)]);
    cnt= cnt+2; % update indexer
end
```


12
10
8
6
4
y
4
6
8
10
12
14
x
12
10
8
6
4
2
y
4
6
8
10
12
14
x
14
12
10
8
6
4
y
4
6
8
10
12
14
x
14
12
10
8
6
4
y
5
10
15
20
x

```
Set 1: best fit is y=0.50009x+3.0001 --> R^2=0.66654
Set 2: best fit is y=0.5x+3.0009 --> R^2=0.66624
Set 3: best fit is y=0.49973x+3.0025 --> R^2=0.66632
Set 4: best fit is y=0.49991x+3.0017 --> R^2=0.66671
```