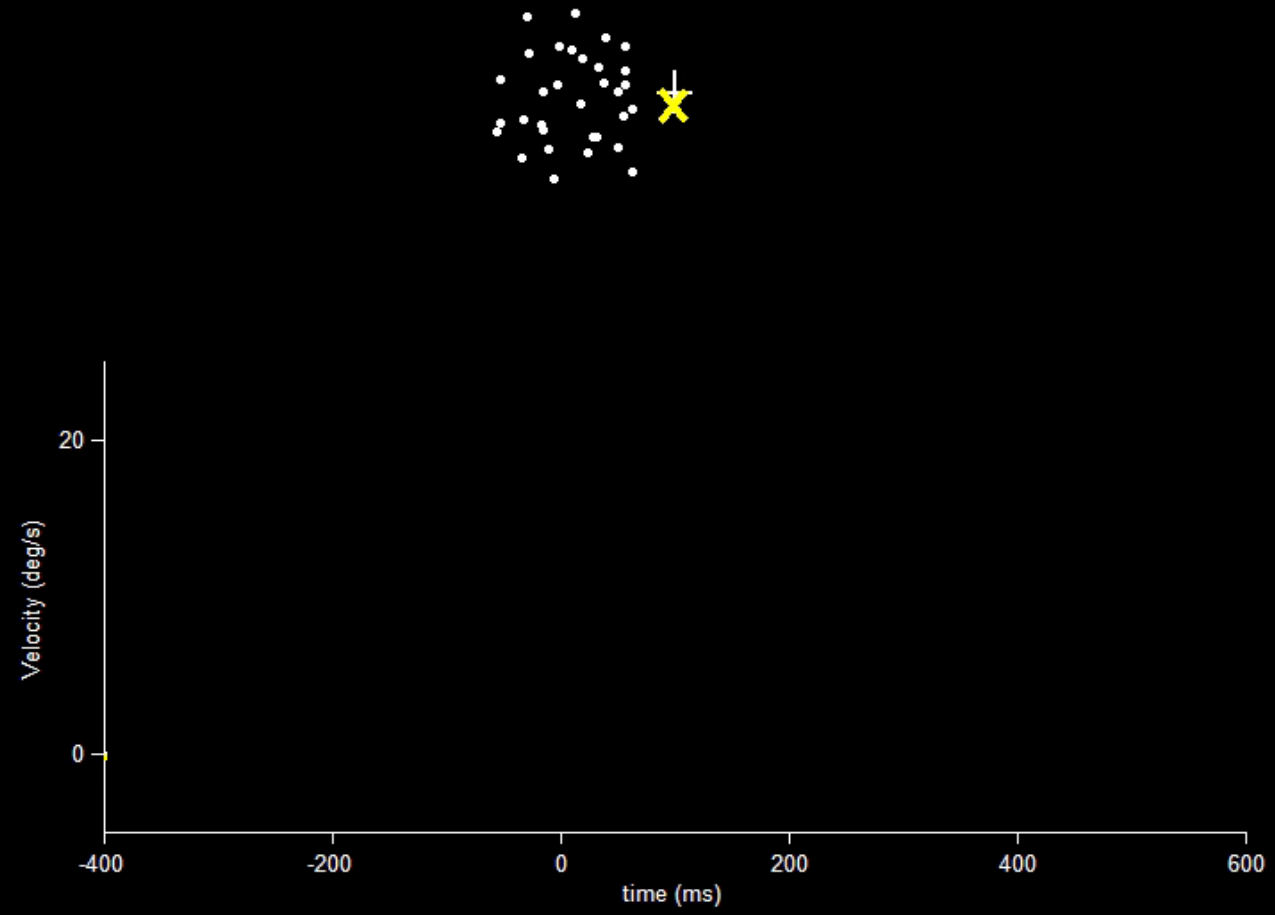
MOVIE AND SOUND

GENERATING MOVIE



GENERATING MOVIE

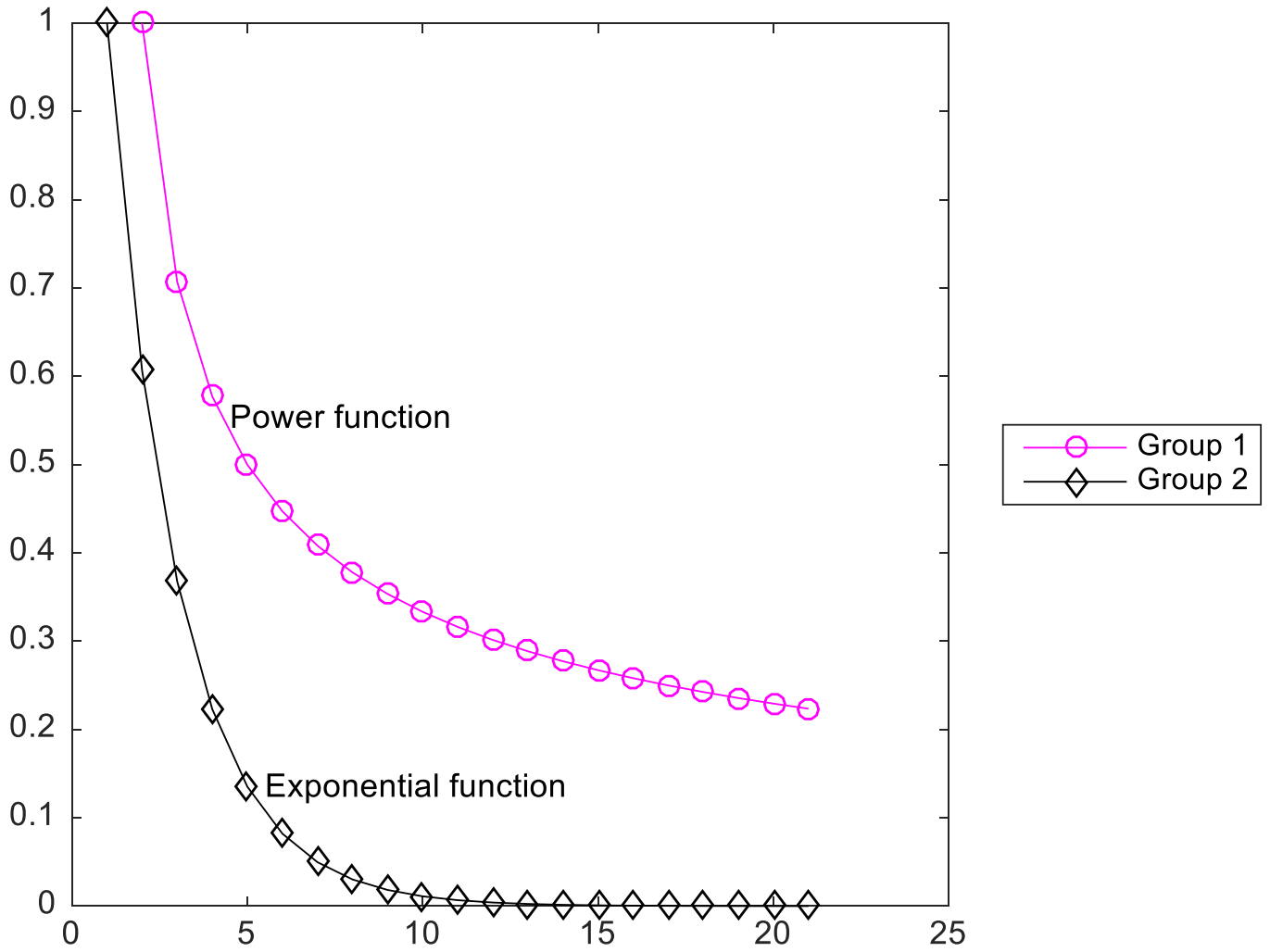
```
aviobj = VideoWriter('sinFunction.avi');

aviobj.FrameRate = 10;
aviobj.Quality = 100;

theta = [0:0.1:4*pi];
y = sin(theta);
open(aviobj);

figure(1)
axis([0 4*pi -1 1]);
set(gca, 'tickdir', 'out', 'box', 'on', 'xtick', [0:pi/2:4*pi], ...
'xticklabel',  [0:90:360 90:90:360]);
xlabel('angle (deg)');
hold on
```

GENERATING MOVIE

```
for i = 1:length(theta)
      plot(theta(i), y(i), 'color', 'k', 'marker', 'o', 'linestyle', '-');
      F = getframe(gcf);
      writeVideo(aviobj, F);
end
close(aviobj);
```

CONTROLLING PLOTS

```
a = 1;              % starting value
b = .5;             % rate parameter
xx = [0:20];
vert_offset = .05;
hor_offset = .50;

y_power = a * xx.^-b;
y_exp = a * exp(b*-xx);

hold on
box on
plot(y_power,'mo-');
plot(y_exp,'kd-');
legend('Group 1', 'Group 2', …
 'Location', 'EastOutside');

hor_p = xx(5) + hor_offset;
vert_p = y_power(5) + vert_offset;
text(hor_p,vert_p,'Power function');

hor_e = xx(6) + hor_offset;
vert_e = y_exp(6) + vert_offset;
text(hor_e,vert_e,'Exponential function');
```

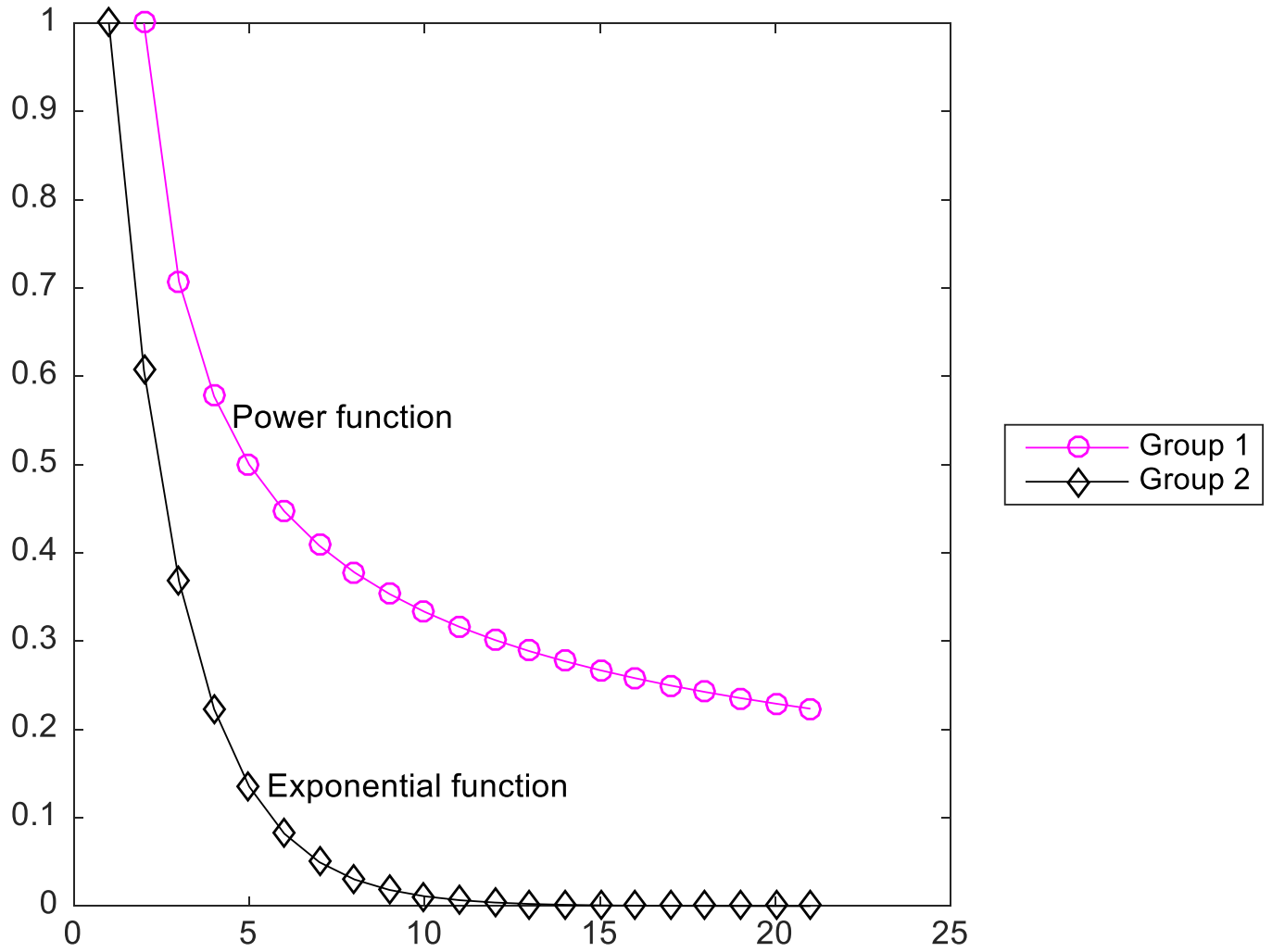


GENERATING MOVIE

1. Plot Power function first, then plot Exponential function

2. Place figure legend

3. Place text label 'Power function', and 'Exponential function' at the end of the movie



READING FRAMES OUT OF A MOVIE FILE

```
myMovieObj = VideoReader('sinFunction.avi');
nFrames = myMovieObj.NumberOfFrames;
for k = 1 : nFrames
       myFrames(k).cdata = read(myMovieObj, k);
end

for k = 1:10:nFrames
       image(myFrames(k));
       set(gca, 'ydir', 'reverse');
       text(50,50,sprintf('Frame Number: %d',k));
       pause(0.5);
end
```

WRITING FRAMES BACK TO A NEW MOVIE FILE

```
xyloObj = VideoReader('sinFunction.avi');
vidFrames = read(xyloObj);
nFrames = size(vidFrames,4);

figure(1)
writerObj = VideoWriter('nFile.avi');
open(writerObj);

for k = nFrames:-1:1
     image(vidFrames(:,:,:,k));
     set(gca, 'ydir', 'reverse');
     frame = getframe;
     writeVideo(writerObj,frame);
end

close(writerObj)
```

PLAYING SOUND

```
beep

load chirp
sound(y)
plot(y, 'k')

load handel
soundsc(y, [-3.25 3.25];
soundsc(y, [-15.25 15.25];

plot(y, 'k')
```

PLAYING SOUND

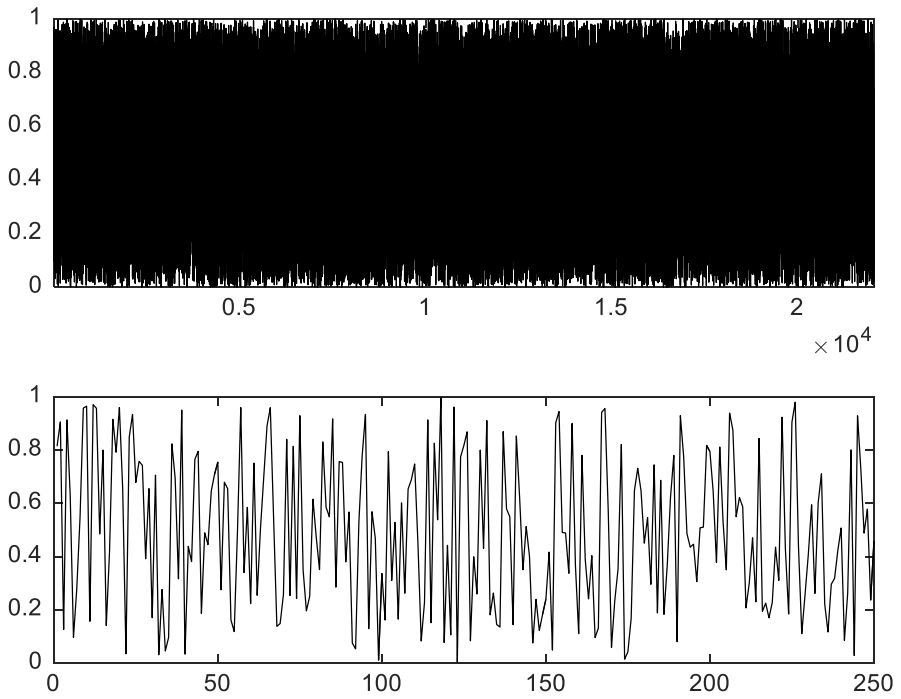
```
load handel
handelplayer = audioplayer(y, Fs);

load chirp;
chirpplayer = audioplayer(y, Fs);

play(handelplayer);
pause(2);
play(chirpplayer);

playblocking(handelplayer);
pause(2)
playblocking(chirpplayer);
```

PLAYING SOUND

```
load gong;   % Loads y and Fs for sound from gong.mat
tooloudplayer = audioplayer(y,Fs);  % Volume is too loud!
toosoftplayer = audioplayer(y/5,Fs);  % Volume is too soft!
goldilocksplayer = audioplayer(y/2,Fs); %Volume is just right!

playblocking(tooloudplayer);
playblocking(toosoftplayer);
playblocking(goldilocksplayer);
```

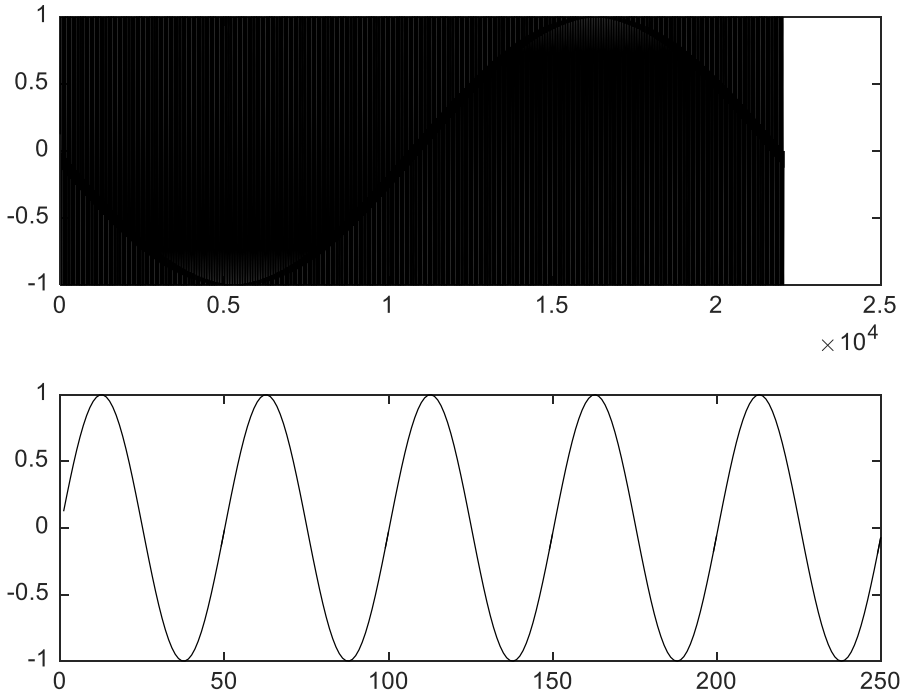
MAKING WHITE NOISE

```
sf = 22050;                              % sample frequency
d = 1.0;                                 % duration
n = sf*d;                                % number of samples
noise = rand(1,n);                       % uniform distribution
noise = noise / max(abs(noise));         % normalize
sound(noise,sf);                         % play sound
subplot(2,1,1)
plot(noise,'k')
xlim([1 n]);
subplot(2,1,2)
plot(noise(1:250),'k')
```



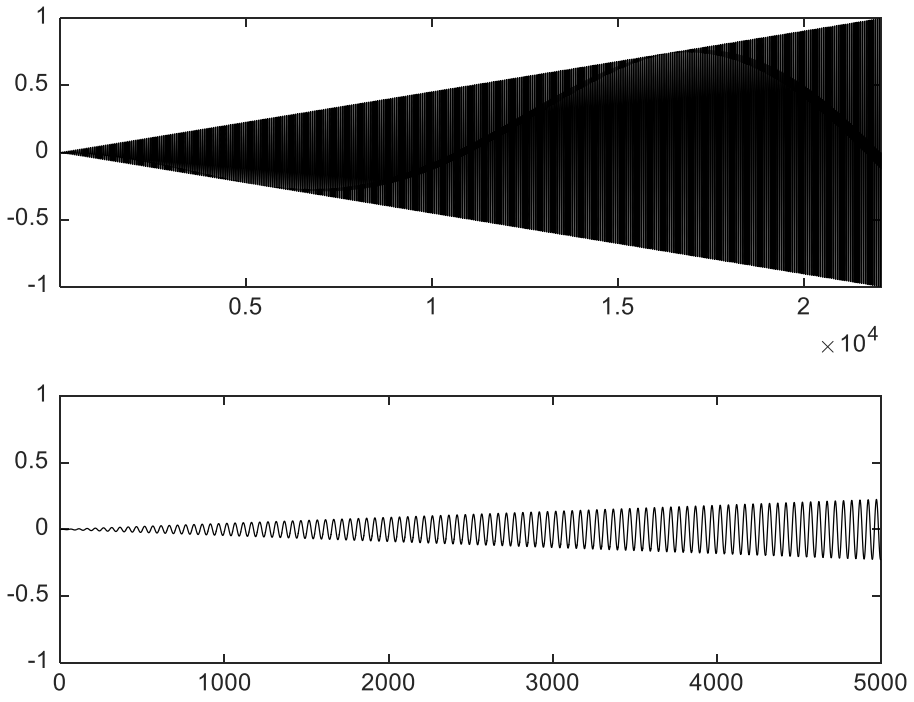
MAKING SINUSOIDAL SOUND

```
cf = 440;                       % carrier frequency (Hz)
sf = 22050;                     % sample frequency (Hz)
d = 1.0;                        % duration (s)
n = sf * d;                     % number of samples
s = (1:n) / sf;                 % time-dependent values
tone = sin(2 * pi * cf * s); % sinusoidal modulation
sound(tone,sf);                 % sound presentation
subplot(2,1,1)
plot(tone,'k')
subplot(2,1,2)
plot(tone(1:250),'k')
```



LINEARLY INCREASING SOUND

```
a = linspace(1/length(tone),1,length(tone));
sound(a.*tone,sf)
plot(a.*tone,'k')
subplot(2,1,1)
plot(a.*tone,'k')
xlim([1 n]);
subplot(2,1,2)
plot(a(1:5000).*tone(1:5000),'k')
ylim([-1 1]);
```



WRITING AUDIO INTO A FILE

```
fs = 8000;        % sampling frequency
t = 0:1/fs:0.25; % length of each note
tspace = 1.0;     % length of pause between notes
fr = 2^(1/12);    % frequency ratio between neighboring keys
A4 = 440;         % reference note for others
B4 = A4*fr^2;
C4 = A4*fr^(-9);
D4 = A4*fr^(-7);
E4 = A4*fr^(-5);
F4 = A4*fr^(-4);
G4 = A4*fr^(-2);
C5 = A4*fr^3;
xspace = zeros(1,tspace*fs);     % set pause
x = [cos(C4*2*pi*t),xspace, cos(D4*2*pi*t),xspace, cos(E4*2*pi*t),xspace, ...
     cos(F4*2*pi*t),xspace, cos(G4*2*pi*t),xspace, cos(A4*2*pi*t),xspace, ...
     cos(B4*2*pi*t),xspace, cos(C5*2*pi*t)];
myScale = audioplayer(x,fs);
play(myScale)
audiowrite('scale.wav',x,fs)
```

READING AUDIO FILE AND PLAYING IT

```
[y,Fs] = audioread('scale.wav');
sound(y,Fs)
```

PRACTICAL PROGRAMMING FOR A SIMPLE TASK

STROOP TASK

```
clear
close all;
figure('Name','Stroop Test')
words = {
    'Red'
    'Green'
    'Blue'
    'Black'
    };
colors = ['rgbk'];
text(.1,.8,sprintf(...
    'Report the COLOR of the text\n as quickly as you can!'),...
    'FontSize',18)
axis off
```

STROOP TASK

```
for t = 1:20
    w = randi(4);
    c = randi(4);
    myWordHandle = text(.2,.5,...
        char(words(w)),'Color',colors(c),...
        'Fontsize',48);
    if w == c
        conditionstring = sprintf('Compatible trial');
    else
        conditionstring = sprintf('Incompatible trial');
    end
    myConditionHandle = text(.2,.2,conditionstring);
    pause(2)
    delete([myWordHandle myConditionHandle])
    pause(1)
end
```

STROOP TASK

Report the COLOR of the text as quickly as you can!

Blue

Incompatible trial

Report the COLOR of the text as quickly as you can!

Blue

Compatible trial

FITTING DATA TO A FUNCTION FOR DESCRIBING DATA

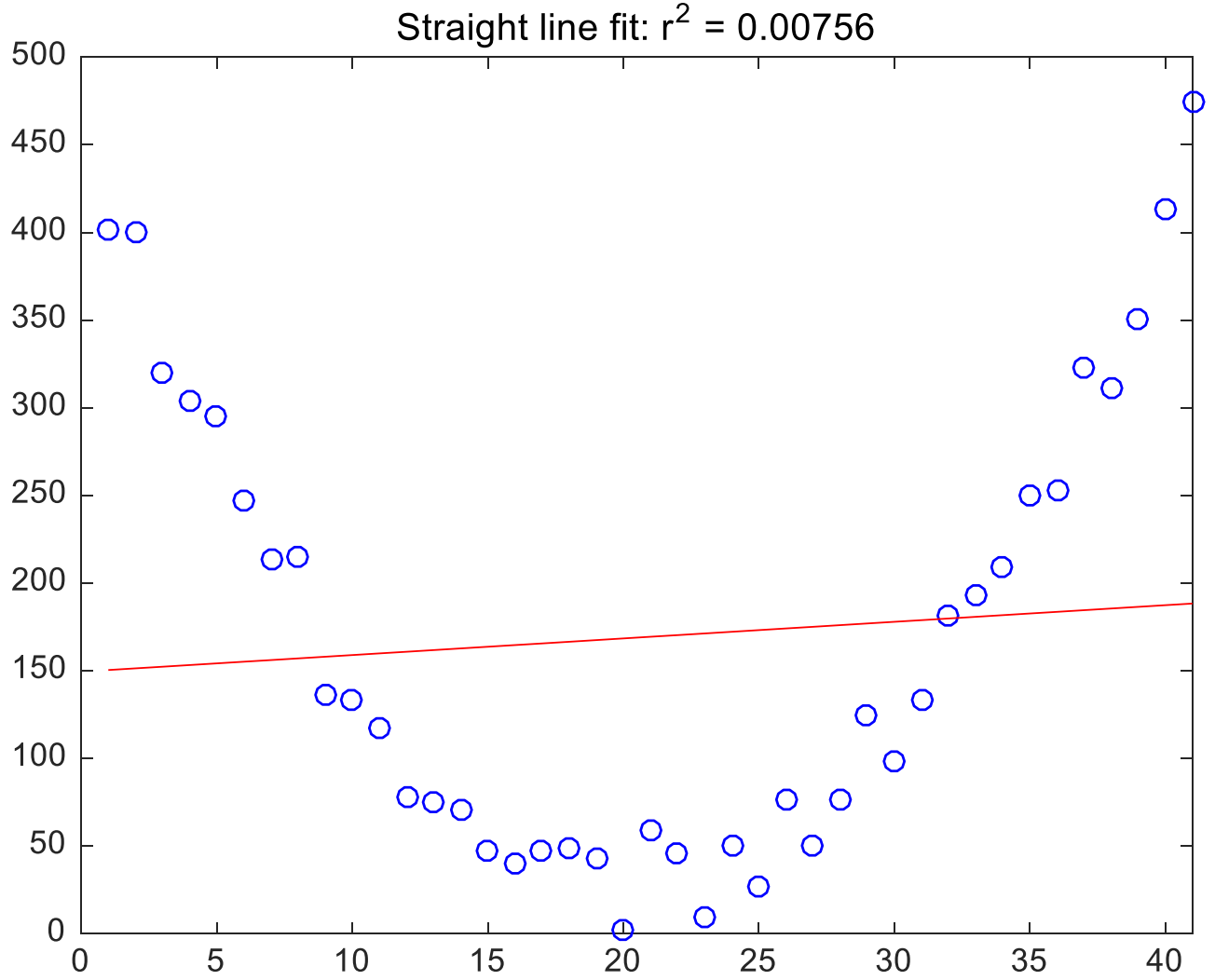
FITTING DATA TO A FUNCTION

```
clear x y
a3 = 0;
a2 = 1;
a1 = 1;
a0 = 0;
x = [-20:20];
randn_coeff = 60;

y = a3*x.^3  + a2*x.^2 + a1*x.^1 + a0*x.^0;
r = rand(length(y))*randn_coeff;
r = r(1,:);
y = y + r;
```

FITTING DATA TO A FUNCTION

```
fitted_coefficients = polyfit(x,y,1);
y_hat1 = fitted_coefficients(1)*x.^1 + ...
    fitted_coefficients(2)*x.^0; %apply polyfit coefficients to x
figure (1)
hold on
plot(y,'bo');          % show original data
plot(y_hat1,'r-');     % show fitted points joined by a line
xlim([0 length(x)]);
box on                 % put a box around the graph
c = corrcoef(y, y_hat1);
message = ['Straight line fit: r^2 = ',num2str(c(1,2)^2,3)];
title(message);
```

FITTING DATA TO A FUNCTION



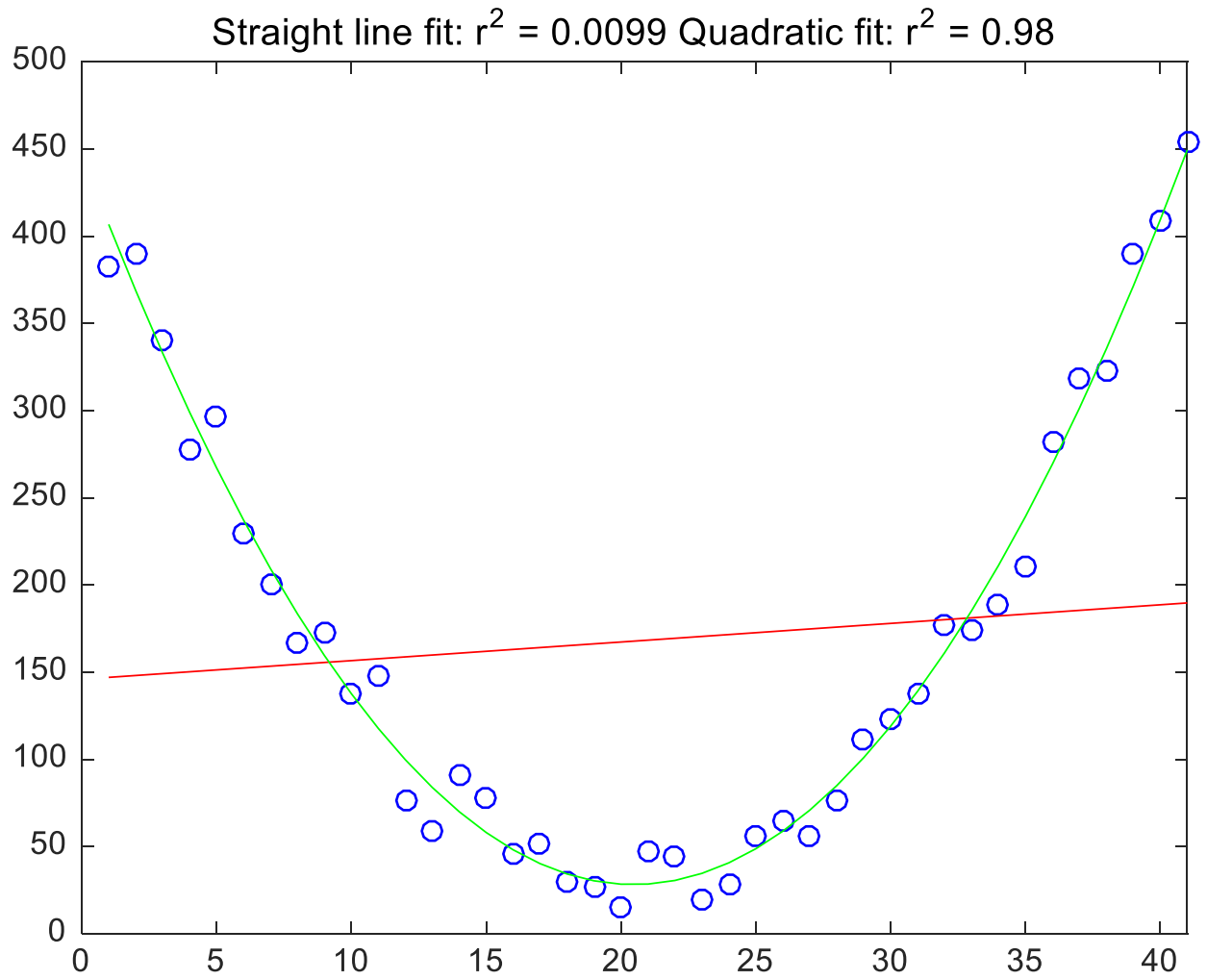
FITTING DATA TO A FUNCTION

```
clear x y
a3 = 0;
a2 = 1;
a1 = 1;
a0 = 0;
x = [-20:20];
randn_coeff = 60;

y = a3*x.^3  + a2*x.^2 + a1*x.^1 + a0*x.^0;
r = rand(length(y))*randn_coeff;
r = r(1,:);
y = y + r;

fitted_coefficients = polyfit(x,y,1);
y_hat1 = fitted_coefficients(1)*x.^1 + fitted_coefficients(2)*x.^0;
fitted_coefficients = polyfit(x,y,2);
y_hat2 = fitted_coefficients(1)*x.^2 + fitted_coefficients(2)*x.^1 + ...
     fitted_coefficients(3)*x.^0;
```

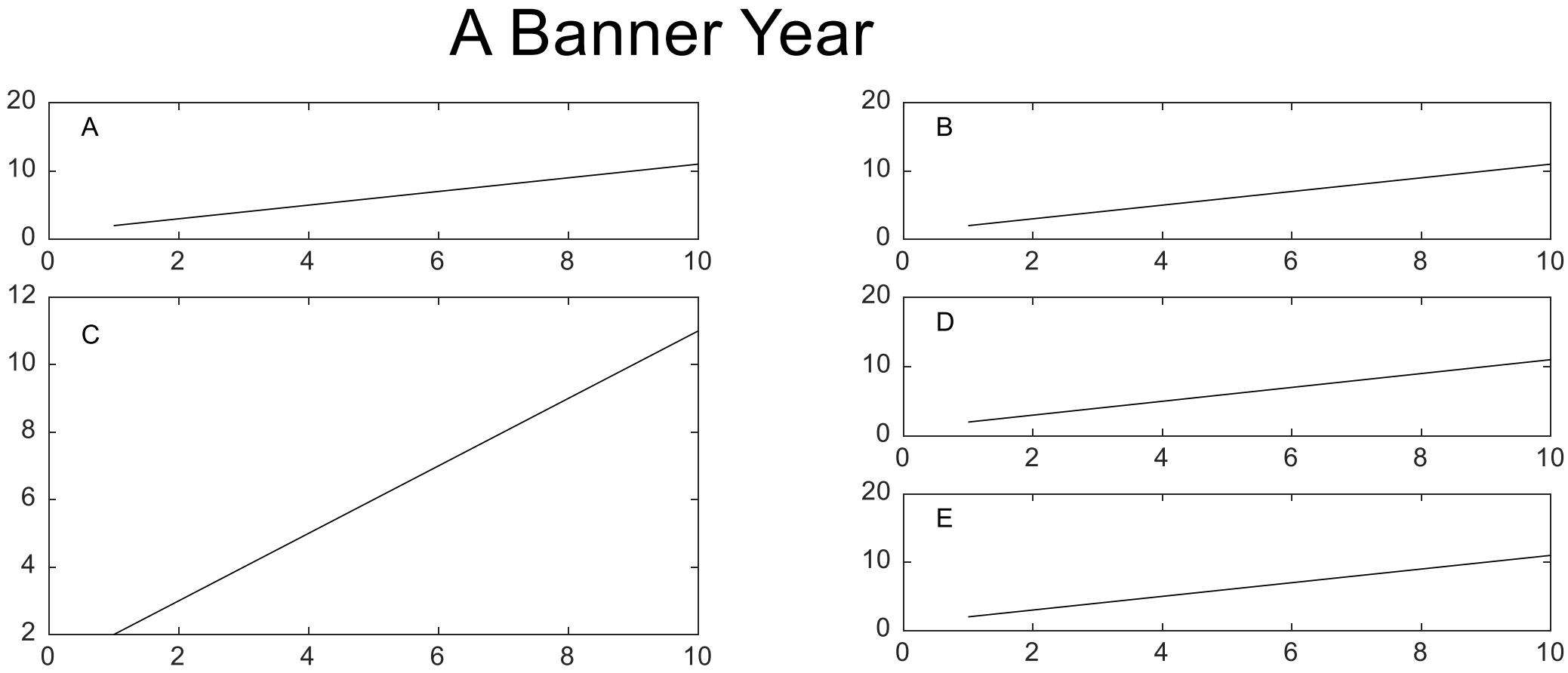
FITTING DATA TO A FUNCTION

```
figure (1)
hold on
plot(y,'bo');          % show original data
plot(y_hat1,'r-');     % show fitted points joined by a line
plot(y_hat2,'g-');     % show fitted points joined by a line
xlim([0 length(x)]);
box on                 % put a box around the graph
c = corrcoef(y, y_hat1);
c2 = corrcoef(y, y_hat2);
message = ['Straight line fit: r^2 = ',num2str(c(1,2)^2,2) ' Quadratic fit:
r^2 = ', num2str(c2(1,2)^2, 2)];
title(message);
```

FITTING DATA TO A FUNCTION



SUBPLOTS



SUBPLOTS

```
Function main
clf
clear x y
x = [1:10];
y = x + 1;

subplot(4,2,1:2);
xlim([0 1]);
ylim([0 1]);
axis off
text(-.05,.05,'              A BannerYear’,'fontsize',24);

subplot(4,2,3);
plot(x,y,'k')
text_in_box(.05,.80,'A');

subplot(4,2,4);
plot(x,y,'k');
text_in_box(.05,.80,'B');
```

SUBPLOTS

```
subplot(4,2,[5 7]);
plot(x,y,'k')
text_in_box(.05,.90,'C')

subplot(4,2,6);
plot(x,y,'k')
text_in_box(.05,.80,'D')

subplot(4,2,8);
plot(x,y,'k')
text_in_box(.05,.80,'E')

function text_in_box(x_place,y_place,s)

xs = xlim;
ys = ylim;
text(x_place*xs(2),y_place*ys(2),s);
```

PLOTTING IN THREE DIMENSIONS

```
[X,Y] = meshgrid(linspace(-2,2,41));
Z = X.*exp(-X.^2 - Y.^2);
plot3(X,Y,Z)
grid on

mesh(X,Y,Z);
box on

surf(X,Y,Z);
box on
set(gca, 'view', [0 90]);

contour(X,Y,Z);
```