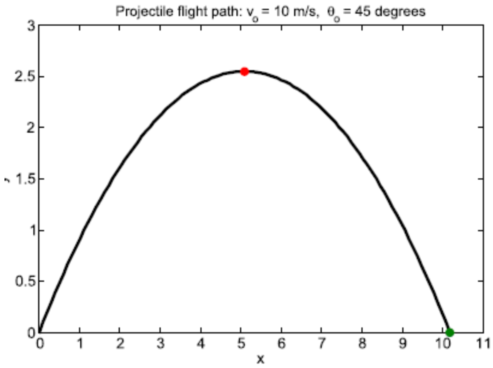
The design process[1] is outlined next. The steps may be listed as follows:

Step 1 *Problem analysis.* The purpose of this problem, and how to implement this problem.

To plot the trajectory of the Fight path. Math: 牛頓運動定律。

Step 2 *Problem statement.* Develop a detailed statement of the mathematical problem to be solved with a computer program. See text p. 87.



Step 3 *Processing scheme.* Define the inputs required and the outputs to be produced by the program.

Step 4 *Algorithm.* Design the step-by-step procedure in a *top-down* process that decomposes the overall problem into subordinate problems. The subtasks to solve the latter are refined by designing an itemized list of steps to be programmed. This list of tasks is the *structure plan* and is written in *pseudocode*

The goal is a plan that is understandable and easily translated into a computer language.

%% 8 steps of the projectile program in p. 89.

Step 5 *Program algorithm.* Translate or convert the algorithm into a computer language (e.g., MATLAB) and debug the syntax errors until the tool executes successfully.

Step 6 *Evaluation the result of your program.* Test all of the options and conduct a validation study of the program.

Step 7 *Application.* Solve the problems the program was designed to solve. If the program is well designed and useful, it can be saved in your working directory (i.e., in your user-developed toolbox) for future use.

```
%%===========================================================================
% Sec 3,1 The program design procedure
%%===========================================================================


%% (I)  the temperature translation problem.
% 1. input temperature in degree F


% 2. calculate C degree by (F-32)*5/9

% 3. display C degree


%% The proctile problem with zero air resistance
% The proctile problem with zero air resistance
% in a gravitational field with constant g.
% Written by Daniel T. Valentine .. September 2006
% Revised by D. T. Valentine ........... 2012/2016
% An eight-step structure plan applied in MATLAB:
%
% 1. Define the input variables.
%
g = 9.81; % Gravity in m/s/s.
vo = input('What is the launch speed in m/s?');
tho = input('What is the launch angle in degrees?');
tho = pi*tho/180; % Conversion of degrees to radians.
%
% 2. Calculate the range and duration of the flight.
%
txmax = (2*vo/g) * sin(tho);
xmax = txmax * vo * cos(tho);
%
% 3. Calculate the sequence of time steps to compute
% trajectory.
%
dt = txmax/100;  % time step
t = 0:dt:txmax;
%
% 4. Compute the trajectory.
%
x = (vo * cos(tho)) .* t;
y = (vo * sin(tho)) .* t - (g/2) .* t.^2;
%
% 5. Compute the speed and angular direction of the
```

```
% projectile. Note that vx = dx/dt, vy = dy/dt.
%
vx = vo * cos(tho);
vy = vo * sin(tho) - g .* t;
v = sqrt(vx.*vx + vy.*vy);% Speed
th = (180/pi) .* atan2(vy,vx);% Angular direction
%
% 6. Compute the time and horizontal distance at the
% maximum altitude.
%
tymax = (vo/g) * sin(tho);
xymax = xmax/2;
ymax = (vo/2) * tymax * sin(tho);
%
% 7. Display in the Command Window and on figures the ouput.
%
disp(['Range in meters =',num2str(xmax),',' ...
' Duration in seconds =', num2str(txmax)]);
disp(' ')
disp(['Maximum altitude in meters = ',num2str(ymax), ...
', Arrival at this altitude in seconds = ', num2str(tymax)])
plot(x,y,'k:',xmax,y(size(t)),'o',xmax/2,ymax,'o')
title(['Projectile flight path: v_o = ', num2str(vo),' m/s' ...
', \theta_o = ', num2str(180*tho/pi),' degrees'])
xlabel('x'), ylabel('y') % Plot of Figure 3.4.
figure % Creates a new figure.
plot(v,th,'r')
title('Projectile speed vs. angle')
xlabel('V'), ylabel('\theta') % Plot of Figure 3.5.
%
% 8. Stop.

%%=====================================================================
% Sec 3.2 Matlab function p. 92
%%=====================================================================

h = inline( 'cos(8*t) + cos(9*t)');
x = 0 : 20/300 : 20;
plot(x, h(x)), grid

h = inline( 'x.^2+y.^2','x','y');
tho=-pi:pi/10:pi;
x=cos(tho);y=sin(tho);
figure;polar(tho,h(x,y))

```

```

 % input the number of the student
clear all;close all;
N=input('   number of student:   ');
score=zeros(2,N);
% input the name and score of the student evaluate the average score
for i=1:N
    str1= input('student name:','s');
    eval(['name',int2str(i),'=str1;']);
%     if (i==1)
%         name=str1;
%     else
%         name=char(name,str1); % Create a character array.
%     end
score(1,i)=input('math score:   ');
score(2,i)=input('english score:   ');
avg(i)=(score(1,i)+score(2,i))/2;% avg(i) = sum(score(:,i))/2;
end

% output value
for i=1:N
    eval(['str1=name',int2str(i),';']);
 fprintf('the average score of %s is %3.2f \n',str1,avg(i));
end
save score_data N score


%% Exercise to write the average as a in-line function
close all; clear all;
load score_data % input N score



%%=========================================================================
% p. 93 matlab function
%%=========================================================================
% Use the following to write a matlab *.m script to evaluate the balance
%  function average = func1(vector)
%  average = sum(vector)/length(vector);

%% define two functions  save in stat2,m
% function [m,s] = stat2(x)
% n = length(x);
% m = avg(x,n);
% s = sqrt(sum((x-m).^2/n));
```

```
% end
%
% function m = avg(x,n)
% m = sum(x)/n;
% end

% values = [12.7, 45.4, 98.9, 26.6, 53.1];
% [ave,stdev] = stat2(values)



clc;clear;close all

money=50;%%本金
newBalance = zeros(1,12);

for k=1:12 %% 月份

   money = money *1.01;%%本金加利息
    newBalance(k)=money;%%每月存款結算
    money=money+50;%%每月定存
end

%%  write this program as the matlab function
% Temperature conversion from C to F
% or F to C as requested by the user
%
Dec = input(' Which way?: 1 => C to F? 0 => F to C: ');
Temp = input(' What is the temperature you want to convert? ');
%
% Note the logical equals sgn (==)
if Dec == 1
    TF = (9/5)*Temp + 32;
    disp(' Temperature in F: ')
    disp(TF)
else
    TC = (5/9)*(Temp-32);
    disp(' Temperature in C: ')
    disp(TC)
end



function x = quadratic(a,b,c)
% Equation:
```

```
% a*x^2 + b*x + c = 0
% Input: a,b,c
% Output: x = [x1 x2], the two solutions of
% this eequation.
if a==0 & b==0 & c==0
disp(' ')
disp('Solution indeterminate')
elseif a==0 & b==0
disp(' ')
disp('There is no solution')
elseif a==0
disp(' ')
disp('Only one root: equation is linear')
disp(' x ')
x1 = -c/b;
x2 = NaN;
elseif b^2 < 4*a*c
disp(' ')
disp(' x1, x2 are complex roots ')
disp(' x1 x2')
x1 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
x2 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
elseif b^2 == 4*a*c
x1 = -b/(2*a);
x2 = x1;
disp('equal roots')
disp(' x1 x2')
else
x1 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
x2 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
disp(' x1 x2')
end
if a==0 & b==0 & c==0
elseif a==0 & b==0
else
disp([x1 x2]);
end
end


%% Exercise 3.2
m=44; n=28;
while m~=n
    if m > n
        m = m-n;
```

```
    else
        n = n-m;
    end
end
display(m)




t = 0:.1:2*pi;
subplot(2,2,1)
plot(t,sin(t))
subplot(2,2,2)
plot(t,cos(t))
subplot(2,2,3)
plot(t,exp(t))
subplot(2,2,4)
plot(t,1./(1+t.^2))
```