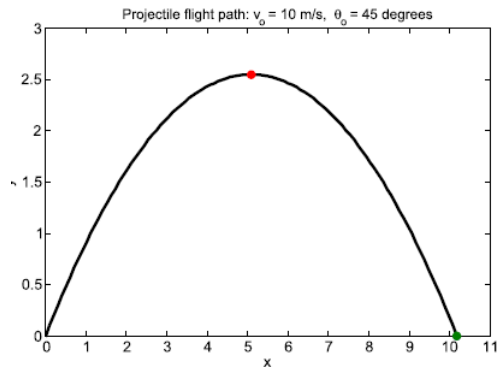


The design process¹ is outlined next. The steps may be listed as follows:

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

To plot the trajectory of the Flight 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.

```
1 %%=====
2 % Sec 3,1 The program design procedure
3 %%=====
4
5
6 %% (I) the temperature translation problem.
7 % 1. input temperature in degree F
8
9
10 % 2. calculate C degree by (F-32)*5/9
11
12 % 3. display C degree
13
14
15 %% The proctile problem with zero air resistance
16 % The proctile problem with zero air resistance
17 % in a gravitational field with constant g.
18 % Written by Daniel T. Valentine .. September 2006
19 % Revised by D. T. Valentine ..... 2012/2016
20 % An eight-step structure plan applied in MATLAB:
21 %
22 % 1. Define the input variables.
23 %
24 g = 9.81; % Gravity in m/s/s.
25 vo = input('What is the launch speed in m/s?');
26 tho = input('What is the launch angle in degrees?');
27 tho = pi*tho/180; % Conversion of degrees to radians.
28 %
29 % 2. Calculate the range and duration of the flight.
30 %
31 txmax = (2*vo/g) * sin(tho);
32 xmax = txmax * vo * cos(tho);
33 %
34 % 3. Calculate the sequence of time steps to compute
35 % trajectory.
36 %
37 dt = txmax/100; % time step
38 t = 0:dt:txmax;
39 %
40 % 4. Compute the trajectory.
41 %
42 x = (vo * cos(tho)) .* t;
43 y = (vo * sin(tho)) .* t - (g/2) .* t.^2;
44 %
45 % 5. Compute the speed and angular direction of the
```

```
46 % projectile. Note that vx = dx/dt, vy = dy/dt.
47 %
48 vx = vo * cos(tho);
49 vy = vo * sin(tho) - g .* t;
50 v = sqrt(vx.*vx + vy.*vy); % Speed
51 th = (180/pi) .* atan2(vy,vx); % Angular direction
52 %
53 % 6. Compute the time and horizontal distance at the
54 % maximum altitude.
55 %
56 tymax = (vo/g) * sin(tho);
57 xymax = xmax/2;
58 ymax = (vo/2) * tymax * sin(tho);
59 %
60 % 7. Display in the Command Window and on figures the output.
61 %
62 disp(['Range in meters =',num2str(xmax),' ' ...
63 ' Duration in seconds =', num2str(txmax)]);
64 disp(' ')
65 disp(['Maximum altitude in meters = ',num2str(ymax), ...
66 ', Arrival at this altitude in seconds = ', num2str(tymax)])
67 plot(x,y,'k:',xmax,y(size(t)), 'o',xmax/2,ymax,'o')
68 title(['Projectile flight path: v_o = ', num2str(vo),' m/s' ...
69 ', \theta_o = ', num2str(180*tho/pi),' degrees'])
70 xlabel('x'), ylabel('y') % Plot of Figure 3.4.
71 figure % Creates a new figure.
72 plot(v,th,'r')
73 title('Projectile speed vs. angle')
74 xlabel('V'), ylabel('\theta') % Plot of Figure 3.5.
75 %
76 % 8. Stop.
77
78 %%=====
79 % Sec 3.2 Matlab function p. 92
80 %%=====
81
82 h = inline( 'cos(8*t) + cos(9*t)');
83 x = 0 : 20/300 : 20;
84 plot(x, h(x)), grid
85
86 h = inline( 'x.^2+y.^2','x','y');
87 tho=-pi:pi/10:pi;
88 x=cos(tho);y=sin(tho);
89 figure;polar(tho,h(x,y))
90
```

```
91
92 % input the number of the student
93 clear all;close all;
94 N=input('  number of student:  ');
95 score=zeros(2,N);
96 % input the name and score of the student evaluate the average score
97 for i=1:N
98     str1= input('student name:','s');
99     eval(['name',int2str(i),'=str1;']);
100 %     if (i==1)
101 %         name=str1;
102 %     else
103 %         name=char(name,str1); % Create a character array.
104 %     end
105 score(1,i)=input('math score:  ');
106 score(2,i)=input('english score:  ');
107 avg(i)=(score(1,i)+score(2,i))/2;% avg(i) = sum(score(:,i))/2;
108 end
109
110 % output value
111 for i=1:N
112     eval(['str1=name',int2str(i),';']);
113     fprintf('the average score of %s is %3.2f \n',str1,avg(i));
114 end
115 save score_data N score
116
117
118 %% Exercise to write the average as a in-line function
119 close all; clear all;
120 load score_data % input N score
121
122
123
124 %%=====
125 % p. 93 matlab function
126 %%=====
127 % Use the following to write a matlab *.m script to evaluate the balance
128 % function average = func1(vector)
129 % average = sum(vector)/length(vector);
130
131 %% define two functions save in stat2,m
132 % function [m,s] = stat2(x)
133 % n = length(x);
134 % m = avg(x,n);
135 % s = sqrt(sum((x-m).^2/n));
```

```
136 % end
137 %
138 % function m = avg(x,n)
139 % m = sum(x)/n;
140 % end
141
142 % values = [12.7, 45.4, 98.9, 26.6, 53.1];
143 % [ave,stdev] = stat2(values)
144
145
146
147 clc;clear;close all
148
149 money=50;%%本金
150 newBalance = zeros(1,12);
151
152 for k=1:12 %% 月份
153
154     money = money *1.01;%%本金加利息
155     newBalance(k)=money;%%每月存款結算
156     money=money+50;%%每月定存
157 end
158
159 %% write this program as the matlab function
160 % Temperature conversion from C to F
161 % or F to C as requested by the user
162 %
163 Dec = input(' Which way?: 1 => C to F? 0 => F to C: ');
164 Temp = input(' What is the temperature you want to convert? ');
165 %
166 % Note the logical equals sign (==)
167 if Dec == 1
168     TF = (9/5)*Temp + 32;
169     disp(' Temperature in F: ')
170     disp(TF)
171 else
172     TC = (5/9)*(Temp-32);
173     disp(' Temperature in C: ')
174     disp(TC)
175 end
176
177
178
179 function x = quadratic(a,b,c)
180 % Equation:
```

```
181 % a*x^2 + b*x + c = 0
182 % Input: a,b,c
183 % Output: x = [x1 x2], the two solutions of
184 % this equation.
185 if a==0 & b==0 & c==0
186 disp(' ')
187 disp('Solution indeterminate')
188 elseif a==0 & b==0
189 disp(' ')
190 disp('There is no solution')
191 elseif a==0
192 disp(' ')
193 disp('Only one root: equation is linear')
194 disp(' x ')
195 x1 = -c/b;
196 x2 = NaN;
197 elseif b^2 < 4*a*c
198 disp(' ')
199 disp(' x1, x2 are complex roots ')
200 disp(' x1 x2')
201 x1 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
202 x2 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
203 elseif b^2 == 4*a*c
204 x1 = -b/(2*a);
205 x2 = x1;
206 disp('equal roots')
207 disp(' x1 x2')
208 else
209 x1 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
210 x2 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
211 disp(' x1 x2')
212 end
213 if a==0 & b==0 & c==0
214 elseif a==0 & b==0
215 else
216 disp([x1 x2]);
217 end
218 end
219
220
221 %% Exercise 3.2
222 m=44; n=28;
223 while m~=n
224     if m > n
225         m = m-n;
```

```
226     else
227         n = n-m;
228     end
229 end
230 display(m)
231
232
233
234
235 t = 0:.1:2*pi;
236 subplot(2,2,1)
237 plot(t,sin(t))
238 subplot(2,2,2)
239 plot(t,cos(t))
240 subplot(2,2,3)
241 plot(t,exp(t))
242 subplot(2,2,4)
243 plot(t,1./(1+t.^2))
244
245
246
247
248
249
250
251
```