%% Sec. 8.1 repeated loop by using for :

A for statement should be used to program a determinate loop, where the number of repeats is known (in principle) *before* the loop is encountered.

```
% binomial coefficient
ncr=1;
n=5;
r=3;
for k=1:r
    ncr=ncr*(n-k+1)/k;
end
disp(ncr)
```

$$\binom{n}{r} = \frac{n(n-1)(n-2)\cdots(n-r+1)}{r!},$$

$$\text{e.g., } \binom{10}{3} = \frac{10!}{3! \times 7!} = \frac{10 \times 9 \times 8}{1 \times 2 \times 3}.$$

A can of orange juice at temperature 25 °C is placed in a fridge, where the ambient temperature $F$ is 10 °C. We want to find out how the temperature of the orange juice changes over a period of time. A standard way of approaching this type of problem is to break the time period up into a number of small steps, each of length $dt$. If $T_i$ is the temperature at the *beginning* of step $i$, we can use the following model to get $T_{i+1}$ from $T_i$:

$$T_{i+1} = T_i - K\,dt\,(T_i - F), \qquad (8.3)$$

where $K$ is a constant parameter depending on the insulating properties of the can, and the thermal properties of orange juice. Assume that units are chosen so that time is in minutes.

```
%     An exaple in p. 181 : Update processes


K = 0.05;
F = 10;
a = 0; % start time
b = 100; % end time
time = a; % initialize time
T = 25; % initialize temperature
load train % prepare to blow the whistle
dt = input( 'dt: ' );
opint = input( 'output interval (minutes): ');
if opint/dt ~= fix(opint/dt)
    sound(y, Fs) % blow the whistle!
    disp( 'output interval is not a multiple of dt!');
    break
end
clc
format bank
```

```
disp( ' Time Temperature' );
disp( [time T] ) % display initial values
for time = a+dt : dt : b
      T = T - K * dt * (T - F);
      if abs(rem(time, opint)) < 1e-6 % practically zero!
            disp( [time T] )
      end
end

%-------------------------------------------------


%      Sec. 8.2 repeated loop by using white
%    8.2.1 :    A guessing game
```

While *condition* is true repeat: statements to be repeated (reset truth value of *condition*).

Note that:  *condition* is the condition to repeat

In general the `while` statement looks like this:

```
while condition
  statements
end
```

```
matnum = floor(10 * rand + 1);
guess = input( 'Your guess please: ' );
% is a sound data
load splat
while guess ~= matnum
      sound(y, Fs)
      if guess > matnum
            disp( 'Too high' )
      else
            disp( 'Too low')
      end;
      guess = input( 'Your next guess please: ' );
end
disp( 'At last!' )
load handel
sound(y, Fs) % hallelujah!
%-------------------------------------------------
```

1. Generate random integer
2. Ask user for guess
3. While guess is wrong:
   - If guess is too low
     - Tell her it is too low
   - Otherwise
     - Tell her it is too high
   - Ask user for new guess
4. Polite congratulations
5. Stop.

% 8.2.3 Doubling time of an investment

Suppose we have invested some money which draws 10% interest per year, compounded. We would like to know how long it takes for the investment to double.

1. Initialize balance, year, interest rate
2. Display headings
3. Repeat Update balance according to interest rate

Display year, balance until balance exceeds twice original balance

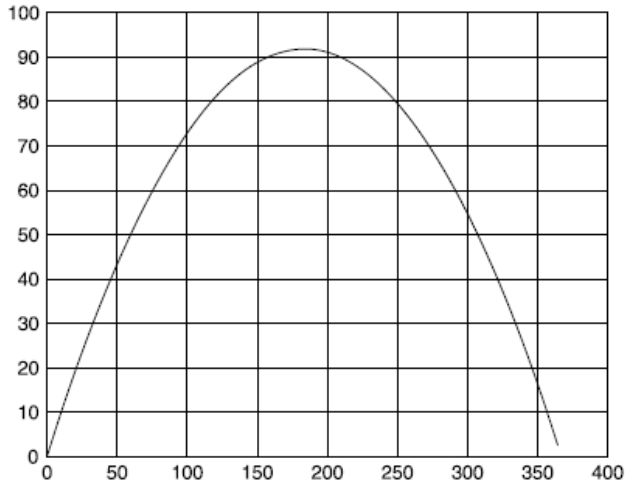
4. Stop.

```
a = 1000;
r = 0.1;
bal = a;
year = 0;
disp('Year Balance')
while bal < 2 * a
      bal = bal + r * bal;
      year = year + 1;
      disp( [year bal] )
end
```

```
%--------------------------------------------------
```

% 8.2.5 Projectile trajectory

The idea is to calculatethe trajectory repeatedly with increasing time, *while* the vertical displacement (*y*) remains positive.



```
dt = 0.1;
g = 9.8;
u = 60;
ang = input( 'Launch angle in degrees: ' );
ang = ang * pi / 180; % convert to radians
x = 0; y = 0; t = 0; % for starters
more(15)
while y >= 0
	disp( [t x y] );
	t = t + dt;
	y = u * sin(ang) * t - g * t^2 / 2;
	x = u * cos(ang) * t;
end


%-------------------------------------------------

%	Now suppose we want to plot the trajectory

dt = 0.1;
g = 9.8;
u = 60;
ang = input( 'Launch angle in degrees: ' );
ang = ang * pi / 180; % convert to radians
xp = zeros(1); yp = zeros(1); % initialize
y = 0; t = 0;
i = 1; % initial vector subscript
while y >= 0
	t = t + dt;
	i = i + 1;
	y = u * sin(ang) * t - g * t^2 / 2;
	if y >= 0
		xp(i) = u * cos(ang) * t;
		yp(i) = y;
	end
end
plot(xp, yp),grid
%-------------------------------------------------
```

```
%      8.2.7 Menus

k = 0;
while k ~= 3
      k = menu( 'Click on your option', 'Do this', ...
            'Do that', 'Quit' );
      if k == 1
            disp( 'Do this ... press any key to continue ...' )
                  pause
      elseif k == 2
            disp( 'Do that ... press any key to continue ...' )
                  pause
      end
end
%-------------------------------------------------
```

Write a program to compute the sum of the series $1^2 + 2^2 + 3^2 \ldots$ such that the sum is as large as possible without exceeding 1000. The program should display how many terms are used in the sum.

If an amount of money $A$ is invested for $k$ years at a nominal annual interest rate $r$ (expressed as a decimal fraction), the value $V$ of the investment after $k$ years is given by

$V = A(1 + r/n)^{nk}$

where $n$ is the number of compounding periods per year. Write a program to compute $V$ as $n$ gets larger and larger, i.e. as the compounding periods become more and more frequent, like monthly, daily, hourly, etc. Take $A = 1000$, $r = 4\%$ and $k = 10$ years.

- A `while` statement should be used to program an indeterminate repeat structure, where the exact number of repeats is *not* known in advance. Another way of saying this is that these statements should be used whenever the truth value of the condition for repeating is changed in the body of the loop. This situation is characterized by the following structure plan:

  While *condition* is true repeat:
  statements to be repeated (reset truth value of *condition*).

  Note that *condition* is the condition to repeat.
- The statements in a `while` construct may sometimes never be executed.
- Loops may be nested to any depth.
- The `menu` statement inside a `while` loop may be used to present a user with a menu of choices.

## EXERCISES

8.1 A person deposits \$1000 in a bank. Interest is compounded monthly at the rate of 1% per month. Write a program which will compute the monthly balance, but write it only *annually* for 10 years (use nested `for` loops, with the outer loop for 10 years, and the inner loop for 12 months). Note that after 10 years, the balance is \$3300.39, whereas if interest had been compounded annually at the rate of 12% per year the balance would only have been \$3105.85.
See if you can vectorize your solution.

8.2 There are many formulae for computing $\pi$ (the ratio of a circle's circumference to its diameter). The simplest is

$$\frac{\pi}{4} = 1 - 1/3 + 1/5 - 1/7 + 1/9 - \ldots \tag{8.4}$$

which comes from putting $x = 1$ in the series

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \frac{x^9}{9} - \ldots \tag{8.5}$$

(a) Write a program to compute $\pi$ using Equation (8.4). Use as many terms in the series as your computer will reasonably allow (start modestly, with 100 terms, say, and re-run your program with more and more each time). You should find that the series converges very slowly, i.e. it takes a lot of terms to get fairly close to $\pi$.

(b) Rearranging the series speeds up the convergence:

$$\frac{\pi}{8} = \frac{1}{1 \times 3} + \frac{1}{5 \times 7} + \frac{1}{9 \times 11} \ldots$$

Write a program to compute $\pi$ using this series instead. You should find that you need fewer terms to reach the same level of accuracy that you got in (a).

(c) One of the fastest series for $\pi$ is

$$\frac{\pi}{4} = 6\arctan\frac{1}{8} + 2\arctan\frac{1}{57} + \arctan\frac{1}{239}.$$

Use this formula to compute $\pi$. Don't use the MATLAB function `atan` to compute the arctangents, since that would be cheating. Rather use Equation (8.5).

(d) Can you vectorize any of your solutions (if you haven't already)?

8.3 The following method of computing $\pi$ is due to Archimedes:

1. Let $A = 1$ and $N = 6$
2. Repeat 10 times, say:
   Replace $N$ by $2N$
   Replace $A$ by $[2 - \sqrt{(4 - A^2)}]^{1/2}$
   Let $L = NA/2$
   Let $U = L/\sqrt{1 - A^2/2}$
   Let $P = (U + L)/2$ (estimate of $\pi$)
   Let $E = (U - L)/2$ (estimate of error)
   Print $N$, $P$, $E$
3. Stop.

Write a program to implement the algorithm.

8.4 Write a program to compute a table of the function

$$f(x) = x\sin\left[\frac{\pi(1 + 20x)}{2}\right]$$

over the (closed) interval $[-1, 1]$ using increments in $x$ of (a) 0.2, (b) 0.1 and (c) 0.01.

Use your tables to sketch graphs of $f(x)$ for the three cases (by hand), and observe that the tables for (a) and (b) give totally the wrong picture of $f(x)$.

Get your program to draw the graph of $f(x)$ for the three cases, superimposed.

8.5 The transcendental number $e$ (2.71828182845904 ...) can be shown to be the limit of

$$(1 + x)^{1/x}$$

as $x$ tends to zero (from above). Write a program which shows how this expression converges to $e$ as $x$ gets closer and closer to zero.

8.6 A square wave of period $T$ may be defined by the function

$$f(t) = \begin{cases} 1 & (0 < t < T) \\ -1 & (-T < t < 0). \end{cases}$$

The Fourier series for $f(t)$ is given by

$$F(t) = \frac{4}{\pi}\sum_{k=0}^{\infty}\frac{1}{2k+1}\sin\left[\frac{(2k+1)\pi t}{T}\right].$$

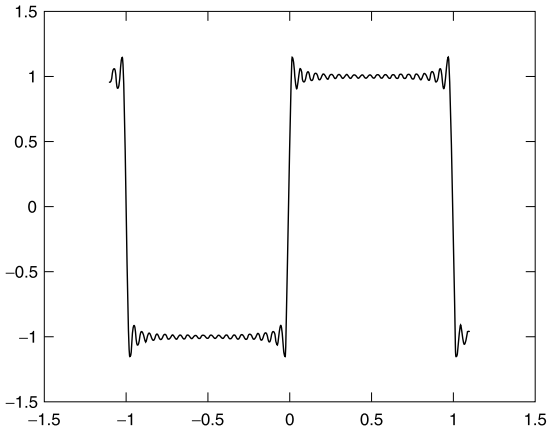



**FIGURE 8.3** Fourier series: The Gibbs phenomenon.

It is of interest to know how many terms are needed for a good approximation to this infinite sum. Taking $T = 1$, write a program to compute and plot the sum to $n$ terms of the series for $t$ from $-1.1$ to 1.1 in steps of 0.01, say. Run the program for different values of $n$, e.g. 1, 3, 6, etc. Superimpose plots of $F(t)$ against $t$ for a few values of $n$.
On each side of a discontinuity a Fourier series exhibits peculiar oscillatory behavior known as the Gibbs phenomenon. Figure 8.3 shows this clearly for the above series with $n = 20$ (and increments in $t$ of 0.01). The phenomenon is much sharper for $n = 200$ and $t$ increments of 0.001.

8.7 If an amount of money $A$ is invested for $k$ years at a nominal annual interest rate $r$ (expressed as a decimal fraction), the value $V$ of the investment after $k$ years is given by

$$V = A(1 + r/n)^{nk}$$

where $n$ is the number of compounding periods per year. Write a program to compute $V$ as $n$ gets larger and larger, i.e. as the compounding periods become more and more frequent, like monthly, daily, hourly, etc. Take $A = 1000$, $r = 4\%$ and $k = 10$ years. You should observe that your output gradually approaches a limit. **Hint**: use a `for` loop which doubles $n$ each time, starting with $n = 1$.
Also compute the value of the formula $Ae^{rk}$ for the same values of $A, r$ and $k$ (use the MATLAB function `exp`), and compare this value with the values of $V$ computed above. What do you conclude?

8.8 Write a program to compute the sum of the series $1^2 + 2^2 + 3^2 \ldots$ such that the sum is as large as possible without exceeding 1000. The program should display how many terms are used in the sum.

8.9 One of the programs in Section 8.2 shows that an amount of $1000 will double in eight years with an interest rate of 10%. Using the same

interest rate, run the program with initial balances of \$500, \$2000 and \$10000 (say) to see how long they all take to double. The results may surprise you.

8.10 Write a program to implement the structure plan of Exercise 3.2.

8.11 Use the Taylor series

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

to write a program to compute $\cos x$ correct to four decimal places ($x$ is in radians). See how many terms are needed to get 4-figure agreement with the MATLAB function `cos`. Don't make $x$ too large; that could cause rounding error.

8.12 A student borrows \$10 000 to buy a used car. Interest on her loan is compounded at the rate of 2% per month while the outstanding balance of the loan is more than \$5000, and at 1% per month otherwise. She pays back \$300 every month, except for the last month, when the repayment must be less than \$300. She pays at the end of the month, *after* the interest on the balance has been compounded. The first repayment is made one month after the loan is paid out. Write a program which displays a monthly statement of the balance (after the monthly payment has been made), the final payment, and the month of the final payment.

8.13 A projectile, the equations of motion of which are given in Chapter 3, is launched from the point O with an initial velocity of 60 m/s at an angle of 50° to the horizontal. Write a program which computes and displays the time in the air, and horizontal and vertical displacement from the point O every 0.5 s, as long as the projectile remains above a horizontal plane through O.

8.14 When a resistor ($R$), capacitor ($C$) and battery ($V$) are connected in series, a charge $Q$ builds up on the capacitor according to the formula

$$Q(t) = CV(1 - e^{-t/RC})$$

if there is no charge on the capacitor at time $t = 0$. The problem is to monitor the charge on the capacitor every 0.1 s in order to detect when it reaches a level of 8 units of charge, given that $V = 9$, $R = 4$ and $C = 1$. Write a program which displays the time and charge every 0.1 seconds until the charge first exceeds 8 units (i.e. the last charge displayed must exceed 8). Once you have done this, rewrite the program to display the charge only while it is strictly less than 8 units.

8.15 Adapt your program for the prime number algorithm in Section 8.2 to find all the prime factors of a given number (even or odd).

```
%8.2
%(a)
clear
terms = 500;
pi = 0;
sign = 1;
for n = 1 : terms
pi = pi + sign * 4 / (2 * n - 1);

%公式 8.4 pi=4(1-1/3+1/5-1/7+1/9...)
sign = sign * (-1);
end
pi

%(b)
clear
terms = 500;
pi = 0;
for n = 1 : terms
pi = pi + 8 / ((4 * n - 3) * (4 * n - 1));
end
pi

%(c)
clear
pi=4*(6* arctan(1/8) +2*arctan(1/57) +arctan(1/239) )
% Function file arctan.m
function y = arctan(x) %課本公式(8.5)
terms = 100;sign=1;y=0;
for n = 1 :2:terms
y = y + sign* (x^n)/n;
sign = sign * (-1);
end


%(d)
% vectorization of (a)
clear
```

```
n=1: 2: terms;
pi = 4*(sum(1./(2*n-1)) - sum(1./(2*n+1)))

% vectorization of (b)
clear
n=1: terms;
pi = 8*(      sum(1./((4 * n - 3) .* (4 * n - 1)))      )


%8.4
clear
clc;clear all;
x1=-1:0.2:1;
y1=x1.*(sin(pi*(1+20*x1)/2));
plot(x1,y1,'g');hold on;
x2=-1:0.1:1;
y2=x2.*(sin(pi*(1+20*x2)/2));
plot(x2,y2,'b');hold on;
x3=-1:0.01:1;
y3=x3.*(sin(pi*(1+20*x3)/2));
plot(x3,y3,'r');

%8.8
clear
sum = 0;
terms =1;
while (sum+terms^2) <= 2000
sum = sum + terms^2;
terms = terms + 1;
end
disp( [terms-1, sum] );

%8.9
clear
format bank
for a=[500 2000 10000]
r = 0.1;
bal = a;
```

```
year = 0;
disp( 'Year Balance' )
while bal < 2 * a
bal = bal + r * bal;
year = year + 1;
disp( [year bal] )
end
end
format


%8.11
clear
x=3;
i=0;
y=0;terms=1;sign=1;
while    roundn(cos(x),-4) ~= roundn(y,-4)
       disp( 'Terms    Cox(x)' )
       disp( [terms y] )
       y=y+sign*x^i/factorial(i);%-x^(i+2)/factorial(i+2)%factorial 階乘
       i=i+2;
       sign=sign*-1;
       terms=terms+1;
end

%8.13
clear
g = 9.81; % Gravity in m/s/s.
vo = 60;%initial velocity
tho = 50;%angle
tho = pi*tho/180; % Conversion of degrees to radians.
%計算飛行的範圍和持續時間。
txmax = (2*vo/g) * sin(tho);
xmax = txmax * vo * cos(tho);
%計算軌蹟的時間步長順序
dt = 0.5; %間格秒數
t = 0:dt:txmax;
%計算軌跡
```

```
x = (vo * cos(tho)) .* t;
y = (vo * sin(tho)) .* t - (g/2) .* t.^2;
plot(x,y,'-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.
disp('在空中秒數 離原點水平距離 垂直距離 ');
disp([t' x' y']);
```