1.1

Give values to variables a and b on the command line, e.g., a=3 and b=5. Write some statements to find the sum, difference, product and quotient of a and b.

%1.1

a = 3; %creattion of number a

b = 5; %creattion of number a

disp (a + b); %ans of a+b

disp (a - b); %ans of a-b

disp (a \* b); %ans of a\*b

disp (a / b); %ans of a/b

8

-2

15

0.6000

1.2

In Section 1.2.5 of the text a script is given for an animation of the Mexican hat problem. Type this into the editor, save it and execute it. Once you finish debugging it and it executes successfully try modifying it. (a) Change the maximum value of n from 3 to 4 and execute the script. (b) Change the time delay in the pause function from 0.05 to 0.1. (c) Change the z=sin(r.\*n)./r; command line to z=cos(r.\*n); and execute the script.

%1.2

% Mexican hat problem as given.

[x,y] = meshgrid(-8:0.5:8);

r = sqrt(x.^2 + y.^2) + eps;

for n=-3:0.05:3

z = sin(r.\*n)./r;

surf(z),view(-37,38),axis([0,40,0,40,-4,4]);

pause(0.05)

end

% Change the maximum value of n from 3 to 4.

[x,y] = meshgrid(-8:0.5:8);

r = sqrt(x.^2 + y.^2) + eps;

for n=-4:0.05:4

z = sin(r.\*n)./r;

surf(z),view(-37,38),axis([0,40,0,40,-4,4]);

pause(0.05)

end

% Change the time delay in the pause function from 0.05 to 0.1.

[x,y] = meshgrid(-8:0.5:8);

r = sqrt(x.^2 + y.^2) + eps;

for n=-4:0.05:4

z = sin(r.\*n)./r;

surf(z),view(-37,38),axis([0,40,0,40,-4,4]);

pause(0.1)

end

% Change the z=sin(r.\*n)./r;

% command line to z=cos(r.\*n);

[x,y] = meshgrid(-8:0.5:8);

r = sqrt(x.^2 + y.^2) + eps;

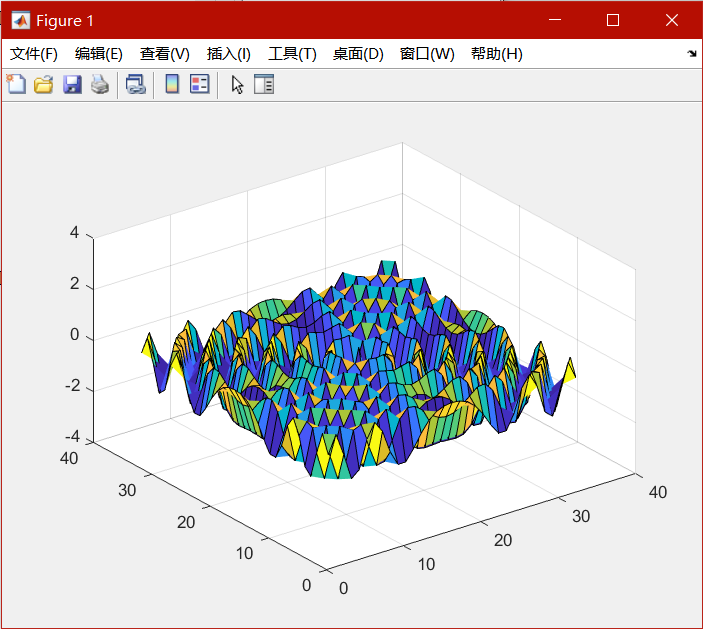
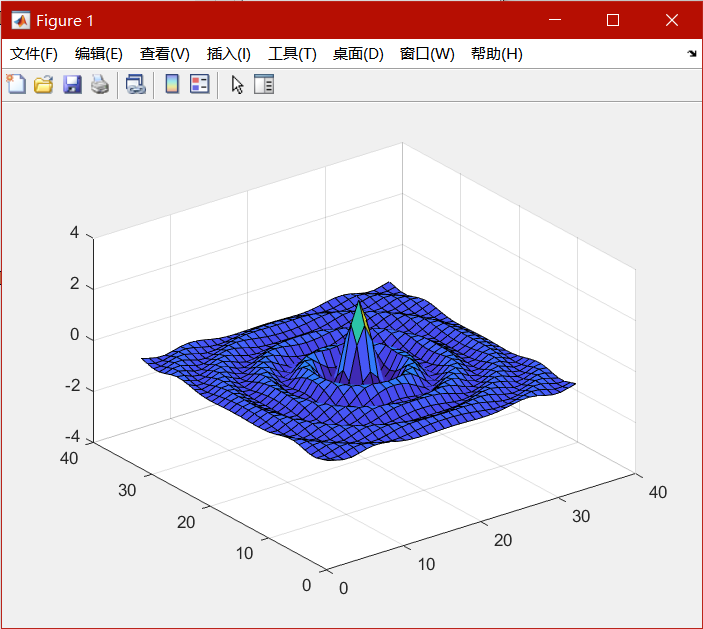
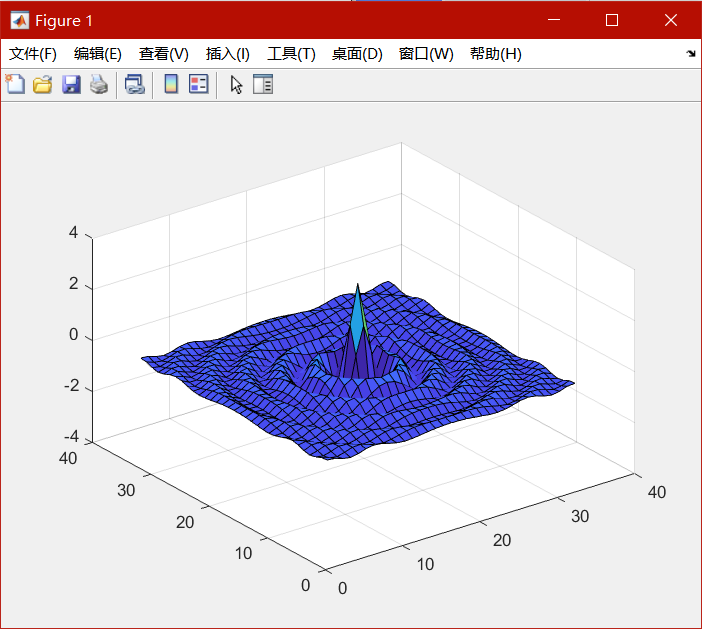
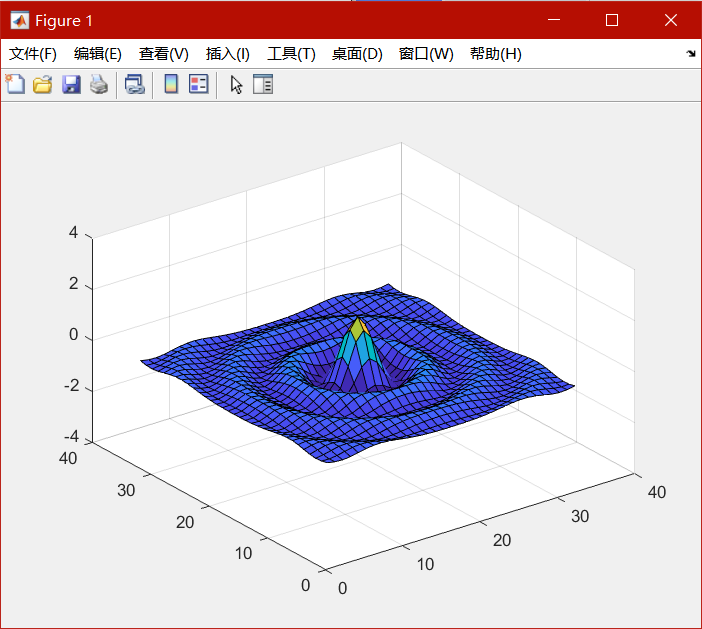
for n=-4:0.05:4

z = cos(r.\*n);

surf(z),view(-37,38),axis([0,40,0,40,-4,4]);

pause(0.1)

End



1.3

Assign a value to the variable x on the command line, e.g., x=4\* pi^2. What is the square root of x? What is the cosine of the square root of x?

%1.3

x = 4\*pi^2; % assign x

disp(x^0.5); % compute and display x^0.5

disp(cos(x^0.5)); % compute and display cos of x^0.5

6.2832

1

1.4

Assign a value to the variable y on the command line as follows: y = -1. What is the square root of y? Show that the answer is ans = 0 + 1.0000i Yes, MATLAB deals with complex numbers (not just real numbers). Hence the symbol i should not be used as an index or as a variable name. By default, it is equal to the square root of −1. (Also, when necessary, j is used in MATLAB as a symbol for √−1. Hence, it also should not be used as an index or as a variable name.) Give an example of how you have used complex numbers in your studies of mathematics and the sciences up to this point in your education.

%1.4

y = -1; % assign y

disp(y^0.5); % compute and display y^0.5

0.0000 + 1.0000i

2.1

Decide which of the following numbers are not acceptable in MATLAB, and state why:

%2.1

9,87

% not acceptable : cannot read an array without '[]'

(b) .0

(c) 25.82

(d) -356231

(e) 3.57\*e2

% not acceptable : cannot use '\*' and e together

(f) 3.57e2.1

% not acceptable : 'e2.1' is unacceptable

(g) 3.57e+2

(h) 3,57e-2

% not acceptable : cannot read an array without '[]'

2.2

State, giving reasons, which of the following are not valid MATLAB variable names:

%2.2

a2

a.2 not valid : can not use '.' in names

2a not valid : can not use numbers ahead

’a’one not valid : can not use ' in names

aone

\_x\_1 not valid : can not use '\_' ahead

miXedUp

pay day not valid : can not use ' ' in names

inf

Pay\_Day

\*min\*2 not valid : can not use '\*' in names

What

2.3

Translate the following expressions into MATLAB:

%2.3

%(a)

disp(p+(w/u));

%(b)

disp(p+(w/(u+v)));

%(c)

disp((p+(w/(u+v)))/(p+(w/(u-v))));

%(d)

disp(x^0.5);

%(e)

disp(y^(y+z));

%(f)

disp(x^(y^z));

%(g)

disp((x^y)^z);

%(h)

disp(x-((x^3)/prod(1:3))+((x^5)/prod(1:5)));

2.4

Translate the following into MATLAB statements:

%2.4

%(a) Add 1 to the value of i and store the result in i.

clear;

i = 222; % the value of i

i = i+1; %Add 1 to the value of i and store the result in i.

disp(i);

223

%(b) Cube i, add j to this, and store the result in i.

clear;

i = [1 2 3;4 5 6;7 8 9]; %Cube i

j = 2;

i = i + j; %add j to this, and store the result in i.

disp(i);

3 4 5

6 7 8

9 10 11

%(c) Set g equal to the larger of the two variables e and f.

clear;

e = 2; % value of e

f = 3; % value of f

if (e>=f) % compare

g = e;

else

g = f;

end

disp(g);

3

%(d) If d is greater than 0, set x equal to ?d.

clear;

d = 2; % value of d

x = 0; % value of x

if (d>0) % compare

x = -d;

end

disp(x);

-2

%(e) Divide the sum of a and b by the product of c and d, and store the

%result in x.

clear;

a = 2; % value of a

b = 3; % value of b

c = 2; % value of c

d = 3; % value of d

x = (a+b)/(c\*d); %compute

disp(x);

0.8333

2.5

%What’s wrong with the following MATLAB statements?

%2.5

%(a) n + 1 = n;

% can not assign to a formula.

%(b) Fahrenheit temp = 9\*C/5 + 32;

% there must be '\_' between two words.

%(c) 2 = x;

% can not assign to the right part.

2.6

%Write a program to calculate x,

%where \_ and a = 2, b = ?10, c = 12 (Answer 3.0)

%2.6

a = 2; % value of a

b = -10; % value of b

c = 12; % value of c

x = ((-b)+(((b^2)-(4\*a\*c))^0.5))/(2\*a);

% compute and store to x

disp (x);

3

2.7

%There are eight pints in a gallon and 1.76 pints in a liter.

%The volume of a tank is given as 2 gallons and 4 pints.

%Write a script that inputs this volume in gallons and pints

%and converts it to liters. (Answer: 11.36)

%2.7

x = input('please input a number\n'); % number of gallons

y = input('please input a number\n'); % number of pints

z = (8\*x) + y;% total pints

a = z/1.76;% pints to liters

disp(a);

please input a number

2

please input a number

4

11.3636

2.8

Write a program to calculate gasoline consumption. It should assign the distance traveled (in kilometers) and the amount of gas used (in liters) and compute the consumption in km/liter as well as in the more usual form of liters/100 km. Write some helpful headings so that your output looks something like this\_

%2.8

x = input('please input distance\n');

% number of distance

y = input('please input liters used\n');

% number of liters used

a = x/y;% km/L compute

b = (y/x)\*100;% L/100km compute

disp(['Distance ','Liters used ','km/L ','L/100km ']);

disp([x,y,a,b]);

please input distance

528

please input liters used

46.23

Distance Liters used km/L L/100km

528.0000 46.2300 11.4212 8.7557

2.9

Write some statements in MATLAB that exchange the contents of two variables a and b, using only one additional variable t.

%2.9

a=2; % value of a

b=3; % value of b

t=b; % give value of b to t

b=a; % give value of a to b

a=t; % give value of t(b) to a

disp(a);

disp(b);

3

2

2.10

Try Exercise 2.9 without using any additional variables!

%2.10

a=2; % value of a

b=3; % value of b

% change value of a and b

a=a+b;

b=a-b;

a=a-b;

disp(a);

disp(b);

3

2

2.11

If C and F are Celsius and Fahrenheit temperatures, respectively, the

formula for conversion from Celsius to Fahrenheit is F = 9C/5 + 32.

%2.11

(a) Write a script that will ask you for the Celsius temperature and display the Fahrenheit equivalent with some sort of comment, such as

The Fahrenheit temperature is:...

Try it out on the following Celsius temperatures (answers in parentheses): 0 (32), 100 (212), ?40 (?40!), 37 (normal human temper?ature: 98.6).

c = input('please input Celsius temperature\n');

% number of Celsius temperature

f = ((c\*9)/5)+32;% celsius2fahrenheit

disp(['The Fahrenheit temperature is:',num2str(f)]);

please input Celsius temperature

0

The Fahrenheit temperature is:32

please input Celsius temperature

100

The Fahrenheit temperature is:212

please input Celsius temperature

-40

The Fahrenheit temperature is:-40

please input Celsius temperature

37

The Fahrenheit temperature is:98.6

(b) Change the script to use vectors and array operations to compute the

Fahrenheit equivalents of Celsius temperatures ranging from 20 to 30 in steps of 1, and display them in two columns with a heading,

like this:

Celsius Fahrenheit

20.00 68.00

21.00 69.80

...

30.00 86.00

disp('Celsius Fahrenheit');

for c = 20:1:30

%list number of Celsius temperature

f = ((c\*9)/5)+32;

% celsius2fahrenheit

disp([num2str(c),' ',num2str(f)]);

end

Celsius Fahrenheit

20 68

21 69.8

22 71.6

23 73.4

24 75.2

25 77

26 78.8

27 80.6

28 82.4

29 84.2

30 86

2.12

Generate a table of conversions from degrees (first column) to radians (second column). Degrees should go from 0? to 360? in steps of 10?. Recall that π radians = 180?.

%2.12

title=['degrees radians'];

disp(title);

for d = 0:10:360

%list number of degrees

r = (d/180)\*pi;

% degrees2radians

disp([num2str(d),' ',num2str(r)]);

end

degrees radians

0 0

10 0.17453

20 0.34907

30 0.5236

40 0.69813

50 0.87266

60 1.0472

70 1.2217

80 1.3963

90 1.5708

100 1.7453

110 1.9199

120 2.0944

130 2.2689

140 2.4435

150 2.618

160 2.7925

170 2.9671

180 3.1416

190 3.3161

200 3.4907

210 3.6652

220 3.8397

230 4.0143

240 4.1888

250 4.3633

260 4.5379

270 4.7124

280 4.8869

290 5.0615

300 5.236

310 5.4105

320 5.5851

330 5.7596

340 5.9341

350 6.1087

360 6.2832

2.13

Set up a matrix (table) with degrees in the first column from 0 to 360 in steps of 30, sines in the second column, and cosines in the third column. Now try to add tangents in the fourth column. Can you figure out what’s going on? Try some variations of the format command.

%2.13

clear;

type compact;

A=zeros(13,3);

%set a martix

for first\_column = 0:30:360

y=(first\_column)/30+1;

%positon of martix

A(y,1)= first\_column;

%input degrees to martix

second\_column=sin((first\_column/180)\*pi);

%compute for sines

A(y,2)= second\_column;

%input sines to martix

third\_column=cos((first\_column/180)\*pi);

%compute for cosines

A(y,3)= third\_column;

%input cossines to martix

end

disp(A);

0 0 1.0000

30.0000 0.5000 0.8660

60.0000 0.8660 0.5000

90.0000 1.0000 0.0000

120.0000 0.8660 -0.5000

150.0000 0.5000 -0.8660

180.0000 0.0000 -1.0000

210.0000 -0.5000 -0.8660

240.0000 -0.8660 -0.5000

270.0000 -1.0000 -0.0000

300.0000 -0.8660 0.5000

330.0000 -0.5000 0.8660

360.0000 -0.0000 1.0000

2.14

Write some statements that display a list of integers from 10 to 20 inclusive, each with its square root next to it.

%2.14

clear;

type compact;

A=zeros(11,2);

%set a martix

for first\_column = 10:1:20

y=(first\_column-9);

%positon of martix

A(y,1)= first\_column;

%input integers to martix

second\_column=(first\_column^0.5);

%compute for roots

A(y,2)= second\_column;

%input roots to martix

end

disp(A);

10.0000 3.1623

11.0000 3.3166

12.0000 3.4641

13.0000 3.6056

14.0000 3.7417

15.0000 3.8730

16.0000 4.0000

17.0000 4.1231

18.0000 4.2426

19.0000 4.3589

20.0000 4.4721

2.15

Write a single statement to find and display the sum of the successive even integers 2, 4, . . . , 200. (Answer: 10,100)

%2.15

clear;

type compact;

answer=sum(2:2:200);

disp(answer);

10100

2.16

Ten students in a class take a test. The marks are out of 10. All the marks are entered in a MATLAB vector, marks. Write a statement to find and display the average mark. Try it on the following:

5 8 0 10 3 8 5 7 9 4 (Answer: 5.9)

Hint: Use the mean function.

%2.16

clear;

type compact;

marks=[5 8 0 10 3 8 5 7 9 4];%input marks

avgmarks=mean(marks);%compute avg

disp(avgmarks);%output

5.9000

2.17

What are the values of x and a after the following statements have been executed?

%2.17

(a) a = 0;

(b) i = 1;

(c) x = 0;

(d) a = a + i;

(e) x = x + i / a;

(f) a = a + i;

(g) x = x + i / a;

(h) a = a + i;

(i) x = x + i / a;

(j) a = a + i;

(k) x = x + i / a;

clear;

type compact;

%compute

a = 0;

i = 1;

x = 0;

a = a + i;

x = x + i / a;

a = a + i;

x = x + i / a;

a = a + i;

x = x + i / a;

a = a + i;

x = x + i / a;

%output

disp(x);

disp(a);

2.0833

4

2.18

Rewrite the statements in Exercise 2.17 more economically by using a for loop. Can you do even better by vectorizing the code?

%2.18

clear;

type compact;

%input

a=0;

i=1;

x=0;

%for loop 4

for counter=1:1:4

a=a+i;%compute

x=x+i/a;%compute

end

%output

disp(x);

disp(a);

2.0833

4

2.19

Work out by hand the output of the following script for n = 4:

n = input( ’Number of terms? ’ );

s = 0;

for k = 1:n

s = s + 1 / (k ^ 2);

end;

disp(sqrt(6 \* s))

If you run this script for larger and larger values of n, you will find that the output approaches a well-known limit. Can you figure out what it is? Now rewrite the script using vectors and array operations.

%2.19

Work out by hand the output of the following script for n = 4: 2.923

for n=100:1000000:10000000

s = 0;

for k = 1:n

s = s + 1 / (k ^ 2);

end

disp(sqrt(6 \* s));

end

3.1321

3.1416

3.1416

3.1416

3.1416

3.1416

3.1416

3.1416

3.1416

3.1416

the output approaches a well-known limit 3.1216.

2.20

Work through the following script by hand. Draw up a table of the values of i, j, and m to show how they change while the script executes. Check your answers by running the script.

v = [3 1 5];

i = 1;

for j = v

i = i + 1;

if i == 3

i = i + 2;

m = i + j;

end

end

%2.20

clear;

type compact;

v = [3 1 5];

i = 1;

for j = v

i = i + 1;

if i == 3

i = i + 2;

m = i + j;

end

disp(i);

end

v = [3 1 5];

i = 1;

for j = v

i = i + 1;

if i == 3

i = i + 2;

m = i + j;

end

disp(j);

end

v = [3 1 5];

i = 1;

for j = v

i = i + 1;

if i == 3

i = i + 2;

m = i + j;

end

disp(m);

end

2 %i

5

6

3 %j

1

5

6 %m

6

6

2.21

The steady-state current I flowing in a circuit that contains a resistance R = 5, capacitance C = 10, and inductance L = 4 in series is given by I = \_ where E = 2 and ω = 2 are the input voltage and angular frequency, respectively. Compute the value of I . (Answer: 0.0396)

%2.21

clear;

type compact;

%input

r=5;

c=10;

l=4;

e=2;

w=2;

%compute

i=e/(((r^2)+(((2\*pi\*w\*l)-(1/(2\*pi\*w\*c)))^2))^0.5);

%display

disp(i);

0.0396

2.22

The electricity accounts of residents in a very small town are calculated as follows:

If 500 units or fewer are used, the cost is 2 cents per unit.

If more than 500 but not more than 1000 units are used, the cost is $10 for the first 500 units and 5 cents for every unit in excess of 500.

If more than 1000 units are used, the cost is $35 for the first 1000 units plus 10 cents for every unit in excess of 1000.

A basic service fee of $5 is charged, no matter how much electricity is used.

Write a program that enters the following five consumptions into a vector and uses a for loop to calculate and display the total charge for each one: 200, 500, 700, 1000, 1500. (Answers: $9, $15, $25, $40, $90)

%2.22

clear;

type compact;

%input amount of used

used=[200 500 700 1000 1500];

for n=used

if n<=500%less than 500

disp(n\*0.02+5);

end

if (500<n)&&(n<=1000)%between 500 and 1000

disp((n-500)\*0.05+10+5);

end

if 1000<n%more than 1000

disp((n-1000)\*0.1+35+5);

end

end

9

15

25

40

90

2.23

Suppose you deposit $50 in a bank account every month for a year. Every month, after the deposit has been made, interest at the rate of 1% is added to the balance: After one month the balance is $50.50, and after two months it is $101.51. Write a program to compute and print the balance each month for a year. Arrange the output to look something like this:

MONTH MONTH-END BALANCE

1 50.50

2 101.51

3 153.02

...

12 640.47

%2.23

clear;

type compact;

A=zeros(12,2);

%set a martix

second\_column=0;

for first\_column = 1:1:12

y=first\_column;

%positon of martix

A(y,1)= first\_column;

%input months to martix

second\_column=second\_column+50;

second\_column=second\_column\*1.01;

%compute money

A(y,2)= second\_column;

%input money to martix

end

disp(A);

1.0000 50.5000

2.0000 101.5050

3.0000 153.0200

4.0000 205.0503

5.0000 257.6008

6.0000 310.6768

7.0000 364.2835

8.0000 418.4264

9.0000 473.1106

10.0000 528.3417

11.0000 584.1252

12.0000 640.4664

2.24

If you invest $1000 for one year at an interest rate of 12%, the return is $1120 at the end of the year. But if interest is compounded at the rate of 1% monthly (i.e., 1/12 of the annual rate), you get slightly more interest because it is compounded. Write a program that uses a for loop to compute the balance after a year of compounding interest in this way. The answer should be $1126.83. Evaluate the formula for this result separately as a check: 1000\*1.0112.

%2.24

clear;

type compact;

money=1000;%money at first

for date=1:1:12

money=money\*1.01; %money compute

end

disp(money);

1.1268e+03

2.25

A plumber opens a savings account with $100,000 at the beginning of January. He then makes a deposit of $1000 at the end of each month for the next 12 months (starting at the end of January). Interest is calculated and added to his account at the end of each month (before the $1000 deposit is made). The monthly interest rate depends on the amount A in his account at the time interest is calculated, in the following way:

A ≤ 1 10 000: 1%

1 10 000 < A ≤ 1 25 000: 1.5%

A > 1 25 000: 2%

Write a program that displays, under suitable headings, for each of the 12 months, the situation at the end of the month as follows: the number of the month, the interest rate, the amount of interest, and the new balance.

(Answer: Values in the last row of output should be 12, 0.02, 2534.58, 130263.78.)

%2.25

clear;

type compact;

format short g;

A=zeros(12,4);

%set a martix

money=100000;

rate = 0;

for first\_column = 1:1:12

y=first\_column;

%positon of martix

A(y,1)= first\_column;

%input month to martix

if money<=110000

rate=0.01;

end

if (110000<money)&&(money<=125000)

rate=0.015;

end

if 125000<money

rate=0.02;

end

%compute for rate to martix

A(y,2)= rate;

%input interest rate to martix

third\_column=(rate)\*money;

%compute for interest to martix

A(y,3)= third\_column;

%input amount of interest to martix

forth\_column=(money+A(y,3)+1000);

money=forth\_column;

%compute for new balance

A(y,4)= forth\_column;

%input new balance to martix

end

disp(A);

1 0.01 1000 1.02e+05

2 0.01 1020 1.0402e+05

3 0.01 1040.2 1.0606e+05

4 0.01 1060.6 1.0812e+05

5 0.01 1081.2 1.102e+05

6 0.015 1653 1.1286e+05

7 0.015 1692.8 1.1555e+05

8 0.015 1733.2 1.1828e+05

9 0.015 1774.2 1.2106e+05

10 0.015 1815.8 1.2387e+05

11 0.015 1858.1 1.2673e+05

12 0.02 2534.6 1.3026e+05

2.26

It has been suggested that the population of the United States may be modeled by the formula \_ where t is the date in years. Write a program to compute and display the population every ten years from 1790 to 2000. Try to plot a graph of the population against time as well (Figure 9.16 shows this graph compared with actual data). Use your program to find out if the population ever reaches a “steady state” (i.e., stops changing).

%2.26

clear;

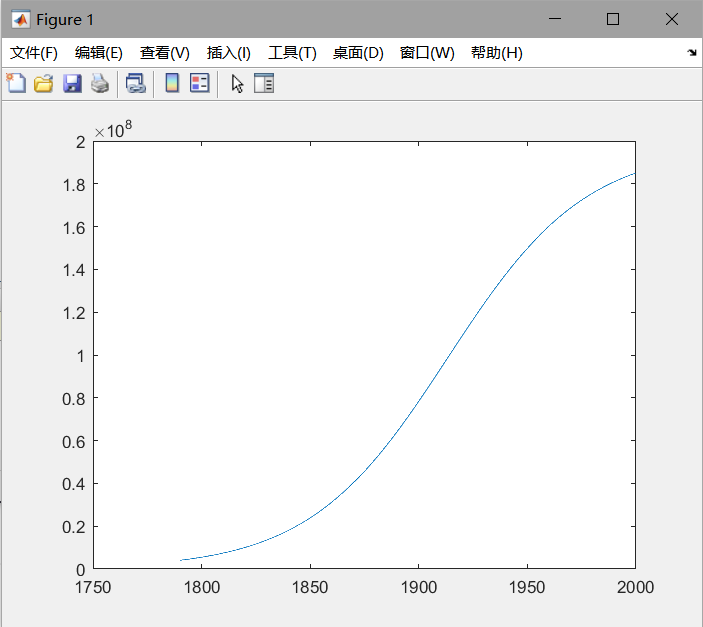
type compact;

format short g;

t=1790:0.0001:2000;

p=(197273000)./(1.+exp(1).^((-0.03134).\*(t-1913.25)));

plot(t,p);



Not stable

2.27

A mortgage bond (loan) of amount L is obtained to buy a house. The interest rate r is 15%. The fixed monthly payment P that will pay off the bond loan over N years is given by the formula.

Write a program to compute and print P if N = 20 and the bond is for $50,000. You should get $658.39.

%2.27

clear;

type compact;

format short g;

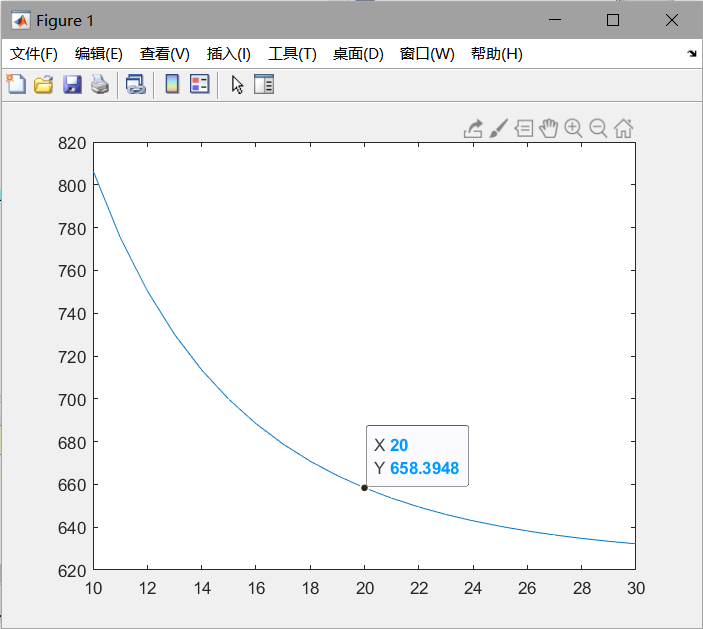
r=0.15;%rate

n=10:1:30;%N

L=50000;%bond

p=(r.\*L.\*((1+r./12).^(12.\*n))./(12.\*((1+r./12).^(12.\*n)-1)));

plot(n,p);



See how P changes with N by running the program for different values of N (use input). Can you find a value for which the payment is less than $625?

clear;

type compact;

format short g;

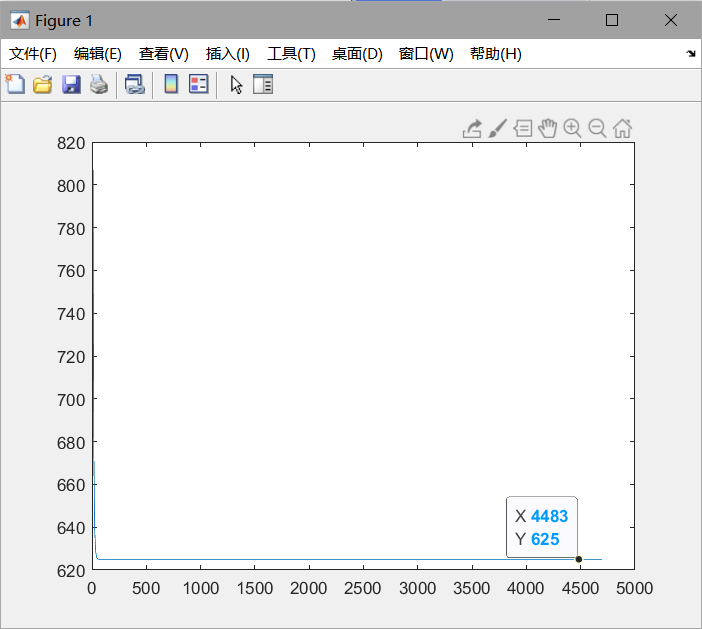
r=0.15;%rate

n=10:1:10000;%N

L=50000;%bond

p=(r.\*L.\*((1+r./12).^(12.\*n))./(12.\*((1+r./12).^(12.\*n)-1)));

plot(n,p);



Can never be less than 625

(c) Go back to N = 20 and examine the effect of different interest rates. You should see that raising the interest rate by 1% (0.01) increases the monthly payment by about $37.

clear;

type compact;

format short g;

r=0.16;

n=20;%N

L=50000;%bond

p=(r.\*L.\*((1+r./12).^(12.\*n))./(12.\*((1+r./12).^(12.\*n)-1)));

disp(p);

695.63

2.28

It is useful to work out how the period of a bond repayment changes if you increase or decrease P . The formula for N is given by \_.

%2.28

(a) Write a new program to compute this formula. Use the built-in function log for the natural logarithm ln. How long will it take to pay off a loan of $50,000 at $800 a month if the interest remains at 15%? (Answer: 10.2 years—nearly twice as fast as when paying $658 a month.)

clear;

type compact;

format short g;

r=0.15;

L=50000;

p=800;

n=(log((p)./(p-(r.\*L)/(12))))./(12.\*log(1+(r./12)));

disp(n);

10.195

(b) Use your program to find out by trial and error the smallest monthly payment that will pay off the loan this side of eternity. Hint: recall that it is not possible to find the logarithm of a negative number, so P must not be less than rL/12.

clear;

type compact;

format short g;

% r=0.15;

% L=50000;

% p=800;

% n=(log((p)./(p-(r.\*L)/(12))))./(12.\*log(1+(r./12)));

% disp(n);

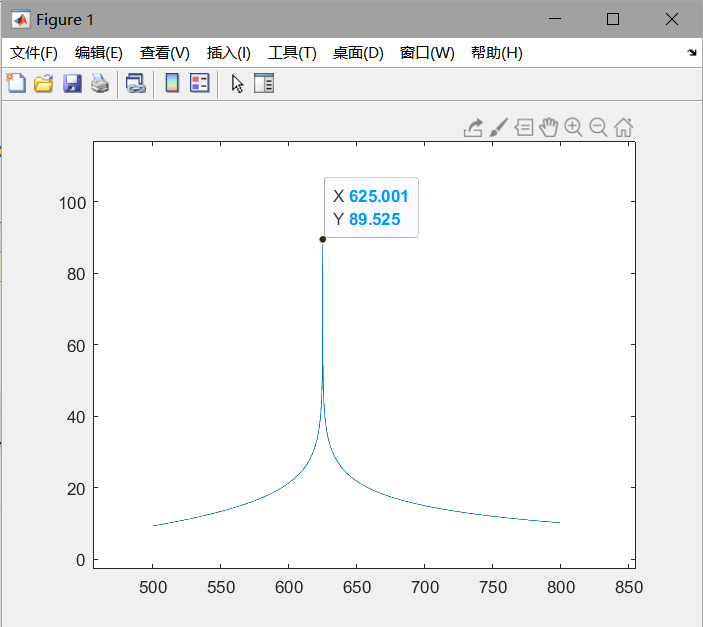
r=0.15;

L=50000;

p=500:0.001:800;

n=(log((p)./(p-(r.\*L)/(12))))./(12.\*log(1+(r./12)));

plot(p,n);



smallest monthly payment : 625

3.1

The structure plan in this example defines a geometric construction. Carry out the plan by sketching the construction:

1. Draw two perpendicular x- and y-axes

2. Draw the points A (10, 0) and B (0, 1)

3. While A does not coincide with the origin repeat: Draw a straight line joining A and B Move A one unit to the left along the x-axis Move B one unit up on the y-axis

4. Stop

%3.1

clear;

type compact;

%1

% x=0;y=0;

% plot(x,y);

% axis([-5 15 -5 15]);

%2

% x1=[10 0];y1=[0 1];

% plot(x1,y1,'.');

% axis([-5 15 -5 15]);

%3

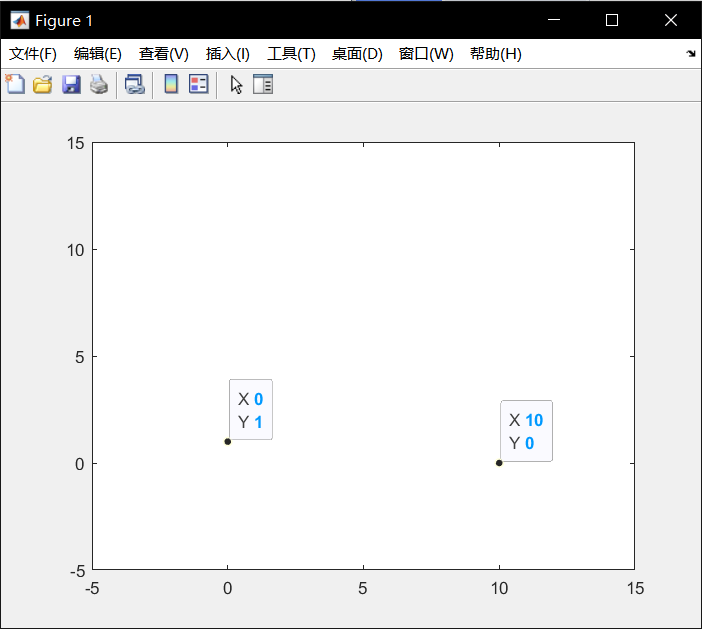
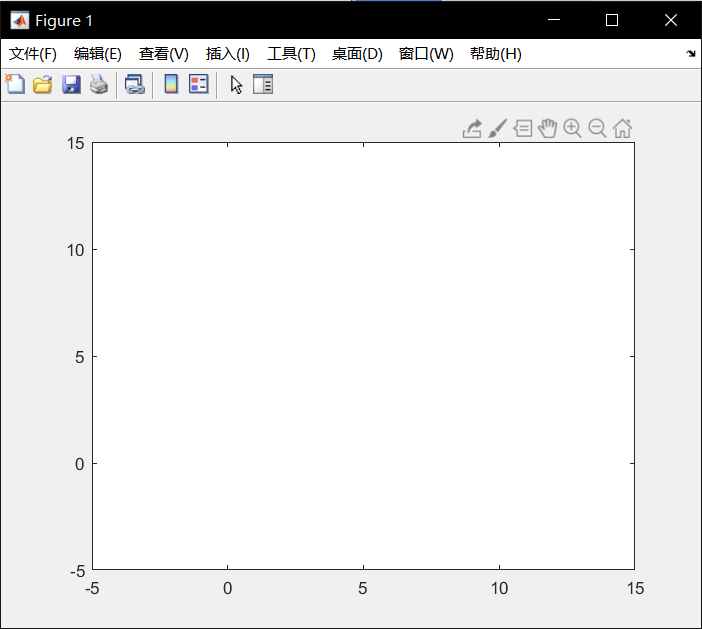
x1=[10 0];y1=[0 1];

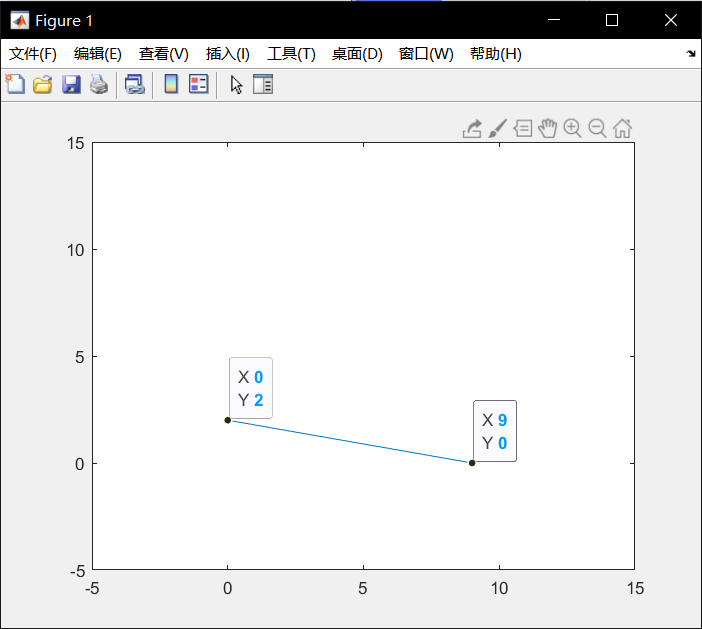
x1(1)=x1(1)-1;

y1(2)=y1(2)+1;

plot(x1,y1,'-');

axis([-5 15 -5 15]);





3.2

Consider the following structure plan, where M and N represent MATLAB variables:

1. Set M = 44 and N = 28

2. While M not equal to N repeat: While M>N repeat: Replace value of M by M ? N While N>M repeat: Replace value of N by N ? M

3. Display M

4. Stop

(a) Work through the structure plan, sketching the contents of M and N during execution. Give the output.

(b) Repeat (a) for M = 14 and N = 24.

(c) What general arithmetic procedure does the algorithm carry out (try more values of M and N if necessary)?

%3.2

clear;

type compact;

% value of m and n input

m=44;

n=28;

while m~=n % while m not equal to n

while m>n % while m bigger than n

m=(m-n);

end

while n>m % while n bigger than m

n=(n-m);

end

end

disp(m);

4

3.3

Write a program to convert a Fahrenheit temperature to Celsius. Test it on the data in Exercise 2.11 (where the reverse conversion is done).

%3.3

clear;

type compact;

f=68:1.8:86;%input Fahrenheit temperature

c=(5/9)\*(f-32);%compute Celsius temperature

disp(c);

列 1 至 7

20.0000 21.0000 22.0000 23.0000 24.0000 25.0000 26.0000

列 8 至 11

27.0000 28.0000 29.0000 30.0000

3.4

Write a script that inputs any two numbers (which may be equal) and displays the larger one with a suitable message or, if they are equal, displays a message to that effect.

%3.4

clear;

type compact;

x = input('please input a number\n'); % number of x

y = input('please input a number\n'); % number of y

if x==y % equal

disp('the two numbers are equal');

elseif x>y %x is igger

disp('the first number is bigger');

disp(x);

else %y is igger

disp('the second number is bigger');

disp(y);

end

please input a number

3

please input a number

5

the second number is bigger

5

3.5

Write a script for the general solution to the quadratic equation ax2 +bx + c = 0. Use the structure plan in Section 3.2.2. Your script should be able to handle all possible values of the data a, b, and c. Try it out on the following

values:

(a) 1, 1, 1 (complex roots)

(b) 2, 4, 2 (equal roots of -1.0)

(c) 2, 2, -12 (roots of 2.0 and -3.0)

The structure plan in Section 3.2.2 is for programming languages that cannot handle complex numbers. MATLAB can. Adjust your script so that it can also find complex roots. Test it on case (a); the roots are -0.5±0.866i.

%3.5

clear;

type compact;

% input abc

a = input('please input a\n');

b = input('please input b\n');

c = input('please input c\n');

if a == 0

if b == 0

if c == 0

disp('Solution indeterminate');

else

disp('There is no solution');

end

else

x = -c/b;

disp(x);

disp('only one root: equation is linear');

end

elseif b^2 < 4\*a\*c

x1 = (-b + (b^2 - 4\*a\*c)^0.5)/(2\*a);

x2 = (-b - (b^2 - 4\*a\*c)^0.5)/(2\*a);

disp('Complex roots');

disp([x1 x2]);

elseif b^2 == 4\*a\*c

x = -b/(2\*a);

disp(x);

disp('equal roots');

else

x1 = (-b + (b^2 - 4\*a\*c)^0.5)/(2\*a);

x2 = (-b - (b^2 - 4\*a\*c)^0.5)/(2\*a);

disp([x1 x2]);

end

please input a

1

please input b

1

please input c

1

Complex roots

-0.5000 + 0.8660i -0.5000 - 0.8660i

please input a

2

please input b

4

please input c

2

-1

equal roots

please input a

2

please input b

2

please input c

-12

2 -3

3.6

Develop a structure plan for the solution to two simultaneous linear equations (i.e., the equations of two straight lines). Your algorithm must be able to handle all possible situations; that is, lines intersecting, parallel, or coincident. Write a program to implement your algorithm, and test it on some equations for which you know the solutions, such as

x + y = 3

2x - y = 3

(x = 2, y = 1).

Hint: Begin by deriving an algebraic formula for the solution to the system:

ax + by = c

dx + ey = f

The program should input the coefficients a, b, c, d, e, and f . We will see in Chapter 6 that MATLAB has a very elegant way of solving systems of equations directly, using matrix algebra. However, it is good for the development of your programming skills to do it the long way, as in this exercise.

%3.6

clear;

type compact;

%ax + by = c

%dx + ey = f

a = input('please input a\n'); % number of a

b = input('please input b\n'); % number of b

c = input('please input c\n'); % number of c

d = input('please input d\n'); % number of d

e = input('please input e\n'); % number of e

f = input('please input f\n'); % number of f

%x + (b/a)y = c/a

%x + (e/d)y = f/d

%x + gy = h

%x + iy = j

g=(b/a);

h=(c/a);

i=(e/d);

j=(f/d);

if g==i

if h==j

disp('coincident');%coincident

else

disp('parallel');%parallel

end

else

%(g-i)y=(h-j)

y=(h-j)/(g-i);

x=h-g\*y;

disp('lines intersecting');%lines intersecting

show=['x = :' num2str(x) ' y = :' num2str(y)];

disp(show);

end

please input a

1

please input b

1

please input c

3

please input d

2

please input e

-1

please input f

3

lines intersecting

x = :2 y = :1

3.7

We wish to examine the motion of a damped harmonic oscillator. The small amplitude oscillation of a unit mass attached to a spring is given by the formula y = e?(R/2)t sin(ω1t), where ω21 = ω2o ? R2/4 is the square of the natural frequency of the oscillation with damping (i.e., with resistance to motion); ω2o = k is the square of the natural frequency of undamped oscillation; k is the spring constant; and R is the damping coefficient. Consider k = 1 and vary R from 0 to 2 in increments of 0.5. Plot y versus t for t from 0 to 10 in increments of 0.1.

Hint: Develop a solution procedure by working backwards through the problem statement. Starting at the end of the problem statement, the solution procedure requires the programmer to assign the input variables first followed by the execution of the formula for the amplitude and ending

with the output in graphical form.

%3.7

clear;

type compact;

for r=0:0.5:2

t=0:0.1:10;

k=1;

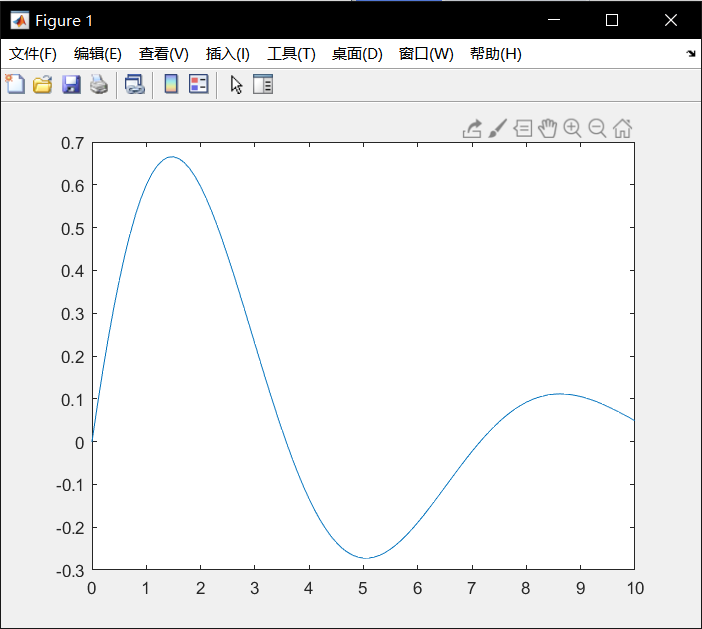
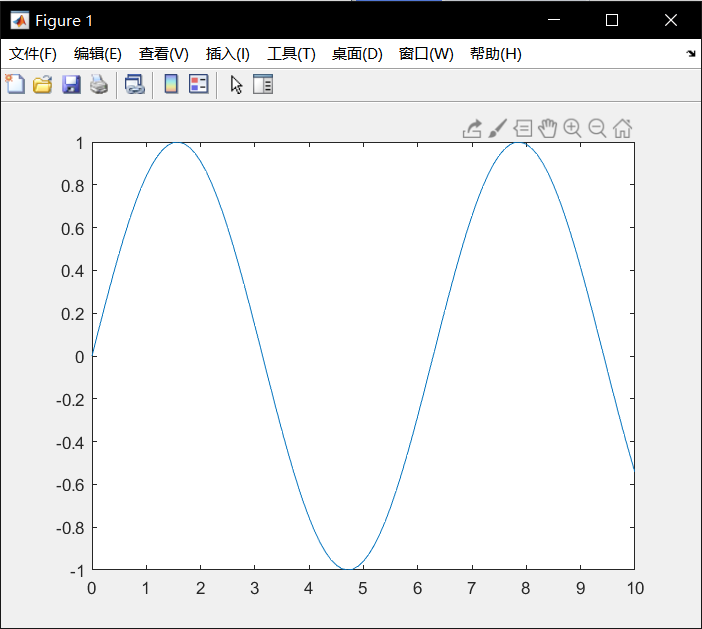
w1=(k-(r^2)/4)^2;

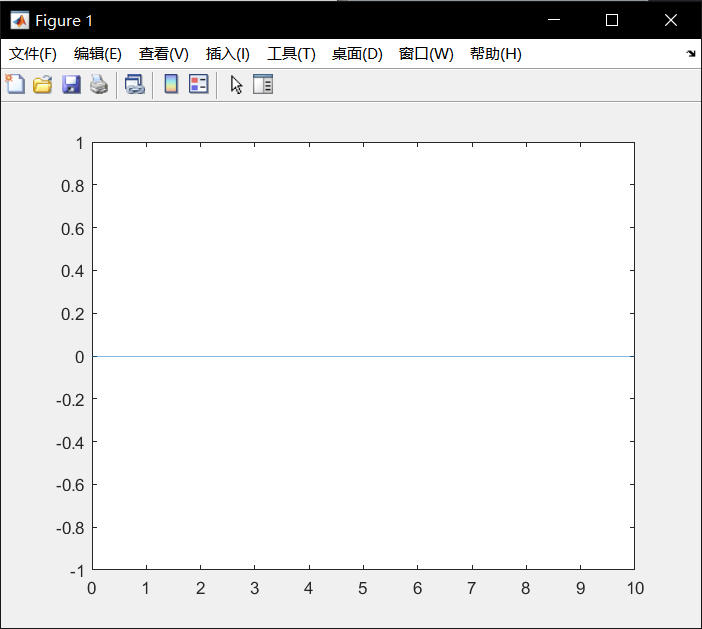
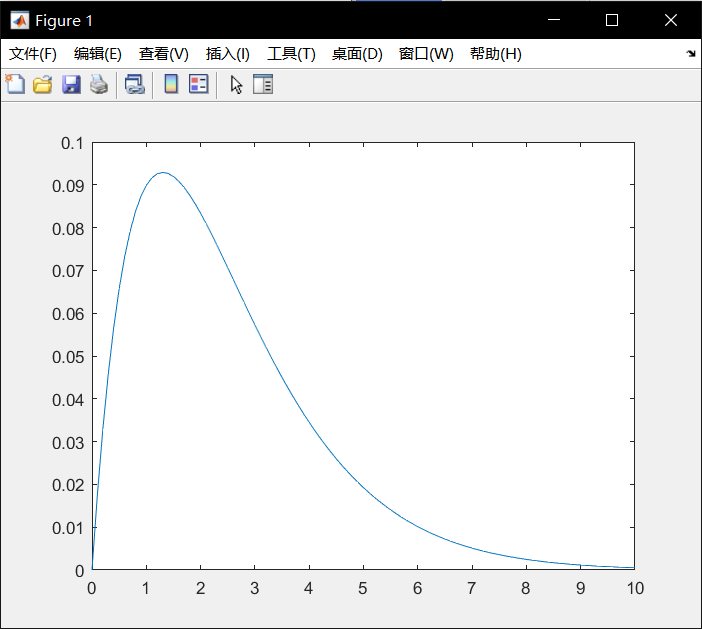
y=exp(-(r./2).\*t).\*sin(w1.\*t);

plot(t,y);

pause(3);

end





3.8

Let’s examine the shape of a uniform cable hanging under its own weight. The shape is described by the formula y = cosh(x/c). This shape is called a uniform catenary. The parameter c is the vertical distance from y = 0 where the bottom of the catenary is located. Plot the shape of the catenary between x = ?10 and x = 10 for c = 5. Compare this with the same result for c = 4.

Hint: The hyperbolic cosine, cosh, is a built-in MATLAB function that is used in a similar way to the sine function, sin.

%3.8

clear;

type compact;

%input

c=5;

x=-10:0.001:10;

y=cosh(x/c);%compute y

%plot

plot(x,y);

pause(5);

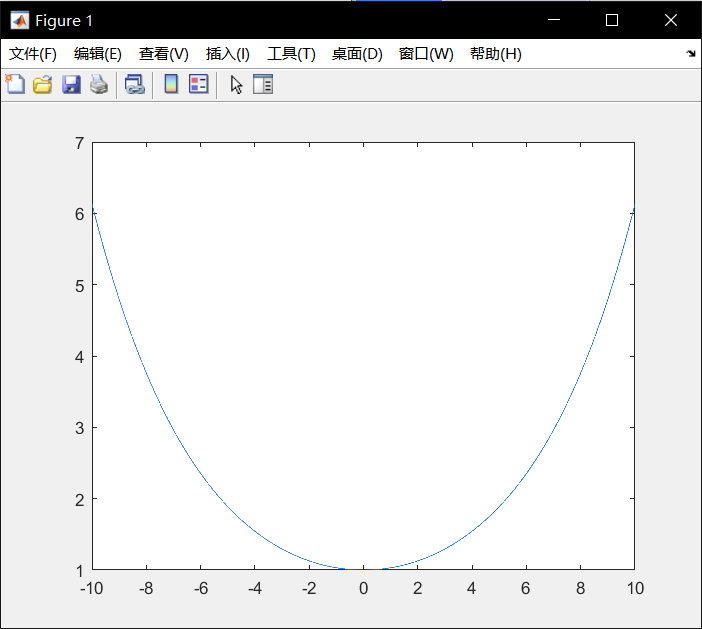
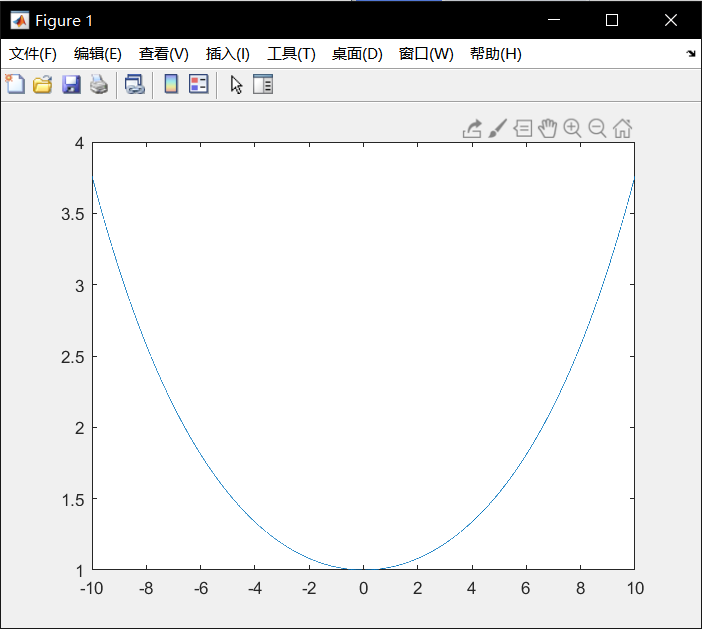
c=4;

x=-10:0.001:10;

y=cosh(x/c);%compute y

%plot

plot(x,y);



4.1

Write some MATLAB statements which will:

(a) find the length C of the hypotenuse of a right-angle triangle in terms of the lengths A and B of the other two sides;

(b) find the length C of a side of a triangle given the lengths A and B of the other two sides and the size in degrees of the included angle θ, using the cosine rule:

C2 = A2 + B2 - 2AB cos(θ).

%4.1

clear;

type compact;

%(a)

a = input('please input a\n'); % number of a

b = input('please input b\n'); % number of b

disp((a^2+b^2)^0.5); %compute and display c

please input a

3

please input b

4

5

clear;

type compact;

%(b)

a = input('please input a\n'); % number of a

b = input('please input b\n'); % number of b

o = input('please input angle\n'); % number of angle

disp((a^2+b^2-2\*a\*b\*cos((o/180)\*pi))^0.5);

%compute and display c

please input a

3

please input b

4

please input angle

90

5

4.2

Translate the following formulae into MATLAB expressions:

(a) ln(x + x2 + a2)

(b) [e3t + t2 sin(4t)] cos2(3t)

(c) 4 tan-1(1) (inverse tangent)

(d) sec2(x) + cot(y)

(e) cot-1(|x/a|) (use MATLAB’s inverse cotangent)

%4.2

clear;

type compact;

%(a)

x=1;

a=2;

a1=log(x+x^2+a^2);

disp(a1);

%(b)

t=1;

b1=(exp(1)^(3\*t)+(t^2)\*sin(4\*t))\*(cos(3\*t)^2);

disp(b1);

%(c)

c1=4\*((tan(1))^(-1));

disp(c1);

%(d)

x=1;

y=2;

d1=sec(x)^2+cot(y);

disp(d1);

%(e)

x=-1;

a=2;

e1=(cot(abs(x/a)))^(-1);

disp(e1);

1.7918

18.9438

2.5684

2.9679

0.5463

4.3

There are 39.37 inches in a meter, 12 inches in a foot, and three feet in a yard. Write a script to input a length in meters (which may have a decimal part) and convert it to yards, feet and inches. (Check: 3.51 meters converts to 3 yds 2 ft 6.19 in.)

%4.3

clear;

type compact;

% number of meters

meters = input('please input meters\n');

inches = 39.37\*meters;%compute for inches

feet = inches/12;%compute for feet

yards = feet/3;%compute for yards

disp(['yards is ' num2str(yards)]);

disp(['feet is ' num2str(feet)]);

disp(['inches is ' num2str(inches)]);

please input meters

1

yards is 1.0936

feet is 3.2808

inches is 39.37

4.4

A sphere of mass m1 impinges obliquely on a stationary sphere of mass m2, the direction of the blow making an angle α with the line of motion of the impinging sphere. If the coefficient of restitution is e it can be shown that the impinging sphere is deflected through an angle β such that

tan(β) = m2(1 + e)tan(α) m1 ? em2 + (m1 + m2)tan2(α)

Write a script to input values of m1, m2, e, and α (in degrees) and to compute the angle β in degrees.

%4.4

clear;

type compact;

m1 = input('please input m1\n'); % number of m1

m2 = input('please input m2\n'); % number of m2

e = input('please input e\n'); % number of e

a1 = input('please input a\n'); % number of a

a=(a1/180)\*pi;

%compute b

b = atan((m2\*(1+exp(1)\*tan(a)))/(m1-exp(1)\*m2+(m1+m2)\*((tan(a))^2)));

b1=(b/pi)\*180;

disp(b1);

please input m1

1

please input m2

1

please input e

1

please input a

30

-67.7415

4.5

Section 2.7 has a program for computing the members of the sequence xn = an/n!. The program displays every member xn computed. Adjust it to display only every 10th value of xn.

Hint: the expression rem(n, 10) will be zero only when n is an exact multiple of 10. Use this in an if statement to display every tenth value of xn.

%4.5

clear;

type compact;

a=10;%input

x(1)=10;

%for n from 2 to 100

for n = 2:1:100

x(n)=(a\*x(n-1))/n;

if rem(n,10)==0

%every 10th value of xn

disp(num2str(n));

disp(x(n));

end

end

10

2.7557e+03

20

41.1032

30

0.0038

40

1.2256e-08

50

3.2879e-15

60

1.2018e-22

70

8.3482e-31

80

1.3972e-39

90

6.7308e-49

100

1.0715e-58

4.6

To convert the variable mins minutes into hours and minutes you would use fix(mins/60) to find the whole number of hours, and rem(mins, 60) to find the number of minutes left over. Write a script which inputs a number of minutes and converts it to hours and minutes.

Now write a script to convert seconds into hours, minutes and seconds. Try out your script on 10 000 seconds, which should convert to 2 hours 46 minutes and 40 seconds.

%4.6

clear;

type compact;

%a

mins = input('please input mins\n');% number of mins

show=(['hours :' num2str(fix(mins/60)) ' mins :' num2str(rem(mins,60))]);

disp(show);

please input mins

100

hours :1 mins :40

clear;

type compact;

%b

seconds = input('please input seconds\n');% number of seconds

show=(['hours :' num2str(fix(fix(seconds/60)/60)) ' mins :' num2str(fix(rem(seconds,3600)/60)) ' seconds :' num2str(rem(seconds,60))]);

disp(show);

please input seconds

10000

hours :2 mins :46 seconds :40

4.7

Design an algorithm (i.e., write the structure plan) for a machine which must give the correct amount of change from a $100 note for any purchase costing less than $100. The plan must specify the number and type of all notes and coins in the change, and should in all cases give as few notes and coins as possible. (If you are not familiar with dollars and cents, use your own monetary system.)

%4.7

clear;

type compact;

% 100 rmb

rmb=100;

x = input('please input a value\n'); % number of value

rmb=rmb-x;

disp('back');%money back

for n=1:1:15

if rmb>=50

disp('50');

rmb=rmb-50;

elseif rmb>=20

disp('20');

rmb=rmb-20;

elseif rmb>=10

disp('10');

rmb=rmb-10;

elseif rmb>=5

disp('5');

rmb=rmb-5;

elseif rmb>=1

disp('1');

rmb=rmb-1;

elseif rmb>=0.5

disp('0.5');

rmb=rmb-0.5;

elseif rmb>=0.1

disp('0.1');

rmb=rmb-0.1;

else

disp('finish');

break;

end

end

please input a value

66

back

20

10

1

1

1

1

finish

4.8

A uniform beam is freely hinged at its ends x = 0 and x = L, so that the ends are at the same level. It carries a uniformly distributed load of W per unit length, and there is a tension T along the x-axis. The deflection y of the beam a distance x from one end is given by

y = WEI

T 2 #

cosh[a(L/2 ? x)]

cosh(aL/2) ? 1$ + W x(L ? x)

2T ,

where a2 = T /EI , E being the Young’s modulus of the beam, and I is the moment of inertia of a cross-section of the beam. The beam is 10 m long, the tension is 1 000 N, the load 100 N/m, and EI is 104.

Write a script to compute and plot a graph of the deflection y against x (MATLAB has a cosh function). To make the graph look realistic you will have to override MATLAB’s automatic axis scaling with the statement

axis( [xmin xmax ymin ymax] )

after the plot statement, where xmin etc. have appropriate values.

%4.8

clear;

type compact;

%initialization

L=10;

t=1000;

w=100;

ei=10^4;

a=(t/ei)^0.5;

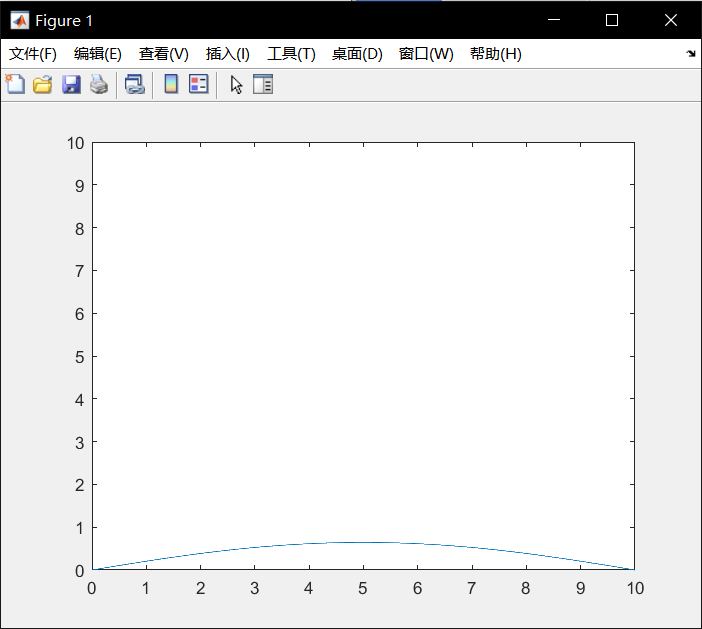
%compute

x=0:0.01:10;

y=(w.\*ei./t.^2).\*(cosh(a.\*(L./2-x))./cosh((a.\*L)./2)-1)+(w.\*x.\*(L-x)./(2.\*t));

plot(x,y);

axis([0 10 0 10]);



5.1

Determine the values of the following expressions yourself before checking your answers using MATLAB. You may need to consult Table 5.3.

%5.1

(a) 1 & -1

(b) 13 & ~(-6)

(c) 0 < -2|0

(d) ~[1 0 2] \* 3

(e) 0 <= 0.2 <= 0.4

(f) 5>4>3

(g) 2>3&1

clear;

type compact;

%(a)

disp(1 & -1);

%(b)

disp(13 & ~(-6));

%(c)

disp(0 < -2|0);

%(d)

disp(~[1 0 2] \* 3);

%(e)

disp((0 <= 0.2) && (0.2 <= 0.4));

%(f)

disp((5 > 4) && (4 > 3));

%(g)

disp((2 > 3) & 1);

1

0

0

0 3 0

1

1

0

5.2

Given that a = [1 0 2] and b = [0 2 2] determine the values of the following expressions. Check your answers with MATLAB.

%5.2

(a) a ~= b

(b) a<b

(c) a<b<a

(d) a<b<b

(e) a | (~a)

(f) b & (~b)

(g) a(~(~b))

(h) a = b == a (determine final value of a)

clear;

type compact;

% initialization

a = [1 0 2];

b = [0 2 2];

% (a) a ~= b

disp(a~=b);

% (b) a<b

disp(a<b);

% (c) a<b<a

disp((a<b)&(b<a));

% (d) a<b<b

disp((a<b)&(b<b));

% (e) a | (~a)

disp(a | (~a));

% (f) b & (~b)

disp(b & (~b));

% (g) a(~(~b))

disp(a(~(~b)));

% (h) a = b == a (determine final value of a)

a = (b == a);

disp(a);

1 1 0

0 1 0

0 0 0

0 0 0

1 1 1

0 0 0

0 2

0 0 1

5.3

Write some MATLAB statements on the command line which use logical vectors to count how many elements of a vector x are negative, zero or positive. Check that they work, e.g., with the vector

[-4 0 5 -3 0 3 7 -1 6]

%5.3

clear;

type compact;

%initialization

a=[-4 0 5 -3 0 3 7 -1 -6];

disp(sum(a<0));%negative

disp(sum(a==0));%zero

disp(sum(a>0));%positive

4

2

3

5.4

The Receiver of Revenue (Internal Revenue Service) decides to change the tax table used in Section 5.5 slightly by introducing an extra tax bracket and changing the tax-rate in the third bracket, as shown in the table on the next page.

Amend the logical vector script to handle this table, and test it on the following list of incomes (dollars): 5000, 10 000, 15 000, 22 000, 30 000, 38 000 and 50 000.

%5.4

clear;

type compact;

% initialization

inc = [5000 10000 15000 22000 30000 38000 50000];

%<10000

tax = 0.1 \* inc .\* (inc <= 10000);

%10000<20000

tax = tax + (inc > 10000 & inc <= 20000).\* (0.2 \* (inc-10000) + 1000);

%20000<40000

tax = tax + (inc > 20000 & inc <= 40000).\* (0.3 \* (inc-20000) + 3000);

%40000<

tax = tax + (inc > 40000) .\* (0.5 \* (inc-40000) + 9000);

disp(tax);

列 1 至 6

500 1000 2000 3600 6000 8400

列 7

14000

5.5

A certain company offers seven annual salary levels (dollars): 12 000, 15 000, 18 000, 24 000, 35 000, 50 000 and 70 000. The number of em?ployees paid at each level are, respectively: 3000, 2500, 1500, 1000, 400, 100 and 25. Write some statements at the command line to find the fol?lowing:

(a) The average salary level. Use mean. (Answer: 32 000)

(b) The number of employees above and below this average salary level. Use logical vectors to find which salary levels are above and below the average level. Multiply these logical vectors element by element with the employee vector, and sum the result. (Answer: 525 above, 8000 below)

(c) The average salary earned by an individual in the company (i.e., the total annual salary bill divided by the total number of employees). (Answer: 17 038.12).

%5.5

clear;

type compact;

%initialization

a=[12000 15000 18000 24000 35000 50000 70000];

b=[3000 2500 1500 1000 400 100 25];

%average salary level

avl=mean(a);

disp(avl);

%higher

hi= sum((a>avl).\*b);

disp(hi);

%lower

lo= sum((a<avl).\*b);

disp(lo);

%average salary

av=sum(a.\*b)./sum(b);

disp(av);

32000

525

8000

1.7038e+04

5.6

Write some statements on the command line to remove the largest element(s) from a vector. Try it out on x = [1 2 5 0 5]. The idea is to end up with [1 2 0] in x. Use find and the empty vector [ ].

%5.6

clear;

type compact;

%initialization

x=[1 2 5 0 5];

x(find(x==max(x)))=[];%delete 5

disp(x);

1 2 0

5.7

The electricity accounts of residents in a very small rural community are calculated as follows:

! if 500 units or less are used the cost is 2 cents per unit;

! if more than 500, but not more than 1000 units are used, the cost is $10 for the first 500 units, and then 5 cents for every unit in excess of 500;

! if more than 1000 units are used, the cost is $35 for the first 1000 units plus 10 cents for every unit in excess of 1000;

! in addition, a basic service fee of $5 is charged, no matter how much electricity is used.

The five residents use the following amounts (units) of electricity in a certain month: 200, 500, 700, 1000 and 1500. Write a program which uses logical vectors to calculate how much they must pay. Display the results in two columns: one for the electricity used in each case, and one for amount owed. (Answers: $9, $15, $25, $40, $90)

%5.7

clear;

type compact;

% initialization

use = [200 500 700 1000 1500];

%<500

amount = 0.02 \* use .\* (use <= 500);

%500<1000

amount = amount + (use > 500 & use <= 1000).\* (0.05 \* (use-500) + 10);

%1000<

amount = amount + (use > 1000) .\* (0.1 \* (use-1000) + 35);

%service fee

amount = amount + 5;

disp(amount);

9 15 25 40 90

6.1

Set up any 3×3 matrix a. Write some command-line statements to perform the following operations on a:

(a) interchange columns 2 and 3;

(b) add a fourth column (of 0s);

(c) insert a row of 1s as the new second row of a (i.e. move the current second and third rows down);

(d) remove the second column.

%6.1

clear;

type compact;

%initialization

A=[1 2 3 4;4 3 2 1;5 6 7 8;8 7 6 5];

%interchange columns 2 and 3

A(:,[2,3])=A(:,[3,2]);

disp(A);

%add a fourth column (of 0s);

A(end+1,:)=0;

disp(A);

%insert a row of 1s as the new second row of a

A=[A(1,:);[1 1 1 1];A(2:5,:)];

disp(A);

%remove the second column

A=[A(:,1),A(:,3:4)];

disp(A);

1 3 2 4

4 2 3 1

5 7 6 8

8 6 7 5

1 3 2 4

4 2 3 1

5 7 6 8

8 6 7 5

0 0 0 0

1 3 2 4

1 1 1 1

4 2 3 1

5 7 6 8

8 6 7 5

0 0 0 0

1 2 4

1 1 1

4 3 1

5 6 8

8 7 5

0 0 0

6.2

Compute the limiting probabilities for the student in Section 6.5 when he starts at each of the remaining intersections in turn, and confirm that the closer he starts to the cafe, the more likely he is to end up there. Compute P^50 directly. Can you see the limiting probabilities in the first row?

%6.2

clear;

type compact;

n = 6;

P = zeros(n); % all elements set to zero

for i = 3:6

P(i,i-1) = 2/3;

P(i-2,i-1) = 1/3;

end

P(1,1) = 1;

P(6,6) = 1;

x = [0 1 0 0 0 0]'; % remember x must be a column vector!

for t = 1:50

x = P \* x;

if t==50

disp( [t x'] );

end

end

% for x = [0 1 0 0 0 0]

50.0000 0.4839 0.0000 0 0.0000 0 0.5161

% for x = [0 0 1 0 0 0]

50.0000 0.2258 0 0.0000 0 0.0000 0.7742

% for x = [0 0 0 1 0 0]

50.0000 0.0968 0.0000 0 0.0000 0 0.9032

% for x = [0 0 0 0 1 0]

50.0000 0.0323 0 0.0000 0 0.0000 0.9677

6.3

Solve the equations

2x - y + z = 4

x + y + z = 3

3x - y - z = 1

using the left division operator. Check your solution by computing the residual. Also compute the determinant (det) and the condition estimator (rcond). What do you conclude?

%6.3

clear;

type compact;

%initialization

a=[2,-1,1;1,1,1;3,-1,-1];

b=[4,3,1]';

x=a\b;%left division

disp(x);

1.0000

-0.0000

2.0000

6.4

This problem, suggested by R.V. Andree, demonstrates ill conditioning (where small changes in the coefficients cause large changes in the solution). Use the left division operator to show that the solution of the system

x + 5.000y = 17.0

1.5x + 7.501y = 25.503

is x = 2, y = 3. Compute the residual.

Now change the term on the right-hand side of the second equation to 25.501, a change of about one part in 12 000, and find the new solution and the residual. The solution is completely different. Also try changing this term to 25.502, 25.504, etc. If the coefficients are subject to experimental errors, the solution is clearly meaningless. Use rcond to find the condition estimator and det to compute the determinant. Do these values confirm ill conditioning?

Another way to anticipate ill conditioning is to perform a sensitivity analysis on the coefficients: change them all in turn by the same small percentage, and observe what effect this has on the solution.

%6.4

clear;

type compact;

%initialization

a=[1,5;1.5,7.501];

b=[17,25.503]';

x=a\b;%left division

disp(x);

disp(cond(a,1));

%25.503

2.0000

3.0000

1.1252e+05

%25.502

7.0000

2.0000

1.1252e+05

%25.504

-3

4

1.1252e+05

clear;

type compact;

%initialization

a=[1,5;1.5,7.501];

b=[17,25.503]';

x=a\b;%left division

disp(x);

disp(cond(a,1));%cond

disp(rcond(a));%rcond

disp(det(a));%det

2.0000

3.0000

1.1252e+05

8.8872e-06

1.0000e-03

%ill conditioning

6.5

Use sparse to represent the Leslie matrix in Section 6.4. Test your representation by projecting the rabbit population over 24 months.

%6.5

clear;

type compact;

% Leslie matrix population model

n = 3;

L = zeros(n); % all elements set to zero

L(1,2) = 9;

L(1,3) = 12;

L(2,1) = 1/3;

L(3,2) = 0.5;

d(1:48)=0;

x = [0 0 1]'; % remember x must be a column vector!

for t = 1:48

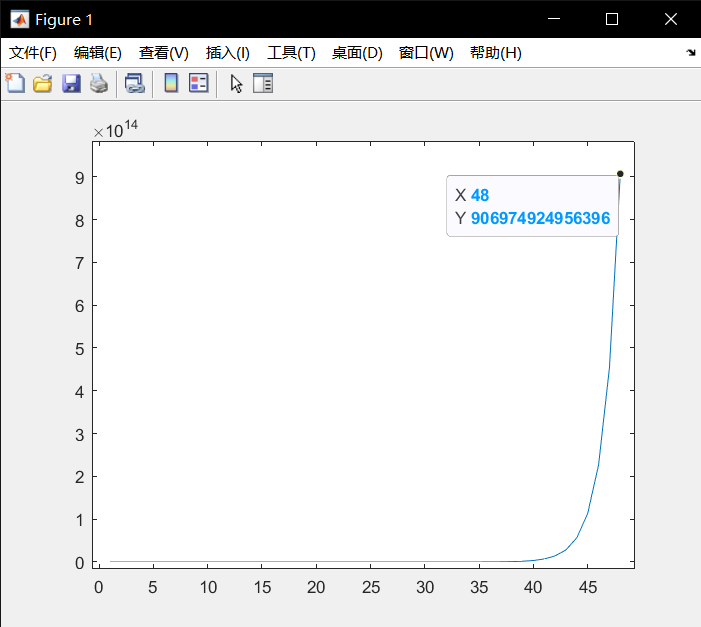
x = L \* x;

d(t)=sum(x);%compute for (t,sum)

end

x1=1:1:48;

plot(x1,d(x1));%disp the graph



6.6

If you are familiar with Gauss reduction it is an excellent programming exercise to code a Gauss reduction directly with operations on the rows of the augmented coefficient matrix. See if you can write a function

x = mygauss(a, b)

to solve the general system Ax = b. Skillful use of the colon operator in the row operations can reduce the code to a few lines!

Test it on A and b with random entries, and on the systems in Section 6.6 and Exercise 16.4. Check your solutions with left division.

%6.6

function x=mygauss(a,b)

x=a\b;

end

clear;

type compact;

%initialization

a = [10 7 8 7;7 5 6 5;8 6 10 9;7 5 9 10];

b = [32 23 33 31]';

%gauss

x = mygauss(a,b);

disp(x);

1.0000

1.0000

1.0000

1.0000

7.1

Change the function stars of Section 7.2 to a function pretty so that it will draw a line of any specified character. The character to be used must be passed as an additional input (string) argument, e.g., pretty(6, ’$’) should draw six dollar symbols.

%7.1

function pretty(n)

asteriks = char(abs('$')\*ones(1,n));

disp( asteriks );

end

clear;

type compact;

pretty(20);%disp

$$$$$$$$$$$$$$$$$$$$

7.2

Write a script newquot.m which uses the Newton quotient [f (x + h) - f (x)]/h to estimate the first derivative of f (x) = x3 at x = 1, using successively smaller values of h: 1, 10-1, 10-2, etc. Use a function M-file for f (x).

Rewrite newquot as a function M-file able to take a handle for f (x) as an input argument.

%7.2

function newton(h)

disp(((1+h)^3-(1)^3)/h);%compute

clear;

type compact;

newton(1);%h=1

newton(0.1);%h=0.1

newton(0.01);%h=0.01

7

3.3100

3.0301

7.3

Write and test a function double(x) which doubles its input argument, i.e., the statement x = double(x) should double the value in x.

%7.3

function [back,x]=double1(x)

back=2\*x;%function

clear;

type compact;

%initialization

x=1;

disp(double1(x));

2

7.4

Write and test a function swop(x, y) which will exchange the values of its two input arguments.

%7.4

function [x,y]=swop(x,y)

a=x;

b=y;

x=b;

y=a;

end

clear;

type compact;

x=1;

y=2;

[x,y]=swop(x,y);%function swop

disp(x);

disp(y);

2

1

7.5

Write your own MATLAB function to compute the exponential function directly from the Taylor series:

ex = 1 + x + x22! + x33! + ...

The series should end when the last term is less than 10-6. Test your function against the built-in function exp, but be careful not to make x too large—this could cause rounding error.

%7.5

function [b]=taylor(x)

b=1;

for t=1:1:100

plus=(x^t)/prod(1:t);%the next to plus

if plus>10e-6%whether enough

b=b+plus;

else

break

end

end

clear;

type compact;

disp(taylor(10));%use function

disp(exp(10));%test

2.2026e+04

2.2026e+04

7.6

If a random variable X is distributed normally with zero mean and unit standard deviation, the probability that 0 ≤ X ≤ x is given by the standard normal function !(x). This is usually looked up in tables, but it may be approximated as follows:

!(x) = 0.5 ? r(at + bt2 + ct3),

where a = 0.4361836, b = ?0.1201676, c = 0.937298, r = exp(?0.5x2)/√2π, and t = 1/(1 + 0.3326x).

Write a function to compute !(x), and use it in a program to write out its values for 0 ≤ x ≤ 4 in steps of 0.1. Check: !(1) = 0.3413.

%7.6

function [back]=deviation(x)

%initialization

a=0.4361836;

b=-0.1201676;

c=0.937298;

r=exp(-0.5.\*(x.^2))./(2.\*pi).^0.5;

t=1./(1+0.3326.\*x);

back=0.5-r.\*(a.\*t+b.\*t.\*t+c.\*t.\*t.\*t);%compute

clear;

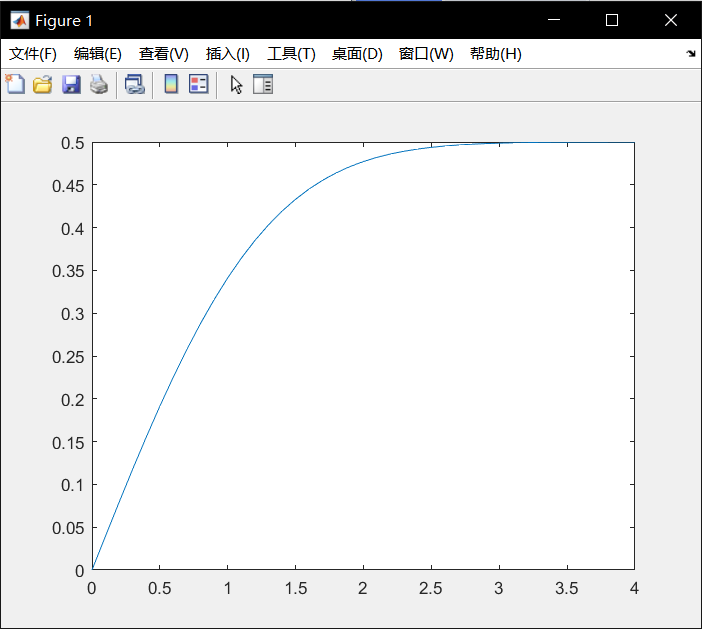
type compact;

disp(deviation(1));%check

x=0:0.1:4;%0-4

plot(x,deviation(x));

0.3413



7.7

Write a function

function [x1, x2, flag] = quad( a, b, c )

which computes the roots of the quadratic equation ax2 + bx + c = 0. The input arguments a, b and c (which may take any values) are the coefficients of the quadratic, and x1, x2 are the two roots (if they exist), which may be equal. See Figure 3.3 in Chapter 3 for the structure plan. The output argument flag must return the following values, according to the number and type of roots:

0: no solution (a = b = 0, c = 0);

1: one real root (a = 0, b = 0, so the root is ?c/b);

2: two real or complex roots (which could be equal if they are real);

99: any x is a solution (a = b = c = 0).

Test your function on the data in Exercise 3.5.

%7.7

function [x1, x2, flag] = quad( a, b, c )

if a == 0

if b == 0

if c == 0

disp('Solution indeterminate');

flag=99;

else

disp('There is no solution');

flag=0;

end

else

x1 = -c/b;

x2 = 0;

disp('only one root: equation is linear');

disp(x1);

flag=1;

end

elseif b^2 < 4\*a\*c

x1 = (-b + (b^2 - 4\*a\*c)^0.5)/(2\*a);

x2 = (-b - (b^2 - 4\*a\*c)^0.5)/(2\*a);

disp('Complex roots');

flag=2;

disp([x1 x2]);

elseif b^2 == 4\*a\*c

x1 = -b/(2\*a);

x2 = x1;

disp('equal roots');

disp(x1);

flag=1;

else

x1 = (-b + (b^2 - 4\*a\*c)^0.5)/(2\*a);

x2 = (-b - (b^2 - 4\*a\*c)^0.5)/(2\*a);

disp([x1 x2]);

flag=2;

end

clear;

type compact;

%initialization

a=1;

b=1;

c=1;

[x1,x2,flag] = quad(a,b,c);%compute

disp(flag);

%initialization

a=2;

b=4;

c=2;

[x1,x2,flag] = quad(a,b,c);%compute

disp(flag);

%initialization

a=2;

b=2;

c=-12;

[x1,x2,flag] = quad(a,b,c);%compute

disp(flag);

Complex roots

-0.5000 + 0.8660i -0.5000 - 0.8660i

2

equal roots

-1

1

2 -3

2

7.8

The Fibonacci numbers are generated by the sequence

1, 1, 2, 3, 5, 8, 13,...

Can you work out what the next term is? Write a recursive function f(n) to compute the Fibonacci numbers F0 to F20, using the relationship

Fn = Fn?1 + Fn?2,

given that F0 = F1 = 1.

%7.8

function back=fib(n)

if n==1

back=1;

elseif n==2

back=[1 1];

else

b=fib(n-1);

%fn=f(n-1)+f(n-2)

back=[b,b(end)+b(end-1)];

end

clear;

type compact;

disp(fib(21));

%use the function of fib 0-20

列 1 至 6

1 1 2 3 5 8

列 7 至 12

13 21 34 55 89 144

列 13 至 18

233 377 610 987 1597 2584

列 19 至 21

4181 6765 10946

7.9

The first three Legendre polynomials are P0(x) = 1, P1(x) = x, and P2(x) = (3x2 ? 1)/2. There is a general recurrence formula for Legendre polynomials, by which they are defined recursively:

(n + 1)Pn+1(x) ? (2n + 1)xPn(x) + nPn?1(x) = 0.

Define a recursive function p(n,x) to generate Legendre polynomials, given the form of P0 and P1. Use your function to compute p(2,x) for a few values of x, and compare your results with those using the analytic form of P2(x) given above.

%7.9

function [back]=leg(n,x)

if n==0

back=1;%n=0

elseif n==1

back=x;%n=1

else

%continue for n>2

back=((2\*n-1)\*x\*leg((n-1),x)-(n-1)\*leg((n-2),x))/n;

end

end

clear;

type compact;

%use the function of leg p2(2)

disp(leg(2,2));

%use the analytic given

disp((3\*2\*2-1)/2);

5.5000

5.5000

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.

%8.1

clear;

type compact;

%vectorize

money=[1000,0,0,0,0,0,0,0,0,0,0];%vector

for y=1:1:10%money of 10 years

money(y+1)=money(y)+sum(money(y)\*((1.01).^12-1));

end

disp(money);%disp

1.0e+03 \*

列 1 至 7

1.0000 1.1268 1.2697 1.4308 1.6122 1.8167 2.0471

列 8 至 11

2.3067 2.5993 2.9289 3.3004

8.2

There are many formulae for computing π (the ratio of a circle’s circumference to its diameter). The simplest is (8.4) which comes from putting x = 1 in the series (8.5)

(a) Write a program to compute π 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 π.

(b) Rearranging the series speeds up the convergence:\_ Write a program to compute π 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 π is\_ Use this formula to compute π. 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)?

%8.2

(a)

clear;

type compact;

num=1:1:100;%100 times

x=4\*(sum((-1).^(num+1)./(2.\*num-1)));

disp(x);

num=1:1:1000;%1000 times

x=4\*(sum((-1).^(num+1)./(2.\*num-1)));

disp(x);

num=1:1:10000;%10000 times

x=4\*(sum((-1).^(num+1)./(2.\*num-1)));

disp(x);

3.1316

3.1406

3.1415

(b)

clear;

type compact;

num=1:1:100;%100 times

x=8\*(sum(1./((num.\*4-3).\*(num.\*4-1))));

disp(x);

num=1:1:1000;%1000 times

x=8\*(sum(1./((num.\*4-3).\*(num.\*4-1))));

disp(x);

num=1:1:10000;%10000 times

x=8\*(sum(1./((num.\*4-3).\*(num.\*4-1))));

disp(x);

3.1366

3.1411

3.1415

(c)

clear;

type compact;

num=1:1:100;%100 times

x=4\*(sum(6.\*(sum((1./8).^(2.\*num-1).\*((-1).^(num+1))./(2.\*num-1)))+2.\*(sum((1/57).^(2.\*num-1).\*((-1).^(num+1))./(2.\*num-1)))+(sum((1/239).^(2.\*num-1).\*((-1).^(num+1))./(2.\*num-1)))));

disp(x);

num=1:1:1000;%1000 times

x=4\*(sum(6.\*(sum((1./8).^(2.\*num-1).\*((-1).^(num+1))./(2.\*num-1)))+2.\*(sum((1/57).^(2.\*num-1).\*((-1).^(num+1))./(2.\*num-1)))+(sum((1/239).^(2.\*num-1).\*((-1).^(num+1))./(2.\*num-1)))));

disp(x);

num=1:1:10000;%10000 times

x=4\*(sum(6.\*(sum((1./8).^(2.\*num-1).\*((-1).^(num+1))./(2.\*num-1)))+2.\*(sum((1/57).^(2.\*num-1).\*((-1).^(num+1))./(2.\*num-1)))+(sum((1/239).^(2.\*num-1).\*((-1).^(num+1))./(2.\*num-1)))));

disp(x);

3.1416

3.1416

3.1416

(d)

%all ready vectorize

8.3

The following method of computing π is due to Archimedes:

1. Let A = 1 and N = 6

2. Repeat 10 times, say:\_

3. Stop.

Write a program to implement the algorithm.

%8.3

clear;

type compact;

%initialization

a=1;

n=6;

for times=1:1:10%repeat 10 times

n=2\*n;

a=(2-(4-a^2)^0.5)^0.5;

l=n\*a/2;

u=l/(1-(a^2)/2)^0.5;

p=(u+l)/2;

e=(u-l)/2;

output=[num2str(n),num2str(p),num2str(e)];

disp(output);%disp

end

123.22160.1158

243.160.027387

483.14610.0067578

963.14270.001684

1923.14190.00042066

3843.14170.00010514

7683.14162.6285e-05

15363.14166.5711e-06

30723.14161.6428e-06

61443.14164.1069e-07

8.4

Write a program to compute a table of the function f (x) 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.

%8.4

clear;

type compact;

%red using increments of 0.2

x1=-1:0.2:1;

y1=x1.\*sin(pi.\*(1+20.\*x1)./2);

plot(x1,y1,'r');

hold on

%green using increments of 0.1

x2=-1:0.1:1;

y2=x2.\*sin(pi.\*(1+20.\*x2)./2);

plot(x2,y2,'g');

hold on

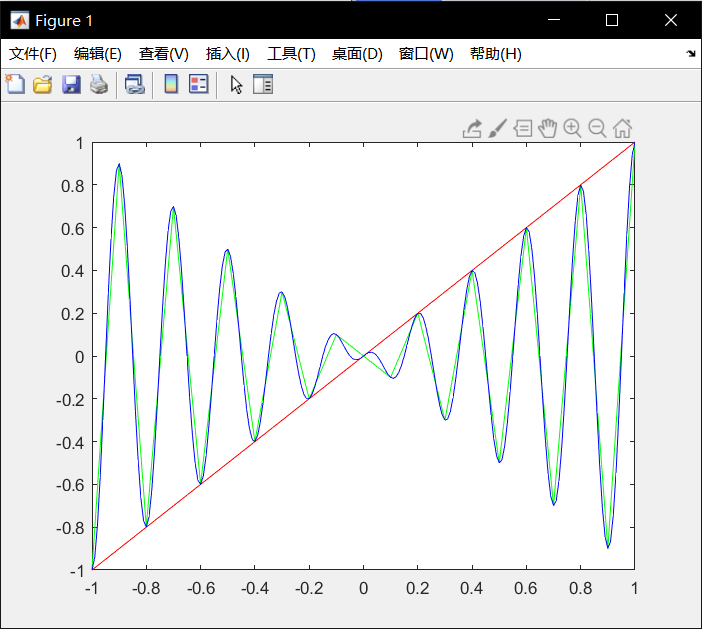
%blue using increments of 0.01

x3=-1:0.01:1;

y3=x3.\*sin(pi.\*(1+20.\*x3)./2);

plot(x3,y3,'b');

hold on



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.

%8.5

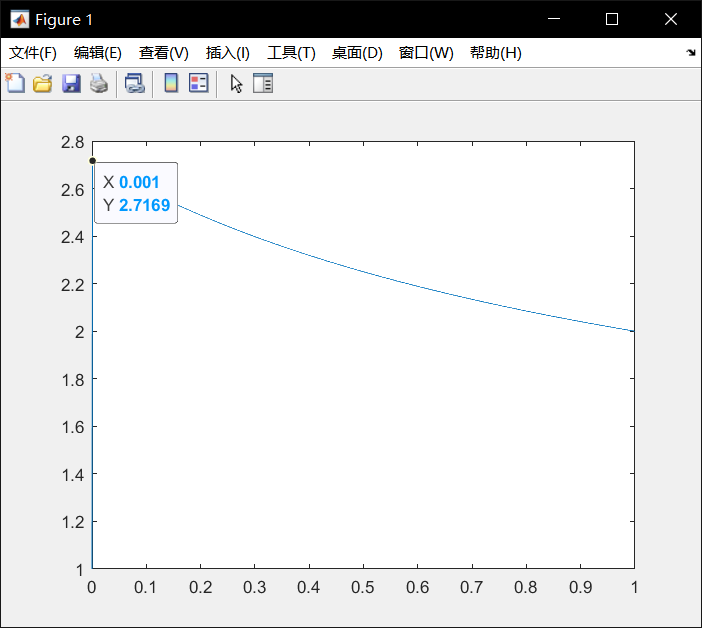
clear;

type compact;

x=0:0.001:1;%x get closer and closer to x

y=(1+x).^(1./x);%value of e

plot(x,y);%plot



8.6

A square wave of period T may be defined by the function f (t) The Fourier series for f (t) is given by F (t). 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.

%8.6

clear;

type compact;

%n=1

for num=1:1:221%times

t=(num-111)/100;%value of t

T=1.1;

k=0:1:1;

%F(t) f(num)=(4./pi).\*sum((1./(2.\*k+1)).\*sin(((2.\*k+1).\*pi.\*t)./T));

end

t = -1.1:0.01:1.1;%value of t

plot(t,f,'r');

hold on

%n=3

for num=1:1:221%times

t=(num-111)/100;%value of t

T=1.1;

k=0:1:3;

%F(t) f(num)=(4./pi).\*sum((1./(2.\*k+1)).\*sin(((2.\*k+1).\*pi.\*t)./T));

end

t = -1.1:0.01:1.1;%value of t

plot(t,f,'g');

hold on

%n=6

for num=1:1:221%times

t=(num-111)/100;%value of t

T=1.1;

k=0:1:6;

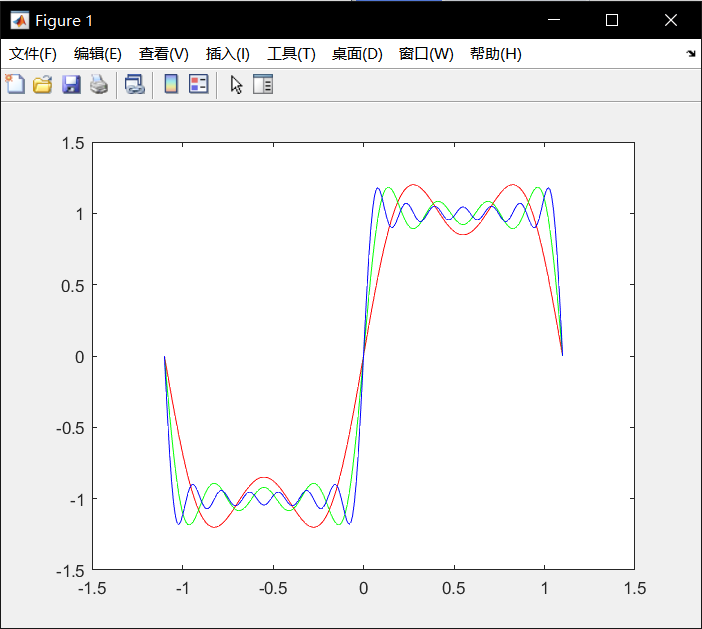
%F(t) f(num)=(4./pi).\*sum((1./(2.\*k+1)).\*sin(((2.\*k+1).\*pi.\*t)./T));

end

t = -1.1:0.01:1.1;%value of t

plot(t,f,'b');

hold on



%n=20

for num=1:1:221%times

t=(num-111)/100;%value of t

T=1.1;

k=0:1:20;

%F(t)

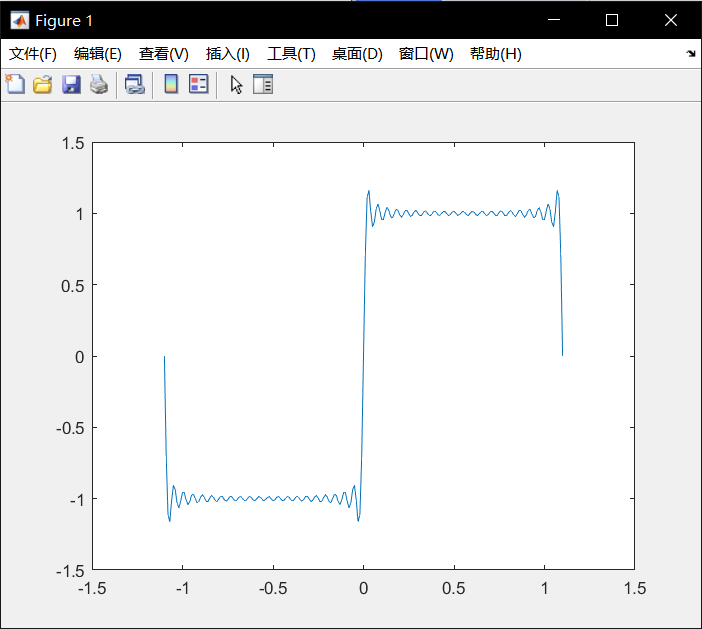
f(num)=(4./pi).\*sum((1./(2.\*k+1)).\*sin(((2.\*k+1).\*pi.\*t)./T));

end

t = -1.1:0.01:1.1;%value of t

plot(t,f);

hold on



%n=200

for num=1:1:221%times

t=(num-111)/100;%value of t

T=1.1;

k=0:1:200;

%F(t)

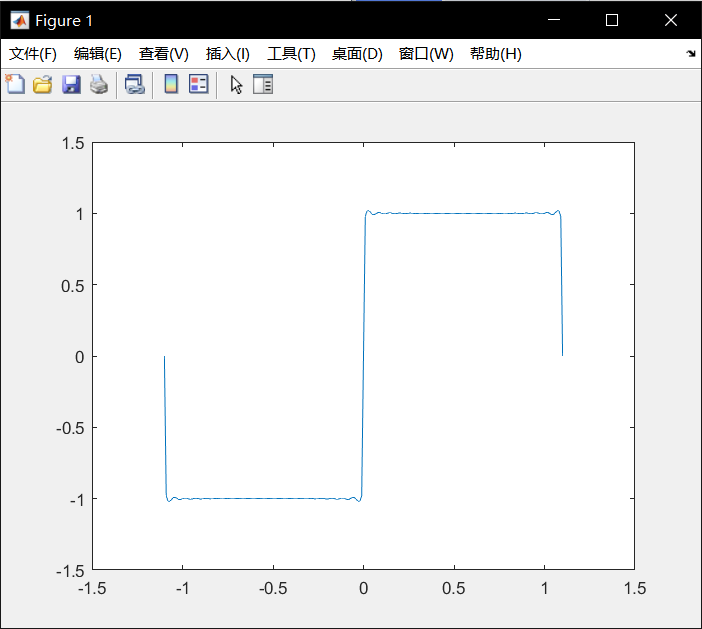
f(num)=(4./pi).\*sum((1./(2.\*k+1)).\*sin(((2.\*k+1).\*pi.\*t)./T));

end

t = -1.1:0.01:1.1;%value of t

plot(t,f);

hold on



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 Aerk 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?

%8.7

clear;

type compact;

%initialization

a=1000;

k=10;

r=0.04;

for n=1:1:10%n=1-10

v=a\*(1+r/n)^(n\*k);

disp(v);%disp

end

1.4802e+03

1.4859e+03

1.4879e+03

1.4889e+03

1.4895e+03

1.4898e+03

1.4901e+03

1.4903e+03

1.4905e+03

1.4906e+03

8.8

Write a program to compute the sum of the series 12 + 22 + 32 ... such that the sum is as large as possible without exceeding 1000. The program should display how many terms are used in the sum.

%8.8

clear;

type compact;

%initialization

sum=0;

for n=1:1:100%enough to use

sum=sum+n^2;

if sum>1000%whether more than 1000

disp(n-1);%disp

break

end

end

13

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.

%8.9

clear;

type compact;

for a = [1000,500,2000,10000]%initialization

r = 0.1;

bal = a;

year = 0;

disp( 'Year Balance' )

while bal <2\*a

bal = bal + r \* bal;%bal

year = year + 1;%sum

disp( [year bal] )%disp

end

end

Year Balance

1 1100

2 1210

3 1331

4 1464.1

5 1610.5

6 1771.6

7 1948.7

8 2143.6

Year Balance

1 550

2 605

3 665.5

4 732.05

5 805.25

6 885.78

7 974.36

8 1071.8

Year Balance

1 2200

2 2420

3 2662

4 2928.2

5 3221

6 3543.1

7 3897.4

8 4287.2

Year Balance

1 11000

2 12100

3 13310

4 14641

5 16105

6 17716

7 19487

8 21436

8.10

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

%8.10

%where is Exercise 3.2?

8.11

Use the Taylor series 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.

%8.11

clear;

type compact;

%initialization

cosvalue=1;

x=pi/4;

for n=2:1:100%enough x

plus=(x^(2\*n-2))/(factorial(2\*n-2));

%how many terms to get 4-figure

if plus>0.0001

cosvalue=cosvalue+((-1)^(n+1))\*plus;

else

disp(n);%disp

disp(cosvalue);%disp

break

end

end

5

0.7071

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.

%8.12

clear;

type compact;

%initialization

borrow=10000;

for n=1:1:1000%enough months

if borrow>5000

borrow=borrow\*1.02;%interest

else

borrow=borrow\*1.01;%interest

end

if borrow>300

borrow=borrow-300;%pay

show=['borrow',num2str(borrow)];

disp(show);%disp per month

else

disp('finish months');

disp(n);

break

end

end

borrow9900

borrow9798

borrow9693.96

borrow9587.8392

borrow9479.596

borrow9369.1879

borrow9256.5717

borrow9141.7031

borrow9024.5372

borrow8905.0279

borrow8783.1285

borrow8658.791

borrow8531.9668

borrow8402.6062

borrow8270.6583

borrow8136.0715

borrow7998.7929

borrow7858.7688

borrow7715.9441

borrow7570.263

borrow7421.6683

borrow7270.1016

borrow7115.5037

borrow6957.8138

borrow6796.97

borrow6632.9094

borrow6465.5676

borrow6294.879

borrow6120.7765

borrow5943.1921

borrow5762.0559

borrow5577.297

borrow5388.843

borrow5196.6198

borrow5000.5522

borrow4800.5633

borrow4548.5689

borrow4294.0546

borrow4036.9951

borrow3777.3651

borrow3515.1388

borrow3250.2901

borrow2982.793

borrow2712.621

borrow2439.7472

borrow2164.1447

borrow1885.7861

borrow1604.644

borrow1320.6904

borrow1033.8973

borrow744.2363

borrow451.6786

borrow156.1954

finish months

54

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.

%8.13

clear;

type compact;

%initialization

v=60;

angle=(50/180)\*pi;%angle

vx=60\*cos(angle);%speed of x

vy=60\*sin(angle);%speed of y

t=(vy\*2)/9.8;

d1=['time in the air : ',num2str(t)];

disp(d1);%disp time in the air

for t=0:0.5:t

d2=['speed of x : ',num2str(t),' ',num2str(vx)];

disp(d2);%disp time in the air

vy1=vy+(-t)\*9.8;

d3=['speed of y : ',num2str(t),' ',num2str(vy1)];

disp(d3);%disp time in the air

end

time in the air : 9.3801

speed of x : 0 38.5673

speed of y : 0 45.9627

speed of x : 0.5 38.5673

speed of y : 0.5 41.0627

speed of x : 1 38.5673

speed of y : 1 36.1627

speed of x : 1.5 38.5673

speed of y : 1.5 31.2627

speed of x : 2 38.5673

speed of y : 2 26.3627

speed of x : 2.5 38.5673

speed of y : 2.5 21.4627

speed of x : 3 38.5673

speed of y : 3 16.5627

speed of x : 3.5 38.5673

speed of y : 3.5 11.6627

speed of x : 4 38.5673

speed of y : 4 6.7627

speed of x : 4.5 38.5673

speed of y : 4.5 1.8627

speed of x : 5 38.5673

speed of y : 5 -3.0373

speed of x : 5.5 38.5673

speed of y : 5.5 -7.9373

speed of x : 6 38.5673

speed of y : 6 -12.8373

speed of x : 6.5 38.5673

speed of y : 6.5 -17.7373

speed of x : 7 38.5673

speed of y : 7 -22.6373

speed of x : 7.5 38.5673

speed of y : 7.5 -27.5373

speed of x : 8 38.5673

speed of y : 8 -32.4373

speed of x : 8.5 38.5673

speed of y : 8.5 -37.3373

speed of x : 9 38.5673

speed of y : 9 -42.2373

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.

%8.14

clear;

type compact;

%initialization

v=9;

r=4;

c=1;

q=0;

for t=0:0.1:100 %enough times

if q>=8

break

else

q=c\*v\*(1-exp(1)^(-(t)/(r\*c)));

dp=[num2str(t),' ',num2str(q)];

disp(dp);

end

end

0 0

0.1 0.22221

0.2 0.43894

0.3 0.65031

0.4 0.85646

0.5 1.0575

0.6 1.2536

0.7 1.4449

0.8 1.6314

0.9 1.8134

1 1.9908

1.1 2.1639

1.2 2.3326

1.3 2.4973

1.4 2.6578

1.5 2.8144

1.6 2.9671

1.7 3.1161

1.8 3.2613

1.9 3.403

2 3.5412

2.1 3.676

2.2 3.8075

2.3 3.9357

2.4 4.0607

2.5 4.1826

2.6 4.3016

2.7 4.4176

2.8 4.5307

2.9 4.6411

3 4.7487

3.1 4.8537

3.2 4.956

3.3 5.0559

3.4 5.1533

3.5 5.2482

3.6 5.3409

3.7 5.4312

3.8 5.5193

3.9 5.6053

4 5.6891

4.1 5.7708

4.2 5.8506

4.3 5.9283

4.4 6.0042

4.5 6.0781

4.6 6.1503

4.7 6.2206

4.8 6.2893

4.9 6.3562

5 6.4215

5.1 6.4851

5.2 6.5472

5.3 6.6078

5.4 6.6668

5.5 6.7244

5.6 6.7806

5.7 6.8354

5.8 6.8889

5.9 6.941

6 6.9918

6.1 7.0414

6.2 7.0898

6.3 7.1369

6.4 7.1829

6.5 7.2278

6.6 7.2716

6.7 7.3142

6.8 7.3558

6.9 7.3964

7 7.436

7.1 7.4746

7.2 7.5123

7.3 7.549

7.4 7.5849

7.5 7.6198

7.6 7.6539

7.7 7.6871

7.8 7.7195

7.9 7.7511

8 7.782

8.1 7.8121

8.2 7.8414

8.3 7.87

8.4 7.8979

8.5 7.9251

8.6 7.9516

8.7 7.9775

8.8 8.0028

clear;

type compact;

%initialization

v=9;

r=4;

c=1;

q=0;

for t=0:0.1:10 %enough times

if q>=8

q=c\*v\*(1-exp(1)^(-(t)/(r\*c)));%q

dp=[num2str(t),' ',num2str(q)];%disp

disp(dp);

else

q=c\*v\*(1-exp(1)^(-(t)/(r\*c)));%q

%not disp

end

end

8.9 8.0274

9 8.0514

9.1 8.0748

9.2 8.0977

9.3 8.1199

9.4 8.1417

9.5 8.1629

9.6 8.1835

9.7 8.2037

9.8 8.2234

9.9 8.2425

10 8.2612

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).

%8.15

clear;

type compact;

%initialization

n=1;

p = input('please input a number\n'); %input

r=rem(p,n);%remainder R when P is divided by N

while p~=0 & n<p^0.5

n=n+1;

r=rem(p,n);%remainder

if r==0

dp=[num2str(p),' is not prime'];

disp(dp);%disp

break

else

dp=[num2str(p),' is prime'];

disp(dp); %disp

break

end

end

please input a number

13

13 is prime

please input a number

14

14 is not prime

9.1

Draw a graph of the population of the USA from 1790 to 2000, using the (logistic) model P (t) where t is the date in years.

Actual data (in 1000s) for every decade from 1790 to 1950 are as follows:

3929, 5308, 7240, 9638, 12 866, 17 069, 23 192, 31 443, 38 558, 50 156, 62 948, 75 995, 91 972, 1 05 711, 1 22 775, 1 31 669, 1 50 697. Superimpose this data on the graph of P (t). Plot the data as discrete circles (i.e. do not join them with lines) as shown in Figure 9.16.

%9.1

clear;

type compact;

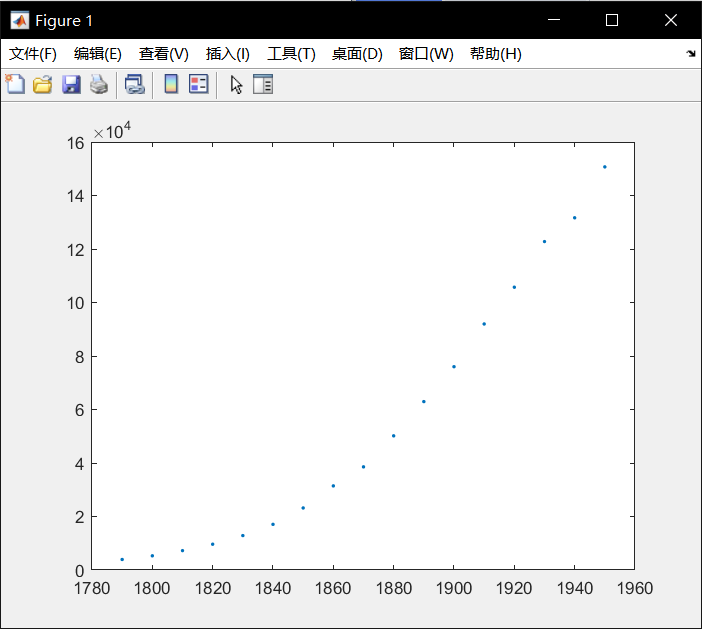
%initialization

p=[3929, 5308, 7240, 9638, 12866, 17069, 23192, 31443, 38558, 50156, 62948, 75995, 91972, 105711, 122775, 131669, 150697];

t=1790:10:1950;

%plot

plot(t,p,'.');



9.2

The Spiral of Archimedes (Figure 9.17) may be represented in polar coordinates by the equation

r = aθ,

where a is some constant. (The shells of a class of animals called nummulites grow in this way.) Write some command-line statements to draw the spiral for some values of a.

%9.2

clear;

type compact;

%initialization

a=1;

q=2;

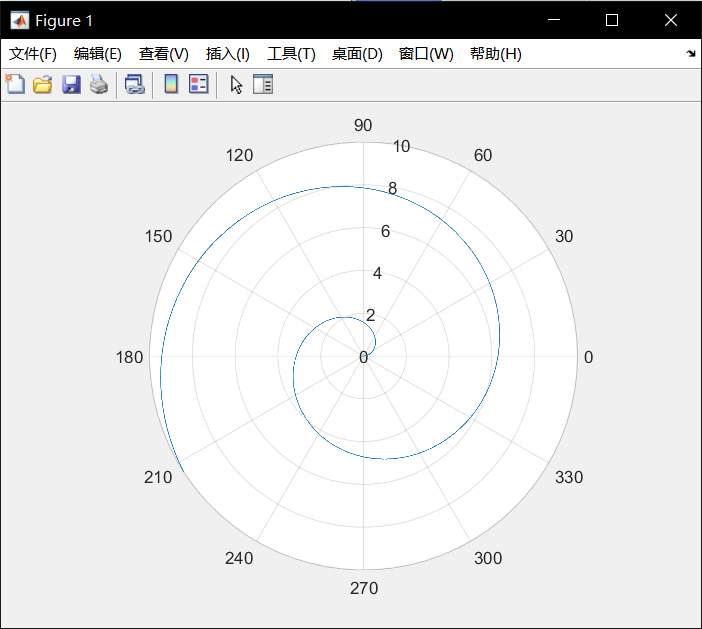
%compute

s=0:0.001:10;

r=a.\*(s);

%plot

polarplot(s,r);



9.3

Another type of spiral is the logarithmic spiral (Figure 9.17), which describes the growth of shells of animals like the periwinkle and the nautilus. Its equation in polar co-ordinates is

r = aqθ ,

where a > 0, q > 1. Draw this spiral.

%9.3

clear;

type compact;

%initialization

a=1;

q=2;

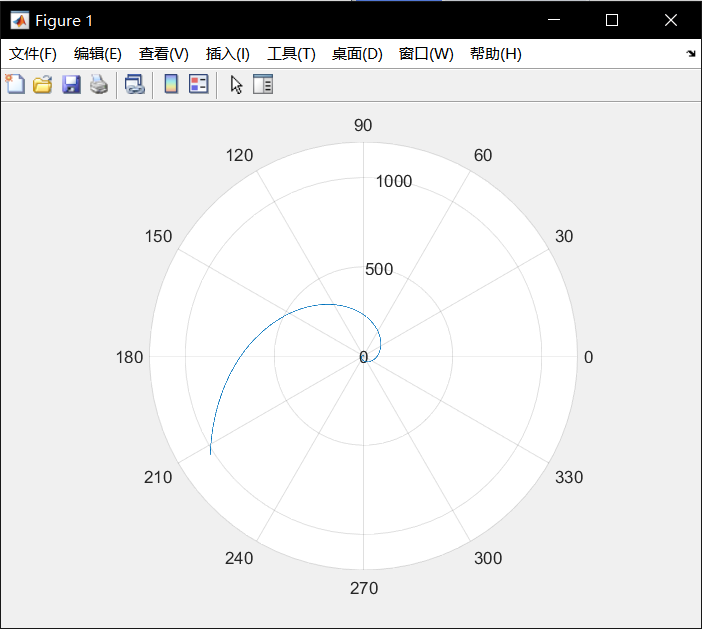
%compute

s=0:0.001:10;

r=a.\*(q.^s);

%plot

polarplot(s,r);



9.4

The arrangement of seeds in a sunflower head (and other flowers, like daisies) follows a fixed mathematical pattern. The nth seed is at position

r = √n,

with angular co-ordinate πdn/180 radians, where d is the constant angle of divergence (in degrees) between any two successive seeds, i.e. between the nth and (n + 1)th seeds. A perfect sunflower head (Figure 9.18) is generated by d = 137.51?. Write a program to plot the seeds; use a circle (o) for each seed. A remarkable feature of this model is that the angle d must be exact to get proper sunflowers. Experiment with some different values, e.g. 137.45 (spokes, from fairly far out), 137.65 (spokes all the way), 137.92 (Catherine wheels).

%9.4

clear;

type compact;

%initialization

n=0:1:10000;

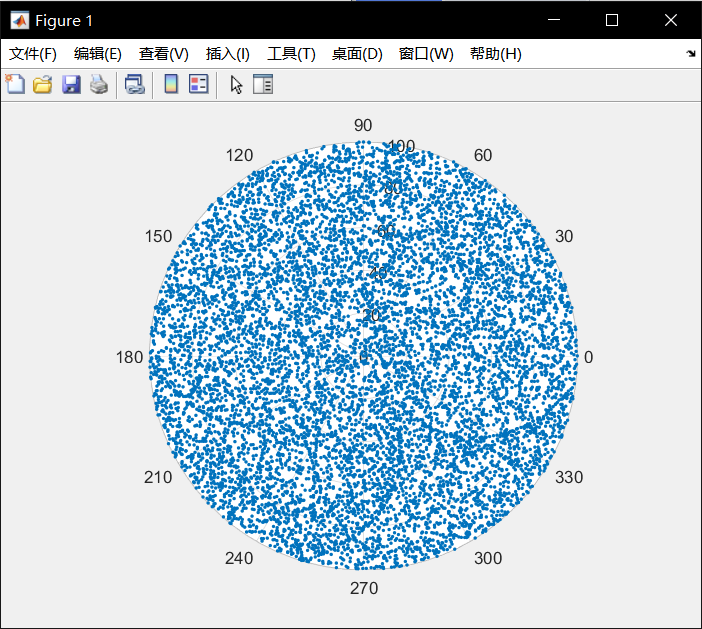
%compute

s=((137.51)/180).\*pi.\*(1+(n-1)).\*(n-1)./2;

r=(n).^0.5;

%plot

polarplot(s,r,'.');



clear;

type compact;

%initialization

n=0:1:30000;

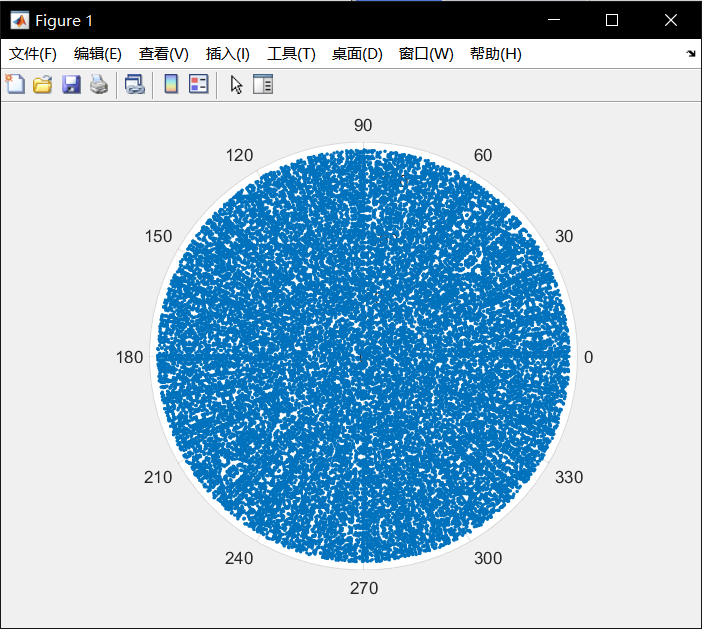
%compute

s=((137.45)/180).\*pi.\*(1+(n-1)).\*(n-1)./2;

r=(n).^0.5;

%plot

polarplot(s,r,'.');



clear;

type compact;

%initialization

n=0:1:30000;

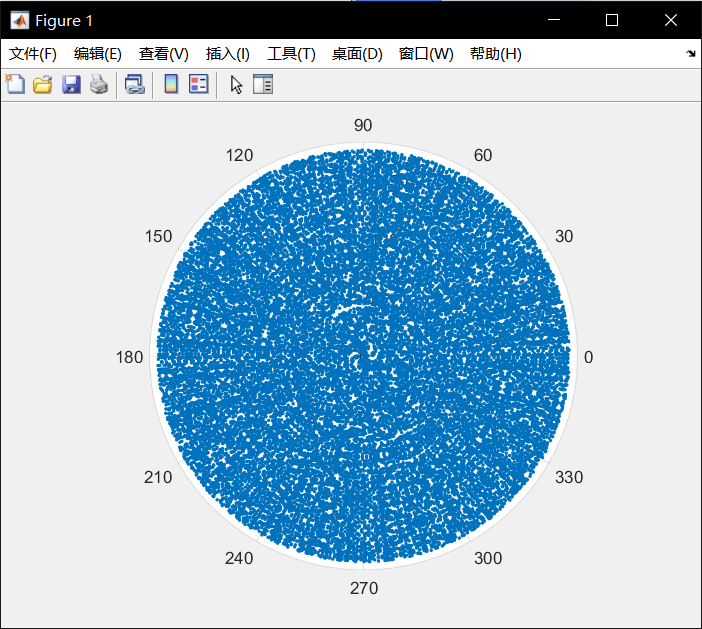
%compute

s=((137.65)/180).\*pi.\*(1+(n-1)).\*(n-1)./2;

r=(n).^0.5;

%plot

polarplot(s,r,'.');



clear;

type compact;

%initialization

n=0:1:30000;

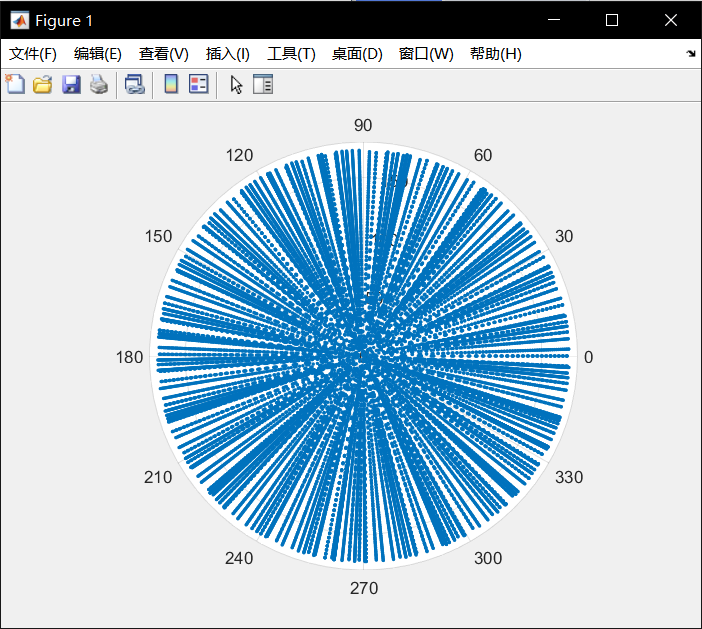
%compute

s=((137.92)/180).\*pi.\*(1+(n-1)).\*(n-1)./2;

r=(n).^0.5;

%plot

polarplot(s,r,'.');



9.5

The equation of an ellipse in polar co-ordinates is given by

r = a(1 - e2)/(1 - e cos θ),

where a is the semi-major axis and e is the eccentricity, if one focus is at the origin, and the semi-major axis lies on the x-axis.

Halley’s Comet, which visited us in 1985/6, moves in an elliptical orbit about the Sun (at one focus) with a semi-major axis of 17.9 A.U. (A.U. stands for Astronomical Unit, which is the mean distance of the Earth from the Sun: 149.6 million km.) The eccentricity of the orbit is 0.967276. Write a program which draws the orbit of Halley’s Comet and the Earth (assume the Earth is circular).

%9.5

clear;

type compact;

%initialization

a=17.9;

e=0.967276;

%compute

s1=0:0.001:10;

r1=(a.\*(1-e.^2))./(1-e.\*cos(s1));

s2=0:0.001:10;

r2=ones(1,10001);

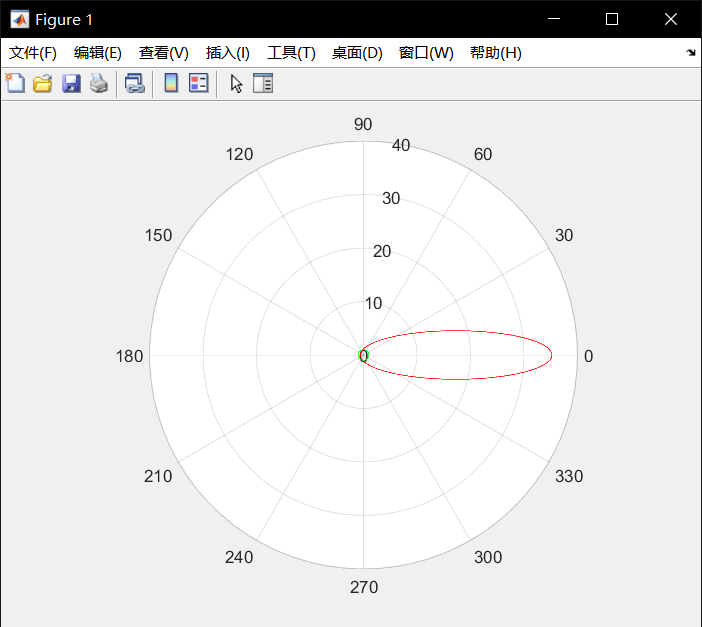
%plot

polarplot(s1,r1,'r');

hold on

polarplot(s2,r2,'g');

hold off



9.6

A very interesting iterative relationship that has been studied a lot recently is defined by

yk+1 = ryk(1 - yk)

(this is a discrete form of the well-known logistic model). Given y0 and r, successive yk s may be computed very easily, e.g. if y0 = 0.2 and r = 1, then y1 = 0.16, y2 = 0.1334, and so on.

This formula is often used to model population growth in cases where the growth is not unlimited, but is restricted by shortage of food, living area, etc.

yk exhibits fascinating behavior, known as mathematical chaos, for values of r between 3 and 4 (independent of y0). Write a program which plots yk against k (as individual points).

Values of r that give particularly interesting graphs are 3.3, 3.5, 3.5668, 3.575, 3.5766, 3.738, 3.8287, and many more that can be found by patient exploration.

%9.6

clear;

type compact;

%initialization

y(1)=0.2;

r=1;

dpk=1:1:11;

%compute

for k=2:1:11

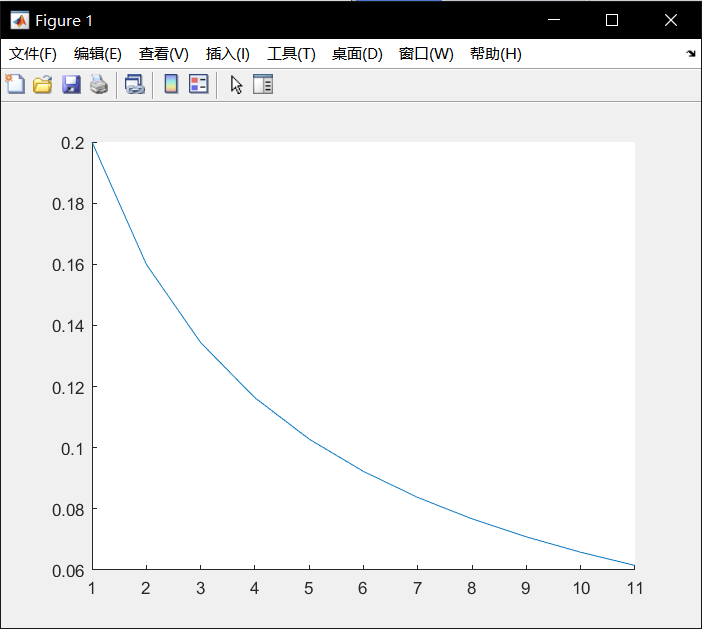
y(k)=r\*y(k-1)\*(1-y(k-1));

end

%plot

hold on;

plot(dpk,y);



clear;

type compact;

% %initialization

% y(1)=0.2;

% r=1;

% dpk=1:1:11;

% %compute

% for k=2:1:11

% y(k)=r\*y(k-1)\*(1-y(k-1));

% end

% %plot

% hold on;

% plot(dpk,y);

%initialization

y(1)=0.2;

r=1;

dpk=1:1:11;

%compute

for k=2:1:11

y(k)=r\*y(k-1)\*(1-y(k-1));

end

%plot

hold on;

plot(dpk,y);

%initialization

y(1)=0.2;

dpk=1:1:101;

for r=[3.3, 3.5, 3.5668, 3.575, 3.5766, 3.738, 3.8287]

%compute

for k=2:1:101

y(k)=r\*y(k-1)\*(1-y(k-1));

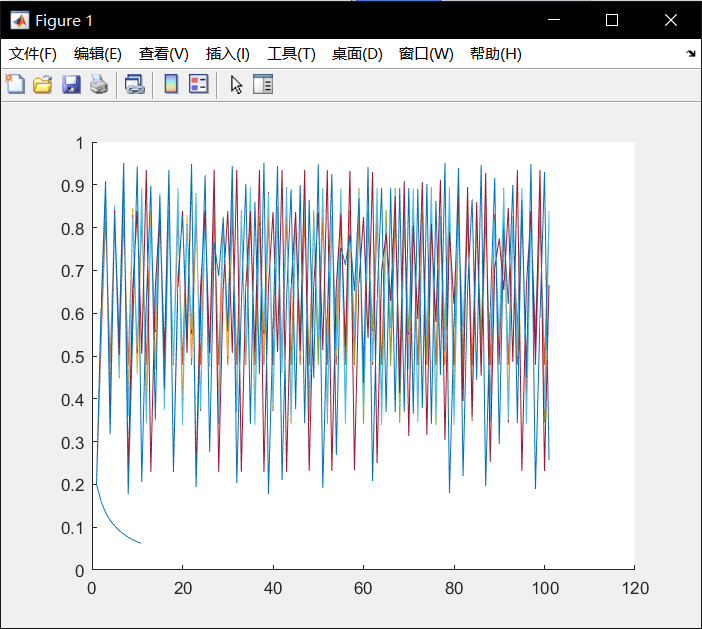
end

%plot

hold on;

plot(dpk,y);

end



9.7

A rather beautiful fractal picture can be drawn by plotting the points (xk , yk ) generated by the following difference equations \_ starting with x0 = y0 = 0. Write a program to draw the picture (plot individual points; do not join them)

%9.7

clear;

type compact;

%initialization

x(1)=0;

y(1)=0;

%compute

for k=1:1:100000

%x(k+1)

x(k+1)=y(k)\*(1+sin(0.7\*x(k)))-1.2\*((abs(x(k)))^0.5);

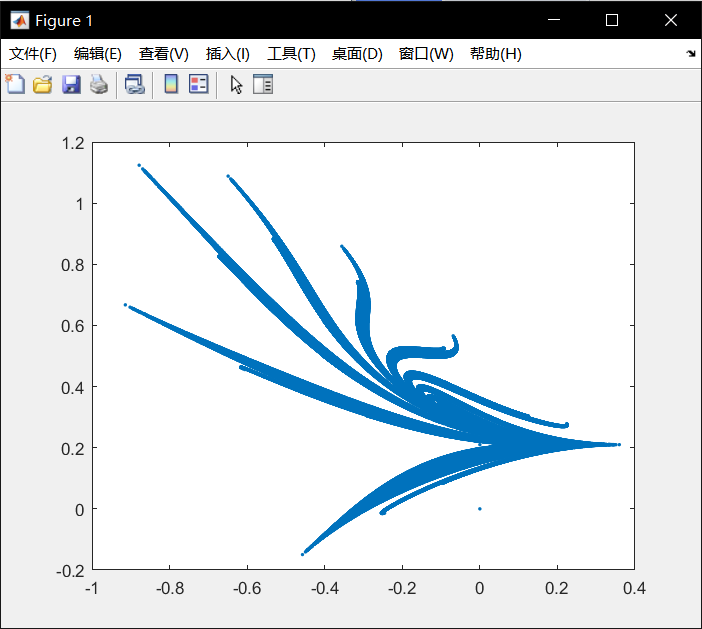
%y(k+1)

y(k+1)=0.21-x(k);

end

%plot

plot(x,y,'.');



11.1

The Newton quotient \_ may be used to estimate the first derivative f 1(x) of a function f (x), if h is ‘small’. Write a program to compute the Newton quotient for the function f (x) = x2 at the point x = 2 (the exact answer is 4) for values of h starting at 1, and decreasing by a factor of 10 each time (use a for loop). The effect of rounding error becomes apparent when h gets ‘too small’, i.e., less than about 10e-12.

%11.1

clear;

type compact;

%initialization

h=1;

x=2;

dpn=1:1:20;

%compute

for n=1:1:20

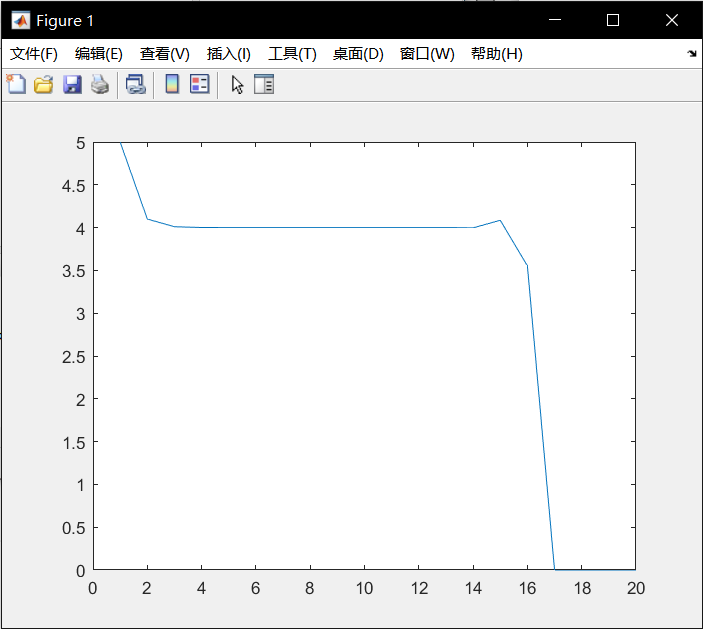
back(n)=((x+h)^2-x^2)/(h);

h=h/10;%smaller

end

%disp

plot(dpn,back)



11.2

The solution of the set of simultaneous equations

ax + by = c

dx + ey = f

(Exercise 3.6) is given by

x = (ce - bf )/(ae - bd),

y = (af - cd)/(ae - bd).

If (ae - bd) is small, rounding error may cause quite large inaccuracies in the solution. Consider the system

0.2038x + 0.1218y = 0.2014,

0.4071x + 0.2436y = 0.4038.

Show that with four-figure floating point arithmetic the solution obtained is x = ?1, y = 3. This level of accuracy may be simulated in the solution of Exercise 3.6 with some statements like

ae = floor( a \* e \* 1e4 ) / 1e4

and appropriate changes in the coding. The exact solution, obtained without rounding, is x = ?2, y = 5. If the coefficients in the equations are themselves subject to experimental error, the ‘solution’ of this system using limited accuracy is totally meaningless.

%11.2

clear;

type compact;

%initialization

a=0.2038;

b=0.1218;

c=0.2014;

d=0.4071;

e=0.2436;

f=0.4038;

%compute

x = (c\*e - b\*f )/(a\*e - b\*d);

y = (a\*f - c\*d)/(a\*e - b\*d);

%disp

disp(x);

disp(y);

-2

5

17.1

Find the derivatives of cos x, and x2 exp2x

%17.1

clear;

type compact;

syms x;

diff(x^2);%compute

disp(ans);%disp

diff(x^2\*exp(2\*x));%compute

disp(ans);%disp

2\*x

2\*x\*exp(2\*x) + 2\*x^2\*exp(2\*x)

17.2

Integrate the function y = 3x2 from x = 0 to x = 10.

%17.2

clear;

type compact;

syms x;

dp = int(3\*(x^2),x,0,10);%compute

disp(dp);%disp

1000