MATLAB FOR MATH 303

ABSTRACT. This is a short notes on Matlab for Math 303. The original source for the exposition and examples is the old notes by Prof. Cai. I have included more examples and updated a few new commands since some of commands in Cai's notes are obsolete.

CONTENTS

1. Getting started with MATLAB

The goal of this quick and dirty approach is to make you get familiarized with MATLAB only for MATH 303, and so you need to consult other documents or reference books for more information. A recommended reference is *ODEs using MATLAB*, Polking and Arnold, 3rd Edition. There are free materials such as the manual provided by Matlab or the documents you can find on the web.

1.1. A quick start.

(1) Open up MATLAB from your computer (In general, it depends on your system how to open; in a windows system just click the icon).
(2) You will see the *command window.*
(3) Type "2+3" and hit the enter key in the command window and see what MATLAB does. You should see the following on your command window:

```
>> 2+3

ans =

     5
>>
```

So far so good. Now you are in the game.
(4) For any serious work, you need to know more of Matlab. You may get helped by Matlab. Click Help and find documents for "getting started" etc.

It is time to move on to the next stage.

1.2. Arithmetic in Matlab.

Now that we know how to add two numbers in Matlab, we may play with more arithmetic computation. One thing is Matlab may not understand human mathematical languages. For example, Matlab does not know how to read 2^3 but knows 2ˆ3. The following table is a short dictionary for you and Matlab.

	What you understand	What Matlab understands
addition	2+3	2+3
subtraction	$2-3$	$2-3$
multiplication	$2\cdot 3$	2*3
division	$4 \div 2 = 4/2$	4/2
power	2^3	2ˆ3
pi	$\pi = 3.1415...$	pi

Example 1.1. Use Matlab to compute

$$3+\frac{(4\cdot 3)^{20}}{2^2}.$$

You need to type (and hit the enter key):

```
>> 3+(4*3)^(20)/2^2
```

Then Matlab will show you

```
>> 3+(4*3)^(20)/2^2

ans =

     9.5844e+20
>>
```

which means

$$3+\frac{(4\cdot 3)^{20}}{2^2}\approx 9.5844\times 10^{20}.$$

1.3. **M-files.** Matlab knows some functions such as sine and cosine functions. But you need to deal with functions which Matlab does not know. There is a way of creating new functions and you can teach (baby) Matlab. Eventually, (educated) Matlab will understand your new functions. I will show you how to create so-called m-files by an example.

Example 1.2. Let Matlab know the following function by heart

$$g(x)=x^2.$$

Write the following code in any text-editor such as notepad in windows system or the editor in Matlab but **not** in the command window.

```
function y=g(x)
g(x)=x.^2;
```

Then save the file as ”g.m” in the current directory (working directory). Now Matlab understands your new function $g(x)=x^2$. You may test it like

```
>> g(2)
ans =

     4
>> g(3)
ans =

     9
```

Warning: Matlab is very sensitive to dots, semicolons, and colons. You must not omit ”.” and ”;” in your ”g.m” file.

1.4. **Plotting the graph of a function.** I will explain two ways of plotting the graphs. The first one is "plot"

Example 1.3. We want to plot the graph of $y = \sin x$ in the interval $[-1, 4]$ with length $h = .1$

```
>> x=-1:.1:4;
>> plot(x,sin(x))
>> title('y=sin(x)')
```

Matlab will show you the graph of sine on $[-1, 4]$. The line by line explanation is

(1) The first line determines the (row) vector
$$x = [-1, -0.9, -0.8, ..., 3.8, 3.9, 4].$$
(2) Matlab computes
$$\begin{aligned} x &= [-1, -0.9, -0.8, ..., 3.8, 3.9, 4] \\ \sin x &= [\sin(-1), \sin(-0.9), \sin(-0.8), ..., \sin(3.8), \sin(3.9), \sin(4)]. \end{aligned}$$
and generates the graph of $\sin x$.
(3) The third line names your graph as $y = \sin(x)$.

A second way to graph is with the command "fplot".

```
>> fplot('sin(x)',[-1,4])
```

Pros and cons, and warnings:

(1) You don't decide the length but Matlab does.
(2) "fplot" only recognizes "x" as the independent variable. So type
```
>> fplot('sin(x)',[-1,4])
```
not
```
>> fplot('sin(t)',[-1,4])
```

Now you are ready for Matlab assignments. Of course, we need to learn more of Matlab. This notes will be updated as we go along.

2. Matrix and linear algebra

2.1. **Matrix computation.** We want to let Matlab know the matrix
$$A = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}.$$
For this, type

```
>> A=[1,2;3,4]
```

A row vector is noting but $1 \times n$ vector. To input the vector
$$\mathbf{x} = \begin{bmatrix} 5 & 6 \end{bmatrix},$$
type

```
>> x=[5,6]
```

To compute $A\mathbf{x}$ type

```
>> A*x
```

Matlab will complain about the illegal operation! For $A\mathbf{x}$ to be defined, $\mathbf{x}$ should be a 2×1 column vector not a 1×2 row vector. To fix this problem, we need to take the transpose. We need to type

```
>> y=[5,6]'
```

where **'** means the transpose of the row vector $[5, 6]$. Then "y" is the 2×1 column vector

$$\mathbf{y} = \begin{bmatrix} 5 \\ 6 \end{bmatrix}.$$

Now we compute $A\mathbf{y}$ using Matlab

```
>> A*y
```

Now Matlab should smile at you and give the desired result.

Example 2.1. Let

$$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} 1 \\ 5 \\ 7 \end{pmatrix}.$$

Use Matlab to compute AB and $B\mathbf{x}$.

You may type like

```
>> A=[1,2,3;4,5,6;7,8,9];
>> B=[1,0,1;0,1,0;1,0,1];
>> x=[1,5,7]'
```

Now Matlab knows the matrices A, B and the column vector $\mathbf{x}$. To get AB, type

```
>> A * B
```

For $B\mathbf{x}$, type

```
>> B * x
```

2.2. **Eigenvalues and eigenvectors.** Given an $n \times n$ matrix A, a scalar λ is called an eigenvalue for A if there exists a nonzero vector $\mathbf{x}$ for which

$$(*) \quad A\mathbf{x} = \lambda\mathbf{x}.$$

Any nonzero vector satisfying $(*)$ is an eigenvector corresponding to λ.

Example 2.2. Let

$$A = \begin{pmatrix} 1 & 2 \\ 0 & 4 \end{pmatrix}.$$

We know A has eigenvalues $\lambda = 1, 4$ and corresponding eigenvectors:

$$\lambda_1 = 1 \quad \Longrightarrow \quad \mathbf{x}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$\lambda_2 = 4 \quad \Longrightarrow \quad \mathbf{x}_2 = \begin{pmatrix} 1 \\ 1.5 \end{pmatrix}$$

Type:

```
>> A=[1,2;0,4];
>> [E,V]=eig(A)
```

We get

```
E =
        1.0000             0.5547
        0                  0.8321



V =
        1                   0
        0                   4
```

Note that

$$0.5547 \times 1.5 \approx 0.8321.$$

Matlab shows us

$$E = \begin{pmatrix} \mathbf{x}_1 & \mathbf{x}_2 \end{pmatrix}, \quad V = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

That is, V is the diagonal matrix consisting of eigenvalues of A, E is the matrix whose column vectors are correponding eigenvectors of A.

Example 2.3. Let

$$A = \begin{pmatrix} -1 & 0 & 2 \\ 2 & 3 & 6 \\ -2 & 0 & -1 \end{pmatrix}$$

Type:

```
>> A=[-1,0,2;2,3,6;-2,0,-1];
>> [E,V]=eig(A)
```

We get

```
E =
        0           0.5000                   0.5000
      1.0000      0.1000- 0.7000i    0.1000+ 0.7000i
        0           0+ 0.5000i               0- 0.5000i



V =
   3.0000                    0                        0
        0            -1.0000+ 2.0000i                  0
        0                    0                  -1.0000- 2.0000i
```

Here i denotes the complex number $i = \sqrt{-1}$.

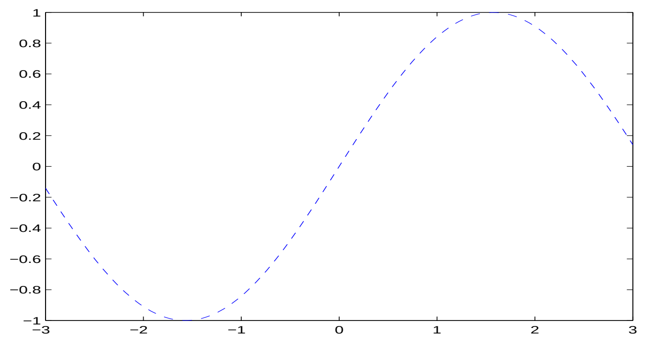

FIGURE 3.1. The dotted graph of $y = \sin x$

3. MORE ON GRAPHS

3.1. **Graph in different styles.** Recall that "fplot" command generates the usual graph of a function, for example,

```
>> fplot('sin(x)',[-3,3])
```

You can get the dotted graph of $y = \sin x$ by adding $' - -'$ at the end (See figure 3.1):

```
>> fplot('sin(x)',[-3,3],'--')
```

Similarly, you can play with the following:

```
>> fplot('sin(x)',[-3,3],'o')
```

There are more styles such as '*' or '.'.

3.2. **Two graphs on the same figure.** Sometimes we want to sketch the graphs in the same coordinate frame. For this, we need to know "hold on" command.

Example 3.1. Plot the graphs of $y = \sin x$ and $y = \cos x$ over $[-3, 3]$ on the same figure (See Figure 3.2).

```
>> fplot('sin(x)',[-3,3])
>> hold on
>> fplot('cos(x)',[-3,3])
```

To get out the "hold on" state and plot a new graph in another figure, use the command "hold off".

Example 3.2. Using different curve styles for each curve, plot the graphs of $y = \sin x, y = \cos x$ and $y = x^2$ over $[-3, 3]$ on the same figure.

```
>> fplot('sin(x)',[-3,3])
>> hold on
```

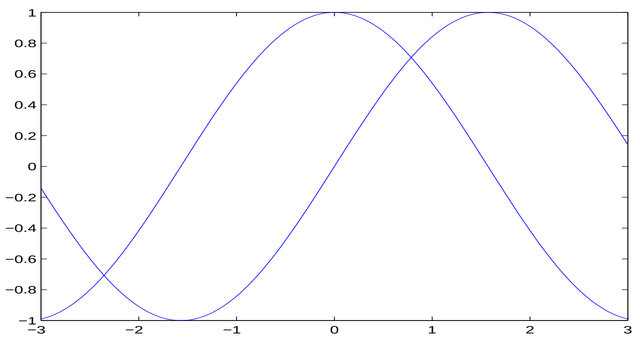

FIGURE 3.2. Sine and cosine together

```
>> fplot('cos(x)',[-3,3],'--')
>> fplot('x^2',[-3,3],'.')
```

3.3. **Grid, labeling and rescaling.** To put a grid on your graph, use the command "grid".

```
>> fplot('sin(x)',[-3,3])
>> grid
```

If you want to put a text in a figure, you may use "gtext". For example, type

```
>> fplot('sin(x)',[-3,3])
>> gtext(' The graph of y=sin x')
```

and click where you want to put the text in the figure.

Sometimes you may want to control your coordinate frame. For example,

```
>> fplot('x^3',[-10,10])
```

shows the graph of $y = x^3$ in the rectangle

$$-10 \leq x \leq 10, \quad -1000 \leq y \leq 1000.$$

Well... the graph looks too big or ugly. You may rescale the graph by putting the graph in a smaller rectangle, say

$$-5 \leq x \leq 7, \quad -20 \leq y \leq 30.$$

The command "axis" can be used for this (See Figure 3.3).

```
>> fplot('x^3',[-10,10])
>> axis([-5,7,-20,30]
```

3.4. **Summary.** The following example uses all the commands we learned in this section.

Example 3.3. Using different curve styles for curves, plot the graph of $y = \sin x$ and $y = \cos x$ over $[-5, 5]$ in the same figure titled "sine and cosine graph". Label

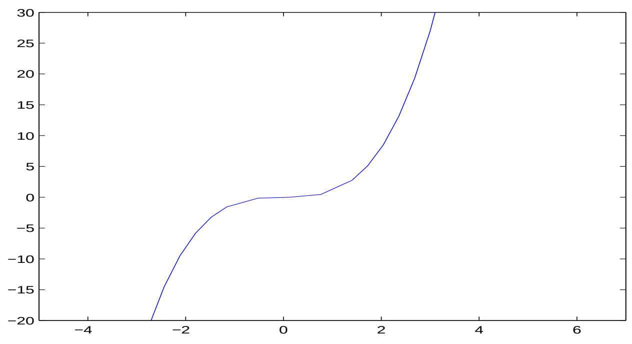

FIGURE 3.3. Rescaling

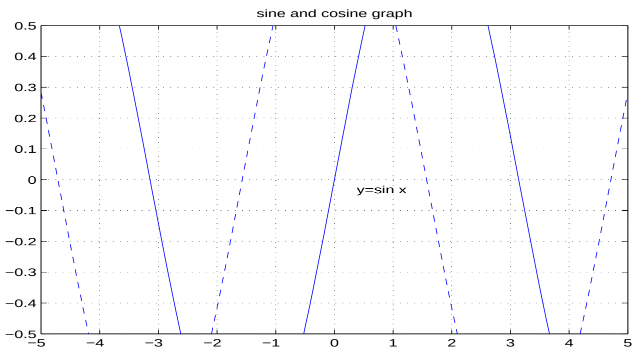

FIGURE 3.4. An ugly graph

the graphs as $y = \sin x$ and $y = \cos x$, and also put a grid. Rescale so that the graph is in the rectangle

$$-5 \leq x \leq 5, \quad -0.5 \leq y \leq 0.5.$$

```
>> fplot('sin(x)',[-5,5])
>> title('sine and cosine graph')
>> hold on
>> fplot('cos(x)',[-5,5],'--')
>> grid
>> gtext('y=sin x')
>> gtext('y=cos x')
>> axis([-5,5,-0.5,0,5])
```

The figure looks ugly for we have cut the tops and valleys of sine and cosine graphs. Oops I forgot to label $y = \cos x$ (See Figure 3.4).

4. More on functions

4.1. **Functions of two variables.** Matlab knows functions of two variables, of course. Consider

$$f(x,y) = 2xy^2 + \sqrt{x} + e^{2y}.$$

Let us write an M-file for $f(x,y)$:

```
function w=f(x,y)
w=2*x*y^2+x^(1/2)+exp(2*y);
```

We can use functions of two variables to handle a family of curves as follows.

Example 4.1. Consider the function

$$y(x) = x^2 + k$$

where k is a parameter. We want to plot $y(x)$ for $k = 1$ and $k = 2$ in the same frame.

One way is to write M-files for $y = x^2 + 1$ (for $k = 1$) and for $y = x^2 + 2$ (for $k = 2$). This is cumbersome especially when you want to plot for many different values of k. Let us be lazy and look at y as a function of two variables $y(x,k) = x^2 + k$.

```
function w=yk(x,k)
w=x^2+k;
```

Now we plot $y(x)$ for $k = 1$ and $k = 2$.

```
>> fplot('yk(x,1)',[-5,5])
>> hold on
>> fplot('yk(x,2)',[-5,5],'--')
```

4.2. **Maximum and minimum from the graphs.** We may find approximations of (local) maximum or minimum by looking at the graph of a function $y = f(x)$.

Example 4.2. Let

$$f(x) = x + \frac{3}{x} + 1, \quad 1 \le x \le 5$$

Let (x_0, y_0) be the point where $f(x)$ attains its minimum at $x = x_0$. Find the point (approximation correct up to two decimal places).

Write an M-file for $f(x)$

```
function w=f(x)
w=x+3/x+1
```

Then we plot the graph on $[1,5]$.

```
>> fplot('f(x)',[1,5])
>> grid
```

By looking at the graph (See Figure 4.1), you see that

$$1.5 < x_0 < 2, \quad 4 < y_0 < 4.5.$$

This is not enough! Use the magnifying glass button (the zoom-in button) on the figure and then look at the ”magnified” graph (See Figure 4.2). You see $x_0 = 1.7 \times \times\times$. We are still not sure. Now we stop looking at the graph and compute

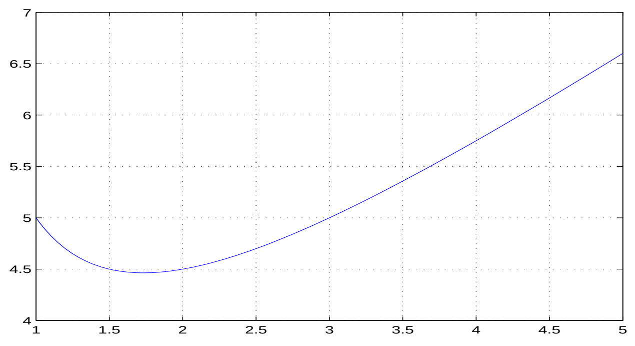

FIGURE 4.1. The graph of $f(x) = x + \frac{3}{x} + 1$

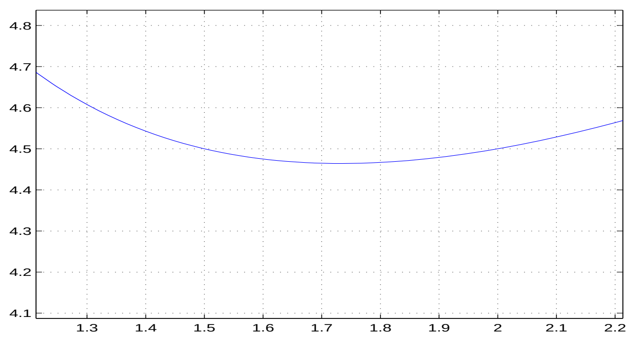

FIGURE 4.2. The magnified graph of $f(x) = x + \frac{3}{x} + 1$

$f(1.71), f(1.72)$ etc instead.

```
>> f(1.72)

ans=
   4.4642

>> f(1.73)

ans=
   4.4641

>> f(1.74)

ans=
   4.4641

>> f(1.75)
```

```
ans=
   4.4643
```

and we see that $x_0 \approx 0.73$ and $y_0 \approx 4.46$. In fact,

$$x_0 = \sqrt{3} \approx 1.732050808, \quad y_0 = 2\sqrt{3} + 1 \approx 4.464101615$$

and so our dirty method is not too bad. There is a better way of finding minimum or maximum. We may come back to this issue later.

5. Laplace transform

We study Laplace transforms using Matlab.

5.1. **"If-else" in Matlab and the step function.**

The step function $u_c(t)$ is defined by

$$u_c(x) = \begin{cases} 0 & \text{if } x < c \\ 1 & \text{if } x \geq c \end{cases}$$

Let $u(x) = u_0(x)$. Then $u_c(x) = u(x - c)$.

To write an M-file for $u(x)$, we need to use "if-else":

```
function w=u(x)
if x<0
 w=0;
else
 w=1;
end
```

Note that $u_5(x) = u(x - 5)$. To write an M-file for $u_5(x)$ we may write:

```
function w=u5(x)
w=u(x-5);
```

This example shows that an M-file (function) can call another M-file.

Example 5.1. Write an M-file for

$$g(x) = \begin{cases} 0 & \text{if } x < 5, \\ (x-5)/5 & \text{if } 5 \leq x < 10, \\ 1 & \text{if } x \geq 10. \end{cases}$$

We may use "if-elseif-else":

```
function w=g(x)
if x<5
 w=0;
elseif  x<10
 w=(x-5)/5;
else
 w=1;
end
```

5.2. **An ODE example.** Consider the initial value problem

$$y''(x) + 4y(x) = g(x), \quad y(0) = 0, \quad y'(0) = 0$$

where

$$g(x) = \begin{cases} 0 & \text{if } x < 5, \\ (x-5)/5 & \text{if } 5 \le x < 10, \\ 1 & \text{if } x \ge 10. \end{cases}$$

Using Laplace transforms we can solve the problem

$$y(x) = [u_5(x)h(x-5) - u_{10}(x)h(x-10)]/5, \quad h(x) = \frac{1}{4}x - \frac{1}{8}\sin 2x.$$

We shall write a few M-files for this problem. First, we write $u(x)$

```
function w=u(x)
if x<0
 w=0;
else
 w=1;
end
```

and then $h(x)$

```
function w=h(x)
w=(1/4)*x-(1/8)*sin(2*x);
```

The solution $y(x)$ is

```
function w=y(x)
w= (1/5)*(u(x-5)*h(x-5)-u(x-10)*h(x-10));
```

Now go the command window and plot the graph of $y(x)$

```
>> fplot('y(x)', [0,20])
```

Compare your graph with the graph (Figure 6.4.3, page 336) in the textbook.

6. More on functions; Maximum and minimum and roots

So far we have used the graphs of functions to get information on functions. Now we consider a few more Matlab commands on functions.

6.1. **Minimum and maximum of function.** To find the minimum of a function $f(x)$ on a closed interval $[a, b]$ we may use `fminbnd`. For example, consider the function (see Figure 6.1)[1]

$$f(x) = x^2 + 1, \quad -2 \le x \le 2.$$

We see that $f(x) = x^2 + 1$ has the minimum at $x = 0$. Let us use Matlab to find the minimum: Write an m-file for $f(x) = x^2 + 1$ or type[2]

```
>> f=inline('x^2+1');
```

To find the point x_m where the minimum occurs, use

[1]The graph is not generated by Matlab. It is created by Maple, a friend of Matlab.

[2]This is another way to define a function in the command window.

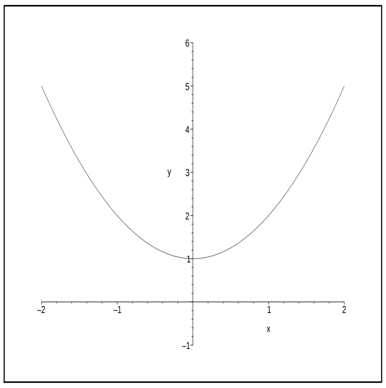

FIGURE 6.1. The graph of $f(x) = x^2 + 1$

```
>> xm=fminbnd('f(x)',-2,2)
```

This gives us $x_m = 0$. To get the minimum value $f(x_m)$, type

```
>> f(xm)
```

To find the maximum of $f(x)$, you can do

```
>> xM=fminbnd('-f(x)',-2,2)
>> f(xM)
```

because $f(x)$ maximizes where $-f(x)$ minimizes.

6.2. $f(x) = 0$**.** Next consider the problem of finding x for which

$$f(x) = 0.$$

This is not an easy problem in general. If you know a root of $f(x) = 0$ is near $x = a$ you may use `fzero('f(x)',a)`. Matlab will try to find the root near $x = a$. Sometimes Matlab gives up searching and will complain by throwing out error messages. Then you may try with different values of a again until either Matlab or you are knocked out.

For this type of searching, you first plot the graph and then use the graph information to guess where the root should be around.

Example 6.1. Let $g(x) = x^2 - 1$. Find the roots of $f(x) = 0$ using Matlab.

Let Matlab know the function $g(x)$ by writing an m-file for $g(x)$ or by typing it on the command window.

```
function w=g(x)
w=x^2-1;
```

Then plot the graph

```
>> fplot('g',[-3,3])
```

We see that there are two roots. One is near $x = -1$ and the other is near $x = 1$. In fact, $x = -1, 1$ are the roots as you know. Let us pretend we do not know the answer. Try

```
>> fzero('g(x)',-1.1)
```

or

```
>> fzero('g(x)',-0.8)
```

You will find $x = -1$. Do the same for $x = 1$.

6.3. **A realistic example.** Consider the problem: For

$$f(x) = \frac{1}{x}\sin x,$$

find the smallest x_0 after which $|f(x)| < 0.01$ for all $x > x_0$. First, write an m-file for $f(x)$.

```
function w=f(x)
w=(1/x)*sin(x)
```

Then plot the graph, say on $[0.1, 120]$.

```
>> fplot('f(x)', [0.1,120])
```

By looking at the graph (zoom-in if necessary) we now have a rough idea that x_0 is somewhere $x = 98.1....$ Now we use `fzero` with $a = 98.1$.

```
>> fzero('f(x)', 98.1)
```

Therefore, $x_0 \approx 98.8054$. Note that there are many roots of $f(x) = 0.01$ or of $f(x) = -0.01$ and so `fzero` without aid of the graph is not so efficient.

7. Systems of first order linear equations

7.1. **Systems of linear equations.** Recall that elementary row operations do not change the solutions of a linear system $A\mathbf{x} = \mathbf{b}$. For example, consider the system

$$\begin{cases} x + 2y = 5 \\ 3x + 4y = 6 \end{cases} \iff [A|b] = \begin{bmatrix} 1 & 2 & 5 \\ 3 & 4 & 6 \end{bmatrix}.$$

The reduced row echelon form of $[A|b]$ (`rref`) is

$$\begin{bmatrix} 1 & 0 & -4 \\ 0 & 1 & 4.5 \end{bmatrix} \iff \begin{cases} x + 0 \cdot y = -4 \\ 0 \cdot x + y = 4.5 \end{cases} \iff \begin{cases} x = -4 \\ y = 4.5 \end{cases}.$$

This can be done easily in Matlab.

```
>> B=[1,2,5;3,4,6];
>> rref(B)
```

From the `rref(B)` you can read off the solution quickly.

7.2. **Fundamental sets.** Consider

$$\mathbf{x}' = A\mathbf{x}$$

where A is an $n \times n$ matrix. Assume that A has n-linearly independent eigenvectors $\mathbf{v}_1, ..., \mathbf{v}_n$ corresponding to the eigenvalues $\lambda_1, ..., \lambda_n$. We may obtain the fundamental set $\mathbf{x}_1, ..., \mathbf{x}_n$ by setting

$$\mathbf{x}_i = e^{\lambda_i t}\mathbf{v}_i.$$

When we want to real solutions even if we have complex eigenvalues λ_i, we need to work a little bit more.

Example 7.1. Find a real fundamental set for

$$\mathbf{x}' = A\mathbf{x}, \quad A = \begin{bmatrix} -1 & 0 & 2 \\ 2 & 3 & 6 \\ -2 & 0 & -1 \end{bmatrix}.$$

First find the eigenvalues and eigenvectors (see Section 2):
Type

```
>> A=[-1,0,2;2,3,6;-2,0,-1];
>> [E,V]=eig(A)
```

We get

```
E =
        0          0.5000                  0.5000
   1.0000     0.1000- 0.7000i    0.1000+ 0.7000i
        0        0+ 0.5000i            0- 0.5000i


V =
   3.0000                  0                       0
        0       -1.0000+ 2.0000i                   0
        0                  0           -1.0000- 2.0000i
```

This means that the eigenvalues are

$$\lambda_1 = 3, \quad \lambda_2 = -1 + 2i, \quad \lambda_3 = -1 - 2i$$

and corresponding eigenvectors are

$$\mathbf{v}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 0.5 \\ 0.1 - 0.7i \\ 0.5i \end{bmatrix}, \quad \mathbf{v}_3 = \begin{bmatrix} 0.5 \\ 0.1 + 0.7i \\ -0.5i \end{bmatrix}.$$

From this, we find a real fundamental set

$$\mathbf{x}_1 = e^{3t}\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{x}_2 = e^{-t}\begin{bmatrix} 0.5\cos(2t) \\ 0.1\cos(2t) + 0.7\sin(2t) \\ -0.5\sin(2t) \end{bmatrix}, \quad \mathbf{x}_3 = e^{-t}\begin{bmatrix} 0.5\sin(2t) \\ 0.1\sin(2t) - 0.7\cos(2t) \\ 0.5\cos(2t) \end{bmatrix}.$$

8. Numerical methods I

8.1. **Miscellaneous topics; Input and output.** The command `input('sentence')` calls for the data. For the output, one may use `fprintf`. For example,

```
>> a=2;b=3; fprintf('a is %f and b is %f\n',a,b)
```

where `%f` indicates that a floating point field should be included to take the value of `a` at the end of string of `fprinf` and `\n` indicates "new line". You may use `%i` for integer value instead of `%f`.

Another output command without specification of format is `disp`(display). To see what `disp` does, type (and hit the enter key)

```
>> disp('I like ma303.')
```

Example 8.1. Write an m-file computing the average of a and b.

```
function w=av(a,b)
a=input('a=');
b=input('b=');
w=(a+b)/2;
fprintf('If a=%f and b=%i\n then a+b is %f',a,b,w)
```

Go the command window type (and hit the enter key)

```
>> av
```

and see the difference between `%f` (for a) and `%i` (for b).

8.2. **The Euler method-version 1.** The Euler method gives an approximation of the solution of the initial value problem

$$\frac{dy}{dt} = f(t,y), \quad y(t_0) = y_0.$$

The approximation formula is

$$\begin{cases} h = t_{n+1} - t_n \\ y_{n+1} = y_n + h \cdot f(t_n, y_n) \end{cases}$$

To give an example, write an m-file for $f(t,y)$.

```
function w=f(t,y)
w=1-t+4*y;
```

In the following m-file `eu`, the input data consists of the initial point (t_0, y_0), the uniform step size h and the number n. Matlab will ask you several questions on the input data. After you answere the questions matlab will show you the approximations. The m-file is in the next page.

```
function [t,y]=eu(t,y,h)
disp('My function is')
type f.m;
disp('-----------------------');
t=input('t(0)=');
y=input('y(0)=');
h=input('The step size h=');
n=input('n=');
disp('-----------------------');
t=t;y=y;
for i=1:n
       y=y+h*f(t,y);
       t=t+h;
fprintf('The Euler approximation is \n t_(%i)=%f \n y_(%i)=%f\n',i,t,i,y);
disp('-----------------------');
end
clear;
end
```

The command window looks like

```
>> eu                 %hit the enter key
My function is

function w=f(t,y)
w=1-t+4*y;
-----------------------
t(0)=0                %you type 0 and hit the enter key
y(0)=1                %you type 1 ....
The step size h=0.1 %you type 0.1 ...
n=3                   %you type 3 ...
-----------------------
The Euler approximation is
 t_(1)=0.100000
 y_(1)=1.500000
-----------------------
The Euler approximation is
 t_(2)=0.200000
 y_(2)=2.190000
-----------------------
The Euler approximation is
 t_(3)=0.300000
 y_(3)=3.146000
-----------------------
>>
```

8.3. **The Backward Euler method.** The backward Euler method uses the following recurrence

$$(\star) \quad y_n = y_{n-1} + hf(t_n, y_n)$$

for the initial value problem

$$y' = f(t, y), \quad y(t_0) = y_0.$$

Since y_n is implicit in the recurrence ($\star$) we need to write a different m-file for each initial value problem. For example, if the problem is

$$y' = 1 - t + 4y, \qquad f(t, y) = 1 - t + 4y$$

then the recurrence ($\star$) becomes

$$y_n = y_{n-1} + h(1 - t_n + 4y_n) \implies (\dagger) \quad y_{n+1} = \frac{(y_{n-1} + h(1 - t_n))}{(1 - 4h)}$$

I emphasize again that the formula (†) depends on your problem, i.e., on $f(t, y)$. An m-file for the Backward Euler method is as follows.

```
function [t,y]=beu(t,y,h)
disp('This m-file is for f(t,y)=1-t+4y');
disp('---------input data--------------');
t=input('t(0)=');
y=input('y(0)=');
h=input('The step size h=');
n=input('n=');
disp('------------------------');
t=t;y=y;
for i=1:n
        t=t+h;
        y=(y+h*(1-t))/(1-4*h);  % <- Modify here for your HW assignment
fprintf('The BEU approximation is \n t_(%i)=%f \n y_(%i)=%f\n',i,t,i,y);
disp('------------------------');
end
clear;
```

In the above, the formula

$$(\dagger) \quad y_{n+1} = \frac{(y_{n-1} + h(1 - t_n))}{(1 - 4h)}$$

is coded as `y=(y+h*(1-t))/(1-4*h);`. That is the only line in `beu.m` you need to modify for your problem. Or better, write your own m-file for the problem.

9. Numerical methods-II

9.1. **The improved Euler method.** Here is an M-file for the improved Euler method of which structure is similar to that of `eu.m`.

```
function [t,y]=ieu(t,y,h)
disp('My function is')
type f.m;
disp('------------------------');
t=input('t_(0)=');
y=input('y_(0)=');
h=input('The step size h=');
n=input('n=');
disp('------------------------');
t=t;y=y;
for i=1:n
```

```
        y=y+(h/2)*(f(t,y)+f(t+h,y+h*f(t,y)));
        t=t+h;
fprintf('The IEU approximation is \n t_(%i)=%f \n y_(%i)=%f\n',i,t,i,y);
disp('-----------------------');
end
clear;
end
```

9.2. **The Runge-Kutta method.** Here is an M-file for the Runge-Kutta method.

```
function [t,y]=rk(t,y,h)
disp('My function is')
type f.m;
disp('-----------------------');
t=input('t_(0)=');
y=input('y_(0)=');
h=input('The step size h=');
n=input('n=');
disp('-----------------------');
t=t;y=y;
for i=1:n
        k1=f(t,y);
        k2=f(t+h/2,y+h*k1/2);
        k3=f(t+h/2,y+h*k2/2);
        k4=f(t+h,y+h*k3);
        y=y+h*(k1+2*k2+2*k3+k4)/6;
        t=t+h;
fprintf('The RK-approximation is \n t_(%i)=%f \n y_(%i)=%f\n',i,t,i,y);
disp('-----------------------');
end
clear;
end
```

To run `rk.m` type on the command window:

```
>> rk
```

Then Matlab asks t_0 first, then y_0. These values are given by the initial data

$$y(0) = 1 \implies (t_0, y_0) = (0, 1) \implies t_0 = 0, y_0 = 1.$$

Next Matlab asks the step size h, then n. Note that

$$t_n = t_0 + nh, \quad y_n \approx \phi(t_n)$$

where $\phi(t)$ is the exact solution.

9.3. **Multistep Methods.** Our goal is to get a good approximation of the initial value problem

$$y' = \frac{dy}{dt} = f(t, y), \quad y(t_0) = y_0.$$

Note our convention $f_n = f(t_n, y_n)$ in the following.

(a) The **fourth order Adams-Bashforth formula** is

$$(*) \quad y_{n+1} = y_n + (h/24)(55f_n - 59f_{n-1} + 37f_{n-2} - 9f_{n-1}).$$

(b) The **fourth order Adams-Moulton formula** says

$$(\dagger) \quad y_{n+1} = y_n + (h/24)(9f_{n+1} + 19f_n - 5f_{n-1} + f_{n-2}).$$

The difficulty of using Adams-Moulton formula is that y_{n+1} is implicit in the formula and we can not compute $f_{n+1} = f(t_{n+1}, y_{n+1})$ directly.

(c) We want to develop a **predictor-corrector method** which uses the fourth order Adams-Bashforth formula as a predictor and the fourth order Adams-Moulton formula as a corrector. For this method, we want to use Runge-Kutta method for the first three values of y_n. The specific algorithm is as follows:

Step 1. Input data: t_0, y_0, h, n

Step 2. Compute y_1, y_2, y_3 by the Runge-Kutta method.

Step 3. Given $y_{n-3}, y_{n-2}, y_{n-1}, y_n$ compute y_{n+1} by the fourth order Adams-Bashforth formula.

Step 4. Now use $y_{n-2}, y_{n-1}, y_{n-1}, y_n, y_{n+1}$ to compute the corrected $f_{n+1} = f(t_{n+1}, y_{n+1})$ by the fourth order Adams-Moulton formula and then compute y_{n+1}.

We want to approximate the solution of

$$y' = 1 - t + 4y, \quad y(0) = 1$$

by using the predictor-corrector method which uses the fourth order Adams-Bashforth formula as a predictor and the fourth order Adams-Moulton formula as a corrector. First, write an m-file for $f(t, y)$.

```
function w=f(t,y)
w=1-t+4*y;
```

and then write `rkg.m` for the Runge-Kutta method for y_1, y_2, y_3:

```
function [T,Y]=rkg(t,y,tf,n)
h=(tf-t)/n;
T=t;Y=y;
for i=1:n
        k1=f(t,y);
        k2=f(t+h/2,y+h*k1/2);
        k3=f(t+h/2,y+h*k2/2);
        k4=f(t+h,y+h*k3);
        y=y+h*(k1+2*k2+2*k3+k4)/6;
        t=t+h;
        T=[T;t];
        Y=[Y;y];
end
```

Finally, write `pc.m` for the predictor-corrector method described above.

```
function [t,y]=pc(t,y,h)
disp('y=f(t,y) where f(t,y) is')
disp('-----------------------');
type f.m;
disp('-----------------------');
t=input('t_(0)=');
y=input('y_(0)=');
h=input('The step size h=');
```

```
n=input('n=');
disp('-----------------------');
tf=t+3*h;
[T,Y]=rkg(t,y,tf,3);
  disp('---------- Runge-Kutta for n=0,1,2,3 ---------')
  fprintf('t= %f \n',T);
  fprintf('y=%f \n',Y);
  disp('---------- Predictor-Corrector for n>3 -------');
t=T;
y=Y;
F=f(T,Y)';
if n<4  disp( 'If n < 4 then the approximation is just the Runge-Kutta method.')
else
 for i=5:n+1
    y(i)=y(i-1)+(h/24)*(55*F(i-1)-59*F(i-2)+37*F(i-3)-9*F(i-4));
    t(i)=t(i-1)+h;
    F(i)=f(t(i),y(i));
    y(i)=y(i-1)+(h/24)*(9*F(i)+19*F(i-1)-5*F(i-2)+F(i-3));
    fprintf('t_(%i)=%f \ny_(%i)=%f\n',i-1,t(i),i-1,y(i));
    disp('-----------------');
 end
end
clear
```

In summary, you need three m-files `f.m`, `rkg.m` and `pc.m` for the predictor and corrector methods. To run `pc.m` type on the command window:

```
>> pc
```

Then Matlab asks t_0 first, then y_0. These values are given by the initial data

$$y(0) = 1 \implies (t_0, y_0) = (0, 1) \implies t_0 = 0, y_0 = 1.$$

Next Matlab asks the step size h, then n. Recall that

$$t_n = t_0 + nh, \quad y_n \approx \phi(t_n)$$

where $\phi(t)$ is the exact solution.

9.4. **The graph of an approximation.** Consider the problem

$$y' = f(t, y) = 2ty^2, \quad y(0) = 0.1.$$

We shall use the Runge-Kutta method to find an approximation of the solutions on $[0, 3] = [t_0, t_f]$ and plot the graph of the approximated solution. In this special case we can find the exact solution

$$y = \phi(t) = \frac{1}{10 - t^2}.$$

First we make an m-file for the function $f(t, y) = 2ty^2$.

```
function w=f(t,y)
w=2*t*y^2;
```

Then save `rkg.m` in your current directory.

```
function [T,Y]=rkg(t,y,tf,n)
h=(tf-t)/n;
T=t;Y=y;
for i=1:n
        k1=f(t,y);
        k2=f(t+h/2,y+h*k1/2);
        k3=f(t+h/2,y+h*k2/2);
        k4=f(t+h,y+h*k3);
        y=y+h*(k1+2*k2+2*k3+k4)/6;
        t=t+h;
        T=[T;t];
        Y=[Y;y];
end
```

To plot the graph of the solution approximated by the Runge-Kutta method with $t_0 = 0$, $y_0 = 0.1$, $t_f = 3$ (the final t-point) and $n = 20$, type

```
>> clear
>> [t,y]=rkg(0,0.1,3,20);
>> plot(t,y,'*')
```

If you want to see how accurate the Runge-Kutta method is you may want to plot the exact solution

$$y = \phi(t) = \frac{1}{10 - t^2}.$$

```
>> hold on
>> t=0:0.1:3;
>> plot(t,1./(10.-t.^2),'red')
```

The approximated solution is given by blue $*$ while the exact solution is given by the red solid line(curve) (See Figure 11.6). The figure shows that the Runge-Kutta method is quite accurate at least on $[0, 3]$. However, as we increase the interval the approximation becomes poorer. For example, plot the approximated solution and the exact solution on $[0, 10]$ to see this phenomenon.

a

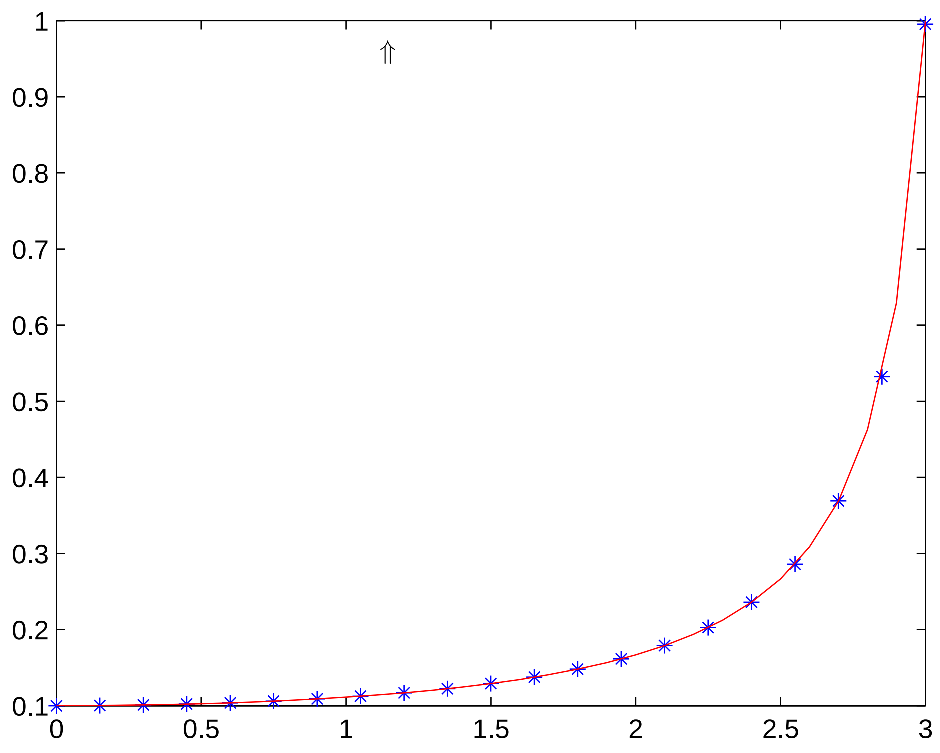

FIGURE 9.1. The graphs of the approximated solution by the blue $*$ and the exact solution by the solid red line.

10. Fourier expansion

10.1. **Periodic functions.** A function $f(x)$ is periodic with period T if

$$f(x+T) = f(x) \quad \text{for all } x.$$

Problem: Given a function $f(x)$ defined on an interval $[a, b)$, make an M-file for the period extension of $f(x)$ with period $T = b - a$.

x

Solution) Given an interval $[a, b)$, let $T = b - a$. Then

$$z(x) = T\left[\frac{x-a}{T}\right] \implies z(x+T) = z(x) + T \text{ for all } x.$$

So the function $\widetilde{f}(x)$ is the desired extension

$$\widetilde{f}(x) := f(x - z(x))$$

since $\widetilde{f}(x) = f(x)$ for $a \le x < b$ and $\widetilde{f}(x+T) = \widetilde{f}(x)$.

In summary, given a function $f(x)$ on $a \le x < b$. To extend $f(x)$ with period $T = b - a$ you need to write the following generic code:

```
function w=ef(x)
z=T*floor((x-a)/T);
y=x-z;
w=f(y);
```

Note that `ef` cannot be run unless you specify a, T, f, i.e., it depends on $f(x)$. I will give you a few examples.

Example 10.1. Let $g(x) = x^2$ for $-2 \le x < 3$. Make an M-file for the periodic extension of $g(x)$.

Answer) Here $a = -2, T = 3 - (-2) = 5$ and $g(x) = x^2$. An m-file for the periodic extension is

```
function w=g(x)
z=5*floor((x+2)/5);
y=x-z;
w=y^2;
```

To make sure we get the right extension, let us plot the graph

```
>> fplot('g(x)',[-7,8])
```

Example 10.2. Let $h(x) = |x|$ for $-3 \le x < 3$. Make an M-file for the periodic extension of $h(x)$.

Answer) Here $a = -3, T = 3 - (-3) = 6$ and $h(x) = |x|$. An m-file for the periodic extension is

```
function w=h(x)
z=6*floor((x+3)/6);
y=x-z;
w=abs(y);
```

where `abs(y)` means $|y|$.

Example 10.3. Let

$$f(x) = \begin{cases} 0 & \text{if } -\pi \le x < 0 \\ x & \text{if } 0 \le x < \pi. \end{cases}$$

Make an M-file for the periodic extension of $f(x)$ with period $T = 2\pi$.

Answer) Here $a = -\pi, T = \pi - (-\pi) = 2\pi$. An m-file for the periodic extension is:

```
function w=f(x)
z=(2*pi)*floor((x+pi)/(2*pi));
y=x-z;
if y<0
  w=0;
else
  w=y;
end
```

10.2. **Fourier expansion.** Given a periodic function $f(x)$ with period $T = 2L$, the Fourier expansion of $f(x)$ is

$$f(x) = \frac{a_0}{2} + \sum_{m=1}^{\infty} a_m \cos(m\pi x/L) + b_m \sin(m\pi x/L)$$

where

$$a_0 = \frac{1}{L}\int_{-L}^{L} f(x)dx, \quad a_m = \frac{1}{L}\int_{-L}^{L} f(x)\cos(m\pi x/L)dx, \quad b_m = \frac{1}{L}\int_{-L}^{L} f(x)\sin(m\pi x/L)dx.$$

Let $s_n(x)$ be its n-th partial sum

$$s_n(x) = \frac{a_0}{2} + \sum_{m=1}^{n} a_m \cos(m\pi x/L) + b_m \sin(m\pi x/L).$$

Example 10.4. Let

$$f(x) = \begin{cases} 0 & \text{if } -\pi \le x < 0 \\ x & \text{if } 0 \le x < \pi. \end{cases}$$

and extend $f(x)$ by $f(x + 2\pi) = f(x)$.

We compute the Fourier coefficients;

$$a_0 = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nx dx = \frac{\pi}{2},$$

$$a_m = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nx dx = \begin{cases} 0 & \text{if } m = 2k \quad (m\text{: even}), \\ \frac{-2}{(2k-1)^2\pi} & \text{if } m = 2k-1 \quad (m\text{: odd}). \end{cases}$$

and

$$b_m = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin nx dx = \frac{(-1)^{m+1}}{m}.$$

Thus the n-th Fourier sum is

$$s_n(x) = \frac{\pi}{4} + \sum_{m=1}^{m} \left(\frac{-2}{(2m-1)^2\pi}\cos(2m-1)x + \frac{(-1)^{m+1}}{m}\sin mx \right).$$

We want to write an M-file for Fourier partial sum.

```
function s=fsum(x,n)
s=pi/4;
for m=1:n
```

```
  s=s+(-2)/((2*m-1)^2*pi) *cos((2*m-1)*x)+(-1)^(m+1)/m *sin(m*x);
end
```

Recall that the periodic extension of $f(x)$ can be implemented in Matlab as

```
function w=f(x)
z=(2*pi)*floor((x+pi)/(2*pi));
y=x-z;
if y<0
  w=0;
else
  w=y;
end
```

Let us plot the graph of $f(x)$ and $s_n(x)$ on the same figure. Let us take $n = 10$.

```
>> hold on
>> fplot('fsum(x,10)',[-10,10])
>> fplot('f(x)',[-10,10],'--')
>> hold off
```

The theory says $s_n(\pi) \to (f(\pi+) + f(\pi-))/2 = \frac{\pi}{2}$ as $n \to \infty$. Do you see it from the figure with $n = 10$ (See Figure 10.4)?

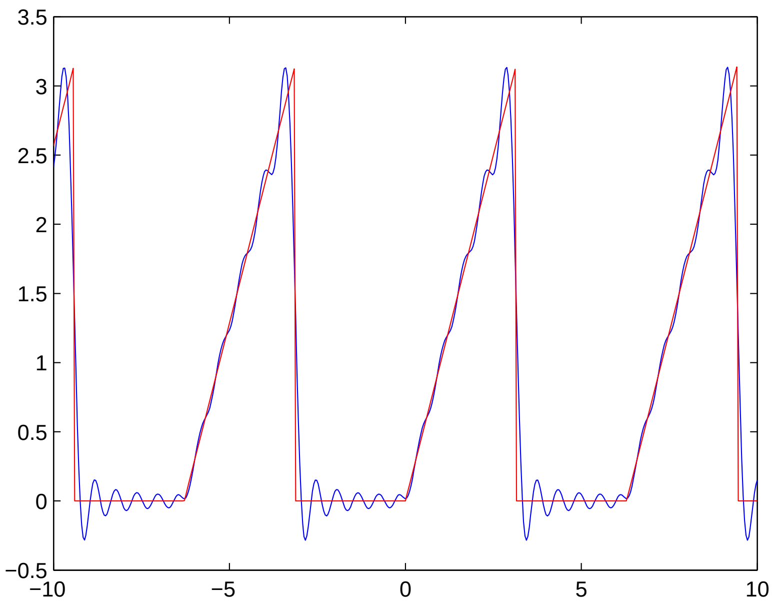

FIGURE 10.1. The graph of $f(x)$ by the red lines and the graph of $s_{10}(x)$ by the blue lines.

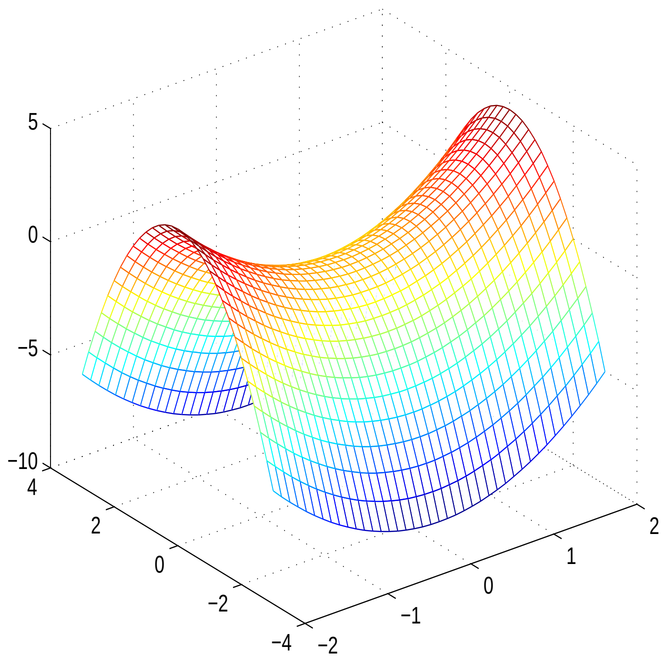

FIGURE 11.1. The 3d-graph of $z = x^2 - y^2$

11. Partial differential equations

11.1. **3d-plot.** Before we go into more serious partial differential equation problems we learn how to plot 3d graphs. We know from calculus that the graph $z = f(x, y)$ is a surface in $\mathbb{R}^3$. Let us plot the graph using Matlab.

Example 11.1. Plot the graph of the function

$$z = f(x, y) = x^2 - y^2, \quad -2 \le x \le 2, \quad -3 \le y \le 3$$

with $\Delta x = 0.1$ and $\Delta y = 0.2$.
Type the following on the command window

```
>> [x,y]=meshgrid(-2:0.1:2,-3:0.2:3);
>> mesh(x,y,x.^2-y.^2)
```

Explanation) The first line generates 40×50 matrix and the second line plots (x, y, z)- mesh with $\Delta x = 0.1$ and $\Delta y = 0.2$. You must write `x.^2-y.^2` (**not** `x^2-y^2`). Otherwise Matlab will complain about missing `.`'s.

11.2. **The heat equation.** Recall that the solution of the heat equation

$$\begin{cases} \alpha^2 u_{xx} = u_t \\ u(x,0) = f(x) \\ u(0,t) = u(L,t) = 0, \end{cases}$$

where $u(x,t)$ denotes the temperature of a rod at (x,t) for $0 \le x \le L$ and $t > 0$, is given by

$$u(x,t) = \sum_{m=1}^{\infty} c_m e^{-\frac{m^2\pi^2}{L^2}\alpha^2 t} \sin\left(\frac{m\pi x}{L}\right), \quad c_m = \frac{2}{L}\int_0^L f(x)\sin\left(\frac{m\pi x}{L}\right) dx.$$

Example 11.2. Consider the heat equation with boundary condition

$$\begin{cases} u_{xx} = u_t \\ u(x,0) = f(x) \\ u(0,t) = u(2,t) = 0 \end{cases}$$

where

$$f(x) = \begin{cases} 0 & \text{if } 0 \le x \le 1 \\ 1 & \text{if } 1 < x \le 2. \end{cases}$$

Here we have $\alpha = 1$ and $L = 2$. The solution $u(x,t)$ is given by

$$u(x,t) = \sum_{m=1}^{\infty} c_m e^{-m^2\pi^2 t/4} \sin\frac{m\pi x}{2}, \quad c_m = \frac{2}{m\pi}\Big(\cos(m\pi/2) - \cos(m\pi)\Big).$$

Define the n-th partial sum

$$u(x,t,n) = \sum_{m=1}^{n} c_m e^{-m^2\pi^2 t/4} \sin\frac{m\pi x}{2}, \quad c_m = \frac{2}{m\pi}\Big(\cos(m\pi/2) - \cos(m\pi)\Big).$$

An M-file for $u(x,t,n)$ can be written as:

```
function w=u(x,t,n)
w=0;
for m=1:n
w=w+(2/(m*pi))*(cos(m*pi/2)-cos(m*pi))*exp(-m^2*pi^2*t/4).*sin(m*pi*x/2);
end
```

Notice the .* above. We will do a 3-dimensional plot and `u.m` needs to be array smart for this. We shall use this m-file `u.m` as the main example throughout.

Example 11.3. Let $n = 50$. Plot the graph of $u(x,t,50)$ versus x for $t = 0, 0.1, 0.2, 0.4, 1, 5$ (See Figure 11.2). Type in the following in the command window

```
>> hold on
>> fplot('u(x,0,50)',[0,2],'blue')
>> fplot('u(x,0.1,50)',[0,2],'magenta')
>> fplot('u(x,0.2,50)',[0,2],'green')
>> fplot('u(x,0.4,50)',[0,2],'yellow')
>> fplot('u(x,1,50)',[0,2],'red')
>> fplot('u(x,5,50)',[0,2],'black')
>> hold off
```

Note that the graph of each $u(x,t)$ for different values of t is colored by

$t = 0$(blue) $\to t = 0.1$(magenta) $\to t = 0.2$(green) $\to t = 0.4$(yellow) $\to t = 1$(red) $\to t = 5$(black)

Example 11.4. Use $n = 50$ for $u(x,t,n)$. How long does it take for the entire rod to cool off to a temperature of no more than 0.2 degrees? (Here we still use $u(x,t)$ of Example 11.2.)

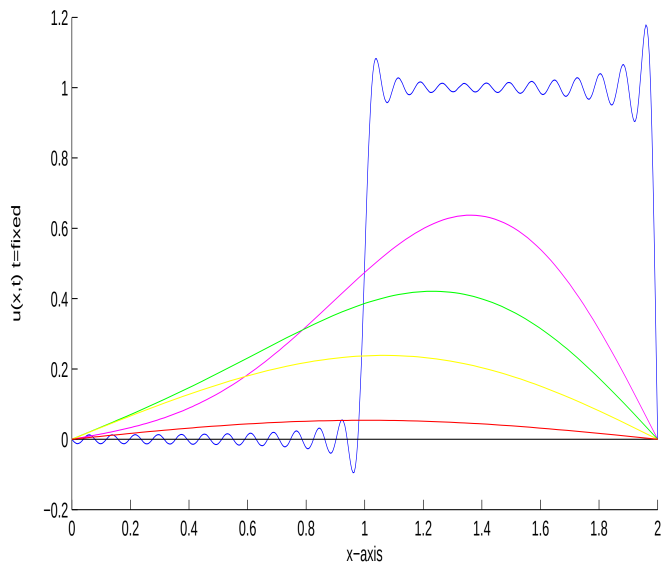

FIGURE 11.2. The graphs of $u(x,t)$ for $t = 0, 0.1, 0.2, 0.4, 1, 5$

Solution) Since we need to find the smallest t_0 for which $u(x,t,50) \leq 0.2$ for all x in $[0,2]$ and $t \geq t_0$, we shall use the method of maximum

$$\max_{0\leq x\leq 2} u(x,t_0,50) \leq 0.2.$$

For this task, recall that the command `xM=fminbnd('-f(x)',a,b)` gives us the point x_M where $f(x)$ attains the maximum value on $[a,b]$. To find the maximum value $f(x_M)$ we also need to evaluate `f(xM)`.

Looking at Figure 11.2 we find that a portion of the graph of $u(x,0.4)$ is above 0.2 but the maximum is pretty close to 0.2 (The yellow curve for $t = 0.4$ in Figure 11.2). And so $0.4 < t_0$ and take this information as the guiding light. Now we proceed analytically. We shall try $t_0 = 0.4$, $t_0 = 0.5$ so on such that $\max\{u(x,t_0,50)\} \leq 0.2$.

```
>> xM=fminbnd('-u(x,0.4,50)',0,2); u(xM,0.4,50)

ans =

    0.2385

>> xM=fminbnd('-u(x,0.5,50)',0,2); u(xM,0.5,50)

ans =

    0.1856
```

This means $0.4 \leq t_0 \leq 0.5$. Next we try

```
>> xM=fminbnd('-u(x,0.45,50)',0,2); u(xM,0.45,50)

ans =

    0.2103
```

This means $0.45 \le t_0 \le 0.5$. Let us try $t_0 = 0.47$;

```
>> xM=fminbnd('-u(x,0.47,50)',0,2); u(xM,0.47,50)

ans =

    0.2000
```

Bingo! The temperature of the entire rod is below 0.2 degrees after $t_0 = 0.47$ (seconds or minutes etc).

Example 11.5. Plot the graph of $u(x, t, n)$ with $x = 1.5$ and $n = 50$ in $0 \le t \le 3$. This graph will show how the temperature changes at $x = 1.5$ as time goes by.

This is simple but tricky if you use `fplot`. Since `fplot` accepts only x as the variable we need to type `u(1.5,x,50)` (**not** `u(1.5,t,50)`). Therefore the command we need is (See Figure 11.3).

```
>> fplot('u(1.5,x,50)',[0,3])
```

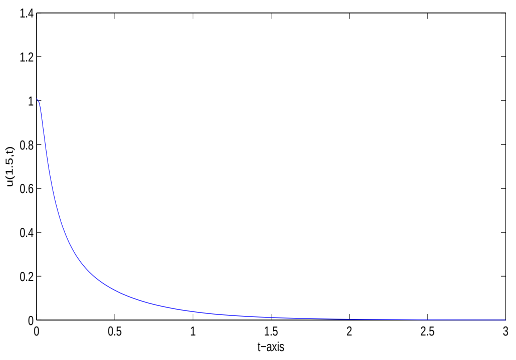

FIGURE 11.3. The graph of $u(1.5, t)$

Example 11.6. Plot the 3d-graph of $u(x, t, n)$ over the rectangle

$$0 \le x \le 2, \quad 0 \le t \le 5$$

with $\Delta x = 0.1, \Delta t = 0.05$ and $n = 50$.

Just type the following in the command window (See Figure 11.4).

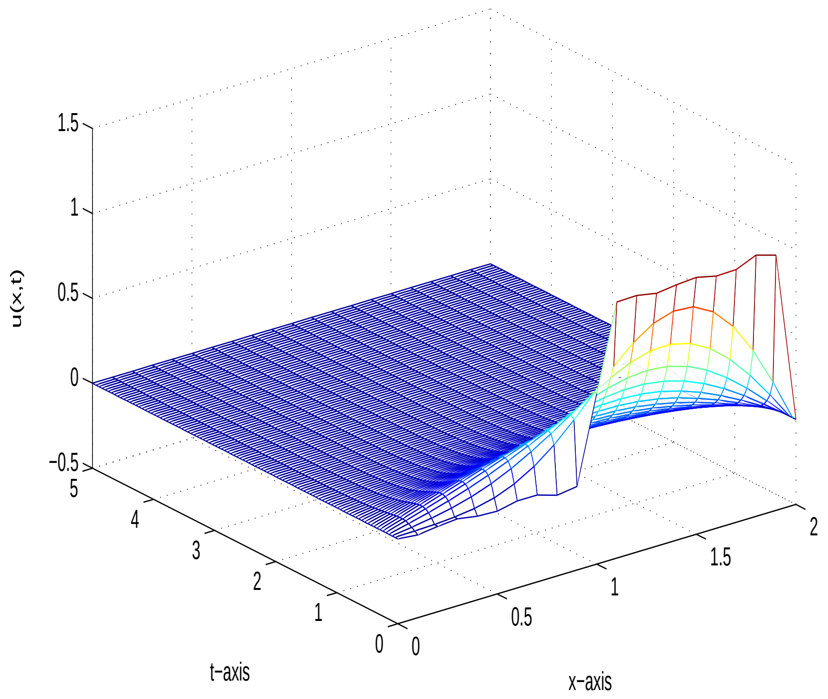

FIGURE 11.4. The 3d-graph of $z = u(x,t)$

```
>> [x,t]=meshgrid(0:0.1:2,0:0.05:5);
>> mesh(x,t,u(x,t,50))
```

The front of the surface $z = u(x,t)$ represents the temperature of $u(x,t)$ when t is small (i.e., early temperature) and the rear of the surface represents the temperature of $u(x,t)$ when t is large (i.e., later temperature). We see that the temperature stabilizes and we have the steady temperature $u(x,t) = 0$ after a long period of time (though $t = 4$ is considered a long period of time for this problem).