1 Helpful tools

help xxx displays help on topic `xxx`

diary logs all actions into a file

echo on/off displays the commands being executed

2 Variables

1. All variables in matlab are matrices!
2. Variables are assigned with "=":

`x = 5;`	$x = 5$
`u = [1 2 3];`	$u = (1, 2, 3)$
`v = [1;2;3];`	$v = (1, 2, 3)^T$
`A = [1 2 3; 4 5 6];`	$A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$

3. There are special functions for creating arrays:

`u=1:n` creates a row vector with values 1 to n.

`u=a:dx:b` creates a row vector with values from a to b at increments dx.

`t=linspace(a,b,n)` returns a row vector of equidistant values in $[a, b]$. If n is omitted, its default value is 100.

`[x,y]=meshgrid(t,t)` creates a two-dimensional mesh based on the vector `t`, which could for instance be the result of `linspace`. The notation `[x,y]` indicates that the function returns two vectors, the x and y coordinates of the mesh.

3 Multiplication operators

Warning: assume you have two variables x and y, which correspond to function values in the points of an interval. Since all variables are vectors, the term `x * y` refers to the inner product of the two vectors, which may not be what you desire. If you want to multiply each entry of x with the corresponding entry of y, you need to use the form `x .* y`. Similarly, use `x ./ y` and `x .^ 2` for division and powers, respectively.

4 m-files

Instead of typing every command into the matlab command window, you can prepare a file with a matlab program and run it. These files have a suffix `.m`, so a typical name looks like `name.m`.

And easy way to produce m-files on the way is the diary option:

```
diary test.m
x = 0:.02:1;
y = sin(2*pi*x);
diary off
```

will produce a file with exactly the same commands, except the last one (). Using the menu on top, you can load the file and then run it. You can also edit it, save the edits and run the file again later.

5 Plotting functions

Note: there may be versions allowing you to plot functions directly. Nevertheless, for our applications, it is more useful to be able to plot vectors.

5.1 Line plots: plot a function $f(x)$.

1. Generate a vector of values for the independent variable x:
2. Generate a vector of values for the independent variable y:
3. Plot the vectors using the `plot` command

```
x = 0:.02:1;
y = sin(2*pi*x);
plot(x,y);
```

5.2 Surface plots: plot a function $f(x, y)$.

1. Generate a mesh of values for the independent variables x, y:
2. Generate a vector of values for the independent variable z:
3. Plot the vectors using the `surf` command

```
t = 0:.02:1;
[x,y] = meshgrid(t,t);
z = sin(2*pi*x).*sin(2*pi*y);
surf(x,y,z);
```

5.3 Vector plots: plot a vector field $(u(x, y), v(x, y))$.

1. Generate a mesh of values for the independent variables x, y:
2. Generate vectors of values for the independent variables u, v:
3. Plot the vectors using the `surf` command

```
t = -1:.1:1;
[x,y] = meshgrid(t,t);
u = -y;
v = x;
quiver(x,y,u,v);
```

5.4 Additional plot topics

`hold on` allows you to combine several plots into one. Use `hold off` to start a new plot.

6 Defining functions

6.1 Handles to anonymous functions: The simplest way to define a function is via a *handle* to an *anonymous function*.

```
 f = @(t,y) y.*(1-y);
```

creates a function handle to a function $f(t, y) = y(1 - y)$, which can be used subsequently, for instance like in `ode45(f,[0 1], y0)`, see section 7.

6.2 Named functions: If your functions are more complex or you think you will want to reuse them, you can put them into a separate file. If you want to call your function *harry*, then you have to create the file `harry.m` which starts with a statement

7 Solving IVP

Matlab has several functions approximating the solution to initial value problems, a useful one is `ode45`. Assume you want to solve the problem

$$\begin{aligned} y' &= f(t,y) \\ y(0) &= y_0 \end{aligned} \tag{7.1}$$

on the interval $I = [0, 1]$ with $f(t, y) = y(1 - y)$ and $y_0 = 2$. This translates into the matlab code

```
f = @(t,y) y.*(1-y);
y0 = 2;
[t y] = ode45(f, [0 1], y0);
```

8 Defining vector valued functions

Like in section 6, we describe how to create a handle to an anonymous function: let $\mathbf{f}(t,\mathbf{y}) = \mathbf{A}\mathbf{y} + \mathbf{b}$, with $\mathbf{A} = \begin{pmatrix} -0.5 & 1 \\ -1 & -0.5 \end{pmatrix}$ and $\mathbf{b} = (1,2)^T$. In matlab, this looks like

```
f = @(t,y) [ -0.5.*y(1)+y(2)+1 ; -y(1)-0.5.*y(2)+2 ];
```

Note the "`.*`"! The "`;`" produces a column vector, which is what we need for `ode45`.

9 Solving IVP for systems of ODE

The goal is the same as in section 7, just that now y and $f(t,y)$ are vectors. Thus, equation (7.1) becomes

$$\begin{pmatrix} y_1' \\ \vdots \\ y_n' \end{pmatrix} = \begin{pmatrix} f_1(t,\mathbf{y}) \\ \vdots \\ f_n(t,\mathbf{y}) \end{pmatrix} \qquad \begin{pmatrix} y_1(0) \\ \vdots \\ y_n(0) \end{pmatrix} = \begin{pmatrix} y_1^0 \\ \vdots \\ y_n^0 \end{pmatrix} \tag{9.1}$$

Here an example for the system

$$\begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} & 1 \\ -1 & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 2 \end{pmatrix} \qquad \mathbf{y}_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

In matlab, we can solve this by

```
f = @(t,y) [ -0.5.*y(1)+y(2)+1 ; -y(1)-0.5.*y(2)+2 ];

[t,y] = ode45(f, [0 10], [1 0]);

plot (y(:,1), y(:,2));
```

In order to plot a phase portrait, you could add the following lines

```
hold on;

[t,y] = ode45(f, [0 10], [2 1]);
plot (y(:,1), y(:,2));

[t,y] = ode45(f, [0 10], [3 0]);
plot (y(:,1), y(:,2));

[t,y] = ode45(f, [0 10], [2 -1]);
plot (y(:,1), y(:,2));
```

10 Examples

In this section, you will find some complete example programs which you can run and modify to your needs.

10.1 Plotting a direction field and solutions to an ODE

```
f = @(t,y) y.*(1-y);

t = -1:.1:2;
[x,y] = meshgrid(t,t);
u = x.^ 0.;
v = f(x,y);
quiver(x,y,u,v);
hold on;

[t1 s1] = ode45(f,[-1 2], .2);
[t2 s2] = ode45(f,[-1 2], 2.);
[t3 s3] = ode45(f,[-1 2], .1);

plot (t1, s1, t2, s2, t3, s3);
```