

## Worksheet 1, for the MATLAB course by Hans G. Feichtinger, Edinburgh, Jan. 9th

**Start MATLAB via: Start > Programs > School applications > Sci.+Eng. > Eng.+Electronics > MATLAB > R2007 (preferably).**

Generally speaking I suggest that you start your session by opening up a diary with the command:

`diary mydiary1.m` and concluding the session with the command `diary off`. If you want to save your workspace you may want to call `save today1.m` in order to save all the current variable (names + values).

Moreover, using the `HELP` command from MATLAB you can get help on more or less every MATLAB command. Try simply `help inv` or `help plot`.

1. Define an arbitrary (random)  $3 \times 3$  matrix  $A$ , and check whether it is invertible. Calculate the inverse matrix. Then define an arbitrary right hand side vector  $b$  and determine the (unique) solution to the equation  $A * x = b$ . Find in two different ways the solution to this problem, by either using the inverse matrix, or alternatively by applying Gauss elimination (i.e. the `RREF` command) to the extended system matrix  $[A, b]$ . In addition look at the output of the command `rref([A, eye(3)])`. What does it tell you?
2. Produce an “arbitrary”  $7 \times 7$  matrix of rank 5. There are at least two simple ways to do this. Either by factorization, i.e. by obtaining it as a product of some  $7 \times 5$  matrix with another random  $5 \times 7$  matrix, or by purposely making two of the rows or columns linear dependent from the remaining ones. Maybe the second method is interesting (if you have time also try the first one):
3. Plotting routines are quite simple in MATLAB. `plot(d)` simply plots the data vector  $d$  of length  $n$  over  $1 \dots n$ . If  $d$  is real-valued this is done in the usual way, if  $d$  is complexvalued, e.g.  $x = \text{rand}(1, n) + i * \text{rand}(1, n)$  then the plot is performed in the complex plane. So try e.g.  
`n = 35; x = rand(1,n) + i * rand(1,n); plot(x,'r*'); title('a plot in the complex plane');`

See what happens if you do in addition: `axis square; plot(x,'k'); hold off; grid; figure(gcf);` If you want to mark the first and the last point on this polygonal curve specifically add `hold on; plot(x(1),'b*'); plot(x(n),'g*');` `hold off; figure(gcf);` This last command “figure(gcf)” brings the last figure to the surface. The command “figure” itself opens up a new figure.

If a matrix is inserted into the plotting command the entries (columns) are plotted column-wise. Just as an example try: `T = zeros(8,6); T(:) = 4 : 51; plot(T); xlabel('plotting a matrix');` From such a plot you can learn the sequence of colors automatically assigned to the first, second and so on column to be depicted as a graph.

4. Generate an arithmetic progression consisting of  $12 + 1$  equidistant points on the interval  $[0, 2\pi]$  by resizing the simple arithmetic progression  $0 : 1/12 : 1$  by the factor  $2 * \pi$ . Recalling then *Euler's formula*

$$e^{ix} = \cos(x) + i * \sin(x)$$

or simple using the fact that the coordinates of a point on the unit circle are expressed as a pair  $(\cos(x), \sin(x))$  for some  $x \in [0, 2\pi)$  we can obtain the set of unit roots of order 12 by the simple command `r12 = exp(i * bas)`. To check that we have obtained the right object we can plot it in the complex plane, simply by the command `plot(r12); axis square;` You may want to mark the corners specifically with the additional command `hold on; plot(r12,'r*'); hold off; figure(gcf);` .

I can/will also make the LATEX code for THIS document and diaries available to you!

How can you realize such a task. Take the unit roots of order 12 above as example. The MATLAB input could be

```
>> format compact
>> diary r12plot.m
>> r = 0 : 1/12 : 1
r =
    0    0.0833    0.1667    0.2500    0.3333    0.4167    0.5000    0.5833
                                0.6667    0.7500    0.8333    0.9167    1.0000

>> bas = 2*pi*r;
>> r12 = exp(i*bas);
>> plot(r12); hold on; plot(r12,'r*'); hold off; title('unit roots of order 12');
>> grid; axis square; figure(gcf);
>> diary off;
```

Then you would edit this file, to allow such a plot for general  $n$ , not just  $n = 12$ . Hence we continue, by calling now the command

```
>> edit r12plot
```

in order to obtain this file:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% PLUROOTS.M
%
% Plot unit-roots
%
% USAGE: rts = pluroots(n);

% this is only a comment which will NOT be displayed upon calling
% 'help pluroots'. Of course the file has to be in your working directory
% in order to be accessible
% 'which pluroots' displays that directory
% 'pwd' prints the current working directory

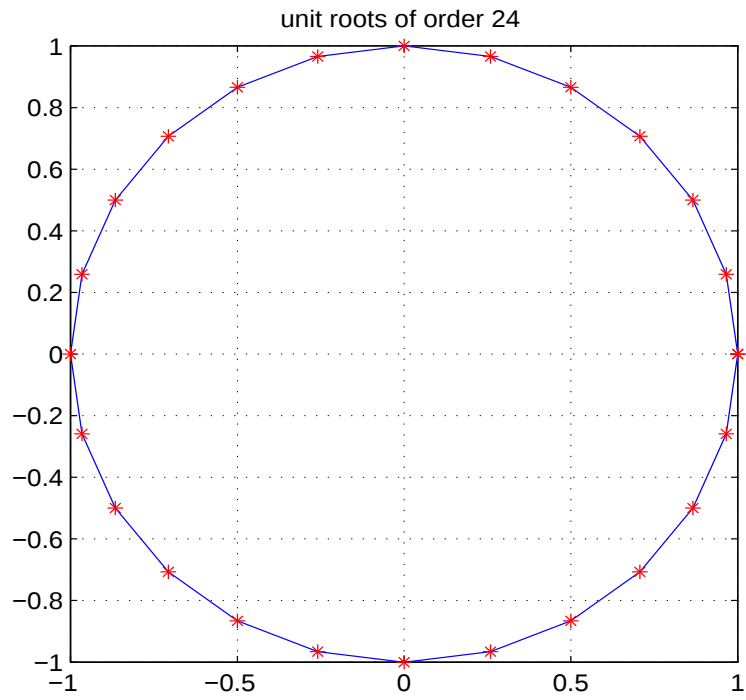
function r12 = pluroots(n);

if nargin == 0; n = 12; end;
r = 0: 1/n :1;
bas = 2*pi*r;
r12 = exp(i*bas);
plot(r12); hold on; plot(r12,'r*'); hold off;
title(['unit roots of order ' num2str(n) ' ']);
grid; axis square; figure(gcf);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

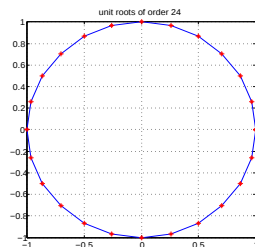
This file can now called as `pluroots(24)` in order to see a similar plot of the unit roots of order 24. You can save and store the figure (by clicking on the icon “file” on the left top the display, then go into export and

choose the mode, such as \*.eps or \*.jpg or \*.pdf format. I usually export into EPS and run a converter called “epstopdf” on the resulting EPS-file, which takes away the unnecessary spaces.

Now we include the thus converted figure into the report:



If you do not go via EPS you get a lot of empty space around, but probably you can fix that using some commands at the time of PDF output from MATLAB. Concrete dimensions obtained experimentally!



TENTATIVE! SO FAR

1. Write a routine (an M-file which you may want to call `testmvmul.m`) which checks that the effect of matrix multiplication of some  $m \times n$ -matrix  $A$  with some column vector  $x$  is just a linear combination of the columns of  $A$  (let us describe them mathematically as  $a^k$ , as a symbol for the  $k$ -th column, i.e. check that  $A * x = \sum_{k=1}^n x_k a^k$ ).
2. Write a routine defining a “random complex matrix” with entries (both real and imaginary part) equidistributed in the interval  $[-1/2, 1/2]$ , and call it `RANDC.M`. Input should be the same as that of `RAND` (call “help rand”);
3. Take an arbitrary rectangular matrix and verify that for any pair of complex-valued vectors  $x \in C^n$  and  $y \in C^m$  one has:

$$\langle A * x, y \rangle = \langle x, A' * y \rangle.$$

*Memorize this by thinking:* moving the matrix to the other side of the scalar product one has to replace the matrix  $A$  by  $A'$ , i.e. one has to add the conjugation and transposition to the matrix (in this way one thinks more symmetrically of this fact).

4. Verify on an arbitrary complex, rectangular matrix that the following two spaces are pairwise orthogonal: the null-space of  $A'$  and the range space or column space of  $A$  (and by the same reasoning the null-space of  $A$  and the column-space of  $A'$ , usually referred to as the *row-space* of  $A$  are orthogonal complements to each other!

EXTRA comment: Therefore in both cases their dimensions add up to  $n$  and  $m$  respectively. This explains the well-known dimension formula:

$$\text{defect}(A) + \text{rank}(A) = n$$

which is best seen via Gauss elimination again: There are  $r = \text{rank}(A)$  pivot elements, and  $n - r$  free variables, determining the dimension of the nullspace of  $A$ . The MATLAB command `null(A)`; provides an orthonormal basis for that space directly.

- 5.
- 6.
- 7.
8. What is known to you about “condition numbers of invertible matrices”?

## TOPICS for project work

The principles of the JPEG compression method

Tomography seen from a linear algebra point of view

MP3 and the Short-time Fourier transform

What makes the FFT a fast algorithm (and how is speed described in this context)

Numerical stability: SVD and the idea of regularization

The principal of symmetric orthonormalization (according to Loewdin)

Frames in Euclidean spaces (redundant sets of generating vectors)

Connections between the Fourier transform, Fourier Series and the FFT

Fourier transform, convolutions, with an application to the heat transform

B-splines and their approximation properties

Gabor frames: a “micro-tonal piano” producing arbitrary signals