Environmental Modeling

Ekkehard Holzbecher

Environmental Modeling

Using MATLAB®

With 148 Figures and 18 Tables

Springer

Ekkehard Holzbecher

Weierstraß-Institut für
Angewandte Analysis und Stochastik (WIAS)
Mohrenstr. 39
10117 Berlin
Deutschland
holzbecher@wias-berlin.de

Library of Congress Control Number: 2007929736

ISBN 978-3-540-72936-5 Springer Berlin Heidelberg New York

Springer is a part of Springer Science+Business Media
springer.com

Typesetting: by the author and Integra, India using a Springer LaTeX macro package
Cover design: deblik, Berlin

Printed on acid-free paper SPIN: 11531234 5 4 3 2 1 0

Dedicated to my wife Susanne
and my children Gesa and Gero

Preface

"Environmental Modeling using MATLAB® " by Ekkehard Holzbecher is an excellent publication and a novel approach covering the intersection of two important, growing worlds – the world of environmental modeling and of mathematical software.

Environmental modeling is a science that uses mathematics and computers to simulate physical and chemical phenomena in the environment (e.g., environmental pollution). This science was initially based on pen-and-paper calculations using simple equations. In the last 50 years, with the development of digital computers, environmental models have become more and more complex, requiring often numerical solutions for systems of partial differential equations.

Mathematical software, such as MATLAB® , has been developed in the last two decades. These packages have been particularly successful for users of personal computers. Mathematical software provides a set of tools for solving equations both analytically and numerically. This is a major improvement in comparison to the programming tools (e.g., FORTRAN) previously used by scientists. Mathematical software offers extremely valuable and cost-effective tools that improve the productivity of the programmer by at least an order of magnitude. The use of these tools also minimizes the risk of programming errors. In addition, mathematical software offers unique visualization tools that allow the user to immediately visualize and often animate simulation results.

Scientists who become familiar with a tool like MATLAB® will never go back to previous ways of computer programming.

The book "Environmental Modeling using MATLAB® " provides a clear, comprehensive, and very instructive introduction to the science of environmental modeling, and more importantly, includes the MATLAB® codes for the actual solutions to the environmental equations. MATLAB® codes are listed in the book and also included as more complete versions in an attached CD.

I highly recommend this book to both beginners and expert environmental professionals. The book will be particularly useful to those scientists who have postponed the learning and using mathematical software. This book will open a new world to them!

Paolo Zannetti
President, The EnviroComp Institute
Editor of Book Series on Environmental Modeling

Foreword

The book has two aims:

A. to introduce basic concepts of environmental modeling and
B. to exercise the application of current mathematical software packages.

To the target group belong all natural scientists who are dealing with the environment: engineers from process and chemical engineering, physicists, chemists, biologists, biochemists, hydrogeologists, geochemists, ecologists. . .!

As the book is concerned with modeling, it inevitably demands some mathematical insight. The book is designed to

1. be a door opener to the field for novices without any background knowledge of environmental modeling and of MATLAB® , and
2. to surprise those, who have some expertise, with advanced methods which they have not been aware of.

For this book MATLAB® was chosen as the computer tool for modeling, because

i. it is powerful, and
ii. it is available at most academic institutions, at all universities and at the research departments of companies.

Other mathematical products could have been selected from the market, which would perform similarly well for most application problems presented in the various chapters. But MATLAB® is rather unique in it's strong capabilities in numerical linear algebra.

There are twenty chapters in the book. The first chapters are concerned with environmental processes and their simulation: (1) transport, consisting of advection, diffusion and dispersion, (2) sorption, (3) decay or degradation, (4) reaction, either kinetic or thermodynamic. Following aim (B) there are subchapters inserted for the introduction of MATLAB® modeling techniques. The first part of the book ends with chapters on ordinary differential equations and parameter estimation (inverse modeling).

The second part of the book starts with chapters on flow modeling. Flow, if present, is an important, but mostly also complex part within an environmental compartment. Core MATLAB® allows simple flow set-ups only. Therefore the focus is on potential flow, which has applications in hydro (water) and aero (air) -dynamics as well as in porous media (seepage and groundwater). Concepts of MATLAB® are deepened within these chapters. At the very end special topics appear: image processing and geo-referencing, graphs, linear systems, the phase space and graphical user interfaces.

Berlin, December 2006

Acknowledgements

A book on such a wide topic as this one includes experiences and knowledge shared with lots of other people. I ask for understanding that I mention no names here. To be fair, giving one name would make it necessary to add numerous others. As one and only exception I want to mention my wife Susanne for reading the entire book, correcting stylistic errors, providing improved formulations and for mental support.

Thanks to Humboldt University Berlin (HUB), the Leibniz Institute of Freshwater Ecology and Inland Fisheries (IGB), to the Weierstrass Institute for Applied Analysis and Stochastics (WIAS), and to Freie Universität Berlin (FUB). Special thanks to the students who attended the modeling courses at FUB, for their questions and for their ideas. Without them I would not have become so familiar with the software. Parts of the book are based on seminar scripts.

Especially I want to thank The MathWorks, Inc (http://www.mathworks.com/) for providing the most recent versions of the MATLAB® software as well as other support by including the book in the MathWorks Book Program.

Contents

De groei van de reken – en geheugencapaciteit van computers is nog steeds indrukwekkend, maar de spectaculaire ontwickelingen zien we momenteel toch gebeuren of het gebiet van de software. Geladen met moderne wiskundprogramma's zijn computers allang geen domme nummerkrakers meer, en de toekomst ligt weer helemaal open voor methoden die een groter appel doen doen op de menselijke geest. In dit artikel willen we ... laten zien hoe analytische formules ... hanteerbaaar worden, door gebruik te maken van – het klinkt tegenstrijdig – het numerieke wiskundepakket MATLAB.

(Maas K./ Olsthoorn T., Snelle oudjes gaan MATLAB, Stromingen, Vol. 3, No. 4, 21–42, 1997; in Dutch)

The growth of performance and storage capacity of computers is still impressive, but we see the most spectacular development in the field of software. Equipped with modern mathematical packages computers are not stupid number crunchers any more, and the future lies again wide open for methods, which appeal more to the human mind. In this contribution we show . . . how analytical formulae become manageable by using – it sounds contradictory - the numerical mathematical tool MATLAB®.

(translated by E.H)

Part I

Primer to Modeling with MATLAB®

1

Introduction

1.1 Environmental Modeling using MATLAB®

There are various types of models in the environmental sciences, and surely there is no unique opinion about the essence of an environmental model. Differences may mainly concern the scope of the models and the modeling methods. Concerning the scope, this book is relatively open; i.e. examples from different branches of environmental science and technology are included, mainly from the hydrosphere and the geosphere, and also from the biosphere and the atmosphere. However, the examples are selected for demonstration purposes and can in no way represent the vast variety of phenomena and approaches, which can be met in publications and studies of all types of environmental systems.

Concerning the methods, the book does not represent the entire field either. In this book modeling is *process-oriented* and *deterministic*. These two terms characterize almost all presented methods, which, according to many opinions, represent the most important approach to understand environmental systems. There are environmental problems, for which other approaches not tackled here work more successfully. Statistical or stochastic methods are not mentioned, for example. Data processing, either graphical or numerical, as for example in Geo-Information Systems (GIS), appears rudimentary in this book.

Processes are in the focus of the presented approach. In the modeling concept of this book processes can be of physical, chemical or biological nature. The reproduction of biological species is a process, death is another; degradation of biochemical species, or decay of radioactive species are other examples. Some relevant processes are explained in detail: diffusion, dispersion, advection, sorption, reactions, kinetic and/or thermodynamic and others.

A view into journal or book publications shows that models of the treated kind, process-oriented and deterministic, are applied to different environmental compartments, to different phases and to different scales, as well as to multi-phase and multi-scale problems. There are models of the entire globe,

of earth atmosphere and oceans, of the global atmosphere, of the sea, of rivers, lakes and glaciers, of watersheds, of the soil, of terrestrial or aquatic sediments, of aquifers, and of parts of streams, and so forth. There are models of technical devices for environmental purposes, in addition. Experimental set-ups in laboratories are simulated in order to understand relevant processes.

The methods presented in this book are deterministic, throughout. A search for any statistics would be in vain. The description of processes is translated into mathematical terms. Often the approach leads to differential equations, which are conditions concerning the change of a variable, like concentration or population density, in space and time. Nowadays the solution of such equations is not as tedious as in former times. Using core MATLAB®, problems in 0 and 1 space dimensions can be solved comfortably. Core MATLAB® is also convenient for solving 2 and 3-dimensional problems with analytical solutions. For more complex modeling in more than one dimension, toolboxes, especially the MATLAB® partial differential toolbox, can be recommended.

The aim of the book is to introduce basic concepts of environmental modeling. Starting from basic concepts the problems are transformed into mathematical formulations. Strategies for the solution of the mathematical problems on the computer are outlined. The main aim of the book is to communicate the entire path of such a modeling approach. At some points algorithmic details will be omitted for the general aim. Who is interested strictly in computer algorithms, will be better served with a book on numerics, applied mathematics or computational methods. It is important that the modeler has a basic understanding of the underlying numerics. There is no need, however, to dive so deep into the algorithms that one would be able to program them oneself. In fact, it is an advantage of the chosen software that modeling tasks, which could be handled only by people with profound programming knowledge and skills, become now available to a wider audience.

Who is addressed? In a broader sense everyone is addressed, who is dealing with or is interested in the simulation of environmental systems on a computer. In a considerable part of the book concepts of environmental modeling are introduced, starting from basic principles, tackling differential equations and numerical solutions. In another similarly big part of the book special implementations are introduced and described. If someone is very familiar with another mathematical software, the book may be of help too, as most of the described models can also be realized using other maths computer programs.

There are several good and excellent books on environmental modeling and on MATLAB®. Richter (1985) deals with ecological systems and with time dependencies (but no space dependencies), as well as Deaton & Winebrake (1999) using STELLA®[1]. Shampine et al. (2003) also present MATLAB® modeling of ordinary differential equations; concerning applications they do not address environmental modeling particularly; concerning

[1] See: http://www.iseesystems.com/

methods, they do not address partial differential equations. Gander & Hrebicek (1997) offer little to the specialized environmental modeler, although some of the presented mathematical tools could be applied to environmental problems. Christakos et al. (2002) focus on the connection of time dependent simulation and GIS using MATLAB®. McCuen (2002) treats statistical methods (which do not appear here) for modeling hydrologic change. Cantrell & Cosner (2004) examine spatial ecology via reaction-diffusion models, without reference to any specific software package. Lynch (2005) addresses scientists and engineers in his general introduction to numerical methods without preference for any specific software and with few references to applied environmental modeling. In his introduction to MATLAB® Kiusalaas (2005) addresses engineers in general. The topic of Zimmerman (2004) is chemical process simulation using FEMLAB[2] code. Hornberger & Wiberg (2005) have the hydrologist's perspective on numerical methods. Trauth (2006) focuses on image- and data-processing, as well as statistical methods for geoscientists. Finlayson (2006) deals with the chemical aspects and gives an introduction to MATLAB® as one of several modeling tools. All these books[3] differ concerning scope and methods; and none of them has the same constellation of scope and methods, as it is presented in this book.

The book is divided into twenty chapters which differ concerning scope and complexity. The first ten chapters form a primer on fundamental concepts and basic environmental modeling. All of the model examples presented are 0- or 1-dimensional. In the further ten chapters more complex models, as for example spatial 2D, are outlined with an explanation of the underlying methods. Concepts of flow modeling are introduced.

In this book the focus on basic '*core*' MATLAB®[4] is intended. There is the hope to address a wider audience, as not all readers may have access to the complete palette of MATLAB® toolboxes. On the other hand, there are lots of powerful commands in core MATLAB® and novice users might be confused being confronted with more specialized tools. It turns out that this is not a severe restriction, as most basic tasks, which are of interest to the environmental modeler, can be performed using core MATLAB®. For advanced higher dimensional and coupled problems the MATLAB® partial differential equation toolbox has to be used, or COMSOL alternatively. COMSOL has developed a multi-physics software environment, which can be applied with MATLAB® in the background, and which is also known under the former name FEMLAB.

Although other mathematical codes have developed a similar extension from a special purpose module to a toolbox for mathematical calculations

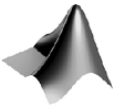
[2] Now COMSOL; see: http://www.comsol.com/

[3] There are numerous other books on MATLAB®, which could not all be checked by the author. The reader can get a list on the MathWorks Website http://www.mathworks.com/support/books

[4] For this book MATLAB version 7, release 14 was mainly used. Thanks to MathWorks for providing access to the most recent MATLAB® versions during the work on the book.

in general, matrix manipulation is the backbone and stronghold of the MATLAB® package and explains its strong competitiveness. Therefore subchapter 1.2 gives a brief reminder of basic matrix operations.

The book is accompanied by a CD-ROM containing advanced and final versions of the program files described in the text. The Mathworks logo



appears where MATLAB® files of the CD-ROM are referenced.

The terms 'modeling' and 'simulation' are synonymously in the concerned scientific and technical literature. However, the term 'model' appears to be more general, encompassing all types of attempts to capture one or more aspects of a real system, and is therefore preferred in this book. The term 'simulation' also fits to the presented approach, as it suggests that processes which are relevant for the behavior of a system are included in the computer simulation. In the sequel the term is used for time-dependent dynamics.

The book contains relatively simple models throughout. It is not the case that complex models constructed by MATLAB® don't exist, but they are not appropriate for an introduction into modeling techniques. For such an aim models should be as simple as possible, even more, when novice modelers are addressed.

Usually the extensive work with a model leads to renewed extensions, which turn simple models into complex ones almost as a rule. Not all models are improved by doing this. Jørgensen (1994) envisages the connection between model complexity and knowledge, gained by the model, as shown in Fig. 1.1. Simple models can be improved by extensions, but there is a certain peak position after which further extensions do not add to the knowledge – rather quite the contrary. An improved model design increases the quality of the model (lets take gained knowledge as a quality measure), but further extensions of the improved model may finally lead to a situation in which the increase of model complexity is counter-productive.

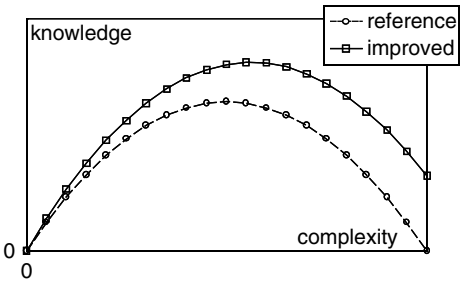


Fig. 1.1. Model evaluation: knowledge gained vs. complexity

The model evaluation study of Constanza & Sklar (1985) provides a plot similar to Fig. 1.1, but with 'articulation' on the x-axis and 'effectiveness' on the y-axis. Chwif & Barretto (2000) envisage a similar picture, putting 'level of detail' on the x-axis and 'model confidence' on the y-axis. All these terms can be taken as different terms for the complexity of a model on one side and its performance on the other side.

The method, how to construct complex models, is another topic which is left out in the book. The major drawback of complex models is the increased number of parameters, sometimes to a drastical extend. The situation may be worsened by the fact that many new parameters are usually difficult to obtain or have to be determined by parameter estimation runs with the model. Another drawback may appear, if the model becomes very sensitive to one or more parameters, i.e. that relatively small changes of a parameter induce a tremendous effect on the output results. A complex model which depends sensitively on numerous unknown parameters can surely not be used as a predictive tool.

However, complex models have their justification. Whether they can be successful also depends on the architecture, design and construction itself, especially on the analytical and/or numerical techniques.

A complex model concerning sediment phosphorus and nitrogen processes is presented by Harper (2000): the SNAPP model is constructed in MATLAB® and contains even a graphical user interface. As another example Luff et al. (2001) present a MATLAB library to calculate pH distributions in marine systems. Kumblad et al. (2003) construct an ecosystem model of the environmental transport and fate of carbon-14 in a bay of the Baltic Sea, just to give another example. A complex MATLAB® surface fluid flow model for rivers, streams and estuaries is presented by Martin & Gorelick (2005).

It is not the aim of modeling to set up complex models. The opposite of that statement is a more suitable goal: the aim of modeling is to find simple models that explain some aspects of a real system. Unfortunately that aim turns out to be a tricky one, because every real system appears to be complex, as long as there is ample knowledge about the driving mechanisms. Moreover, if a system is complex, a simple model can explain a few aspects at the most and that may nor be enough to solve a real problem.

1.2 Introduction to MATLAB®

MATLAB® is a mathematical software, originated and mainly developed by mathematicians (Moler 2004). The name envisages a laboratory for matrix calculations, where the mathematical term of a matrix refers to an array of numbers such as

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} \tag{1.1}$$

Linear algebra is the name of the mathematical field in which calculations with matrices are treated. Some basic terms are listed in the appendix of this chapter.

While MATLAB® was designed for numerical linear algebra in the beginning, it has become a tool for all types of mathematical calculations in the meantime. Nowadays, MATLAB® has been applied in nearly every field of scientific or technical calculations. In the academic branch there is almost no university where MATLAB® is not available.

With MATLAB® innumerable types of mathematical operations can be performed. Of course, numerous linear algebra calculations are available, such as inversion of matrices, eigenvalue and eigenvector determination, which can be applied to perform various tasks, for example, the solution of systems of linear equations. One may perform basic statistics, numerical differentiation and integration, evaluate all types of functions, solve dynamical systems and partial differential equations, estimate parameters and so forth. All this is part of core MATLAB®, a collection of basic mathematical tools[5].

Before some details of linear algebra are examined, an introduction into the work with MATLAB® is necessary. This should be read by novices, but can be skipped by those who have already worked with the program.

Getting Started with MATLAB®

When MATLAB® is opened, the user obtains a graphical user interface on the display, as it is shown in Fig. 1.2, containing several windows. The main window, to start with, is the 'Command Window', where commands are given and answered. In the command window the MATLAB® prompt '>>' stands at the position where the user command is shown on the display, during and after entering.

In order to start type the command:

```
a = 2
```

Press the return button and the program gives an answer, here with the information that a variable `a` was created in the machine containing the value `2`:

```
a =
     2
```

A new prompt appears after the answer of the system, in order to enable the user to give the next command. Note that only the line after the last prompt in the command window can be used for a new command. The former lines remain in the command window to allow the user to have an overview

[5] Core MATLAB® can be extended by numerous toolboxes for special purposes, for details see: http://www.mathworks.com/products/product_listing; most interesting for environmental modeling, as it is tackled here, are the optimization toolbox and the partial differential equations toolbox.

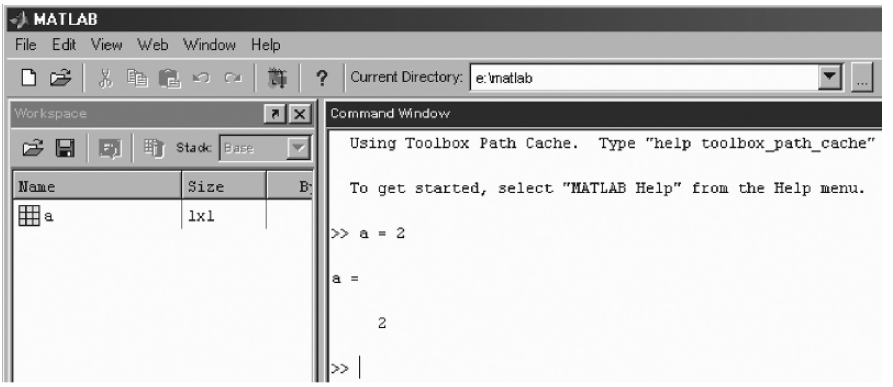


Fig. 1.2. Appearance of MATLAB® graphical user interface

on the previous work and the produced answers. Confirm that the ; as closing character of the command, for example

```
a = 2;
```

prevents that the answer is shown in the command window.

The command window is good for an introduction into MATLAB®. Finally, the work with M-files replaces extensive operating in the command window (see Chap. 2.5). Nevertheless, for certain tasks, the command window will remain the most direct and simple way to compute with MATLAB®.

Aside from the command window, the user may select numerous other views of the desktop. The different options are depicted in Fig. 1.3. Very important is the workspace view, where all variables of the current session are visible and directly available. The workspace of the just started session, shown in Fig. 1.2, is depicted on the left side of the figure. The workspace appears only if the view is selected in the 'Desktop' submenu, as shown in Fig. 1.3. Using `who` or `whos` in the command window is an alternative way to access the workspace (and its contents).

Here, `a` is the only variable in the workspace which is of 'double' type and of 1×1 size (a single variable and not a 'real' matrix). A double-click on the block-panel symbol, left of the variable name in the workspace, delivers an array editor, in which the contents of variables can be viewed directly. In the simple example case the result is given in Fig. 1.4. With the array editor it is not only possible to view variables, but also to change them. The user can easily explore the use of the editor on her/his own.

To mention is the 'command history' view, in which all commands are listed. An example with one command only is listed in Fig. 1.5. The user can initiate the repeated command, mostly with some workspace variables changed, by double-click in the command history window. It is an alternative method to copy a former command in the history view and to paste it

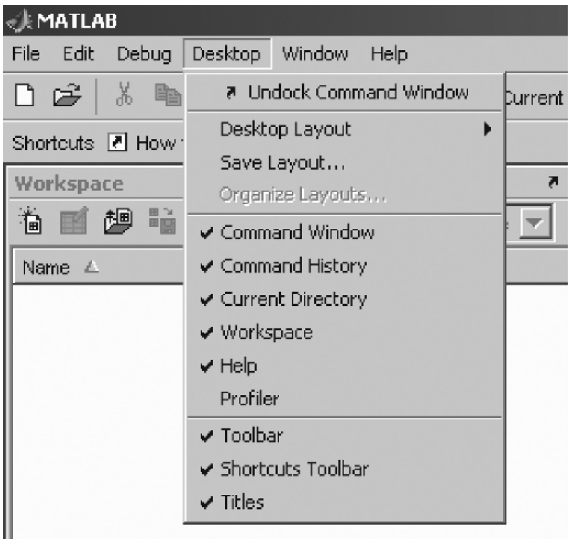


Fig. 1.3. Submenu-entries of desktop main entry, listing all possible views of the desktop

in the command window. The user may wish to perform some alterations in the command and can do that easily, before the command is executed after pressing the return button. The up-arrow and down-arrow keys of the keyboards offer an alternative, allowing a sequential loop through former commands.

Matrices in MATLAB®

The name 'MATLAB' is a combination of 'matrix' and 'laboratory'. With respect to the suite of various mathematical tools, which are made available by recent versions of the software, one might think the origin of the MATLAB® software is numerical linear algebra.

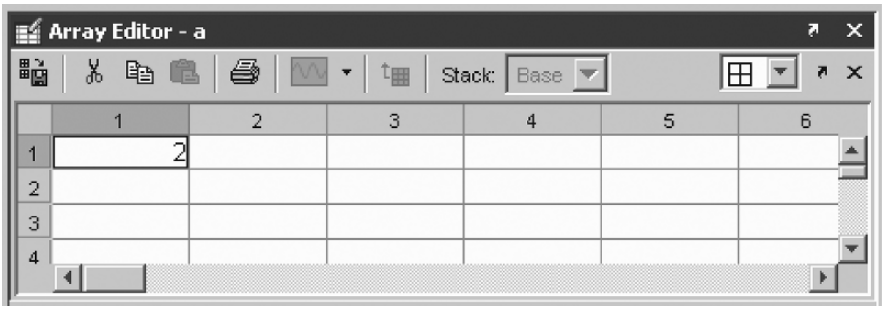


Fig. 1.4. MATLAB® array editor

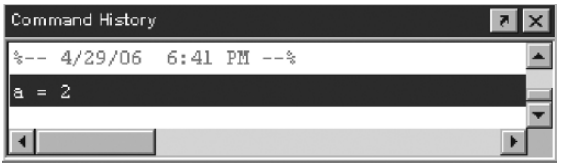


Fig. 1.5. MATLAB® command history window

A *matrix* is a 2-dimensional array of numbers, for which examples are given right below. Matrices can be specified directly by the user. Entries in lines are separated by blanks; lines are separated by ';'.

```
A = [1 2 3; 4 5 6]

A =
    1    2    3
    4    5    6
```

The example matrix has 2 rows and 3 columns. Matrix dimensions are 2 and 3. `A` is a 2×3 matrix. It is thus non-square, as a *square matrix* has the same number of rows and columns. Once a matrix is constructed, its elements can be called by using usual round brackets, which is exemplified by:

```
A(2,1)

ans =
       4
```

The element in the second row and first column of **A** is 4. As no variable is used in the command, MATLAB® uses the notation `ans =` in order to indicate an answer to the given command. Sub-matrices of a matrix can be called by using '`:`', as the following example illustrates:

```
A(2,2:3)

ans =
      5  6
```

Elements in the second line, second and third column are given in the answer. The '`:`' without any numbers is used to indicate the entire range. In the example, the entire first column of **A** is given

```
A(:,1)

ans =
       1
       4
```

There are several special commands to input special types of matrices. *Vectors* are multi-element matrices, for which either the number of rows or the number of columns is 1. Row vectors with constant increment can be specified as follows:

```
v = [2:0.5:5]

v =
    2.0000  2.5000  3.0000  3.5000  4.0000  4.5000  5.0000
```

v is a row vector, containing all values between 2 and 5 with increment 0.5[6]. A column can easily be obtained by using the *transponation* operation, which in MATLAB® is performed by the ':

```
v'

ans =
      2.0000
      2.5000
      3.0000
      3.5000
      4.0000
      4.5000
      5.0000
```

Matrices containing 1s are given by:

```
B = ones(2,3)

B =
    1    1    1
    1    1    1
```

Matrices containing zeros are produced analogously:

```
B = zeros(3,1)

B =
    0
    0
    0
```

How matrices containing a constant, different from 0 and 1, can be obtained easily, is demonstrated by the following command:

```
C = 4.5*ones(3,5)

C =
    4.5000    4.5000    4.5000    4.5000    4.5000
    4.5000    4.5000    4.5000    4.5000    4.5000
    4.5000    4.5000    4.5000    4.5000    4.5000
```

The `ones`–matrix is multiplied by a single value, a so called scalar, here 4.5. The `*` stands for multiplication. As will be explained in more details in the next sub-chapter, there are several multiplication operations in linear algebra and in MATLAB®. In the previous command line the `*` stands for *scalar*

[6] The *comma* in common numbers is a *dot* in all mathematics software products, thus also in MATLAB®.

multiplication, where all elements of the matrix are multiplied by the same scalar value.

The command for random matrices is

```
C = rand(2,5)
```

```
C =
    0.9501    0.6068    0.8913    0.4565    0.8214
    0.2311    0.4860    0.7621    0.0185    0.4447
```

Random values between 0 and 1 are entries of the matrix. If there is only one integer argument in the preceding matrix types, a square matrix results:

```
D = rand(2)
```

```
D =
    0.9501  0.6068
    0.2311  0.4860
```

As mentioned above matrices are 2-dimensional arrays. Single numbers can be regarded as 1-dimensional arrays. MATLAB® can, of course, handle arrays of higher dimensions. We demonstrate this by introducing the `randn` command:

```
E = randn(2,4,2)
```

```
E(:,:,1) =
    0.0000    1.0950    0.4282    0.7310
   -0.3179   -1.8740    0.8956    0.5779
E(:,:,2) =
    0.0403    0.5689   -0.3775   -1.4751
    0.6771   -0.2556   -0.2959   -0.2340
```

which is a 3-dimensional array of random numbers with mean value $\mu = 0$ and standard deviation $\sigma = 1$. In the same manner, all previous matrix generating commands can be applied to obtain higher dimensional arrays if the number of arguments in the call exceeds 2. Multi-dimensional arrays can be viewed using the array editor, but they cannot be edited within the editor. In order to do this, address single elements from the command window, or specify 2-dimensional sub-arrays:

```
E1 = E(:,:,2)
```

and edit those. Last but not least, lets mention that MATLAB® has the empty matrix as zero-dimensional array:

```
e = [ ]
```

Basic Matrix Operations

It is expected that readers are already familiar with matrix operations and basics of linear algebra. The purpose of the following is (1) to be a reminder for those, to whom matrices are not (yet) part of daily practice and (2) to introduce the notation used in the following chapters of this book.

A matrix is a 2-dimensional array of numbers. A matrix has a certain number of lines and columns, and the single numbers in the matrix are called elements (sometimes the term 'entries' is used here as alternative). The matrix in (1.1) has 2 lines and 2 columns, and the element in the second line and first column is 3. A single number can be conceived as special case of a matrix with one line and one column. Thus matrix algebra is a generalization of the usual calculations with single numbers. However, in order to distinguish 'real' arrays from single numbers, bold letters are used for matrices and vectors.[7]

Basic operations as known from single numbers can be generalized for matrices. Matrices can be added. The sum of the matrices $\mathbf{A}$ and $\mathbf{B}$

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ b_{n1} & b_{n2} & \cdots & b_{nm} \end{pmatrix} \tag{1.2}$$

is given by:

$$\mathbf{A} + \mathbf{B} = \begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1m} + b_{1m} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2m} + b_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} + b_{n1} & a_{n2} + b_{n2} & \cdots & a_{nm} + b_{nm} \end{pmatrix} \tag{1.3}$$

In order to add two matrices, both need to have the same number of lines and columns. In each element of the matrix $\mathbf{A}$+$\mathbf{B}$, the sum of the corresponding elements of $\mathbf{A}$ and $\mathbf{B}$ appears. One may also say that in order to obtain the element in the i-th row and j-th column of $\mathbf{A}$+$\mathbf{B}$, the elements in the i-th row and j-th column of $\mathbf{A}$ and $\mathbf{B}$ have to be added:

$$(\mathbf{A} + \mathbf{B})_{ij} = a_{ij} + b_{ij} \tag{1.4}$$

Example in MATLAB®:

```
A = [1 2; 3 4]; B = [-1 0; 1 2]; C = A+B

C =
    0    2
    4    6
```

When the number of columns or the number of lines do not coincide, MATLAB® produces an error:

```
D = [5 6 7 8];
A+D
??? Error using ==> +
Matrix dimensions must agree.
```

[7] A vector is a matrix consisting of one line or one column only. Terms as line-vectors or column-vectors are used, too.

Clearly the subtraction of matrices is defined analogously. One may also formally introduce subtraction by the definition that subtraction of $\mathbf{B}$ is the addition of $-\mathbf{B}$. As one may expect $-\mathbf{B}$ contains the negative of the elements of $\mathbf{B}$ and is the inverse of $\mathbf{B}$ with respect to the addition operation. The generalizations of matrix multiplication and division are slightly more complex.

It was already mentioned that there are several multiplication operations. Correspondingly there are several division operations. Aside from scalar multiplication, there are several matrix multiplications. The standard matrix multiplication for the two matrices $\mathbf{A}$ and $\mathbf{B}$, given by

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1k} \\ a_{21} & a_{22} & \cdots & a_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nk} \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \cdots & b_{1m} \\ b_{21} & b_{22} & \cdots & b_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ b_{k1} & b_{k2} & \cdots & b_{km} \end{pmatrix} \tag{1.5}$$

in order to obtain a new matrix $\mathbf{A} \cdot \mathbf{B}$, is defined by the following formula:

$$(\mathbf{A} \cdot \mathbf{B})_{ij} = \sum_{l=1}^{k} a_{ik} b_{kj} \tag{1.6}$$

This is a formula for the element in the i-th row and j-th column of the matrix $\mathbf{AB}$. Matrices can be multiplied if the first matrix has the same number of columns as the second matrix has columns (inner dimension). In formula (1.6) that number is k. Elements in lines of the first matrix are multiplied with columns of the second matrix, and the products are summed in order to obtain an entry in the result matrix $\mathbf{A} \cdot \mathbf{B}$.

Example in MATLAB:

```
C = A*B

C =
    1    4
    1    8
```

If the inner dimensions of the matrices do not agree an error message results. Matrix multiplication is a generalization of the multiplication of single numbers. Clearly, if the product $\mathbf{A} \cdot \mathbf{B}$ is possible, the product $\mathbf{B} \cdot \mathbf{A}$ is only possible if $\mathbf{A}$ and $\mathbf{B}$ are square matrices. Even then the identity $\mathbf{A} \cdot \mathbf{B} = \mathbf{B} \cdot \mathbf{A}$ is not valid generally (see exercises below).

The multiplication, described by formula (1.6), is the standard multiplication of matrices, denoted by a '$\cdot$'-dot in the formulae and by a * in MATLAB® commands. Analogously to the definition of addition, given in (1.4), there exists also an element-wise multiplication:

$$(\mathbf{A} \cdot \mathbf{B})_{ij} = a_{ij} b_{ij} \tag{1.7}$$

In order to perform this multiplication, matrices $\mathbf{A}$ and $\mathbf{B}$ need to have the same number of rows and columns. In formulae element-wise multiplication is

denoted by .* in MATLAB® commands, distinguishing element-wise operation from the standard matrix multiplication. In formulae we use the '$\cdot$'-dot or omit the operator symbol entirely. There are scalar multiplication and vector product as further operations which are explained below.

Division of matrices can be defined for both multiplications. To start with the simple case: element-wise division is performed with element values. In MATLAB® element-wise division is denoted by ./. Element-wise division with the same matrix delivers a matrix containing 1 in each entry, which is the unit matrix with respect to element-wise multiplication.

Example in MATLAB®:

```
C = A./B

Warning: Divide by zero.
(Type ''warning off MATLAB:divideByZero'' to suppress this
warning.)
C =
    -1   Inf
     3     2
```

Obviously, in three entries the element-wise division is performed. In the second entry of the first row `Inf` stands for *infinity*, which is the result of a division by zero[8].

Example in MATLAB®:

```
C = A./A

C =
    1    1
    1    1
```

The matrix with entries 1 everywhere is the unit matrix of pointwise matrix multiplication.

The unit matrix with respect to matrix multiplication is given by:

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 \end{pmatrix} \tag{1.8}$$

This matrix is a diagonal matrix, as there are non-zero elements only in the main diagonal from the top left to the bottom right. The unit matrix within the matrix algebra corresponds to the 1 in usual multiplications using

[8] In contrast to school knowledge, division by zero is allowed in MATLAB®. The result is infinity. MATLAB® shows a warning (but no error) in order to remind the user that such an operation may result in some errors in further operations.