Martin H. Trauth

MATLAB® Recipes for Earth Sciences

Third Edition

Martin H. Trauth

MATLAB® Recipes for Earth Sciences

Third Edition

With Contributions by
Robin Gebbers and Norbert Marwan
and illustrations by Elisabeth Sillmann

Springer

Privatdozent Dr. rer. nat. habil. Martin H. Trauth
University of Potsdam
Department of Earth and Environmental Sciences
Karl-Liebknecht-Str. 24
14476 Potsdam
Germany
trauth@geo.uni-potsdam.de

ISBN 978-3-642-12761-8 e-ISBN 978-3-642-12762-5
DOI 10.1007/978-3-642-12762-5
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2010930277

© Springer-Verlag Berlin Heidelberg 2006, 2007, 2010
This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilm or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.
The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting and book design by Elisabeth Sillmann, www.blaetterwaldDesign.de Landau, Germany

Cover design: deblik, Berlin

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The book *MATLAB Recipes for Earth Sciences* is designed to help undergraduates, and PhD students, post-doctoral researchers, and professionals find quick solutions for common problems in data analysis in earth sciences. It provides a minimum amount of theoretical background, and demonstrates the application of all described methods through the use of examples. The MATLAB software is used since it not only provides numerous ready-to-use algorithms for most methods of data analysis but also allows the existing routines to be modified and expanded, or new software to be developed. The book contains MATLAB scripts, or *M-files*, to solve typical problems in earth sciences, such as simple statistics, time-series analysis, geostatistics, and image processing, and also demonstrates the application of selected advanced techniques of data analysis such as nonlinear time-series analysis, adaptive filtering, bootstrapping, and terrain analysis. It comes with a compact disk that contains all MATLAB recipes and example data files as well as presentation files for instructors. The MATLAB codes can be easily modified for application to the reader's data and projects.

This revised and updated Third Edition includes new sections on software-related issues (Sections 2.4, 2.5, 2.8 and 2.9). Chapter 2 was difficult to update since MATLAB has expanded so much over the years, and I have deliberately tried to restrict this chapter to demonstrating of those tools actually used in the book. A second difficulty arose from the current move by *The MathWorks Inc.* to use and incorporate objects and classes in some areas of their MATLAB routines, although there does not seem to be any intention of abandoning the existing procedural code. Again, I have restricted the introduction and use of objects and classes to the absolute minimum, even at the expense of omitting one of the new features of MATLAB. Some functions, however, such as those for distribution fitting use this new concept of object-oriented programming, and I hope that the reader will forgive me for not explaining all the details of the MATLAB code when using it. The other new sections are on distribution fitting (Section 3.9), and on nonlinear and weighted regression (Section 4.10), as these techniques are widely used in, for instance, isotope geochemistry and geochronology. Sections 8.7

to 8.9 introduce some advanced methods in image analysis such the extraction of color-intensity transects from laminated sediments, automatic grain size analysis, and the quantification of charcoal in microscope images. These techniques are frequently used in my research projects and are always in demand during the short courses that I teach.

In order to derive the maximum benefit from this book the reader will need to have access to the MATLAB software and be able to execute the recipes while reading the book. The MATLAB recipes display various graphs on the screen that are not shown in the printed book. The tutorial-style book does, however, contain numerous figures making it possible to go through the text without actually running MATLAB on a computer. I have developed the recipes using MATLAB 7 Release R2010a, but most of them will also work with earlier software releases. While undergraduates participating in a course on data analysis might go through the entire book, the more experienced reader may use only one particular method to solve a specific problem. The concept of the book and the contents of its chapters are therefore outlined below, in order to make it easier to use for readers with a variety of different requirements.

- *Chapter 1* – This chapter introduces some fundamental concepts of samples and populations. It also links the various types of data, and questions to be answered from the data, to the methods described in the succeeding chapters.

- *Chapter 2* – A tutorial-style introduction to MATLAB designed for earth scientists. Readers already familiar with the software are advised to proceed directly to the succeeding chapters. The Third Edition now includes new sections on data structures and classes of objects, on generating M-files to regenerate graphs and on publishing M-files.

- *Chapters 3 and 4* – Fundamentals in univariate and bivariate statistics. These two chapters contain basic concepts in statistics, and also introduces advanced topics such as resampling schemes and cross validation. The reader already familiar with basic statistics might skip these two chapters. The Third Edition now includes new sections on fitting normal distributions to observations and on nonlinear and weighted regression analysis.

- *Chapters 5 and 6* – Readers who wish to work with time series are recommended to read both of these chapters. Time-series analysis and signal processing are closely linked. A good knowledge of statistics is required

to work successfully with these methods. These two chapters are independent of the preceding chapters.

- *Chapters 7 and 8* – I recommend reading through both of these chapters since the processing methods used for spatial data and for images have much in common. Moreover, spatial data and images are often combined in earth sciences, for instance when projecting satellite images onto digital elevation models. The Third Edition now includes new sections on color-intensity transects of laminated sediments, automated grain size analysis from photos and quantifying charcoal in microscope images.

- *Chapter 9* – Data sets in earth sciences often have many variables and many data points. Multivariate methods are applied to a great variety of large data sets, including satellite imagery. Any reader particularly interested in multivariate methods is advised to read Chapters 3 and 4 before proceeding to this chapter.

- *Chapter 10* – Methods to analyze circular and spherical data are widely used in earth sciences. Structural geologists measure and analyze the orientation of slickensides (or striae) on a fault plane. The statistical analysis of circular data is also used in paleomagnetic applications. Microstructural investigations include the analysis of the grain shapes and quartz c-axis orientations in thin sections.

While the book *MATLAB Recipes for Earth Sciences* is about data analysis it does not attempt to cover modeling. For this subject, I recommend the excellent book *Environmental Modeling Using MATLAB* by Ekkehard Holzbecher (Springer 2007), which first introduces basic concepts of modeling and then provides a great overview of modeling examples using MATLAB. Holzbecher's book uses a very similar concept to *MATLAB Recipes for Earth Sciences* as it gives a brief introduction to the theory, and then explains MATLAB examples. Neither book provides a complete introduction to all available techniques, but they both provide a quick overview of basic concepts for data analysis and modeling in earth sciences. One of the few critical reviewers of the First Edition of *MATLAB Recipes for Earth Sciences* raised the question of why I had not included a chapter on finite-element and finite-difference modeling, and on solving differential equations – in his opinion a major omission in the book. However, this is far beyond of the scope of the book and my own expertise. Students and colleagues interested in this topic are directed to the book

MATLAB Guide to Finite Elements: An Interactive Approach by Peter I. Kattan (Springer 2007). While my book may be considered by some to be a little light on image processing, I have included in Chapter 8 three new sections on the analysis of sediment images. I would also strongly recommend to anyone interested in this topic the very successful book *Digital Image Processing Using MATLAB* by Gonzales, Woods and Eddins (Gatesmark Publishing 2009), for which a 2nd edition has just been published.

I have taken all other critiques quite seriously and invite all readers to also comment on the Third Edition: the book is constantly changing and evolving. As the Third Edition appears on the bookshelves I will create a new folder on the hard disk of my computer named *Fourth Edition*, where new ideas will be collected. The book has benefited from the comments of many people, in particular my contributing authors Robin Gebbers and Norbert Marwan, my colleagues Ira Ojala, Lydia Olaka, Jim Renwick, Jochen Rössler, Rolf Romer, Annette Witt, and the students Matthias Gerber, Mathis Hain, Martin Homann, Stefanie von Lonski, Oliver Rach, Marius Walter and Max Zitzmann. I very much appreciate the expertise and patience of Elisabeth Sillmann at *blaetterwaldDesign* who created the graphics and the complete page designs of the book. I am much obliged to Ed Manning for professional proofreading of the text. I also acknowledge Naomi Fernandez from the *Book Program* and Kate Fiore from *Academic Support* at *The MathWorks Inc.*, Claudia Olrogge and Annegret Schumann at *The MathWorks GmbH Deutschland*, Christian Witschel, Chris Bendall and their team at *Springer*, and Andreas Bohlen, Brunhilde Schulz and their team at *UP Transfer GmbH*. I also thank the *NASA/GSFC/METI/ERSDAC/JAROS* and the *U. S./Japan ASTER Science Team* and the director Mike Abrams for allowing me to include the ASTER images in this book.

Potsdam, April 2010

Martin Trauth

Contents

1 Data Analysis in Earth Sciences

1.1 Introduction

Earth scientists make observations and gather data about the natural processes that operate on planet Earth. They formulate and test hypotheses on the forces that have acted on a particular region to create its structure and also make predictions about future changes to the planet. All of these steps in exploring the Earth involve the acquisition and analysis of numerical data. An earth scientist therefore needs to have a firm understanding of statistical and numerical methods as well as the ability to utilize relevant computer software packages, in order to be able to analyze the acquired data.

This book introduces some of the most important methods of data analysis employed in earth sciences and illustrates their use through examples using the MATLAB® software package. These examples can then be used as recipes for the analysis of the reader's own data, after having learned their application with synthetic data. This introductory chapter deals with data acquisition (Section 1.2), the various types of data (Section 1.3) and the appropriate methods for analyzing earth science data (Section 1.4). We therefore first explore the characteristics of typical data sets and subsequently investigate the various ways of analyzing data using MATLAB.

1.2 Data Collection

Most data sets in earth sciences have a very limited sample size and also contain a significant number of uncertainties. Such data sets are typically used to describe rather large natural phenomena, such as a granite body, a large landslide or a widespread sedimentary unit. The methods described in this book aim to find a way of predicting the characteristics of a larger *population* from a much smaller *sample* (Fig. 1.1). An appropriate sampling strategy is the first step towards obtaining a good data set. The development

M.H. Trauth, *MATLAB® Recipes for Earth Sciences*, 3rd ed.,
DOI 10.1007/978-3-642-12762-5_1, © Springer-Verlag Berlin Heidelberg 2010

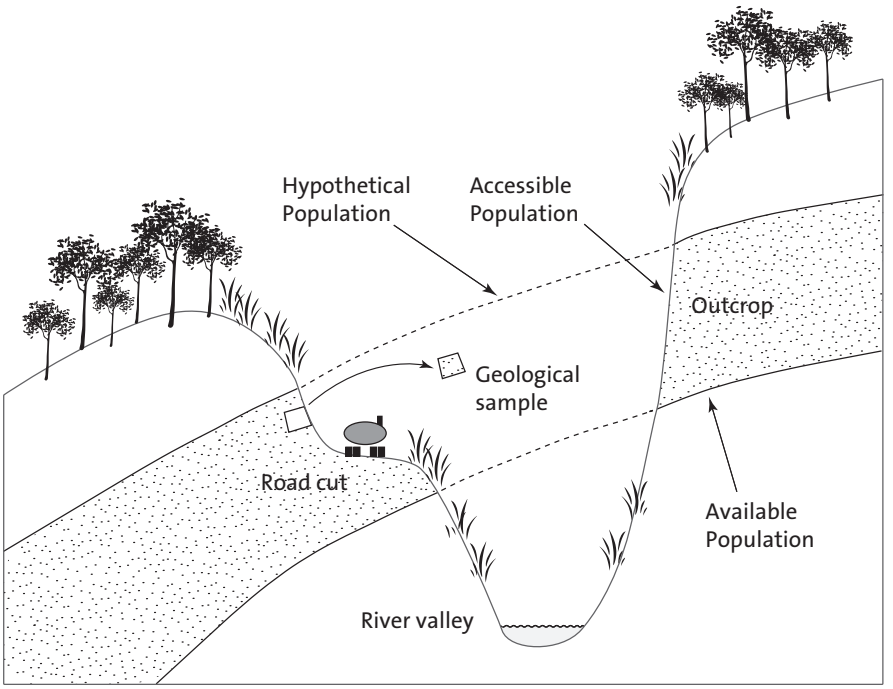


Fig. 1.1 Samples and populations. Deep valley incision has eroded parts of a sandstone unit (*hypothetical population*). The remaining sandstone (*available population*) can only be sampled from outcrops, i.e., road cuts and quarries (*accessible population*). Note the difference between a statistical sample as a representative of a population and a geological sample as a piece of rock.

of a successful strategy for field sampling requires decisions on the *sample size* and the *spatial sampling scheme*.

The sample size includes the sample volume, the sample weight and the number of samples collected in the field. The sample weights or volumes can be critical factors if the samples are later analyzed in a laboratory and most statistical methods also have a minimum requirement for the sample size. The sample size also affects the number of subsamples that can be collected from a single sample. If the population is heterogeneous then the sample needs to be large enough to represent the population's variability, but on the other hand samples should be as small as possible in order to minimize the time and costs involved in their analysis. The collection of smaller pilot samples is recommended prior to defining a suitable sample size.

The design of the spatial sampling scheme is dependent on the availabil-

ity of outcrops or other material suitable for sampling. Sampling in quarries typically leads to clustered data, whereas sampling along road cuts, shoreline cliffs or steep gorges results in one-dimensional traverse sampling schemes. A more uniform sampling pattern can be designed where there is 100 % exposure or if there are no financial limitations. A regular sampling scheme results in a gridded distribution of sample locations, whereas a uniform sampling strategy includes the random location of a sampling point within a grid square. Although these sampling schemes might be expected to provide superior methods for sampling collection, evenly-spaced sampling locations tend to miss small-scale variations in the area, such as thin mafic dykes within a granite body or the spatially-restricted occurrence of a fossil (Fig. 1.2).

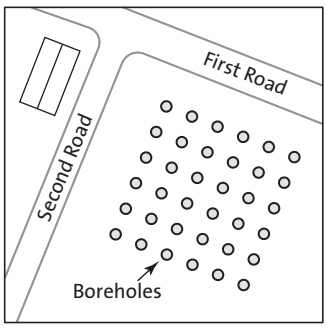
The correct sampling strategy will depend on the objectives of the investigation, the type of analysis required and the desired level of confidence in the results. Having chosen a suitable sampling strategy, the quality of the sample can be influenced by a number of factors resulting in the samples not being truly representative of the larger population. Chemical or physical alteration, contamination by other material or displacement by natural and anthropogenic processes may all result in erroneous results and interpretations. It is therefore recommended that the quality of the samples, the method of data analysis employed and the validity of the conclusions drawn from the analysis be checked at each stage of the investigation.

1.3 Types of Data

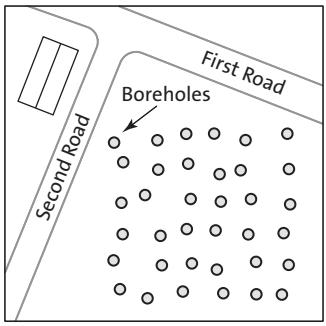
Most earth science data sets consist of numerical measurements, although some information can also be represented by a list of names such as fossils and minerals (Fig. 1.3). The available methods for data analysis may require certain types of data in earth sciences. These are

- *nominal data* – Information in earth sciences is sometimes presented as a list of names, e. g., the various fossil species collected from a limestone bed or the minerals identified in a thin section. In some studies, these data are converted into a binary representation, i. e., *one* for present and *zero* for absent. Special statistical methods are available for the analysis of such data sets.

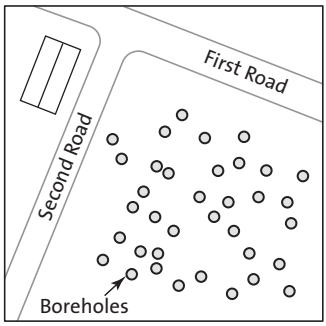
- *ordinal data* – These are numerical data representing observations that can be ranked, but in which the intervals along the scale are irregularly spaced. Mohs' hardness scale is one example of an ordinal scale. The hard-



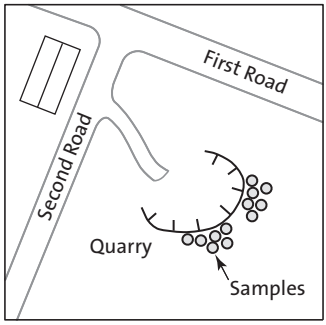
a



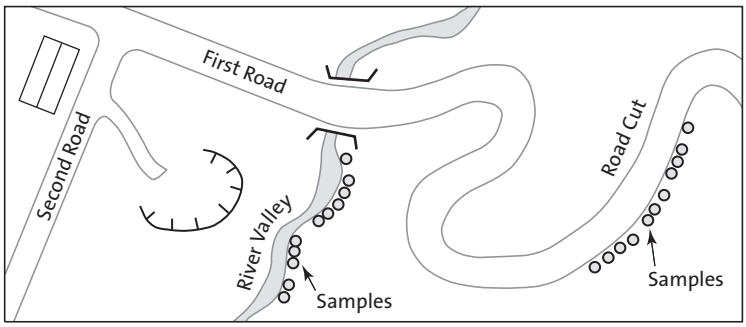
b



c



d



e

Fig. 1.2 Sampling schemes. **a** *Regular sampling* on an evenly-spaced rectangular grid, **b** *uniform sampling* by obtaining samples randomly located within regular grid squares, **c** *random sampling* using uniformly-distributed *xy* coordinates, **d** *clustered sampling* constrained by limited access in a quarry, and **e** *traverse* sampling along road cuts and river valleys.

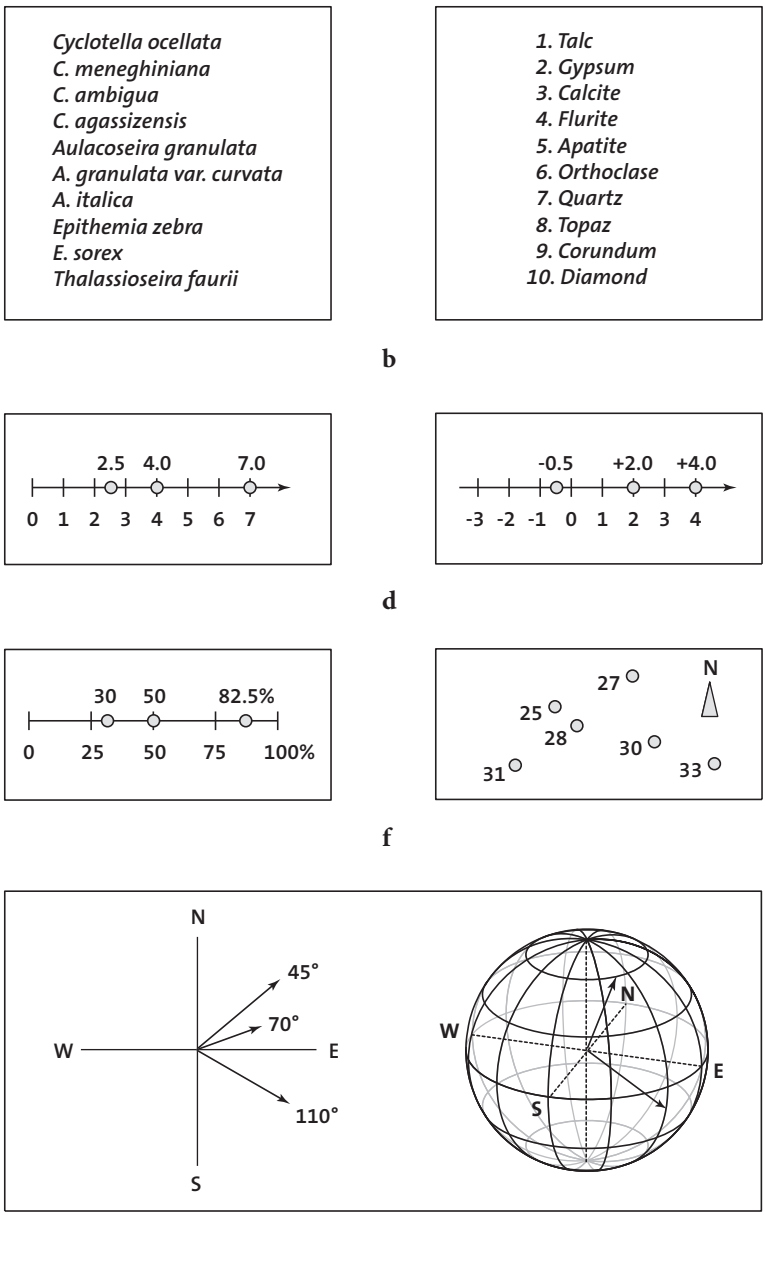


Fig. 1.3 Types of earth science data. **a** Nominal data, **b** ordinal data, **c** ratio data, **d** interval data, **e** closed data, **f** spatial data, and **g** directional data. All of these data types are described in this book.

ness value indicates the material's resistance to scratching. Diamond has a hardness of 10, whereas the value for talc is 1, but in terms of absolute hardness diamond (hardness 10) is four times harder than corundum (hardness 9) and six times harder than topaz (hardness 8). The Modified Mercalli Scale, which attempts to categorize the effects of earthquakes, is another example of an ordinal scale; it ranks earthquakes from intensity I (barely felt) to XII (total destruction).

- *ratio data* – These data are characterized by a constant length of successive intervals, therefore offering a great advantage over ordinal data. The zero point is the natural termination of the data scale, and this type of data allows for either discrete or continuous data sampling. Examples of such data sets include length or weight data.

- *interval data* – These are ordered data that have a constant length of successive intervals, but in which the data scale is not terminated by zero. Temperatures C and F represent an example of this data type even though arbitrary zero points exist for both scales. This type of data may be sampled continuously or in discrete intervals.

In addition to these standard data types, earth scientists frequently encounter special kinds of data such as

- *closed data* – These data are expressed as proportions and add up to a fixed total such as 100 percent. Compositional data represent the majority of closed data, such as element compositions of rock samples.

- *spatial data* – These are collected in a 2D or 3D study area. The spatial distribution of a certain fossil species, the spatial variation in thickness of a sandstone bed and the distribution of tracer concentrations in groundwater are examples of this type of data, which is likely to be the most important data type in earth sciences.

- *directional data* – These data are expressed in angles. Examples include the strike and dip of bedding, the orientation of elongated fossils or the flow direction of lava. This is another very common type of data in earth sciences.

Most of these different types of data require specialized methods of analysis, which are outlined in the next section.

1.4 Methods of Data Analysis

Data analysis uses precise characteristics of small samples to hypothesize about the general phenomenon of interest. Which particular method is used to analyze the data depends on the data type and the project requirements. The various methods available include:

- *Univariate methods* – Each variable is assumed to be independent of the others, and is explored individually. The data are presented as a list of numbers representing a series of points on a scaled line. Univariate statistical methods include the collection of information about the variable, such as the minimum and maximum values, the average, and the dispersion about the average. Examples are the sodium content of volcanic glass shards that have been affected by chemical weathering, or the sizes of snail shells within a sediment layer.

- *Bivariate methods* – Two variables are investigated together to detect relationships between these two parameters. For example, the correlation coefficient may be calculated to investigate whether there is a linear relationship between two variables. Alternatively, the bivariate regression analysis may be used to find an equation that describes the relationship between the two variables. An example of a bivariate plot is the *Harker Diagram*, which is one of the oldest methods of visualizing geochemical data from igneous rocks and simply plots oxides of elements against SiO_2.

- *Time-series analysis* – These methods investigate data sequences as a function of time. The time series is decomposed into a long-term trend, a systematic (periodic, cyclic, rhythmic) component and an irregular (random, stochastic) component. A widely used technique to describe cyclic components of a time series is that of spectral analysis. Examples of the application of these techniques include the investigation of cyclic climatic variations in sedimentary rocks, or the analysis of seismic data.

- *Signal processing* – This includes all techniques for manipulating a signal to minimize the effects of noise in order to correct all kinds of unwanted distortions or to separate various components of interest. It includes the design and realization of filters, and their application to the data. These methods are widely used in combination with time-series analysis, e. g., to increase the signal-to-noise ratio in climate time series, digital images or geophysical data.

- *Spatial analysis* – This is the analysis of parameters in 2D or 3D space and hence two or three of the required parameters are coordinate numbers. These methods include descriptive tools to investigate the spatial pattern of geographically distributed data. Other techniques involve spatial regression analysis to detect spatial trends. Also included are 2D and 3D interpolation techniques, which help to estimate surfaces representing the predicted continuous distribution of the variable throughout the area. Examples are drainage-system analysis, the identification of old landscape forms and lineament analysis in tectonically active regions.

- *Image processing* – The processing and analysis of images has become increasingly important in earth sciences. These methods involve importing and exporting, compressing and decompressing, and displaying images. Image processing also aims to enhance images for improved intelligibility, and to manipulate images in order to increase the signal-to-noise ratio. Advanced techniques are used to extract specific features, or analyze shapes and textures, such as for counting mineral grains or fossils in microscope images. Another important application of image processing is in the use of satellite remote sensing to map certain types of rocks, soils and vegetation, as well as other parameters such as soil moisture, rock weathering and erosion.

- *Multivariate analysis* – These methods involve the observation and analysis of more than one statistical variable at a time. Since the graphical representation of multidimensional data sets is difficult, most of these methods include dimension reduction. Multivariate methods are widely used on geochemical data, for instance in tephrochronology, where volcanic ash layers are correlated by geochemical fingerprinting of glass shards. Another important usage is in the comparison of species assemblages in ocean sediments for the reconstruction of paleoenvironments.

- *Analysis of directional data* – Methods to analyze circular and spherical data are widely used in earth sciences. Structural geologists measure and analyze the orientation of slickensides (or striae) on a fault plane, circular statistical methods are common in paleomagnetic studies, and microstructural investigations include the analysis of grain shapes and quartz c-axis orientations in thin sections.

Some of these methods of data analysis require the application of numerical methods such as interpolation techniques. While the following text

deals mainly with statistical techniques it also introduces several numerical methods commonly used in earth sciences.

Recommended Reading

Borradaile G (2003) Statistics of Earth Science Data – Their Distribution in Time, Space and Orientation. Springer, Berlin Heidelberg New York

Carr JR (1994) Numerical Analysis for the Geological Sciences. Prentice Hall, Englewood Cliffs, New Jersey

Davis JC (2002) Statistics and Data Analysis in Geology, Third Edition. John Wiley and Sons, New York

Gonzalez RC, Woods RE, Eddins SL (2009) Digital Image Processing Using MATLAB – 2nd Edition. Gatesmark Publishing, LLC

Hanneberg WC (2004) Computational Geosciences with Mathematica. Springer, Berlin Heidelberg New York

Holzbecher E (2007) Environmental Modeling using MATLAB. Springer, Berlin Heidelberg New York

Middleton GV (1999) Data Analysis in the Earth Sciences Using MATLAB. Prentice Hall, New Jersey

Press WH, Teukolsky SA, Vetterling WT, Flannery BP (2007) Numerical Recipes: The Art of Scientific Computing – 3rd Edition. Cambridge University Press, Cambridge

Swan ARH, Sandilands M (1995) Introduction to Geological Data Analysis. Blackwell Sciences, Oxford

2 Introduction to MATLAB

2.1 MATLAB in Earth Sciences

MATLAB® is a software package developed by *The MathWorks Inc.*, founded by Cleve Moler, Jack Little and Steve Bangert in 1984, which has its headquarters in Natick, Massachusetts (http://www.mathworks.com). MATLAB was designed to perform mathematical calculations, to analyze and visualize data, and to facilitate the writing of new software programs. The advantage of this software is that it combines comprehensive math and graphics functions with a powerful high-level language. Since MATLAB contains a large library of ready-to-use routines for a wide range of applications, the user can solve technical computing problems much more quickly than with traditional programming languages, such as C++ and FORTRAN. The standard library of functions can be significantly expanded by add-on toolboxes, which are collections of functions for special purposes such as image processing, creating map displays, performing geospatial data analysis or solving partial differential equations.

During the last few years, MATLAB has become an increasingly popular tool in earth sciences. It has been used for finite element modeling, processing of seismic data, analyzing satellite imagery, and for the generation of digital elevation models from satellite data. The continuing popularity of the software is also apparent in published scientific literature, and many conference presentations have also made reference to MATLAB. Universities and research institutions have recognized the need for MATLAB training for staff and students, and many earth science departments across the world now offer MATLAB courses for undergraduates. *The MathWorks Inc.* provides classroom kits for teachers at a reasonable price, and it is also possible for students to purchase a low-cost edition of the software. This student version provides an inexpensive way for students to improve their MATLAB skills.

The following sections contain a tutorial-style introduction to MATLAB, to the setup on the computer (Section 2.2), the syntax (Section 2.3), data

M.H. Trauth, *MATLAB® Recipes for Earth Sciences*, 3rd ed.,
DOI 10.1007/978-3-642-12762-5_2, © Springer-Verlag Berlin Heidelberg 2010

input and output (Sections 2.4 and 2.5), programming (Section 2.6), and visualization (Section 2.7). Advanced sections are also included on generating M-files to regenerate graphs (Section 2.8) and on publishing M-files (Section 2.9). It is recommended to go through the entire chapters in order to obtain a good knowledge of the software before proceeding to the following chapter. A more detailed introduction can be found in the *MATLAB 7 Getting Started Guide* (The MathWorks 2010) which is available in print form, online and as PDF file.

In this book we use MATLAB Version 7.10 (Release 2010a), the Image Processing Toolbox Version 7.0, the Mapping Toolbox Version 3.1, the Signal Processing Toolbox Version 6.13, the Statistics Toolbox Version 7.3 and the Wavelet Toolbox Version 4.5.

2.2 Getting Started

The software package comes with extensive documentation, tutorials and examples. The first three chapters of the book *MATLAB 7 Getting Started Guide* (The MathWorks 2010) are directed at beginners. The chapters on programming, creating graphical user interfaces (GUI) and development environments are aimed at more advanced users. Since *MATLAB 7 Getting Started Guide* provides all the information required to use the software, this introduction concentrates on the most relevant software components and tools used in the following chapters of this book.

After the installation of MATLAB, the software is launched either by clicking the shortcut icon on the desktop or by typing

```
matlab
```

in the operating system prompt. The software then comes up with several window panels (Fig. 2.1). The default desktop layout includes the *Current Folder* panel that lists the files in the directory currently being used. The *Command Window* presents the interface between the software and the user, i. e., it accepts MATLAB commands typed after the prompt, `>>`. The *Workspace* panel lists the variables in the MATLAB workspace, which is empty when starting a new software session. The *Command History* panel records all operations previously typed into the Command Window and enables them to be recalled by the user. In this book we mainly use the Command Window and the built-in *Editor*, which can be launched by typing

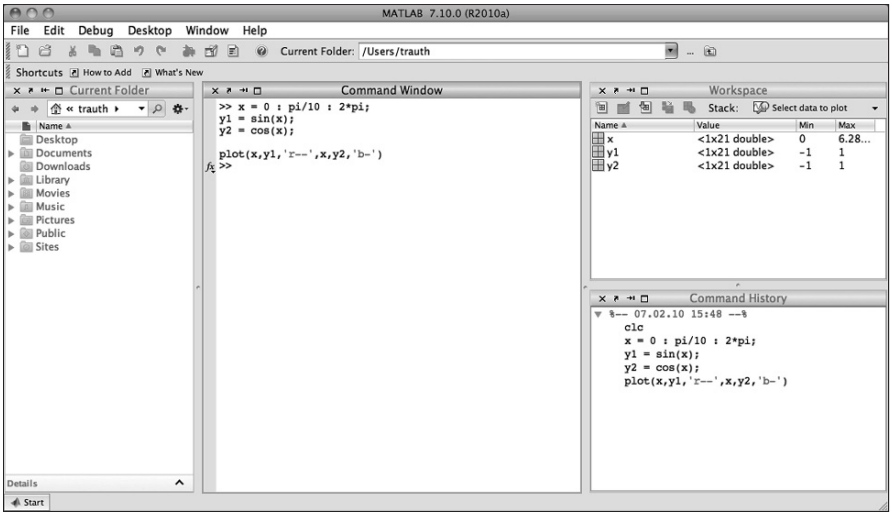


Fig. 2.1 Screenshot of the MATLAB default desktop layout including the *Current Folder* (left in the figure), the *Command Window* (center), the *Workspace* (upper right) and *Command History* (lower right) panels. This book uses only the Command Window and the built-in *Editor*, which can be called up by typing `edit` after the prompt. All information provided by the other panels can also be accessed through the Command Window.

```
edit
```

or by selecting the *Editor* from the *Desktop* menu. By default, the software stores all of your MATLAB-related files in the startup folder named *MATLAB*. Alternatively, you can create a personal working directory in which to store your MATLAB-related files. You should then make this new directory the working directory using the *Current Folder* panel or the *Folder Browser* at the top of the MATLAB desktop. The software uses a *search path* to find MATLAB-related files, which are organized in directories on the hard disk. The default search path includes only the MATLAB directory that has been created by the installer in the applications folder and the default working directory. To see which directories are in the search path or to add new directories, select *Set Path* from the *File* menu, and use the *Set Path* dialog box. The modified search path is saved in a file *pathdef.m* on your hard disk. The software will then in future read the contents of this file and direct MATLAB to use your custom path list.

2.3 The Syntax

The name MATLAB stands for *matrix laboratory*. The classic object handled by MATLAB is a *matrix*, i. e., a rectangular two-dimensional *array* of numbers. A simple 1-by-1 matrix or array is a scalar. Matrices with one column or row are vectors, time series or other one-dimensional data fields. An *m*-by-*n* matrix or array can be used for a digital elevation model or a grayscale image. Red, green and blue (RGB) color images are usually stored as three-dimensional arrays, i. e., the colors red, green and blue are represented by an *m*-by-*n*-by-3 array.

Before proceeding, we need to clear the workspace by typing

```
clear
```

after the prompt in the Command Window. Clearing the workspace is always recommended before working on a new MATLAB project. Entering matrices or arrays in MATLAB is easy. To enter an arbitrary matrix, type

```
A = [2 4 3 7; 9 3 -1 2; 1 9 3 7; 6 6 3 -2]
```

which first defines a variable `A`, then lists the elements of the matrix in square brackets. The rows of `A` are separated by semicolons, whereas the elements of a row are separated by blank spaces, or alternatively, by commas. After pressing *return*, MATLAB displays the matrix

```
A =
     2     4     3     7
     9     3    -1     2
     1     9     3     7
     6     6     3    -2
```

Displaying the elements of `A` could be problematic for very large matrices, such as digital elevation models consisting of thousands or millions of elements. To suppress the display of a matrix or the result of an operation in general, the line should be ended with a semicolon.

```
A = [2 4 3 7; 9 3 -1 2; 1 9 3 7; 6 6 3 -2];
```

The matrix `A` is now stored in the workspace and we can carry out some basic operations with it, such as computing the sum of elements,

```
sum(A)
```

which results in the display

```
ans =
    18    22     8    14
```

Since we did not specify an output variable, such as `A` for the matrix entered above, MATLAB uses a default variable `ans`, short for *answer* or *most recent answer*, to store the results of the calculation. In general, we should define variables since the next computation without a new variable name will overwrite the contents of `ans`.

The above example illustrates an important point about MATLAB: the software prefers to work with the columns of matrices. The four results of `sum(A)` are obviously the sums of the elements in each of the four columns of `A`. To sum all elements of `A` and store the result in a scalar `b`, we simply need to type

```
b = sum(sum(A));
```

which first sums the columns of the matrix and then the elements of the resulting vector. We now have two variables, `A` and `b`, stored in the workspace. We can easily check this by typing

```
whos
```

which is one the most frequently-used MATLAB commands. The software then lists all variables in the workspace, together with information about their sizes or dimensions, number of bytes, classes and attributes (see Section 2.5 for details about classes and attributes of objects).

```
Name        Size              Bytes  Class     Attributes
A           4x4                 128  double
ans         1x4                  32  double
b           1x1                   8  double
```

Note that by default MATLAB is case sensitive, i. e., `A` and `a` can define two different variables. In this context, it is recommended that capital letters be used for matrices and lower-case letters for vectors and scalars. We could now delete the contents of the variable `ans` by typing

```
clear ans
```

Next, we will learn how specific matrix elements can be accessed or exchanged. Typing

```
A(3,2)
```

simply yields the matrix element located in the third row and second column, which is `9`. The matrix indexing therefore follows the rule (*row, col-*

umn). We can use this to replace single or multiple matrix elements. As an example, we type

```
A(3,2) = 30
```

to replace the element `A(3,2)` by `30` and to display the entire matrix.

```
A =
     2     4     3     7
     9     3    -1     2
     1    30     3     7
     6     6     3    -2
```

If we wish to replace several elements at one time, we can use the *colon operator*. Typing

```
A(3,1:4) = [1 3 3 5]
```

or

```
A(3,:) = [1 3 3 5]
```

replaces all elements of the third row of the matrix `A`. The colon operator also has several other uses in MATLAB, for instance as a shortcut for entering matrix elements such as

```
c = 0 : 10
```

which creates a row vector containing all integers from 0 to 10. The resultant MATLAB response is

```
c =
     0   1   2   3   4   5   6   7   8   9   10
```

Note that this statement creates 11 elements, i. e., the integers from 1 to 10 and the zero. A common error when indexing matrices is to ignore the zero and therefore expect 10 elements instead of 11 in our example. We can check this from the output of `whos`.

```
Name        Size              Bytes  Class     Attributes
A           4x4                 128  double
ans         1x1                   8  double
b           1x1                   8  double
c           1x11                 88  double
```

The above command creates only integers, i. e., the interval between the vector elements is one unit. However, an arbitrary interval can be defined, for example 0.5 units. This is later used to create evenly-spaced time axes for

time series analysis. Typing

```
c = 1 : 0.5 : 10
```

results in the display

```
c =
  Columns 1 through 6
    1.0000    1.5000    2.0000    2.5000    3.0000    3.5000
  Columns 7 through 12
    4.0000    4.5000    5.0000    5.5000    6.0000    6.5000
  Columns 13 through 18
    7.0000    7.5000    8.0000    8.5000    9.0000    9.5000
  Column 19
   10.0000#
```

which autowraps the lines that are longer than the width of the Command Window. The display of the values of a variable can be interrupted by pressing *Ctrl+C* (*Control+C*) on the keyboard. This interruption affects only the output in the Command Window, whereas the actual command is processed before displaying the result.

MATLAB provides standard arithmetic operators for addition, `+`, and subtraction, `-`. The asterisk, `*`, denotes matrix multiplication involving inner products between rows and columns. For instance, we multiply the matrix `A` with a new matrix `B`

```
B = [4 2 6 5; 7 8 5 6; 2 1 -8 -9; 3 1 2 3];
```

the matrix multiplication is then

```
C = A * B'
```

where `'` is the complex conjugate transpose, which turns rows into columns and columns into rows. This generates the output

```
C =
    69   103   -79    37
    46    94    11    34
    75   136   -76    39
    44    93    12    24
```

In linear algebra, matrices are used to keep track of the coefficients of linear transformations. The multiplication of two matrices represents the combination of two linear transformations into a single transformation. Matrix multiplication is not commutative, i. e., `A*B'` and `B*A'` yield different results in most cases. Similarly, MATLAB allows matrix divisions, right, `/`, and left, `\`, representing different transformations. Finally, the software also allows powers of matrices, `^`.

In earth sciences, however, matrices are often simply used as two-dimensional arrays of numerical data rather than an array representing a linear transformation. Arithmetic operations on such arrays are carried out element-by-element. While this does not make any difference in addition and subtraction, it does affect multiplicative operations. MATLAB uses a dot as part of the notation for these operations.

As an example, multiplying `A` and `B` element-by-element is performed by typing

```
C = A .* B
```

which generates the output

```
C =
     8     8    18    35
    63    24    -5    12
     2     3   -24   -45
    18     6     6    -6
```

2.4 Data Storage and Handling

This section deals with how to store, import and export data with MATLAB. Many of the data formats typically used in earth sciences have to be converted before being analyzed with MATLAB. Alternatively, the software provides several import routines to read many binary data formats in earth sciences, such as those used to store digital elevation models and satellite data.

A computer generally stores data as *binary digits* or *bits*. A bit is analogous to a two-way switch with two states, on = 1 and off = 0. The bits are joined together to form larger groups, such as bytes consisting of 8 bits, in order to store more complex types of data. Such groups of bits are then used to encode data, e. g., numbers or characters. Unfortunately, different computer systems and software use different schemes for encoding data. For instance, the characters in the widely-used text processing software Microsoft Word differ from those in Apple Pages. Exchanging binary data is therefore difficult if the various users use different computer platforms and software. Binary data can be stored in relatively small files if both partners are using similar systems of data exchange. The transfer rate for binary data is generally faster than that for the exchange of other file formats.

Various formats for exchanging data have been developed during recent decades. The classic example for the establishment of a data format

that can be used with different computer platforms and software is the *American Standard Code for Information Interchange* (ASCII) that was first published in 1963 by the American Standards Association (ASA). As a 7-bit code, ASCII consists of 2^7=128 characters (codes 0 to 127). Whereas ASCII-1963 was lacking lower-case letters, in the ASCII-1967 update lower-case letters as well as various control characters such as *escape* and *line feed* and various symbols such as brackets and mathematical operators were also included. Since then, a number of variants appeared in order to facilitate the exchange of text written in non-English languages, such as the expanded ASCII containing 255 codes, e. g., the Latin-1 encoding.

The simplest way to exchange data between a certain piece of software and MATLAB is using the ASCII format. Although the newer versions of MATLAB provide various import routines for file types such as Microsoft Excel binaries, most data arrive in the form of ASCII files. Consider a simple data set stored in a table such as

```
SampleID  Percent C     Percent S
101           0.3657        0.0636
102           0.2208        0.1135
103           0.5353        0.5191
104           0.5009        0.5216
105           0.5415        -999
106           0.501         -999
```

The first row contains the names of the variables and the columns provide the data for each sample. The absurd value `-999` indicates missing data in the data set. Two things have to be changed to convert this table into MATLAB format. First, MATLAB uses `NaN` as the representation for *Not-a-Number* that can be used to mark missing data or gaps. Second, a percent sign, `%`, should be added at the beginning of the first line. The percent sign is used to indicate nonexecutable text within the body of a program. This text is normally used to include comments in the code.

```
%SampleID Percent C     Percent S
101           0.3657        0.0636
102           0.2208        0.1135
103           0.5353        0.5191
104           0.5009        0.5216
105           0.5415        NaN
106           0.501         NaN
```

MATLAB will ignore any text appearing after the percent sign and continue processing on the next line. After editing this table in a text editor, such as the *MATLAB Editor*, it can be saved as ASCII text file *geochem.txt* in the current working directory (Fig. 2.2). The MATLAB workspace should first

be cleared by typing

```
clear
```

after the prompt in the Command Window. MATLAB can now import the data from this file with the `load` command.

```
load geochem.txt
```

MATLAB then loads the contents of file and assigns the matrix to a variable `geochem` specified by the filename *geochem.txt*. Typing

```
whos
```

yields

```
Name          Size              Bytes  Class     Attributes
geochem       6x3                 144  double
```

The command `save` now allows workspace variables to be stored in a binary format.

```
save geochem_new.mat
```

MAT-files are double precision binary files using *.mat* as extension. The advantage of these binary MAT-files is that they are independent of the com-

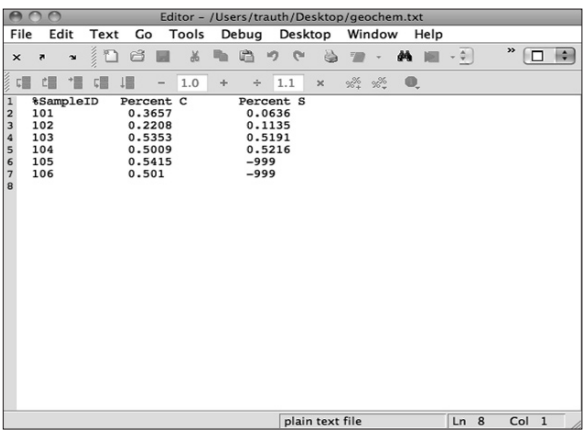


Fig. 2.2 Screenshot of MATLAB *Editor* showing the content of the file *geochem.txt*. The first line of the text is commented by a percent sign at the beginning of the line, followed by the actual data matrix.

puter platforms running different floating-point formats. The command

```
save geochem_new.mat geochem
```

saves only the variable `geochem` instead of the entire workspace. The option `-ascii`, for example

```
save geochem_new.txt geochem -ascii
```

again saves the variable `geochem`, but in an ASCII file named *geochem_new.txt*. In contrast to the binary file *geochem_new.mat,* this ASCII file can be viewed and edited using the MATLAB Editor or any other text editor.

2.5 Data Structures and Classes of Objects

The default data type or *class* in MATLAB is *double precision* or `double`, which stores data in a 64-bit array. This double precision array allows storage of the sign of a number (first bit), the exponent (bits 2 to 12) and roughly 16 significant decimal digits between approximately 10^{-308} and 10^{+308} (bits 13 to 64). As an example, typing

```
clear

rand('seed',0)
A = rand(3,4)
```

creates a 3-by-4 array of random numbers with double precision. We use the function `rand` that generates uniformly distributed pseudorandom numbers within the open interval (0,1). To obtain identical data values, we reset the random number generator by using the integer 0 as *seed* (see Chapter 3 for more details on random number generators and types of distributions). Since we did not use a semicolon here we get the output

```
A =
    0.2190    0.6793    0.5194    0.0535
    0.0470    0.9347    0.8310    0.5297
    0.6789    0.3835    0.0346    0.6711
```

By default, the output is in a scaled fixed point format with 5 digits, e. g., 0.2190 for the `(1,1)` element of `A`. Typing

```
format long
```

switches to a fixed point format with 16 digits for double precision. Recalling `A` by typing