Chapter 2

Mathematica

2.1 Introduction

2.1.1 History

In 1979–1981, Stephen Wolfram constructed SMP (*Symbolic Manipulation Program*), the first modern computer algebra system (SMP was essentially Version Zero of *Mathematica*).

In 1986–1988, Stephen Wolfram developed the first version of *Mathematica*. The concept of *Mathematica* was a single system that could handle many specific problems (e.g., symbolic, numerical, algebraic, graphical). In 1987, Wolfram founded a company, *Wolfram Research*, which continues to extend *Mathematica*.

In 1991, the second version of *Mathematica* appears with more built-in functions, MathLink protocol for interprocess and network communication, sound support, notebook front end.

In 1996, the third version introduced interactive mathematical typesetting system, exporting HTML, hyperlinks, and many other functions.

In 1999, the fourth version of *Mathematica* appears with important enhancements in speed and efficiency in numerical calculation, publishing documents in a variety of formats, and enhancements to many built-in functions.

In 2003, in the fifth version of *Mathematica* the core coding was improved and the horizons of *Mathematica* are more extended (e.g., in numerical linear algebra, in numerical solutions for differential

I.K. Shingareva, C. Lizárraga-Celaya, *Maple and Mathematica*, 2nd ed.,
DOI 10.1007/978-3-211-99432-0_2, © Springer-Verlag Vienna 2009

equations, in supporting an extended range of import and export graphic and file structures, in solving equations and inequalities symbolically over different domains).

In 2007, version 6 is introduced. In this upgrade, *Mathematica* has been substantially redesigned in its internal architecture for better functionality. It increases interactivity, more adaptive visualization, ease of data integration, symbolic interface construction among others new features.

In 2008, version 7 comes with a lot of achievements and several new areas of Mathematica. Among them we would like to mention integrated image processing, parallel high-performance computing, discrete calculus, new computable data sources, new algorithms for visualization and graphics and for solving transcendental, differential, and difference equations.

2.1.2 Basic Features

Symbolic, numerical, acoustic, graphical, parallel computations,

Static and Dynamic computations,

Extensibility and elegance,

Available for MS Windows, Linux, UNIX, Mac OS operating systems,

Powerful and logical language,

Extensive library of mathematical functions and specialized packages,

An interactive front end with notebook interface,

Interactive mathematical typesetting system,

Free resources, The *Mathematica* Learning Center, Wolfram Demonstrations Project, Wolfram Information Center.

2.1.3 Design

Mathematica consists of two basic parts: *the kernel*, computational engine and the interface, *front end*. These two parts are separate, but communicate with each other via the *MathLink* protocol.

The kernel interprets the user input and performs all computations. The kernel assigns the labels `In[number]` to the input expression and `Out[number]` to the output. These labels can be used for keeping the computation order. In the book, we will not include these labels in the examples. The result of kernels work can be viewed with the function `InputForm`.

```
Plot3D[Cos[x y],x,-Pi,Pi,y,-Pi,Pi]//InputForm
```

The interface between the user and the kernel is called *front end* and is used to display the input and the output generated by the kernel. The medium of the front end is the *Mathematica notebook*.

2.1.4 Changes for New Versions

There are significant changes to numerous *Mathematica* functions incorporated to the new versions of the system. Let us describe in general differences for $ver < 6$ and $ver \geq 6$ (in the next Chapters we note some of them for particular cases). A complete list of all changes can be found in the *Documentation Center* and on the *Wolfram Website*. Moreover, if you apply some functions (for $ver < 6$), *Mathematica* explains the corresponding changes for the current version.

Most notable changes (for $ver \geq 6$):

Changes in menus, dialogue boxes, and palettes,

Numerous changes in graphics functions,

The semicolon symbol (;) for complete suppressing graphics output,

Numerous functions (for $ver < 6$) located in packages have been incorporated to the *Mathematica* kernel. So many packages going obsolete.

Numerous functions (for $ver < 6$) are now located in different packages.

Numerous functions (for $ver < 6$) are available on the *Wolfram Website*.

Numerous functions (for *ver* < 6) are located in the "Legacy" packages.

Numerous functions (for *ver* <6) have been eliminated and incorporated to other functions.

Significant changes and enhancements in animation functions.

2.2 Basic Concepts

2.2.1 First Steps

We type *Mathematica* command and press the `RightEnter` key or `Shift+ Enter` (or `Enter` to continue the command on the next line). *Mathematica* evaluates the command, displays the result, and inserts a horizontal line (for the next input).

```
?Arc*
N[EulerGamma,40]
Solve[11*x^3-9*x+17==0,x]
Expand[(y+1)^10]
Plot[{4*Sin[2*x],Cos[2*x]^2},{x,0,2*Pi}]
Plot3D[Cos[x^2+y^2],{x,0,Pi},{y,0,Pi}]
```

The first line gives you information about *Mathematica'* s functions beginning with `Arc`, the second line returns a 40-digit approximation of Euler's constant γ, the third line solves the equation $11x^3 - 9x + 17 = 0$ for x, the fourth line expresses $(y+1)^{10}$ in a polynomial form, the fifth line plots the functions $4\sin(2x)$ and $\cos^2(2x)$ on the interval $[0, 2\pi]$, and the last line plots the function $\cos(x^2+y^2)$ on the rectangle $[0,\pi]\times[0,\pi]$.

2.2.2 Help System

Mathematica contains many sources of online help:

Wolfram Documentation Center, *Wolfram Demonstrations Project* (for *ver* ≥ 6), *Mathematica Virtual Book* (for *ver* ≥ 7),

`Help` menu, to mark a function and to press F1,

to type `?func` (information about a function), `??func` (for more extensive information), `Options[func]` (for information about options),

to use the symbols (`?,*`), e.g., `?Inv*`, `?*Plot`, `?*our*`.

2.2.3 Notebook and Front End

Mathematica notebooks are electronic documents that may contain *Mathematica* output, text, graphics (see `?Notebook`). You can work simultaneously with many notebooks.

Cells: a *Mathematica* notebook consists of a list of cells. Cells are indicated along the right edge of the notebook by blue brackets. Cells can contain subcells, and so on. The kernel evaluates a notebook cell by cell.

Operations with cells: a horizontal line across the screen is the beginning of the new cell, to open (close) cells or to change their format click (double-click) on the cell brackets, e.g., for changing the background cell color, click on a cell bracket, then

`Format` ⟶ `BackgroundColor` ⟶ `Yellow`.

Different types of cells: input cells (for evaluation), text cells (for comments), Title, Subtitle, Section, Subsection, etc., can be found in the menu `Format` ⟶ `Style`.

Palettes can be used for building or editing mathematical expressions, texts, graphics and allows one to access by clicking the mouse to the most common mathematical symbols, e.g.,

`Palettes` ⟶ `BasicMathInput`.

Useful features of the Front End

Completing existing symbol names, `Ctrl-k`,

Reminding the syntax of a function, `Ctrl-Shift-k`,

Typing shortcut keys and abbreviations, `Esc abbreviation Esc`,

Editing graphics boxes with the mouse.

In *Mathematica* (for $ver \geq 6$), the *Mathematica language* and the *Front End* are one and the same, this allows us to access programmatically to the whole Front End, and apply its components in any program.

2.2.4 Packages

In Mathematica, there exist many specialized functions and modules which are not loaded initially. They must be loaded separately from files in the *Mathematica* directory. These files are of the form `filename.m`.

The full name of a package consists of a `context` and a `short` name, and it is written as `` context`short ``.

Operations with packages:

`` <<context` ``, to load a package corresponding to a context,

`` <<c1`c2` ``, to load a package corresponding to a sequence of contexts,

`$Packages`, to display a list of loaded packages,

`$Context`, to display the current context,

`$ContextPath`, to get a list of contexts,

`` Names["context`*"] ``, to get a list of the functions in a package.

Problem 2.1 Show various maps of North America.

```
<<WorldPlot`
WorldPlot[{NorthAmerica,RandomColors}]
WorldPlot[NorthAmerica,WorldBackground->Hue[0.85],
 WorldGrid->None,WorldFrame->{Purple,Thickness[.012]},
 WorldProjection->LambertAzimuthal]
```

□

Problem 2.2 Print the names of all packages and the names of all functions located in the `FiniteFields` package.

```
SetDirectory[ToFileName[{$InstallationDirectory,
 "AddOns","Packages"}]]; FileNames[]
 SetDirectory[ToFileName[{$InstallationDirectory,
"AddOns","LegacyPackages"}]]; FileNames[]
SetDirectory[ToFileName[{$InstallationDirectory,
 "AddOns","ExtraPackages"}]]; FileNames[]
fNames[p_String]:=(Needs[StringJoin[p,"`"]];
 Names[StringJoin[p,"`*"]]); fNames["FiniteFields"]
```

□

2.2.5 Numerical Evaluation

Mathematica gives exact answers to arithmetic expressions with integers and reduces fractions. When the result is an irrational number, the output is returned in unevaluated form.

```
{135+638,-13*77,467/31,(3+8*4+9)/2,(-5)^(22),-5^(1/3)}
```

Mathematica represents the numbers in the decimal number system using a user-specified precision.

Numerical approximations:

`N[expr]`, `expr//N`, numerical approximation of `expr` (to 6 significant digits),

`N[expr,n]`, `NumberForm[expr,n]`, numerical approximation of the expression to n significant digits,

`EngineeringForm[expr,n]`, `ScientificForm[expr,n]`, engineering and scientific notations of numerical approximation of `expr` to n significant digits.

```
{Sqrt[17],Pi+2*Pi*I,1/3+1/5+1/7,N[E,30],N[Pi],Pi//N}
{x=N[Sin[Exp[Pi]]], ScientificForm[x,10],
 EngineeringForm[x,10], NumberForm[x,10]}
```

`Compile[{x1,...},fb]`, `Compile[{{x1,t1},{x2,t2}...},fb]`,

machine code of a function for improving the speed in numerical calculations (`fb` is the function body, and `t1`,`t2`,... are the types of variables, e.g., `_Real`, `_Complex`, `_Integer`).

```
n=0.5; f1[x_]:=(x^n*Exp[-x^n])^(1/n);
f2=Compile[{x},Evaluate[(x^n*Exp[-x^n])^(1/n)]];
Timing[Table[f1[x],{x,0.,50.,0.01}]]
Timing[Table[f2[x],{x,0.,50.,0.01}]]
```

2.3 *Mathematica* Language

Mathematica language is a very powerful programming language based on systems of transformation rules, functional, procedural, and object-oriented programming techniques. This distinguishes it from traditional programming languages. It supports a large collection of *data structures* or *Mathematica objects* (functions, sequences, sets, lists, arrays, tables, matrices, vectors, etc.) and operations on these objects (type-testing, selection, composition, etc.). The library can be enlarged with custom programs and packages.

2.3.1 Basic Principles

Symbol in *Mathematica*, `symb`, refers to a symbol with the specified name, e.g., expressions, functions, objects, optional values, results, argument names.

A name of symbol, `name`, is a combination of letters, digits, or certain special characters, not beginning with a digit, e.g., `a12new`. Once defined, a symbol retains its value until it is changed, cleared, or removed.

Expression, `expr`, is a symbol that represents an ordinary *Mathematica* expression in readable form. The head of `expr` can be obtained with `Head[expr]`. The structure and various forms of `expr` can be analyzed with `TreeForm`, `FullForm[expr]`, `InputForm[expr]`, e.g.,

```
l1={5, 1/2, 9.1, 2+3*I, x, {A,B}, a+b, a*b}
{Head /@ l1,FullForm[l1],InputForm[l1],TreeForm[l1],
 TraditionalForm[l1]}
```

Mathematica is case sensitive, there is a difference between lowercase and uppercase letters, e.g., `Sin[Pi]` and `sin[Pi]` are different. All *Mathematica* functions begin with a capital letter. Some functions (e.g., `PlotPoints`) use more than one capital. To avoid conflicts, it is best to begin with a lower-case letter for all user-defined symbols.

The result of each calculation is displayed, but it can be suppressed by using a semicolon (;), e.g., `Plot[Sin[x],x,0,2*Pi]; a=9; b=3; c=a*b`

Basic arithmetic operators and the corresponding functions:

`+`	`-`	`-`	`*`	`/`	`^`
`Plus`	`Subtract`	`Minus`	`Times`	`Divide`	`Power`

A missing symbol (∗), `2a`, or a space, `a b`, also imply multiplication. So it is best to indicate explicitly all arithmetic operations.
`Times[2,3,4,5]`, `Power[2,3]`.

Additional arithmetic operators and their equivalent functions:

`x++`	`x--`	`++x`	`--x`
`Increment`	`Decrement`	`PreIncrement`	`PreDecrement`
`x+=y`	`x-=y`	`x*=y`	`x/=y`
`AddTo`	`SubtractFrom`	`TimesBy`	`DivideBy`

In these operations x must have a numeric value, e.g., `x=5; x++;`

Logic and relation functions and their equivalent operators:

`And[x,y]`	`Or[x,y] Xor[x,y]`	`Not[x,y]`	`Implies[x,y]`
`x&&y`	`x||y`	`!x`	`x=>y`
`Equal[x,y]`	`Unequal[x,y]`	`Less[x,y]`	`Greater[x,y]`
`x==y`	`x!=y`	`x<y`	`x>y`
`LessEqual[x,y]`	`x<=y`	`GreaterEqual[x,y]`	`x>=y`

Logic expressions can be compared using `LogicalExpand`.

Patterns: Mathematica language is based on pattern matching.

A pattern is an expression which contains an underscore character (_). The pattern can stand for any expression. Patterns can be constructed from the templates, e.g.,

`x_`	`x_/;cond`	`pattern?test`	`x_:IniValue`
`x^n_`	`x_^n_`	`{x_, y_}`	`f[x_]   f_[x_]`

Problem 2.3 Define the function f with any argument named x, define an expression satisfying a given condition.

```
f[x_]:=Abs[x]; f[t]
ex=t+Log[a]+Log[b];{ex/.Log[b_]->Sin[b],ex/.Log[b]->Sin[b]}
```

□

Basic transformation rules: `->,  :>,  =,  :=,  ^:=,  ^=`

The rule `lhs->rhs` transforms `lhs` to `rhs`. *Mathematica* regards the left-hand side as a *pattern*.

The rule `lhs:>rhs` transforms `lhs` to `rhs`, evaluating `rhs` only after the rule is actually used.

```
n=3; a1=(a+b)^n; {a1/.Power[x_,n]->Power[Expand[x],n],
                  a1/.Power[x_,n]:>Power[Expand[x],n]}
```

The assignment `lhs=rhs` (or `Set`) specifies that the rule `lhs->rhs` should be used whenever it applies.

The assignment `lhs:=rhs` (or `SetDelayed`) specifies that `lhs:>rhs` should be used whenever it applies, i.e., `lhs:=rhs` does not evaluate `rhs` immediately but leaves it unevaluated until the rule is actually called.

```
a1=TrigReduce[Sin[x]^2]; a2:=TrigReduce[Sin[x]^2]; {a1,a2}
x=a+b; {a1,a2}
```

Note. In many cases both assignments produce identical results, but the operator (`:=`) must be used, e.g., for defining piecewise and recursive functions (see Sect. 4.8).

The rule `lhs^:=rhs` assigns `rhs` to be the delayed value of `lhs`, and associates the assignment with symbols that occur at level one in `lhs`, e.g.,

```
D[int[f_,l_List],var_]^:=int[D[f,var],l];D[int[f[t],{t,0,n}],t]
Unprotect[D]; D[int[f_,l_List],var_]:=int[D[f,var],l];
Protect[D]; D[int[f[t],{t,0,n}],t]
```

The rule `lhs^=rhs` assigns `rhs` to be the value of `lhs`, and associates the assignment with symbols that occur at level one in `lhs`, e.g.,

```
Unprotect[{Cos,Sin}]; Cos[k_*Pi]:=(-1)^k/;IntegerQ[k];
Sin[k_*Pi]:=0/;IntegerQ[k]; Protect[{Cos, Sin}];
IntegerQ[n]^=True;          {Cos[n*Pi], Sin[n*Pi]}
```

Transformation rules are useful for making substitutions without making the definitions permanent and are applied to an expression using the operator `/.` (`ReplaceAll`) or `//.` (`ReplaceRepeated`).

```
sol=Solve[x^2+4*x-10==0,x]; x+1/.sol
```

Unassignment of definitions:

`Clear[symb]` clears the symbol's definition and values, but does not clear its attributes, messages, or defaults,

`ClearAll[symb]` clears all definitions, values, attributes, messages, defaults,

`Remove[symb]` removes symbol completely,

`symb=.` clears a symbol's definition, e.g., `a=5; a=.; ?a`.

To clear all global symbols defined in a *Mathematica* session can be produced by several ways, e.g.,

```
Clear["Global`*"]; ClearAll["Global`*"]; Remove["`*"];
```

To recall a symbol's definition: `?symb`, e.g., `a=9; ?a`

To recall a list of all global symbols that have been defined (during a session): `` ?`* ``.

Initialization: in general, it is useful to start working with the following initialization:

```
ClearAll["Global`*"];                    Remove["Global`*"];
```

The difference between the operators (=) and (==): the operator `lhs=rhs` is used to assign `rhs` to `lhs`, and the equality operator `lhs==rhs` indicates equality (not assignment) between `lhs` and `rhs`, e.g.,

```
{eq=A==B,eq,eq[[2]],eq[[1]]}
```

An expression, `expr`, is a symbol that represents an ordinary *Mathematica* expression and there is a wide set of functions to work with it.

Data types: every expression is represented as a tree structure in which each node (and leaf) has a particular data type. For the analysis of any node and branch can be used a variety number of functions, e.g., `Length`, `Part`, a group of functions ending in the letter `Q` (`DigitQ`, `IntegerQ`, etc.). For example, the function `SameQ` or its equivalent `lhs===rhs` yields `True` if `lhs` is identical to `rhs`, and yields `False` otherwise.

A boolean expression, `bexpr`, is formed with the *logical operators* and the relation operators, e.g.,

```
LogicalExpand[!(p&&q)===LogicalExpand[!p||!q]]
```

An equation, `eq`, is represented using the binary operator `==`, and has two operands, the left-hand side `lhs` and the right-hand side `rhs`.

Inequalities, `ineq`, are represented using the relational operators and have two operands, the left-hand side `lhs` and the right-hand side `rhs`.

Strings: a string, `str`, is a sequence of characters having no value other than itself and can be used as labels for graphs, tables, and other displays. The strings are enclosed within double-quotes, e.g., `"abc"`.

Some useful string manipulation functions:

`StringLength[str]`, a number of characters in `str`,

`StringJoin[str1,...]` or `str1<>...`, concatenation of strings,

`StringReverse[str]` reverses the characters in `str`,

`StringDrop[str,n,m]` eliminates characters in `str` from n to m,

`StringTake[str,n,m]` takes characters in `str` from n to m,

`StringInsert[str1,str2,n1,n2,...]` inserts `str2` at each of the positions n_1, n_2, …, of `str1`,

`StringReplace[str,str1->nstr1,...]` replaces `str1` by `nstr1`,

`StringPosition[str,substr]`, a list of the start and end positions of substring.

Types of brackets:

Parentheses `(expr)`: grouping, `(x+9)*3`,

Square brackets `[expr]`: function arguments, `Sin[x]`,

Curly brackets `{expr}`: lists, `{a,b,c}`.

Types of quotes:

Back-quotes `` `expr` ``:

`` fullname=context`short ``, `` ` `` is a context mark,

`` `x` `` is a format string character, `` StringForm["`1`,`2`",a,b] ``,

`` ` `` is a number mark, `` 12.8` ``, machine precision approximate number,

`` ` `` is a precision mark, `` 12.8`10 ``, arbitrary precision number with precision 10,

``` `` ``` is an accuracy mark, ``` 12.8``15 ```, arbitrary precision number with accuracy 15.

Double-quotes `"expr"`: to create strings.

Previous results (during a session) can be used with symbols `%` (the last result), `%%` (the next-to-last result), and so on.

Comments can be included within the characters `(*comments*)`.

Function application: `expr//func` is equivalent to `fun[expr]`.

Incorrect response: if some functions take an "infinite" computation time, you may have entered or executed the command incorrectly. To terminate a computation, you can use:

`Evaluation` $\longrightarrow$ `Quit Kernel` $\longrightarrow$ `Local`.

2.3.2 Constants

Types of numbers: integer, rational, real, complex, root, e.g.,

```
{-5,5/6,-2.3^-4,ScientificForm[-2.3^-4],3-4*I,Root[#^2+#+1&,2]}
```

Mathematical constants: symbols for definitions of selected mathematical constants, e.g., `Catalan`, `Degree`, `E`, `EulerGamma`, `I`, `Pi`, `Infinity`, `GoldenRatio`, e.g., {`60Degree//N, N[E,30]`}.

Scientific constants: valuable tools for scientists and engineers in Physics and Chemistry can be applied with the packages `` Units` `` and `` PhysicalConstants` ``.

Note. Many functions from the `` Miscellaneous`ChemicalElements` `` package (for *ver* < 6) have been incorporated to the new `ElementData` function.
```

2.3.3 Functions

Two classes of functions: pure functions and functions defined in terms of a variable (*predefined* and *user-defined* functions).

Pure functions are defined without a reference to any specific variable. The arguments are labeled `#1,#2,...`, and an ampersand `&` is used at the end of definition.

```
f:=Sin[#1]&; g:=Sin[#1^2+#2^2]&;
{f[x],f[Pi],g[x,y],g[Pi,Pi]}
```

Predefined functions. Most of the mathematical functions are predefined.

Special functions. Mathematica includes all the common special functions of mathematical physics. We will discuss some of the more commonly used functions.

The names of mathematical functions are complete English words or the traditional abbreviations (for a few very common functions), e.g., `Conjugate`, `Mod`. Person's name mathematical functions have names of the form `PersonSymbol`, for example, the Legendre polynomials $P_n(x)$, `LegendreP[n,x]`.

Elementary trascendental functions:

`Exp[x]`, the exponential function,

`Log[x]`, `Log[b,x]`, the natural logarithm and the general logarithm,

`Sin`, `Cos, Tan, Cot, Sec, Csc`, the trigonometric functions,

`Sinh`, `Cosh`, `Tanh`, `Coth`, `Sech`, `Csch`, the hyperbolic functions,

`ArcSin`, `ArcSinh, ArcCos, ArcCosh`, ... (see `?Arc*`), the inverse trigonometric and hyperbolic functions.

```
{Sin[30 Degree], Sin[Pi/3], Exp[5]//N}
```

Other useful functions:

```
Max[x]  Min[x]    Round[x] Floor[x]   Ceiling[x] IntegerPart[x]
FractionalPart[x] Abs[x]   Sign[x]    n!   Factorial[x]     n!!
Factorial2[x]     Prime[n] Fibonacci[n]    Quotient[x]   Mod[x]
GCD[x] LCM[x] Timing[expr] RandomInteger[] RandomReal[]Random[]
RandomReal[{xmin,xmax}]    RandomComplex[]       IntegerQ[expr]
Print[expr1,...] PolynomialQ[expr] NumericQ[expr] VectorQ[expr]
```

`Abs[x]`, `Sign[x]`, the absolute value and the sign functions,

`n!`, `Prime[n]`, `Fibonacci[n]`, the factorial, the n-th prime, and the Fibonacci functions,

`Round`, `Floor`, `Ceiling`, `IntegerPart`, `FractionalPart`, converting real numbers to nearby integers, and the fractional part or real numbers,

`Quotient`, `Mod`, `GCD`, `LCM`, the quotient and the remainder (in the Division Algorithm), the greatest common divisor and the least common multiple,

`Random[]`, `Random[type,range]`, the random functions,

`Timing`, the kernel computation time of the expression,

`IntegerQ`, `PolynomialQ`, ... (see `?*Q`), a group of functions ending in the letter `Q` can be used for testing for certain conditions and return a value `True` or `False`,

`Print`, this function can be used in loops, strings, and other displays.

```
{Quotient[25,3], Mod[25,3], Random[Real,{2,9},10]}
{Fibonacci[10000]//Timing,  PolynomialQ[x^3*y+Sqrt[y]*y,y]}
```

User-defined functions are defined using the pattern `x_`, e.g., the functions of one or n variables $f(x)$=`expr`, $f(x_1, \dots, x_n)$=`expr`, the vector functions of one or n variables (for more details see Sect. 4.8).

```
f[x_]:=expr;      f=Function[x,expr];      f[x1_,...,xn_]:=expr;
f[t_]:={x1[t],...,xn[t]};      f=Function[t,{x1[t],...,xn[t]}];
          f[x1_,...,xn_]:={f1[x1_,...,xn_],...fn[x1_,...,xn_]};
```

Evaluation of a function or an expression without assigning a value can be performed using the replacement operator `/.`,

`f[a]`	`f[a,b]`	`expr /. x->a`	`expr /.{x->a,x->b}`

Composition functions are defined with the operation `Composition` (the arguments of this operation are pure functions `f1, f2,...`) and using the functions `Nest`, `NestList`, e.g., the composition function $(f_1 \circ f_2 \circ \ \cdots \ \circ f_n)(x)$ or $(f \circ f \circ \ \cdots \ \circ f)(x)$ (n times).

`Composition[f1,f2,...]`	`Composition[f,...,f]`	`f1@f2@...`
`Nest[f,expr,n]`	`NestList[f,expr,n]`	`f@f@f...`

Problem 2.4 Define the function $f(x,y) = 1 - \sin(x^2 + y^2)$ and evaluate $f(1,2)$, $f(0,a)$, $f(a,b)$.

```
f[x_,y_]:=1-Sin[x^2+y^2];
f1=Function[{x,y},1-Sin[x^2+y^2]]; {N[f[1,2]],f[0,a],
 Simplify[f[a,b]], N[f1[1,2]], f1[0,a], Simplify[f1[a,b]]}
```

□

Problem 2.5 Define the vector function $h(x,y) = \langle \cos(x-y), \sin(x-y) \rangle$ and calculate $h(1,2)$, $h(\pi,-\pi)$, and $h\big(\cos(a^2), \cos(1-a^2)\big)$.

```
h[x_,y_]:={Cos[x-y],Sin[x-y]};
{N[h[1,2]],h[Pi,-Pi],h[Cos[a^2],Cos[1-a^2]]//FullSimplify}
```

□

Problem 2.6 Graph the real roots of the equation $x^3 + (a-3)^3 x^2 - a^2 x + a^3 = 0$ for $a \in [0,1]$.

```
SetOptions[Plot,ImageSize->300,PlotStyle->
 {Hue[0.9],Thickness[0.01]},PlotPoints->100,PlotRange->All];
i=1; nD=10; sol=Solve[x^3+(a-3)^3*x^2-a^2*x+a^3==0,x];
r[i_,b_]:=N[sol[[i,1,2]]/.{a->b},nD];
g=Table[Plot[r[i,b],{b,0,1}],{i,1,3}]; GraphicsRow[g]
```

□

Problem 2.7 For the functions $f(x)=x^2$ and $h(x)=x+\sin x$ calculate the composition functions $(f\circ h\circ f)(x)$ and $(f\circ f\circ f\circ f)(x)$.

```
f[x_]:=x^2; fF:=#1^2&; hF:=#1+Sin[#1]&;
{fF@hF@fF[x],fF@fF@fF@fF[x],Nest[f,x,4],NestList[f,x,4]}
```

□

Piecewise functions can be defined using the conditional operator (`/;`) or the functions `Piecewise` and `UnitStep`.

```
f[x_]:=expr/;cond           f[x_]:=UnitStep[x-a]*UnitStep[b-x];
f[x_]:=Piecewise[{{cond1,val1},...,{condn,valn}}];         f[x]
```

Problem 2.8 Define and graph the function $f(x)=\begin{cases} 0, & |x|>1 \\ 1-x, & 0\le x\le 1 \\ 1+x, & -1\le x\le 0 \end{cases}$ in [-3, 3].

```
f[x_]:=0/;Abs[x]>1; f[x_]:=1-x/;0<=x<=1;
f[x_]:=1+x/;-1<=x<=0; Plot[f[x],{x,-3,3},
 PlotStyle->{Hue[0.85],Thickness[0.01]}]
```

□

Problem 2.9 Graph the periodic extension of $f(x)=\begin{cases} x, & 0\le x<1 \\ 1, & 1\le x<2 \\ 3-x, & 2\le x<3 \end{cases}$ in [0, 20].

```
f[x_]:=x/;0<=x<1; f[x_]:=1/;1<=x<2;
f[x_]:=3-x/;2<=x<=3; f[x_]:=f[x-3]/;x>3; Plot[f[x],{x,0,20}]
```

□

Recursive functions are the functions that are defined in terms of themselves.

Problem 2.10 Calculate Fibonacci numbers, $F(n)=F(n-1)+F(n-2)$, $F(0)=0$, $F(1)=1$, using the memory feature and the analytic function. Compare the computation time in both cases.

```
$RecursionLimit=Infinity;
Fib[0]=0; Fib[1]=1; Fib[n_]:=Fib[n]=Fib[n-1]+Fib[n-2];
Table[Fib[i],{i,10,40}]
F[n_]:=(((1+Sqrt[5])/2)^n-((1-Sqrt[5])/2)^n)/Sqrt[5];
Fib[3000]//Timing
Expand[F[3000]]//Timing
```

□

2.3.4 Modules

A module is a local object that consists of several functions which one needs to use repeatedly (see `?Module`). A module can be used to define a function (if the function is too complicated to write by using the notation `f[x_]:=expr`), to create a matrix, a graph, a logical value, etc.

`Block` is similar to `Module`, the main difference between them is that `Block` treats *the values* assigned to symbols as local, but *the names* as global, whereas `Module` treats *the names* of local variables as local.

`With` is similar to `Module`, the principal difference between them is that `With` uses *local constants* that are evaluated only once, but `Module` uses *local variables* whose values may change many times.

```
Module[{var1,...},body];          Module[{var1=val1,...},body];
Block[{var1,...},expr];            Block[{var1=val1,...},expr];
With[{var1=val1,var2=val2,...},expr];
```

Here `var1,...` are local variables, `val1,...` are initial values of local variables, `body` is the body of the module (as a sequence of statements separated by semicolons). The final result of the module is the result of the last statement (without a semicolon). Also `Return[expr]` can be used to return an expression.

Problem 2.11 Define a module and module function, `mF`, that calculate the maximum of x and y, and then find the maximum of the values (34/9, 674/689).

```
Module[{x=34/9,y=674/689},If[x>y,x,y]]
mF[x_,y_]:=Module[{m},If[x>y,m=x,m=y]]; mF[34/9,674/689]
```

□

Problem 2.12 Compare the following calculations using `With`, `Module`, and `Block`.

```
f1[x_]:=With[{a=2,b=1},Tan[a*x]-b];
f2[x_]:=Module[{i},Sum[x^(-2*i),{i,1,9}]];
f3[x_]:=Block[{i},Sum[x^(-2*i),{i,1,9}]];
{f1[x], {a,b}, f2[4], f2[x], f2[i], f3[4], f3[x], f3[i]}
```

□

Problem 2.13 Consider a 2-degree of freedom holonomic system described by the Lagrange equations, namely, the motion of a double pendulum of mass m and length L in the vertical plane due to gravity (see Sect. 1.3.4). Construct a module for deriving the Lagrange equations.

```
eqLagrange[q_,L_,Q_]:=Module[
 {Lq,Ldq,Ldq1,dLdq,dLdq1,eq},
 L1=L/.{dq[[1]]->"A'",dq[[2]]->"B'"}; n=Length[q];
 Lq=Map[D[L1,#1]&,q]/.{"A'"->dq[[1]],"B'"->dq[[2]]};
 Ldq=Map[D[L,#1]&,dq]; Ldq1=Ldq/.{dq[[1]]->"A'",dq[[2]]->"B'"};
 dLdq=Sum[Map[D[#1,q[[j]]]&,Ldq1]*dq[[j]]+Map[
 D[#1,dq[[j]]]&,Ldq]*d2q[[j]],{j,1,n}]; dLdq1=dLdq/.
 {"A'"->dq[[1]],"B'"->dq[[2]]}; eq=Factor[dLdq1-Lq-Q]];
q={A,B}; Q={0,0}; dq=Map[#1'&,q]; d2q=Map[#1''&,q];
T=m*x^2*dq[[1]]^2+m*x^2*dq[[1]]*dq[[2]]*Cos[q[[1]]
 -q[[2]]]+1/2*m*x^2*dq[[2]]^2;
P=-m*g*x*Cos[q[[1]]]-m*g*x*(Cos[q[[1]]]+Cos[q[[2]]]);
g=omega^2*x; L=T-P; eqLagrange[q,L,Q]
```
□

2.3.5 Control Structures

In Mathematica language there are the following *two control structures*: the selection structures `If`, `Which`, `Switch` and the repetition structures `Do`, `While`, `For`,

`If[cond,exprTrue]`	`If[cond,exprTrue,exprFalse]`
	`If[cond,exprTrue,exprFalse,exprNeither]`
`Which[cond1,expr1,...]`	`Switch[expr,patt1,val1,patt2,val2,...]`
`Do[expr,{i,i1,i2,iStep}]`	`Do[expr,{i,i1,i2,iS},{j,j1,j2,jS},...]`
`While[cond,expr]`	`For[i=i1,cond,iStep,expr]`

where `exprTrue`, `exprFalse`, `exprNeither` are expressions that execute, respectively, if the condition `cond` is `True`, `False`, and is neither `True` or `False`; `i,i1,i2` (`j,j1,j2`) are the loop variable and the initial and the last values of `i` (`j`).

The result of `Which` is the expression `expr1,expr2,...` corresponding to the first true condition `cond1,cond2,...`.

In `Switch`, the expression `expr` is compared with patterns `patt1,...` until a match is found and the corresponding value `val1,...` is the result. These operators can be nested.

The operators `Break[]`, `Continue[]`, `Goto[name]` inside the loops are used for breaking out of a loop, to proceed directly to the next iteration, or for transferring the control to the point `Label[name]`.

Problem 2.14 Compare the following calculations using `Which` and `Switch`.

```
f1[x_]:=Which[x>0,1,x==0,1/2,x<0,-1]; f1[0]
Plot[f1[x],{x,-2,2},PlotStyle->Thickness[0.01]]
f2[x_]:=Switch[x,_Integer,x,_List,Join[x,{c}],_Real,N[x]];
{f2[4], f2[Sum[2.3+0.01*i,{i,1,10}]], f2[{a,b}]}
```

□

Problem 2.15 Find all numbers from 1 to 30 which are not multiples of 2 or 5.

```
Do[If[Mod[i,2]==0||Mod[i,5]==0,,Print[i]],{i,1,30}]
```

□

Problem 2.16 Calculate 20! using `Do`, `While`, `For` loops.

```
fact=1; n=20; Do[fact=fact*i, {i,1,n}]; fact
fact=1; i=20; While[i>0, fact=fact*i; i--]; fact
For[fact=1; i=1, i<=20, i++, fact=fact*i]; fact
```

□

Problem 2.17 Define the function *double factorial* for any integer n,

$$n!! = \begin{cases} n(n-2)(n-4)\ \ldots\ (4)(2), & n=2i, i=1,2,\ldots, \\ n(n-1)(n-3)\ \ldots\ (3)(1), & n=2i+1, i=0,1,\ldots \end{cases}$$

```
doubleFactorial[n_]:=Module[{p=1,i1},
 If[Mod[n,2]==0,i1=2,i1=1]; Do[p=p*i,{i,i1,n,2}];p];
n1=20; n2=41; {doubleFactorial[n1],doubleFactorial[n2],
 Factorial2[n1],Factorial2[n2]}
Print["20!!=",doubleFactorial[n1]];
Print["41!!=",doubleFactorial[n2]];
```

□

Problem 2.18 Calculate the values x_i, where $x_i=(x_{i-1}+1/x_{i-1})$, $x_0=1$, until $|x_i - x_{i-1}| \leq \varepsilon$ ($i=1,2,\ldots,n$, $n=10$, $\varepsilon=10^{-3}$).

```
nD=10; n=10; x=1; xp=1; epsilon=10^(-3); i=1;
While[Abs[x-xp]<epsilon||i<=n, xp=x; x=N[x+1/x,nD];
     Print["x=",x]; i++]
```

□

Problem 2.19 Find an integer $N(x)$, $x \in [a,b]$ such that $\sum_{i=1}^{N(x)} i^{-x} \geq c$, where $a=1$, $b=2$, and $c=2$.

```
nD=10; a=1; b=2; c=2; Do[s=0; i=1; x=k;
 While[s<c && i<=100, s=s+N[i^(-x),nD]; i++];
 Print["s=",s,"   ","x=",x,"   ","N(x)=",i],{k,a,b,N[1/100,nD]}]
```

□

2.3.6 Objects and Operations

Lists are the fundamental objects in *Mathematica*. The other objects (e.g., sets, matrices, tables, vectors, arrays, tensors, objects containing data of mixed type) are represented as lists. A list is an ordered set of objects separated by commas and enclosed in curly braces, {`elements`}, or defined with the function `List[elements]`.

Nested lists are lists that contain other lists. There are many functions which manipulate lists and here we review some of the most basic.

Basic manipulation functions:

Create empty lists: { }, `List[]`.

Listable function. Many *Mathematica* functions performed on a list will be performed on each element of the list.

```
{list1={1,2,3,4,5}, Sqrt[list1]}
```

The basic standard operations should be applied to lists with the same number of elements.

```
{list1={1,3,5,7}, list2={2,4,6,8}, list1/list2}
```

Create lists and nested lists:

`Range[n,m,step]`, lists of numbers,
`Table[expr,{i,n,m,step}]`, lists according to a formula,
`Array[f,n,nIni]`, list elements are functions $f[n]$,
`Table[expr,{i,n,m},{j,k,l},...]`, nested lists according to a formula,
`Array[f,{n,m},{nIni,mIni},...]`, nested lists with elements $f[i,j]$,
`Characters[str]`, `CharacterRange[ch1,ch2]`, lists of characters.

```
f[x_]:=x^3-x^2-x-1; g[x_,y_]:=x^2+y^2;
{Range[0,Pi,Pi/3],Table[Sqrt[i],{i,1,20,4}],Array[f,20,0],
 Table[Sqrt[i+j],{i,1,4},{j,1,7}],Array[g,{3,5},{0,0}]}
{Characters["Maple&Mathematica"],CharacterRange["c","l"]}
```

Determine the structure of lists:

`Length[list]`, `Depth[list]`, `AtomQ[list]`, `LeafCount[list]`,
`StringLength[str]`, `Dimensions[list]`, `ArrayDepth[list]`.

```
vec={a,b,c}; mat={{a,b,c},{d,e,f}}; tens={{{a,b,c},{d,e,f}},
 {{g,h,i},{j,k,l}}}; l1={vec,mat,tens}; {Length/@l1,
 Dimensions/@l1, TensorRank/@l1,ArrayDepth/@l1, Depth/@l1}
```

Extract an i-th element from a list or a nested list:

`Part[list,i]` or `list[[i]]`,
`First[list]`, `Last[list]`, the first and the last elements of `list`.

```
{list1=Range[1,20,3], list2=Table[(i+j)^2,{i,1,4},{j,1,7}]}
{Part[list1, 4], list1[[4]], list2[[2,4]]}
```

Create a substructure (a part of a list):

`Rest[list]`, a list without the first element of `list`,
`Delete[list,n]`, a list without the element in the n-th position of `list`,
`Take[list,{n,m}]`, a list with the elements taken within the range,
`Drop[list,{n,m}]`, a list with the elements deleted within the range,
`Select[list,crit]`, a list with the elements satisfying a criterion,
`Cases[list,patt]`, a list with the elements matching a pattern,
`DeleteCases[list,patt]`, removing elements matching a pattern.

```
f1[x_]:=Sqrt[x^2-x-1]; list1=Array[f1,10,0]
{Take[list1,{2,7}],Drop[list1,{2,7}],Select[{0,1,-2,Pi,a},#>0&]}
Cases[{1,-2,I,"a",b},x_?StringQ]
DeleteCases[{1,-2,I,"a",b},x_?StringQ]
```

Insert an element into a list:

`Append[list,x]`, insert an element x to the right of the last element of `list`,
`Prepend[list,x]`, insert an element x to the left of the last element of `list`,
`Insert[list,x,n]`, insert an element x in position n.

```
list1=Table[i^2,{i,1,20,2}]; {Append[list1,a],Insert[list1,a,5]}
```

Replace the n-th element of a list by x: `ReplacePart[list,x,n]`.

```
{list1=CharacterRange["0","9"], ReplacePart[list1,a,5]}
```

Rearrange lists:

`Sort[list]` and `Reverse[list]`, sorting and reversing of lists,
`RotateLeft[list,n]`, `RotateRight[list,n]`, cycling of lists.

```
list1=Table[Random[Integer,{1,20}],{i,1,10}]
{Sort[list1],RotateLeft[list1,2],RotateRight[list1,-2]}
```

Concatenation of lists:

`Join[list1,list2,...]`, `StringJoin[str1,str2,...]`,
`Flatten[{list1,list2],...},1]`.

```
{list1=Range[-10,10],list2=Table[i^2,{i,1,10}]}
{Join[list1,list2],Flatten[{list1,list2},1],StringJoin["A","B"]}
```

Manipulation with nested lists:

`TreeForm[list]`, visualization of nested lists as a tree,
`Depth[list]`, the number of levels in a nested list,
`Level[list,levels]`, a list of all sublists of `list` on levels,
`Level[list,levels,f]`, applies function `f` to the sequence of sublists,
`Flatten[list]`, convert a nested list into a simple list,
`Flatten[list,n]`, partial flattening to level n,
`Partition[list,n]` , converts a simple list into sublists of length n.

```
tensor={{{a,b,c},{d,e,f}},{{g,h,i},{j,k,l}}}; TreeForm[tensor]
list1={{a1,a2,a3},{a4,a5,a6}}; list2=Flatten[list1];
{Depth[list1], Level[list1,2], Level[list1,1],
 Level[list1,{1},Subtract], Partition[list2, 2]}
```

Sets are represented as lists:

`Union[list1,...]`, a list of the distinct elements of lists,
`Intersection[list1,...]`, intersection of lists,
`Subset[list]`, a list of all subsets of the elements in `list`.

Note. The package ``Combinatorica` `` includes various useful set functions (for more details see Sect. 4.1).

Vectors are represented as lists, vectors are simple lists. Vectors can be expressed as single columns with `ColumnForm[list,horiz,vert]`.

```
d[i_]:=1/i; f[i_]:=1/i^2; {vec={a,b,c},
 Length[vec],Dimensions[vec],ArrayDepth[vec],VectorQ[vec]}
v1=N[Array[d,5]]; v2=N[Array[f,5]]; {v1,ColumnForm[v1,Right],
 v2,ColumnForm[v2,Right],a*v1+b*v2,v1.v2}
```

Tables, matrices, and tensors are represented as nested lists. There is no difference between the way they are stored: they can be generated using the functions `MatrixForm[list]`, `TableForm[list]`, or using the nested list functions (see above). Matrices and tables can also be conveniently generated using the *palette*,

`Insert` $\longrightarrow$ `Table/Matrix` or `Palettes` $\longrightarrow$ `BasicMathInput`.

```
A1={{a1,a2},{a3,a4}}; {MatrixForm[A1],TableForm[A1]}
```

Basic manipulation functions with matrices, tables, and tensors:

A matrix is a list of vectors. Matrices can be combined using the operations: addition (`+`), subtraction (`-`), scalar (`*`) and matrix multiplication (`.`), in more detail about matrix manipulations, see Chapter 5.

```
m1={{a1,a2},{a3,a4}}; m2={{b1,b2},{b3,b4}}; {MatrixForm[m1],
MatrixForm[m2], 4*m1-m2//MatrixForm, m1.m2//MatrixForm}
```

Some useful functions for tables:

`TableForm[list,ops]`, generating tables with some properties,
`PaddedForm[list,{n,m}]`, formatting numerical tables.

```
t1={{a1,2*a2,a3},{-a4,a5,-a6}}; TableForm[t1]
TableForm[t1,TableAlignments->Right, TableHeadings->
 {{"r1","r2"}, {"c1","c2","c3"}},TableSpacing->{3,3}]
t2=Table[{i,N[Sin[i]],N[Cos[i]]},{i,1,5}]; TableForm[t2]
f1:=PaddedForm[i,3]; f2:=PaddedForm[N[Sin[i]],{12,5}];
t3=Table[{f1,f2},{i,1,5}]; TableForm[t3]
```

A tensor is a list of matrices with the same dimensionality (in more detail about tensor manipulations, see Sect. 5.13).

Some useful functions for tensors:

`Table`, `Array`, creating tensors,
`TreeForm`, `MatrixForm`, visualizing tensors as a tree or a matrix,
`Length`, `Dimensions`, `TensorRank`, determining the tensor structure.

```
{Table[i1*i2,{i1,2},{i2,3}], Array[(#1*#2)&,{2,3}]}
tens={{{a,b,c},{d,e,f}},{{g,h,i},{j,k,l}}}
{TreeForm[tens], MatrixForm[tens], Length[tens],
 Dimensions[tens],TensorRank[tens],TensorQ[tens]}
```

Apply a function to each element of an object:

`Map[f,expr]` or `f/@expr`, at the first level in `expr`,
`Map[f,expr,levels]`, apply `f` to parts of `expr`,
`Apply[f,expr]` or `f@@expr`, replace the head of `expr` with the function `f`,
`Thread[f[args],head]`, "threads" `f` over any objects with `head` that appear in the arguments, `args`, of the function `f`.

```
l1={{a,b},{c,d}}; l2={{e,f},{g,h}}; eq1=a==b;
{f1/@l1, Map[f2,l1,{2}], f3/@(x^3+x+2)}
(Apply[Plus,(Range[1,100])^2]//N)===(Sum[i^2,{i,1,100}]//N)
{Thread[#^n&@eq1,Equal],Thread[eq1^n,Equal],eq1^n}
{l1==l2,Thread[l1==l2]}
```

Problem 2.20 A sequence of numbers $\{x_i\}$ $(i = 0, \dots, n)$ is defined by $x_{i+1} = ax_i(1 + x_i)$ $(0 < a < 10)$, where a is a given parameter. Define a list of coordinates $[i, x_i]$ such that $a = 3$, $x_0 = 0.1$, $n = 100$. Plot the graph of the sequence $\{x_i\}$.

```
n=100; l=Array[x,n,0]; a=3; l[[1]]=0.1;
Do[l[[i]]=a*l[[i-1]]*(1-l[[i-1]]),{i,2,n}]
l1=Table[{i,l[[i]]},{i,2,n}]
ListPlot[l1,PlotStyle->{PointSize[0.02],Hue[0.7]}]
```

□

Problem 2.21 Observe the function behavior $y(x) = \cos(6(x - a \sin x))$, $x \in [-\pi, \pi]$, $a \in [\frac{1}{2}, \frac{3}{2}]$ (in more detail about animation, see Sect. 3.6).

```
y[x_,a_]:=Cos[6*(x-a*Sin[x])]; n=20;
Animate[Plot[y[x,a],{x,-Pi,Pi},PlotRange->{-1,1},
  PlotStyle->Hue[0.7]],{a,1/2,3/2,1/n}]
```

□

2.3.7 Dynamic Objects

In *Mathematica* (for *ver* ≥ 6), the new kind of output, the *dynamic output* has been introduced allowing to create dynamic interfaces of different types. Numerous new functions for creating various dynamic interfaces have been developed. We mention the most important of them:

```
Dynamic[expr]   Slider[Dynamic[x]]      Slider[x,{x1,x2,xStep}]
Manipulate[expr,{x,x1,x2,xStep}]     TabView[{expr1,expr2,...}]
SlideView[{expr1,expr2,...}]     DynamicModule[{x=x0,...},expr]
Manipulator[x,{x1,x2}]       Animator[x,{x1,x2}]     Pane[expr]
```

```
{Slider[Dynamic[par]],Dynamic[Plot[Sin[1/var*par],{var,-1,1},
  PlotRange->{{-1,1},{-1,1}},ImageSize->500]]}
DynamicModule[{x=.1},{Slider[Dynamic[x]],
 Dynamic[Plot[Sin[1/var*x],{var,-1,1},PlotRange->
 {{-1,1},{-1,1}},ImageSize->500]]}]
Manipulate[Expand[(z+1)^n],{n,3,10,1}]
TabView[Table[Plot[Sin[1/x*par],{x,-1,1}],{par,-1,1,0.1}]]
SlideView[Table[Plot[Sin[1/x*par],{x,-1,1}],{par,-1,1,0.1}]]
```

Problem 2.22 Display a dynamic object without controls and the animation frame.

```
t=0; Row[{Pane[Animator[Dynamic[t],{-1,1}],{0,0}],
 Dynamic[Plot[Sin[1/x*t],{x,-1,1},ImageSize->300,
 PlotRange->{{-1,1},{-1,1}}]]}]
```

□

Part II

Mathematics: *Maple* and *Mathematica*

Chapter 3

Graphics

3.1 Simple Graphs

Graphs of real values of `expr` or the functions $f(x)$, $f(x,y)$, $x \in [x_1, x_2]$, $(x,y) \in [x_1,x_2] \times [y_1,y_2]$.

Maple:

```
f:=expr;               plot(f, x1..x2);
f:=x->expr;            plot(f(x),x= x1..x2, ops);
f:=(x,y)->expr;        plot3d(f(x,y),x=x1..x2,y=y1..y2,ops);
```

```
f1:=x->sin(x): f2:=sin(x); f3:=(x,y)->x^2-x-y^2-y-8;
plot(f1(x),x=-2*Pi..2*Pi); plot(f2,x=-2*Pi..2*Pi);
plot3d(f3(x,y),x=-6..6,y=-6..6,axes=boxed);
```

Mathematica:

```
Plot[f[x],{x,x1,x2}]        Plot3D[f[x,y],{x,x1,x2},{y,y1,y2}]
```

```
f[x_,y_]:=x^2-x-y^2-y-8; Plot[Sin[x],{x,-2*Pi,2*Pi}]
Plot3D[f[x,y],{x,-6,6},{y,-6,6},Boxed->True]
```

3.2 Various Options

In Maple, all the graphs can be drawn with various versions of `plot` and the package `plots`. The function `plot` has various forms (e.g., `logplot`, `odeplot`, `plot3d`) and various optional arguments which

I.K. Shingareva, C. Lizárraga-Celaya, *Maple and Mathematica*, 2nd ed.,
DOI 10.1007/978-3-211-99432-0_3, © Springer-Verlag Vienna 2009

define the final figure (see `?plot[options]`, `?plot3d[options]`), e.g., light setting, legends, axis control, titles, gridlines, real-time rotation of 3D graphs, wide variety of coordinate systems, etc.

```
g1:=x->exp(-(x-3)^2*cos(Pi*x)^2);
plot(g1(x),x=0..6,tickmarks=[4,4],title=`Graph of g(x)`);
plot(sin(x)/x,x=-3*Pi..3*Pi,scaling=constrained);
plot(sin(x)/x,x=-3*Pi..3*Pi,style=point);
plot(tan(x),x=-2*Pi..2*Pi,y=-4..4,discont=true);
Points:=[[1,2],[2,3],[3,5],[4,7],[6,13],[7,17],[8,19]];
plot(Pints,style=point);
plot(Points,style=line);
g2:=(x,y)->x^2*sin(2*y)+y^2*sin(2*x);
plot3d(g2(x,y),x=0..Pi,y=0..Pi,grid=[20,20],style=patch);
```

In Mathematica, there are many options available for function graphics which can define the final picture (in more detail, see `Options [Plot]`, `Options[Plot3D]`), e.g., light modeling, legends, axis control, titles, gridlines, etc. The general rule for defining options is:

```
Plot[f[x],{x,x1,x2},opName->value,...]
           Plot3D[f[x,y],{x,x1,x2},{y,y1,y2},opName->value,...]
```

Here `opName` is the option name.

Note. In *Mathematica* (*ver* < 6), a semicolon (;) was used to suppress the special line `-Graphics-` which follows the graphs. In *Mathematica* (*ver* ≥ 6), the semicolon suppresses the graph output completely. The options `DisplayFunction->Identity` and `DisplayFunction->$DisplayFunction` (also used in *ver* < 6 for suppressing the graph output) are no longer needed.

Formula for color graphs: Mathematica makes it easy to compute the RGB formula for color graphs, using `Insert` → `Color`. In addition, the names of predefined colors are available in the new `ColorData` function. The list of all colors can be obtained by typing `ColorData["Legacy","Names"]`, and the RGB formula of a particular color, e.g. `Coral`, by typing `ColorData["Legacy","Coral"]`. Additionally, all predefined color schemes can be inserted by using `Palettes` → `ColorSchemes`.

Some useful options for 2D graphs:

```
f[x_]:=Exp[-(x-3)^2*Cos[Pi*x]^2];  <<PlotLegends`
Plot[f[x],{x,-Pi,2*Pi},PlotRange->All,PlotLabel->"f[x]"]
Plot[f[x],{x,-Pi,2*Pi},AxesLabel->{"x,sm","y,sec"}]
Plot[f[x],{x,0,2*Pi},AspectRatio->Automatic]
Plot[f[x],{x,-Pi,2*Pi},AxesOrigin->{3,0},PlotRange->{{2,4},
 {0,1}},Frame->True,Axes->False,GridLines->{{2,2.5,3,Pi,
 3.5,4},Automatic},FrameTicks->{Automatic,{0.2,0.8}}]
f1[x_]:=Sin[x]; f2[x_]:=Cos[x]; f3[x_]:=Sin[x]-Cos[x];
Plot[{f1[x],f2[x],f3[x]},{x,-Pi,Pi},PlotStyle->
 {GrayLevel[0.5],Dashing[{0.02,0.03}],Thickness[0.02]},
 PlotLegend->{f1[x],f2[x],f3[x]},Filling->{1->{3},2->{3}},
 Ticks->{{0,Pi/4,Pi/2,3*Pi/4,Pi},Automatic}]
Plot[{f1[x],f2[x]},{x,-Pi,Pi},PlotStyle->{Red,Blue}]
Plot[{f1[x],f2[x],f3[x]},{x,-Pi,Pi},PlotStyle->
 {RGBColor[0.501961,1,0],RGBColor[1,0.501961,1],
  RGBColor[1,0.501961,0]}]
```

Options for 3D graphs: most 2D graph options are valid for `Plot3D`. Here we present some useful special options for 3D graphs:

```
f[x_,y_]:=Exp[-(x+y)]; Plot3D[f[x,y],{x,-Pi,2*Pi},
 {y,-Pi,2*Pi},PlotPoints->{25,40},Filling->Bottom]
Plot3D[f[x,y],{x,-Pi,2*Pi},{y,-Pi,2*Pi},
 Mesh->False,Boxed->False]
Plot3D[f[x,y],{x,-Pi,2*Pi},{y,-Pi,2*Pi},Mesh->8,
 MeshShading->{{Orange,Green},{Brown,Yellow}},
 PlotRange->All]
Plot3D[f[x,y],{x,-Pi,2*Pi},{y,-Pi,2*Pi},Mesh->8,
 MeshStyle->Gray]
Plot3D[f[x,y],{x,-Pi,2*Pi},{y,-Pi,2*Pi},BoxRatios->{1,2,1}]
Plot3D[f[x, y],{x,-Pi,2*Pi},{y,-Pi,2*Pi},FaceGrids->All,
 Axes->False]
Plot3D[f[x, y],{x,-Pi,2*Pi},{y,-Pi,2*Pi},
 PlotStyle->Glow[White],Lighting->"Neutral"]
Plot3D[f[x,y],{x,-Pi,2*Pi},{y,-Pi,2*Pi},
 PlotStyle->FaceForm[],Lighting->"Neutral"]
Plot3D[f[x,y],{x,-Pi,2*Pi},{y,-Pi,2*Pi},ViewPoint->{-1,2,1}]
```

In Mathematica, the 3D graphs can be examined from any viewpoint using real-time 3D manipulation (with the mouse) and various options such as `ViewPoint`, `ViewCenter`, `ViewVertical`, `ViewVector`, `ViewRange`, `ViewAngle` (the `3DViewPointSelector` has been eliminated in *Mathematica* 6).

The global options for 2D and 3D graphs:

Maple:

```
with(plots);      setoptions(NameOpt1=val1,...,NameOptn=valn);
                setoptions3d(NameOpt1=val1,...,NameOptn=valn);
```

```
with(plots); setoptions(axes=boxed,title="graph of f(x)");
       setoptions3d(axes=normal,title="graph of g(x,y)");
f:=x->x^2*sin(x^2); g:=(x,y)->sin(x^2+y^2);
plot(f(x),x=-Pi..Pi); plot3d(g(x,y),x=-Pi..Pi,y=-Pi..Pi);
```

Mathematica:

```
SetOptions[symb,NameOpt1->val1,...,NameOptn->val1]
```

```
SetOptions[Plot,Frame->True,PlotLabel->"f[x]"]
SetOptions[Plot3D,Boxed->False,PlotLabel->"f[x,y]"]
Plot[x^2*Sin[x^2],{x,-Pi,Pi}]
Plot3D[Sin[x^2+y^2],{x,-Pi,Pi},{y,-Pi,Pi}]
```

3.3 Multiple Graphs

A list or a set of graphs in the same figure.

Maple:

```
L1:=[f1(x),...,fn(x)]:   L2:=[F1,...,Fn]:
S1:={f1(x),...,fn(x)}:   S2:={F1,...,Fn}:     plot(L1,x=a..b);
plot(S1,x=a..b);         plot(L2,x=a..b);     plot(S2,x=a..b);
```

Here `L1,L2` and `S1,S2` are the lists and sets of functions and expressions, respectively.

```
f:=x->sin(x)/x; g:=x->cos(x)/x;
plot([f(x),g(x)],x=0..10*Pi,y=-1..2,
 linestyle=[SOLID,DOT],color=[red,blue]);
```

Mathematica:

```
Plot[{f1[x],f2[x],...,fn[x]},{x,x1,x2}]
GraphicsGrid[{{Plot[f11...],...,Plot[f1m...]},
           ...{Plot[fn1...],...,Plot[fnm...]}}]
                    GraphicsRow[{Plot[f1...],...,Plot[fn...]}]
                 GraphicsColumn[{Plot[f1...],...,Plot[fn...]}]
```

```
Plot[{Sin[x]/x,Cos[x]/x},{x,0,10*Pi},
 PlotRange->{{0,10*Pi},{-1,2}}]
GraphicsGrid[{{Plot[Cos[x],{x,-Pi,Pi}],Plot[Sin[x],
 {x,-Pi,Pi}]}}]
GraphicsGrid[{{Plot[Cos[x],{x,-Pi,Pi}]},{Plot[Sin[x],
 {x,-Pi,Pi}]}}]
GraphicsRow[{Plot[Cos[x],{x,-Pi,Pi}],Plot[Sin[x],{x,-Pi,Pi}]}]
GraphicsColumn[{Plot[Cos[x],{x,-Pi,Pi}],Plot[Sin[x],
 {x,-Pi,Pi}]}]
```

Merging various saved graphic objects, an array of graphic objects.

Maple:

```
with(plots):                L1:=[G1,...,Gn]:     L2:=[H1,...,Hn]:
S1:={G1,...,Gn}:            S2:={H1,...,Hn}:  display(L1,x=a..b);
display(S1,x=a..b);  display3d(L2,x=a..b);  display(S2,x=a..b);
G:=array(1..n): G[i]:=plot(fun[i],x=a..b):          display(G);
```

Here `L1,L2` and `S1,S2` are the lists and sets of saved 2D and 3D graphs, respectively.

```
with(plots);  f:=x->abs(sin(x));  g:=x->-cos(x);
G1:=plot(f(x),x=-Pi..Pi): G2:=plot(g(x),x=-Pi..Pi):
display({G1,G2},title="f(x) and g(x)");
G:=array(1..2);  G[1]:=plot(sin(x),x=-Pi..Pi):
G[2]:=plot(cos(x),x=0..Pi): display(G);
```

Mathematica:

```
Show[{g1,...,gn}]
                 GraphicsGrid[{{g11,...,g1m},...{gn1,...,gnm}}]
GraphicsRow[{g1,...,gn}]            GraphicsColumn[{g1,...,gn}]
ListPlot[{list1,list2...}]       ListLinePlot[{list1,list2...}]
```

```
f1[x_]:=Abs[Sin[x]]; f2[x_]:=-Cos[x];
g1=Plot[f1[x],{x,-Pi,Pi}]; g2=Plot[f2[x],{x,-2*Pi,2*Pi}];
Show[{g1,g2},Frame->True,PlotLabel->"f1 and f2",
 AspectRatio->1,PlotRange->All]
GraphicsGrid[{{g1,g2}},Frame->True,AspectRatio->1]
GraphicsGrid[{{g1},{g2}},Frame->True,Frame->All]
GraphicsRow[{g1,g2},Background->RGBColor[0.5,1.,1.]]
GraphicsColumn[{g1,g2},Alignment->Top]
list1=Table[{i,i^3},{i,-5,5}]; list2=Table[{i,i^2},{i,-5,5}];
g11=ListPlot[{list1,list2}]; g12=ListLinePlot[{list1,list2}];
GraphicsColumn[{g11,g12}]
g3=Plot3D[2*x^2-3*y^2,{x,-1,1},{y,-1,1}]; g4=Plot3D[3*x+y,
 {x,-1,1},{y,-1,1}]; Show[g3,g4,BoxRatios->{1,1,2}]
```

Problem 3.1 Graph the stability diagram of the Mathieu differential equation $x'' + [a - 2q\cos(2t)]x = 0$ in the (a, q)-plane (see Sect. 9.4).

Maple:

```
with(plots):
S1:=[MathieuA(i,q) $ i=0..5]; S2:=[MathieuB(i,q) $ i=1..6];
G1:=plot(S1,q=0..35,-20..45,color=red):
G2:=plot(S2,q=0..35,-20..45,color=blue):
display({G1,G2},thickness=3);
```

Mathematica:

```
s1=Table[MathieuCharacteristicA[i,q],{i,0,5}]
s2=Table[MathieuCharacteristicB[i,q],{i,1,6}]
g1=Plot[Evaluate[s1],{q,0,35},PlotStyle->Red];
g2=Plot[Evaluate[s2],{q,0,35},PlotStyle->Blue];
Show[{g1,g2},PlotRange->{{0,35},{-20,45}},Frame->True,
 PlotLabel->"Stability Regions",Axes->False,AspectRatio->1]
```

□

3.4 Text in Graphs

Drawing text strings on 2D and 3D graphs.

Maple:

```
textplot([[x1,y1,str1],..,[xn,yn,strn]],ops);
          textplot3d([[x1,y1,z1,str1],..,[xn,yn,zn,strn]],ops);
```

```
with(plots): f:=x->4*x^3+6*x^2-9*x+2;
G1:=plot([f(x),D(f)(x),(D@@2)(f)(x)],x=-3..3):
G2:=textplot([1.2,100,"f(x) and derivatives"],
    font=[HELVETICA,BOLD,13],color=plum): display([G1,G2]);
P1:=plot3d(exp(-(x^2+y^2)),x=-6..6,y=-6..6,grid=[25,25]):
P2:=plot3d(-5-4*sin(sqrt(x^2+y^2)),x=-6..6,y=-6..6):
P3:=textplot3d([1,1,2,"a"],font=[SYMBOL,25],color=blue):
display3d([P1,P2,P3],orientation=[34,79]);
```

Mathematica:

```
Graphics[Text["string",{xt,yt}],BaseStyle->val,ops]
         Graphics3D[Text["string",{xt,yt,zt}],FormatType->val]
         Text[Grid[{{"s11",...,"s1m"},...,{"sn1",...,"snm"}}]]
```

```
f:=4*x^3+6*x^2-9*x+2; fD1=D[f,x]; fD2=D[f,{x,2}];
g11=Plot[{f,fD1,fD2},{x,-3,3},PlotStyle->{Red,Blue,Green},
 BaseStyle->{FontFamily->"Helvetica",FontSlant->"Italic",
 FontWeight->"Bold",FontSize->15}];
g12=Graphics[Text["f[x] and derivatives",{0,50}]];
Show[{g11,g12},Frame->True,Axes->False,PlotRange->All]
g31=Plot3D[Exp[-(x^2+y^2)],{x,-6,6},{y,-6,6},BaseStyle->
 {FontFamily->"Symbol",FontSize->25},PlotRange->All];
g32=Plot3D[-5-4*Sin[Sqrt[x^2+y^2]],{x,-6,6},{y,-6,6},
 PlotRange->All]; g33=Graphics3D[Text["a",{0,9,0}]];
Show[{g31,g32,g33}]
g41=Graphics3D[Table[With[{p={i,j,k}/3},{RGBColor[p*3],
 Cuboid[p,p+.32]}],{i,3},{j,3},{k,3}],BaseStyle->
 {FontFamily->"Ariel",FontSize->25}]; g42=Graphics3D[
 Text["Rubik's Cube",{1,1,1}]]; Show[{g41,g42}]
g43=Graphics3D[Text[Grid[{{"X","0","X"},{"0","X","X"},
 {"0","0","X"}}],{1,1,1}]]; Show[{g41, g43}]
```

3.5 Special Graphs

Grid lines for 2D graphs

Maple:

```
plot(f(x),x=x1..x2,gridlines=true);
with(plots):       conformal(z,z=z1..z2,ops):
                   coordplot(coordsystem,[xrange,yrange],ops);
```

Mathematica:

```
Plot[f[x],{x,x1,x2},GridLines->{{xGL},{yGL}},GridLinesStyle->s]
```

Problem 3.2 Plot $f(x) = x\sin(1/x)$ and $g(x) = \dfrac{x^2 - x + 1}{x^2 + x - 1}$ together with the corresponding grid lines.

Maple:

```
with(plots): A:=4: f:=x->sin(1/x)*x;
plot(f(x),x=-Pi/2..Pi/2,color=blue,thickness=3,gridlines=true):
G1:=plot(f(x),x=-Pi/2..Pi/2,color=blue,thickness=3):
M1:=conformal(z,z=-A-I..A+I,grid=[20,10],color=grey):
display([G1,M1]); g:=x->(x^2-x+1)/(x^2+x-1); R:=-5..5;
G2:=plot(g(x),x=R,R,discont=true,thickness=3):
M2:=coordplot(cartesian,[R,R],view=[R,R],grid=[10,10],
 color=[grey,grey]): display([G2,M2],axes=boxed,
 scaling=constrained,xtickmarks=5,ytickmarks=5);
```

Mathematica:

```
f[x_]:=x*Sin[1/x]; g[x_]:=(x^2-x+1)/(x^2+x-1);
Plot[f[x],{x,-Pi/2,Pi/2},Frame->True,Axes->False,PlotStyle->
 {Blue,Thickness[0.01]},GridLines->{Automatic,Automatic}]
Plot[g[x],{x,-5,5},Frame->True,Axes->False,PlotStyle->
 {Blue,Thickness[0.007]},GridLines->{Automatic,Automatic}]
Plot[g[x],{x,-5,5},Frame->True,PlotStyle->{Blue,Thickness[0.01]},
 GridLines->{Automatic,Automatic},Exclusions->{x^2+x-1==0}]
```

□

Bounded regions for 2D graphs

Maple:

```
plot(f(x),x=a..b,filled=true);
with(plots):                     inequal(ineqs,x=a..b,y=c..d,ops);
```

More information about the function `inequal` see `?plot[options]`.

Mathematica:

```
Plot[f1[x],...,{x,x1,x2},Filling->val]
               RegionPlot[Ineq,{x,x1,x2},{y,y1,y2}]
               RegionPlot3D[Ineq,{x,x1,x2},{y,y1,y2},{z,z1,z2}]
```

Here `value` can take various forms, e.g. `Top`, `Bottom`, `Axis`, `{i}`, `{i->{j}}`, etc. Various bounded regions for 2D (and 3D) graphs can be constructed with this new functions `RegionPlot`, `RegionPlot3D`, and the new option `Filling`.

```
Plot[{Cos[x],Cos[2*x],Cos[3*x]},{x,-Pi/2, Pi/2}]
Plot[{Cos[x],Cos[2*x],Cos[3*x]},{x,-Pi/2,Pi/2},
 Filling->{1->{{2},GrayLevel[0.6]},{2->{{3},Red}}}]
```

Problem 3.3 Plot the region bounded by $f(x) = -(x+1)^2 + 10$ and the axis $x = 0$.

Maple:

```
with(plots): f:=x->-(x+1)^2+10: S:=[fsolve(f(x)=0,x)];
f1:=plot([f(x),0],x=-4.5..4.5,-2..10,thickness=3):
f2:=plot(f(x),x=S[1]..S[2],filled=true,color=grey):
display([f1,f2]);
```

Mathematica:

```
f[x_]:=-(x+1)^2+10; S=NSolve[f[x]==0,x]
Plot[{f[x],0},{x,S[[1,1,2]],S[[2,1,2]]},PlotStyle->
 {{Red,Thickness[0.01]},{Green,Thickness[0.01]}},
 Filling->{1->{0,GrayLevel[0.6]}}]
```

□

Problem 3.4 Plot the region that satisfies the inequality $2x - 2y > 1$.

Maple:

```
with(plots): Ineq:=x->2*x-2*y>1; A:=(color=blue);
B:=(color=grey); C:=(color=green,thickness=10);
inequal(Ineq(x),x=-2..2, y=-2..2,
     optionsfeasible=A,optionsexcluded=B,optionsopen=C);
```

Mathematica:

```
RegionPlot[2*x-2*y>1,{x,-2,2},{y,-2,2},BoundaryStyle->
 {Green,Thickness[0.02]},PlotStyle->{Blue},
 Background->GrayLevel[0.6]]
```

□

Logarithmic plots in the plane

Maple:

```
with(plots):                                  logplot(f,range,ops);
semilogplot(f,range,ops);                  loglogplot(f,range,ops);
```

```
with(plots); with(stats):
al:=stats[random, normald](20);
Points:=[seq([0.2*i,exp(0.1*i)+0.1*al[i]],i=1..20)];
G1:=logplot(Points, style=point, color=green):
G2:=logplot(x+sin(x),x=0.5..3,style=line,color=red):
display({G1,G2}); f:=x->x^5+exp(-x^5);
loglogplot(f(x),x=0.1..100,axes=boxed);
```

Mathematica:

```
LogPlot[f,{x,x1,x2}]             LogPlot[{f1,...,fn},{x,x1,x2}]
LogLogPlot[f,{x,x1,x2}]       LogLogPlot[{f1,...,fn},{x,x1,x2}]
LogLinearPlot[f,{x,x1,x2}]LogLinearPlot[{f1,...,fn},{x,x1,x2}]
ListLogPlot[{{x1,y1},{x2,y2},...}]    ListLogPlot[{l1,...,ln}]
ListLogLinearPlot[{{x1,y1},...}]   ListLogLinearPlot[{l1,...}]
ListLogLogPlot[{{x1,y1},{x2,y2},...}] ListLogLogPlot[{l1,...}]
```

```
al=Table[Random[NormalDistribution[]],{20}]
points=Table[{0.2*i,Exp[0.1*i]+0.1*al[[i]]},{i,1,20}]
g1=ListLogPlot[points,PlotStyle->{PointSize[0.02],
 Hue[0.7]}]; g2=LogPlot[x+Sin[x],{x,0.5,3},PlotStyle->Red];
f[x_]:=x^5+Exp[-x^5]; Show[{g1,g2}]
LogLogPlot[Evaluate[N[f[x],30]],{x,0.1,100}]
```

Plots of piecewise functions

Maple:

```
f:=proc(x)  if cond1  then expr1  else expr2 end if;  end proc;
plot('f(x)',x=a..b,ops);                        plot(f,a..b,ops);
with(plots):    f:=x->piecewise(cond1,expr1,cond2,expr2,expr3);
                                             plot(f(x),x=a..b,ops);
```

```
with(plots): setoptions(thickness=5);
f:=proc(x) if x<0 then 0 elif x<1 then x else 1 fi; end;
plot('f(x)',x=-2..2); plot(f,-2..2);
g:=x->piecewise(x<0,0,x<1,x,1);   plot(g(x),x=-2..2);
```

Mathematica:

```
f1[x_]:=var1/;cond1; ... fn[x_]:=varn/;condn; ...
f2[x_]:=Piecewise[{{val1,cond1},...,{valn,condn}}];
f3[x_]:=UnitStep[x];          Plot[{f1[x],f2[x],f3[x]},{x,a,b}]
```

```
f1[x_]:=0/;x<0; f1[x_]:=x/;0<=x<=1; f1[x_]:=1/;x>1;
f2[x_]:=Piecewise[{{0,x<0},{x,0<=x<=1},{1,x>1}}];
f3[x_]:=x*UnitStep[x]-x*UnitStep[x-1]+UnitStep[x-1];
GraphicsColumn[{Plot[f1[x],{x,-2,2},PlotStyle->Hue[0.5]],
 Plot[f2[x],{x,-2,2},PlotStyle->Hue[0.7]],
 Plot[f3[x],{x,-2,2},PlotStyle->Hue[0.9]]}]
```

Density plots

A density plot of the function $f(x, y)$, $(x, y) \in [x_1, x_2] \times [y_1, y_2]$.

Maple:

```
with(plots):        densityplot(f(x,y),x=x1..x2,y=y1..y2,ops);
```

Mathematica:

```
DensityPlot[f[x,y],{x,x1,x2},{y,y1,y2},ops]
          ListDensityPlot[{{a11,...,a1n},...{an1,...,ann}},ops]
```

Problem 3.5 Construct the density plot of $f(x,y) = xe^{-x^2-y^2}$, $(x,y) \in [-2,2] \times [-2,2]$, with color gradient and the corresponding legend.

Maple:

```
with(plots):
A:=colorstyle=HUE,style=patchnogrid,numpoints=5000,axes=boxed;
G1:=densityplot((x,y)->x*exp(-x^2-y^2),-2..2,-2..2,A):
G2:=densityplot((x,y)->0.2*y,3..3.5,-2..2,A):
G3:=textplot([seq([3.8,-1.95+i/8*3.9,
 sprintf("%.1f",-0.4+i/10)],i=0..8)]):
display({G1,G2,G3},scaling=constrained);
```

Mathematica:

```
f1[x_,y_]:=x*Exp[-x^2-y^2]; f2[x_,y_]:=0.2*y;
g1=DensityPlot[Evaluate[f1[x,y]],{x,-2,2},{y,-2,2},
 ColorFunction->Hue,PlotPoints->100,PlotRange->All];
g2=DensityPlot[Evaluate[f2[x,y]],{x,3,3.5},{y,-2,2},
 ColorFunction->Hue,PlotPoints->100,AspectRatio->Automatic,
 PlotRange->All]; g3=Graphics[Table[Text[StyleForm[
 NumberForm[N[-0.4+i/10],2],FontSize->10],{2.7,-1.95+i/8*3.9}],
 {i,0,8}],PlotRange->All]; Show[{g1,g2,g3}]
list1=Table[Random[Real,{1,10}],{x,1,10},{y,1,10}];
ListDensityPlot[list1,PlotRange->{{1,8},{1,8}}]
```

□

Bar graphs: different types of 2D and 3D bar graphs

Maple:

```
with(stats): with(stats[statplots]):       histogram(data,ops);
with(Statistics):                      Histogram([data1,...],ops);
BarChart([data1,...,],ops);          ColumnGraph([data1,...],ops);
LineChart([data1,...],ops);             PieChart([data1,...,],ops);
                                     PieChart['interactive'](data);
```

Mathematica:

```
<<BarCharts`                        <<PieCharts`       <<Histograms`
BarChart[{data1,...},ops]     StackedBarChart[{data1,...},ops]
PercentileBarChart[{data1,...},ops]     PieChart[{y1,...},ops]
Histogram[{x1,...},ops]   GeneralizedBarChart[{data1,...},ops]
                        BarChart3D[{{z11,...},{z21,...},...},ops]
Histogram3D[{{x1,y1},...},ops]
```

Problem 3.6 Compare different bar graphs.

Maple:

```
with(stats): with(stats[statplots]):
list1:=[random[normald](200)]:
histogram(list1); histogram(list1,area=count);
histogram(list1,color=blue);
with(Statistics): NX:=RandomVariable(Normal(0,1));
G1:=DensityPlot(NX,range=-3..3,thickness=3,color=red):
list1:=Sample(Normal(0,1),1500);
G2:=Histogram(list1,range=-3..3,color=blue):
plots[display]({G1,G2});
list2:=[i^2 $ i=1..5]; PieChart(list2,sector=0..360,
 color=[blue,red,green,yellow,white]);
BarChart(list2); ColumnGraph(list2); LineChart(list2);
```

Mathematica:

```
<<BarCharts`; <<PieCharts`; <<Histograms`;
list1=Table[Random[Integer,{1,5}],{i,1,5}];
list2=Table[i^2,{i,1,5}]; list3={"a1","a2","a3","a4","a5"};
g1=BarChart[list1,BarStyle->{Hue[0.7]},BarEdges->True,
 BarOrientation->Horizontal]; g2=BarChart[{list1,list2},
 BarStyle->{Hue[0.5],Hue[0.8]}]; g3=StackedBarChart[
 {list1,list2},BarStyle->{Hue[0.5],Hue[0.8]}];
g4=PercentileBarChart[{list1,list2},BarLabels->list3,
 BarStyle->{Hue[0.5],Hue[0.6]},BarSpacing->0.2];
g5=PieChart[list1,PieLabels->list3,PieExploded->All,
 PieStyle->{Hue[0.5],Hue[0.6],Hue[0.75],Hue[0.85],Hue[0.9]}];
g6=PieChart[list1,PieLabels->list3,PieExploded->{1},
 PieStyle->{Hue[0.5],Hue[0.6],Hue[0.75],Hue[0.85],Hue[0.9]}];
GraphicsGrid[{{g1,g2},{g3,g4},{g5,g6}}]
```

```
list4={{1,2,3},{6,5,4},{9,10,11}}; BarChart3D[list4,
 AxesLabel->{"x","y","z"},BarSpacing->{0.3,0.3},
 BarStyle->Hue[Random[]]]
n=9; list5=Table[PDF[NormalDistribution[n,3],x],{x,0,2*n}];
list6=Table[Hue[0.85],{2*n+1}]; list6[[10]]=Blue;
BarChart[list5,BarLabels->Range[0,2*n],BarStyle->list6]
```

In *Mathematica 7*, several new functions for automated dynamic charting have been developed and added to the *Mathematica* kernel, e.g. `BarChart`, `PieChart`, `BubbleChart`, `SectorChart`, `BarChart3D`, `PieChart3D`, `Histogram`, `Histogram3D`, etc. The packages `BarCharts`, `PieCharts`, and `Histograms` are no longer needed. We mention here some examples:

```
f[{{x1_,x2_},{y1_,y2_}},___]:=Rectangle[{x1,y1},
 {x2,y2}]; l1=Table[Random[Integer,{1,5}],{i,1,5}];
l2=Table[i^2,{i,1,5}]; BarChart[{l1,l2},
 ChartElementFunction->f, ChartLegends->{"A","B"}]
PieChart[l1,ChartElementFunction->"GlassSector",
 ChartStyle->"Pastel"]
PieChart3D[l1,ChartStyle->"Pastel"]
Histogram[RandomReal[NormalDistribution[0,1],1000]]
```

□

3.6 Animations

2D and 3D animations of functions

Maple:

```
with(plots):                        animatecurve(f(x),x=a..b,ops);
                               animate(f(x,t),x=a..b,t=t1..t2,ops);
                  animate3d(f(x,y,t),x=a..b,y=c..d,t=t1..t2,ops);
display([G1,...,GN],insequence=true);
display3d([G1,...,GN],insequence=true);
```

Mathematica:

2D and 3D animations of plot sequences can be produced using the new function `Animate`, where the speed, the direction, and the pause of animation can be controlled by the corresponding buttons (also see Sect. 2.3.7).

```
Animate[exprPlot,{t,t1,t2,tStep},ops]
```

```
f[x_,t_]:=Sin[x+t]+Sin[x-2*t];
Animate[Plot[Sin[-(x-t)^2],{x,-2,2},PlotRange->{-1,1}],
 {t,0.2,0.5}]
Animate[Plot[f[x,t],{x,-10,10},PlotRange->{-10,10}],
 {t,0,30,0.001}]
Animate[Plot3D[Sin[x-y-t],{x,0,Pi},{y,0,Pi},PlotRange->All],
 {t,0,2*Pi}]
Animate[Plot[x*Sin[x*t],{x,-2*Pi,0},PlotRange->{-6,6}],
 {t,0.2,0.5}]
Animate[Plot[Sin[x*i],{x,0,Pi},PlotRange->{-1,1}],{i,1,10}]
```

Problem 3.7 Show that the solutions $u_1(x,t)=\cos(x-2t)$ and $u_2(x,t)=\cos(x+2t)$, of the wave equation are traveling waves.

Maple:

```
with(plots): N:=200;  A:=array(1..2):
u1:=(x,t)->cos(x-2*t); u2:=(x,t)->cos(x+2*t);
setoptions(thickness=3,scaling=constrained,axes=none);
A[1]:=animate(u1(x,t),x=0..4*Pi,t=1..10,frames=N):
A[2]:=animate(u2(x,t),x=Pi/2..9*Pi/2,t=1..10,
      color=green,frames=N):  display(A);
```

Mathematica:

```
u1[x_,t_]:=Cos[x-2*t]; u2[x_,t_]:=Cos[x+2*t];
SetOptions[Plot,Axes->None,Frame->False,
 PlotStyle->Thickness[0.02],AspectRatio->1];
Animate[GraphicsRow[{Plot[u1[x,t],{x,0,4*Pi},PlotStyle->Red],
 Plot[u2[x,t],{x,Pi/2,9*Pi/2},PlotStyle->Blue]}],{t,1,5}]
```

□

Problem 3.8 Observe the Lissajous curves in polar coordinates.

Maple:

```
with(plots):
animatecurve([sin(7*x),cos(11*x),x=0..2*Pi],coords=polar,
   numpoints=300,frames=300,color=blue,thickness=3);
```

Mathematica:

```
m=70; n=3; g={};
l1=Table[{Sin[n*x],Cos[(n+2)*x]},{x,0,2*Pi,1/m}]//N;
k=Length[l1]; Do[g=Append[g,Evaluate[ListPolarPlot[
 Take[l1,{1,i}],PlotMarkers->{"o",10},
 PlotRange->{{-1,1},{-1,1}},AspectRatio->1,
 PlotStyle->Blue]]],{i,1,k}]; ListAnimate[g]
Animate[PolarPlot[{Sin[Prime[n]*x],Cos[Prime[n+1]*x]},
 {x,0,2*Pi},PlotPoints->40,PlotStyle->Blue,Frame->True,
 FrameTicks->False,MaxRecursion->3],{n,1,20,2},
 AnimationRate->2]
```

In *Mathematica* (*ver* ≥ 6), some of the functions are located in the *Wolfram Research Website* and are available by downloading the corresponding packages from it. Here we put one example of the function `MovieParametricPlot` that is available in the legacy standard Add-On package `Animation` (for *ver* $= 5.2$ and updated for *ver* $= 6$). This package can be downloaded from the *Wolfram Research Website* according to the corresponding path in your computer, e.g. `Mathematica` → 6.0 → `AddOns` → `LegacyPackages` → `Graphics`. The first line in the solution we recommend for turning off the obsolescence messages.

```
Off[General::obspkg]; Off[General::newpkg]; Off[General::wrsym];
<<Graphics`Animation`
MovieParametricPlot[{Sin[n*t],Cos[(n+2)*t]},{t,0,2*Pi},
 {n,1,5,2},PlotStyle->Blue,Frame->True,FrameTicks->False]
```

□

Problem 3.9 Animations of two sequences of points in 2D.

Maple:

```
with(plots): n:=100: G:=[]:
L1:=[seq([cos(j*Pi/n),sin(j*Pi/n)],j=0..n)]:
L2:=[seq([-cos(j*Pi/n),-sin(j*Pi/n)],j=0..n)]:
for i from 1 to n do
 G:=[op(G),plot([L1[1..i],L2[1..i]], x=-1..1,y=-1..1,
     symbol=circle,style=point,color=[blue,red])]: od:
display(G,insequence=true);
```

Mathematica:

```
n=100; g={};
l1=Table[{Cos[j*Pi/n],Sin[j*Pi/n]},{j,0,n}]//N;
l2=Table[{-Cos[j*Pi/n],-Sin[j*Pi/n]},{j,0,n}]//N;
Do[g=Append[g,Evaluate[ListPlot[{Take[l1,{1,i}],
 Take[l2,{1,i}]},PlotMarkers->{"o",9},AspectRatio->1,
 PlotStyle->{Blue,Red},PlotRange->{-1,1}]]],{i,1,n}];
ListAnimate[g]
```

□

Problem 3.10 Animations of two sequences of points in 3D.

Maple:

```
with(plots): n:=250: G:=[]: k:=20:
L1:=[seq([cos(k*j*Pi/n),j,sin(k*j*Pi/n)],j=0..n)]:
L2:=[seq([-cos(k*j*Pi/n),-j,-sin(k*j*Pi/n)],j=0..n)]:
for i from 1 to n do
  G:=[op(G),spacecurve({L1[1..i],L2[1..i]},style=line,
     axes=none,thickness=3,shading=zhue)]: od:
display3d(G,insequence=true);
```

Mathematica:

```
n=25; m=150; g={}; k=20;
Animate[ParametricPlot3D[{{Cos[k*j*t/m],t*j/m,Sin[k*t*j/m]},
 {-Cos[k*j*t/m],-t*j/m,-Sin[k*j*t/m]}},{t,0,2*Pi},
 PlotPoints->70,MaxRecursion->1,Boxed->False,Axes->False,
 ColorFunction->Function[{x,y},Hue[x]],PlotStyle->
 {{Thickness[0.02]},{Thickness[0.02]}},PlotRange->
 {{-1,1},{-1,1},{-1,1}}],{j,0,n},AnimationRate->0.9]
n=250; g={}; k=20;
l1=Table[{Cos[k*j*Pi/n],j/n,Sin[k*j*Pi/n]},{j,0,n}]//N;
l2=Table[{-Cos[k*j*Pi/n],-j/n,-Sin[k*j*Pi/n]},{j,0,n}]//N;
Do[g=Append[g,Evaluate[GraphicsRow[{ListPointPlot3D[
 Take[l1,{1,i}],ColorFunction->Function[{x,y,z},Hue[z]],
 Axes->False,BoxRatios->{1,1,1},Boxed->False,
 PlotStyle->{{PointSize[0.05]},{PointSize[0.05]}},
 PlotRange->{{-1,1},{-1,1},{-1,1}}],ListPointPlot3D[
 Take[l2,{1,i}],ColorFunction->Function[{x,y,z},Hue[z]],
 Axes->False,BoxRatios->{1,1,1},Boxed->False,
 PlotStyle->{{PointSize[0.05]},{PointSize[0.05]}},
 PlotRange->{{-1,1},{-1,1},{-1,1}}]}]]],{i,1,n}];
ListAnimate[g]
```

□

Problem 3.11 Animations of two structures in 3D.

Maple:

```
with(plots):  n:=40: G:=NULL:
f1:=(x,y)->(x^2-y^2)/9-9: f2:=(x,y)->(x^2-y^2)/9:
G1:=plot3d(f1(x,y),x=-9..9,y=-9..9,style=patchnogrid):
G2:=plot3d(f2(x,y),x=-9..9,y=-9..9,style=patchnogrid):
for i from 1 to n do
 G:=G,display3d([G1,G2],orientation=[i*180/n,50]): od:
display3d([G],scaling=constrained,insequence=true);
```

Mathematica:

```
n=30; g={}; f1[x_,y_]:=(x^2-y^2)/9-9; f2[x_,y_]:=(x^2-y^2)/9;
g1=Plot3D[f1[x,y],{x,-9,9},{y,-9,9},Mesh->False];
g2=Plot3D[f2[x,y],{x,-9,9},{y,-9,9},Mesh->False];
Do[g=Append[g,Show[{g1,g2},SphericalRegion->True,
 ViewCenter->{0.5,0.5,0.5},ViewPoint->{i/10,2,2},
 Boxed->False,Axes->False,PlotRange->{{-10,10},
 {-10,10},{-10,10}}]],{i,-n,n}];
ListAnimate[g,AnimationRate->4]
```

□

Problem 3.12 Observe the motion of a point rolling along a curve.

Maple:

```
with(plots):
G:=plot(cos(x),x=-Pi..Pi,axes=boxed,thickness=2,color=blue):
animate(pointplot,[[[t,cos(t)]],symbol=circle,symbolsize=20,
        color=red],t=-Pi..Pi,frames=100,background=G);
```

Mathematica:

```
g=Plot[Cos[x],{x,-Pi,Pi},AspectRatio->1,Frame->True,
  PlotStyle->{Blue,Thickness[0.02]}];
Animate[Show[g,Graphics[{Purple,Disk[{x,Cos[x]},0.1]}],
  PlotRange->{{-Pi,Pi},{-1.5,1.5}}],{x,-Pi,Pi}]
```

□