

Mathematics made anew
Volume 1

MuPAD - *A Practical Guide*

Kai Gehrs
Frank Postel

Tools and Texts for Computer Aided Learning



Kai Gehrs · Frank Postel

MuPAD – A Practical Guide

Mathematics Made Anew:
Tools and Texts for Computer Aided Learning

SciFace Software

Kai Gehrs
Frank Postel
University of Paderborn
AutoMATH Institut
Warburger Straße 100
33095 Paderborn
schule@mupad.de

MuPAD – A Practical Guide / by Kai Gehrs and Frank Postel.
Mathematics Made Anew: Tools and Texts for Computer Aided Learning, vol 1. Paderborn: SciFace Software GmbH & Co. KG, November 2003.

First German Edition: February 2001
Second German Edition: June 2002, revised and extended edition
English Translation of Second German Edition: November 2003

SciFace Software GmbH & Co. KG Paderborn

All rights reserved. This work may not be translated or copied in whole or in part in any form without the written permission of SciFace Software GmbH & Co. KG.

© 2003 SciFace Software GmbH & Co. KG, Paderborn

The use of general descriptive names, trade names, trademarks, etc. in this publication, even if the former are not especially identified, is not to be taken as a sign that such names, as understood by the Trade Marks and Merchandise Marks Act, may accordingly be used freely by anyone.

Cover Design: SciFace Software GmbH & Co. KG, Paderborn

Preface to the First Edition

The development of MuPAD began back in 1990 at the University of Paderborn with a research project on solutions of specific problems occurring in the area of dynamical systems. Rather soon MuPAD evolved into a universal tool for symbolic and exact computing as well as for numerical calculations. Ere long MuPAD provided for high quality two- and three-dimensional graphical presentations of complex mathematical issues that can be visualized and interactively manipulated.

Already in 1993 MuPAD received the “German-Austrian Hochschul-Software Preis” from the Forschungsgemeinschaft; 1994 MuPAD was decorated with the “European Academic Software Award.”

Outsourcing part of the MuPAD research team in February 1997, SciFace Software GmbH & Co. KG was founded to continually develop MuPAD and meet increased customer demand by creating modern user interfaces and the various multilingual documentation.

By an ongoing integration of academic research of the University team, SciFace Software transformed MuPAD into a competitive computer algebra system. As a result of this successful joint endeavor of SciFace Software with the University of Paderborn the enterprise was awarded with the “Förderpreis des Technologie Forum Paderborn e.V.” for outstanding cooperation of academic and commercial venture.

Amongst other fields of application, MuPAD is used in schools from junior high school level up to university level, serving for teaching and research purposes. Increasingly, MuPAD is used in high schools supporting and extending the instruction of mathematics.

The strong demand and various activities of SciFace Software in the field of math-instruction are reflected by close cooperate efforts with prominent German publishing companies. These joint efforts provide for teaching software and solutions for an interactive and exploring approach to mathematical issues and promote web-based learning.

We greatly appreciate your questions and comments regarding “MuPAD for teaching.” Your participation in the development of a modern mathematical software is highly valued. Please send questions, suggestions and comments to schule@mupad.de.

Paderborn, June 2001
Dr. Andreas Sorgatz,
SciFace Software

About this Booklet

This booklet originated in a series of MuPAD-Notebooks used in MuPAD training courses aimed at high-school teachers. This has greatly influenced the style of writing: In problems and as well in the main course of this volume the reader is frequently asked for active participation. We therefore recommend to have MuPAD at reach while reading the book.

It is the intention of the authors to facilitate a fast and simple approach to the computer algebra system MuPAD, particularly aiming at the use of MuPAD for teaching purposes in schools from junior high school level up to university level. Each section in this volume will begin with a table listing the MuPAD commands newly introduced in that section. This table contains a brief survey for use and application. We confined ourselves to simple examples as our goal lies in mastering MuPAD not mathematics.

After a brief instruction on how to utilize the MuPAD Pro notebook-concept we demonstrate how to deal with problems occurring in calculus, linear algebra and stochastics using a computer algebra system. We place special emphasis on the capability of MuPAD to produce graphical output. This is a most important aspect of modern computer algebra systems, facilitating both for student and teacher a likewise intuitive and thorough presentation and understanding of complex mathematical issues. Thus students are enabled to develop the necessary intuitive and imaginative skills for their ongoing mathematical study.

In this booklet we used MuPAD Pro 2.0 which is available for Windows operating systems. The concept of notebooks which is elaborated on in the first chapter, however, is solely included in MuPAD Pro.

Still, all MuPAD input lines and their respective output are independent of the particular operating system chosen and can be worked with MuPAD 2.0 under Linux or MacOS.

Kai Gehrs
Frank Postel
Paderborn, June 2001

Preface to the Second Edition

In this second edition of the “Practical Guide to MuPAD” we corrected minor mistakes that had crept into the first edition. The meat of the first edition remained untouched, however, apart from slight modifications: Visualizing the binomial distribution in section 5.4 is much simpler now using the current version of MuPAD Pro 2.5 in contrast to version MuPAD 2.0 which this volume originally based on. Additionally, we added an example in section 6 to illustrate the use of the newly included 3d-viewer. MuPAD 2.5 complies continually with all functionality introduced in the first edition of this volume.

Kai Gehrs
Paderborn, May 2002

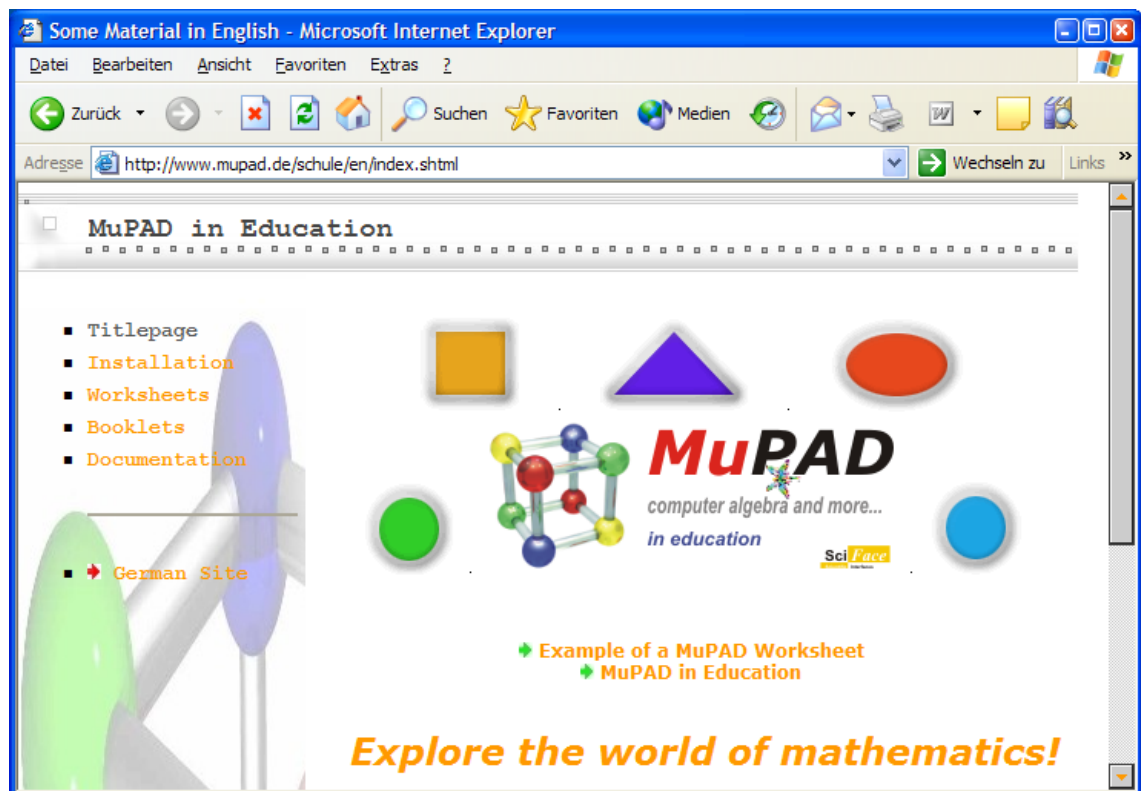
Many thanks to Stefan Huckemann for translating this booklet from the original German edition and also many thanks to Christopher Creutzig for his intensive TeX-nical support.

Kai Gehrs
Paderborn, November 2003

MuPAD in the Web

Also visit the web site of the "MuPAD in Education"-team at the University of Paderborn, Germany at:

<http://www.mupad.de/schule/en>



On technical question as well as on questions about using MuPAD in education in schools, colleges and universities you may contact our "MuPAD in Education"-team at the University of Paderborn, Germany via Email to:

<mailto:schule@mupad.de>

Contents

1	Working with MuPAD-Notebooks under Windows	3
1.1	What is a MuPAD-Notebook?	3
1.2	Loading and Saving Notebooks	4
1.3	Changing and Deleting within a Region	4
2	Basic Calculus	6
2.1	Computing with MuPAD	6
2.2	Standard Calculus	10
2.3	Definite and Indefinite Integration	14
2.4	An Extremal Problem	19
2.5	Working with MuPAD Libraries	21
2.6	Help, I Am Lost ...	22
2.7	Differential Equations - An Application	23
3	Linear Algebra	25
3.1	Creating Matrices	25
3.2	Row and Column Number of a Matrix	27
3.3	Accessing Matrix Entries	27
3.4	Elementary Matrix Calculus	29
3.5	More Tricks and Tips Creating Matrices	32
3.6	Coefficient Domains for Matrices	33
4	The linalg Library	35
4.1	Linear Dependence and Independence of Vectors	37
4.2	An Application from Analytic Geometry	39
5	Statistics and Probability	41
5.1	Statistical Computations	41
5.2	Simulating Simple Laplace Experiments	43
5.3	Relative Frequency and the Law of Numbers	44
5.4	Computing the Binomial Distribution	46
6	Plotting Made Easy with MuPAD	49
6.1	Function Graphs	49
6.2	Working with the plot Library: More Examples	53
6.3	An Application: Modeling of an Electric Bulb	57

6.4 3D-Graphics Gallery	63
-----------------------------------	----

1 Working with MuPAD-Notebooks under Windows

1.1 What is a MuPAD-Notebook?

A *MuPAD-Notebook* is a work-sheet containing several sections made up from mere text, mathematical formulas, MuPAD input, MuPAD output and graphical plots.

A dot • (usually in red color) preceding text indicates *MuPAD input*, a so called *input region*. If the cursor is in such a region then at the right hand side of the MuPAD Pro-status-bar “Cmd” (command) will be displayed.

Sections immediately below MuPAD input usually display text in blue color. That is the standard color for *MuPAD output*. A section containing MuPAD output is called an *output region*. If the cursor is in such a region then at the right hand side of the MuPAD Pro-status-bar “Outp” (output) will be displayed.

Moreover there are *text regions*, i.e. sections containing *ordinary formatted text*. Here, the MuPAD Pro-status-bar will read “Text” at its right hand side.

For *executing* a MuPAD command (i.e. computing MuPAD input) a carriage return will do, if the cursor is placed anywhere within the corresponding input region. The respective MuPAD output will appear immediately below, e.g.:

• $1/5 + 2/3 - 5$

$$-\frac{62}{15}$$

Using <SHIFT>+<RETURN> it is possible to extend a MuPAD-command over several lines, starting the computation with a <RETURN> in the last line. Again an example:

• 2+

3+

4+

5+

6+

7

27

In contrast to preceding versions of MuPAD it is no longer necessary to complete MuPAD input with a semicolon at the end. A semicolon now serves to separate MuPAD input command that extends over several lines within an input region or within a MuPAD procedure.

A MuPAD input may also be concluded by a colon “:”. Then the computation will be executed, the output will, however, not be displayed on the screen. Try it out yourself by terminating, e.g., the command `sin(1.4123)` with a colon.

Likewise you may place several commands into one single MuPAD-input line:

```
• f := sin(x): F := int(f, x); diff(F, x)
      - cos(x)
      sin(x)
```

If screen-output of single command (that is not last) is desired, this command has to be terminated by semicolon.

1.2 Loading and Saving Notebooks

Similarly to ordinary word-processing editors notebooks can be saved and loaded again. Saving will be done by selecting “Save” and “Save As...” in the “File”-menu. Give a name to your notebooks and acknowledge saving. (MuPAD notebooks carry the file-extension “.mnb”).

You may load a MuPAD notebook by selecting “File Open...”. You may very well hold several notebooks simultaneously open. Each notebooks refers to its individual MuPAD session then, not messing up with others.

Hints

- ☞ Under “File/Export...” notebooks can be saved amongst other in the RTF-format (Rich-text-format) which can be processed by most word-processors under windows. Doing so, you may integrate a notebook into a Microsoft word-document after having executed all MuPAD computation. In the same way MuPAD notebooks can be exported to the HTML-format and, hence, can simply be published in the web.
- ☞ Vice versa is possible as well: Firstly prepare your notebook by editing in your favorite word-processor. Then save it in RTF-format and load it under MuPAD by selecting “File/Open:” In the “File/Open”-dialog select file type “rtf”.
- ☞ Under the menu entry “Help,” “Introduction” supplies for a variety of pre-edited notebooks facilitating beginners work with MuPAD.

1.3 Changing and Deleting within a Region


Like in ordinary word-processing editors contents of regions can be altered later on. Text can be inserted, deleted, copied and more. Additionally, “Drag & Drop” as well as “Copy & Paste” are at hand.

Insert new input and text regions by selecting “Input Above (<CTRL>+<F5>),” “Input Below (<F5>),” “Text Above (<CTRL>+<F6>),” “Text Below (<F6>)” in the menu “Insert.” The context menu of the mouse may prove to be especially useful for fast access to these inserting commands (click the right mouse button). The experienced user will additionally welcome keyboard shortcuts, e.g. the <F5>-key will create a new input region below.

To delete a region place the cursor anywhere within that region and select the region by pressing <F7> or choosing the entry “Select Region” in the “Notebook”-menu. Then press or choose “Cut” from the “Edit”-menu. Be careful to avoid premature deleting. There is no way to undo an unintentional delete command.

In similar fashion entire regions can be copied or moved. If you need specific help mastering MuPAD Pro, click onto the “Help” icon in the symbol-bar or refer to the MuPAD Pro Help which is found under “Help Topics (<ALT>+<F1>)” in the “Help”-menu. Alternatively, click onto the “Browse Help” icon in the MuPAD Pro symbol-bar.

Hints

-  When working through a MuPAD notebook, you might want to keep MuPAD from creating a new input line after each executed input command. That very feature, however, is most helpful when composing a new notebook. You may select, whatever feature you prefer, under the entry “Options..” in the “View”-menu.

Appreciate MuPAD's capabilities mastering computations arising in calculus.

New MuPAD functions

<code>sqrt</code>	square root
<code>float</code>	floating point representation for numerical calculus
<code>DIGITS</code>	number of significant decimal digits in a floating point number for numerical calculus
<code>delete</code>	deleting the value of an identifier
<code>expand</code>	expanding expressions
<code>normal</code>	canceling in rational expressions
<code>simplify</code>	simplifying expressions

- $2 + 1/3 - 10 \cdot 2^{20}$

$$\begin{array}{r} 31457273 \\ - \quad 3 \end{array}$$

- 10^{1000}

[illegible]

Apart from symbolic and exact calculus MuPAD features numerical computing as already introduced above. Certain numerical computations can be done with arbitrary precision. Here we compute the first 1000 decimal digits of π :

- `DIGITS := 1000: float(PI)`

```
3.14159265358979323846264338327950288419716939937510582097
4944592307816406286208998628034825342117067982148086513282
3066470938446095505822317253594081284811174502841027019385
2110555964462294895493038196442881097566593344612847564823
3786783165271201909145648566923460348610454326648213393607
2602491412737245870066063155881748815209209628292540917153
6436789259036001133053054882046652138414695194151160943305
7270365759591953092186117381932611793105118548074462379962
7495673518857527248912279381830119491298336733624406566430
8602139494639522473719070217986094370277053921717629317675
2384674818467669405132000568127145263560827785771342757789
6091736371787214684409012249534301465495853710507922796892
5892354201995611212902196086403441815981362977477130996051
8707211349999998372978049951059731732816096318595024459455
3469083026425223082533446850352619311881710100031378387528
8658753320838142061717766914730359825349042875546873115956
2863882353787593751957781857780532171226806613001927876611
195909216420199
```

The variable `DIGITS` indicates how many significant digits will go into our computation. As we do not wish to do all subsequent numerical calculations with such high precision, reset the variable by a `delete` command to its original value which was 10:

- `delete DIGITS:`

The above example teaches how to assign values to variables in MuPAD using the assignment operator `:=`:

- `f := sin(x)`

$\sin(x)$

After assignment `f` has the value `sin(x)`. The value of a variable is read by calling the variable:

- `f`

$\sin(x)$

The variable `f` may serve different purposes later. Then, either assign it a new value or delete it using the command `delete`:

- `delete f:`

A variable having no value at all returns itself when called for:

- `f`

`f`

A variable having no value may, e.g., represent an unknown parameter in an equation.

Note that the function `delete` applied to the variable `DIGITS` did not delete the value of `DIGITS`, rather it reset the variable to its original value. Variables like this are called environment variables. These govern certain global settings for MuPAD.

Hints

- ☞ Locate in MuPAD Pro the newly introduced “command menu bar,” visible upstairs. Use it to insert a specific command into your input region. Clicking on $\pi \approx 3.14$, say, you will find a `float(%?)` at your cursor position. Just fill in at the `%?` marked spots as desired. In functions with several arguments use the <tab>-key to navigate along the several `%?`.

Exercises

- ✎ Compute exactly $\sqrt{27} - 4\sqrt{3}$ and $\cos(\pi/8)$. Then, give a precise 5-digit approximation for these numbers. How do the treatments of $1/3 + 1/3 + 1/3$ and $1.0/3 + 1/3 + 1/3$ differ in MuPAD?
- ✎ The function `expand(expr)` expands the expression `expr`. Test the functionality using the following expressions: $(a + b)^2$, $(a + b)^3$ and $(a + b)^5$.
- ✎ `normal(expr)` cancels and furnishes a common denominator for a rational expression `expr`. Test the functionality using the following expressions: $x/(1 + x) - 2/(1 - x)$ and $(x^2 - 1)(1 + x)$.
- ✎ Apply the function `simplify` to the following expressions:

1. $(\sin(x))^2 + (\cos(x))^2$

2. $\sin(x + \pi/2)$

3. $\cos(x - \pi/2)$

2.2 Standard Calculus

New MuPAD functions

<code>solve</code>	solving equalities and inequalities
<code>diff</code>	differentiating
<code>limit</code>	limits
<code>%i</code>	referencing the output of <i>i</i> commands before
<code>subs</code>	substituting expressions (e. g. within equations)
<code>plotfunc2d</code>	2-dimensional rendering of function graphs

In the following consider standard calculus applied to the function

$$f(x) = \frac{x^2 - 5x + 6}{x - 1}.$$

First assign the function term to the variable `f`:

- `delete x:`
`f := (x^2 - 5*x + 6)/(x - 1)`

$$\frac{x^2 - 5x + 6}{x - 1}$$

Note that we deleted the value `x` (precautiously) in order to use it for an independent variable within the function definition of `f`.

Inspecting the function term, observe instantaneously that 1 is not contained in the domain of `f`. Thus, straightly proceeding, compute the intersection points of `f` with the coordinate axes. Firstly the zeroes:

- `solve(f = 0, x)`

$$\{2, 3\}$$

then secondly the y-coordinate of the intersection of `f` with the y-axis:

- `subs(f, x = 0)`

$$-6$$

Now we search for extremal points of `f` using the command `diff`. Firstly the necessary condition: $f'(x) = 0$:

- `f1 := diff(f, x)`

$$\frac{(2x - 5)}{(x - 1)} - \frac{(x^2 - 5x + 6)}{(x - 1)^2}$$

We assigned the variable `f1` to the derivative of `f` and compute its zero:

- `S := solve(f1 = 0, x)`

$$\{\sqrt{2} + 1, 1 - \sqrt{2}\}$$

Again, note that MuPAD computes exactly. Obtain an approximation of possible extremal values by applying the function `float` to the set `S`:

- `float(S)`

$$\{-0.4142135624, 2.414213562\}$$

Now, plug the zeroes of the first derivative into the second derivative to ensure the sufficient condition:

- `f2:= diff(f, x, x)`

$$\frac{2}{x-1} - \frac{2 \cdot (2 \cdot x - 5)}{(x-1)^2} + \frac{2 \cdot (-5 \cdot x + x^2 + 6)}{(x-1)^3}$$

Hints

☞ Note above that writing `x` twice yielded the second derivative. Likewise you will obtain the third, fourth, etc. derivative.

☞ The command `diff(f, x $ n)`, `n` being a positive integer, yields the `n`-th derivative of `f`. You might as well write `diff(f, x, x)` for the second and `diff(f, x, x, x, x, x, x)` for the sixth derivative, that is precisely the `$`-syntax:

- `x $ 6`

$$x, x, x, x, x, x$$

- `diff(x^7, x $ 6)`

$$5040 \cdot x$$

Plugging in possible extremal values into the second derivative can be accomplished using the function `subs` (substitute). We have already done so when computing the intersection of the function with the y-axis. Beginning with the first zero:

- `xe := subs(f2, x = S[1])`

$$-\sqrt{2} + \frac{\sqrt{2} \cdot (-5 \cdot \sqrt{2} + (\sqrt{2} + 1)^2 + 1)}{2} + 3$$

- `float(xe)`

1.414213562

That is a positive value, hence we have a local minimum at $\sqrt{2} + 1$. Above we utilized the command `S[i]` that refers to the *i*-th element of a set (or a list). Checking on the second zero of f' :

- `float(subs(f2, x = S[2]))`

-1.414213562

This turns out to be a local maximum for f .

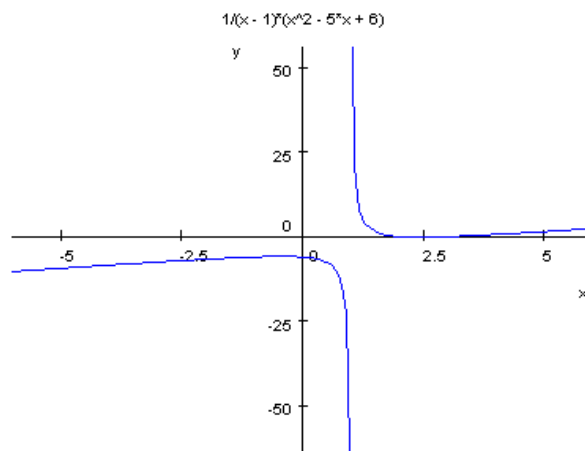
There are no inflection points for f as the second derivative never vanishes:

- `S := solve(f2 = 0, x)`

\emptyset

Finally, plot the graph of f over the domain $[-6, 6]$:

- `plotfunc2d(f, x = -6..6)`



This plot can be worked with interactively. A double click on the plot starts the MuPAD graphics tool VCam, the menu bar then displays the functionality of VCam. Clicking onto “Properties” in the “Edit”-menu you will access various properties of the graphical plot.

Hints

- ☞ It is possible to create graphical plots interactively. Place the cursor anywhere within your notebook. In the “Insert”-menu you will find the entries “2D Plot” and “3D Plot.” Choosing one creates a grayed out rectangle; double clicking on which starts VCam with its menu bar upstairs.

- ☞ The definitions used to interactively create a graphical plot can be retrieved as MuPAD commands. In VCam's "Edit"-menu you will find the entry "Command to MuPAD Notebook" which inserts the appropriate MuPAD commands creating the plot in an input line directly under the plot.
- ☞ A plot can be saved to a file in various graphical formats (cf. "Save Graphics" in VCam's "Edit"-menu). Also, using Copy & Paste a plot can be inserted into Microsoft Word, say. Double clicking onto a thus embedded graphical object, its properties can be edited by the application that produced the plot.
- ☞ The size of notebooks including graphical plots increases considerably. Inserting plots as symbols reduces size again. To do so select within the the dialog-window "View" the option "View as Symbol." That dialog-window is found in the sub-menu "Object Properties" of the "Edit"-menu.
- ☞ The above introduced commands `diff`, `solve`, `subs` and `plotfunc2d` have corresponding symbols $\frac{\partial}{\partial x}f(x)$, $\{x|f=0\}$, $f(x)|_{x=y}$ and a symbol for the graph of a function within the command bar of MuPAD Pro.

Exercises

- ✎ Plot the function $1/\sin(x)$ in the interval $[0, 1]$.
- ✎ Plot the function $\sin(x + y)$ on the domain $[0, \pi] \times [0, \pi]$.
- ✎ Apply standard calculus to the functions:
 - (a) $f(x) = x^2 \cdot e^{-x}$
 - (b) $f(x) = 1 - e^{-x}$
 - (c) $f(x) = \ln(x^2 - 1)$
- ✎ Find all zeroes of the function $f(x) = x^5 - 5x^4 + 5x^3 + 5x^2 - 6x$ and verify your result by plugging in into f .
- ✎ Find the following limits: $\lim_{x \rightarrow +\infty} (10 - x/(x^2 + 1)) \cdot (1.000.000/0.001x + 2)$ and $\lim_{x \rightarrow +\infty} x/(1/x + x^3 - 1)$.
Hint: Use the functions `limit` and `infinity`.
- ✎ Find left-hand and right-hand limits of $f(x) = (x - 3)/(x^2 - 5x + 6)$ for $x \rightarrow 2$. Plot the function on the domain $[-5, 5]$.
Hint: Use the function `limit` with options `Left` and `Right`.
- ✎ Prove with MuPAD using the definition of the derivative that the following holds: $f(x) = \sin(x) \Rightarrow f'(x) = \cos(x)$.

- ✎ Find for $f(x) = \ln(\ln(x))$ and $f(x) = u(x) \cdot v(x)$ the following derivatives: $f'(x), f''(x), f^{(20)}$.

Hint: Use the function `diff` and the operator `$`.

- ✎ Suppose that trade of a merchandise is governed by a bargain offering function $y_O = 100 - 20/x^2$ and a customer demand function $y_D = 90 - 15/x$. Selling the good at 95 \$ find out whether either backlogs of offer or demand prevail. If so compute the backlog.

Hint: Solve the equations $y_O(x) = 95$ and $y_D(x) = 95$ each for x and consider the difference of solutions.

2.3 Definite and Indefinite Integration

New MuPAD functions

<code>int</code>	definite and indefinite integration
<code>taylor</code>	Taylor-series of a function
<code>expr</code>	converting a series into an ordinary expression
<code>numeric::solve</code>	numerical solving of equations

Integrating is far more complex than differentiating. Still, MuPAD is of help as well:

- `delete x:`

```
int(x^4, x)
```

$$\frac{x^5}{5}$$

The command `int` triggers definite and indefinite integration. One more example:

- `F := int(sin(x)*cos(x)*x^2, x)`

$$\frac{\cos(2 \cdot x)}{8} + \frac{x \cdot \sin(2 \cdot x)}{4} - \frac{x^2 \cdot \cos(2 \cdot x)}{4}$$

Let us check:

- `f := diff(F, x)`

$$\frac{x^2 \cdot \sin(2 \cdot x)}{2}$$

The output looks different from the integrand we started with above. Yet, the two terms are equal. Verify with MuPAD:

- `s := sin(x)*cos(x)*x^2 - f`

$$x^2 \cdot \cos(x) \cdot \sin(x) - \frac{x^2 \cdot \sin(2 \cdot x)}{2}$$

- `simplify(s)`

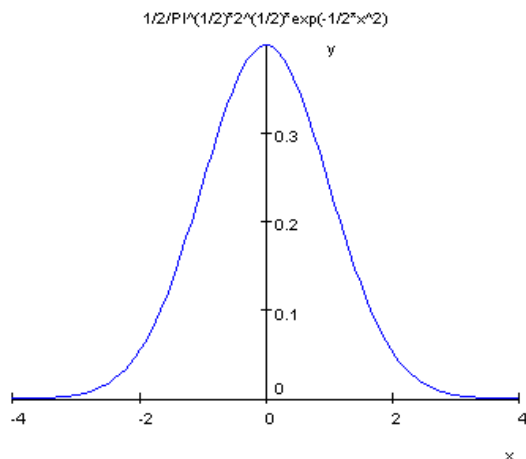
0

In the area of integration a computer algebra system proves to be particularly helpful. As a simple example consider Gauss' normal distribution that frequently pops up in stochastics. The values of the Gauss distribution come as definite integrals of the function $\varphi_{0,1}(x) = 1/\sqrt{2\pi} \cdot e^{-x^2/2}$. Usually, looking up these listed values is a tedious job. With the support of a computer algebra system, however, we can easily reproduce the values of the distribution with almost arbitrary precision. We compute in the following half the area under the centered standard bell-shaped Gauss curve. Commencing with a plot:

- `delete x:`
`gauss := 1/(sqrt(2*PI))*exp(-1/2*x^2)`

$$\frac{\sqrt{2} \cdot e^{-\frac{x^2}{2}}}{2 \cdot \sqrt{\pi}}$$

- `plotfunc2d(gauss, x = -4..4)`



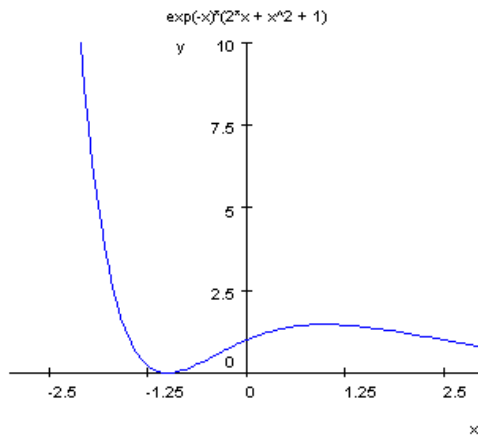
- `F := int(gauss, x):`
- `limit(F, x = infinity)`

$\frac{1}{2}$

As expected, the result is $1/2$.

In conclusion of this section let us consider the tangent problem treated in high school level calculus. The graph of the function $f(x) = (x^3 + 2x + 1) \cdot e^{-x}$ and its tangent contacting at $P(0|?)$ bound an area. First consider the graph of the function

- `f := exp(-x)*(x^2 + 2*x + 1):`
`plotfunc2d(f, x = -3..3, y = 0..10)`



In order to obtain the equation of the tangent compute the first two terms of the Taylor series of f :

- `t := taylor(f, x = 0, 2)`
 $1 + x + O(x^2)$

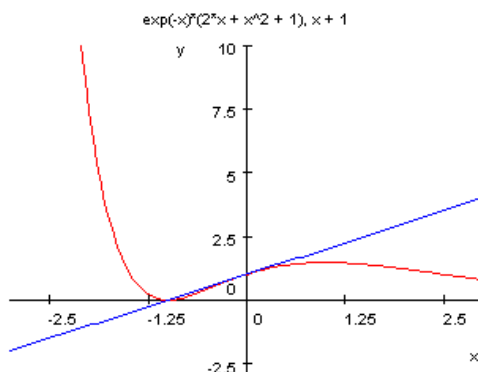
and convert it to an ordinary MuPAD expression using the command `expr` (i.e. eliminating the error term):

- `t := expr(t)`
 $x + 1$

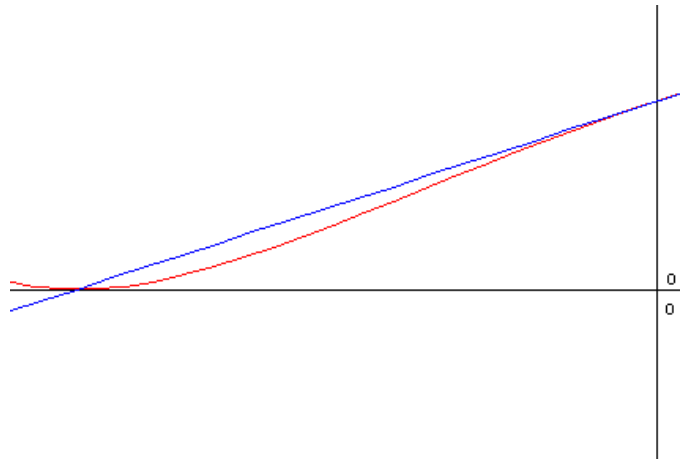
Now, we can process t and f alike.

Pass the functions f and t to the function `plotfunc2d` with reasonable domain and range (`x = -3..3`, `y = -3..10` are reasonable in our case). Then a graphical plot of f and the tangent t will be rendered.

- `plotfunc2d(f, t, x = -3..3, y = -3..10)`



At this stage we compute the area between the function f and the tangent t . As the plot suggest, the area is almost naught. We enlarge hence the specific part of the plot using VCam. By repeatedly double clicking on the lens symbol in VCam's symbol bar obtain:



Now it is easier to anticipate a successful procedure to compute the area in question. First compute the intersection point of the function graph and the tangent line. We utilize the function `numeric::solve` from the `numeric` library. The function `solve` is of no use here, as we are concerned with a transcendental equation. Thus, we have to solve numerically:

```
• numeric::solve(f = t, x)
  {-1.0}
```

The area, finally, is thus given by the following definite integral

```
• int(t - f, x = -1..0)
  -2/e + 11/2
```

This is in floating point approximation:

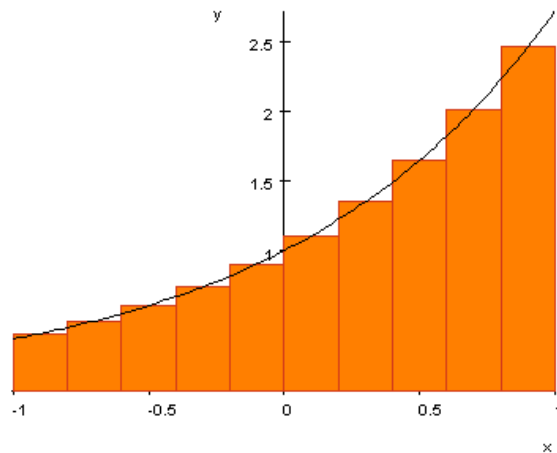
```
• float(%)
  0.06343634308
```

Hints

- ☞ The MuPAD library `student` features standard numerical integration procedures used in school teaching. Additionally, graphical visualization is provided for. For example the functions `student::riemann` and `student::plotRiemann` exhibit Riemann integration. Consider, say, a Riemann integration of the function

$f(x) = e^x$ over the interval $[-1, 1]$, partitioning the interval into 10 segments. The calling syntax is very similar to the syntax encountered when calling `plotfunc2d`: First pass the function, then the domain (i.e. the interval), and finally the number of desired subdivisions. The latter corresponds of course to the precision the area under the graph is computed with:

• `plot(student::plotRiemann(exp(x), x = -1..1, 10))`



Via `?student` you will get a survey featuring all integration techniques available in MuPAD's `student` library.

Exercises

- ✎ Find the definite integrals of the functions $f(x) = \sin(x)$ and $f(x) = \cos(x)$ over the interval $[0, 2]$.
- ✎ Compute the following definite integrals: $\int_0^{\pi/2} \sin(x) \cos(x) dx$, $\int_0^1 1/\sqrt{1-x^2} dx$.
- ✎ Find the following indefinite integrals:
 - (1) $\int e^{4x+1} dx$
 - (2) $\int x e^{1-x} dx$
 - (3) $\int (\ln(x))^2 dx$
 - (4) $\int \ln(x)/x^2 dx$
 - (5) $\int t^2 x \cdot e^{tx} dx$ and $\int t^2 x \cdot e^{tx} dt$
 - (6) $\int \ln(x) - (x^2 - 1)/(e^2 - 1) dx$
- ✎ Find the volume of the body given by rotating the function $f(x) = e^{2x+1}$ about the interval $[0, 4]$. Make use of the formula $V = \pi \int_0^4 (e^{2x+1})^2 dx$.
- ✎ Consider the family of functions $f_k(x) = (x^2 - kx) \cdot e^x$ for $k \in \mathbb{R}$. Find $k \in \mathbb{R}$ such that the area between the graph of f_k and the x-axis will be equal to 4.

- ✂ The function $f(x) = (x^2 + 3x + 2) \cdot e^{-x}$ and its tangent contacting at $P(0|?)$ bound an area.
- Plot the graph of the function $f(x)$.
 - Find the tangent contacting as above.
 - Plot both function and its tangent in one coordinate system.
 - Find the points where function and tangent intersect. Let your integration interval be delimited by these points.

2.4 An Extremal Problem

New MuPAD functions

assume attach a property to an identifier

Suppose we want to make cylindrical brass cans each containing one liter, minimizing the amount of brass required. Which are optimal height h and radius r defining the cylinders?

As the amount of brass required is related to the surface of cans, consider the well known formula giving the surface of a cylinder:

$$A(r, h) = 2\pi r^2 + 2\pi r h$$

Converting into MuPAD we take into account that r and h in question assume positive values only:

- delete** r, h :
`assume(r > 0): assume(h > 0):`
- A** := $2 \cdot \text{PI} \cdot r^2 + 2 \cdot \text{PI} \cdot r \cdot h$

$$2 \cdot h \cdot r \cdot \pi + 2 \cdot r^2 \cdot \pi$$

In advance, we deleted r and h to ensure that they do not attain any preassigned values.

Now, taking into account the constraining condition in our problem, namely that the volume be 1, consider the volume of the cylinder:

- V** := $r^2 \cdot \text{PI} \cdot h$

$$h \cdot r^2 \cdot \pi$$

Using the constraining condition we may eliminate one parameter, h say, from the equation above (we do our computations in the metric system where 1000 cubic centimeters make up for one liter):

- L** := `solve(1000 = V, h)`

$$\left\{ \frac{1000}{r^2 \cdot \pi} \right\}$$

and plug in the value of h into the function term of A :

- $A := \text{subs}(A, h = L[1])$

$$\frac{2000}{r} + 2 \cdot r^2 \cdot \pi$$

This is a function in the one parameter r , the extremal values of which solve our extremal problem. Get the first derivative:

- $A1 := \text{diff}(A, r)$

$$4 \cdot r \cdot \pi - \frac{2000}{r^2}$$

find its zeroes:

- $L := \text{solve}(A1 = 0, r)$

$$\left\{ \sqrt[3]{\frac{500}{\pi}} \right\}$$

Get the second derivative and check for the sufficient condition plugging in the zero:

- $A2 := \text{diff}(A, r, r)$

$$4 \cdot \pi + \frac{4000}{r^3}$$

- $\text{subs}(A2, r = L[1])$

$$12 \cdot \pi$$

The value is positive; we thus found the minimizing radius r . All we have to do now is to find the corresponding height h .

h is obtained by plugging in the minimizing r into the equation $V = 1000$ and solving for h

- $s := \text{subs}(V = 1000, r = L[1])$

$$h \cdot \pi \cdot \sqrt[3]{\frac{500}{\pi}}^2 = 1000$$

- $\text{solve}(s, h)$

$$\left\{ \frac{1000}{\pi \cdot \sqrt[3]{\frac{500}{\pi}}^2} \right\}$$

2.5 Working with MuPAD Libraries

Most of MuPAD's mathematical know-how is structured within a collection of so-called *libraries*. A library is again a collection of functions solving problems within a specific field, like linear algebra, calculus, algebra, numerics and many more.

For example the MuPAD library `numeric` provides for a vast variety of numerical routines. The `info` command lists all functions available in a specific library:

- `info(numeric)`

```
Library 'numeric': functions for numerical mathematics
```

```
-- Interface:
numeric::butcher,          numeric::complexRound,
numeric::cubicSpline,     numeric::det,
numeric::eigenvalues,     numeric::eigenvectors,
numeric::expMatrix,       numeric::fMatrix,
numeric::factorCholesky,  numeric::factorLU,
numeric::factorQR,        numeric::fft,
numeric::fsolve,          numeric::gldata,
numeric::gtdata,          numeric::indets,
numeric::int,              numeric::inverse,
numeric::invfft,          numeric::lagrange,
numeric::linsolve,        numeric::matlinsolve,
numeric::ncdata,          numeric::odesolve,
numeric::odesolve2,       numeric::odesolveGeometric,
numeric::polyroots,       numeric::polysysroots,
numeric::quadrature,       numeric::rationalize,
numeric::realroot,         numeric::realroots,
numeric::singularvalues,  numeric::singularvectors,
numeric::solve,            numeric::sort,
numeric::spectralradius
```

Pick for instance the function `numeric::fsolve` which finds zeroes of a function within a preassigned domain:

- `numeric::fsolve(sin(x), x = 2..4)`

```
[x = 3.141592654]
```

As you see, it is not necessary in advance to load a function of a library explicitly. When calling a function for a first time it is automatically loaded and executed (it is for this reason that a first call of a MuPAD function may consume a considerable amount of time; any subsequent call of that same function will be executed much faster).

When working interactively with libraries, you may want to “export” functions, i.e. make them *globally* known to the system. Then, they can be called directly without reference to the corresponding library. Consider, e.g., the function `numeric::det` for numerical computation of matrix determinants:

```
• export(numeric, det):
• det(matrix([[2, 1], [2, 2]]))
2.0
```

Library function, however, can be exported only when their name corresponds to a yet “free” identifier, i.e. an identifier without value.

Crucial functions like `diff`, `simplify` or `int` are not stored inside a library. These globally available functions are referred to as the “standard library,” which is sometimes abridged as `stdlib`, as in `info(stdlib)` or `?stdlib`.

2.6 Help, I Am Lost ...

New MuPAD functions

`?cmd` opens the help page corresponding to the command `cmd`





You know the name of a MuPAD command, but you do not know the exact meaning or the precise calling syntax? Then throw a glance at the corresponding help page. Suppose you want help concerning the function `limit`. Just execute the command `?limit` which starts MuPAD’s help browser. The command `limit` will appear selected and grayed out in the lower part of the window. Acknowledge the selection by pressing “Display” to start the MuPAD help system, that displays then the corresponding help page.

MuPAD help is based on hypertext. Key words come in colored and provide for links to corresponding pages in the manual; links can be double-clicked on. Within the hand-book you may jump to the directory, to the index, to the preceding or the following chapter.





After the detailed specification of a command you will find examples illustrating the use. Copy & paste any example to your notebook simply by double-clicking onto the colored symbol “>>” preceding the example.

What if you want to employ MuPAD to solve a specific problem, find a characteristic polynomial to a matrix, say, and you do not know of the specific functions? Again the help-browser is equally helpful. Either select “Find” (finding specific strings like “chara” for characteristic polynomial) or select “Content” and choose the entry “Linear Algebra” there, to be lead to the help page of MuPAD’s `linalg` library. There you will quickly find the desired command.

Exercises

-  Find out which MuPAD functions serve for solving of differential equations.
-  Find out which MuPAD function serves for solving of linear equations.
-  What is the name of the MuPAD library providing for elementary number theory?
-  Find the help page listing all MuPAD libraries.

Hints

-  Within “Contents” in the MuPAD help browser you will find an overview on each of the MuPAD libraries.
-  Context sensitive help answers questions concerning the help browser.
-  Windows help explains how to use the help browser.
-  The electronic and interactive version of MuPAD’s Tutorial is part of the MuPAD Pro package. In the help-menu you will find the entry “Open Tutorial.”

2.7 Differential Equations - An Application**New MuPAD functions**

- `ode` defines a differential equation
- `solve` solves a differential equation
- `S[i]` returns the i-th element of a set S

Let the temperature of a baking oven be determined by

$$T(t) = 200 - 180 e^{-0.3t}$$

measured in Centigrade. A pizza is well done after baking 20 minutes at approx. 200°C. The increase of temperature is related proportionally to the difference of oven and pizza temperature with a factor of 0.2. After what time will the pizza have reached 95% of its terminal temperature?

Converting the temperature function to MuPAD:

- `T := 200 - 180*exp(-0.3*t)`
 $-180 \cdot e^{-0.3 \cdot t} + 200$

Suppose that we turn on the heat the very moment ($t = 0$) the pizza is placed in the oven. Then the temperature $P(t)$ of the pizza at time t is given by

$$\frac{dP(t)}{dt} = 0.2 \cdot (T(t) - P(t)).$$

Assuming the pizza started originally with 20°C , convert this initial value problem to MuPAD using the command

• `o := ode({diff(p(t), t) = 0.2*(T-p(t)), p(0)=20}, p(t))`

$$\text{ode} \left(\left\{ p(0) = 20, 0.2 \cdot p(t) + \frac{\partial}{\partial t} p(t) + 36.0 \cdot e^{-0.3 \cdot t} - 40.0 \right\}, p(t) \right)$$

This differential equation is solved by MuPAD with the previously used command `solve`:

• `s := solve(o)`

$$\left\{ -340.0 \cdot e^{-0.2 \cdot t} + \frac{360.0 \cdot e^{-0.2 \cdot t}}{\sqrt[10]{e^t}} \right\}$$

The answer to our question above is hence given by

• `solve(s[1] = 200*0.95, t)`

We omitted the output as the symbolic solution is rather complex.

3 Linear Algebra

New MuPAD functions

`linalg` MuPAD's library for linear algebra
`matrix` creating vectors and matrices

Facilitating your approach to linear algebra, we show in this section how to create matrices and how to do mathematics with them. You will gain insight into MuPAD's library `linalg`, that comprises of many function in the field of linear algebra.

3.1 Creating Matrices

Pass the following matrix to MuPAD:

$$\begin{pmatrix} 1 & 4 & 0 & 3 & 3 \\ 2 & 0 & 0 & 2 & 0 \\ 7 & 0 & 3 & 2 & 1 \end{pmatrix}$$

- `matrix([[1, 4, 0, 3, 3],
 [2, 0, 0, 2, 0],
 [7, 0, 3, 2, 1]])`

$$\begin{pmatrix} 1 & 4 & 0 & 3 & 3 \\ 2 & 0 & 0 & 2 & 0 \\ 7 & 0 & 3 & 2 & 1 \end{pmatrix}$$

A matrix is created with the MuPAD command `matrix`. Every row of the matrix is enclosed in brackets `[...]`. Around all rows, there is a another pair of brackets and the whole thing is passed to the command, enclosed in parentheses `(...)`.

Exercises

- ✎ Try out input, even input that may not be syntactically accepted by MuPAD. What is happening if you pass rows of different length to a `matrix` command?

We illustrate specific syntaxes:

- `A := matrix([[8, 3, 5, 6]])`

$$\begin{pmatrix} 8 & 3 & 5 & 6 \end{pmatrix}$$

Compare this matrix (a 1×4 -matrix which is called a row vector with four entries) to a 4×1 matrix (a column vector):

- `A := matrix([8, 3, 5, 6])`

$$\begin{pmatrix} 8 \\ 3 \\ 5 \\ 6 \end{pmatrix}$$

You see that MuPAD is making life easy for us. In fact, we should have entered the more complicated (which we might very well have done)

- `A := matrix([[8], [3], [5], [6]])`





$$\begin{pmatrix} 8 \\ 3 \\ 5 \\ 6 \end{pmatrix}$$

Entering matrices is very simple using the MuPAD Pro command bar. Click onto the respective matrix symbol and select appropriate row and column numbers. Creating thus a 4×4 matrix you will find the following lines in the current input region:

- `matrix([[%?,%?,%?,%?], [%?,%?,%?,%?], [%?,%?,%?,%?], [%?,%?,%?,%?]])`

Use <Tab> and <Shift-Tab> to navigate forward and backward along the different %? entries.

Exercises

-  Create a column vector with entries 1, 2, 3.
-  Try to enter the three unit vectors of \mathbb{R}^3 to MuPAD
-  Give an example for a diagonal matrix.
-  Create the 4×5 -zero matrix

3.2 Row and Column Number of a Matrix

New MuPAD functions

`linalg::ncols` number of columns for a matrix
`linalg::nrows` number of rows for a matrix

MuPAD features `linalg::ncols` (number of columns) and `linalg::nrows` (number of rows) to access number of columns and number of rows of a matrix:

- `M := matrix([[1, 7], [3, 1], [1.5, 0]])`

$$\begin{pmatrix} 1 & 7 \\ 3 & 1 \\ 1.5 & 0 \end{pmatrix}$$

- `linalg::ncols(M)`

2

- `linalg::nrows(M)`

3

3.3 Accessing Matrix Entries

New MuPAD functions

`A[i, j]` coefficient of a matrix A located in i^{th} row and j^{th} column
`linalg::col` extract columns of a matrix
`linalg::row` extract rows of a matrix

Accessing matrix coefficients is done using the operator `[]`. Consider a matrix:

- `M := matrix([[1, 4, 0, -1], [3, 4, 7, 8], [-5, -6, 7, 1]])`

$$\begin{pmatrix} 1 & 4 & 0 & -1 \\ 3 & 4 & 7 & 8 \\ -5 & -6 & 7 & 1 \end{pmatrix}$$

The coefficient located in the second row and the first column is given by:

- `M[2, 1]`

3

More coefficients:

- `M[1, 3]`

0

- `M[2, 3]`

7

Similarly matrix coefficient can be altered without building an entirely new matrix:

- `M[1, 3] := 2;`

- `M`

$$\begin{pmatrix} 1 & 4 & 2 & -1 \\ 3 & 4 & 7 & 8 \\ -5 & -6 & 7 & 1 \end{pmatrix}$$

You see that we replaced the entry in the first row and third column with the value 2.

The bracket operator does even more than extracting single entries. You may extract a whole sub matrix:

- `M1 := M[1..3, 2..3]`

$$\begin{pmatrix} 4 & 2 \\ 4 & 7 \\ -6 & 7 \end{pmatrix}$$

- `row1 := M[1..1, 1..4]`

$$(1 \ 4 \ 2 \ -1)$$

Almost the same functionality is provided for by the functions `linalg::cols` and `linalg::rows` extracting rows and columns:


- `row1 := linalg::row(M, 1)`


$$(1 \ 4 \ 2 \ -1)$$

- `columns132 := linalg::col(M, [1, 3, 2])`

$$\left[\begin{pmatrix} 1 \\ 3 \\ -5 \end{pmatrix}, \begin{pmatrix} 2 \\ 7 \\ 7 \end{pmatrix}, \begin{pmatrix} 4 \\ 4 \\ -6 \end{pmatrix} \right]$$

Exercises

-  Call for a coefficient with row number (column number) greater than the row number (column number) of the matrix. What happens?

-  Create the matrix $X = \begin{pmatrix} 1 & -2 & 1 & 5 \\ 3 & 0 & 1 & 1 \\ 2 & 2 & 2 & 2 \\ 0 & 2 & 1 & 12 \end{pmatrix}$ using MuPAD. Then define suitable

2×2 -sub matrices A, B, C and D with $X = \begin{pmatrix} A & B \\ B & C \end{pmatrix}$.

3.4 Elementary Matrix Calculus

New MuPAD functions

<code>linalg::transpose</code>	transposing a matrix
<code>+</code>	adding matrices
<code>*</code>	multiplying matrices
<code>$\hat{-1}$</code>	inverting matrices

In order to do some matrix calculus create two matrices with symbolic entries:

- `A := matrix([[a11, a12], [a21, a22]])`

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

- `B := matrix([[b11, b12], [b21, b22]])`

$$\begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix}$$

These two can be easily added and multiplied with one another:

- $A + B$

$$\begin{pmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{pmatrix}$$

- $A * B$

$$\begin{pmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} & a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} & a_{21} \cdot b_{12} + a_{22} \cdot b_{22} \end{pmatrix}$$

Of course, MuPAD provides for comfortable matrix inversion. Inverting a matrix by hand is a rather tedious task, like solving a system of linear equations; mathematically, though, it is not a big challenge.

Let us invert the following matrix:

- ```
A := matrix([
 [2, 0, 1, 3, 5], [4, 1, -1, 0, 4], [1, -1, 1, 0, 1],
 [0, 2, 1, -6, 7], [-1, -2, 0, 3, 4]
])
```

$$\begin{pmatrix} 2 & 0 & 1 & 3 & 5 \\ 4 & 1 & -1 & 0 & 4 \\ 1 & -1 & 1 & 0 & 1 \\ 0 & 2 & 1 & -6 & 7 \\ -1 & -2 & 0 & 3 & 4 \end{pmatrix}$$

All we have to do is enter  $1/A$  or  $A^{-1}$ :

- ```
iA := A^(-1)
```

$$\begin{pmatrix} -\frac{1}{49} & \frac{6}{35} & \frac{62}{245} & -\frac{3}{49} & -\frac{5}{49} \\ \frac{17}{49} & -\frac{4}{35} & -\frac{123}{245} & \frac{2}{49} & -\frac{13}{49} \\ \frac{49}{17} & -\frac{35}{11} & \frac{245}{73} & \frac{49}{2} & -\frac{49}{13} \\ \frac{49}{29} & -\frac{35}{2} & \frac{245}{44} & \frac{49}{11} & -\frac{49}{2} \\ \frac{147}{1} & -\frac{35}{35} & -\frac{245}{13} & -\frac{147}{3} & -\frac{147}{5} \end{pmatrix}$$

Checking our result we have indeed:

- $iA * A; A * iA$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

If a matrix has no inverse, MuPAD returns the result **FAIL**.

Exercises

 Given matrices A, B, C and D by

$$A = \begin{pmatrix} 2 & 0 \\ 1 & -1 \end{pmatrix}, B = \begin{pmatrix} 0 & -1 \\ 3 & 3 \end{pmatrix},$$

$$C = \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}, D = \begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix},$$

convert these matrices to MuPAD and compute:

$$A \cdot (B + C) + A \cdot C \cdot D,$$

$$(-A - (B - (A + C - (B + C) - D))),$$

$$A^T - 2 \cdot A \cdot B \cdot D \cdot A \cdot D \cdot B \cdot D.$$

Hint: The superscript T denotes the transposed matrix. It can be computed via `linalg::transpose(A)`.

3.5 More Tricks and Tips Creating Matrices

New MuPAD functions

`Diagonal` an option for `matrix`: Creating a diagonal matrix
`->` definition of a mapping function

Diagonal matrices can be created immediately using the the option `Diagonal`:

- `matrix(4, 4, 1, Diagonal)`

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

- `matrix(4, 4, x -> x^2, Diagonal)`

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 16 \end{pmatrix}$$

- `matrix(3, 4, [1, 2, 1], Diagonal)`

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Instead of passing a list of lines, a matrix can be created using a function mapping positions to entries accordingly:

- `matrix(3, 4, (i,j) -> (i^2 + j^2))`

$$\begin{pmatrix} 2 & 5 & 10 & 17 \\ 5 & 8 & 13 & 20 \\ 10 & 13 & 18 & 25 \end{pmatrix}$$

3.6 Coefficient Domains for Matrices

New MuPAD functions

`Dom::Matrix` creator for matrices over a coefficient domain
`Dom::Integer` domain of positive and negative integers and zero
`Dom::Rational` domain of all rational numbers

Thus far we have not spent a thought on the type of entries in a matrix. MuPAD features matrices over specific coefficient domains. Let us work over the domain of rationals, say:

- `MQ := Dom::Matrix(Dom::Rational):`
- `A := MQ([[1, -3, 3], [3, -5, 3], [6, -6, 4]])`

$$\begin{pmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{pmatrix}$$

Computing the inverse:

- `A^(-1)`

$$\begin{pmatrix} -\frac{1}{8} & -\frac{3}{8} & \frac{3}{8} \\ \frac{3}{8} & -\frac{1}{8} & \frac{3}{8} \\ \frac{3}{4} & -\frac{3}{4} & \frac{1}{4} \end{pmatrix}$$

Observe that the entries are now made up from rational numbers, whereas the matrix A consisted of integers only.

We now define that same matrix A explicitly over the integers:

- `MZ := Dom::Matrix(Dom::Integer):`
- `A2 := MZ(A)`

$$\begin{pmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{pmatrix}$$

Computing the inverse of that very matrix MuPAD now returns:

- `A2^(-1)`

FAIL

Hence, as expected, this matrix has no inverse over the predefined coefficient range.

When using the default coefficient ring, it is possible to admit any coefficient entries, including parameters (identifiers). Unimpressed MuPAD computes:

- `A := matrix([[2, 3], [a, 4]])`

$$\begin{pmatrix} 2 & 3 \\ a & 4 \end{pmatrix}$$

- `A^2`

$$\begin{pmatrix} 3 \cdot a + 4 & 18 \\ 6 \cdot a & 3 \cdot a + 16 \end{pmatrix}$$

Hints



MuPAD provides with the `Dom` libraries for a great number of coefficient domains for matrices, e.g. `Dom::Integer`, `Dom::Rational`, `Dom::Real`, `Dom::Float` and `Dom::Complex`.

4 The `linalg` Library

New MuPAD functions

<code>linalg::matlinsolve</code>	solving systems of linear equations
<code>linalg::eigenvectors</code>	eigenvectors of a matrix
<code>linalg::charpoly</code>	characteristic polynomial of a matrix
<code>linalg::jordanForm</code>	the Jordan canonical form of a matrix
<code>linalg::gaussElim</code>	the Gauss elimination process
<code>linalg::addRow</code>	adding one row to another
<code>linalg::addCol</code>	adding one column to another
<code>linalg::multRow</code>	multiplying a row with a scalar
<code>linalg::multCol</code>	multiplying a column with a scalar
<code>linalg::expr2Matrix</code>	construct a matrix from a system of linear equations

The MuPAD library `linalg` provides for a variety of important functions in the field of linear algebra. The function `info` yields a survey comprising of all functionality in the package:

- `info(linalg)`

```
Library 'linalg': the linear algebra package
```

```
-- Interface:
linalg::addCol,      linalg::addRow,
linalg::adjoint,     linalg::angle,
linalg::basis,       linalg::charmat,
linalg::charpoly,    linalg::col,
linalg::companion,   linalg::concatMatrix,
linalg::crossProduct, ...
-- Exported:
conjugate, exp, norm, normal
```

Because of the size of output we confined ourselves to list here only part of all library functions.

Now let us explore some functions in greater detail. Let us start by creating a matrix:

- `A := matrix([[1, -3, 3], [3, -5, 3], [6, -6, 4]])`

$$\begin{pmatrix} 1 & -3 & 3 \\ 3 & -5 & 3 \\ 6 & -6 & 4 \end{pmatrix}$$

The system of *eigenvalues and eigenvectors* is computed by the following call:

- `linalg::eigenvectors(A)`

$$\left[\left[-2, 2, \left[\begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \right] \right], \left[4, 1, \left[\begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 1 \end{pmatrix} \right] \right] \right]$$

In this case the result is a list of two sub-lists. Each sub-list contains first an eigenvalue (-2 and 4 in our case), then its algebraic multiplicity (2 and 1 in our case, i.e. 2 is a double zero to the characteristic polynomial of A) and, finally, a list of eigenvectors to the respective eigenvalue. The sequence of eigenvectors is an eigensystem to the respective eigenvalue.

The characteristic polynomial of a matrix is found as follows:

- `linalg::charpoly(A, x)`

$$x^3 - 12 \cdot x - 16$$

Here we pass as a second parameter the unknown of the characteristic polynomial (in our case x) to the function.




Diagonalizing a matrix is simply done by the command:

- `linalg::jordanForm(A)`

$$\begin{pmatrix} -2 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & -2 \end{pmatrix}$$

In addition the package provides for functions for elementary line and column operations such as `linalg::addRow` and `linalg::addCol`, `linalg::multCol` and `linalg::multRow`, and many more, thus allowing single step by step operations of the Gaussian elimination process, say.

Exercises

-  Find out about the functionality of `linalg::gaussElim`. Which information is obtained when passing along the option `All`?
-  How do you compute the determinant of a square matrix in MuPAD? Give some examples.
-  Find the general solution to the following system of linear equations: $x + y + z = 10$, $2x - y + z = 8$, $3x + 2z = 18$. Make use of the functions `linalg::expr2Matrix` and `linalg::matlinsolve`. Which function in the standard library would have solved this problem immediately?

4.1 Linear Dependence and Independence of Vectors

Here we shall study in detail linear dependence of vectors. Profiting again from learn-

ing by doing in MuPAD, consider an example involving three vectors $\begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix},$

$\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. Setting $a \cdot \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} + b \cdot \begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix} + c \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = 0$ leads to a homogeneous

system of linear equations: $\begin{pmatrix} 1 & 1 & 0 \\ 2 & 4 & 0 \\ -1 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$. If $a = b = c = 0$ is the

only solution to this system, then the three vectors are linearly independent. Otherwise, they are linearly dependent. Let us create the three vectors in MuPAD:

- `v1 := matrix([1, 2, -1]):`
- `v2 := matrix([1, 4, 2]):`
- `v3 := matrix([0, 0, 1]):`

The proper coefficient matrix to the system of linear equations is then given by:

- `A := v1 . v2 . v3`

$$\begin{pmatrix} 1 & 1 & 0 \\ 2 & 4 & 0 \\ -1 & 2 & 1 \end{pmatrix}$$

The dot-operator concatenates in this context vectors to build a matrix.

In MuPAD we solve this system of equations using the command:

- `linalg::matlinsolve(A, matrix([0, 0, 0]))`

$$\begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

We passed the coefficient matrix and the right-hand side vector as parameters. The result is $a = b = c = 0$, i.e. the vectors in question are linearly independent.

It is well known that a set of three linearly independent vectors forms a base of a three-dimensional vector space. It is moreover well known that every vector $\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ of the vector space has unique coordinates with respect to this base, i.e. there are uniquely

determined numbers x_1, x_2, x_3 such that $x_1 \cdot \begin{pmatrix} 1 \\ 2 \\ -1 \end{pmatrix} + x_2 \cdot \begin{pmatrix} 1 \\ 4 \\ 2 \end{pmatrix} + x_3 \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ holds. We shall now try to find the coordinates for the vector $\begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$ with respect to our base. That is actually fairly simple, all we have to do is to solve the inhomogeneous system of linear equations $\begin{pmatrix} 1 & 1 & 0 \\ 2 & 4 & 0 \\ -1 & 2 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}$. We will do this in analogy to the approach above:

- `linalg::matlinsolve(A, matrix([1, 2, 3]))`

$$\begin{pmatrix} 1 \\ 0 \\ 4 \end{pmatrix}$$

For an arbitrary vector $\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$ we obtain coordinates as follows:

- `L := linalg::matlinsolve(A, matrix([a1, a2, a3]))`

$$\begin{pmatrix} 2 \cdot a_1 - \frac{a_2}{2} \\ \frac{a_2}{2} - a_1 \\ 4 \cdot a_1 - \frac{3 \cdot a_2}{2} + a_3 \end{pmatrix}$$

That is the general solution to the above system of linear equations. Indeed, immediately verify:

- `L[1] * v1 + L[2] * v2 + L[3] * v3`

$$\begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}$$

Exercises

- ✎ Find out whether the vectors $\begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \begin{pmatrix} 3 \\ 0 \\ -1 \end{pmatrix}, \begin{pmatrix} -1 \\ 2 \\ 0 \end{pmatrix}$ are linearly dependent or independent.
- ✎ Prove using MuPAD that two vectors in \mathbb{R}^3 are linearly dependent if and only if they are scalar multiples of one another.

4.2 An Application from Analytic Geometry

New MuPAD functions

`linalg::angle` the angle between two vectors

Consider a plane

$$E: \vec{x} = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} + l \cdot \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} + m \cdot \begin{pmatrix} -3 \\ 1 \\ 4 \end{pmatrix}$$

and a straight line

$$g: \vec{x} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} + k \cdot \begin{pmatrix} 4 \\ -1 \\ 2 \end{pmatrix}.$$

Find the intersection point of plane and line. Equating both plane and line equation provides the system of linear equations

$$\begin{pmatrix} 1 & -3 & -4 \\ -1 & 1 & 1 \\ -1 & 4 & -2 \end{pmatrix} \cdot \begin{pmatrix} l \\ m \\ k \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix},$$

which yields the coefficient matrix

• `A := matrix([[1, -3, -4], [-1, 1, 1], [-1, 4, -2]])`

$$\begin{pmatrix} 1 & -3 & -4 \\ -1 & 1 & 1 \\ -1 & 4 & -2 \end{pmatrix}$$

The right hand side of the system is then given by the vector:

• `b := matrix([1, -1, 2])`

$$\begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}$$

Solve this system using the command introduced in the previous section:

• `L := linalg::matlinsolve(A, b)`

$$\begin{pmatrix} \frac{5}{11} \\ \frac{1}{11} \\ -\frac{2}{11} \end{pmatrix}$$

The third component of the solution vector is the value for k in the line equation above yielding the intersection point:

• `matrix([3, 0, 1]) + L[3] * matrix([4, -1, 2])`

$$\begin{pmatrix} 7 \\ 11 \\ 5 \end{pmatrix}$$

Exercises

- ✎ Adding and multiplication of matrices follows the distributive law. i.e. $A \cdot (B + C) = A \cdot B + A \cdot C$. Verify this law for all 2×2 matrices.

Hint: Make use of the function `expand`.

- ✎ For which values of x does the matrix

$$\begin{pmatrix} x-1 & x-1 & x+2 \\ x-1 & 2 \cdot x-3 & x+5 \\ 2 \cdot (x-1) & 3 \cdot x-4 & 3 \cdot x+4 \end{pmatrix}$$

have no inverse?

- ✎ Let the matrices A and B be given by

$$A := \begin{pmatrix} 0 & -2 & x \\ 2 & 3 & -2 \\ 0 & 4 & -4 \end{pmatrix} \quad B := \begin{pmatrix} 1 & 3 & 0 \\ 0 & -2 & 3 \\ x & -3 & 5 \end{pmatrix}.$$

For which $x \in \mathbb{R}$ is the sum of A and B not invertible?

- ✎ Find the general solution of the system of linear equations $Ax = b$ where

$$A := \begin{pmatrix} 1 & 2 & 0 & 4 \\ 4 & 8 & 1 & 18 \\ -1 & -2 & -1 & -6 \\ 2 & 4 & 1 & 10 \end{pmatrix} \quad b := \begin{pmatrix} 1 \\ 17 \\ -8 \\ 11 \end{pmatrix}.$$

- ✎ Consider the points A, B, C and D in \mathbb{R}^3 given by $A(1, 2, 3)$, $B(-1, 0, 2)$, $C(-1, 2, 3)$ and $D(0, 4, 4)$. Find the equation of the line $G1$ passing through A and B and the equation of the line $G2$ passing through C and D . At what angle do $G1$ and $G2$ intersect?

Hint: Use the function `linalg::angle`.

- ✎ Prove using MuPAD that $\det(A \cdot B) = \det(A) \cdot \det(B)$ holds for arbitrary 3×3 matrices A and B .

Hint: Make use of the functions `linalg::det` and `expand`.

5 Statistics and Probability

This chapter illustrates the use of MuPAD in stochastics:

5.1 Statistical Computations

New MuPAD functions

`fact` faculty
`binomial` determining a binomial coefficient

We begin with simple problems occurring in statistics:

1. Throw a die 30 times taking into account the order of throws. Then we have

- `6^30`

221073919720733357899776

possible different outcomes.

2. Seating 25 people on 25 seats. What is the number of all possible seating arrangements? The answer is

- `fact(25)`

15511210043330985984000000

For 100 people on 100 seats we have:

- `fact(100)`

933262154439441526816992388562667004907159682643816
 214685929638952175999932299156089414639761565182862
 53697920827223758251185210916864000000000000000000
 00000

Well, this number is really big, it can hardly be overseen. To get an idea of its size, we do the computation as follows:

- `float(fact(100))`

$$9.332621544 \cdot 10^{157}$$

Suppose we would give the 100 people 15 minutes time to change from one seating arrangement to another, it would then take them:

- `float(fact(100)/35040)`

$$2.663419391 \cdot 10^{153}$$

years to try out all possible seating arrangements.

3. Playing Lotto, the number of possible choices “6 out of 49” is given by

- `binomial(49, 6)`

$$13983816$$

The probability picking 6 right numbers when choosing “6 out of 49” is thus approximately:

- `float(1/binomial(49, 6))`

$$0.00000007151123842$$

Exercises

- ✎ Make yourself familiar with the functions `fact` and `binomial`. Do a couple of experiments with real big numbers - you will experience (hardly) any restrictions.
- ✎ Throwing a die 20 times how many different outcomes are possible?
- ✎ Playing cards you are dealt 8 cards out of 32. How many different hands are possible?
- ✎ After a talk, a picture of 5 mathematicians is to be taken. There is an argument concerning the formation. The mathematicians solve it by agreeing to take a picture of each possible formation separately. The photographer has a 32-picture film. Can he succeed?
- ✎ After having successfully solved their argument the 5 mathematicians of the preceding problem meet again to take a picture. As the 5 are of two different nationalities, they want this time to consider formations regarding nationality only. How many different pictures will now be taken? Is this problem uniquely posed and solvable at all?

5.2 Simulating Simple Laplace Experiments

New MuPAD functions

`random` generating random numbers
`print` function for data output

Simple Laplace experiments can be simulated using MuPAD's random number generator `random`. Let us firstly simulate tossing an ideal coin. The number 1 will stand for face up ("heads") the number 2 for face down ("tails"). Tossing the coin only once, we define a variable called `CoinToss` for our random experiment:

- `CoinToss := random(1..2):`

Now simulate one toss:

- `CoinToss()`

2

Tossing the coin 10 times we place the function call in a loop and print the outcome of each toss to the screen:

- `for i from 1 to 10 do print(CoinToss()) end_for`

1

2

2

1

1

2

1

2

2

2

Of course, we may similarly simulate tossing an ideal die: Slightly change the above lines to work with a random number generator generating random numbers from 1 to 6:

- `DiceToss := random(1..6):`
- `for i from 1 to 10 do print(DiceToss()) end_for`

4

4

3

3

2

1




4

4

6

1

Exercises

-  Suppose that 30 balls marked with the numbers 1 through 30 lie in an urn. Apart from their numbers the balls are all alike and indistinguishable. Simulate 10 drawings of a single ball out of the urn when the ball drawn is put back into the urn before the next draw. The numbers of the balls drawn are recorded.
-  Do the above experiment again. Since chance is involved, you should obtain a different sequence of numbers now.
-  Preceding class, one student is chosen to give a brief survey of what she or he learned in the previous class. The teacher chooses the student at random by picking a card out of a box containing cards each naming a student. What are the chances for a student being called three times in a row?

5.3 Relative Frequency and the Law of Numbers

Building upon the results of the preceding section we study relative frequencies here. Again, consider an ideal die and count the relative frequencies of rolling a 6. Write a procedure that counts in a variable `rf` the number of rolled 6s and divides this number by `n` the total number of rolls. We pass the total number `n` of rolls to the function:

```
• Roll := random(1..6):  
• RelativeFrequency := proc(n)  
  local rf;  
  begin  
    rf := 0;  
    for i from 1 to n do  
      if Roll() = 6 then rf := rf + 1 end_if;  
    end_for;  
    return(rf/n)  
  end;
```

With the call:

```
• RelativeFrequency(10)
```

$\frac{1}{10}$

we get the relative frequency of 6s in 10 rolls of a die. It is well known that the probability for a 6 is $\frac{1}{6}$ in each roll.

The Law of Numbers states that increasing the number of rolls (i.e. increasing **n**) the relative frequency will near the probability. We check this by doing experiments with 10, 100, 1,000, etc. and eventually 100,000 rolls and computing the corresponding relative frequencies:

```
• float(RelativeFrequency(10))
```

0.2

```
• float(RelativeFrequency(100))
```

0.17

```
• float(RelativeFrequency(1000))
```

0.157

```
• float(RelativeFrequency(10000))
```

0.1688

```
• float(RelativeFrequency(100000))
```

0.16768

Exercises

- ✎ Modify the procedure above to obtain relative frequencies for 1s and 6s.
Hint: Find out about the `if`-command in the above procedure.
- ✎ Write a new procedure that computes the relative frequency of 0 in Roulette. Again, simulate `n` experiments.
Hint: You may utilize most of the procedure above, think about the `random` function in the procedure.

5.4 Computing the Binomial Distribution

New MuPAD functions

<code>sum</code>	computing sums
<code>plot::bars</code>	plotting data with MuPAD
<code>plot</code>	graphical plotting on the screen

Now let us do `n` independent experiments with two possible outcomes: “win” and “loose.” We assume that the win-probability for each experiment is given by p ($0 < p < 1$). Then the loose-probability is given by $1 - p$ (the probability of the complementary event). We compute the probability for k wins in n independent experiments. The well known formula reads $B(n, p, k) = \binom{n}{k} \cdot p^k \cdot (1 - p)^{n-k}$. We implement it in MuPAD:

```
• B := (n, p, k) -> binomial(n,k) * p^k * (1-p)^(n-k):
```

This functions describes any random experiment with binomial distribution.

Consider the following example: Toss an ideal die 50 times. Find the probability of the following events:

- (1) Rolling exactly ten 6s,
- (2) roll at least two and at most eighteen 6s.

The probability p going into the binomial formula is $1/6$ and $n = 50$ is the number of experiments. The probability of the event described in (1) is thus:

```
• B(50, 1/6, 10)
```

$$\frac{46712912853763555176556110382080078125}{404140638732382030321569800228268146688}$$

As done before, an approximation is given by:

```
• float(%)
```

0.1155857847

The probability of the event described in (2) is a sum of single probabilities :

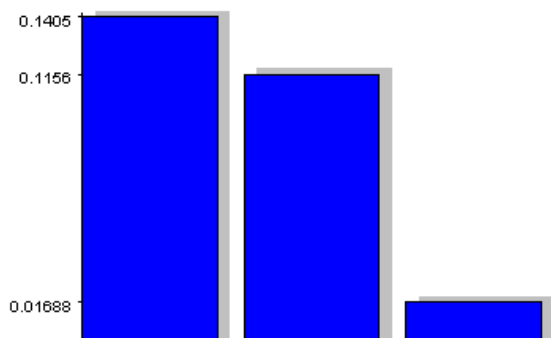
- `float(sum(B(50, 1/6, k), k = 2..18))`
0.9985409078

Of course, our binomial distribution can be plotted graphically. Just use the command `plot::bars` to obtain a bar diagram:

- `diagramm := plot::bars([B(50, 1/6, 7), B(50, 1/6, 10), B(50, 1/6, 14)])`
`plot::Scene()`

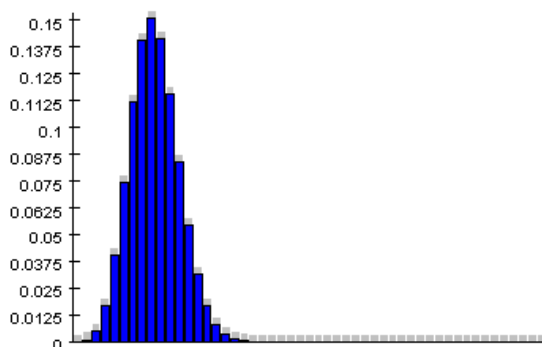
And plot this diagram to the screen using the command `plot`:

- `plot(diagramm)`



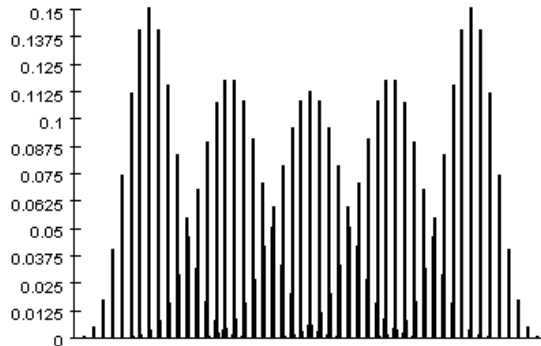
To display the complete binomial distribution utilize the sequential operator `$`. Then a short input will do:

- `plot(plot::bars([B(50, 1/6, i) $ i = 0..50]))`



Now display binomial distributions for different probabilities in one plot:

• `plot(plot::bars([[B(50, j/6, i) $ i = 0..50] $ j=1..5]))`



Exercises

- ✎ Tossing an ideal coin 100 times compute the probabilities of the following events:
 - (1) Exactly 50 heads,
 - (2) at least 40 and at most 60 heads,
 - (3) not a single head.
 Make use of the function B above.
- ✎ Find the probability rolling at least 400 and no more than 500 even sides when rolling an ideal die 1,500 times .
- ✎ Turning the wheel of Fortune 4 times, each time the numbers from 0 to 9 are possible. Consider the following events:
 - (a) Exactly two even numbers,
 - (b) at least three numbers are less than 5,
 - (c) Two or three numbers are greater than 2.

6 Plotting Made Easy with MuPAD

Already in the chapters dealing with calculus and stochastics we learned some skills for graphically visualizing mathematical issues. In this chapter let us delve into plotting a little deeper.

New MuPAD functions

<code>plot::Function2d</code>	plotting 2-dimensional function graphs
<code>plot::Function3d</code>	plotting 3-dimensional function graphs
<code>plot::Ellipse2d</code>	plotting ellipses and circles
<code>plot::Curve2d</code>	plotting 2-dimensional curves
<code>plot::Curve3d</code>	plotting 3-dimensional curves
<code>plot::Polygon</code>	plotting polygons
<code>plot::Pointlist</code>	visualizing sequences of points
<code>plot::Scene</code>	joining different graphical objects within a scene
<code>plot::Rectangle2d</code>	plotting of rectangles
<code>plot::xrotate</code>	plotting solids of revolution
<code>Axes=Corner</code>	plotting option placing coordinate axes to the left and bottom side
<code>Axes=Box</code>	plotting option placing the coordinate system in a box
<code>GridLines=Automatic</code>	plotting option that covers the entire coordinate system with a rectangular grid

6.1 Function Graphs

In the preceding chapters we introduced the function `plotfunc2d` to plot 2-dimensional function graphs. Here we commence with another function for two-dimensional function-graph plotting: `plot::Function2d`, a function from the `plot` library.

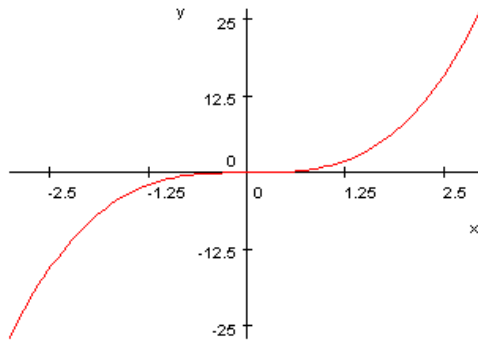
Basically this function does the same as `plotfunc2d`. The concept of the `plot` library, however, provides for many advantages when plotting complex graphical material.

Let us plot the function $f(x) = x^3$ over the domain from -3 to 3 . Contrary to the previous approach we will first create the function graph as a plot object:

- `f := plot::Function2d(x^3, x = -3..3):`

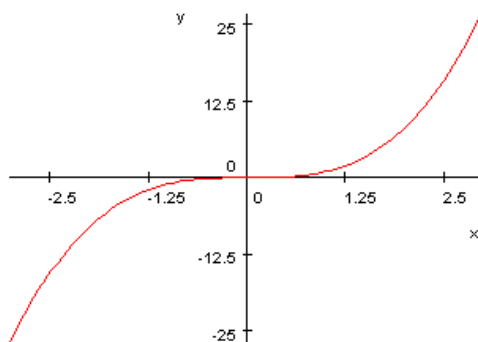
Note that the graph of f is not yet displayed on the screen. This is done with the `plot` command:

- `plot(f)`



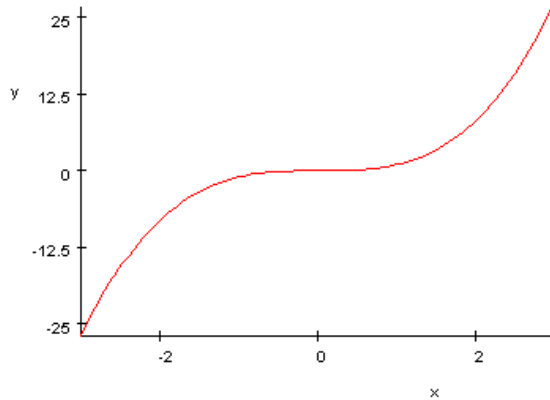
We could have done this in one line as well:

- `plot(plot::Function2d(x^3, x = -3..3))`



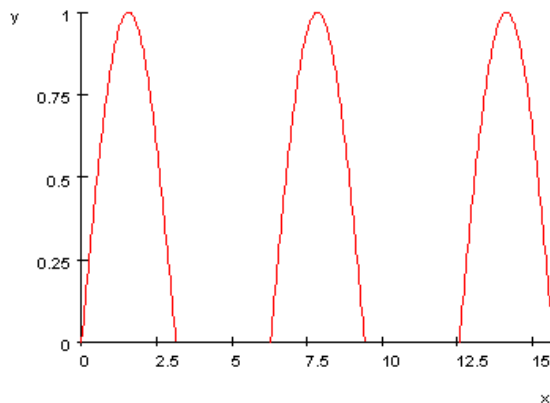
Additionally, it is possible to select axes style and axes labeling. For example:

- `plot(plot::Function2d(x^3, x = -3..3),
 Axes = Corner, Ticks = [4, 5])`



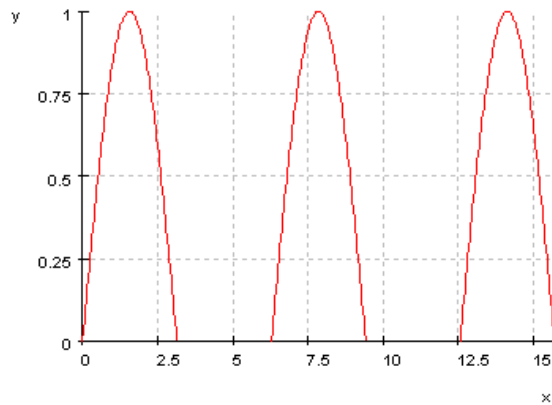
It is also possible to restrict the image range of the function to be plotted. Illustrating that we plot the graph of the sine-function over the domain $[0, 6\pi]$. In contrast to usual displaying we only want to plot the function when assuming values in $[0, 1]$:

- `f := plot::Function2d(sin(x), x = 0..6*PI, y = 0..1):
 plot(f)`



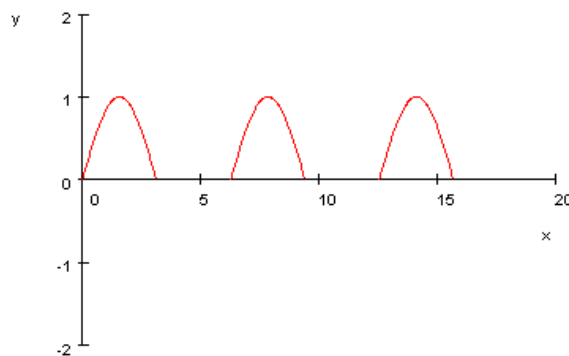
Inserting grid lines facilitates reading of coordinates. We choose the option `GridLines = Automatic`:

- `plot(f, GridLines = Automatic)`



Finally, we edit the visible area, i.e. the output region displayed. Suppose we want to plot the function on domain $0 \leq x \leq 20$ and range $-2 \leq y \leq 2$. This is provided for by the option `ViewingBox=[0,20,-2,2]`:

- `plot(f, ViewingBox = [0, 20, -2, 2])`



In order to plot several graphs within one single coordinate system, first define the corresponding graphical objects without plotting them, though:

- `f := plot::Function2d(sin(x), x = 0..2*PI, y = -2..2):`
`g := plot::Function2d(cos(x), x = 0..2*PI, y = -2..2):`
`h := plot::Function2d(tan(x), x = 0..2*PI, y = -2..2):`

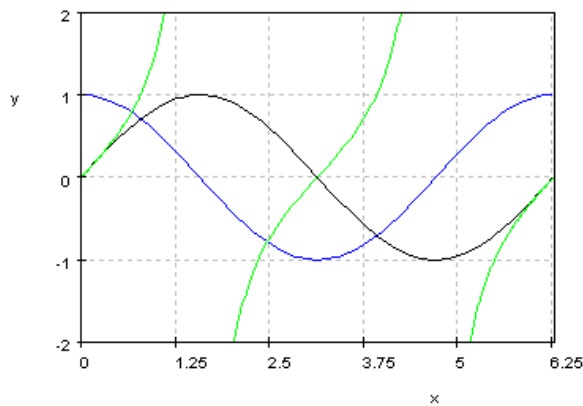
Secondly assign properties to these objects, e.g. colors. Let the graph of f be black, the one of g blue and the one of h be green:

- `f::Color := RGB::Black:`
`g::Color := RGB::Blue:`
`h::Color := RGB::Green:`

Here you see the advantage of working with the `plot` library: Conveniently work with graphical objects, manipulate and change properties without having to display the objects on screen.

Finally we plot all objects in one single coordinate system that will additionally be equipped with a coordinate grid and the axis style `Box`:

- `plot(f, g, h, Axes = Box, GridLines = Automatic)`



6.2 Working with the `plot` Library: More Examples

The `plot` library features a vast variety of graphical objects and graphical functions, some among the prominent are `plot::Function3d`, `plot::Ellipse2d`, `plot::Curve2d`, `plot::Curve3d`, `plot::Polygon`, `plot::Pointlist` and `plot::Scene`.

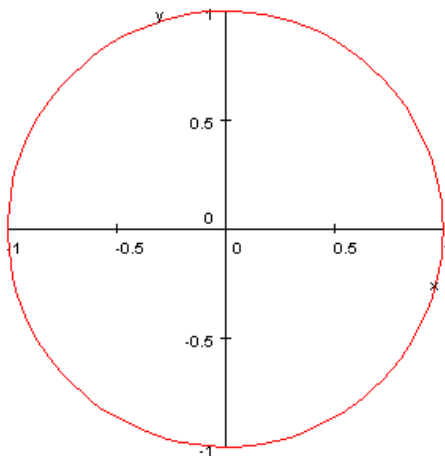
Again, all these functions do not print directly onto the screen, rather, they provide for graphical objects representing the graph to be plotted. They allow for composition in groups, editing of plot options and attributes even after creation - as we have elaborated on in the preceding section using the example of `plot::Function2d`.

Now we introduce some functions of the `plot` library permitting the rendering of more graphical objects in 2D. Let us start with rendering circles and ellipses:

- `circle := plot::Ellipse2d([0, 0], 1, 1):`

With this line the plot object `circle` is defined which will be plotted to the screen by the command `plot:`

- `plot(circle)`



Suppose we want to circumscribe the circle by a rectangle. Then define the rectangle as a separate graphical object and render both objects in a single graphical scene:

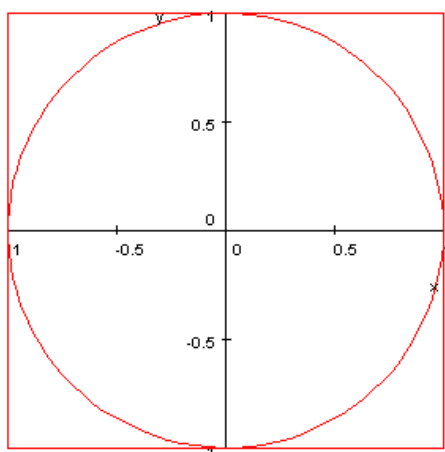
To achieve this goal we proceed as well tried before. First assign the variable `rectangle` to a newly created rectangle.

Recall that the circle is still stored to the variable `circle`, we do not have to create it again in MuPAD.

We can thus create a graphical scene that we just call `scene` and join both objects within that scene.

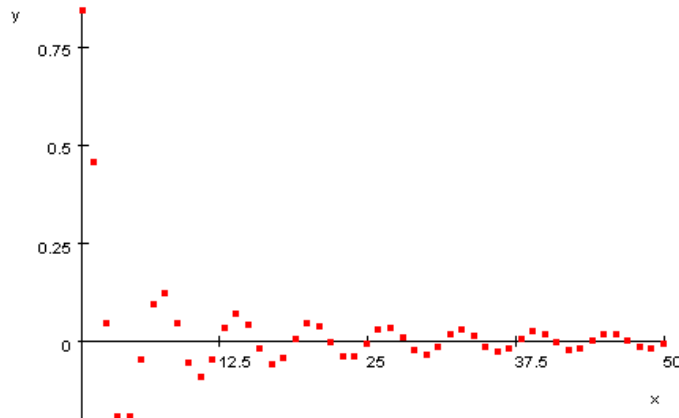
In conclusion we display the entire scene in one coordinate system as usual with the command `plot`:

- `rectangle := plot::Rectangle2d([-1, -1], 2, 2):`
- `scene := plot::Scene(rectangle, circle):`
- `plot(scene)`



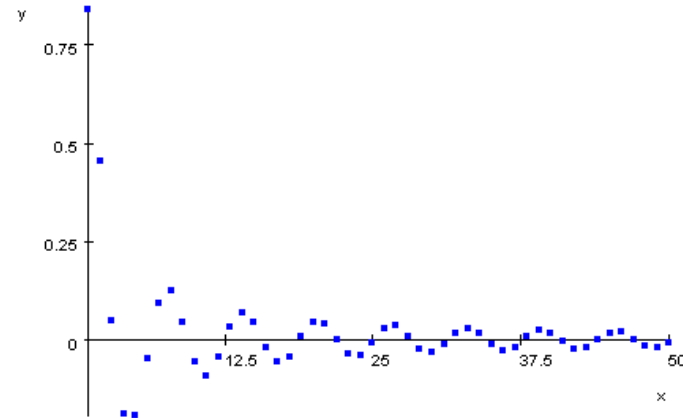
In another example consider the visualization of a sequence of numbers in MuPAD, $a_n = \sin(n)/n$, say. For plotting the first 50 elements enter:

- `PointSequence := plot::Pointlist([n, sin(n)/n] \ $ n = 1..50):`
`plot(PointSequence)`



We might want to have blue points: Just assign the new color to the attribute color of the point list `PointSequence` and plot it again:

- `PointSequence::Color := RGB::Blue: plot(PointSequence)`

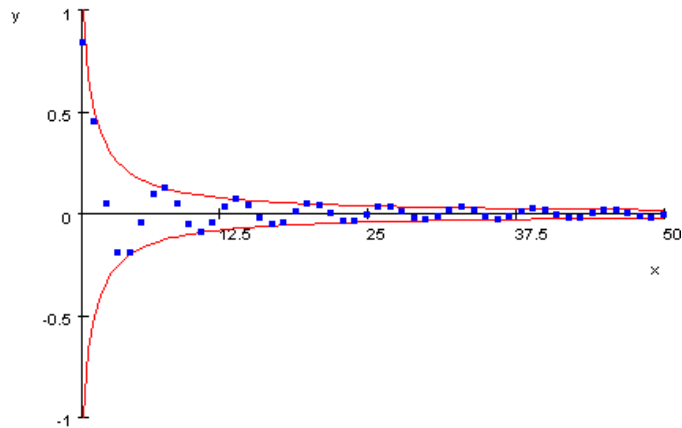


Next we will visualize the bounds of this sequence graphically. For this we enclose the sequence by the two functions $f(x) = 1/x$ and $g(x) = -1/x$.

Using the function `plotfunc2d` there would be no way to include the point list in the plot as well. With the functions of the `plot` library (its functions provide for single graphical entities such as points and function graphs that are defined as separate objects), however, our task can be accomplished nimbly and quickly by the following lines:

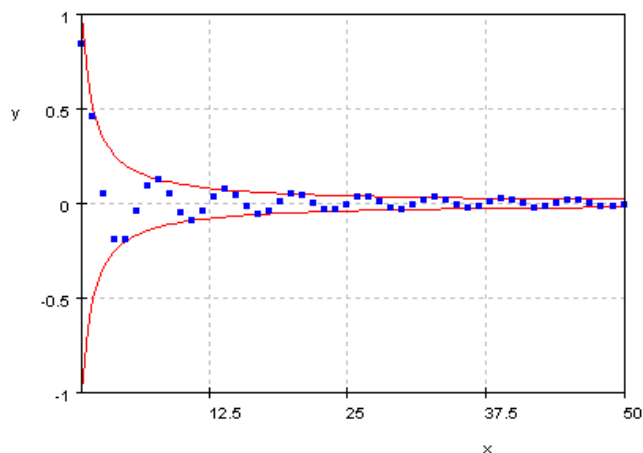
- `f := plot::Function2d(1/x, x = 0..50, y = -1..1):`
- `g := plot::Function2d(-1/x, x = 0..50, y = -1..1):`

- `scene := plot::Scene(f, g, PointSequence):`
`plot(scene)`



Here, we may likewise access and work with the object `scene`. We might want to enclose the plot in a box and insert a grid. Do not bother about any of the other objects, just pass the proper assignments to the object `scene` and plot it again:

- `scene::Axes := Box: scene::GridLines:= Automatic:`
`plot(scene)`



Exercises

- ✎ Plot the graphs of the functions `sin`, `cos` and `tan` to a common coordinate system on the interval $[-4, 4]$.
- ✎ Confine the image range of the above plot to `y = -1..1`.
- ✎ How do you plot only the positive portion of the function $f(x) = x^3 - x$?
- ✎ Insert `GridLines` as introduced above into the preceding graph.

- ✎ Using the above introduced functionality of the `plot` library, try to visualize the following issue: Every convergent sequence of real numbers is bounded.

Hint: Consider the sequences $a_n = (n+1)/(2 \cdot n+1)$, $b_n = (2 \cdot n^2)/((-1)^n \cdot n^3)$, plot them to a common scene and try to enclose them, as done above, by a tube made of two functions.

- ✎ Treat the sequence $c_n = (1+1/n)^n$ similarly. Using MuPAD, compute the limit and plot the limit as a constant function $y = e$ (parallel to the x -axis of the coordinate system) together with the sequence to a single scene.

6.3 An Application: Modeling of an Electric Bulb

Here, using MuPAD, we model an electric bulb. For the following graphical output let us define the following defaults:

- `OptionO2D := Smoothness=[4], LineWidth=12:`
`OptionS2D := GridLines=Automatic, Title="ContourLine":`
`OptionO3D := Smoothness=[1,1]:`
`OptionS3D := CameraPoint=[-30,-500,0],`
`Title="SideView":`
`Interval := x=0..5.5:`
`a :=40: b := 8/10: c := 10:`

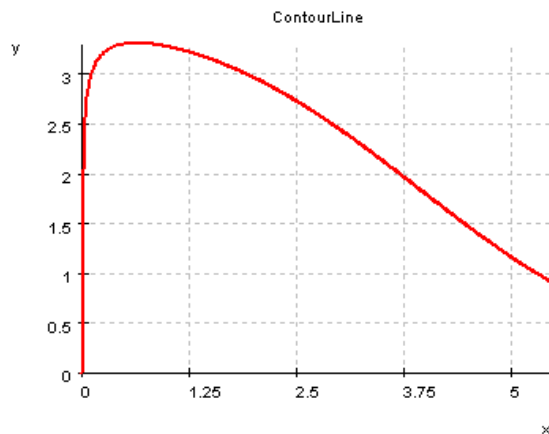
First select a function for the line bounding the glass part of the bulb. Call it `Contour`:

- `Contour = (K := x -> a * sqrt(x) * exp(-b*x) /`
`(1 + c * sqrt(x) * exp(-b*x)))`

$$\text{Contour} = \left(x \rightarrow \frac{a \cdot \sqrt{x} \cdot \exp(-b \cdot x)}{1 + c \cdot \sqrt{x} \cdot \exp(-b \cdot x)} \right)$$

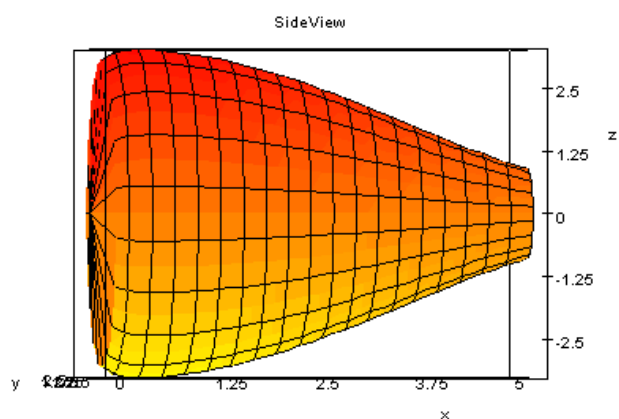
The contour function appears to have fallen from heaven. Yet is the result of a longer process testing several function graphs which we do not elaborate on here. We confine ourselves to investigate the function. Although it is rather complicated it provides for a good example of MuPAD's skills taking from us the tedious task of the forthcoming computations by hand. Sketching the graph is simply accomplished by:

- `plot(plot::Function2d(K(x), Interval, OptionO2D),`
`OptionS2D)`



Quite a start! Now rotate the graph about the given interval $0..5.5$ on the x -axis. The solid of rotation will again be sketched employing MuPAD. We use the function `plot::xrotate` from the MuPAD `plot` library.

- `plot(plot::xrotate(K(x), Interval, OptionO3D),
OptionS3D):`



The volume of this solid of rotation about the interval $0..5.5$ is given by the following definite integral:

- `Volume = (V := PI * int(K(x)^2, Interval))`

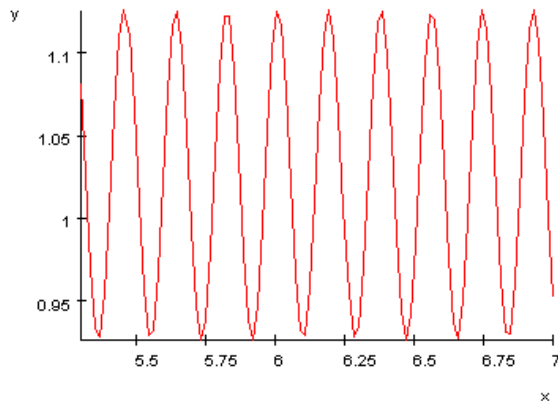
$$\text{Volume} = \pi \cdot \int_0^{5.5} \frac{1600 \cdot x \cdot e^{\left(-\frac{4 \cdot x}{5}\right)^2}}{\left(10 \cdot \sqrt{x} \cdot e^{\left(-\frac{4 \cdot x}{5}\right)} + 1\right)^2} dx$$

As usual approximate the volume by:

- `float(V)`
109.3625511

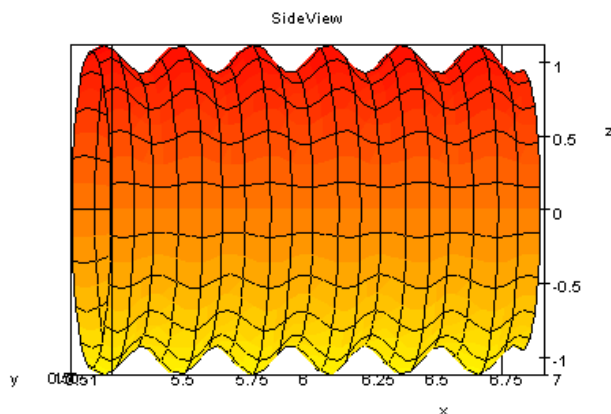
Thus we are done with the glass part of the electric bulb and left with its screw thread that is screwed in into the lamp holder. We choose the sine function for the thread and increase the period drastically to simulate the wave-like form of the thread:

- `plot(plot::Function2d(1/10*sin(400*x) + K(5.3) + 0.03,
x = 5.3..7))`



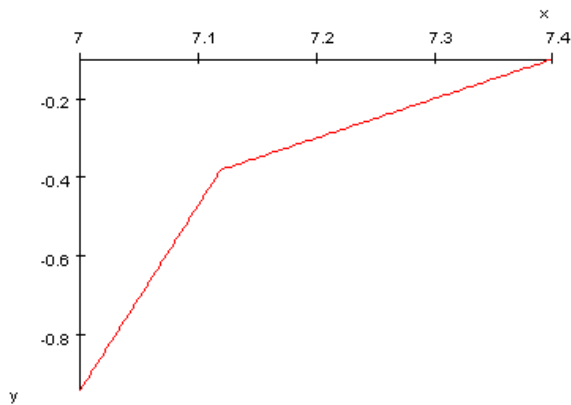
You will have noted that this thread will not do to really screw in our bulb. For the sake of simplicity we allowed ourselves to cheat a bit contenting with this fix. Rotating this function about the x -axis as well we have the following picture:

- `plot(plot::xrotate(1/10*sin(400*x) + K(5.3) + 0.03,
x = 5.3..7, OptionO3D),
OptionS3D)`



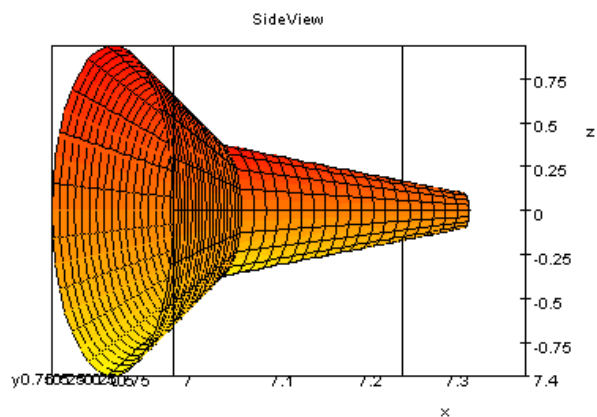
Finally we need the point of contact supplying our bulb with electric current. We place the point of contact right left to the thread; it will be modeled by simple linear functions:

- `plot(plot::Function2d(4.7*(x-7.2), x = 7..7.1189),
plot::Function2d((x-7.5), x = 7.1189..7.4))`



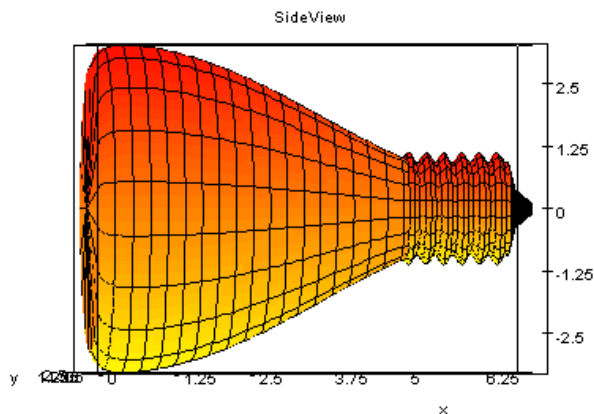
At first glance choosing these two functions might not make too much sense. Take into account, however, that the interval $7..7.2$ in concern is negligibly small. The solid of rotation is now:

- `plot(plot::xrotate(4.7*(x-7.2), x = 7..7.1189,
OptionO3D),
plot::xrotate((x-7.5), x = 7.1189..7.4,
OptionO3D),
OptionS3D)`



Combining all objects to a one big solid of rotation yields:

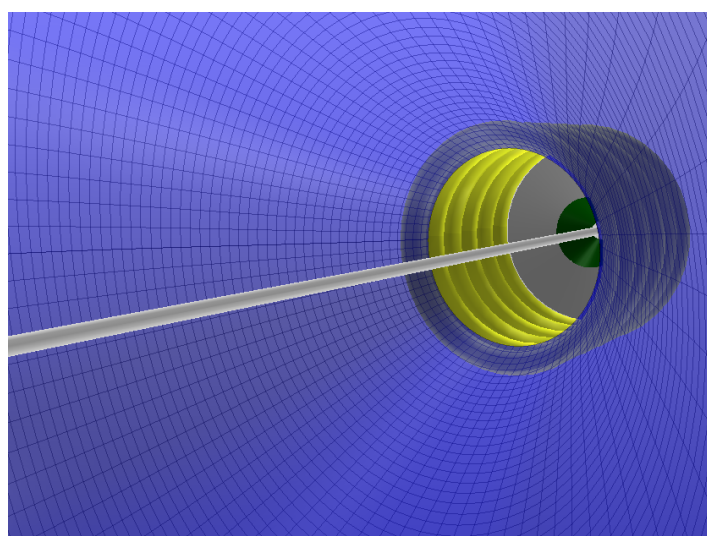
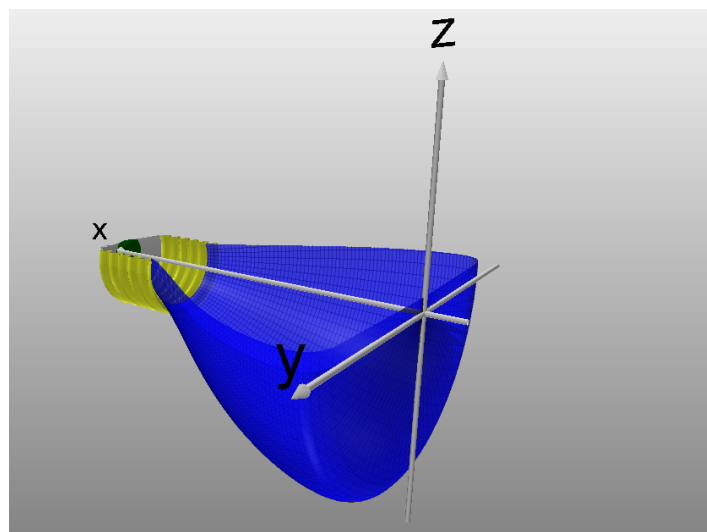
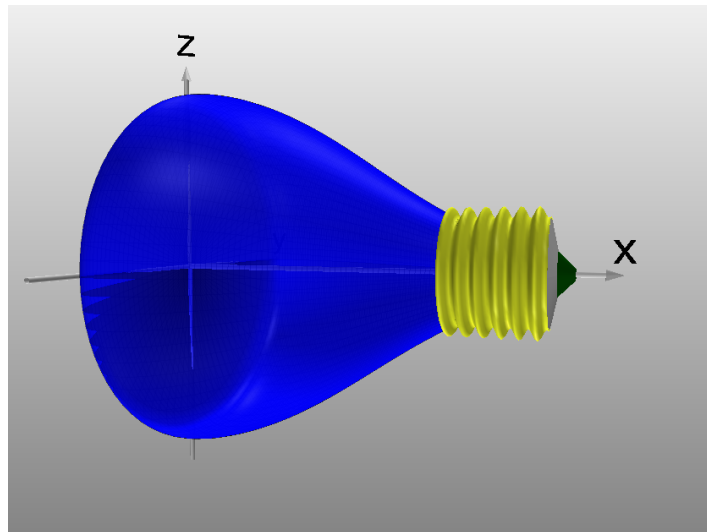
- `plot(plot::xrotate(K(x), Interval, Smoothness = [3, 3]),
plot::xrotate(1/10*sin(400*x) + K(5.3) + 0.03,
x = 5.3..7, OptionO3D),
plot::xrotate(4.7*(x-7.2), x = 7..7.1189, OptionO3D),
plot::xrotate((x-7.5), x = 7.1189..7.4, OptionO3D),
OptionS3D)`



Hints

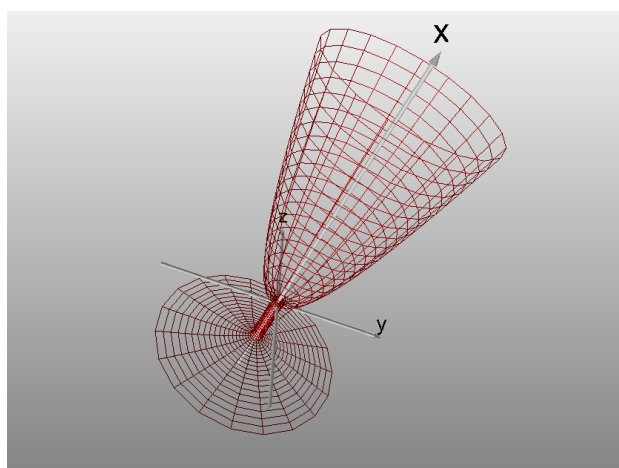
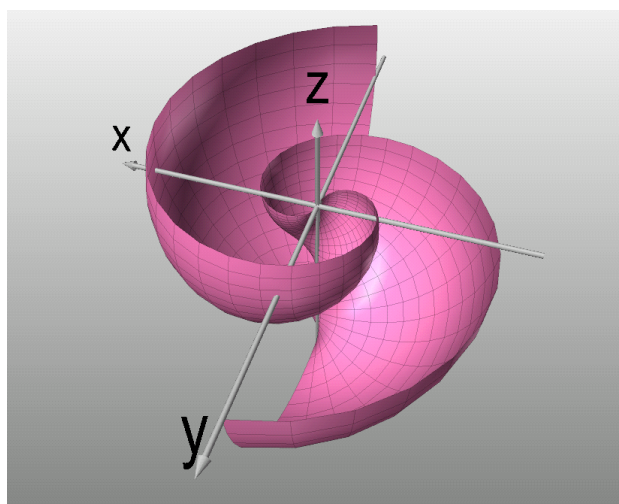
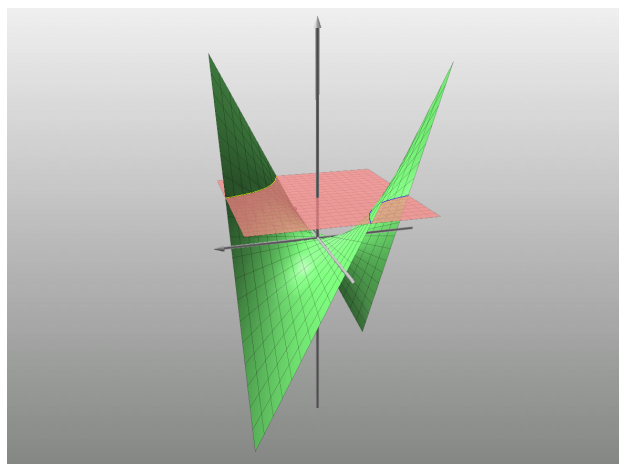
- ☞ Give this plot a finishing touch: In the new 3D-Viewer of MuPAD Pro 2.5 for Windows activate the plot by a double click. Then select “Edit” in VCam’s menu bar and choose “Open in 3D-Viewer.” Here the plot will be rendered nearly in photo realistic quality.
- ☞ In VCam’s command bar you will find symbols for zooming in, rotating and many more. The interface of the viewer is self explanatory. Clicking with the right mouse button onto a graphical object a window will be opened. There you may lay a mesh over the object, tune properties like transparency and colors, create a skeleton (wire frame) from the object and many more.

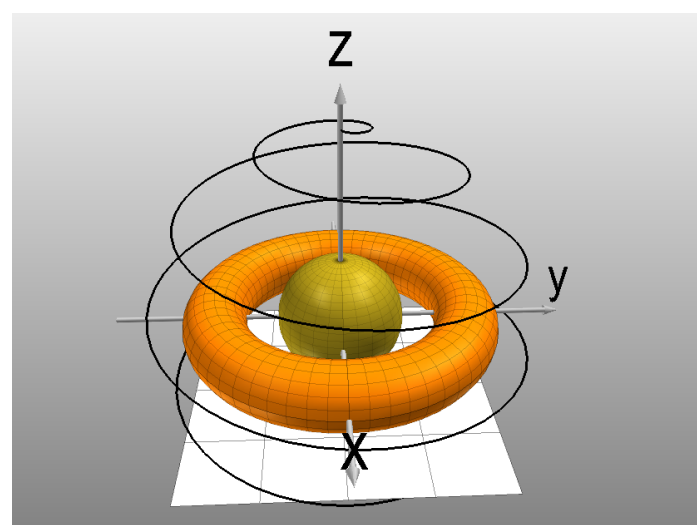
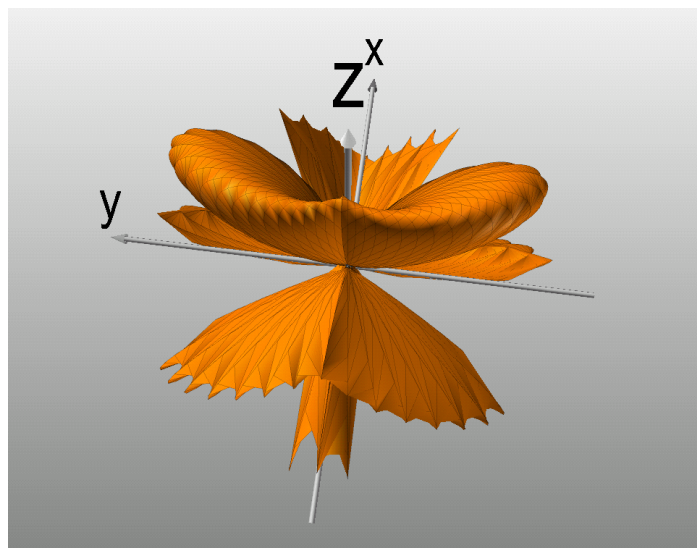
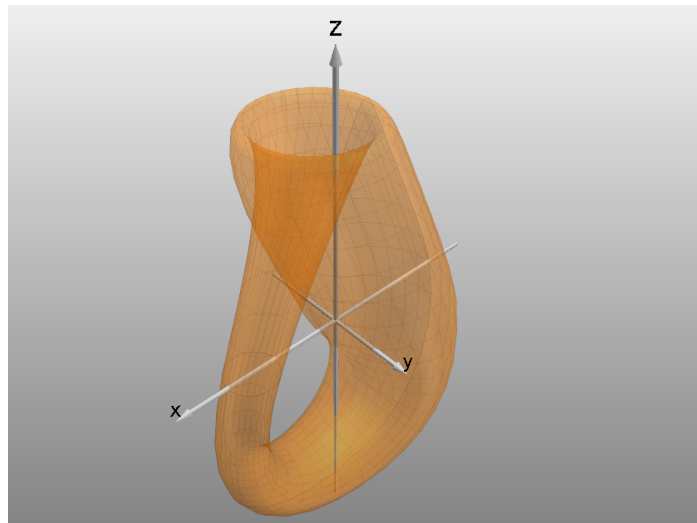
We have included three plots of our bulb illustrating some capabilities of the 3D-Viewer, like zooming and rotating, transparent surfaces, mesh lines and cutting threedimensional objects with respect to the z-axis:



6.4 3D-Graphics Gallery

In this section we list some further 3D graphics created with MuPAD 2.5 for Windows:





Bibliography

- [1] Baumann: *Analysis*, 1. Klett, 1997
- [2] Burkhart: <http://www.weihenstephan.org/burkhart/mupad/mupad.html>
- [3] Eckart, Jehle, Vogel: *Analytische Geometrie*, BSV Mathematik, 1994
- [4] Feuerpfeil, Heigl, Wiedling: *Praktische Stochastik*, BSV Mathematik, 1994
- [5] Grabinger: *Projekte und Aufgaben zur Analytischen Geometrie*, Schroedel, 1999
- [6] Griesel, Postel: *Grundkurs Analysis Gesamtband*, Mathematik heute, Schroedel / Schöningh, 1991
- [7] Heugl, Klinger, Lechner: *Mathematikunterricht mit Computeralgebra-Systemen*, Addison-Wesley, 1996.
- [8] Jahnke, Lauter, Rüdiger: *Mathematik Sekundarstufe II*, Analysis Leistungskurse. Cornelsen, 1993
- [9] Oevel, Postel, Rüschler, Wehmeier: *The MuPAD Tutorial*, Springer, 2000
- [10] Oevel: *Einführung in die numerische Mathematik*, Spektrum, 1996

Index

`:=`, 8
`All`, 36
`DIGITS`, 8, 9
`Diagonal`, 32
`Dom`, 34
 `Dom::Complex`, 34
 `Dom::Float`, 34
 `Dom::Integer`, 34
 `Dom::Rational`, 34
 `Dom::Real`, 34
`PI`, 7
`$`, 11, 14, 47
`delete`, 8, 9
`diff`, 10, 11, 13, 14, 22
`expand`, 9, 40
`float`, 7, 11
`info`, 21, 35
`int`, 14, 22
`linalg`, 22, 25, 35
 `addCol`, 36
 `addRow`, 36
 `angle`, 40
 `cols`, 28
 `det`, 40
 `expr2Matrix`, 36
 `gaussElim`, 36
 `matlinsolve`, 36
 `multCol`, 36
 `multRow`, 36
 `ncols`, 27
 `nrows`, 27
 `rows`, 28
 `transpose`, 31
`normal`, 9
`numeric`, 17, 21
 `det`, 22
 `fsolve`, 21
 `solve`, 17
`plotfunc2d`, 13
`plot`, 47, 49, 53–55, 57, 58
 `Curve2d`, 53
 `Curve3d`, 53
 `Ellipse2d`, 53
 `Function2d`, 53
 `Function3d`, 53
 `Pointlist`, 53
 `Polygon`, 53
 `Scene`, 53
 `bars`, 47
 `xrotate`, 58
`random`, 43, 46
`simplify`, 9, 22
`solve`, 13, 17, 24
`sqrt`, 7
`stdlib`, 22
`student`, 17, 18
 `plotRiemann`, 17
 `riemann`, 17
`subs`, 11, 13

Mathematics made anew

Volume 1

MuPAD is a computer algebra system which can deal with problems in mathematics as well as in natural sciences and engineering. It is a very valuable utility for pupils and students, teachers and scientists.

This booklet offers an easy, fast and very practical access to MuPAD. It assists in using MuPAD directly, without forcing the reader to deal with too much technical details.

All included examples in analysis, linear algebra und stochastics come from applications in school and are kept as simple as possible in order to offer pupils and undergraduate students a easy introduction to MuPAD. This supports teachers in integrating MuPAD into their lectures and lessons.

<http://www.sciface.com>

schule@mupad.de

SciFace Software • Paderborn

SciFace

