

Frank Postel

MuPAD as a Tool, Tutee and Tutor

Primary talk at the ACDCA '99

presented by

Frank Postel

MuPAD Research Group
Department of Mathematics
University of Paderborn
Warburger Straße 100
33095 Paderborn
Germany

frankp@mupad.de

www.mupad.de

Table of Contents

1. The Computer as a Tool and Tutee	1-1
1.1 Introduction	1-1
1.2 MuPAD – A Flexible and Modern Computer Algebra System.....	1-1
1.3 MuPAD as a Tool and Tutee	1-2
1.4 Some Critical Inspections.....	1-10
2. The Computer as a Tutor.....	2-11
3. A Tutorial System for Introductory Algebra	3-12
3.1 The Components of the Software	3-12
3.1.1 The Environment Module.....	3-12
3.1.2 The Expert Module.....	3-13
3.1.3 The Tutor Module.....	3-13
3.2 Technical Aspects of the Expert Module.....	3-14
3.3 Open Technical and Didactical Questions.....	3-15
3.4 Summary	3-15
4. Software Components for Mathematical Documents	4-16
4.1 Graphical User Interfaces for Mathematical Electronical Documents	4-16
4.1.1 A Calculator Control for Mathematical Computations.....	4-16
4.1.2 A Flexible and Interactive Graphics Control.....	4-18
4.2 Summary	4-20
5. References	5-20

1. The Computer as a Tool and Tutee

1.1 Introduction

Computer nowadays play three different roles in mathematical education, and I want to stress on two of them for the introductory part of this article:

- i) **The Computer as a Tool:** Tools such as computer algebra systems (Derive, Maple, MuPAD, Mathematica, etc.), spreadsheets (like Microsoft Excel) or mathematical microworlds¹ (like Cabri géomètre) provide the students with powerful learning aids, and they extend the class of problems manageable by students.
- ii) **The Computer as a Tutee:** The user teaches the computer to perform complex or repetitive routine tasks by assembling primitive computer procedures into algorithms and writing those algorithms in a language that the computer understands. The user teaches the computer how to assemble its existing knowledge into more powerful tools. [Fe91]

Before I illustrate how these two aspects can be achieved with the computer algebra system MuPAD, I give a brief introduction to MuPAD for readers who are new to this software.

1.2 MuPAD – A Flexible and Modern Computer Algebra System

MuPAD is a general computer algebra system such as Derive, Maple, Mathematica or Macsyma, for instance. It has very comfortable interactive user interfaces. Some basic features are:

- Symbolic and exact computations of limits, derivatives, sums, definite and indefinite integrals, solutions of equations and differential equations, inequalities, systems of algebraic equations and many more.
- Numerical computations (solving of equations, system of equations, integrals, etc.) of arbitrary precision size.
- Functions from several domains of mathematics, such as linear algebra, statistics, combinatorics, number theory, commutative algebra, and many more.
- Menu-driven creation and manipulation of two and three-dimensional graphical scenes.
- Easy to learn and very comfortable programming language with Pascal-like syntax and object oriented facilities.
- Extended online-help, organized as hypermedia documents. Each MuPAD command is explained by many examples which can be easily copied to a MuPAD session just by clicking on a marker position inside the text. This ensures an easy way to become familiar with MuPAD.

Figure 1 shows a screenshot of a running MuPAD application with several so called MuPAD notebooks. The version shown is the german release of MuPAD Pro 1.4.2 for Windows 9x/NT (an english version exists as well). In addition to MuPAD computations MuPAD notebooks include images, text and OLE objects. They can be saved, edit or printed, as well as included in Microsoft Word, Microsoft Excel, Internet Explorer or any other application which can include OLE objects. The academic version of this release is distributed by Springer Verlag. It consists of MuPAD Pro 1.4.2 and the MuPAD Tutorium, a tutorial introduction to the system (see also [OePoRüWe99]) for those who are new to computer algebra. The academic version is supposed to be used by students and teachers, and is offered for a special price². MuPAD is also available for Linux and MacOS 7.x.

¹ The term microworld was originally used by Papert to describe "a computer-based interactive learning environment where the pre-requisites are built into the system and where learners can become active, constructing architects of their own learning". In: [Ta94]

² "MuPAD Pro Akademische Edition, Version 1.4.2". CD-ROM mit Handbuch, ISBN 3-540-14790-X, 1999. http://www.springer.de/cgi-bin/search_book.pl?isbn=3-540-14790-X

MuPAD is known to be an *open* and *flexible* computer algebra system as the source code of most of the mathematical functions can be inspected, the system can be extended by user-defined routines and user-defined data types, and C/C++ code can be dynamically linked to the MuPAD kernel for pure speed and flexibility. In addition, several protocols to communicate with other tools are supported.

For more information about MuPAD have a look at www.sciface.com, send an email to info@sciface.com or write to SciFace Software GmbH & Co. KG, Technologiepark 12, D-33100 Paderborn, Germany.

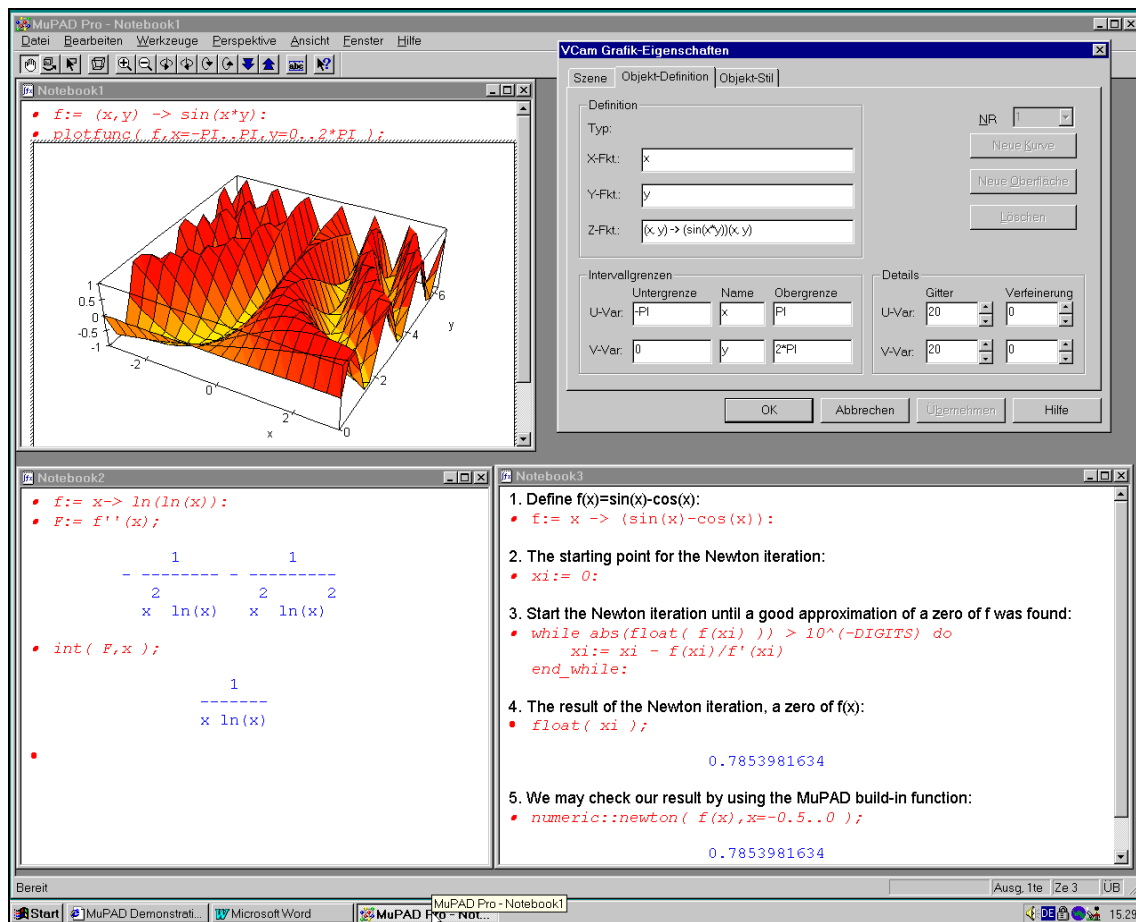


Figure 1: MuPAD Pro 1.4.2 for Windows 9x/NT (german release)

1.3 MuPAD as a Tool and Tutee

In MuPAD the student works within an open environment giving him full access to the mathematical knowledge. This may motivate him or her to discover mathematical knowledge in a more experimental way (which may support traditional teaching methods), it may allow him to focus on specific learning goals or to deal with advanced topics of mathematics.

It is not a secret that the technical overhead for using tools such as MuPAD, Maple or Mathematica is quite high and that parts of these technical concepts are hard to teach, or are even hard to understand for the teacher and the student. Let me give some few examples:

- 1) Different possibilities exist in MuPAD for declaring a function in one variable and hence different ways for evaluating a function at certain points:

```

• f := x^2-2:
• eval( subs( f, x=1 ) )
  -1

• f := x -> x^2-2:
• f(1)
  -1

```

- 2) One may prefer the second way but what happens when computing the derivative of f ?

```

• f1:= f'(x)
  2 x

• f1(1)
  2 x(1)

```

What happens with the input $f1(1)$? It seems that the function $f1$ was not evaluated at $x=1$. This unexpected behavior of MuPAD is based on the fact that the operation for differentiation does not return a function but an expression, so the information about the function variable was lost.³ The reader may try this in his or her favourite tool! (Similar problems occur when adding or multiplying functions, for instance.)

- 3) Does your favourite tool support the working with sequences? One could handle sequences as functions, but this way will not be conducive to the conceptual understanding of the fundamental differences between sequences and functions. Especially when introducing limits of functions a different handling of sequences and functions in a computer algebra system is needed.

I could extend the list with odd examples where the tool does not act as one might expect and therefore forces the teacher to explain the students how to avoid the weaknesses of the tool.

This motivated us to develop packages which serve for certain areas of school mathematics such as calculus or linear algebra and analytical geometry. These packages support the teacher in the development of teaching materials and support the student in working with the system by decreasing the technical overhead.

Let me introduce the new MuPAD package **analysis** for doing calculus. It offers functions which serve as easy-to-use and powerful utilities for the student and teacher:⁴

A New Object Class for Functions
 We define the function $f(x) = \sin(\pi x)$:

```

• f:= x :-> sin(PI*x)
  x :-> sin(PI*x)

```

You may noticed the new operator `:->` for the definition of a function. In fact, it returns an object (a real function) of a new object class. Such objects can be manipulated in a more appropriate way than procedures, the standard way for defining functions in MuPAD. For example, to raise the second power of f we just have to enter:

```

• g:= f^2
  x :-> sin(PI*x)^2

```

Note that the result of such computations is again a function in x . We may evaluate g at $x=1$ in the usual way:

```

• g(1)
  0

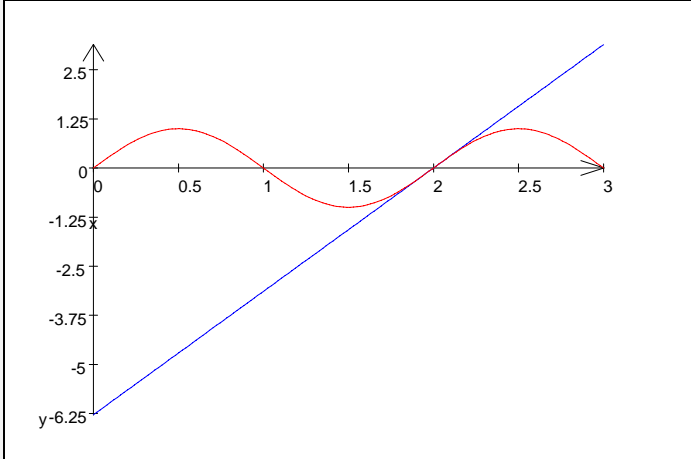
```

We plot f together with its tangent at $x=2$:

```

• draw( [f,tangent( f,2 )], x=0..3, [RGB::Red, RGB::Blue] )

```



The functions `tangent` and `draw` are part of the **analysis** package. `draw` allows to plot different types of objects (tangents, sequences, functions, table of values, ...) by a single command. The differentiation of f can be computed as follows:

³ In fact, MuPAD evaluates the input $f1(1)$ as follows: $f1(1) = (2*x)(1) = 2(1)*x(1) = 2*x(1)$.

⁴ The language used by these packages (function names as well as warning and error messages given by the functions of the package) is german. Anyway, they may be translated into the english language, for example, as I did it for this article.

```

• f1:= f'
      x :-> PI*cos(PI*x)

• f2:= f''
      x :-> -PI^2*sin(PI*x)

```

Note that f1 and f2 are again functions in x. The standard MuPAD function `int` also works for the new objects of real functions:

```

• int( f );
      x :-> -1/PI*cos(PI*x)

```

A New Object Class for Sequences

The package `analysis` defines a new object class for sequences:

```

• a := n --> n^2; b := n --> sin(n)
      n --> n^2
      n --> sin(n)

```

```

• a(5)
      25

```

Look what happens if we try to evaluate a sequence for non-natural indices:

```

• a(0)
Error: sequences are defined only for natural arguments

```

```

• a( 1/3 )
Error: sequences are defined only for natural arguments

```

We can easily calculate with sequences, e.g., compute the sum or product of two sequences:

```

• a + b, a * b
      n --> sin(n)+n^2, n --> n^2*sin(n)

```

or to compute the limit of 1/a and b when n tends to infinity:

```

• limit( 1/a ), limit( b )
      0, undefined

```

Visualization of Sequences

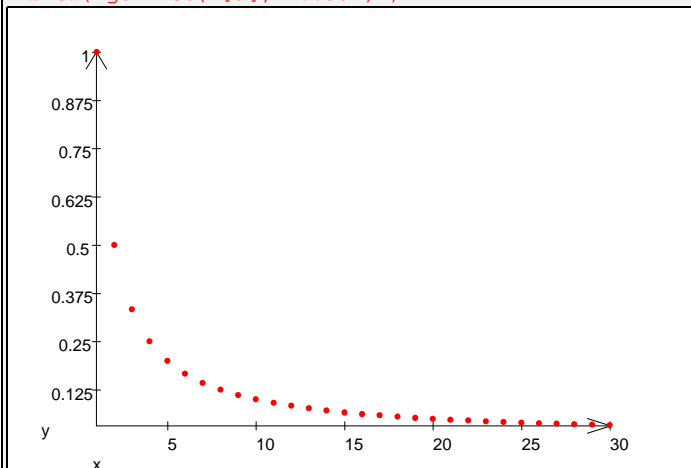
Sequences may be visualized in different ways:

1. The function `genPlot` generates a simple plot of the values of a:

```

• a := n --> n^2;
• show( genPlot( [a], 1..30 ) )

```

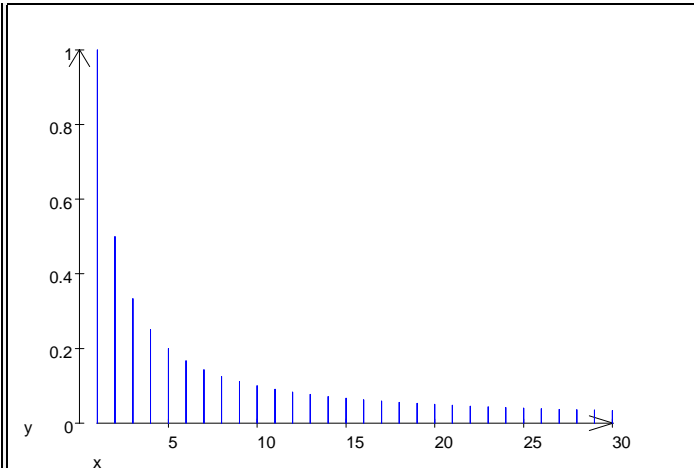


2. The function `Sequence::genBars` is similar to `genPlot` but connect each point by a vertical line:

```

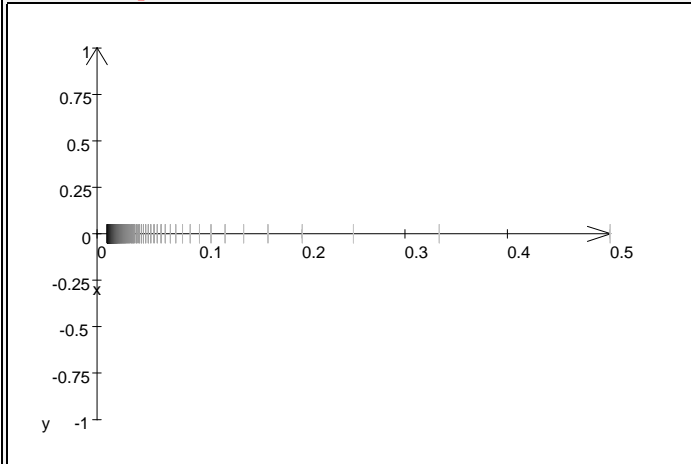
• show( Sequence::genBars( a, 1..30 ) )

```



3. The function `Sequence::visLim` shows a „history“ of the values, which means, that $a(n)$ is represented by a small vertical bar which grey scale depends on the value of n :

```
• show( Sequence::visLim( a,1..100,0..0.5, Colored ) )
```



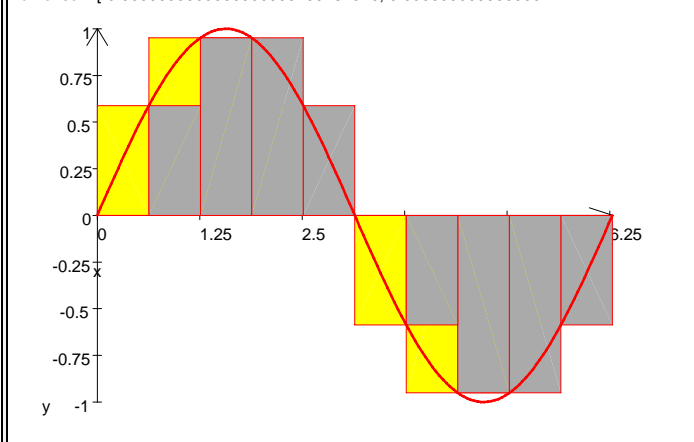
And More?

It would be out of the scope of my article to demonstrate the full functionality of the new package `analysis`. Let us finish this introduction with some functions which may be useful for the subject of numerical integration:

```
• intRiemann( sin(x),0..2*PI,N=10 )
[-0.0000000000000000005109182370, 0.0000000000000000003065509422]
```

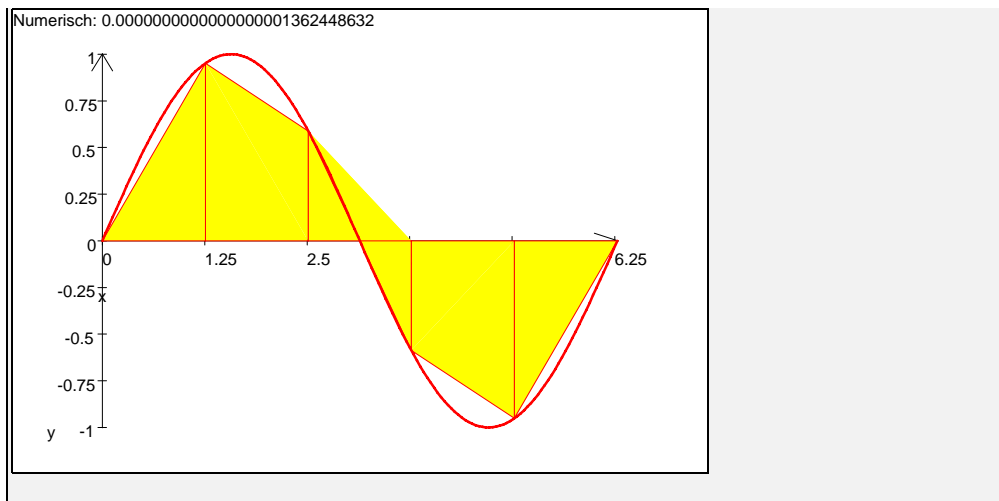
```
• show( graphRiemann( sin(x),0..2*PI,N=10 ) )
```

Numerisch: [-0.0000000000000000005109182370, 0.000000000000000000000000]



```
• intTrapezoid( sin(x), x=0..2*PI, N=5 )
0.000000000000000000000001362448632
```

```
• show( graphTrapezoid( sin(x), x=0..2*PI, N=5 ) )
```



We wanted to enable the student to describe the solution of a given problem in a similar way as when using pencil and paper. The student may translate his or her solution into small procedures for later use and therefore extend the package with further routines (the "tutee" aspect). The following example introduces the Newton iteration formula for finding a zero of a function:

```

The Newton Iteration
We define a function in x:
• f:= x :-> x^2 - 3
  x :-> x^2 - 3

and give a starting point for the iteration:
• x0:= 5
  5

The first approximation of a zero of f is given by the zero of the tangent of f at x0:
• t1:= tangent( f, Xo = x0)
  x :-> 10*x - 28

• s1:= solve( term(t1),x )
  {14/5}

We extract the solution and name it x1. Do we already have a good approximation?
• x1:= op( s1,1 ): float( f(x1) )
  4.84

No, we don't have and therefore continue the iteration with the new starting point x1:
• t2:= tangent(f, Xo = x1)
  x :-> (28/5)*x - 271/25

A better approximation of a zero of f is given by the zero of the tangent of f at x1:
• x2:= op( solve( term(t2),x ),1 )
  271/140

• float( f(x2) )
  0.7469897959

But the approximation is still not good enough, so we have to continue the iteration process:
• t3:= tangent(f, Xo = x2)
  x :-> 271/70*x - 132241/19600

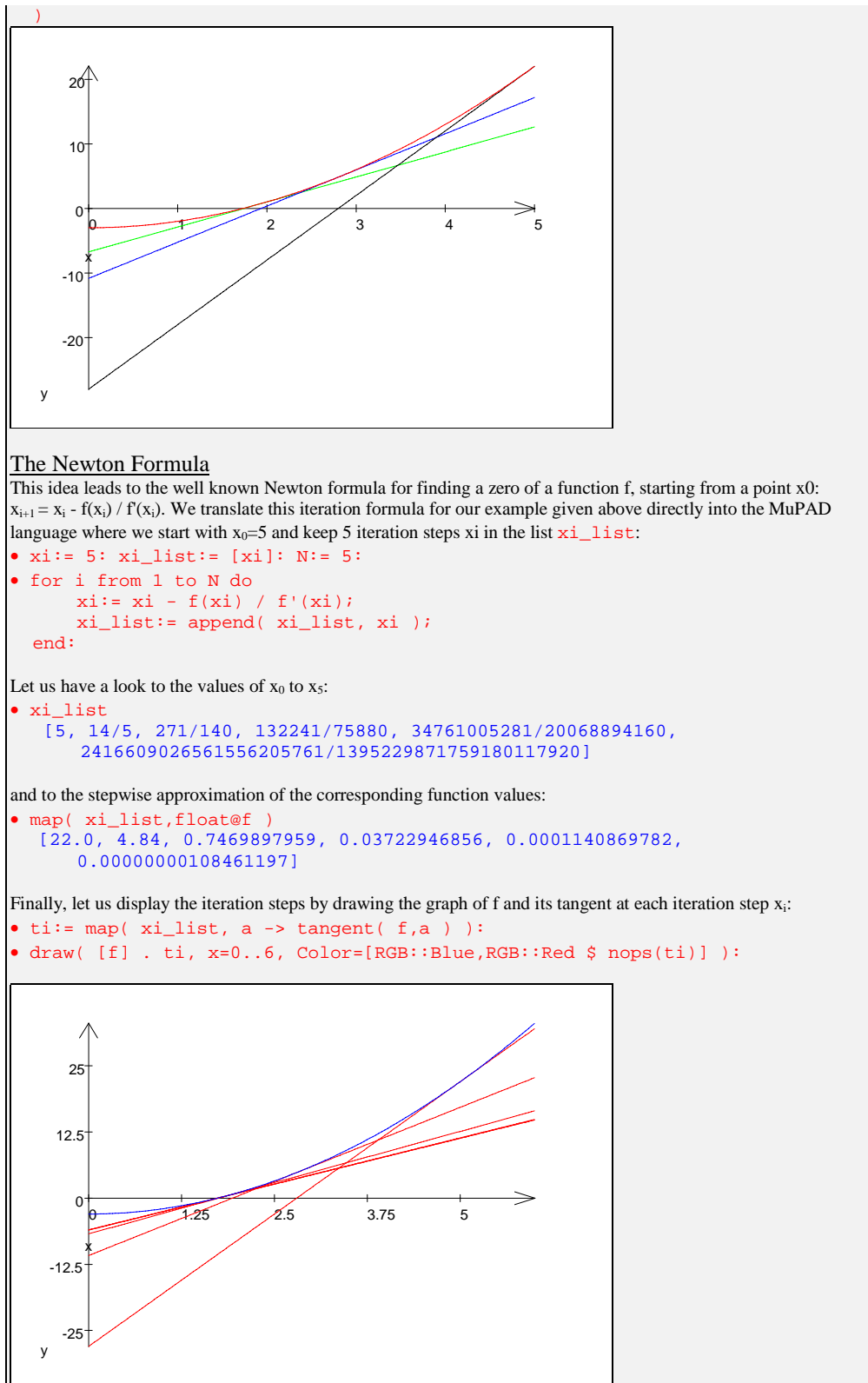
• x3:= op( solve( term(t3),x ),1 )
  132241/75880

• float( f(x3) )
  0.03722946856

Here we stop the iteration (although we still don't have a good approximation) and draw a graphic of the iteration
steps we performed:

• draw( [f,t1,t2,t3],x=0..5,
  Color=[RGB::Red,RGB::Black,RGB::Blue,RGB::Green]

```

As we have seen MuPAD computes the derivative and antiderivative of functions. When a student just has learned what a derivative is and how to compute it, he or she can use MuPAD for checking results of his or her solutions computed by hand. But what if the result of the student is not correct? MuPAD, and usually any other tool mentioned above, does not give any help or information about the computation steps (in fact, the algorithms used by the system must not be consistent with the way the student would compute, and symbolic integration is a very good example for this).

To support the student's learning process we extended the package by functions for differentiation and symbolic integration which give the student help on demand. Let me give some examples:

Help on Differentiation

The MuPAD function `diff` for the differentiation of functions does not give any information about the computation strategy:

```
• diff( ln(x)*x, x )
                                ln(x) + 1
```

In order to give the student some more information how to perform the differentiation the package `analysis` offers a new function `differentiation`:

```
• expose( differentiation ( ln(x)*x ) )
diff(x :-> x*ln(x))
|
|-- apply the product-rule:
|   |
|   +-- (u * v)' = u'v + v'u
|   |
|   +-- x :-> x
|   |   |
|   |   |-- = u, u' = 1
|   |
|   +-- x :-> ln(x)
|   |   |
|   |   |-- = v, v' = x^(-1)
|   |
|   |-- x :-> ln(x) + x/x
|   |
|   |-- the result
```

The result of the function call is a tree which reflects the way how the differentiation can be obtained. Another example:

```
• expose( differentiation( sin( ln(x) ) ) )
diff(x :-> sin(ln(x)))
|
|-- apply the chain-rule
|   |
|   +-- (f(g))' = f'(g) * g'
|   |
|   +-- {z2 :-> sin(z2), z2 = ln(x)}
|   |   |
|   |   |-- the outer term (f)
|   |   |   |
|   |   |   |-- cos(z2)
|   |   |   |   |
|   |   |   |   +-- outer differentiation using z2 = ln(x)
|   |   |   |   |
|   |   |   |   |-- result after resubstitution: cos(ln(x))
|   |   |
|   |   +-- x :-> ln(x)
|   |       |
|   |       |-- the inner term (g), ln(x)' = x^(-1)
|   |
|   |-- x :-> cos(ln(x))*x^(-1)
|   |
|   |-- the result
```

Help on Integration

The MuPAD function `int` performs indefinite and definite integration but without giving any information about the computation strategy:

```
• int( x+sin(x),x )
                                2
                                x
                                -- - cos(x)
                                2
```

The package `analysis` offers the student two functions which gives help on demand. The function `intTip` gives the student a hint which integration method could be applied to compute the indefinite integral of a function `f`. Some examples:

```
• expose( intTip( x+sin(x) ) )
int(x :-> x + sin(x))
|
|-- apply the sum-rule (e.g., apply integration to each subterm)
|   |
|   +-- int( S1 + S2 ) dx = int( S1 ) dx + int( S2 ) dx
|   |
|   +-- x :-> x
```

```

      |-- = S1
      |-- x :-> sin(x)
      |-- = S2

```

Similar to the function `differentiation` the result of the function call is a tree reflecting the structure of the integration method:

```

• tip:= intTip( (x^2+1)/(x-1) ): expose( tip )
  int(x :-> (x^2 + 1)/(x - 1)
    |-- start with polynomial division
    |-- int( p(x) / q(x) )dx = int( g(x)dx + int( r(x) / q(x) )dx
    |-- x :-> x + 1
    |--   |-- = g(x)
    |-- x :-> 2*(x - 1)^(-1)
    |--   |-- = r(x) / q(x)

```

Here the system responds with the hint first to perform a polynomial division on $f(x)$. As you can see the computation is done step by step.

If the student needs more help he or she has to apply `intTip` again on the operands of the given tree. We demonstrate this to compute the integration of the rest of the polynomial division which is the third operand of the tree:

```

• rest:= term( note( tip )[3] )
      2
      ----
      x - 1

• tip2:= intTip( rest ): expose( tipp2 )
  int(x :-> 2*(x - 1)^(-1)
    |-- apply the factor-rule
    |-- int( s * f(x) ) dx = s * int( f(x) ) dx, s aus R
    |-- 2
    |--   |-- = s
    |--   |-- (x - 1)^(-1)
    |--     |-- = f(x)

```

The second function is named `integration`. It applies a certain integration method chosen by the student and returns a tree reflecting the way how the method was applied:

```

• expose( integrate(x+sin(x),Mode="Sum" ) )
  int(x :-> x + sin(x)
    |-- apply the sum-rule (e.g., apply integration to each subterm)
    |-- int(S1 + S2) dx = int(S1) dx + int(S2) dx
    |-- x :-> x
    |--   |-- = S1
    |-- x :-> sin(x)
    |--   |-- = S2
    |-- x :-> int(x, x) + int(sin(x), x)
    |--   |-- the result

```

The system responds with an error message if the chosen method could not be applied.

This package is currently under development and will be distributed within the next major MuPAD release. The MuPAD inputs given above do not run under the current MuPAD release 1.4.2!

Please note that the student deals with new objects and functions but is still working in the open environment provided by the computer algebra system. The package only *extend* the MuPAD system thanks to its modern and powerful programming language by new objects and functions designed for the use in classrooms.

1.4 Some Critical Inspections

Areas still exist where mathematical software tools do not satisfy most of the didactical needs. Introductory algebra (high school Algebra I) may serve as an example here. In order to be suitable for this field, a mathematical software tool must be able to:

- Handle integers, rational numbers (both in exact and mixed representation), decimal fractions, unknowns and parameters as well as equations and system of equations.
- Distinguish between the marked and the invisible multiplication character.

- Distinguish between the two characters ":" and "/" for division, such as $\frac{4}{6} = \frac{4:2}{6:2} = \frac{2}{3}$.

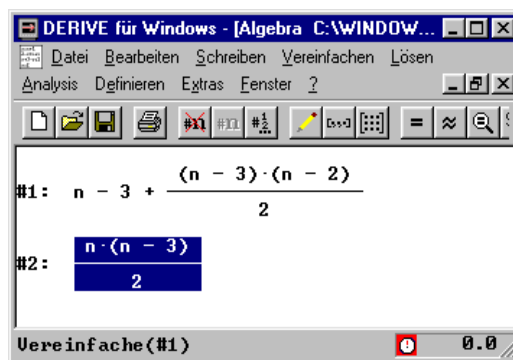
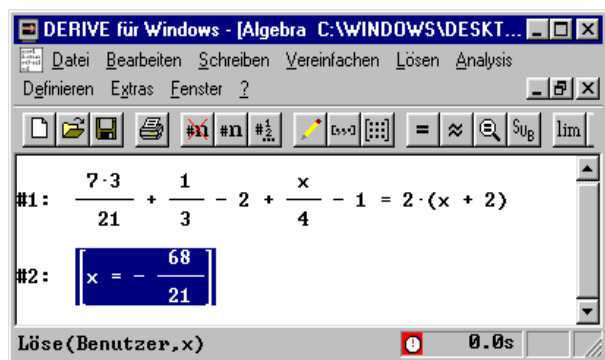
- Keep the order of subexpressions like in $2x + x = x(2+1) = x \cdot 3 = 3x$ or $xt + 2t = (x+2)t$.

Most of the mathematical software tools transform such an expression into their own internal representation which causes an automatic reordering of subexpressions. They usually do not give possibilities to avoid such automatic simplifications.

- Compute the range of definition, present the solution set and distinct cases if necessary (e.g., when dividing $xt = 5$ by t or multiplying $\frac{1}{x} < 2$ by x).
- With respect to elementary algebra, the system should compute step-by-step, like in:

$$\begin{aligned} 2(x - (1+t)) + x &\stackrel{\text{distribute}}{=} 2(x-1-t) + x \stackrel{\text{distribute}}{=} 2x - 2 - 2t + x \stackrel{\text{collect like terms}}{=} -2 - 2t + 2x + x \\ &\stackrel{\text{simplify}}{=} -2 - 2t + 3x. \end{aligned}$$

The last item, computing step-by-step, is one of the most critical and wide discussed disadvantage especially of CAS or algebraic calculators. They usually do not give proper information to the student about solution steps. Most of all CAS's only offer "black box" methods to solve equations or to perform term manipulation. Let us see, for example, how Derive acts:



Derive solves the equation #1 (given in the left snapshot) in one step using the command "solve", and simplifies the term #1 (see the right snapshot) in one step using the command "simplify". The student does not see which operations the system did apply to get the solution of both computations. (Of course, Derive only serves as an example, other systems like MuPAD, Maple, etc. would act in a similar way.)

Since the system does the cumbersome computations does this mean that students do not need to learn algebraic manipulations by hand? Trying to answer this, one is led to the old issue about the relationship between skills and understanding:[Dr94]

Whether and to what extent are manipulations necessary for conceptual understanding?

No generally accepted answer to this complex issue has been given yet, and none is to be expected in the (even near) future. What I believe is that we should think about decreasing the amount of doing term manipulations in schools but nevertheless, learning of introductory algebra will not become obsolete!

A second problem arises when using such tools for mathematical education:

To whom should the student turn if he or she runs into difficulties when trying to solve a problem?

Especially if such tools are used outside of schools, students are without any guidance and may be alone with their problems and questions. They can easily get lost, waste time in irrelevant activities, make mistakes during their solution attempts, and even develop counterproductive misconceptions.

2. The Computer as a Tutor

One special class of didactically based software tools which try to give answers to the two questions above, are intelligent tutorial system (ITS). They use methods of artificial intelligence to provide "intelligence" to guide students like a private teacher.

What are the requirements of an ITS? Such systems can be divided into four main parts:

1. The *Expert Module*: The system must be an expert on the subject in question. The system must be able to answer student questions, to solve tasks put to the student, and to analyze student answers for bugs and misconceptions.
2. The *Environment Module*: The system must know how to present the subject matter in an appropriate way, and must allow the student to enter her/her problems in an appropriate way.
3. The *Student Module*: The system must have an idea of each student's knowledge and skills and be able to adapt its own hypothetical student model dynamically to the student's learning progress.
4. The *Tutor Module*: The system must have knowledge about the curriculum and offer the student a repertoire of tutorial strategies in order to be able to intervene tutorially in an optimal way at any point.

Here we have the third role for computers in education, which completes the classification given in the introductory part of this article:

- iii) **The Computer as a Tutor**: The computer presents information or questions, the student responds, the computer assesses the response and determines what to present next. The computer tutor can keep records of work by each student being tutored, it can present a wide range of subjects, and it can adapt instructions to needs of individual students.[Fe91]

Such software tools have only a low status besides mathematical tools discussed before, and possible reasons for that might be:[Ho94]

1. The costs for the development of such software tools are extremely high.
2. While research is far advanced in some fields, there is as yet no such tool for teaching in school that meets the high requirements of an *intelligent tutorial system*. Prototypes were designed mainly for arithmetics, algebraic term transformations, equations and equation systems, etc.
3. Teachers may have a general distrust toward tutorial systems because of negative experience with programmed instruction in earlier years, and subsequently with simple and low-yield drill and practice programs for simple skills.

In order to reduce the enormous development cost subclasses of tutorial systems were defined, such as so-called *task-oriented tutorial systems* (TTS). Two essential educational goals of a TTS are:[Ho94]

1. The students know which operators are required or permissible for solving the task (e.g., transformation rules for transforming terms or equations). What is to be exercised here is the skill to apply the operators in the context of a problem solution consisting of several steps.
2. The students should know and be able to apply heuristic methods to solve problems.

A TTS, similar to an ITS, can be divided into four components, three of them are described here. The first component, the *Expert Module*, should

1. find a solution for each problem of the problem class. The solution is appropriate to the knowledge state of the student;
2. be able to check a student solution for correctness and quality. It is able to classify errors as they occur;
3. be "transparent", that means, it uses only knowledge and methods the student is supposed to learn and use.

The *Environment Module* for the dialogue between student and tutor should

1. minimize the number of actions (keystrokes, mouseclicks, etc.) which are necessary for the communication with the system;
2. represent the problems in that way that the representation reflects the structure of the problems;
3. give as much information as possible about the problem-solving process;
4. detect possible mistakes done by the student during the communication (e.g., input of syntactical correct mathematical expressions).

The third component for tutorial aspects, the *Tutor Module*, should

1. monitor each step the student makes toward a solution. For this, the tutor makes use of the expert module;
2. offer help at any stage in the problem-solving process to the student in form of hierarchically graded help. Help begins with general heuristic hints and ends with prescribing the very step toward a solution the expert would have chosen in this situation.

The selection of problems is done by the student or by the tutor selecting it from a prestructured problem collection.

In the next chapter a mathematical software for introductory algebra (high school Algebra I) is described which meets some of the aspects of a problem-oriented tutorial system. The software was developed on the basis of MuPAD.

3. A Tutorial System for Introductory Algebra

SciFace Software and the MuPAD Research Group at the University of Paderborn was hired by Cornelsen Software GmbH⁵, a big german publisher and school software developer, to develop a mathematical software for introductory algebra which meets some of the aspects of a TTS.

The software can deal with arbitrary problems of the following problem classes:

1. Algebraic term manipulation using operations such as "collect same terms", distributive a product, factorize, cancel (sub-)terms or building the common denominator of rational terms.
2. Solving linear equations and inequations, with at most one parameter.
3. Solving linear rational equations and inequalities, with at most one parameter.
4. Solving systems of two linear equations by the addition method, substitution method and the method of equating.
5. Solving systems of three linear equations by the addition method.

The system ask the student to compute the range of definition (if it is a proper subset of \mathbb{Q}), the solution set and to distinct cases if necessary (e.g., when dividing $xt = 5$ by t or multiplying the inequality $\frac{1}{x} < 2$ by x).

3.1 The Components of the Software

3.1.1 The Environment Module

The user interface, developed by the Cornelsen Verlag, consists of a graphical editor for entering mathematical expressions and selecting mathematical operations. The editor provides templates for certain mathematical structures such as fractions, solution sets or systems of equations. Operations can be selected from a menu, whereas some of them can be specified by text input (e.g., "V" for "simplify"; german: Vereinfachen) as well. Input (expressions and operations) made above the current input line can not be manipulated afterwards but can be browsed by using the mouse if the whole computation does not fit into the windows size.

⁵ www.cornelsen.de

3.1.2 The Expert Module

For an arbitrary problem of a certain problem class and at any stage of the solution process the expert module is able to:

- T** — check if a given mathematical operation is correct and a step towards the solution.
- T** — check the result of a solution step for mathematical correctness and compability to the corresponding operation.
- A** — compute the result of a solution step.
- A** — suggest an appropriate mathematical operation as a next solution step.

The answers (**A**) of the expert module are not only mathematically correct but also compatible to the didactics of introductory algebra. The expert module implements problem-solving strategies for each problem class. A solving strategy for linear equations, for instance, consists of the following "productions":

1. Cleaning up the equation by distributing, collecting terms, and simplifying.
2. Moving all variable terms to one side of the equation and moving all constant terms to the other side of the equation.
3. Isolate the variable by multiplying both sides of the equation with the reciprocal of the variables coefficient and simplifying.

Of course, this strategy can be followed by several solutions of varying grain-size. For example solving the equation $2(x+5) = 3x-5$ could lead to solutions such as:

$2(x+5) = 3x-5 \quad :2$ $x+5 = \frac{3}{2}x - \frac{5}{2} \quad -\frac{3}{2}x$ $x - \frac{3}{2}x + 5 = -\frac{5}{2} \quad -5$ $1) \quad x - \frac{3}{2}x = -5 - \frac{5}{2} \quad \text{simplify}$ $-\frac{1}{2}x = -\frac{15}{2} \quad \cdot(-2)$ $x = 15$ $L = \{15\}$	$2(x+5) = 3x-5 \quad \text{distribute}$ $2x+10 = 3x-5 \quad -3x$ $2x-3x+10 = -5 \quad -10$ $2) \quad 2x-3x = -15 \quad \text{simplify}$ $-x = -15 \quad \cdot(-1)$ $x = 15$ $L = \{15\}$	$2(x+5) = 3x-5 \quad :2$ $x+5 = \frac{3}{2}x - \frac{5}{2} \quad -\frac{3}{2}x$ $x - \frac{3}{2}x + 5 = -\frac{5}{2} \quad -5$ $3) \quad x - \frac{3}{2}x = -5 - \frac{5}{2} \quad \text{simplify}$ $-\frac{1}{2}x = -\frac{15}{2} \quad \cdot(-2)$ $x = 15$ $L = \{15\}$
---	--	---

The expert module is implemented in MuPAD. It is important to note that the student selects the operation *and* computes the result of the solution step! In the software tools discussed before the tool computes the result of an operation chosen by the student. The software, i.e. the tutor module as described in the next subsection, just carefully watches the student's steps and only intervenes if the student does a mistake in his or her computations.

3.1.3 The Tutor Module

The tutor module is not implemented in MuPAD itself but as a C++ control by Cornelsen Verlag which calls the MuPAD functions of the expert module. It implements the strategy for solving problems of each supporting problem class (see below). As an example the strategy for solving linear equations is as follows:

1. Check the input of the student whether the expression belongs to the class of linear equations.
2. Ask the expert module if the range of definition is a proper subset of Q. If so, the student has to compute the range of definition.
3. Let the student solve the equation until x is eliminated to one side of the equation whereby the other side of the equation is "mostly simplified". Take care of possible distinctions of cases during the solution process (e.g., if dividing the equation $xt = 5$ by t).
4. The student is asked for the solution set.

The tutor module carefully watches the student's steps and checks the correctness of each computation step. It informs the student, if he or she made a mistake or a step which is not a step towards the solution. It offers help to the student on demand by proposing the next computation step or by computing the result with respect to a given operation. As an example, if the student would ask the system to propose a valid next computation step in order to solve the equation $2(x+5) = 3x-5$, the system would give the answer to distribute the left side of the equation.

It is important that the system allows the student to practice as independently as possible, that means that the system should remain quite and not intervene as long as the student generates correct solution steps (see also [AnLeMi90]).

In the following table a number of valid operations to solve the linear equation $2(x+5) = 3x - 5$. The second column contains some operations which the system would refuse as a student's choice.

$2(x+5) = 3x - 5$		
Valid operations entered by the student	Invalid operations entered by the student	Comment
distribute	simplify	There is nothing to simplify.
:2	:(x+5)	This operation would lead to a distinction of cases which is not necessary here.
+5	+10	Mathematically correct, but not a step forward to a solution.
-3x	:	

Note that a set of valid and invalid operations is neither selected from a predefined working data base nor determined by the system in advance. It is determined dynamically for each step of the solution process and for each new input the student enters.

3.2 Technical Aspects of the Expert Module

The didactical aspects of the software as well as the mathematical requirements were laid down by Cornelsen Verlag. SciFace Software and the MuPAD Research Group developed the expert module based on MuPAD and the communication protocol between the interface module and the expert module.

As a modern general computer algebra system, MuPAD offers powerful functions for solving equations, inequalities as well as system of equations. These functions are necessary, for example, to determine the equivalence of two equations when checking the mathematical correctness of the student's input. Moreover, many functions dealing with rewriting and simplifying mathematical expressions exist. They are necessary, for instance, to determine the equivalence of two expressions.

But the main advantage of MuPAD for implementing the expert module, is the feature of creating new data types (so-called *domains*) and their integration into the MuPAD system. For such a new data type, each operation (like addition or multiplication) and almost each MuPAD function (like for computing the greatest common divisor of two numbers) can be redefined! Doing this, automatic simplifications applied by MuPAD on the system data types can be avoided and controlled. Existing MuPAD functions can be used for the new data types.

New data types for integers, rational numbers, rational numbers in mixed representation, rational expressions, sums and products and some more were implemented. These data types define how to create and print their objects and how to compute with these objects. They consist of functions which, for example, define how to cancel or simplify rational expressions.

Let me introduce a small part of the MuPAD code for the implementation of mixed rational numbers, just to give an impression about the domains concept of MuPAD.


```

MixedRatNum:= domain("MixedRatNum"):

MixedRatNum::new:= proc(n,r)
begin
  if n = 0 then return( r ) else return( new(MixedRatNum,n,r) ) end_if
end_proc:

MixedRatNum::print:= proc(x)
  local n, s;
begin
  if (n:= IntPart(x)) < 0
  then s:= "-"; n:= expr2text(-n)
  else s:= ""; n:= expr2text(n)
  end_if;

  return( s . n . "#" . expr2text(RatPart(x)) )
end_proc:

MixedRatNum::_plus:= proc(x,y)
begin
  if args(0) <> 2 then error( "invalid use of MuPAD function" )
  elif domtype(y) <> MixedRatNum or x <> -y
  then return(Sum::new( x,y,Flatten ))
  else return( IntNum::zero )
  end_if
end_proc:

...

```

It would be out of the scope of this article to go into details (and possibly not of the reader's interest). What should become clear is that the mathematical part of the software, which is based on MuPAD, is programmed in the programming language of MuPAD. The MuPAD engine, i.e., the MuPAD kernel written in C++, had not to be changed or extended.

3.3 Open Technical and Didactical Questions

Some aspects to satisfy the requirements of a TTS are still missing, e.g., a hierarchically graded help. Another missing aspect of the software is a matching for *buggy rules* [AnLeMi90], e.g., when the student tries to add two fractions like $\frac{a}{b} + \frac{c}{d} = \frac{a+b}{c+d}$. Such errors will cause the system to interrupt and to respond with a generic error message, error matches are not implemented. Anyway, it should not be underestimate that this is a requirement of high complexity!

The design of the system raises didactical and technical questions, a few are presented here:

- When the student chooses an operation the system checks whether the computation step is correct and a "step toward" the solution. Solution steps which for a moment complicate the problem are not allowed. Why is the system restrictive in such situations? Could not the student learn from those solution steps? What should the system report in such situations?
- When solving equations or system of equations the goal of the problem is quite clear: Eliminate x and simplify the solution as much as possible. But what does "simplifying" mean? Is the term $2(t+3)$ more simplified than the equivalent term $2t+6$? It is difficult for the programmer to deal with different *standard forms* of terms because they do not allow to define the end of a computation. As a consequence the expert module defines the *standard form* of an expression. But does this definition correspond with the idea of the student? Do we need different *standard forms* for different problems?
- Consider the equation $2(x+2)=2-3(x-2)$. When the student choose the operation "distribute" which side of the equation is meant? The left or the right one? As the user interface does not offer the possibility to select subterms of an expression the system must check for both cases. Moreover, if in that case the student asks the system to compute the result of the operation "distribute", the system can not know to which side of the equation the student wanted to apply the operation and therefore must choose one of him or apply the operation to both sides simultaneously.

3.4 Summary

A mathematical software for introductory algebra (high school Algebra I) was developed which meets some of the aspects of a problem-oriented tutorial system.

It was specially designed to meet the didactical needs of introductory algebra and should guide the student through the solving process. The system gives help when problems arise, and this for arbitrary problems given by the teacher or the student itself. Hence, the software can be seen as highly flexible and interactive.

The use of a computer algebra system to develop such software tools significantly decreases the cost of the development. Nevertheless, one should not underestimate that they still are very high. Experiences of using this software will show where problems and weaknesses still exist. This will help us to improve the development of mathematical software tools which are designed to meet the needs of the student and teacher, and not to force them to meet the needs of the software.

Let me quote J.R. Anderson et al in [AnLeMi90] about their ideal for the design of such tutorial software systems which reflects my opinion as well: "Our ideal for the design [...] is the one-on-one human tutor. Compared with a human tutor, the computer tutor falls short on many accounts. The human tutor has access to the expressive power of natural language and is better able to understand and interpret the behaviour of the student. However, we hope to have captured an essential advantage that a private tutor can offer to a student: The [Software] allows the student to learn mathematical skills through guided practice that is individualized to that student's strengths and weaknesses."

4. Software Components for Mathematical Documents

In the last chapter of this article I will introduce one of the main recent developments within the MuPAD project.

4.1 Graphical User Interfaces for Mathematical Electronical Documents

Presentation tools (like Microsoft PowerPoint), internet browsers or professional authoring tools like Macromedia's Authorware or Director are presently used for writing educational mathematical documents. However, material written with such a software is restricted with respect to the mathematical presentation. Components for presenting mathematical expressions in an adequate quality (like TeX gives) or plotting of two- or three-dimensional graphics are still missing. When mathematical documents should get more interactivity and dynamic by allowing the reader to change parameters of a mathematical computation or by changing plotting parameters (zoom in and out, rotate a graphical object, etc.), for example, the author has to use software which usually can not be embedded. The reader works in an environment which is completely independent of the document, and the author does not have any access to the reader's activities.

The MuPAD Research Group together with SciFace Software is developing software components which can be used as plugins for these presentation tools based on the ActiveX technology under Windows 9x/NT [Ar97]. Each of these Controls defines a set of interface functions which can be used from outside to interact with the Control. For example, a Control for visualization may offer a function for drawing a graph of a given function, or functions for rotating or zooming a graph.

The author of a Web page, for example, uses JavaScript to access the interface functions and to implement the communication between several Controls. This enables an author of interactive mathematical documents to use the presentation software of his choice and to employ software components for sophisticated mathematical tasks.

In the following the two main Controls which are currently under development are introduced.

4.1.1 A Calculator Control for Mathematical Computations

The Calculator Control for mathematical computations consists of the following components:

1. An Editor for writing mathematical expressions (MathML compatible).
2. Communication with MuPAD.
3. A set of mathematical operations which can be applied to expressions or selected (sub-)expressions.
4. Text representing user information, comments, warnings or error messages given by the mathematical engine (MuPAD).

It is important to note that the Control itself does not provide interface components like buttons or menus. They are defined in the surrounding application (the internet browser, the authoring tool, etc.). This enables the author to specify the operations to be made available to the reader for each instance of the Control. The interface of the Control offers functions for extracting the values of certain computation steps which allows the author to access this computation on following pages of his document, to check the correctness of the computation or to call the Graphics Control (see below) in order to plot a graph of the expression, for instance. Hence, a document is no longer static but highly flexible and dynamic.

Figure 2 shows a snapshot of a (quite primitive) Visual Basic testapplication in which the Control is embedded. All buttons are defined by few lines of Visual Basic code. Clicking on a button executes a Visual Basic routine which calls an interface function of the Control. With the buttons at the bottom of the application the Control is instructed which command has to be applied next. Pressing the "Ausführen" button (german: Execute) will send the operation to MuPAD which performs the operation and sends the result back to the Control.

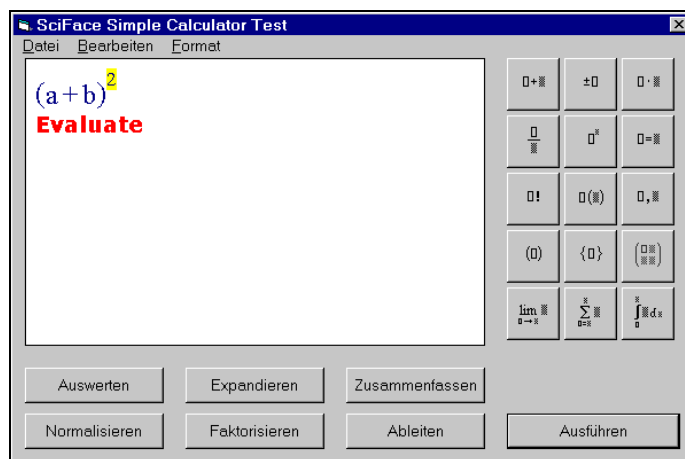


Figure 2: The Control embedded in a Visual Basic testapplication⁶

Figure 3 shows another instance of the Control now embedded into a running Microsoft PowerPoint slideshow.

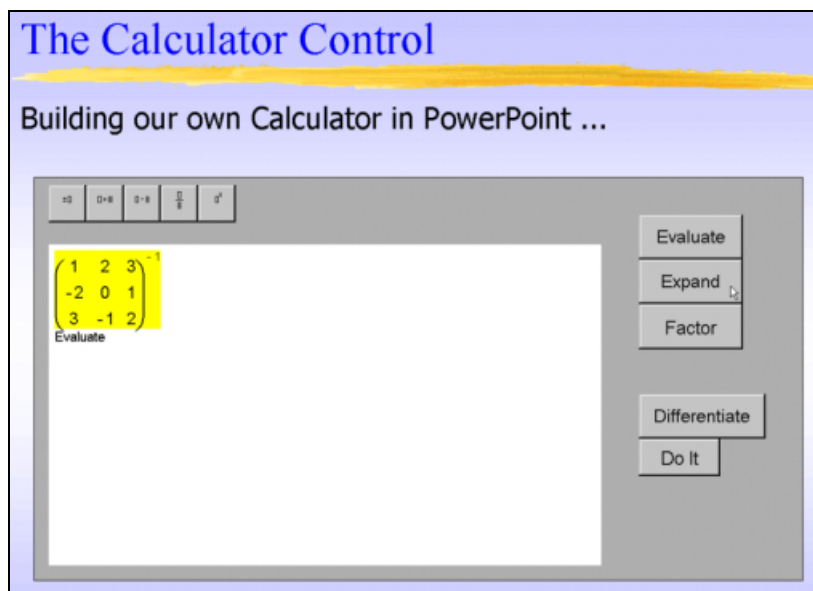


Figure 3: The Control now embedded into a running Microsoft PowerPoint slideshow

This demonstrates how easy it is to extend documents with mathematical power and flexibility. The following table is the source code of the implementation of the five templates of the editor ("negate", "sum", "product", "fraction" and "power") and the five operations ("Evaluate", "Expand", "Factor", "Differentiate" and "Do It") on the slide of Figure 3:

⁶ The operations are described in german; anyway, as the author commit the names of the operations, english names may be used as well.

<pre> Private Sub CommandButton1_Click() Calc1.SetOperation ("_eval") Calc1.Execute End Sub Private Sub CommandButton2_Click() Calc1.SetOperation ("_expand") Calc1.Execute End Sub Private Sub CommandButton3_Click() Calc1.SetOperation ("_factor") Calc1.Execute End Sub Private Sub CommandButton4_Click() Calc1.SetOperation ("_diff") End Sub Private Sub CommandButton6_Click() Calc1.Execute End Sub </pre>	<pre> Private Sub CommandButton7_Click() Calc1.InsertTemplate ("_negate") End Sub Private Sub CommandButton8_Click() Calc1.InsertTemplate ("_plus") End Sub Private Sub CommandButton9_Click() Calc1.InsertTemplate ("_mult") End Sub Private Sub CommandButton10_Click() Calc1.InsertTemplate ("_divide") End Sub Private Sub CommandButton11_Click() Calc1.InsertTemplate ("_power") End Sub </pre>
---	---

Each operation is defined by a command template. It consists of a MuPAD library function which performs the MuPAD command, a name by which it is accessed from the Control, the main argument and possibly further arguments, and a command string which is displayed in the Control after applying this command. The reader does not see the corresponding MuPAD functions behind a certain operation (unless the author prefers it). For easy use we will provide the author with a rich set of pre-defined command templates.

The MuPAD functions can be organized in special MuPAD libraries offering functions such as step-by-step computations, for checking the readers results or for giving information about computational steps. As an example we may use the MuPAD library which serves as the expert module of the tutoring software for elementary algebra introduced in chapter three of this article. Using the Calculator Control would solve some of the problems mentioned above, e.g., it would allow the student to select subterms in order to specify on which terms the chosen operation should be applied.

This Control is still in an early development stage. Only one expression is visible in the Control at a time. The final version will be more attractive as all computation steps of a calculation including the applied commands and extra arguments will be visible. The results of operations will be organized in a right sided tree reflecting the structure of the computation. Following a branch of this tree each node is a valid consequence of the operations being applied before. If an input (e.g., a mathematical expression or a statement for a definition of a variable) is changed, subsequent results will be marked as (possibly) invalid as long as they are re-evaluated in the new context.

4.1.2 A Flexible and Interactive Graphics Control

The Graphics Control for visualizing and manipulating three-dimensional objects offers graphical primitives for points, vectors, plains, spheres, cones, graphs of functions and so on. These objects can be collected in graphical scenes, they can be selected and manipulated, rotated and zoomed. The reader can ask for the 3D-coordinates of a point, the distance between two points, change the coordinate system and so on.

Figure 4 shows a snapshot of a (quite primitive) Visual Basic testapplication in which the Control is embedded. The user asked for the 3D-coordinates of a certain point on the yellow plane.

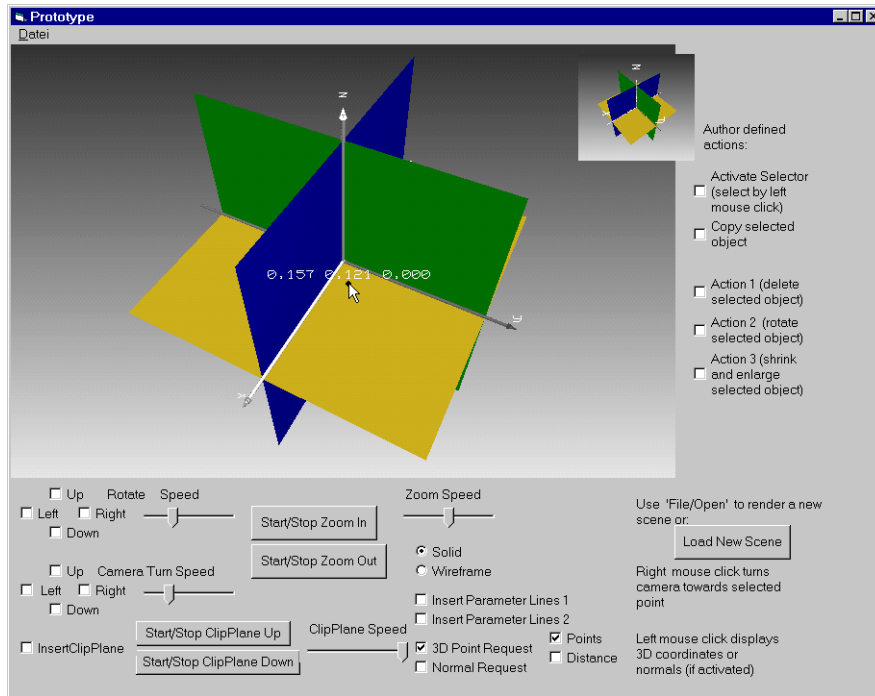


Figure 4: The graphics Control embedded in a Visual Basic testapplication:
Ask for 3D-coordinates

Figure 5 shows the graphic scene after selecting and rotating the green plane. Note that this operation and the ones on the right-hand side of the testapplication are implemented by the author in few lines of Visual Basic code calling the interface functions of the Control.

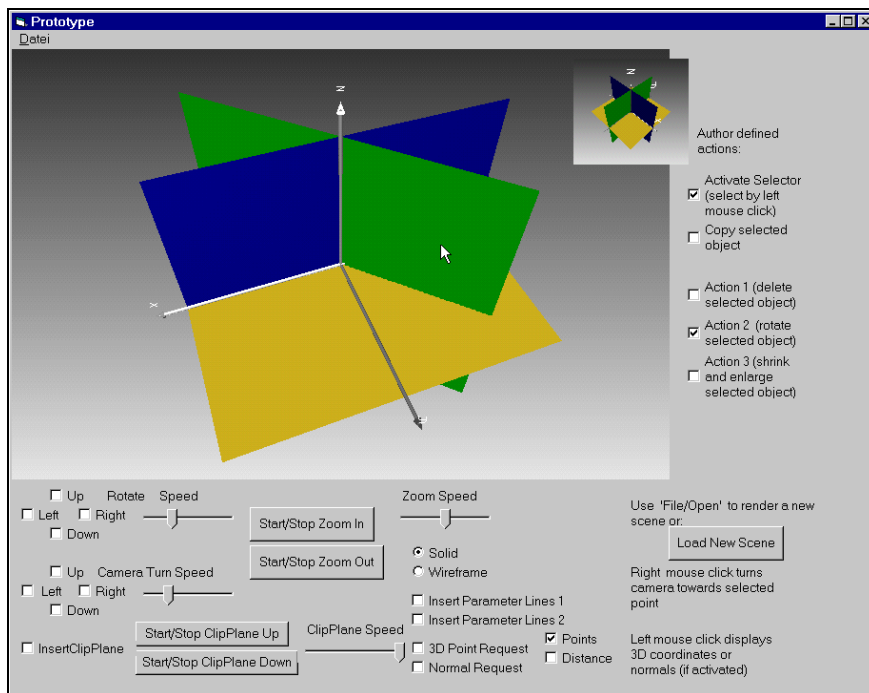


Figure 5: Rotate a selected object

The same Control can be embedded into the Microsoft Internet Explorer, a Microsoft PowerPoint slideshow or any other software which allows to include ActiveX controls.

Similar to the Calculator Control, the Graphics Control does not provide interface components like buttons or menus. They are defined in the application in which the Control is embedded (e.g., an internet browser or authoring tool). The interface of the Control additionally provides basic routines which allows the author to

implement his or her own graphical operations. Again, to ease use we will provide the author with a rich set of predefined graphical operations.

The Control also offers interface functions for the communication with the Calculator Control (see previous subsection) and the MuPAD kernel. This enables the author to map geometrical operations to their corresponding algebraic representation using the computer algebra system. For example, the operation "rotate an object around the axes" can be mapped to a matrix representing the geometrical operation.

4.2 Summary

We develop software components which can be used as plugins in a wide variety of "container software", e.g., internet browser like the Internet Explorer, Microsoft Word, presentation tools like Microsoft PowerPoint and professional authoring tools like Macromedia's Authorware or Director. This enables an author of interactive mathematical electronic documents to use the presentation software of his choice and to employ software components for sophisticated mathematical tasks.

The Controls define interfaces which enable the author to adapt *each instance* of such a Control to his needs, using a programming language like Java Script, C++, Visual Basic or the scripting language of an authoring tool.

Default palettes, default instances of Controls and special MuPAD libraries will be provided to ease the author's burden in using and defining instances of these Controls.

The use of these Controls together with a connection to a powerful computer algebra system will provide the author to write highly flexible and interactive mathematical documents.

5. References

- [AnBoRe85] Anderson, J.R.; Boyle, C.F.; Reiser, B.J.:
Intelligent Tutoring Systems
In: Science, Volume 228, pp. 456-462. 1985.
- [AnLeMi90] Anderson, J.R.; Milson, R.; Lewis, M.W.:
The Teacher's Apprentice Project: Building an Algebra Tutor
In: R. Freedle (Ed.): Artificial Intelligence and the future of Testing. Lawrence Erlbaum Associates, Publisher. Hillsdale, New Jersey Hove and London 1990.
- [Ar97] Armstrong, T.:
Designing and Using ActiveX Controls
M & T Books, New York, 1997.
- [Dr94] Dreyfus, T.:
The Role of Cognitive Tools in Mathematics Education
In: R. Biehler, R.W. Scholz, R. Sträßer, B. Winkelmann (Eds.): Didactics of Mathematics as a Scientific Discipline, pp. 201-211. Kluwer Academic Publishers 1994.
- [Fe91] Fey, J.:
Computers in US Mathematics Education: Recent Research and Development Results
In: Schriftenreihe Didaktik der Mathematik Band 21, pp. 77-100. Verlag Hölder-Pichler-Tempsky, Wien 1991. Verlag B.G. Teubner, Stuttgart 1991.

- [Ho94] Holland, G.:
Intelligent Tutorial Systems
In: R. Biehler, R.W. Scholz, R. Sträßer, B. Winkelmann (Eds.): *Didactics of Mathematics as a Scientific Discipline*, pp. 213-223. Kluwer Academic Publishers 1994.
- [Ho91] Holland, G.:
Schülermodellierung bei aufgabenorientierten Intelligenten Tutoriellen Systemen
In: *Schriftenreihe Didaktik der Mathematik Band 21*, pp. 127-134. Verlag Hölder-Pichler-Tempsky, Wien 1991. Verlag B.G. Teubner, Stuttgart 1991.
- [Ku99] Kutzler, B.:
The Algebraic Calculator as a Pedagogical Tool for Teaching Mathematics
Online version: www.kutzler.com/bk/a-pt/ped-tool.html, 1999.
- [OePoRüWe99] Oevel, W.; Postel, F.; Rüscher, G.; Wehmeier, S.:
Das MuPAD Tutorium
SciFace Software GmbH&Co.KG, Paderborn 1999.
- [Ta94] Tall, D.:
Computer Environments for the Learning of Mathematics
In: R. Biehler, R.W. Scholz, R. Sträßer, B. Winkelmann (Eds.): *Didactics of Mathematics as a Scientific Discipline*, pp. 189-199. Kluwer Academic Publishers 1994.