The calculator and calculus packages*
Scientific calculations with LaTeX

Robert Fuster
Universitat Politècnica de València
rfuster@mat.upv.es

2012/06/10

Abstract

The calculator package allows us to use LaTeX as a calculator, with which we can perform many of the common scientific calculations (with the limitation in accuracy imposed by the TeX arithmetic).

This package introduces several new instructions that allow you to do several calculations with integer and decimal numbers using LaTeX. Apart from add, multiply or divide, we can calculate powers, square roots, logarithms, trigonometric and hyperbolic functions ...

In addition, the calculator package supports some elementary calculations with vectors in two and three dimensions and square 2×2 and 3×3 matrices.

The calculus package adds to the calculator package several utilities to use and define various functions and their derivatives, including elementary functions, operations with functions, polar coordinates and vector-valued real functions.

Contents

*This document corresponds to calculator and calculus 1.0a, dated 2012/06/10.

1 Introduction

The calculator package defines some instructions which allow us to realize algebraic operations (and to evaluate elementary functions) in our documents. The operations implemented by the calculator package include routines of assignment of variables, arithmetical calculations with real and integer numbers, two and three dimensional vector and matrix arithmetics and the computation of square roots, trigonometrical, exponential, logarithmic and hyperbolic functions. In addition, some important numbers, such as $\sqrt{2}$, π or e, are predefined.

The name of all these commands is spelled in capital letters (with very few exceptions: the commands `\DEGtoRAD` and `\RADtoDEG` and the control sequences that define special numbers, as `\numberPI`) and, in general, they all need one or more mandatory arguments, the first one(s) of which is(are) number(s) and the last one(s) is(are) the name(s) of the command(s) where the results will be stored.[1] The new commands defined in this way work in any LaTeX mode.

By example, this instruction

```
\MAX{3}{5}{\solution}
```

stores `5` in the command `\solution`. In a similar way,

```
\FRACTIONSIMPLIFY{10}{12}{\numerator}{\denominator}
```

defines `\numerator` and `\denominator` as `5` i `6`, respectively.

The *data* arguments should not be necessarily explicit numbers; it may also consist in commands the value of which is a number. This allows us to chain several calculations, since in the following example:

[1] Logically, the control sequences that represent special numbers (as `\numberPI`) does not need any argument.

Ex. 1

$$\frac{2.5^2}{\sqrt{12}}+\mathrm{e}^{3.4} = \frac{6.25}{3.4641}+29.96432$$
$$= 1.80421+29.96432$$
$$= 31.76854$$

```
%  \tempA=2,5^2
  \SQUARE{2.5}{\tempA}
%  \tempB=sqrt(12)
  \SQUAREROOT{12}{\tempB}
% \tempC=exp(3,4)
  \EXP{3.4}{\tempC}
%  \divisio=\tempA/tempB
  \DIVIDE{\tempA}{\tempB}{\divisio}
% \sol=\divisio+\tempC
  \ADD{\divisio}{\tempC}{\sol}
\begin{align*}
\frac{2.5^2}{\sqrt{12}}+\mathrm{e}^{3.4}
      &= \frac{\tempA}{\tempB}+\tempC \\
      &= \divisio+\tempC \\
      &=\sol
\end{align*}
```

Observe that, in this example, we have followed exactly the same steps that we would do to calculate $\frac{2.5^2}{\sqrt{12}}+\mathrm{e}^{3.4}$ with a standard calculator: We would calculate the square, the root and the exponential and, finally, we would divide and add the results.

It does not matter if the arguments *results* are or not predefined. But these commands act as declarations, so that its scope is local in environments and groups.

Ex. 2

The `\sol` command contains the square of 5:

$$5^2 = 25$$

Now, the `\sol` command is the square root of 5:

$$\sqrt{5} = 2.23605$$

On having gone out of the `center` environment, the command recovers its previous value: 25

```
\SQUARE{5}\sol
The \texttt{\textbackslash sol}
command contains the square of $5$:
\[5^2=\sol\]
\begin{center}
\SQUAREROOT{5}\sol
Now, the \texttt{\textbackslash sol}
command is the square root of $5$:
\[\sqrt{5}=\sol\]
\end{center}
On having gone out of the \texttt{center}
environment,
the command recovers its previous value:
\sol
```

The calculus package goes a step further and allows us to define and use in a user-friendly manner various functions and their derivatives.

For exemple, using the calculus package, you can define the $f(t) = t^2e^t - \cos 2t$ function as follows:

```
%     \PRODUCTfunction{\SQUAREfunction}{\EXPfunction}{\tempfunctionA}
%     \SCALEVARIABLEfunction{2}{\COSfunction}{\tempfunctionB}
%     \SUBTRACTfunction{\tempfunctionA}{\tempfunctionB}{\Ffunction}
```

Then you cau compute any value of the new function `\Ffunction` and its derivative: typing

`\Ffunction{`⟨*num*⟩`}{`⟨`\sol`⟩`}{`⟨`\Dsol`⟩`}`

the values of $f(num)$ and $f'(num)$ will be stored in `\sol` and `\Dsol`.

File I
The calculator package

2 Predefined numbers

The calculator package predefines the following numbers:

`\numberPI`	$3.14159 \approx \pi$	`\numberHALFPI`	$1.57079 \approx \pi/2$
`\numberTHREEHALFPI`	$4.71237 \approx 3\pi/2$	`\numberTHIRDPI`	$1.0472 \approx \pi/3$
`\numberQUARTERPI`	$0.78539 \approx \pi/4$	`\numberFIFTHPI`	$0.62831 \approx \pi/5$
`\numberSIXTHPI`	$0.52359 \approx \pi/6$	`\numberTWOPI`	$6.28317 \approx 2\pi$
`\numberE`	$2.71828 \approx \mathrm{e}$	`\numberINVE`	$0.36787 \approx 1/\mathrm{e}$
`\numberETWO`	$7.38902 \approx \mathrm{e}^2$	`\numberINVETWO`	$0.13533 \approx 1/\mathrm{e}^2$
`\numberLOGTEN`	$2.30258 \approx \log 10$		
`\numberGOLD`	$1.61803 \approx \phi$	`\numberINVGOLD`	$0.61803 \approx 1/\phi$
`\numberSQRTTWO`	$1.41421 \approx \sqrt{2}$	`\numberSQRTTHREE`	$1.73205 \approx \sqrt{3}$
`\numberSQRTFIVE`	$2.23607 \approx \sqrt{5}$		
`\numberCOSXXX`	$0.86603 \approx \cos\pi/6$	`\numberCOSXLV`	$0.70711 \approx \cos\pi/4$

3 Operations with numbers

3.1 Assignments and comparisons

The first command we describe here is used to store a number in a control sequence. The other two commands in this section determine the maximum and minimum of a pair of numbers.

`\COPY{`⟨***num***⟩`}{`⟨`\cmd`⟩`}` stores the number *num* to the command `\cmd`.

Ex. 3

```
\COPY{-1.256}{\sol}
\sol
```

-1.256

`\MAX{`⟨***num1***⟩`}{`⟨***num2***⟩`}{`⟨`\cmd`⟩`}` stores in `\cmd` the maximum of the numbers *num1* and *num2*.

Ex. 4

```
\MAX{1.256}{3.214}{\sol}
\[\max(1.256,3.214)=\sol\]
```

$$\max(1.256, 3.214) = 3.214$$

\MIN{⟨*num1*⟩}{⟨*num2*⟩}{⟨\cmd⟩} stores in \cmd the minimum of *num1* and *num2*.

Ex. 5

```
\MIN{1.256}{3.214}{\sol}
\sol
```

1.256

3.2 Real arithmetic

3.2.1 The four basic operations

The following commands calculate the four arithmetical basic operations.

\ADD{⟨*num1*⟩}{⟨*num2*⟩}{⟨\cmd⟩} Sum of numbers *num1* and *num2*.

Ex. 6

```
\ADD{1.256}{3.214}{\sol}
$1.256+3.214=\sol$
```

$1.256 + 3.214 = 4.47$

\SUBTRACT{⟨*num1*⟩}{⟨*num2*⟩}{⟨\cmd⟩} Difference *num1*-*num2*.

Ex. 7

```
\SUBTRACT{1.256}{3.214}{\sol}
$1.256-3.214=\sol$
```

$1.256 - 3.214 = -1.95801$

\MULTIPLY{⟨*num1*⟩}{⟨*num2*⟩}{⟨\cmd⟩} Product *num1* ×*num2*.

Ex. 8

```
\MULTIPLY{1.256}{3.214}{\sol}
$1.256\times3.214=\sol$
```

$1.256 \times 3.214 = 4.03677$

\DIVIDE{⟨*num1*⟩}{⟨*num2*⟩}{⟨\cmd⟩} Quotient *num1* /*num2*.[2]

Ex. 9

```
\DIVIDE{1.256}{3.214}{\sol}
$1.256/3.214=\sol$
```

$1.256/3.214 = 0.39078$

In addition, the \LENGTHDIVIDE command divides two lengths and returns a number.

\LENGTHDIVIDE{⟨*length1*⟩}{⟨*length2*⟩}{⟨\cmd⟩}

Ex. 10

```
\LENGTHDIVIDE{1in}{1cm}{\sol}
One inch equals $\sol$ centimeters.
```

One inch equals 2.54 centimeters.

[2] This command uses a modified version of the division algorithm of Claudio Beccari.

3.2.2 Powers with integer exponent

`\SQUARE{⟨num⟩}{⟨\cmd⟩}` Square of the number *num*.

Ex. 11

$(-1.256)^2 = 1.57751$

```
\SQUARE{-1.256}{\sol}
$(-1.256)^2=\sol$
```

`\CUBE{⟨num⟩}{⟨\cmd⟩}` Cube of *num*.

Ex. 12

$(-1.256)^3 = -1.98134$

```
\CUBE{-1.256}{\sol}
$(-1.256)^3=\sol$
```

`\POWER{⟨num⟩}{⟨exp⟩}{⟨\cmd⟩}` The *exp* power of *num*.

The exponent, *exp*, must be an integer (if you want to calculate powers with non integer exponents, use the `\EXP` command).

Ex. 13

$$\begin{aligned}(-1.256)^{-5}&=-0.31989\\(-1.256)^{5}&=-3.1256\\(-1.256)^{0}&=1\end{aligned}$$

```
\POWER{-1.256}{-5}{\sola}
\POWER{-1.256}{5}{\solb}
\POWER{-1.256}{0}{\solc}
\[
\begin{aligned}
(-1.256)^{-5}&=\sola
\\
(-1.256)^{5}&=\solb
\\
(-1.256)^{0}&=\solc
\end{aligned}
\]
```

3.2.3 Absolute value, integer part and fractional part

`\ABSVALUE{⟨num⟩}{⟨\cmd⟩}` Absolute value of *num*.

Ex. 14

$\left\vert -1.256\right\vert = 1.256$

```
\ABSVALUE{-1.256}{\sol}
$\left\vert-1.256\right\vert=\sol$
```

`\INTEGERPART{⟨num⟩}{⟨\cmd⟩}` Integer part of *num*.[3]

Ex. 15

The integer part of 1.256 is 1, but the integer part of -1.256 is -2.

```
\INTEGERPART{1.256}{\sola}
\INTEGERPART{-1.256}{\solb}
The integer part of  $1.256$ is $\sola$,
but the integer part of $-1.256$ is $\solb$.
```

[3] The integer part of x is the largest integer that is less than or equal to x.

\FLOOR is an alias of \INTEGERPART.

Ex. 16

```
\FLOOR{1.256}{\sol}
The integer part of $1.256$ is $\sol$.
```

The integer part of 1.256 is 1.

\FRACTIONALPART{⟨*num*⟩}{⟨*cmd*⟩} Fractional part of *num*.

Ex. 17

```
\FRACTIONALPART{1.256}{\sol}
\sol

\FRACTIONALPART{-1.256}{\sol}
\sol
```

0.256
0.744

3.2.4 Truncation and rounding

\TRUNCATE[⟨*n*⟩]{⟨*num*⟩}{⟨*cmd*⟩} truncates the number *num* to *n* decimal places.

\ROUND[*n*]{⟨*num*⟩}{⟨*cmd*⟩} rounds the number *num* to *n* decimal places.

The optional argument *n* may be 0, 1, 2, 3 or 4 (the default is 2).[4]

Ex. 18

```
\TRUNCATE[0]{1.25688}{\sol}
\sol

\TRUNCATE[2]{1.25688}{\sol}
\sol

\TRUNCATE[4]{1.25688}{\sol}
\sol
```

1
1.25
1.2568

Ex. 19

```
\ROUND[0]{1.25688}{\sol}
\sol

\ROUND[2]{1.25688}{\sol}
\sol

\ROUND[4]{1.25688}{\sol}
\sol
```

1
1.26
1.2569

3.3 Integers

The operations described here are subject to the same restrictions as those referring to decimal numbers. In particular, although TEX does not have this restriction in its integer arithmetic, the largest integer that can be used is 16383.

[4]Note than \TRUNCATE[0] is equivalent to \INTEGERPART only for non-negative numbers.

3.3.1 Integer division, quotient and remainder

`\INTEGERDIVISION{⟨num1⟩}{⟨num2⟩}{⟨\cmd1⟩}{⟨\cmd2⟩}` stores in the *\cmd1* and *\cmd2* commands the quotient and the remainder of the integer division of the two integers *num1* and *num2*. The remainder is a non-negative number smaller than the divisor.[5]

Ex. 20

$435 = 27 \times 16 + 3$
$27 = 435 \times 0 + 27$
$-435 = 27 \times (-17) + 24$
$435 = -27 \times (-16) + 3$
$-435 = -27 \times 17 + 24$

```
\INTEGERDIVISION{435}{27}{\sola}{\solb}
$435=27\times\sola+\solb$

\INTEGERDIVISION{27}{435}{\sola}{\solb}
$27=435\times\sola+\solb$

\INTEGERDIVISION{-435}{27}{\sola}{\solb}
$-435=27\times(\sola)+\solb$

\INTEGERDIVISION{435}{-27}{\sola}{\solb}
$435=-27\times(\sola)+\solb$

\INTEGERDIVISION{-435}{-27}{\sola}{\solb}
$-435=-27\times\sola+\solb$
```

`\INTEGERQUOTIENT{⟨num1⟩}{⟨num2⟩}{⟨\cmd⟩}` Integer part of the quotient of *num1* and *num2*. These two numbers are not necessarily integers.

Ex. 21

16
0
-17

```
\INTEGERQUOTIENT{435}{27}{\sol}
\sol

\INTEGERQUOTIENT{27}{435}{\sol}
\sol

\INTEGERQUOTIENT{-43.5}{2.7}{\sol}
\sol
```

`\MODULO{⟨num1⟩}{⟨num2⟩}{⟨\cmd⟩}` Remainder of the integer division of *num1* and *num2*.

Ex. 22

$$435 \equiv 3 \pmod{27}$$
$$-435 \equiv 24 \pmod{27}$$

```
\MODULO{435}{27}{\sol}
\[
435 \equiv \sol \pmod{27}
\]
\MODULO{-435}{27}{\sol}
\[
-435 \equiv \sol \pmod{27}
\]
```

[5] The scientific computing systems (such as Matlab. Scilab or Mathematica) do not always return a non-negative residue —especially when the divisor is negative—. However, the most reasonable definition of integer quotient is this one: *the quotient of the division D/d is the largest number q for which $dq \leq D$*. With this definition, the remainder $r = D - qd$ is a non-negative number.

3.3.2 Greatest common divisor and least common multiple

`\GCD{⟨num1⟩}{⟨num2⟩}{⟨\cmd⟩}` Greatest common divisor of the integers *num1* and *num2*.

Ex. 23

$\gcd(435, 27) = 3$

```
\GCD{435}{27}{\sol}
$\gcd(435,27)=\sol$
```

`\LCM{⟨num1⟩}{⟨num2⟩}{⟨\cmd⟩}` Least common multiple of *num1* and *num2*.

Ex. 24

$\operatorname{lcm}(435, 27) = 3915$

```
\newcommand{\lcm}{\operatorname{lcm}}
\LCM{435}{27}{\sol}
$\lcm(435,27)=\sol$
```

3.3.3 Simplifying fractions

`\FRACTIONSIMPLIFY{⟨num1⟩}{⟨num2⟩}{⟨\cmd1⟩} {⟨\cmd2⟩}` stores in the *\cmd1* and *\cmd2* commands the numerator and denominator of the irreducible fraction equivalent to *num1* / *num2*.

Ex. 25

$435/27 = 145/9$

```
\FRACTIONSIMPLIFY{435}{27}{\sola}{\solb}
$435/27=\sola/\solb$
```

3.4 Elementary functions

3.4.1 Square roots

`\SQUAREROOT {⟨num⟩}{⟨\cmd⟩}` Square root of the number *num*.

Ex. 26

$\sqrt{1.44} = 1.2$

```
\SQUAREROOT{1.44}{\sol}
$\sqrt{1.44}=\sol$
```

If the argument *num* is negative, the package returns a warning message.

Instead of `\SQUAREROOT`, you can use the alias `\SQRT`.

3.4.2 Exponential and logarithm

The `\EXP` and `\LOG` commands compute, by default, exponentials and logarithms of the natural base e. They admit, however, an optional argument to choose another base.

\EXP {⟨*num*⟩}{⟨\cmd⟩} Exponential of the number *num*.

Ex. 27

```
\EXP{0.5}{\sol}
$\exp(0.5)=\sol$
```

$\exp(0.5) = 1.64871$

The argument *num* must be in the interval $[-9.704, 9.704]$. [6]

Moreover, the \EXP command accepts an optional argument, to compute expressions such as a^x:

\EXP [⟨*num1*⟩]{⟨*num2*⟩}{⟨\cmd⟩} Exponential with base *num1* of *num2*. *num1* must be a positive number.

Ex. 28

```
\EXP[10]{1.3}{\sol}
$10^{1.3}=\sol$

\EXP[2]{0.33333}{\sol}
$2^{1/3}=\sol$
```

$10^{1.3} = 19.95209$
$2^{1/3} = 1.25989$

\LOG {⟨*num*⟩}{⟨\cmd⟩} logarithm of the number *num*.

Ex. 29

```
\LOG{0.5}{\sol}
$\log 0.5=\sol$
```

$\log 0.5 = -0.69315$

\LOG [⟨*num1*⟩]{⟨*num2*⟩}{⟨\cmd⟩} Logarithm in base *num1* of *num2*.

Ex. 30

```
\LOG[10]{0.5}{\sol}
$\log_{10} 0.5=\sol$
```

$\log_{10} 0.5 = -0.30103$

3.4.3 Trigonometric functions

The arguments, in functions \SIN, \COS, ..., are measured in radians. If you measure angles in degrees (sexagesimal or not), use the \DEGREESSIN, \DEGREESCOS, ... commands.

\SIN {⟨*num*⟩}{⟨\cmd⟩} Sine of *num*.

\COS {⟨*num*⟩}{⟨\cmd⟩} Cosine of *num*.

\TAN {⟨*num*⟩}{⟨\cmd⟩} Tangent of *num*.

[6] 9.704 is the logarithm of 16383, the largest number that supports the TEX's arithmetic.

\COT {⟨*num*⟩}{⟨\cmd⟩} Cotangent of *num*.

Ex. 31

$\sin\pi/3 = 0.86601$
$\cos\pi/3 = 0.5$
$\tan\pi/3 = 1.73201$
$\cot\pi/3 = 0.57736$

```
\SIN{\numberTHIRDPI}{\sol}
$\sin \pi/3=\sol$

\COS{\numberTHIRDPI}{\sol}
$\cos \pi/3=\sol$

\TAN{\numberTHIRDPI}{\sol}
$\tan \pi/3=\sol$

\COT{\numberTHIRDPI}{\sol}
$\cot \pi/3=\sol$
```

\DEGREESSIN {⟨*num*⟩}{⟨\cmd⟩} Sine of *num* sexagesimal degrees.

\DEGREESCOS {⟨*num*⟩}{⟨\cmd⟩} Cosine of *num* sexagesimal degrees.

\DEGREESTAN {⟨*num*⟩}{⟨\cmd⟩} Tangent of *num* sexagesimal degrees.

\DEGREESCOT {⟨*num*⟩}{⟨\cmd⟩} Cotangent of *num* sexagesimal degrees.

Ex. 32

$\sin 60^{\textrm o} = 0.86601$
$\cos 60^{\textrm o} = 0.49998$
$\tan 60^{\textrm o} = 1.73201$
$\cot 60^{\textrm o} = 0.57736$

```
\DEGREESSIN{60}{\sol}
$\sin 60^{\textrm o}=\sol$

\DEGREESCOS{60}{\sol}
$\cos 60^{\textrm o}=\sol$

\DEGREESTAN{60}{\sol}
$\tan 60^{\textrm o}=\sol$

\DEGREESCOT{60}{\sol}
$\cot 60^{\textrm o}=\sol$
```

The latter commands support an optional argument that allows us to divide the circle in an arbitrary number of *degrees* (not necessarily 360).

\DEGREESSIN [⟨*degrees*⟩]{⟨*num*⟩}{⟨\cmd⟩}

\DEGREESCOS [⟨*degrees*⟩]{⟨*num*⟩}{⟨\cmd⟩}

\DEGREESTAN [⟨*degrees*⟩]{⟨*num*⟩}{⟨\cmd⟩}

\DEGREESCOT [⟨*degrees*⟩]{⟨*num*⟩}{⟨\cmd⟩}

By example, `\DEGREESCOS[400]{50}` computes the cosine of 50 gradians (a right angle has 100 gradians, the whole circle has 400 gradians), which are equivalent to 45 (sexagesimal) degrees or $\pi/4$ radians. Or to 1 *degree*, if we divide the circle into 8 parts!

Ex. 33

0.70709
0.70709
0.7071
0.70709

```
\DEGREESCOS[400]{50}{\sol}
\sol

\DEGREESCOS{45}{\sol}
\sol

\COS{\numberQUARTERPI}{\sol}
\sol

\DEGREESCOS[8]{1}{\sol}
\sol
```

Moreover, we have a couple od commands to convert between radians and degrees,

`\DEGtoRAD` {⟨***num***⟩}{⟨`\`*cmd*⟩} Equivalence in radians of *num* sexagesimal degrees.

`\RADtoDEG` {⟨***num***⟩}{⟨`\`*cmd*⟩} Equivalence in sexagesimal degrees of *num* radians.

Ex. 34

1.0472

```
\DEGtoRAD{60}{\sol}
\sol
```

and two other commands to reduce arguments to basic intervals:

`\REDUCERADIANSANGLE` {⟨***num***⟩}{⟨`\`*cmd*⟩} Reduces the arc *num* to the interval $]-\pi, \pi]$.

`\REDUCEDEGREESANGLE` {⟨***num***⟩}{⟨`\`*cmd*⟩} Reduces the angle *num* to the interval $]-180, 180]$.

Ex. 35

3.14159
90

```
\MULTIPLY{\numberTWOPI}{10}{\TWENTYPI}
\ADD{\numberPI}{\TWENTYPI}{\TWENTYONEPI}
\REDUCERADIANSANGLE{\TWENTYONEPI}{\sol}
\sol

\REDUCEDEGREESANGLE{3690}{\sol}
\sol
```

3.4.4 Hyperbolic functions

`\SINH` {⟨***num***⟩}{⟨`\`*cmd*⟩} stores in `\`*cmd* the hyperbolic sine of *num*.

`\COSH` {⟨***num***⟩}{⟨`\`*cmd*⟩} Hyperbolic cosine of *num*.

`\TANH` {⟨***num***⟩}{⟨`\`*cmd*⟩} Hyperbolic tangent of *num*.

`\COTH` {⟨***num***⟩}{⟨`\`*cmd*⟩} Hyperbolic cotangent of *num*.

Ex. 36

1.61328
1.89807
0.84995
1.17651

```
\SINH{1.256}{\sol}
\sol

\COSH{1.256}{\sol}
\sol

\TANH{1.256}{\sol}
\sol

\COTH{1.256}{\sol}
\sol
```

4 Matrix arithmetic

The calculator package defines the commands described below to operate on vectors and matrices. We only work with two or three-dimensional vectors and 2×2 and 3×3 matrices. Vectors are represented in the form `(a1,a2)` or `(a1,a2,a3);`[7] and, in the case of matrices, columns are separated *à la matlab* by semicolons: `(a11,a12;a21,a22)` or `(a11,a12,a13;a21,a22,a23;a31,a32,a33)`.

4.1 Vector operations

4.1.1 Assignments

`\VECTORCOPY(`⟨***x,y***⟩`)(`⟨***\cmd1,\cmd2***⟩`)` copy the entries of vector (⟨*x,y*⟩) to the *cmd1* and *cmd2* commands.

`\VECTORCOPY(`⟨***x,y,z***⟩`)(`⟨***\cmd1,\cmd2,\cmd3***⟩`)` copy the entries of vector (***x***,***y***,***z***) to the *cmd1*, *cmd2* and *cmd3* commands.

Ex. 37

$(1, -1)$
$(1, -1, 2)$

```
\VECTORCOPY(1,-1)(\sola,\solb)
$(\sola,\solb)$

\VECTORCOPY(1,-1,2)(\sola,\solb,\solc)
$(\sola,\solb,\solc)$
```

4.1.2 Vector addition and subtraction

`\VECTORADD(`⟨***x1,y1***⟩`)(`⟨***x2,y2***⟩`)(`⟨***\cmd1,\cmd2***⟩`)`

`\VECTORADD(`⟨***x1,y1,z1***⟩`)(`⟨***x2,y2,z2***⟩`) (`⟨***\cmd1,\cmd2,\cmd3***⟩`)`

`\VECTORSUB(`⟨***x1,y1***⟩`)(`⟨***x2,y2***⟩`)(`⟨***\cmd1,\cmd2***⟩`)`

[7] But they are *column* vectors.

\VECTORSUB(⟨*x1,y1,z1*⟩)(⟨*x2,y2,z2*⟩) (⟨*\cmd1,\cmd2,\cmd3*⟩)

Ex. 38

$(1, -1, 2) + (3, 5, -1) = (4, 4, 1)$
$(1, -1, 2) - (3, 5, -1) = (-2, -6, 3)$

```
\VECTORADD(1,-1,2)(3,5,-1)(\sola,\solb,\solc)
$(1,-1,2)+(3,5,-1)=(\sola,\solb,\solc)$

\VECTORSUB(1,-1,2)(3,5,-1)(\sola,\solb,\solc)
$(1,-1,2)-(3,5,-1)=(\sola,\solb,\solc)$
```

4.1.3 Scalar-vector product

\SCALARVECTORPRODUCT{⟨*num*⟩}(⟨*x,y*⟩) (⟨*\cmd1,\cmd2*⟩)

\SCALARVECTORPRODUCT{⟨*num*⟩}(⟨*x,y,z*⟩) (⟨*\cmd1,\cmd2,\cmd3*⟩)

Ex. 39

$2(3, 5) = (6, 10)$
$2(3, 5, -1) = (6, 10, -2)$

```
\SCALARVECTORPRODUCT{2}(3,5)(\sola,\solb)
$2(3,5)=(\sola,\solb)$

\SCALARVECTORPRODUCT{2}(3,5,-1)(%
                              \sola,\solb,\solc)
$2(3,5,-1)=(\sola,\solb,\solc)$
```

4.1.4 Scalar product and euclidean norm

\SCALARPRODUCT(⟨*x1,y1*⟩)(⟨*x2,y2*⟩){⟨*\cmd*⟩}

\SCALARPRODUCT(⟨*x1,y1,z1*⟩)(⟨*x2,y2,z2*⟩){⟨*\cmd*⟩}

\VECTORNORM(⟨*x,y*⟩){⟨*\cmd*⟩}

\VECTORNORM(⟨*x,y,z*⟩){⟨*\cmd*⟩}

Ex. 40

$(1, -1) \cdot (3, 5) = -2$
$(1, -1, 2) \cdot (3, 5, -1) = -4$
$\|(3, 4)\| = 5$
$\|(1, 2, -2)\| = 3$

```
\SCALARPRODUCT(1,-1)(3,5){\sol}
$(1,-1)\cdot(3,5)=\sol$

\SCALARPRODUCT(1,-1,2)(3,5,-1){\sol}
$(1,-1,2)\cdot(3,5,-1)=\sol$

\VECTORNORM(3,4)\sol
$\left\|(3,4)\right\|=\sol$

\VECTORNORM(1,2,-2)\sol
$\left\|(1,2,-2)\right\|=\sol$
```

4.1.5 Unit vector parallel to a given vector (normalized vector)

\UNITVECTOR(⟨***x,y***⟩)(⟨***cmd1***,***cmd2***⟩)

\UNITVECTOR(⟨***x,y,z***⟩)(⟨***cmd1***,***cmd2***,***cmd3***⟩)

Ex. 41

$(0.59999, 0.79999)$
$(0.33333, 0.66666, -0.66666)$

```
\UNITVECTOR(3,4)(\sola,\solb)
$(\sola,\solb)$

\UNITVECTOR(1,2,-2)(\sola,\solb,\solc)
$(\sola,\solb,\solc)$
```

4.1.6 Absolute value (in each entry of a given vector)

\VECTORABSVALUE(⟨***x,y***⟩)(⟨***cmd1***,***cmd2***⟩)

\VECTORABSVALUE(⟨***x,y,z***⟩)(⟨***cmd1***,***cmd2***,***cmd3***⟩)

Ex. 42

$(3, 4)$
$(3, 4, 1)$

```
\VECTORABSVALUE(3,-4)(\sola,\solb)
$(\sola,\solb)$

\VECTORABSVALUE(3,-4,-1)(\sola,\solb,\solc)
$(\sola,\solb,\solc)$
```

4.2 Matrix operations

4.2.1 Assignments

\MATRIXCOPY (⟨***a11,a12;a21,a22***⟩) (⟨***cmd11***,***cmd12***;***cmd21***,***cmd22***⟩)

Use this command to store the matrix $\begin{bmatrix} a11 & a12 \\ a21 & 22 \end{bmatrix}$ in *cmm11*, *cmm12*, *cmm21*, *cmm22*. The analogous 3×3 version is

\MATRIXCOPY (⟨***a11,a12,a13;*** [...] ***,a33***⟩) (⟨***cmd11***,***cmd12***,***cmd13***; [...] ,***cmd33***⟩)

Ex. 43

$$\begin{bmatrix} 1 & -1 & 2 \\ 3 & 0 & 5 \\ -1 & 1 & 4 \end{bmatrix}$$

```
\MATRIXCOPY(1, -1, 2;
             3,  0, 5;
            -1,  1, 4)%
           (\sola,\solb,\solc;
            \sold,\sole,\solf;
            \solg,\solh,\soli)
$\begin{bmatrix}
       \sola & \solb & \solc \\
       \sold & \sole & \solf \\
       \solg & \solh & \soli
     \end{bmatrix}$
```

Henceforth, we will present only the syntax for commands operating with 2×2 matrices. In all cases, the syntax is similar if we work with 3×3 matrices. In the examples, we will work with either 2×2 or 3×3 matrices.

4.2.2 Transposed matrix

\TRANSPOSEMATRIX ($\langle$***a11,a12;a21,a22***$\rangle$) ($\langle$*\cmd11,\cmd12;\cmd21,\cmd22*$\rangle$)

Ex. 44

$$\begin{bmatrix} 1 & -1 \\ 3 & 0 \end{bmatrix}^T = \begin{bmatrix} 1 & 3 \\ -1 & 0 \end{bmatrix}$$

```
\TRANSPOSEMATRIX(1,-1;3,0)%
          (\sola,\solb;\solc,\sold)
$\begin{bmatrix}
       1 & -1 \\ 3 & 0
     \end{bmatrix}^T=\begin{bmatrix}
       \sola & \solb \\ \solc & \sold
     \end{bmatrix}$
```

4.2.3 Matrix addition and subtraction

\MATRIXADD ($\langle$***a11,a12;a21,a22***$\rangle$) ($\langle$***b11,b12;b21,b22***$\rangle$) ($\langle$*\cmd11,\cmd12;\cmd21,\cmd22*$\rangle$)

\MATRIXSUB ($\langle$***a11,a12;a21,a22***$\rangle$) ($\langle$***b11,b12;b21,b22***$\rangle$) ($\langle$*\cmd11,\cmd12;\cmd21,\cmd22*$\rangle$)

Ex. 45

$$\begin{bmatrix} 1 & -1 \\ 3 & 0 \end{bmatrix} + \begin{bmatrix} 3 & 5 \\ -3 & 2 \end{bmatrix} = \begin{bmatrix} 4 & 4 \\ 0 & 2 \end{bmatrix}$$
$$\begin{bmatrix} 1 & -1 \\ 3 & 0 \end{bmatrix} - \begin{bmatrix} 3 & 5 \\ -3 & 2 \end{bmatrix} = \begin{bmatrix} -2 & -6 \\ 6 & -2 \end{bmatrix}$$

```
\MATRIXADD(1,-1;3,0)(3,5;-3,2)%
          (\sola,\solb;\solc,\sold)
$\begin{bmatrix}
       1 & -1 \\ 3 & 0
     \end{bmatrix}+
\begin{bmatrix}
       3 & 5 \\ -3 & 2
     \end{bmatrix}=\begin{bmatrix}
       \sola & \solb \\ \solc & \sold
     \end{bmatrix}$

\MATRIXSUB(1,-1;3,0)(3,5;-3,2)%
          (\sola,\solb;\solc,\sold)
$\begin{bmatrix}
       1 & -1 \\ 3 & 0
     \end{bmatrix}-
\begin{bmatrix}
       3 & 5 \\ -3 & 2
     \end{bmatrix}=\begin{bmatrix}
       \sola & \solb \\ \solc & \sold
     \end{bmatrix}$
```

4.2.4 Scalar-matrix product

\SCALARMATRIXPRODUCT{$\langle$***num***$\rangle$} ($\langle$***a11,a12;a21,a22***$\rangle$) ($\langle$*\cmd11,\cmd12;\cmd21,\cmd22*$\rangle$)

Ex. 46

$$3\begin{bmatrix} 1 & -1 & 2 \\ 3 & 0 & 5 \\ -1 & 1 & 4 \end{bmatrix} = \begin{bmatrix} 3 & -3 & 6 \\ 9 & 0 & 15 \\ -3 & 3 & 12 \end{bmatrix}$$

```
\SCALARMATRIXPRODUCT{3}(1,-1,2;
                        3, 0,5;
                       -1, 1,4)%
                      (\sola,\solb,\solc;
                       \sold,\sole,\solf;
                       \solg,\solh,\soli)
$3\begin{bmatrix}
     1 & -1 & 2 \\ 3 & 0 & 5 \\ -1 & 1 & 4
 \end{bmatrix}
  =\begin{bmatrix}
       \sola & \solb & \solc \\
       \sold & \sole & \solf \\
       \solg & \solh & \soli
   \end{bmatrix}$
```

4.2.5 Matriu-vector product

\MATRIXVECTORPRODUCT (⟨***a11,a12;a21,a22***⟩)(⟨***x,y***⟩) (⟨*\cmd1,\cmd2*⟩)

Ex. 47

$$\begin{bmatrix} 1 & -1 \\ 0 & 2 \end{bmatrix} \begin{bmatrix} 3 \\ 5 \end{bmatrix} = \begin{bmatrix} -2 \\ 10 \end{bmatrix}$$

```
\MATRIXVECTORPRODUCT(1,-1;
                     0, 2)(3,5)(\sola,\solb)
$\begin{bmatrix}
     1 & -1 \\ 0 & 2
 \end{bmatrix}
 \begin{bmatrix}
      3 \\ 5
 \end{bmatrix}
  =\begin{bmatrix}
         \sola \\ \solb
   \end{bmatrix}$
```

4.2.6 Product of two square matrices

\MATRIXPRODUCT (⟨***a11,a12;a21,a22***⟩) (⟨***b11,b12;b21,b22***⟩) (⟨*\cmd11,\cmd12;\cmd21,\cmd22*⟩)

Ex. 48

$$\begin{bmatrix} 1 & -1 & 2 \\ 3 & 0 & 5 \\ -1 & 1 & 4 \end{bmatrix} \begin{bmatrix} 3 & 5 & -1 \\ -3 & 2 & -5 \\ 1 & -2 & 3 \end{bmatrix} = \begin{bmatrix} 8 & -1 & 10 \\ 14 & 5 & 12 \\ -2 & -11 & 8 \end{bmatrix}$$

```
\MATRIXPRODUCT(1,-1,2;3,0,5;-1,1,4)%
              (3,5,-1;-3,2,-5;1,-2,3)%
                (\sola,\solb,\solc;
                 \sold,\sole,\solf;
                 \solg,\solh,\soli)
\begin{multline*}
\begin{bmatrix}
       1 & -1 & 2 \\ 3 & 0 & 5 \\ -1 & 1 & 4
     \end{bmatrix}
\begin{bmatrix}
       3 & 5 & -1 \\ -3 & 2 & -5 \\ 1 & -2 & 3
     \end{bmatrix}\\
=\begin{bmatrix}
       \sola & \solb & \solc \\
       \sold & \sole & \solf \\
       \solg & \solh & \soli
\end{bmatrix}
\end{multline*}
```

4.2.7 Determinant

\DETERMINANT (⟨***a11,a12;a21,a22***⟩) {⟨*cmd*⟩}

Ex. 49

$$\begin{vmatrix} 1 & -1 & 2 \\ 3 & 0 & 5 \\ -1 & 1 & 4 \end{vmatrix} = 18$$

```
\DETERMINANT(1,-1,2;3,0,5;-1,1,4){\sol}
\SpecialUsageIndex{\DETERMINANT}%
$\begin{vmatrix}
       1 & -1 & 2 \\ 3 & 0 & 5 \\ -1 & 1 & 4
     \end{vmatrix}=\sol$
```

4.2.8 Inverse matrix

\INVERSEMATRIX (⟨***a11,a12;a21,a22***⟩) (⟨*cmd11*,*cmd12*;*cmd21*,*cmd22*⟩)

Ex. 50

$$\begin{bmatrix} 1 & -1 \\ 3 & 5 \end{bmatrix}^{-1} = \begin{bmatrix} 0.625 & 0.125 \\ -0.375 & 0.125 \end{bmatrix}$$

```
\INVERSEMATRIX(1,-1;3,5)(%
            \sola,\solb;\solc,\sold)
$\begin{bmatrix}
       1 & -1 \\ 3 & 5
 \end{bmatrix}^{-1}=
 \begin{bmatrix}
       \sola & \solb \\ \solc & \sold
 \end{bmatrix}$
```

If the given matrix is singular, the calculator package returns a warning message and the *cmd11*, ..., commands are marqued as undefined.

4.2.9 Absolute value (in each entry)

`\MATRIXABSVALUE` (⟨***a11,a12;a21,a22***⟩) (⟨*\cmd11,\cmd12;\cmd21,\cmd22*⟩)

Ex. 51

$$\begin{bmatrix} 1 & 1 & 2 \\ 3 & 0 & 5 \\ 1 & 1 & 4 \end{bmatrix}$$

```
\MATRIXABSVALUE(1,-1,2;3,0,5;-1,1,4)%
          (\sola,\solb,\solc;
           \sold,\sole,\solf;
           \solg,\solh,\soli)
$\begin{bmatrix}
       \sola & \solb & \solc \\
       \sold & \sole & \solf \\
       \solg & \solh & \soli
 \end{bmatrix}$
```

4.2.10 Solving a linear system

`\SOLVELINEARSYSTEM` (⟨***a11,a12;a21,a22***⟩)(⟨***b1,b2***⟩)(⟨*\cmd1,\cmd2*⟩) solves the linear system $\begin{pmatrix} a11 & a12 \\ a21 & a22 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} b1 \\ b2 \end{pmatrix}$ and stores the solution in (*\cmd1*, *\cmd2*).

Ex. 52

Solving the linear system

$$\begin{bmatrix} 1 & -1 & 2 \\ 3 & 0 & 5 \\ -1 & 1 & 4 \end{bmatrix} \mathsf{X} = \begin{bmatrix} -4 \\ 4 \\ -2 \end{bmatrix}$$

we obtain $\mathsf{X} = \begin{bmatrix} 3 \\ 5 \\ -1 \end{bmatrix}$

```
\SOLVELINEARSYSTEM(1,-1,2;3,0,5;-1,1,4)%
                  (-4,4,-2)%
                  (\sola,\solb,\solc)
Solving the linear system
\[
\begin{bmatrix}
      1 & -1 & 2 \\ 3 & 0 & 5 \\ -1 & 1 & 4
     \end{bmatrix}\mathsf{X}=\begin{bmatrix}
      -4\\4\\-2
     \end{bmatrix}
\]
we obtain
$\mathsf{X}=\begin{bmatrix}
       \sola \\ \solb\\ \solc
     \end{bmatrix}$
```

If the given matrix is singular, the package calculator returns a warning message. When system is indeterminate, in the bi-dimensional case one of the solutions is computed; if the system is incompatible, then the `\sola`, ..., commands are marqued as undefined. For three equations systems, only determinate systems are solved.[8]

[8] This is the only command that does not behave the same way with 2×2 and 3×3 matrices.

File II
The **calculus** package

5 What is a *function*?

From the point of view of this package, a *function* f is a pair of formulae: the first one calculates $f(t)$; the other, $f'(t)$. Therefore, any function is applied using three arguments: the value of the variable t, and two command names where $f(t)$ and $f'(t)$ will be stored. For example,

`\SQUAREfunction{`⟨*num*⟩`}{`⟨*sol*⟩`}{`⟨*Dsol*⟩`}`

computes $f(t) = t^2$ and $f'(t) = 2t$ (where $t =$*num*), and stores the results in the commands *sol* and *Dsol*.[9]

Ex. 53

If $f(t) = t^2$, then

$$f(3) = 9 \text{ and } f'(3) = 6$$

```
\SQUAREfunction{3}{\sol}{\Dsol}
If $f(t)=t^2$, then
\[
  f(3)=\sol \mbox{ and } f'(3)=\Dsol
\]
```

For all functions defined here, you must use the following syntax:

functionname`{`⟨*num*⟩`}{`⟨*cmd1*⟩`}{`⟨*cmd2*⟩`}`

being *num* a number (or a command whose value is a number), and *cmd1* and *cmd2* two control sequence names where the values of the function and its derivative (in this number) will be stored.

The key difference between this *functions* and the instructions defined in the calculator package is the inclusion of the derivative; for example, the `\SQUARE{3}{\sol}` instruction computes, only, the square power of number 3, while `\SQUAREfunction{3}{\sol}{\Dsol}` finds, also, the corresponding derivative.

6 Predefined functions

The calculus package predefines the most commonly used elementary functions, and includes several utilities for defining new ones. The predefined functions are the following:

[9]Do not spect any control about the existence or differentiability of the function; if the function or the derivative are not well defined, a TeX error will occur.

`\ZEROfunction`	$f(t) = 0$	`\ONEfunction`	$f(t) = 1$
`\IDENTITYfunction`	$f(t) = t$	`\RECIPROCALfunction`	$f(t) = 1/t$
`\SQUAREfunction`	$f(t) = t^2$	`\CUBEfunction`	$f(t) = t^3$
`\SQRTfunction`	$f(t) = \sqrt{t}$		
`\EXPfunction`	$f(t) = \exp t$	`\LOGfunction`	$f(t) = \log t$
`\COSfunction`	$f(t) = \cos t$	`\SINfunction`	$f(t) = \sin t$
`\TANfunction`	$f(t) = \tan t$	`\COTfunction`	$f(t) = \cot t$
`\COSHfunction`	$f(t) = \cosh t$	`\SINHfunction`	$f(t) = \sinh t$
`\TANHfunction`	$f(t) = \tanh t$	`\COTHfunction`	$f(t) = \coth t$
`\HEAVISIDEfunction`	$f(t) = \begin{cases} 0 & \text{si } t < 0 \\ 1 & \text{si } t \geq 0 \end{cases}$		

In the following example, we use the `\LOGfunction` function to compute a table of the log function and its derivative.

Ex. 54

x	$\log x$	$\log' x$
1	0	1
2	0.69315	0.5
3	1.0986	0.33333
4	1.38629	0.25
5	1.60942	0.2
6	1.79176	0.16666

```
$\begin{array}{cll}
x & \log x & \log' x \\
\LOGfunction{1}{\logx}{\Dlogx}
1 &\logx & \Dlogx\\
\LOGfunction{2}{\logx}{\Dlogx}
2 &\logx & \Dlogx\\
\LOGfunction{3}{\logx}{\Dlogx}
3 &\logx & \Dlogx\\
\LOGfunction{4}{\logx}{\Dlogx}
4 &\logx & \Dlogx\\
\LOGfunction{5}{\logx}{\Dlogx}
5 &\logx & \Dlogx\\
\LOGfunction{6}{\logx}{\Dlogx}
6 &\logx & \Dlogx
\end{array}$
```

7 Operations with functions

We can define new functions using the following *operations* (the last argument is the name of the new function):

`\CONSTANTfunction{`⟨*num*⟩`}{`⟨*\Function*⟩`}` defines *\Function* as the constant function *num*.

Example. Definition of the $F(t) = 5$ function:

`\CONSTANTfunction{5}{\F}`

`\SUMfunction{`⟨*\function1*⟩`}{`⟨*\function2*⟩`} {`⟨*\Function*⟩`}` defines *\Function* as the sum of functions *\function1* and *\function2*.

Example. Definition of the $F(t) = t^2 + t^3$ function:

`\SUMfunction{\SQUAREfunction}{\CUBEfunction}{\F}`

`\SUBTRACTfunction{`⟨*\function1*⟩`}{`⟨*\function2*⟩`} {`⟨*\Function*⟩`}` defines *\Function* as the difference of functions *\function1* and *\function2*.

Example. Definition of the $F(t) = t^2 - t^3$ function:

`\SUBTRACTfunction{\SQUAREfunction}{\CUBEfunction}{\F}`

`\PRODUCTfunction{`⟨*function1*⟩`}{`⟨*function2*⟩`} {`⟨*Function*⟩`}` defines *Function* as the product of functions *function1* and *function2*

Example. Definition of the $F(t) = \mathrm{e}^t \cos t$ function:

`\PRODUCTfunction{\EXPfunction}{\COSfunction}{\F}`

`\QUOTIENTfunction{`⟨*function1*⟩`}{`⟨*function2*⟩`} {`⟨*Function*⟩`}` defines *Function* as the quotient of functions *function1* and *function2*.

Example. Definition of the $F(t) = \mathrm{e}^t / \cos t$ function:

`\QUOTIENTfunction{\EXPfunction}{\COSfunction}{\F}`

`\COMPOSITIONfunction{`⟨*function1*⟩`}{`⟨*function2*⟩`} {`⟨*Function*⟩`}` defines *Function* as the composition of functions *function1* and *function2*.

Example. Definition of the $F(t) = \mathrm{e}^{\cos t}$ function:

`\COMPOSITIONfunction{\EXPfunction}{\COSfunction}{\F}`

(note than `\COMPOSITIONfunction{f}{g}{\F}` means `\F=` $f \circ g$).

`\SCALEfunction{`⟨*num*⟩`}{`⟨*function*⟩`}{`⟨*Function*⟩`}` defines *Function* as the product of number *num* and function *function*.

Example. Definition of the $F(t) = 3\cos t$ function:

`\SCALEfunction{3}{\COSfunction}{\F}`

`\SCALEVARIABLEfunction{`⟨*num*⟩`}{`⟨*function*⟩`} {`⟨*Function*⟩`}` scales the variable by factor *num* and then applies the function *function*.

Example. Definition of the $F(t) = \cos 3t$ function:

`\SCALEVARIABLEfunction{3}{\COSfunction}{\F}`

`\POWERfunction{`⟨*function*⟩`}{`⟨*num*⟩`}{`⟨*Function*⟩`}` defines *Function* as the power of function *function* to the exponent *num* (a positive integer). Example. Definition of the $F(t) = t^5$ function:

`\POWERfunction{\IDENTITYfunction}{5}{\F}`

`\LINEARCOMBINATIONfunction{`⟨*num1*⟩`}{`⟨*function1*⟩`} {`⟨*num2*⟩`}{`⟨*function2*⟩`}{`⟨*Function*⟩`}` defines *Function* as the linear combination of functions *function1* and *function2* multiplied, respectively, by numbers *num1* and *num2*.

Example. Definition of the $F(t) = 2t - 3\cos t$ function:

`\LINEARCOMBINATIONfunction{2}{\IDENTITYfunction}{-3}{\COSfunction}{\F}`

By combining properly this operations and the predefined functions, many elementary functions can be defined.

Ex. 55

If
$$f(t)=3t^2-2\mathrm{e}^{-t}\cos t$$
then
$$\begin{gathered} f(5)=74.99619\\ f'(5)=29.99084 \end{gathered}$$

```
% exp(-t)
   \SCALEVARIABLEfunction
       {-1}{\EXPfunction}
       {\NEGEXPfunction}

% exp(-t)cos(t)
   \PRODUCTfunction
       {\NEGEXPfunction}
       {\COSfunction}
       {\NEGEXPCOSfunction}

% 3t^2-2exp(-t)cos(t)
   \LINEARCOMBINATIONfunction
       {3}{\SQUAREfunction}
      {-2}{\NEGEXPCOSfunction}
      {\myfunction}

\myfunction{5}{\sol}{\Dsol}

If
\[
    f(t)=3t^2-2\mathrm{e}^{-t}\cos t
\]
then
\[
\begin{gathered}
    f(5)=\sol\\
    f'(5)=\Dsol
\end{gathered}
\]
```

8 Polynomial functions

Although polynomial functions can be defined using linear combinations of power functions, to facilitate our work, the calculus package includes the following commands to define more easily the polynomials of 1, 2, and 3 degrees: `\newlpoly` (new *linear* polynomial), `\newqpoly` (new *quadratic* polynomial), and `\newcpoly` (new *cubic* polynomial):

`\newlpoly{`⟨*\Function*⟩`}{`⟨*a*⟩`}{`⟨*b*⟩`}` stores the $p(t) = a + b\,t$ function in the `\Function` command.

`\newqpoly{`⟨*\Function*⟩`} {`⟨*a*⟩`}{`⟨*b*⟩`}{`⟨*c*⟩`}` stores the $p(t) = a + b\,t + c\,t^2$ function in the `\Function` command.

`\newcpoly{`⟨*\Function*⟩`}{`⟨*a*⟩`}{`⟨*b*⟩`}{`⟨*c*⟩`}{`⟨*d*⟩`}` stores the $p(t) = a + b\,t + c\,t^2 + d\,t^3$ function in the `\Function` command.

Ex. 56

$p'(2) = 8$

```
% \mypoly=1-x^2+x^3
    \newcpoly{\mypoly}{1}{0}{-1}{1}
    \mypoly{2}{\sol}{\Dsol}
$p'(2)=\Dsol$
```

These declarations behave similarly to to the declaration `\newcommand`: If the name you want to assign to the new function is that of an already defined command, the calculus package returns an error message and do not redefines this command. To obtain any alternative behavior, our package includes three other sets of declarations:

`\renewlpoly`, `\renewqpoly`, `\renewcpoly` redefine the already existing command `\Function`. If this command does not exist, then it is not defined and an error message occurs.

`\ensurelpoly`, `\ensureqpoly`, `\ensurecpoly` define a new function. If the command `\Function` already exists, it is not redefined.

`\forcelpoly`, `\forceqpoly`, `\forcecpoly` define a new function. If the command `\Function` already exists, it is redefined.

9 Vector-valued functions (or parametrically defined curves)

The instruction

`\PARAMETRICfunction{`⟨*\Xfunction*⟩`}{`⟨*\Yfunction*⟩`} {`⟨*\myvectorfunction*⟩`}`

defines the new vector-valued function $f(t) = (x(t), y(t))$.

The first and second arguments are a pair of functions already defined and, the third, the name of the new function we define. Once we have defined them, the new vector functions requires five arguments:

`\myvectorfunction{`⟨*num*⟩`}{`⟨*\cmd1*⟩`} {`⟨*\cmd2*⟩`}{`⟨*\cmd3*⟩`}{`⟨*\cmd4*⟩`}`

where

- *num* is a number t,
- *\cmd1* and *\cmd2* are two command names where the values of the $x(t)$ function and its derivative $x'(t)$ will be stored, and
- *\cmd3* and *\cmd4* will store $y(t)$ and $y'(t)$.

In short, in this context, a vector function is a pair of scalar functions.

Instead of `\PARAMETRICfunction` we can use the alias `\VECTORfunction`.

Ex. 57

For the $f(t) = (t^2, t^3)$ function we have

$$f(4) = (16, 64), f'(4) = (8, 48)$$

```
For the $f(t)=(t^2,t^3)$ function we have
\VECTORfunction
    {\SQUAREfunction}{\CUBEfunction}{\F}

\F{4}{\solx}{\Dsolx}{\soly}{\Dsoly}

\[
f(4)=(\solx,\soly), f'(4)=(\Dsolx,\Dsoly)
\]
```

10 Vector-valued functions in polar coordinates

The following instruction:

\POLARfunction{$\langle$*\rfunction*$\rangle$}{$\langle$*\Polarfunction*$\rangle$}

declares the vector function $f(\phi) = (r(\phi)\cos\phi, r(\phi)\sin\phi)$. The first argument is the $r = r(\phi)$ function, (an already defined function). For example, we can define the *Archimedean spiral* $r(\phi) = 0{,}5\phi$, as follows:

```
\SCALEfunction{0.5}{\IDENTITYfunction}{\rfunction}
\POLARfunction{\rfunction}{\archimedes}
```

11 Low-level instructions

Probably, many users of the package will not be interested in the implementation of the commands this package includes. If this is your case, you can ignore this section.

11.1 The \newfunction declaration and its variants

All the functions predefined by this package use the \newfunction declaration. This control sequence works as follows:

\newfunction{$\langle$*\Function*$\rangle$}{$\langle$*Instructions to compute* \y *and* \Dy *from* \t$\rangle$}

where the second argument is the list of the instructions you need to run to calculate the value of the function \y and the derivative \Dy in the \t point.

For example, if you want to define the $f(t) = t^2 + \mathrm{e}^t \cos t$ function, whose derivative is $f'(t) = 2t + \mathrm{e}^t(\cos t - \sin t)$, using the high-level instructions we defined earlier, you can write the following instructions:

```
\PRODUCTfunction{\EXPfunction}{\COSfunction}{\ffunction}
\SUMfunction{\SQUAREfunction}{\ffunction}{\Ffunction}
```

But you can also define this function using the \newfunction command as follows:

```
\newfunction{\Ffunction}{%
   \SQUARE{\t}{\tempA}              % A=t^2
   \EXP{\t}{\tempB}                 % B=e^t
   \COS{\t}{\tempC}                 % C=cos(t)
   \SIN{\t}{\tempD}                 % D=sin(t)
   \MULTIPLY{2}{\t}{\tempE}         % E=2t
   \MULTIPLY{\tempB}{\tempC}{\tempC} % C=e^t cos(t)
   \MULTIPLY{\tempB}{\tempD}{\tempD} % D=e^t sin(t)
   \ADD{\tempA}{\tempC}{\y}          % y=t^2 + e^t cos(t)
   \ADD{\tempE}{\tempC}{\tempC}     % C=t^2 + e^t cos(t)
   \SUBTRACT{\tempC}{\tempD}{\Dy}    % y'=t^2 + e^t cos(t) - e^t sin(t)
}
```

It must be said, however, that the `\newfunction` declaration behaves similarly to `\newcommand` or `\newlpoly`: If the name you want to assign to the new function is that of an already defined command, the calculus package returns an error message and does not redefines this command. To obtain any alternative behavior, our package includes three other versions of the `\newfunction` declarations: the `\renewfunction`, `\ensurefunction` and `\forcefunction` declarations. Each of these declarations behaves differently:

`\newfunction` defines a new function. If the command `\`*`Function`* already exists, it is not redefined and an error message occurs.

`\renewfunction` redefines the already existing command `\`*`Function`*. If this command does not exists, then it is not defined and an error message occurs.

`\ensurefunction` defines a new function. If the command `\`*`Function`* already exists, it is not redefined.

`\forcefunction` defines a new function. If the command `\`*`Function`* already exists, it is redefined.

11.2 Vector functions and polar coordinates

You can (re)define a vector function $f(t) = (x(t), y(t))$ using the `\newvectorfunction` declaration or any of its variants `\renewvectorfunction`, `\ensurevectorfunction` and `\forcevectorfunction`:

`\newvectorfunction{`⟨*Function*⟩`}{`⟨*Instructions to compute* `\x`, `\Dx`, `\y` *and* `\Dy` *from* `\t`⟩`}`

For example, you can define the function $f(t) = (t^2, t^3)$ in the following way:

```
\newvectorfunction{\F}{%
   \SQUARE{\t}{\x}         % x=t^2
   \MULTIPLY{2}{\t}{\Dx}   % x'=2t
   \CUBE{\t}{\y}           % y=t^3
   \MULTIPLY{3}{\x}{\Dy}   % y'=3t^2
}
```

Finally, to define the $r = r(\phi)$ function, in polar coordinates, we have the declarations `\newpolarfunction`, `\renewpolarfunction`, `\ensurepolarfunction` and `\forcepolarfunction`.

`\newpolarfunction{`⟨*Function*⟩`}{`⟨*Instructions to compute* `\r` *and* `\Dr` *from* `\t`⟩`}`

For example, you can define the *cardioide* curve $r(\phi) = 1+\cos\phi$, using high level instructions,

```
\SUMfunction{\ONEfunction}{\COSfunction}{\ffunction} % y=1 + cos t
\POLARfunction{\ffunction}{\cardioide}
```

or, with the `\newpolarfunction` declaration,

```
\newpolarfunction{\cardioide}{%
   \COS{\t}{\r}
   \ADD{1}{\r}{\r}         % r=1+cos t
   \SIN{\t}{\Dr}
   \MULTIPLY{-1}{\Dr}{\Dr} % r'=-sin t
}
```

12 Implementation (calculator)

```
⟨*calculator⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{calculator}[2012/06/10 v.1.0a]
```

12.1 Internal lengths and special nmbers

`\cctr@lengtha` and `\cctr@lengthb` will be used in internal calculations and comparisons.

```
\newdimen\cctr@lengtha
\newdimen\cctr@lengthb
```

`\cctr@epsilon` — `\cctr@epsilon` will store the closest to zero length in the TEX arithmetic: one scaled point (1 sp = 1/65536 pt). This means the smallest positive number will be $0.00002 \approx 1/65536 = 1/2^{16}$.

```
\newdimen\cctr@epsilon
\cctr@epsilon=1sp
```

`\cctr@logmaxnum` — The largest TEX number is $16383.99998 \approx 2^{14}$; `\cctr@logmaxnum` is the logarithm of this number, $9.704 \approx \log 16384$.

```
\def\cctr@logmaxnum{9.704}
```

12.2 Warning messages

```
\def\cctr@Warndivzero#1#2{%
      \PackageWarning{calculator}%
        {Division by 0.\MessageBreak
         I can't define #1/#2}}

\def\cctr@Warnnogcd{%
      \PackageWarning{calculator}%
        {gcd(0,0) is not well defined}}

\def\cctr@Warnnoposrad#1{%
      \PackageWarning{calculator}%
                     {The argument in square root\MessageBreak
                      must be non negative\MessageBreak
                      I can't define sqrt(#1)}}

\def\cctr@Warnnointexp#1#2{%
      \PackageWarning{calculator}%
                     {The exponent in power function\MessageBreak
```

```
                    must be an integer\MessageBreak
                    I can't define #1^#2}}

\def\cctr@Warnsingmatrix#1#2#3#4{%
      \PackageWarning{calculator}%
        {Matrix (#1 #2 ; #3 #4) is singular\MessageBreak
         Its inverse is not defined}}

\def\cctr@WarnsingTDmatrix#1#2#3#4#5#6#7#8#9{%
      \PackageWarning{calculator}%
        {Matrix (#1 #2 #3; #4 #5 #6; #7 #8 #9) is singular\MessageBreak
         Its inverse is not defined}}

\def\cctr@WarnIncLinSys{\PackageWarning{xpicture}{%
      Incompatible linear system}}

\def\cctr@WarnIncTDLinSys{\PackageWarning{xpicture}{%
      Incompatible or indeterminate linear system\MessageBreak
      For 3x3 systems I can solve only determinate systems}}

\def\cctr@WarnIndLinSys{\PackageWarning{xpicture}{%
      Indeterminate linear system.\MessageBreak
      I will choose one of the infinite solutions}}

\def\cctr@WarnZeroLinSys{\PackageWarning{xpicture}{%
      0x=0 linear system. Every vector is a solution!\MessageBreak
      I will choose the (0,0) solution}}

\def\cctr@Warninftan#1{%
             \PackageWarning{calculator}{%
                    Undefined tangent.\MessageBreak
                           The cosine of #1 is zero and, then,\MessageBreak
                           the tangent of #1 is not defined}}

\def\cctr@Warninfcotan#1{%
             \PackageWarning{calculator}{%
                    Undefined cotangent.\MessageBreak
                           The sine of #1 is zero and, then,\MessageBreak
                           the cotangent of #1 is not defined}}

\def\cctr@Warninfexp#1{%
             \PackageWarning{calculator}{%
                    The absolute value of the variable\MessageBreak
                    in the exponential function must be less than
                    \cctr@logmaxnum\MessageBreak
                   (the logarithm of the max number I know)\MessageBreak
                    I can't define exp(#1)}}

\def\cctr@Warninfexpb#1#2{%
             \PackageWarning{calculator}{%
```

```
                        The base\MessageBreak
                        in the exponential function must be positive.
                        \MessageBreak
                        I can't define #1^(#2)}}

\def\cctr@Warninflog#1{%
             \PackageWarning{calculator}{%
                        The value of the variable\MessageBreak
                        in the logarithm function must be positive\MessageBreak
                        I can't define log(#1)}}
```

12.3 Operations with numbers

Assignements and comparisons

`\COPY` `\COPY{`⟨*#1*⟩`}{`⟨*#2*⟩`}` defines the *#2* command as the number *#1*.

```
\def\COPY#1#2{\edef#2{#1}\ignorespaces}
```

`\GLOBALCOPY` Global version of `\COPY`. The new defined command *#2* is not changed outside groups.

```
\def\GLOBALCOPY#1#2{\xdef#2{#1}\ignorespaces}
```

`\@OUTPUTSOL` `\@OUTPUTSOL{`⟨*#1*⟩`}`: an internal macro to save solutions when a group is closed.

The global c.s. `\cctr@outa` preserves solutions. Whenever we use any temporary parameters in the definition of an instruction, we use a group to ensure the local character of those parameters. The instruction `\@OUTPUTSOL` is a bypass to export the solution.

```
\def\@OUTPUTSOL#1{\GLOBALCOPY{#1}{\cctr@outa}\endgroup\COPY{\cctr@outa}{#1}}
```

`\@OUTPUTSOLS` Analogous to `\@OUTPUTSOL`, preserving a pair of solutions.

```
\def\@OUTPUTSOLS#1#2{\GLOBALCOPY{#1}{\cctr@outa}
                       \GLOBALCOPY{#2}{\cctr@outb}\endgroup
                       \COPY{\cctr@outa}{#1}\COPY{\cctr@outb}{#2}}
```

`\MAX` `\MAX{`⟨*#1*⟩`}{`⟨*#2*⟩`}{`⟨*#3*⟩`}` defines the *#3* command as the maximum of numbers *#1* and *#2*.

```
\def\MAX#1#2#3{%
   \ifdim #1\p@ < #2\p@
      \COPY{#2}{#3}\else\COPY{#1}{#3}\fi\ignorespaces}
```

`\MIN` `\MIN{`⟨*#1*⟩`}{`⟨*#2*⟩`}{`⟨*#3*⟩`}` defines the *#3* command as the minimum of numbers *#1* and *#2*.

```
\def\MIN#1#2#3{%
   \ifdim #1\p@ > #2\p@
      \COPY{#2}{#3}\else\COPY{#1}{#3}\fi\ignorespaces}
```

Real arithmetic

`\ABSVALUE` `\ABSVALUE{`⟨*#1*⟩`}{`⟨*#2*⟩`}` defines the *#2* command as the absolute value of number *#1*.

```
\def\ABSVALUE#1#2{%
    \ifdim #1\p@<\z@
          \MULTIPLY{-1}{#1}{#2}\else\COPY{#1}{#2}\fi}
```

Product, sum and difference

\MULTIPLY `\MULTIPLY{`⟨*#1*⟩`}{`⟨*#2*⟩`}{`⟨*#3*⟩`}` defines the *#3* command as the product of numbers *#1* and *#2*.

```
\def\MULTIPLY#1#2#3{\cctr@lengtha=#1\p@
       \cctr@lengtha=#2\cctr@lengtha
       \edef#3{\expandafter\strip@pt\cctr@lengtha}\ignorespaces}
```

\ADD `\ADD{`⟨*#1*⟩`}{`⟨*#2*⟩`}{`⟨*#3*⟩`}` defines the *#3* command as the sum of numbers *#1* and *#2*.

```
\def\ADD#1#2#3{\cctr@lengtha=#1\p@
       \cctr@lengthb=#2\p@
       \advance\cctr@lengtha by \cctr@lengthb
       \edef#3{\expandafter\strip@pt\cctr@lengtha}\ignorespaces}
```

\SUBTRACT `\SUBTRACT{`⟨*#1*⟩`}{`⟨*#2*⟩`}{`⟨*#3*⟩`}` defines the *#3* command as the difference of numbers *#1* and *#2*.

```
\def\SUBTRACT#1#2#3{\ADD{#1}{-#2}{#3}}
```

Divisions We define several kinds of *divisions*: the quotient of two real numbers, the integer quotient, and the quotient of two lengths. The basic algorithm is a lightly modified version of the Beccari's division.

\DIVIDE `\DIVIDE{`⟨*#1*⟩`}{`⟨*#2*⟩`}{`⟨*#3*⟩`}` defines the *#3* command as the quotient of numbers *#1* and *#2*.

```
\def\DIVIDE#1#2#3{%
     \begingroup
```

Absolute values of dividend and divisor

```
       \ABSVALUE{#1}{\cctr@tempD}
       \ABSVALUE{#2}{\cctr@tempd}
```

The sign of quotient

```
       \ifdim#1\p@<\z@\ifdim#2\p@>\z@\COPY{-1}{\cctr@sign}
          \else\COPY{1}{\cctr@sign}\fi
       \else\ifdim#2\p@>\z@\COPY{1}{\cctr@sign}
               \else\COPY{-1}{\cctr@sign}\fi
            \fi
```

Integer part of quotient

```
       \@DIVIDE{\cctr@tempD}{\cctr@tempd}{\cctr@tempq}{\cctr@tempr}
       \COPY{\cctr@tempq.}{\cctr@Q}
```

Fractional part up to five decimal places. `\cctr@ndec` is the number of decimal places already computed.

```
       \COPY{0}{\cctr@ndec}
       \@whilenum \cctr@ndec<5 \do{%
```

Each decimal place is calculated by multiplying by 10 the last remainder and dividing it by the divisor. But when the remainder is greater than 1638.3, an overflow occurs, because 16383.99998 is the greatest number. So, instead, we multiply the divisor by 0.1.

```
           \ifdim\cctr@tempr\p@<1638\p@
```

```
                \MULTIPLY{\cctr@tempr}{10}{\cctr@tempD}
             \else
                \COPY{\cctr@tempr}{\cctr@tempD}
                \MULTIPLY{\cctr@tempd}{0.1}{\cctr@tempd}
             \fi
             \@DIVIDE{\cctr@tempD}{\cctr@tempd}{\cctr@tempq}{\cctr@tempr}
             \COPY{\cctr@Q\cctr@tempq}{\cctr@Q}
             \ADD{1}{\cctr@ndec}{\cctr@ndec}}%
```

Adjust the sign and return the solution.

```
         \MULTIPLY{\cctr@sign}{\cctr@Q}{#3}
         \@OUTPUTSOL{#3}}
```

`\@DIVIDE` The `\@DIVIDE(`⟨*#1*⟩`)` `(`⟨*#2*⟩`)(`⟨*#3*⟩`)(`⟨*#4*⟩`)` command computes *#1*/*#2* and returns an integer quotient (*#3*) and a real remainder (*#4*).

```
  \def\@DIVIDE#1#2#3#4{%
         \@INTEGERDIVIDE{#1}{#2}{#3}
         \MULTIPLY{#2}{#3}{#4}
         \SUBTRACT{#1}{#4}{#4}}
```

`\@INTEGERDIVIDE` `\@INTEGERDIVIDE` divides two numbers (not necessarily integer) and returns an integer (this is the integer quotient only for nonnegative integers).

```
\def\@INTEGERDIVIDE#1#2#3{%
         \cctr@lengtha=#1\p@
         \cctr@lengthb=#2\p@
         \ifdim\cctr@lengthb=\z@
            \let#3\undefined
            \cctr@Warndivzero#1#2%
         \else
            \divide\cctr@lengtha\cctr@lengthb
            \COPY{\number\cctr@lengtha}{#3}
         \fi\ignorespaces}
```

`\LENGTHDIVIDE` The quotient of two lengths must be a number (not a length). For example, one inch over one centimeter equals 2.54. `\LENGTHDIVIDE{`⟨*#1*⟩`}{`⟨*#2*⟩`}{`⟨*#3*⟩`}` stores in *#3* the quotient of the lenghts *#1* and *#2*.

```
\def\LENGTHDIVIDE#1#2#3{%
         \begingroup
         \cctr@lengtha=#1
         \cctr@lengthb=#2
         \edef\cctr@tempa{\expandafter\strip@pt\cctr@lengtha}%
         \edef\cctr@tempb{\expandafter\strip@pt\cctr@lengthb}%
         \DIVIDE{\cctr@tempa}{\cctr@tempb}{#3}
         \@OUTPUTSOL{#3}}
```

Powers

`\SQUARE` `\SQUARE{`⟨*#1*⟩`}{`⟨*#2*⟩`}` stores *#1* squared in *#2*.

```
\def\SQUARE#1#2{\MULTIPLY{#1}{#1}{#2}}
```

\CUBE \CUBE{⟨*#1*⟩}{⟨*#2*⟩} stores *#1* cubed in *#2*.

```
\def\CUBE#1#2{\MULTIPLY{#1}{#1}{#2}\MULTIPLY{#2}{#1}{#2}}
```

\POWER \POWER{⟨*#1*⟩}{⟨*#2*⟩}{⟨*#3*⟩} stores in *#3* the power $\#1^{\#2}$

```
\def\POWER#1#2#3{%
        \begingroup
        \INTEGERPART{#2}{\cctr@tempexp}
        \ifdim \cctr@tempexp\p@<#2\p@
          \cctr@Warnnointexp{#1}{#2}
          \let#3\undefined
        \else
```

This ensures that power will be defined only if the exponent is an integer.

```
           \@POWER{#1}{#2}{#3}\fi\@OUTPUTSOL{#3}}
```

```
\def\@POWER#1#2#3{%
       \begingroup
       \ifdim #2\p@<\z@
```

For negative exponents, $a^n = (1/a)^{-n}$.

```
              \DIVIDE{1}{#1}{\cctr@tempb}
              \MULTIPLY{-1}{#2}{\cctr@tempc}
              \@POWER{\cctr@tempb}{\cctr@tempc}{#3}
           \else
              \COPY{0}{\cctr@tempa}
              \COPY{1}{#3}
              \@whilenum \cctr@tempa<#2 \do {%
                 \MULTIPLY{#1}{#3}{#3}
                 \ADD{1}{\cctr@tempa}{\cctr@tempa}}%
        \fi\@OUTPUTSOL{#3}}
```

Integer arithmetic and related things

\INTEGERDIVISION \INTEGERDIVISION{⟨*#1*⟩}{⟨*#2*⟩}{⟨*#3*⟩}{⟨*#4*⟩} computes the division *#1*/*#2* and returns an integer quotient and a positive remainder.

```
\def\INTEGERDIVISION#1#2#3#4{%
        \begingroup
        \ABSVALUE{#2}{\cctr@tempd}
        \@DIVIDE{#1}{#2}{#3}{#4}
        \ifdim #4\p@<\z@
           \ifdim #1\p@<\z@
              \ifdim #2\p@<\z@
                 \ADD{#3}{1}{#3}
              \else
                 \SUBTRACT{#3}{1}{#3}
              \fi
              \ADD{#4}{\cctr@tempd}{#4}
        \fi\fi\@OUTPUTSOLS{#3}{#4}}
```

\MODULO \MODULO{⟨*#1*⟩}{⟨*#2*⟩}{⟨*#3*⟩} returns the remainder of division *#1*/*#2*.

```
\def\MODULO#1#2#3{%
       \begingroup
       \INTEGERDIVISION{#1}{#2}{\cctr@temp}{#3}\@OUTPUTSOL{#3}}
```

\INTEGERQUOTIENT

\INTEGERQUOTIENT{⟨*#1*⟩}{⟨*#2*⟩}{⟨*#3*⟩} returns the integer quotient of division *#1*/*#2*.

```
\def\INTEGERQUOTIENT#1#2#3{%
       \begingroup
       \INTEGERDIVISION{#1}{#2}{#3}{\cctr@temp}\@OUTPUTSOL{#3}}
```

\INTEGERPART

\INTEGERPART{⟨*#1*⟩}{⟨*#2*⟩} returns the integer part of *#2*.

```
\def\@@INTEGERPART#1.#2.#3)#4{\ifnum #11=1 \COPY{0}{#4}
                              \else \COPY{#1}{#4}\fi}
\def\@INTEGERPART#1#2{\expandafter\@@INTEGERPART#1..){#2}}
\def\INTEGERPART#1#2{\begingroup
                     \ifdim #1\p@<\z@
                        \MULTIPLY{-1}{#1}{\cctr@temp}
                        \INTEGERPART{\cctr@temp}{#2}
                        \ifdim #2\p@<\cctr@temp\p@
                           \SUBTRACT{-#2}{1}{#2}
                        \else \COPY{-#2}{#2}
                        \fi
                     \else
                        \@INTEGERPART{#1}{#2}
                     \fi\@OUTPUTSOL{#2}}
```

\FLOOR

\FLOOR is an alias for \INTEGERPART.

```
\let\FLOOR\INTEGERPART
```

\FRACTIONALPART

\FRACTIONALPART{⟨*#1*⟩}{⟨*#2*⟩} returns the fractional part of *#2*.

```
\def\@@FRACTIONALPART#1.#2.#3)#4{\ifnum #2=11  \COPY{0}{#4}
                                 \else \COPY{0.#2}{#4}\fi}
\def\@FRACTIONALPART#1#2{\expandafter\@@FRACTIONALPART#1..){#2}}
\def\FRACTIONALPART#1#2{\begingroup
                     \ifdim #1\p@<\z@
                        \INTEGERPART{#1}{\cctr@tempA}
                        \SUBTRACT{#1}{\cctr@tempA}{#2}
                     \else
                        \@FRACTIONALPART{#1}{#2}
                     \fi\@OUTPUTSOL{#2}}
```

\TRUNCATE

\TRUNCATE[⟨*#1*⟩]{⟨*#2*⟩}{⟨*#3*⟩} truncates *#2* to *#1* (0, 1, 2 (default), 3 or 4) digits.

```
\def\TRUNCATE{\@ifnextchar[\@@TRUNCATE\@TRUNCATE}
\def\@TRUNCATE#1#2{\@@TRUNCATE[2]{#1}{#2}}
\def\@@TRUNCATE[#1]#2#3{%
    \begingroup
    \INTEGERPART{#2}{\cctr@tempa}
    \ifdim \cctr@tempa\p@ = #2\p@
       \expandafter\@@@TRUNCATE#2.00000)[#1]{#3}
    \else
       \expandafter\@@@TRUNCATE#200000.)[#1]{#3}
```

```
    \fi
    \@OUTPUTSOL{#3}}
\def\@@@TRUNCATE#1.#2#3#4#5#6.#7)[#8]#9{%
    \ifcase #8
       \COPY{#1}{#9}
    \or\COPY{#1.#2}{#9}
    \or\COPY{#1.#2#3}{#9}
    \or\COPY{#1.#2#3#4}{#9}
    \or\COPY{#1.#2#3#4#5}{#9}
    \fi}
```

\ROUND \ROUND[⟨*#1*⟩]{⟨*#2*⟩}{⟨*#3*⟩} rounds *#2* to *#1* (0, 1, 2 (default), 3 or 4) digits.

```
\def\ROUND{\@ifnextchar[\@@ROUND\@ROUND}
\def\@ROUND#1#2{\@@ROUND[2]{#1}{#2}}
\def\@@ROUND[#1]#2#3{%
       \begingroup
       \ifdim#2\p@<\z@
           \MULTIPLY{-1}{#2}{\cctr@temp}
           \@@ROUND[#1]{\cctr@temp}{#3}\COPY{-#3}{#3}
       \else
          \@@TRUNCATE[#1]{#2}{\cctr@tempe}
          \SUBTRACT{#2}{\cctr@tempe}{\cctr@tempc}
          \POWER{10}{#1}{\cctr@tempb}
          \MULTIPLY{\cctr@tempb}{\cctr@tempc}{\cctr@tempc}
          \ifdim\cctr@tempc\p@<0.5\p@
          \else
             \DIVIDE{1}{\cctr@tempb}{\cctr@tempb}
             \ADD{\cctr@tempe}{\cctr@tempb}{\cctr@tempe}
          \fi
          \@@TRUNCATE[#1]{\cctr@tempe}{#3}
       \fi
       \@OUTPUTSOL{#3}}
```

\GCD \GCD{⟨*#1*⟩}{⟨*#2*⟩}{⟨*#3*⟩} Greatest common divisor, using the Euclidean algorithm

```
\def\GCD#1#2#3{%
       \begingroup
       \ABSVALUE{#1}{\cctr@tempa}
       \ABSVALUE{#2}{\cctr@tempb}
       \MAX{\cctr@tempa}{\cctr@tempb}{\cctr@tempc}
       \MIN{\cctr@tempa}{\cctr@tempb}{\cctr@tempa}
       \COPY{\cctr@tempc}{\cctr@tempb}
       \ifnum \cctr@tempa = 0
          \ifnum \cctr@tempb = 0
             \cctr@Warnnogcd
             \let#3\undefined
          \else
          \COPY{\cctr@tempb}{#3}
          \fi
       \else
```

Euclidean algorithm: if $c \equiv b \pmod{a}$ then $\gcd(b, a) = \gcd(a, c)$. Iterating this property, we obtain $\gcd(b, a)$ as the last nonzero residual.

```
          \@whilenum \cctr@tempa > \z@ \do {%
             \COPY{\cctr@tempa}{#3}%
             \MODULO{\cctr@tempb}{\cctr@tempa}{\cctr@tempc}%
             \COPY\cctr@tempa\cctr@tempb%
             \COPY\cctr@tempc\cctr@tempa}
       \fi\ignorespaces\@OUTPUTSOL{#3}}
```

\LCM

\LCM{⟨*#1*⟩}{⟨*#2*⟩}{⟨*#3*⟩} Least common multiple.

```
\def\LCM#1#2#3{%
       \GCD{#1}{#2}{#3}%
       \ifx #3\undefined \COPY{0}{#3}
       \else
          \DIVIDE{#1}{#3}{#3}
          \MULTIPLY{#2}{#3}{#3}
          \ABSVALUE{#3}{#3}
       \fi}
```

\FRACTIONSIMPLIFY

\FRACTIONSIMPLIFY{⟨*#1*⟩}{⟨*#2*⟩}{⟨*#3*⟩}{⟨*#4*⟩} Fraction simplification: *#3*/*#4* is the irreducible fraction equivalent to *#1*/*#2*.

```
\def\FRACTIONSIMPLIFY#1#2#3#4{%
       \ifnum #1=\z@
          \COPY{0}{#3}\COPY{1}{#4}
       \else
          \GCD{#1}{#2}{#3}%
          \DIVIDE{#2}{#3}{#4}
          \DIVIDE{#1}{#3}{#3}
          \ifnum #4<0 \MULTIPLY{-1}{#4}{#4}\MULTIPLY{-1}{#3}{#3}\fi
       \fi\ignorespaces}
```

Elementary functions

Square roots

\SQUAREROOT

\SQUAREROOT{⟨*#2*⟩}{⟨*#2*⟩} defines *#2* as the square root of *#1*, using the Newton's method: $x_{n+1} = x_n - (x_n^2 - \#1)/(2x_n)$.

```
\def\SQUAREROOT#1#2{%
       \begingroup
       \ifdim #1\p@ = \z@
         \COPY{0}{#2}
       \else
         \ifdim #1\p@ < \z@
             \let#2\undefined
             \cctr@Warnnoposrad{#1}%
          \else
```

We take *#1* as the initial approximation.

```
             \COPY{#1}{#2}
```

`\cctr@lengthb` will be the difference of two successive iterations.

We start with `\cctr@lengthb=5\p@` to ensure almost one iteration.

```
            \cctr@lengthb=5\p@
```

Successive iterations

```
            \@whilenum \cctr@lengthb>\cctr@epsilon \do {%
```

Copy the actual approximation to `\cctr@tempw`

```
               \COPY{#2}{\cctr@tempw}
               \DIVIDE{#1}{\cctr@tempw}{\cctr@tempz}
               \ADD{\cctr@tempw}{\cctr@tempz}{\cctr@tempz}
               \DIVIDE{\cctr@tempz}{2}{\cctr@tempz}
```

Now, `\cctr@tempz` is the new approximation.

```
               \COPY{\cctr@tempz}{#2}
```

Finally, we store in `\cctr@lengthb` the difference of the two last approximations, finishing the loop.

```
               \SUBTRACT{#2}{\cctr@tempw}{\cctr@tempw}
               \cctr@lengthb=\cctr@tempw\p@%
               \ifnum
                  \cctr@lengthb<\z@ \cctr@lengthb=-\cctr@lengthb
               \fi}
         \fi\fi\@OUTPUTSOL{#2}}
```

`\SQRT` `\SQRT` is an alias for `\SQUAREROOT`.

```
\let\SQRT\SQUAREROOT
```

Trigonometric functions For a variable close enough to zero, the sine and tangent functions are computed using some continued fractions. Then, all trigonometric functions are derived from well-known formulas.

`\SIN` `\SIN{`⟨*#1*⟩`}{`⟨*#2*⟩`}`. Sine of *#1*.

```
\def\SIN#1#2{%
   \begingroup
```

Exact sine for $t \in \{\pi/2, -\pi/2, 3\pi/2\}$

```
   \ifdim #1\p@=-\numberHALFPI\p@ \COPY{-1}{#2}
   \else
        \ifdim #1\p@=\numberHALFPI\p@ \COPY{1}{#2}
        \else
             \ifdim #1\p@=\numberTHREEHALFPI\p@ \COPY{-1}{#2}
              \else
```

If $|t| > \pi/2$, change t to a smaller value.

```
                  \ifdim#1\p@<-\numberHALFPI\p@
                     \ADD{#1}{\numberTWOPI}{\cctr@tempb}
                     \SIN{\cctr@tempb}{#2}
                  \else
                     \ifdim #1\p@<\numberHALFPI\p@
```

Compute the sine.

```
                        \@BASICSINE{#1}{#2}
                     \else
                        \ifdim #1\p@<\numberTHREEHALFPI\p@
                          \SUBTRACT{\numberPI}{#1}{\cctr@tempb}
                          \SIN{\cctr@tempb}{#2}
                       \else
                         \SUBTRACT{#1}{\numberTWOPI}{\cctr@tempb}
                         \SIN{\cctr@tempb}{#2}
\fi\fi\fi\fi\fi\fi\@OUTPUTSOL{#2}}
```

\@BASICSINE `\@BASICSINE{`⟨*#1*⟩`}{`⟨*#2*⟩`}` applies this approximation:

$$\sin x = \cfrac{x}{1 + \cfrac{x^2}{2\cdot 3 - x^2 + \cfrac{2\cdot 3x^2}{4\cdot 5 - x^2 + \cfrac{4\cdot 5x^2}{6\cdot 7 - x^2 + \cdots}}}}$$

```
\def\@BASICSINE#1#2{%
       \begingroup
       \ABSVALUE{#1}{\cctr@tempa}
```

Exact sine of zero

```
          \ifdim\cctr@tempa\p@=\z@ \COPY{0}{#2}
          \else
```

For t very close to zero, $\sin t \approx t$.

```
             \ifdim \cctr@tempa\p@<0.009\p@\COPY{#1}{#2}
             \else
```

Compute the continued fraction.

```
                \SQUARE{#1}{\cctr@tempa}
                \DIVIDE{\cctr@tempa}{42}{#2}
                \SUBTRACT{1}{#2}{#2}
                \MULTIPLY{#2}{\cctr@tempa}{#2}
                \DIVIDE{#2}{20}{#2}
                \SUBTRACT{1}{#2}{#2}
                \MULTIPLY{#2}{\cctr@tempa}{#2}
                \DIVIDE{#2}{6}{#2}
                \SUBTRACT{1}{#2}{#2}
                \MULTIPLY{#2}{#1}{#2}
          \fi\fi\@OUTPUTSOL{#2}}
```

\COS `\COS{`⟨*#1*⟩`}{`⟨*#2*⟩`}`. Cosine of *#1*: $\cos t = \sin(t + \pi/2)$.

```
\def\COS#1#2{%
        \begingroup
        \ADD{\numberHALFPI}{#1}{\cctr@tempc}
        \SIN{\cctr@tempc}{#2}\@OUTPUTSOL{#2}}
```

\TAN `\TAN{`⟨*#1*⟩`}{`⟨*#2*⟩`}`. Tangent of *#1*.

```
\def\TAN#1#2{%
          \begingroup
```

Tangent is infinite for $t = \pm\pi/2$

```
          \ifdim #1\p@=-\numberHALFPI\p@
             \cctr@Warninftan{#1}
             \let#2\undefined
          \else
             \ifdim #1\p@=\numberHALFPI\p@
                 \cctr@Warninftan{#1}
                 \let#2\undefined
             \else
```

If $|t| > \pi/2$, change t to a smaller value.

```
                \ifdim #1\p@<-\numberHALFPI\p@
                   \ADD{#1}{\numberPI}{\cctr@tempb}
                   \TAN{\cctr@tempb}{#2}
                \else
                   \ifdim #1\p@<\numberHALFPI\p@
```

Compute the tangent.

```
                      \@BASICTAN{#1}{#2}
                   \else
                      \SUBTRACT{#1}{\numberPI}{\cctr@tempb}
                      \TAN{\cctr@tempb}{#2}
          \fi\fi\fi\fi\@OUTPUTSOL{#2}}
```

\@BASICTAN `\@BASICTAN{`⟨*#1*⟩`}{`⟨*#2*⟩`}` applies this approximation:

$$\tan x = \cfrac{1}{\cfrac{1}{x} - \cfrac{1}{\cfrac{3}{x} - \cfrac{1}{\cfrac{5}{x} - \cfrac{1}{\cfrac{7}{x} - \cfrac{1}{\cfrac{9}{x} - \cfrac{1}{\cfrac{11}{x} - \cdots}}}}}}$$

```
\def\@BASICTAN#1#2{%
       \begingroup
       \ABSVALUE{#1}{\cctr@tempa}
```

Exact tangent of zero.

```
          \ifdim\cctr@tempa\p@=\z@ \COPY{0}{#2}
          \else
```

For t very close to zero, $\tan t \approx t$.

```
             \ifdim\cctr@tempa\p@<0.04\p@
                \COPY{#1}{#2}
             \else
```

Compute the continued fraction.

```
                \DIVIDE{#1}{11}{#2}
                \DIVIDE{9}{#1}{\cctr@tempa}
                \SUBTRACT{\cctr@tempa}{#2}{#2}
                \DIVIDE{1}{#2}{#2}
                \DIVIDE{7}{#1}{\cctr@tempa}
                \SUBTRACT{\cctr@tempa}{#2}{#2}
                \DIVIDE{1}{#2}{#2}
                \DIVIDE{5}{#1}{\cctr@tempa}
                \SUBTRACT{\cctr@tempa}{#2}{#2}
                \DIVIDE{1}{#2}{#2}
                \DIVIDE{3}{#1}{\cctr@tempa}
                \SUBTRACT{\cctr@tempa}{#2}{#2}
                \DIVIDE{1}{#2}{#2}
                \DIVIDE{1}{#1}{\cctr@tempa}
                \SUBTRACT{\cctr@tempa}{#2}{#2}
                \DIVIDE{1}{#2}{#2}
          \fi\fi\@OUTPUTSOL{#2}}
```

`\COT` `\COT{`⟨*#1*⟩`}{`⟨*#2*⟩`}`. Cotangent of *#1*: If $\cos t = 0$ then $\cot t = 0$; if $\tan t = 0$ then $\cot t = \infty$. Otherwise, $\cot t = 1/\tan t$.

```
\def\COT#1#2{%
        \begingroup
        \COS{#1}{#2}
        \ifdim #2\p@ = \z@
        \COPY{0}{#2}
        \else
        \TAN{#1}{#2}
        \ifdim #2\p@ = \z@
        \cctr@Warninfcotan{#1}
        \let#2\undefined
        \else
        \DIVIDE{1}{#2}{#2}
        \fi\fi\@OUTPUTSOL{#2}}
```

`\DEGtoRAD` `\DEGtoRAD{`⟨*#1*⟩`}{`⟨*#2*⟩`}`. Convert degrees to radians.

```
\def\DEGtoRAD#1#2{\DIVIDE{#1}{57.29578}{#2}}
```

`\RADtoDEG` `\RADtoDEG{`⟨*#1*⟩`}{`⟨*#2*⟩`}`. Convert radians to degrees.

```
\def\RADtoDEG#1#2{\MULTIPLY{#1}{57.29578}{#2}}
```

`\REDUCERADIANSANGLE` Reduces to the trigonometrically equivalent arc in $]-\pi, \pi]$.

```
\def\REDUCERADIANSANGLE#1#2{%
         \COPY{#1}{#2}
         \ifdim #1\p@ < -\numberPI\p@
                \ADD{#1}{\numberTWOPI}{#2}
                \REDUCERADIANSANGLE{#2}{#2}
         \fi
         \ifdim #1\p@ >  \numberPI\p@
```

```
                \SUBTRACT{#1}{\numberTWOPI}{#2}
                \REDUCERADIANSANGLE{#2}{#2}
        \fi
        \ifdim #1\p@ = -180\p@ \COPY{\numberPI}{#2} \fi}
```

\REDUCEDEGREESANGLE Reduces to the trigonometrically equivalent angle in $]-180, 180]$.

```
\def\REDUCEDEGREESANGLE#1#2{%
        \COPY{#1}{#2}
        \ifdim #1\p@ < -180\p@
                \ADD{#1}{360}{#2}
                \REDUCEDEGREESANGLE{#2}{#2}
        \fi
        \ifdim #1\p@ >  180\p@
                \SUBTRACT{#1}{360}{#2}
                \REDUCEDEGREESANGLE{#2}{#2}
        \fi
        \ifdim #1\p@ = -180\p@ \COPY{180}{#2} \fi}
```

Trigonometric functions in degrees Four next commands compute trigonometric functions in *degrees*. By default, a circle has 360 degrees, but we can use an arbitrary number of divisions using the optional argument of these commands.

\DEGREESSIN `\DEGREESSIN[`⟨*#1*⟩`]{`⟨*#2*⟩`}{`⟨*#3*⟩`}`. Sine of *#2 degrees*.

```
\def\DEGREESSIN{\@ifnextchar[\@@DEGREESSIN\@DEGREESSIN}
```

\DEGREESCOS `\DEGREESCOS[`⟨*#1*⟩`]{`⟨*#2*⟩`}{`⟨*#3*⟩`}`. Cosine of *#2 degrees*.

```
\def\DEGREESCOS{\@ifnextchar[\@@DEGREESCOS\@DEGREESCOS}
```

\DEGREESTAN `\DEGREESTAN[`⟨*#1*⟩`]{`⟨*#2*⟩`}{`⟨*#3*⟩`}`. Tangent of *#2 degrees*.

```
\def\DEGREESTAN{\@ifnextchar[\@@DEGREESTAN\@DEGREESTAN}
```

\DEGREESCOT `\DEGREESCOT[`⟨*#1*⟩`]{`⟨*#2*⟩`}{`⟨*#3*⟩`}`. Cotangent of *#2 degrees*.

```
\def\DEGREESCOT{\@ifnextchar[\@@DEGREESCOT\@DEGREESCOT}
```

\@DEGREESSIN `\@DEGREESSIN` computes the sine in sexagesimal *degrees*.

```
\def\@DEGREESSIN#1#2{%
     \begingroup
     \ifdim #1\p@=-90\p@ \COPY{-1}{#2}
     \else
          \ifdim #1\p@=90\p@ \COPY{1}{#2}
          \else
                   \ifdim #1\p@=270\p@ \COPY{-1}{#2}
          \else
            \ifdim#1\p@<-90\p@
                 \ADD{#1}{360}{\cctr@tempb}
                 \DEGREESSIN{\cctr@tempb}{#2}
            \else
               \ifdim #1\p@<90\p@
                     \DEGtoRAD{#1}{\cctr@tempb}
```

```
                    \@BASICSINE{\cctr@tempb}{#2}
                  \else
                       \ifdim #1\p@<270\p@
                         \SUBTRACT{180}{#1}{\cctr@tempb}
                          \DEGREESSIN{\cctr@tempb}{#2}
                         \else
                              \SUBTRACT{#1}{360}{\cctr@tempb}
                              \DEGREESSIN{\cctr@tempb}{#2}
  \fi\fi\fi\fi\fi\fi\@OUTPUTSOL{#2}}
```

\@DEGREESCOS

\@DEGREESCOS computes the cosine in sexagesimal *degrees*.

```
\def\@DEGREESCOS#1#2{%
        \begingroup
        \ADD{90}{#1}{\cctr@tempc}
        \DEGREESSIN{\cctr@tempc}{#2}\@OUTPUTSOL{#2}}
```

\@DEGREESTAN

\@DEGREESTAN computes the tangent in sexagesimal *degrees*.

```
\def\@DEGREESTAN#1#2{%
          \begingroup
          \ifdim #1\p@=-90\p@
          \cctr@Warninftan{#1}
          \let#2\undefined
          \else
            \ifdim #1\p@=90\p@
            \cctr@Warninftan{#1}
            \let#2\undefined
          \else
          \ifdim #1\p@<-90\p@
            \ADD{#1}{180}{\cctr@tempb} \DEGREESTAN{\cctr@tempb}{#2}
          \else
              \ifdim #1\p@<90\p@
                   \DEGtoRAD{#1}{\cctr@tempb}
                   \@BASICTAN{\cctr@tempb}{#2}
                 \else
                     \SUBTRACT{#1}{180}{\cctr@tempb}
                     \DEGREESTAN{\cctr@tempb}{#2}
   \fi\fi\fi\fi\@OUTPUTSOL{#2}}
```

\@DEGREESCOT

\@DEGREESCOT computes the cotangent in sexagesimal *degrees*.

```
\def\@DEGREESCOT#1#2{%
        \begingroup
        \DEGREESCOS{#1}{#2}
        \ifdim #2\p@ = \z@
        \COPY{0}{#2}
        \else
        \DEGREESTAN{#1}{#2}
        \ifdim #2\p@ = \z@
        \cctr@Warninfcotan{#1}
        \let#2\undefined
        \else
```

```
        \DIVIDE{1}{#2}{#2}
        \fi\fi\@OUTPUTSOL{#2}}
```

For an arbitrary number of *degrees*, we normalise to 360 degrees and, then, call the former functions.

\@@DEGREESSIN — \@@DEGREESSIN computes the sine. A circle has *#1 degrees*.

```
\def\@@DEGREESSIN[#1]#2#3{\@CONVERTDEG{#1}{#2}
        \@DEGREESSIN{\@DEGREES}{#3}}
```

\@@DEGREESCOS — \@@DEGREESCOS computes the sine. A circle has *#1 degrees*.

```
\def\@@DEGREESCOS[#1]#2#3{\@CONVERTDEG{#1}{#2}
        \DEGREESCOS{\@DEGREES}{#3}}
```

\@@DEGREESTAN — \@@DEGREESTAN computes the sine. A circle has *#1 degrees*.

```
\def\@@DEGREESTAN[#1]#2#3{\@CONVERTDEG{#1}{#2}
        \DEGREESTAN{\@DEGREES}{#3}}
```

\@@DEGREESCOT — \@@DEGREESCOT computes the sine. A circle has *#1 degrees*.

```
\def\@@DEGREESCOT[#1]#2#3{\@CONVERTDEG{#1}{#2}
        \DEGREESCOT{\@DEGREES}{#3}}
```

\@CONVERTDEG — \@CONVERTDEG normalises to sexagesimal degrees.

```
\def\@CONVERTDEG#1#2{\DIVIDE{#2}{#1}{\@DEGREES}
        \MULTIPLY{\@DEGREES}{360}{\@DEGREES}}
```

Exponential functions

\EXP — \EXP[⟨*#1*⟩]{⟨*#2*⟩}{⟨*#3*⟩} computes the exponential $\#3 = \#1^{\#2}$. Default for *#1* is number e.

```
\def\EXP{\@ifnextchar[\@@EXP\@EXP}
```

\@@EXP — \@@EXP[⟨*#1*⟩]{⟨*#2*⟩}{⟨*#3*⟩} computes $\#3 = \#1^{\#2}$

```
\def\@@EXP[#1]#2#3{%
      \begingroup
```

#1 must be a positive number.

```
      \ifdim #1\p@<\cctr@epsilon
          \cctr@Warninfexpb{#1}{#2}
          \let#3\undefined
      \else
```

$a^b = \exp(b \log a)$.

```
          \LOG{#1}{\cctr@log}
          \MULTIPLY{#2}{\cctr@log}{\cctr@log}
          \@EXP{\cctr@log}{#3}
      \fi\@OUTPUTSOL{#3}}
```

\@EXP \@EXP{⟨*#1*⟩}{⟨*#2*⟩} computes $\#3 = \mathrm{e}^{\#2}$

```
\def\@EXP#1#2{%
      \begingroup
      \ABSVALUE{#1}{\cctr@absval}
```

If $|t|$ is greater than `\cctr@logmaxnum` then $\exp t$ is too large.

```
      \ifdim \cctr@absval\p@>\cctr@logmaxnum\p@
              \cctr@Warninfexp{#1}
             \let#2\undefined
      \else
          \ifdim #1\p@ < \z@
```

We call `\@BASICEXP` when $t \in [-6, 3]$. Otherwise we use the equality $\exp t = (\exp t/2)^2$.

```
            \ifdim #1\p@ > -6.00002\p@
               \@BASICEXP{#1}{#2}
            \else
               \DIVIDE{#1}{2}{\cctr@expt}
               \@EXP{\cctr@expt}{\cctr@expy}
               \SQUARE{\cctr@expy}{#2}
            \fi
          \else
               \ifdim #1\p@ < 3.00002\p@
                  \@BASICEXP{#1}{#2}
               \else
                  \DIVIDE{#1}{2}{\cctr@expt}
                  \@EXP{\cctr@expt}{\cctr@expy}
                  \SQUARE{\cctr@expy}{#2}
               \fi
\fi\fi\@OUTPUTSOL{#2}}
```

\@BASICEXP \@BASICEXP{⟨*#1*⟩}{⟨*#2*⟩} applies this approximation:

$$\exp x \approx 1 + \cfrac{2x}{2 - x + \cfrac{x^2/6}{1 + \cfrac{x^2/60}{1 + \cfrac{x^2/140}{1 + \cfrac{x^2/256}{1 + \cfrac{x^2}{396}}}}}}$$

```
\def\@BASICEXP#1#2{%
       \begingroup
       \SQUARE{#1}\cctr@tempa
       \DIVIDE{\cctr@tempa}{396}{#2}
       \ADD{1}{#2}{#2}
       \DIVIDE\cctr@tempa{#2}{#2}
       \DIVIDE{#2}{256}{#2}
       \ADD{1}{#2}{#2}
       \DIVIDE\cctr@tempa{#2}{#2}
       \DIVIDE{#2}{140}{#2}
```

```
       \ADD{1}{#2}{#2}
       \DIVIDE\cctr@tempa{#2}{#2}
       \DIVIDE{#2}{60}{#2}
       \ADD{1}{#2}{#2}
       \DIVIDE\cctr@tempa{#2}{#2}
       \DIVIDE{#2}{6}{#2}
       \ADD{2}{#2}{#2}
       \SUBTRACT{#2}{#1}{#2}
       \DIVIDE{#1}{#2}{#2}
       \MULTIPLY{2}{#2}{#2}
       \ADD{1}{#2}{#2}\@OUTPUTSOL{#2}}
```

Hyperbolic functions

\COSH

\COSH. Hyperbolic cosine: $\cosh t = (\exp t + \exp(-t))/2$.

```
\def\COSH#1#2{%
       \begingroup
       \ABSVALUE{#1}{\cctr@absval}
       \ifdim \cctr@absval\p@>\cctr@logmaxnum\p@
          \cctr@Warninfexp{#1}
          \let#2\undefined
       \else
          \EXP{#1}{\cctr@expx}
          \MULTIPLY{-1}{#1}{\cctr@minust}
          \EXP{\cctr@minust}{\cctr@expminusx}
          \ADD{\cctr@expx}{\cctr@expminusx}{#2}
          \DIVIDE{#2}{2}{#2}
       \fi\@OUTPUTSOL{#2}}
```

\SINH

\SINH. Hyperbolic sine: $\sinh t = (\exp t - \exp(-t))/2$.

```
\def\SINH#1#2{%
       \begingroup
       \ABSVALUE{#1}{\cctr@absval}
       \ifdim \cctr@absval\p@>\cctr@logmaxnum\p@
          \cctr@Warninfexp{#1}
          \let#2\undefined
       \else
          \EXP{#1}{\cctr@expx}
          \MULTIPLY{-1}{#1}{\cctr@minust}
          \EXP{\cctr@minust}{\cctr@expminusx}
          \SUBTRACT{\cctr@expx}{\cctr@expminusx}{#2}
          \DIVIDE{#2}{2}{#2}
       \fi\@OUTPUTSOL{#2}}
```

\TANH

\TANH. Hyperbolic tangent: $\tanh t = \sinh t / \cosh t$.

```
\def\TANH#1#2{%
       \begingroup
       \ABSVALUE{#1}{\cctr@absval}
       \ifdim \cctr@absval\p@>\cctr@logmaxnum\p@
          \cctr@Warninfexp{#1}
```

```
            \let#2\undefined
        \else
            \SINH{#1}{\cctr@tanhnum}
            \COSH{#1}{\cctr@tanhden}
            \DIVIDE{\cctr@tanhnum}{\cctr@tanhden}{#2}
        \fi\@OUTPUTSOL{#2}}
```

\COTH `\COTH`. Hyperbolic cotangent $\coth t = \cosh t / \sinh t$.

```
\def\COTH#1#2{%
        \begingroup
        \ABSVALUE{#1}{\cctr@absval}
        \ifdim \cctr@absval\p@>\cctr@logmaxnum\p@
            \cctr@Warninfexp{#1}
            \let#2\undefined
        \else
            \SINH{#1}{\cctr@tanhden}
            \COSH{#1}{\cctr@tanhnum}
            \DIVIDE\cctr@tanhnum\cctr@tanhden{#2}
        \fi\@OUTPUTSOL{#2}}
```

Logarithm

\LOG `\LOG[`⟨*#1*⟩`]{`⟨*#2*⟩`}{`⟨*#3*⟩`}` computes the logarithm $\#3 = \log_{\#1} \#2$. Default for *#1* is number e.

```
\def\LOG{\@ifnextchar[\@@LOG\@LOG}
```

\@LOG `\@LOG{`⟨*#1*⟩`}{`⟨*#2*⟩`}` computes $\#2 = \log \#1$

```
\def\@LOG#1#2{%
        \begingroup
```

The argument t must be positive.

```
        \ifdim #1\p@<\cctr@epsilon
            \cctr@Warninflog{#1}
            \let#2\undefined
        \else
        \ifdim #1\p@ > \numberETWO\p@
```

If $t > \mathrm{e}^2$, $\log t = \log \mathrm{e} + \log(t/\mathrm{e}) = 1 + \log(t/\mathrm{e})$

```
            \DIVIDE{#1}{\numberE}{\cctr@ae}
            \@LOG{\cctr@ae}{#2}
            \ADD{1}{#2}{#2}
        \else
            \ifdim #1\p@ < 1\p@
```

If $t < 1$, $\log t = \log(1/\mathrm{e}) + \log(t\mathrm{e}) = -1 + \log(t\mathrm{e})$

```
                \MULTIPLY{\numberE}{#1}{\cctr@ae}
                \LOG{\cctr@ae}{#2}
                \SUBTRACT{#2}{1}{#2}
            \else
```

For $t \in [1, \mathrm{e}^2]$ we call `\@@BASICLOG`.

```
              \@BASICLOG{#1}{#2}
\fi\fi\fi\@OUTPUTSOL{#2}}
```

`\@@LOG` `\@@LOG[`⟨*#1*⟩`]{`⟨*#2*⟩`}{`⟨*#3*⟩`}` computes $\#3 = \log_{\#1} \#2 = \log(\#2)/\log(\#1)$

```
\def\@@LOG[#1]#2#3{\begingroup
        \@LOG{#1}{\cctr@loga}
        \@LOG{#2}{\cctr@logx}
        \DIVIDE{\cctr@logx}{\cctr@loga}{#3}\@OUTPUTSOL{#3}}
```

`\@BASICLOG` `\@BASICLOG{`⟨*#1*⟩`}{`⟨*#2*⟩`}` applies the Newton's method to calculate $x = \log t$:

$$x_{n+1} = x_n + \frac{t}{\mathrm{e}^{x_n}} - 1$$

```
\def\@BASICLOG#1#2{\begingroup
%  We take $\textit{\#1}-1$ as the initial approximation.
%    \begin{macrocode}
              \SUBTRACT{#1}{1}{\cctr@tempw}
```

We start with `\cctr@lengthb=5\p@` to ensure almost one iteration.

```
              \cctr@lengthb=5\p@%
              \cctr@lengtha=\cctr@epsilon%
```

Successive iterations

```
        \@whilenum \cctr@lengthb>\cctr@lengtha \do {%
            \COPY{\cctr@tempw}{\cctr@tempoldw}
            \EXP{\cctr@tempw}{\cctr@tempxw}
            \DIVIDE{#1}{\cctr@tempxw}{\cctr@tempxw}
            \ADD{\cctr@tempw}{\cctr@tempxw}{\cctr@tempw}
            \SUBTRACT{\cctr@tempw}{1}{\cctr@tempw}
            \SUBTRACT{\cctr@tempw}{\cctr@tempoldw}{\cctr@tempdif}
            \cctr@lengthb=\cctr@tempdif\p@%
            \ifnum
               \cctr@lengthb<\z@ \cctr@lengthb=-\cctr@lengthb
            \fi}%
        \COPY{\cctr@tempw}{#2}\@OUTPUTSOL{#2}}
```

12.4 Matrix arithmetics

Vector operations

`\VECTORSIZE` The *size* of a vector is 2 or 3. `\VECTORSIZE(`⟨*#1*⟩`){`⟨*#2*⟩`}` stores in *#2* the size of `(`⟨*#1*⟩`)`.
Almost all vector commands needs to know the vector size.

```
\def\VECTORSIZE(#1)#2{\expandafter\@VECTORSIZE(#1,,){#2}}
\def\@VECTORSIZE(#1,#2,#3,#4)#5{\ifx$#3$\COPY{2}{#5}
                                \else\COPY{3}{#5}\fi\ignorespaces}
```

`\VECTORCOPY` `\VECTORCOPY(`⟨*#1,#2*⟩`)(`⟨*#3,#4*⟩`)` stores *#1* and *#2* in *#3* and *#4*.
`\VECTORCOPY(`⟨*#1,#2,#3*⟩`)(`⟨*#4,#5#6*⟩`)` stores *#1*, *#2* and *#3* in *#4* and *#5* and *#6*.

```
\def\@@VECTORCOPY(#1,#2)(#3,#4){%
```

```
    \COPY{#1}{#3}\COPY{#2}{#4}}

\def\@@@VECTORCOPY(#1,#2,#3)(#4,#5,#6){%
    \COPY{#1}{#4}\COPY{#2}{#5}\COPY{#3}{#6}}

\def\VECTORCOPY(#1)(#2){%
        \VECTORSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@VECTORCOPY(#1)(#2)
        \else \@@@VECTORCOPY(#1)(#2)\fi}
```

\VECTORGLOBALCOPY

\VECTORGLOBALCOPY is the global version of \VECTORCOPY

```
\def\@@VECTORGLOBALCOPY(#1,#2)(#3,#4){%
    \GLOBALCOPY{#1}{#3}\GLOBALCOPY{#2}{#4}}

\def\@@@VECTORGLOBALCOPY(#1,#2,#3)(#4,#5,#6){%
    \GLOBALCOPY{#1}{#4}\GLOBALCOPY{#2}{#5}\GLOBALCOPY{#3}{#6}}

\def\VECTORGLOBALCOPY(#1)(#2){%
        \VECTORSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@VECTORGLOBALCOPY(#1)(#2)
        \else \@@@VECTORGLOBALCOPY(#1)(#2)\fi}
```

\@OUTPUTVECTOR

```
\def\@@OUTPUTVECTOR(#1,#2){%
    \VECTORGLOBALCOPY(#1,#2)(\cctr@outa,\cctr@outb)
    \endgroup\VECTORCOPY(\cctr@outa,\cctr@outb)(#1,#2)}

\def\@@@OUTPUTVECTOR(#1,#2,#3){%
    \VECTORGLOBALCOPY(#1,#2,#3)(\cctr@outa,\cctr@outb,\cctr@outc)
    \endgroup\VECTORCOPY(\cctr@outa,\cctr@outb,\cctr@outc)(#1,#2,#3)}

\def\@OUTPUTVECTOR(#1){\VECTORSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@OUTPUTVECTOR(#1)
        \else \@@@OUTPUTVECTOR(#1)\fi}
```

\SCALARPRODUCT

Scalar product of two vectors.

```
\def\@@SCALARPRODUCT(#1,#2)(#3,#4)#5{%
       \MULTIPLY{#1}{#3}{#5}
       \MULTIPLY{#2}{#4}\cctr@tempa
       \ADD{#5}{\cctr@tempa}{#5}}

\def\@@@SCALARPRODUCT(#1,#2,#3)(#4,#5,#6)#7{%
       \MULTIPLY{#1}{#4}{#7}
       \MULTIPLY{#2}{#5}\cctr@tempa
       \ADD{#7}{\cctr@tempa}{#7}
       \MULTIPLY{#3}{#6}\cctr@tempa
       \ADD{#7}{\cctr@tempa}{#7}}
```

```
\def\SCALARPRODUCT(#1)(#2)#3{%
        \begingroup
        \VECTORSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@SCALARPRODUCT(#1)(#2){#3}
        \else \@@@SCALARPRODUCT(#1)(#2){#3}\fi\@OUTPUTSOL{#3}}
```

\VECTORADD Sum of two vectors.

```
\def\@@VECTORADD(#1,#2)(#3,#4)(#5,#6){%
       \ADD{#1}{#3}{#5}
       \ADD{#2}{#4}{#6}}

\def\@@@VECTORADD(#1,#2,#3)(#4,#5,#6)(#7,#8,#9){%
       \ADD{#1}{#4}{#7}
       \ADD{#2}{#5}{#8}
       \ADD{#3}{#6}{#9}}

\def\VECTORADD(#1)(#2)(#3){%
        \VECTORSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@VECTORADD(#1)(#2)(#3)
        \else \@@@VECTORADD(#1)(#2)(#3)\fi}
```

\VECTORSUB Difference of two vectors.

```
\def\@@VECTORSUB(#1,#2)(#3,#4)(#5,#6){%
       \VECTORADD(#1,#2)(-#3,-#4)(#5,#6)}

\def\@@@VECTORSUB(#1,#2,#3)(#4,#5,#6)(#7,#8,#9){%
       \VECTORADD(#1,#2,#3)(-#4,-#5,-#6)(#7,#8,#9)}

\def\VECTORSUB(#1)(#2)(#3){%
        \VECTORSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@VECTORSUB(#1)(#2)(#3)
        \else \@@@VECTORSUB(#1)(#2)(#3)\fi}
```

\VECTORABSVALUE Absolute value of a each entry of a vector.

```
\def\@@VECTORABSVALUE(#1,#2)(#3,#4){%
       \ABSVALUE{#1}{#3}\ABSVALUE{#2}{#4}}

\def\@@@VECTORABSVALUE(#1,#2,#3)(#4,#5,#6){%
       \ABSVALUE{#1}{#4}\ABSVALUE{#2}{#5}\ABSVALUE{#3}{#6}}

\def\VECTORABSVALUE(#1)(#2){%
        \VECTORSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@VECTORABSVALUE(#1)(#2)
        \else \@@@VECTORABSVALUE(#1)(#2)\fi}
```

`\SCALARVECTORPRODUCT` Scalar-vector product.

```
\def\@@SCALARVECTORPRODUCT#1(#2,#3)(#4,#5){%
        \MULTIPLY{#1}{#2}{#4}
        \MULTIPLY{#1}{#3}{#5}}

\def\@@@SCALARVECTORPRODUCT#1(#2,#3,#4)(#5,#6,#7){%
        \MULTIPLY{#1}{#2}{#5}
        \MULTIPLY{#1}{#3}{#6}
        \MULTIPLY{#1}{#4}{#7}}

\def\SCALARVECTORPRODUCT#1(#2)(#3){%
         \VECTORSIZE(#2){\cctr@size}
         \ifnum\cctr@size=2
            \@@SCALARVECTORPRODUCT{#1}(#2)(#3)
         \else \@@@SCALARVECTORPRODUCT{#1}(#2)(#3)\fi}
```

`\VECTORNORM` Euclidean norm of a vector.

```
\def\VECTORNORM(#1)#2{%
        \begingroup
        \SCALARPRODUCT(#1)(#1){\cctr@temp}
        \SQUAREROOT{\cctr@temp}{#2}\@OUTPUTSOL{#2}}
```

`\UNITVECTOR` Unitary vector parallel to a given vector.

```
\def\UNITVECTOR(#1)(#2){%
        \begingroup
        \VECTORNORM(#1){\cctr@tempa}
        \DIVIDE{1}{\cctr@tempa}{\cctr@tempa}
        \SCALARVECTORPRODUCT{\cctr@tempa}(#1)(#2)\@OUTPUTVECTOR(#2)}
```

Matrix operations

Here, we need to define some internal macros to simulate commands with more than nine arguments.

`\@TDMATRIXCOPY` This command copies a 3×3 matrix to the commands `\cctr@solAA`, `\cctr@solAB`, …, `\cctr@solCC`.

```
\def\@TDMATRIXCOPY(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
        \COPY{#1}{\cctr@solAA}
        \COPY{#2}{\cctr@solAB}
        \COPY{#3}{\cctr@solAC}
        \COPY{#4}{\cctr@solBA}
        \COPY{#5}{\cctr@solBB}
        \COPY{#6}{\cctr@solBC}
        \COPY{#7}{\cctr@solCA}
        \COPY{#8}{\cctr@solCB}
        \COPY{#9}{\cctr@solCC}}
```

`\@TDMATRIXSOL` This command copies the commands `\cctr@solAA`, `\cctr@solAB`, …, `\cctr@solCC` to a 3×3 matrix. This macro is used to store the results of a matrix operation.

```
\def\@TDMATRIXSOL(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
        \COPY{\cctr@solAA}{#1}
        \COPY{\cctr@solAB}{#2}
        \COPY{\cctr@solAC}{#3}
        \COPY{\cctr@solBA}{#4}
        \COPY{\cctr@solBB}{#5}
        \COPY{\cctr@solBC}{#6}
        \COPY{\cctr@solCA}{#7}
        \COPY{\cctr@solCB}{#8}
        \COPY{\cctr@solCC}{#9}}
```

\@TDMATRIXGLOBALSOL

```
\def\@TDMATRIXGLOBALSOL(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
        \GLOBALCOPY{\cctr@solAA}{#1}
        \GLOBALCOPY{\cctr@solAB}{#2}
        \GLOBALCOPY{\cctr@solAC}{#3}
        \GLOBALCOPY{\cctr@solBA}{#4}
        \GLOBALCOPY{\cctr@solBB}{#5}
        \GLOBALCOPY{\cctr@solBC}{#6}
        \GLOBALCOPY{\cctr@solCA}{#7}
        \GLOBALCOPY{\cctr@solCB}{#8}
        \GLOBALCOPY{\cctr@solCC}{#9}}
```

\@TDMATRIXNOSOL This command undefines a 3×3 matrix when a matrix problem has no solution.

```
\def\@TDMATRIXNOSOL(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
          \let#1\undefined
          \let#2\undefined
          \let#3\undefined
          \let#4\undefined
          \let#5\undefined
          \let#6\undefined
          \let#7\undefined
          \let#8\undefined
          \let#9\undefined
          }
```

\@@TDMATRIXSOL This command stores or undefines the solution.

```
\def\@@TDMATRIXSOL(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
        \ifx\cctr@solAA\undefined
          \@TDMATRIXNOSOL(#1,#2,#3;#4,#5,#6;#7,#8,#9)%
        \else
          \@TDMATRIXSOL(#1,#2,#3;#4,#5,#6;#7,#8,#9)\fi}
```

\@NUMBERSOL This command stores the scalar solution of a matrix operation.

```
\def\@NUMBERSOL#1{\COPY{\cctr@sol}{#1}}
```

\MATRIXSIZE Size (2 or 3) of a matrix.

```
\def\MATRIXSIZE(#1)#2{\expandafter\@MATRIXSIZE(#1;;)(#2}}
\def\@MATRIXSIZE(#1;#2;#3;#4)#5{\ifx$#3$\COPY{2}{#5}
                                \else\COPY{3}{#5}\fi\ignorespaces}
```

\MATRIXCOPY Store a matrix in 4 or 9 commands.

```
\def\@@MATRIXCOPY(#1,#2;#3,#4)(#5,#6;#7,#8){%
    \COPY{#1}{#5}\COPY{#2}{#6}\COPY{#3}{#7}\COPY{#4}{#8}}

\def\@@@MATRIXCOPY(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
     \@TDMATRIXCOPY(#1,#2,#3;#4,#5,#6;#7,#8,#9)
     \@TDMATRIXSOL}

\def\MATRIXCOPY(#1)(#2){%
        \MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@MATRIXCOPY(#1)(#2)
        \else \@@@MATRIXCOPY(#1)(#2)\fi}
```

\MATRIXGLOBALCOPY Global version of \MATRIXCOPY.

```
\def\@@MATRIXGLOBALCOPY(#1,#2;#3,#4)(#5,#6;#7,#8){%
    \GLOBALCOPY{#1}{#5}\GLOBALCOPY{#2}{#6}\GLOBALCOPY{#3}{#7}\GLOBALCOPY{#4}{#8}}

\def\@@@MATRIXGLOBALCOPY(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
     \@TDMATRIXCOPY(#1,#2,#3;#4,#5,#6;#7,#8,#9)
     \@TDMATRIXGLOBALSOL}

\def\MATRIXGLOBALCOPY(#1)(#2){%
        \MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@MATRIXGLOBALCOPY(#1)(#2)
        \else \@@@MATRIXGLOBALCOPY(#1)(#2)\fi}
```

\@OUTPUTMATRIX

```
\def\@@OUTPUTMATRIX(#1,#2;#3,#4){%
    \MATRIXGLOBALCOPY(#1,#2;#3,#4)(\cctr@outa,\cctr@outb;\cctr@outc,\cctr@outd)
    \endgroup\MATRIXCOPY(\cctr@outa,\cctr@outb;\cctr@outc,\cctr@outd)(#1,#2;#3,#4)}

\def\@@@OUTPUTMATRIX(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
    \MATRIXGLOBALCOPY(#1,#2,#3;#4,#5,#6;#7,#8,#9)(%
        \cctr@outa,\cctr@outb,\cctr@outc;
        \cctr@outd,\cctr@oute,\cctr@outf;
        \cctr@outg,\cctr@outh,\cctr@outi)
    \endgroup\MATRIXCOPY(%
        \cctr@outa,\cctr@outb,\cctr@outc;
        \cctr@outd,\cctr@oute,\cctr@outf;
        \cctr@outg,\cctr@outh,\cctr@outi)(#1,#2,#3;#4,#5,#6;#7,#8,#9)}

\def\@OUTPUTMATRIX(#1){\MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@OUTPUTMATRIX(#1)
        \else \@@@OUTPUTMATRIX(#1)\fi}
```

\TRANSPOSEMATRIX Matrix transposition.

```
\def\@@TRANSPOSEMATRIX(#1,#2;#3,#4)(#5,#6;#7,#8){%
```

```
       \COPY{#1}{#5}\COPY{#3}{#6}\COPY{#2}{#7}\COPY{#4}{#8}}

\def\@@@TRANSPOSEMATRIX(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
       \@TDMATRIXCOPY(#1,#4,#7;#2,#5,#8;#3,#6,#9)
       \@TDMATRIXSOL}

\def\TRANSPOSEMATRIX(#1)(#2){%
        \begingroup
        \MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@TRANSPOSEMATRIX(#1)(#2)
        \else \@@@TRANSPOSEMATRIX(#1)(#2)\fi\@OUTPUTMATRIX(#2)}
```

\MATRIXADD Sum of two matrices.

```
\def\@@MATRIXADD(#1;#2)(#3;#4)(#5,#6;#7,#8){%
       \VECTORADD(#1)(#3)(#5,#6)
       \VECTORADD(#2)(#4)(#7,#8)}

\def\@@@MATRIXADD(#1;#2;#3)(#4;#5;#6){%
       \VECTORADD(#1)(#4)(\cctr@solAA,\cctr@solAB,\cctr@solAC)
       \VECTORADD(#2)(#5)(\cctr@solBA,\cctr@solBB,\cctr@solBC)
       \VECTORADD(#3)(#6)(\cctr@solCA,\cctr@solCB,\cctr@solCC)
       \@TDMATRIXSOL}

\def\MATRIXADD(#1)(#2)(#3){%
        \begingroup
        \MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@MATRIXADD(#1)(#2)(#3)
        \else \@@@MATRIXADD(#1)(#2)(#3)\fi\@OUTPUTMATRIX(#3)}
```

\MATRIXSUB Difference of two matrices.

```
\def\@@MATRIXSUB(#1;#2)(#3;#4)(#5,#6;#7,#8){%
       \VECTORSUB(#1)(#3)(#5,#6)
       \VECTORSUB(#2)(#4)(#7,#8)}

\def\@@@MATRIXSUB(#1;#2;#3)(#4;#5;#6){%
       \VECTORSUB(#1)(#4)(\cctr@solAA,\cctr@solAB,\cctr@solAC)
       \VECTORSUB(#2)(#5)(\cctr@solBA,\cctr@solBB,\cctr@solBC)
       \VECTORSUB(#3)(#6)(\cctr@solCA,\cctr@solCB,\cctr@solCC)
       \@TDMATRIXSOL}

\def\MATRIXSUB(#1)(#2)(#3){%
        \begingroup
        \MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@MATRIXSUB(#1)(#2)(#3)
        \else \@@@MATRIXSUB(#1)(#2)(#3)\fi\@OUTPUTMATRIX(#3)}
```

\MATRIXABSVALUE Absolute value (of each entry) of a matrix.

```
\def\@@MATRIXABSVALUE(#1;#2)(#3;#4){%
       \VECTORABSVALUE(#1)(#3)\VECTORABSVALUE(#2)(#4)}

\def\@@@MATRIXABSVALUE(#1;#2;#3)(#4;#5;#6){%
       \VECTORABSVALUE(#1)(#4)\VECTORABSVALUE(#2)(#5)\VECTORABSVALUE(#3)(#6)}

\def\MATRIXABSVALUE(#1)(#2){%
        \begingroup
        \MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@MATRIXABSVALUE(#1)(#2)
        \else \@@@MATRIXABSVALUE(#1)(#2)\fi\@OUTPUTMATRIX(#2)}
```

\MATRIXVECTORPRODUCT Matrix-vector product.

```
\def\@@MATRIXVECTORPRODUCT(#1;#2)(#3)(#4,#5){%
       \SCALARPRODUCT(#1)(#3){#4}
       \SCALARPRODUCT(#2)(#3){#5}}

\def\@@@MATRIXVECTORPRODUCT(#1;#2;#3)(#4)(#5,#6,#7){%
       \SCALARPRODUCT(#1)(#4){#5}
       \SCALARPRODUCT(#2)(#4){#6}
       \SCALARPRODUCT(#3)(#4){#7}}

\def\MATRIXVECTORPRODUCT(#1)(#2)(#3){%
        \begingroup
        \MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@MATRIXVECTORPRODUCT(#1)(#2)(#3)
        \else \@@@MATRIXVECTORPRODUCT(#1)(#2)(#3)\fi\@OUTPUTVECTOR(#3)}
```

\VECTORMATRIXPRODUCT Vector-matrix product.

```
\def\@@VECTORMATRIXPRODUCT(#1)(#2,#3;#4,#5)(#6,#7){%
       \SCALARPRODUCT(#1)(#2,#4){#6}
       \SCALARPRODUCT(#1)(#3,#5){#7}}

\def\@@@VECTORMATRIXPRODUCT(#1,#2,#3)(#4;#5;#6)(#7){%
       \SCALARVECTORPRODUCT{#1}(#4)(#7)
       \SCALARVECTORPRODUCT{#2}(#5)(\cctr@tempa,\cctr@tempb,\cctr@tempc)
       \VECTORADD(#7)(\cctr@tempa,\cctr@tempb,\cctr@tempc)(#7)
       \SCALARVECTORPRODUCT{#3}(#6)(\cctr@tempa,\cctr@tempb,\cctr@tempc)
       \VECTORADD(#7)(\cctr@tempa,\cctr@tempb,\cctr@tempc)(#7)}

\def\VECTORMATRIXPRODUCT(#1)(#2)(#3){%
        \begingroup
        \VECTORSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@VECTORMATRIXPRODUCT(#1)(#2)(#3)
        \else \@@@VECTORMATRIXPRODUCT(#1)(#2)(#3)\fi\@OUTPUTVECTOR(#3)}
```

\SCALARMATRIXPRODUCT Scalar-matrix product.

```
\def\@@SCALARMATRIXPRODUCT#1(#2;#3)(#4,#5;#6,#7){%
      \SCALARVECTORPRODUCT{#1}(#2)(#4,#5)
      \SCALARVECTORPRODUCT{#1}(#3)(#6,#7)}

\def\@@@SCALARMATRIXPRODUCT#1(#2;#3;#4){%
      \SCALARVECTORPRODUCT{#1}(#2)(\cctr@solAA,\cctr@solAB,\cctr@solAC)
      \SCALARVECTORPRODUCT{#1}(#3)(\cctr@solBA,\cctr@solBB,\cctr@solBC)
      \SCALARVECTORPRODUCT{#1}(#4)(\cctr@solCA,\cctr@solCB,\cctr@solCC)
      \@TDMATRIXSOL}

\def\SCALARMATRIXPRODUCT#1(#2)(#3){%
       \begingroup
       \MATRIXSIZE(#2){\cctr@size}
       \ifnum\cctr@size=2
          \@@SCALARMATRIXPRODUCT{#1}(#2)(#3)
       \else \@@@SCALARMATRIXPRODUCT{#1}(#2)(#3)\fi\@OUTPUTMATRIX(#3)}
```

\MATRIXPRODUCT Product of two matrices.

```
\def\@@MATRIXPRODUCT(#1)(#2,#3;#4,#5)(#6,#7;#8,#9){%
      \MATRIXVECTORPRODUCT(#1)(#2,#4)(#6,#8)
      \MATRIXVECTORPRODUCT(#1)(#3,#5)(#7,#9)}

\def\@@@MATRIXPRODUCT(#1;#2;#3)(#4){%
      \VECTORMATRIXPRODUCT(#1)(#4)(\cctr@solAA,\cctr@solAB,\cctr@solAC)
      \VECTORMATRIXPRODUCT(#2)(#4)(\cctr@solBA,\cctr@solBB,\cctr@solBC)
      \VECTORMATRIXPRODUCT(#3)(#4)(\cctr@solCA,\cctr@solCB,\cctr@solCC)
      \@TDMATRIXSOL}

\def\MATRIXPRODUCT(#1)(#2)(#3){%
       \begingroup
       \MATRIXSIZE(#1){\cctr@size}
       \ifnum\cctr@size=2
          \@@MATRIXPRODUCT(#1)(#2)(#3)
       \else \@@@MATRIXPRODUCT(#1)(#2)(#3)\fi\@OUTPUTMATRIX(#3)}
```

\DETERMINANT Determinant of a matrix.

```
\def\@@DETERMINANT(#1,#2;#3,#4)#5{%
      \MULTIPLY{#1}{#4}{#5}
      \MULTIPLY{#2}{#3}{\cctr@tempa}
      \SUBTRACT{#5}{\cctr@tempa}{#5}}

\def\@@@DETERMINANT(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
      \DETERMINANT(#5,#6;#8,#9){\cctr@det}\MULTIPLY{#1}{\cctr@det}{\cctr@sol}
      \DETERMINANT(#6,#4;#9,#7){\cctr@det}\MULTIPLY{#2}{\cctr@det}{\cctr@det}
                                    \ADD{\cctr@sol}{\cctr@det}{\cctr@sol}
      \DETERMINANT(#4,#5;#7,#8){\cctr@det}\MULTIPLY{#3}{\cctr@det}{\cctr@det}
                                    \ADD{\cctr@sol}{\cctr@det}{\cctr@sol}
      \@NUMBERSOL}

\def\DETERMINANT(#1)#2{%
```

```
        \begingroup
        \MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@DETERMINANT(#1){#2}
        \else \@@@DETERMINANT(#1){#2}\fi\@OUTPUTSOL{#2}}
```

\INVERSEMATRIX Inverse of a matrix.

```
\def\@@INVERSEMATRIX(#1,#2;#3,#4)(#5,#6;#7,#8){%
       \ifdim \cctr@@det\p@ <\cctr@epsilon  % Matrix is singular
          \let#5\undefined
          \let#6\undefined
          \let#7\undefined
          \let#8\undefined
          \cctr@Warnsingmatrix{#1}{#2}{#3}{#4}%
       \else \COPY{#1}{#8}
          \COPY{#4}{#5}
          \MULTIPLY{-1}{#3}{#7}
          \MULTIPLY{-1}{#2}{#6}
          \DIVIDE{1}{\cctr@det}{\cctr@det}
          \SCALARMATRIXPRODUCT{\cctr@det}(#5,#6;#7,#8)(#5,#6;#7,#8)
       \fi}

\def\@@@INVERSEMATRIX(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
              \ifdim \cctr@@det\p@ <\cctr@epsilon  % Matrix is singular
          \@TDMATRIXNOSOL(\cctr@solAA,\cctr@solAB,\cctr@solAC;
                          \cctr@solBA,\cctr@solBB,\cctr@solBC;
                          \cctr@solCA,\cctr@solCB,\cctr@solCC)
          \cctr@WarnsingTDmatrix{#1}{#2}{#3}{#4}{#5}{#6}{#7}{#8}{#9}%
       \else
          \@ADJMATRIX(#1,#2,#3;#4,#5,#6;#7,#8,#9)
          \@SCLRDIVVECT{\cctr@det}(\cctr@solAA,\cctr@solAB,\cctr@solAC)(%
                               \cctr@solAA,\cctr@solAB,\cctr@solAC)
          \@SCLRDIVVECT{\cctr@det}(\cctr@solBA,\cctr@solBB,\cctr@solBC)(%
                               \cctr@solBA,\cctr@solBB,\cctr@solBC)
          \@SCLRDIVVECT{\cctr@det}(\cctr@solCA,\cctr@solCB,\cctr@solCC)(%
                               \cctr@solCA,\cctr@solCB,\cctr@solCC)
       \fi
       \@@TDMATRIXSOL}

\def\@SCLRDIVVECT#1(#2,#3,#4)(#5,#6,#7){%
             \DIVIDE{#2}{#1}{#5}\DIVIDE{#3}{#1}{#6}\DIVIDE{#4}{#1}{#7}}

\def\@ADJMATRIX(#1,#2,#3;#4,#5,#6;#7,#8,#9){%
          \DETERMINANT(#5,#6;#8,#9){\cctr@solAA}
          \DETERMINANT(#6,#4;#9,#7){\cctr@solBA}
          \DETERMINANT(#4,#5;#7,#8){\cctr@solCA}
          \DETERMINANT(#8,#9;#2,#3){\cctr@solAB}
          \DETERMINANT(#1,#3;#7,#9){\cctr@solBB}
          \DETERMINANT(#2,#1;#8,#7){\cctr@solCB}
          \DETERMINANT(#2,#3;#5,#6){\cctr@solAC}
```

```
        \DETERMINANT(#3,#1;#6,#4){\cctr@solBC}
        \DETERMINANT(#1,#2;#4,#5){\cctr@solCC}}

\def\INVERSEMATRIX(#1)(#2){%
       \begingroup
       \DETERMINANT(#1){\cctr@det}
       \ABSVALUE{\cctr@det}{\cctr@@det}
       \MATRIXSIZE(#1){\cctr@size}
       \ifnum\cctr@size=2
          \@@INVERSEMATRIX(#1)(#2)
       \else
          \@@@INVERSEMATRIX(#1)(#2)\fi\@OUTPUTMATRIX(#2)}
```

`\SOLVELINEARSYSTEM` Solving a linear system (two equations and two unknowns or three equations and three unknowns).

```
\def\@INCSYS#1#2{\cctr@WarnIncLinSys
      \let#1\undefined\let#2\undefined}

\def\@SOLPART#1#2#3#4{\cctr@WarnIndLinSys
                      \DIVIDE{#1}{#2}{#3}
                      \COPY{0}{#4}}

\def\@TDINCSYS(#1,#2,#3){\cctr@WarnIncTDLinSys
                         \let#1\undefined
                         \let#2\undefined
                         \let#3\undefined}

\def\@@SOLVELINEARSYSTEM(#1,#2;#3,#4)(#5,#6)(#7,#8){%
       \DETERMINANT(#1,#2;#3,#4)\cctr@deta
       \DETERMINANT(#5,#2;#6,#4)\cctr@detb
       \DETERMINANT(#1,#5;#3,#6)\cctr@detc
       \ABSVALUE{\cctr@deta}{\cctr@@deta}
       \ABSVALUE{\cctr@detb}{\cctr@@detb}
       \ABSVALUE{\cctr@detc}{\cctr@@detc}
       \ifdim \cctr@@deta\p@>\cctr@epsilon% Regular matrix. Determinate system
          \DIVIDE{\cctr@detb}{\cctr@deta}{#7}
          \DIVIDE{\cctr@detc}{\cctr@deta}{#8}
       \else  % Singular matrix    \cctr@deta=0
          \ifdim \cctr@@detb\p@>\cctr@epsilon% Incompatible system
             \@INCSYS#7#8
          \else
             \ifdim \cctr@@detc\p@>\cctr@epsilon% Incompatible system
                \@INCSYS#7#8
             \else
                \MATRIXABSVALUE(#1,#2;#3,#4)(\cctr@tempa,\cctr@tempb;
                                           \cctr@tempc,\cctr@tempd)
                \ifdim \cctr@tempa\p@ > \cctr@epsilon
                                            % Indeterminate system
                   \@SOLPART{#5}{#1}{#7}{#8}
                \else
```

```
                    \ifdim \cctr@tempb\p@ > \cctr@epsilon
                                               % Indeterminate system
                       \@SOLPART{#5}{#2}{#8}{#7}
                    \else
                       \ifdim \cctr@tempc\p@ > \cctr@epsilon
                                               % Indeterminate system
                          \@SOLPART{#6}{#3}{#7}{#8}
                       \else
                          \ifdim \cctr@tempd\p@ > \cctr@epsilon
                                               % Indeterminate system
                             \@SOLPART{#6}{#4}{#8}{#7}
                          \else
                             \VECTORNORM(#5,#6){\cctr@tempa}
                             \ifdim \cctr@tempa\p@ > \cctr@epsilon
                                               % Incompatible system
                                \@INCSYS#7#8
                             \else
                                   \cctr@WarnZeroLinSys
                                   \COPY{0}{#7}\COPY{0}{#8}
                                               % 0x=0 Indeterminate system
        \fi\fi\fi\fi\fi\fi\fi\fi}

\def\@@@SOLVELINEARSYSTEM(#1)(#2)(#3){%
       \DETERMINANT(#1){\cctr@det}
       \ABSVALUE{\cctr@det}{\cctr@@det}
       \ifdim\cctr@@det\p@<\cctr@epsilon
          \@TDINCSYS(#3)
          \else
             \@ADJMATRIX(#1)
             \MATRIXVECTORPRODUCT(\cctr@solAA,\cctr@solAB,\cctr@solAC;
                                  \cctr@solBA,\cctr@solBB,\cctr@solBC;
                                  \cctr@solCA,\cctr@solCB,\cctr@solCC)(#2)(#3)
             \@SCLRDIVVECT{\cctr@det}(#3)(#3)
          \fi}

\def\SOLVELINEARSYSTEM(#1)(#2)(#3){%
        \begingroup
        \MATRIXSIZE(#1){\cctr@size}
        \ifnum\cctr@size=2
           \@@SOLVELINEARSYSTEM(#1)(#2)(#3)
        \else
           \@@@SOLVELINEARSYSTEM(#1)(#2)(#3)
        \fi\@OUTPUTVECTOR(#3)}
```

Predefined numbers

`\numberPI` The number π

```
\def\numberPI{3.14159}
```

`\numberTWOPI` 2π

```
\MULTIPLY{\numberPI}{2}{\numberTWOPI}
```

`\numberHALFPI` $\pi/2$

```
\DIVIDE{\numberPI}{2}{\numberHALFPI}
```

`\numberTHREEHALFPI` $3\pi/2$

```
\MULTIPLY{\numberPI}{1.5}{\numberTHREEHALFPI}
```

`\numberTHIRDPI` $\pi/3$

```
\DIVIDE{\numberPI}{3}{\numberTHIRDPI}
```

`\numberQUARTERPI` $\pi/4$

```
\DIVIDE{\numberPI}{4}{\numberQUARTERPI}
```

`\numberFIFTHPI` $\pi/5$

```
\DIVIDE{\numberPI}{5}{\numberFIFTHPI}
```

`\numberSIXTHPI` $\pi/6$

```
\DIVIDE{\numberPI}{6}{\numberSIXTHPI}
```

`\numberE` The number e

```
\def\numberE{2.71828}
```

`\numberINVE` 1/e

```
\DIVIDE{1}{\numberE}{\numberINVE}
```

`\numberETWO` e^2

```
\SQUARE{\numberE}{\numberETWO}
```

`\numberINVETWO` $1/\mathrm{e}^2$

```
\SQUARE{\numberINVE}{\numberINVETWO}
```

`\numberLOGTEN` $\log 10$

```
\def\numberLOGTEN{2.30258}
```

`\numberGOLD` The golden ratio ϕ

```
\def\numberGOLD{1.61803}
```

`\numberINVGOLD` $1/\phi$

```
\def\numberINVGOLD{0.61803}
```

`\numberSQRTTWO` $\sqrt{2}$

```
\def\numberSQRTTWO{1.41421}
```

`\numberSQRTTHREE` $\sqrt{3}$

```
\def\numberSQRTTHREE{1.73205}
```

`\numberSQRTFIVE` $\sqrt{5}$

```
\def\numberSQRTFIVE{2.23607}
```

`\numberCOSXLV` $\cos 45^o$ (or $\cos \pi/4$)

```
\def\numberCOSXLV{0.70711}
```

`\numberCOSXXX` $\cos 30^o$ (or $\cos \pi/6$)

```
\def\numberCOSXXX{0.86603}
```

⟨/calculator⟩

13 Implementation (**calculator**)

⟨*calculus⟩

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{calculus}
  [2012/06/10 v.1.0a]
```

This package requires the calculator package.

```
\RequirePackage{calculator}
```

13.1 Error and info messages

For scalar functions

Error message to be issued when you attempt to define, with `\newfunction`, an already defined command:

```
\def\ccls@ErrorFuncDef#1{%
      \PackageError{calculus}%
        {\noexpand#1 command already defined}
        {The \noexpand#1 control sequence is already defined\MessageBreak
         If you want to redefine the \noexpand#1 command as a
         function\MessageBreak
         please, use the \noexpand\renewfunction command}}
```

Error message to be issued when you attempt to redefine, with `\renewfunction`, an undefined command:

```
\def\ccls@ErrorFuncUnDef#1{%
      \PackageError{calculus}%
        {\noexpand#1 command undefined}
        {The \noexpand#1 control sequence is not currently defined\MessageBreak
         If you want to define the \noexpand#1 command as a function\MessageBreak
         please, use the \noexpand\newfunction command}}
```

Info message to be issued when `\ensurefunction` does not changes an already defined command:

```
\def\ccls@InfoFuncEns#1{%
      \PackageInfo{calculus}%
      {\noexpand#1 command already defined\MessageBreak
       the \noexpand\ensurefunction command will not redefine it}}
```

For polar functions

```
\def\ccls@ErrorPFuncDef#1{%
      \PackageError{calculus}%
```

```
      {\noexpand#1 command already defined}
      {The \noexpand#1 control sequence is already defined\MessageBreak
       If you want to redefine the \noexpand#1
       command as a polar function\MessageBreak
       please, use the \noexpand\renewpolarfunction command}}

\def\ccls@ErrorPFuncUnDef#1{%
    \PackageError{calculus}%
      {\noexpand#1 command undefined}
      {The \noexpand#1 control sequence
       is not currently defined.\MessageBreak
       If you want to define the \noexpand#1 command as a polar
       function\MessageBreak
       please, use the \noexpand\newpolarfunction command}}

\def\ccls@InfoPFuncEns#1{%
    \PackageInfo{calculus}%
    {\noexpand#1 command already defined\MessageBreak
     the \noexpand\ensurepolarfunction command does not redefine it}}
```

For vector functions

```
\def\ccls@ErrorVFuncDef#1{%
    \PackageError{calculus}%
      {\noexpand#1 command already defined}
      {The \noexpand#1 control sequence is already defined\MessageBreak
       If you want to redefine the \noexpand#1 command as a vector
       function\MessageBreak
       please, use the \noexpand\renewvectorfunction command}}

\def\ccls@ErrorVFuncUnDef#1{%
    \PackageError{calculus}%
      {\noexpand#1 command undefined}
      {The \noexpand#1 control sequence is not currently
       defined.\MessageBreak
       If you want to define the \noexpand#1 command as a vector
       function\MessageBreak
       please, use the \noexpand\newvectorfunction command}}

\def\ccls@InfoVFuncEns#1{%
    \PackageInfo{calculus}%
    {\noexpand#1 command already defined\MessageBreak
     the \noexpand\ensurevectorfunction command does not redefine it}}
```

13.2 New functions

New scalar functions

`\newfunction` The `\newfunction{#1}{#2}` instruction defines a new function called #1. #2 is the list of instructions to calculate the function `\y` and his derivative `\Dy` from the `\t` variable.

```
\def\newfunction#1#2{%
     \ifx #1\undefined
```

```
        \ccls@deffunction{#1}{#2}
      \else
         \ccls@ErrorFuncDef{#1}
      \fi}
```

`\renewfunction` \renewfunction redefines #1, as a new function, if this command is already defined.

```
\def\renewfunction#1#2{%
      \ifx #1\undefined
         \ccls@ErrorFuncUnDef{#1}
      \else
         \ccls@deffunction{#1}{#2}
      \fi}
```

`\ensurefunction` \ensurefunction defines the new function #1 (only if this macro is undefined).

```
\def\ensurefunction#1#2{%
      \ifx #1\undefined\ccls@deffunction{#1}{#2}
      \else
         \ccls@InfoFuncEns{#1}
      \fi}
```

`\forcefunction` \forcefunction defines (if undefined) or redefines (if defined) the new function #1.

```
\def\forcefunction#1#2{%
      \ccls@deffunction{#1}{#2}}
```

`\ccls@deffunction` The private \ccls@deffunction command makes the real work. The new functions will have three arguments: ##1, a number, ##2, the value of the new function in that number, and ##3, the derivative.

```
\def\ccls@deffunction#1#2{%
    \def#1##1##2##3{%
            \begingroup
            \def\t{##1}%
                 #2
                \xdef##2{\y}%
                \xdef##3{\Dy}%
            \endgroup}\ignorespaces}
```

New polar functions

`\newpolarfunction` The \newpolarfunction{#1}{#2} instruction defines a new polar function called #1. #2 is the list of instructions to calculate the radius \r and his derivative \Dr from the \t arc variable.

```
\def\newpolarfunction#1#2{%
      \ifx #1\undefined
        \ccls@defpolarfunction{#1}{#2}
      \else
         \ccls@ErrorPFuncDef{#1}
      \fi}
```

`\renewpolarfunction` \renewpolarfunction redefines #1 if already defined.

```
\def\renewpolarfunction#1#2{%
```

```
        \ifx #1\undefined
           \ccls@ErrorPFuncUnDef{#1}
        \else
           \ccls@defpolarfunction{#1}{#2}
        \fi}
```

\ensurepolarfunction

`\ensurepolarfunction` defines (only if undefined) #1.

```
\def\ensurepolarfunction#1#2{%
        \ifx #1\undefined\ccls@defpolarfunction{#1}{#2}
        \else
           \ccls@InfoPFuncEns{#1}
        \fi}
```

\forcepolarfunction

`\forcepolarfunction` defines (if undefined) or redefines (if defined) #1.

```
\def\forcepolarfunction#1#2{%
        \ccls@defpolarfunction{#1}{#2}}
```

\ccls@defpolarfunction

The private `\ccls@defpolarfunction` command makes the real work. The new functions will have three arguments: ##1, a number (the polar radius), ##2, ##3, ##4, and ##5, the x and y component functions and its derivatives at ##1.

```
\def\ccls@defpolarfunction#1#2{%
        \def#1##1##2##3##4##5{%
        \begingroup
          \def\t{##1}
        #2
        \COS{\t}\ccls@cost
        \MULTIPLY\r\ccls@cost{\x}
        \SIN{\t}\ccls@sint
        \MULTIPLY\r\ccls@sint{\y}
        \MULTIPLY\ccls@cost\Dr\Dx
        \SUBTRACT{\Dx}{\y}{\Dx}
        \MULTIPLY\ccls@sint\Dr\Dy
        \ADD{\Dy}{\x}{\Dy}
        \xdef##2{\x}
        \xdef##3{\Dx}
        \xdef##4{\y}
        \xdef##5{\Dy}
        \endgroup}\ignorespaces}
```

New vector functions

\newvectorfunction

The `\newvectorfunction{#1}{#2}` instruction defines a new vector (parametric) function called #1. #2 is the list of instructions to calculate `\x`, `\y`, `\Dx` and `\Dy` from the `\t` arc variable.

```
\def\newvectorfunction#1#2{%
        \ifx #1\undefined
          \ccls@defvectorfunction{#1}{#2}
        \else
           \ccls@ErrorVFuncDef{#1}
        \fi}
```

\renewvectorfunction

\renewvectorfunction redefines #1 if already defined.

```
\def\renewvectorfunction#1#2{%
       \ifx #1\undefined
          \ccls@ErrorVFuncUnDef{#1}
       \else
          \ccls@defvectorfunction{#1}{#2}
       \fi}
```

\ensurevectorfunction

\ensurevectorfunction defines (only if undefined) #1.

```
\def\ensurevectorfunction#1#2{%
       \ifx #1\undefined\ccls@defvectorfunction{#1}{#2}
       \else
          \ccls@InfoVFuncEns{#1}
       \fi}
```

\forcevectorfunction

\forcevectorfunction defines (if undefined) or redefines (if defined) #1.

```
\def\forcevectorfunction#1#2{%
       \ccls@defvectorfunction{#1}{#2}}
```

\ccls@defvectorfunction

The private \ccls@defvectorfunction command makes the real work. The new functions will have three arguments: ##1, a number, ##2, ##3, ##4, and ##5, the x and y component functions and its derivatives at ##1.

```
\def\ccls@defvectorfunction#1#2{%
       \def#1##1##2##3##4##5{%
       \begingroup
         \def\t{##1}
       #2
       \xdef##2{\x}
       \xdef##3{\Dx}
       \xdef##4{\y}
       \xdef##5{\Dy}
       \endgroup}\ignorespaces}
```

13.3 Polynomials

Linear (first degreee) polynomials

\newlpoly

The \newlpoly{#1}{#2}{#3} instruction defines the linear polynomial
$\#1 = \#2 + \#3t$.

```
\def\newlpoly#1#2#3{%
    \newfunction{#1}{%
        \ccls@lpoly{#2}{#3}}}
```

\renewlpoly

We define also the \renewlpoly, \ensurelpoly and \forcelpoly variants.

```
\def\renewlpoly#1#2#3{%
    \renewfunction{#1}{%
        \ccls@lpoly{#2}{#3}}}
```

\ensurelpoly

```
\def\ensurelpoly#1#2#3{%
    \ensurefunction{#1}{%
        \ccls@lpoly{#2}{#3}}}
```

\forcelpoly

```
\def\forcelpoly#1#2#3{%
    \forcefunction{#1}{%
        \ccls@lpoly{#2}{#3}}}
```

\ccls@lpoly The \ccls@lpoly{#1}{#2} macro defines the new polynomial function.

```
\def\ccls@lpoly#1#2{%
        \MULTIPLY{#2}{\t}{\y}
        \ADD{\y}{#1}{\y}
        \COPY{#2}{\Dy}}
```

Quadratic polynomials

\newqpoly The \newqpoly{#1}{#2}{#3}{#4} instruction defines the quadratic polynomial $\#1 = \#2 + \#3t + \#4t^2$.

```
\def\newqpoly#1#2#3#4{%
    \newfunction{#1}{%
        \ccls@qpoly{#2}{#3}{#4}}}
```

\renewqpoly

```
\def\renewqpoly#1#2#3#4{%
    \renewfunction{#1}{%
        \ccls@qpoly{#2}{#3}{#4}}}
```

\ensureqpoly

```
\def\ensureqpoly#1#2#3#4{%
    \ensurefunction{#1}{%
        \ccls@qpoly{#2}{#3}{#4}}}
```

\forceqpoly

```
\def\forceqpoly#1#2#3#4{%
    \forcefunction{#1}{%
        \ccls@qpoly{#2}{#3}{#4}}}
```

\ccls@qpoly The \ccls@qpoly{#1}{#2} macro defines the new polynomial function.

```
\def\ccls@qpoly#1#2#3{%
        \MULTIPLY{\t}{#3}{\y}
           \MULTIPLY{2}{\y}{\Dy}
           \ADD{#2}{\Dy}{\Dy}
        \ADD{#2}{\y}{\y}
        \MULTIPLY{\t}{\y}{\y}
        \ADD{#1}{\y}{\y}}
```

Cubic polynomials

\newcpoly The `\newcpoly{#1}{#2}{#3}{#4}{#5}` instruction defines the cubic polynomial
$\#1 = \#2 + \#3t + \#4t^2 + \#5t^3$.

```
\def\newcpoly#1#2#3#4#5{%
    \newfunction{#1}{%
        \ccls@cpoly{#2}{#3}{#4}{#5}}}
```

\renewcpoly

```
\def\renewcpoly#1#2#3#4#5{%
    \renewfunction{#1}{%
        \ccls@cpoly{#2}{#3}{#4}{#5}}}
```

\ensurecpoly

```
\def\ensurecpoly#1#2#3#4#5{%
    \ensurefunction{#1}{%
        \ccls@cpoly{#2}{#3}{#4}{#5}}}
```

\forcecpoly

```
\def\forcecpoly#1#2#3#4#5{%
    \forcefunction{#1}{%
        \ccls@cpoly{#2}{#3}{#4}{#5}}}
```

\ccls@cpoly The `\ccls@cpoly{#1}{#2}` macro defines the new polynomial function.

```
\def\ccls@cpoly#1#2#3#4{%
        \MULTIPLY{\t}{#4}{\y}
           \MULTIPLY{3}{\y}{\Dy}
        \ADD{#3}{\y}{\y}
           \MULTIPLY{2}{#3}{\ccls@temp}
           \ADD{\ccls@temp}{\Dy}{\Dy}
        \MULTIPLY{\t}{\y}{\y}
           \MULTIPLY{\t}{\Dy}{\Dy}
        \ADD{#2}{\y}{\y}
           \ADD{#2}{\Dy}{\Dy}
        \MULTIPLY{\t}{\y}{\y}
        \ADD{#1}{\y}{\y}
        }
```

13.4 Elementary functions

\ONEfunction The `\ONEfunction`: $y(t) = 1$, $y'(t) = 0$

```
\newfunction{\ONEfunction}{%
      \COPY{1}{\y}
      \COPY{0}{\Dy}}
```

\ZEROfunction The `\ZEROfunction`: $y(t) = 0$, $y'(t) = 0$

```
\newfunction{\ZEROfunction}{%
      \COPY{0}{\y}
      \COPY{0}{\Dy}}
```

`\IDENTITYfunction` The `\IDENTITYfunction`: $y(t) = t$, $y'(t) = 1$

```
\newfunction{\IDENTITYfunction}{%
      \COPY{\t}{\y}
      \COPY{1}{\Dy}}
```

`\RECIPROCALfunction` The `\RECIPROCALfunction`: $y(t) = 1/t$, $y'(t) = -1/t^2$

```
\newfunction{\RECIPROCALfunction}{%
      \DIVIDE{1}{\t}{\y}
      \SQUARE{\y}{\Dy}
      \MULTIPLY{-1}{\Dy}{\Dy}}
```

`\SQUAREfunction` The `\SQUAREfunction`: $y(t) = t^2$, $y'(t) = 2t$

```
\newfunction{\SQUAREfunction}{%
      \SQUARE{\t}{\y}
      \MULTIPLY{2}{\t}{\Dy}}
```

`\CUBEfunction` The `\CUBEfunction`: $y(t) = t^3$, $y'(t) = 3t^2$

```
\newfunction{\CUBEfunction}{%
      \SQUARE{\t}{\Dy}
      \MULTIPLY{\t}{\Dy}{\y}
      \MULTIPLY{3}{\Dy}{\Dy}}
```

`\SQRTfunction` The `\SQRTfunction`: $y(t) = \sqrt{t}$, $y'(t) = 1/(2\sqrt{t})$

```
\newfunction{\SQRTfunction}{%
      \SQRT{\t}{\y}
      \DIVIDE{0.5}{\y}{\Dy}}
```

`\EXPfunction` The `\EXPfunction`: $y(t) = \exp t$, $y'(t) = \exp t$

```
\newfunction{\EXPfunction}{%
     \EXP{\t}{\y}
     \COPY{\y}{\Dy}}
```

`\COSfunction` The `\COSfunction`: $y(t) = \cos t$, $y'(t) = -\sin t$

```
\newfunction{\COSfunction}{%
     \COS{\t}{\y}
     \SIN{\t}{\Dy}
     \MULTIPLY{-1}{\Dy}{\Dy}}
```

`\SINfunction` The `\SINfunction`: $y(t) = \sin t$, $y'(t) = \cos t$

```
\newfunction{\SINfunction}{%
     \SIN{\t}{\y}
     \COS{\t}{\Dy}}
```

`\TANfunction` The `\TANfunction`: $y(t) = \tan t$, $y'(t) = 1/(\cos t)^2$

```
\newfunction{\TANfunction}{%
     \TAN{\t}{\y}
     \COS{\t}{\Dy}
     \SQUARE{\Dy}{\Dy}
     \DIVIDE{1}{\Dy}{\Dy}}
```

\COTfunction The \COTfunction: $y(t) = \cot t$, $y'(t) = -1/(\sin t)^2$

```
\newfunction{\COTfunction}{%
      \COTAN{\t}{\y}
      \SIN{\t}{\Dy}
      \SQUARE{\Dy}{\Dy}
      \DIVIDE{-1}{\Dy}{\Dy}}
```

\COSHfunction The \COSHfunction: $y(t) = \cosh t$, $y'(t) = \sinh t$

```
\newfunction{\COSHfunction}{%
      \COSH{\t}{\y}
      \SINH{\t}{\Dy}}
```

\SINHfunction The \SINHfunction: $y(t) = \sinh t$, $y'(t) = \cosh t$

```
\newfunction{\SINHfunction}{%
      \SINH{\t}{\y}
      \COSH{\t}{\Dy}}
```

\TANHfunction The \TANHfunction: $y(t) = \tanh t$, $y'(t) = 1/(\cosh t)^2$

```
\newfunction{\TANHfunction}{%
      \TANH{\t}{\y}
      \COSH{\t}{\Dy}
      \SQUARE{\Dy}{\Dy}
      \DIVIDE{1}{\Dy}{\Dy}}
```

\COTHfunction The \COTHfunction: $y(t) = \coth t$, $y'(t) = -1/(\sinh t)^2$

```
\newfunction{\COTHfunction}{%
      \COTANH{\t}{\y}
      \SINH{\t}{\Dy}
      \SQUARE{\Dy}{\Dy}
      \DIVIDE{-1}{\Dy}{\Dy}}
```

\LOGfunction The \LOGfunction: $y(t) = \log t$, $y'(t) = 1/t$

```
\newfunction{\LOGfunction}{%
      \LOG{\t}{\y}
      \DIVIDE{1}{\t}{\Dy}}
```

\HEAVISIDEfunction The \HEAVISIDEfunction: $y(t) = \begin{cases} 0 & \text{if } t < 0 \\ 1 & \text{if } t \geq 0 \end{cases}$, $y'(t) = 0$

```
\newfunction{\HEAVISIDEfunction}{%
      \ifdim \t\p@<\z@ \COPY{0}{\y}\else\COPY{1}{\y}\fi
       \COPY{0}{\Dy}}
```

13.5 Operations with functions

\CONSTANTfunction \CONSTANTfunction defines #2 as the constant function $f(t) = \#1$.

```
\def\CONSTANTfunction#1#2{%
              \def#2##1##2##3{%
                          \xdef##2{#1}%
                          \xdef##3{0}}}
```

\SUMfunction

`\SUMfunction` defines #3 as the sum of functions #1 and #2.

```
\def\SUMfunction#1#2#3{%
            \def#3##1##2##3{%
             \begingroup
                    #1{##1}{\ccls@SUMf}{\ccls@SUMDf}%
                    #2{##1}{\ccls@SUMg}{\ccls@SUMDg}%
                    \ADD{\ccls@SUMf}{\ccls@SUMg}{\ccls@SUMfg}
                    \ADD{\ccls@SUMDf}{\ccls@SUMDg}{\ccls@SUMDfg}
                         \xdef##2{\ccls@SUMfg}%
                         \xdef##3{\ccls@SUMDfg}%
             \endgroup}\ignorespaces}
```

\SUBTRACTfunction

`\SUBTRACTfunction` defines #3 as the difference of functions #1 and #2.

```
\def\SUBTRACTfunction#1#2#3{%
            \def#3##1##2##3{%
             \begingroup
                    #1{##1}{\ccls@SUBf}{\ccls@SUBDf}%
                    #2{##1}{\ccls@SUBg}{\ccls@SUBDg}%
                    \SUBTRACT{\ccls@SUBf}{\ccls@SUBg}{\ccls@SUBfg}
                    \SUBTRACT{\ccls@SUBDf}{\ccls@SUBDg}{\ccls@SUBDfg}
                         \xdef##2{\ccls@SUBfg}%
                         \xdef##3{\ccls@SUBDfg}%
             \endgroup}\ignorespaces}
```

\PRODUCTfunction

`\PRODUCTfunction` defines #3 as the product of functions #1 and #2.

```
\def\PRODUCTfunction#1#2#3{%
            \def#3##1##2##3{%
             \begingroup
                    #1{##1}{\ccls@PROf}{\ccls@PRODf}%
                    #2{##1}{\ccls@PROg}{\ccls@PRODg}%
                    \MULTIPLY{\ccls@PROf}{\ccls@PROg}{\ccls@PROfg}
                    \MULTIPLY{\ccls@PROf}{\ccls@PRODg}{\ccls@PROfDg}
                    \MULTIPLY{\ccls@PRODf}{\ccls@PROg}{\ccls@PRODfg}
                    \ADD{\ccls@PROfDg}{\ccls@PRODfg}{\ccls@PRODfg}
                         \xdef##2{\ccls@PROfg}%
                         \xdef##3{\ccls@PRODfg}%
             \endgroup}\ignorespaces}
```

\QUOTIENTfunction

`\QUOTIENTfunction` defines #3 as the quotient of functions #1 and #2.

```
\def\QUOTIENTfunction#1#2#3{%
            \def#3##1##2##3{%
             \begingroup
                    #1{##1}{\ccls@QUOf}{\ccls@QUODf}%
                    #2{##1}{\ccls@QUOg}{\ccls@QUODg}%
                    \DIVIDE{\ccls@QUOf}{\ccls@QUOg}{\ccls@QUOfg}
                    \MULTIPLY{\ccls@QUOf}{\ccls@QUODg}{\ccls@QUOfDg}
                    \MULTIPLY{\ccls@QUODf}{\ccls@QUOg}{\ccls@QUODfg}
                    \SUBTRACT{\ccls@QUODfg}{\ccls@QUOfDg}{\ccls@QUOnum}
                    \SQUARE{\ccls@QUOg}{\ccls@qsquaretempg}
                    \DIVIDE{\ccls@QUOnum}{\ccls@qsquaretempg}{\ccls@QUODfg}
```

```
                        \xdef##2{\ccls@QUOfg}%
                        \xdef##3{\ccls@QUODfg}%
            \endgroup}\ignorespaces}
```

\COMPOSITIONfunction

\COMPOSITIONfunction defines #3 as the composition of functions #1 and #2.

```
\def\COMPOSITIONfunction#1#2#3{%  #3=#1(#2)
            \def#3##1##2##3{%
             \begingroup
                   #2{##1}{\ccls@COMg}{\ccls@COMDg}%
                   #1{\ccls@COMg}{\ccls@COMf}{\ccls@COMDf}%
                   \MULTIPLY{\ccls@COMDg}{\ccls@COMDf}{\ccls@COMDf}
                        \xdef##2{\ccls@COMf}%
                        \xdef##3{\ccls@COMDf}%
             \endgroup}\ignorespaces}
```

\SCALEfunction

\SCALEfunction defines #3 as the product of number #1 and function #2.

```
\def\SCALEfunction#1#2#3{%
            \def#3##1##2##3{%
             \begingroup
                   #2{##1}{\ccls@SCFf}{\ccls@SCFDf}%
                   \MULTIPLY{#1}{\ccls@SCFf}{\ccls@SCFaf}
                   \MULTIPLY{#1}{\ccls@SCFDf}{\ccls@SCFDaf}
                        \xdef##2{\ccls@SCFaf}%
                        \xdef##3{\ccls@SCFDaf}%
             \endgroup}\ignorespaces}
```

\SCALEVARIABLEfunction

\SCALEVARIABLEfunction scales the variable by number #1 and aplies function #2.

```
\def\SCALEVARIABLEfunction#1#2#3{%
            \def#3##1##2##3{%
             \begingroup%
                   \MULTIPLY{#1}{##1}{\ccls@SCVat}
                   #2{\ccls@SCVat}{\ccls@SCVf}{\ccls@SCVDf}%
                   \MULTIPLY{#1}{\ccls@SCVDf}{\ccls@SCVDf}
                        \xdef##2{\ccls@SCVf}%
                        \xdef##3{\ccls@SCVDf}%
             \endgroup}\ignorespaces}
```

\POWERfunction

\POWERfunction defines #3 as the power of function #1 to exponent #2.

```
\def\POWERfunction#1#2#3{%
            \def#3##1##2##3{%
             \begingroup
                   #1{##1}{\ccls@POWf}{\ccls@POWDf}%
                   \POWER{\ccls@POWf}{#2}{\ccls@POWfn}
                  \SUBTRACT{#2}{1}{\ccls@nminusone}
                  \POWER{\ccls@POWf}{\ccls@nminusone}{\ccls@POWDfn}
                 \MULTIPLY{#2}{\ccls@POWDfn}{\ccls@POWDfn}
                 \MULTIPLY{\ccls@POWDfn}{\ccls@POWDf}{\ccls@POWDfn}
                        \xdef##2{\ccls@POWfn}%
                        \xdef##3{\ccls@POWDfn}%
             \endgroup}\ignorespaces}
```

\LINEARCOMBINATIONfunction

`\LINEARCOMBINATIONfunction` defines the new function #5 as the linear combination #1#2+#3#4. #1 and #3 are two numbers. #1 and #3 are two functions.

```
\def\LINEARCOMBINATIONfunction#1#2#3#4#5{%
            \def#5##1##2##3{%
             \begingroup
                    #2{##1}{\ccls@LINf}{\ccls@LINDf}%
                    #4{##1}{\ccls@LINg}{\ccls@LINDg}%
                    \MULTIPLY{#1}{\ccls@LINf}{\ccls@LINf}
                    \MULTIPLY{#3}{\ccls@LINg}{\ccls@LINg}
                    \MULTIPLY{#1}{\ccls@LINDf}{\ccls@LINDf}
                    \MULTIPLY{#3}{\ccls@LINDg}{\ccls@LINDg}
                    \ADD{\ccls@LINf}{\ccls@LINg}{\ccls@LINafbg}
                    \ADD{\ccls@LINDf}{\ccls@LINDg}{\ccls@LINDafbg}
                         \xdef##2{\ccls@LINafbg}%
                         \xdef##3{\ccls@LINDafbg}%
             \endgroup}\ignorespaces}
```

\POLARfunction

`\POLARfunction` defines the polar curve #2. #1 is a previously defined function.

```
\def\POLARfunction#1#2{%
       \PRODUCTfunction{#1}{\COSfunction}{\ccls@polarx}
       \PRODUCTfunction{#1}{\SINfunction}{\ccls@polary}
       \PARAMETRICfunction{\ccls@polarx}{\ccls@polary}{#2}}
```

\PARAMETRICfunction

`\PARAMETRICfunction` defines the parametric curve #3. #1 and #2 are the components functions (two previuosly defined functions).

```
\def\PARAMETRICfunction#1#2#3{%
       \def#3##1##2##3##4##5{%
                #1{##1}{##2}{##3}
                #2{##1}{##4}{##5}}}
```

\VECTORfunction

`\VECTORfunction`: an alias of `\PARAMETRICfunction`.

```
\let\VECTORfunction\PARAMETRICfunction
```

```
% </calculus>
```

Change History

v1.0
  General: First public version 1

v1.0a
  General: calculator.dtx modified to make it autoinstallable. calculus.dtx embedded in calculus.dtx 1

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.