Automatic typesetting of formulas using computer algebra

Marcelo Castier and Vladimir F. Cabral

Abstract

This paper describes new procedures, written in the Mathematica® programming language, for quickly typesetting mathematical formulas in LaTeX syntax. Two main procedures provide direct interface with the user. The first of them obtains the LaTeX representation of a single formula. The second procedure analyzes a set of formulas, searching for common terms and symmetries, and breaks the original formulas input by the user in a series of calculations of intermediate terms. In either case, a list of symbols used in the formulas is automatically generated in LaTeX format. The procedures may speed up the writing of technical publications and eliminate common sources of error in their preparation.

Introduction

Several current tools can assist preparation of technical documents using computers. Voice recognition software such as ViaVoice™ and Dragon NaturallySpeaking® transform speech directly into typeset text with good accuracy. Literature references can be downloaded from databases, and software such as Natbib, Reference Manager®, and ProCite® will format them according to the rules of many scientific journals. Cross-referencing of tables, equations, and figures eliminates the need for their manual renumbering, if the manuscript has to be modified. However, authors usually typeset mathematical formulas manually, which can be tedious and time-consuming due to the need for careful reviews of complex expressions.

In fact, it may be more difficult to guarantee the correctness of a formula typeset for publication than its programmed version in a scientific language, such as Fortran or C. In the latter case, numerical tests can help locate programming errors. On the other hand, the verification of formulas typeset for publication is generally made by visual inspection.

An additional aspect related to typesetting formulas for publication is the preparation of lists of symbols. It is not unusual to find publications in which some symbols are missing from these lists.

Modern computer algebra systems (CAS), such as Maple™ and Mathematica, provide a user-friendly environment for symbolic computations, allowing the derivation of complex formulas. Moreover, both Maple and Mathematica have commands for exporting formulas to other programs in different formats. Therefore, they have the basic functionality needed for the automatic implementation of formulas, which has been used by some authors.

Motivated by the difficulty of manual symbolic computations in the area of general relativity, Klioner (1998) used Mathematica to develop a program for operations with indexed objects that can provide its results in TeX or LaTeX. Piecuch (1993) and Strange et al. (2001) used Maple to obtain LaTeX code in applications to problems in quantum chemistry. Maple was also used by Sharf (1996) for the generation of LaTeX code in the analysis of beam elements for the simulation of multibody systems. Weinzierl (2004) describes a new CAS, called gTybalt, which is freely available and has the possibility of producing TeX output. However, some operations, such as integration, are not implemented yet, which currently limits the applicability of the program. Talole and Pradke (2003) developed a program that exports text, numerical data, and plots from a Matlab® calculation to a LaTeX document. An important contribution in this area is the development of Mathscape (Barnett, 1998), which is a program in Mathematica for the automatic typesetting of formulas in LaTeX whose features are in many ways complementary to those available in the set of procedures presented here.

In this paper, we use Mathematica, and the comments henceforth about the ability to export formulas are limited to this CAS. Mathematica can export formulas as images, or in MathML or TeX formats. The use of images is inconvenient because some final editing of the formulas is often required. The MathML or TeX codes cannot be used as input in the current versions of Microsoft® Equation Editor or MathType™, which are the most commonly used equation editors for Microsoft Word. Therefore, the exchange of formulas between Mathematica and these editors that have graphical user interfaces is less flexible than would be desirable.

Use of MathML will probably spread in the future as a way of interchanging information about physical properties and models for their evaluation (Frenkel et al., 2004). However, we focus on the use of TeX or LaTeX directly, because the latter is the de facto standard used internally by many technical publishers. Mathematica has a command to generate the representation of a formula in TeX. Although very useful, this command only takes one formula at a time and does not generate the list of symbols, among other limitations.

Here, we present two new procedures. The first generates LaTeX code for a single formula. The second procedure performs a simultaneous analysis of several formulas, identifies their common and symmetrical terms, and obtains the LaTeX representation as a sequence of intermediate formulas, thereby breaking the original expressions input by the user into a form that is more convenient for presentation. Both procedures automatically prepare a list of symbols, also coded in LaTeX, classifying them as Roman or Greek letters, or indexes.

These new procedures extend the capabilities of Thermath (Castier, 1999), a program whose current version contains approximately 6000 lines of code written in the Mathematica programming language. The original purpose of Thermath was the computer implementation of thermodynamic models for the calculation of physicochemical properties of mixtures, by providing complete subroutines automatically written in a scientific programming language. In a typical application, given a thermodynamic model, several properties are derived using computer algebra for operations such as derivation and integration in a Mathematica session. It often happens that the derived properties have formulas that are longer and more complex than the thermodynamic model that originates them. Using its internal procedures, Thermath analyzes these formulas and implements them automatically in a complete subroutine, with a drastic reduction in the need for manual coding.

Thermath has been extended to other applications such as the automatic implementation of expressions (Dominguez et al., 2002) in a format compatible with the INTBIS/INTLIB package for solving sets of nonlinear equations with interval analysis (Kearfott and Novoa, 1990), and the preparation of code for the simulation of separation equipment in the chemical industry (Alfradique et al., 2002).

The new procedures presented in this paper add the possibility of aiding in the preparation of technical documents, not only related to physicochemical properties but in many areas that require the manual typesetting of long formulas. These procedures perform extensive and intricate symbol manipulations in the expressions. Here, we present only a general description, and refer to the code, which is available from the authors on request, for all the details.

Automatic generation of LaTeX code for a single formula

Mathematica has a function called `TeXForm` that translates formulas into TeX syntax. Let us illustrate its usage with the typesetting of a simple formula: the van der Waals equation of state. Given that the emphasis here is not on the technical aspects of the equation of state, we refrain from discussing the meaning of its symbols. The `TeXForm` function is used as follows:

```
TeXForm[P == R*T/(v - b) - a/v^2]
```

This command produces one line of output, broken here in additional lines only to fit the column width of *TUGboat, as also done in some of the other examples of this paper.*

```
P = -\left( \frac{a}{v^2} \right)  +
 \frac{R\,T}{-b + v}
```

This output, obtained in a Mathematica session, can be cut and pasted into a document. Even though this certainly reduces the need for manual typesetting, several improvements are possible, such as automatically assigning a label to the formula for cross-referencing and generating a list of symbols.

Thermath contains a procedure, `prinTeX`, that performs several actions: (1) prepares lines that load the LaTeX breqn package for the automatic breaking of long formulas in several lines; (2) prepares an equation label containing six randomly generated digits; (3) identifies all the symbols that appear in the equation, classifying them as Roman or Greek letters or indexes; (4) implements the formula using the Mathematica function `TeXForm`. The verbatim input in Mathematica is:

```
prinTeX[P == R*T/(v - b) - a/v^2]
```

The verbatim LaTeX code obtained as output is:

```
%
%The following lines should be placed after the
%\documentclass {class} line
%in the LATEX file.
%
\usepackage[cmbase]{flexisym}
\usesymbols{msabm}
\usepackage[debug]{breqn}
\setkeys{breqn}{compact}
%
%The following lines should be placed where
%the formula should appear in the text.
%
\begin{dmath}\label{e:eqn485282}
P = -\left( \frac{a}{v^2} \right)  +
\frac{R\,T}{-b + v}
\end{dmath}

%
%The following lines create the list of symbols.
%
\section*{List of Symbols}
%
%

\subsection*{Roman Letters}
%
```

```
$a$          \\
$b$          \\
$P$          \\
$R$          \\
$T$          \\
$v$          \\
%
%The list of symbols was successfully created.
```

The parts of this output, i.e., the loading commands for breqn, the formula, and the list of symbols can then be cut and pasted at their proper positions in a LaTeX document. For example, the loading commands for breqn were pasted at the beginning of the LaTeX document of this paper for *TUGboat*. The formula and the list of symbols were pasted here. Due to differences between the `ltugboat` document class used by *TUGboat* and the `elsart` document class from Elsevier, used for testing the software, it was necessary to manually add a `\newline` command in the line after the subsection names to improve formatting. Upon processing with the LaTeX compiler, the following text is obtained:

$$P = -\left(\frac{a}{v^2}\right) + \frac{R\,T}{-b+v} \tag{1}$$

List of Symbols

Roman Letters

a
b
P
R
T
v

If several formulas are prepared using `prinTeX`, the loading commands for breqn need to be pasted only once at the beginning of the LaTeX document and the lists of symbols of each formula have to be manually combined to consolidate the list of symbols of the document, which most commonly constitutes one of the final sections of technical papers.

Even though Equation 1 is correct, this example also illustrates one of the difficulties with CAS. Comparing the input and output, we observe that Mathematica interchanges the order of the two terms in the right hand side of the equation and does the same in the denominator $(v-b)$. Therefore, the formula is not printed as usually represented in the literature. Unfortunately, there seems to be no straightforward solution to this problem in Mathematica. Barnett (1998) developed a function called `toEach`, in the context of Mathscape, that can reverse the order of operations, but this function was not tested here. Instead, we circumvented the problem by using the command `HoldForm`, which keeps an expression unevaluated and therefore not subject to the automatic reordering of terms performed by Mathematica. The corresponding input is:

```
prinTeX[HoldForm[P == (R*T)/(v - b) - a/v^2]]
```

After processing this input with the LaTeX compiler, the traditional representation of the van der Waals equation of state is obtained:

$$P = \frac{R\,T}{v-b} - \frac{a}{v^2} \tag{2}$$

The list of symbols remains unchanged and, for this reason, is not presented.

Let us consider a more complex example, which requires integration of the van der Waals equation of state at constant temperature. The Mathematica input is:

```
P = (R*T)/(v - b) - a/v^2
prinTeX[W == HoldForm[Integrate[P,
{v,alpha, beta}]] ==
Simplify[Integrate[P, {v, alpha, beta}]]]
```

The output is:

$$\begin{aligned} W = \int_{\alpha}^{\beta} P\,dv = a\left(-\left(\frac{1}{\alpha}\right) + \frac{1}{\beta}\right) \\ + R\,T\,\ln\left(\frac{b-\beta}{-\alpha+b}\right) \end{aligned} \tag{3}$$

List of Symbols

Roman Letters

a
b
R
T
W

Greek Letters

α
β

Note that the command `HoldForm` leaves the integral unevaluated between the two equal signs. The list of symbols now contains a subsection where the two Greek letters used as integration limits are identified. A current limitation of the pattern matching procedure implemented in `prinTeX` is that it does not identify dummy variables. For instance, v is a dummy integration variable, and it is not included in the list of symbols.

Automatic generation of LaTeX code for multiple formulas

In many cases, several formulas are derived using computer algebra during a Mathematica session, and instead of generating LaTeX code for each formula, it may be more convenient to generate code for all of them simultaneously. For this, we developed two procedures that are used sequentially: ordeqTeX and createTeX.

Procedure ordeqTeX analyzes the expressions to be represented in LaTeX. During this analysis, subexpressions that appear several times are recursively identified and ordered, in such a way that a meaningful calculation sequence of subexpressions is obtained. The procedure also searches for subexpressions with symmetrical indexes. In addition, ordeqTeX can sort the subexpressions according to their dependence with respect to a list of variables input by the user, which may be useful for authors writing about the functional structure of their formulas.

Procedure ordeqTeX is similar to a procedure already present in the first version of Thermath, ordeq, whose logical analysis of expressions is discussed by Castier (1999). An important difference between them is the level of fragmentation into subexpressions. Consider, for instance, that $^1/_x$ is a subexpression that appears several times in a large formula. For automatic programming in a numerical language, such as Fortran or C, it is convenient to store the result of the subexpression in an intermediate variable, in order to avoid unnecessary calculations. However, a large number of simple substitutions may obscure the presentation of a formula in a text. For this purpose, the formulas should be less fragmented than for numerical calculations — but to what extent is a subjective decision.

In ordeqTeX, simple fractions such as the example in the above discussion, powers in which the exponent is a number, multiplications and sums of only two terms are not replaced by intermediate variables. However, the pattern matching algorithm implemented in procedure ordeqTeX can be easily changed to use other criteria.

Procedure ordeqTeX prepares detailed information about the structure of the formulas and of the subexpressions, which is then passed to procedure createTeX. This procedure prepares a LaTeX code that presents all subexpressions and final expressions in a feasible computation sequence.

Even though createTeX replaces long formulas by sequences of subexpressions, it may happen that some of these subexpressions are longer than one line of LaTeX output. In order to avoid the need for manual intervention for breaking long lines, we used the (freely available) LaTeX package breqn, which automatically chooses the breakpoints. For convenience, the output of the prinTeX and createTeX procedures includes commands for loading and using breqn, and each formula is given a unique label for cross-referencing. In the case of prinTeX, a six-digit random number is used to generate the label. In the case of createTeX, the number results from joining the name of the set of formulas being implemented, specified by the user, with a unique sequential number assigned to each subexpression.

For the preparation of the list of symbols, we use the fact that expressions are internally stored as trees in Mathematica. Using a recursive procedure developed for Thermath, the trees are spanned, searching for all the symbols they contain. From this first list of symbols, those that represent intrinsic Mathematica functions or operators, such as Plus, Times, Log, Exp, etc., are discarded.

To distinguish between intrinsic Mathematica functions and symbols entered by the user, we use the Mathematica function Attributes to test each symbol. Intrinsic Mathematica functions have non-empty lists of attributes, whereas a symbol entered by the user has an empty list of attributes, unless a special attribute has been explicitly assigned to the symbol. It is usually unnecessary to specify attributes to symbols, but it may happen, for example, that some matrices are intrinsically symmetrical. In these cases, it is convenient to assign the Mathematica attribute Orderless to the symbols that represent these values. Therefore, the symbols entered by users are located as those without attribute or only with the Orderless attribute.

From the remaining list, the symbols that represent numerical values, either integer, real or complex, are also discarded. At this point, the list will only have the symbols entered by the user. It then remains to verify which of the symbols are variables and which are only indexes. The convention adopted in the identification procedure is that a symbol is an index when it is the argument of a symbol entered by the user. For instance, in the expression Sin(x(i)), x is the argument of an intrinsic Mathematica function, Sin, and therefore is not an index. On the other hand, i is the argument of x, which is not an intrinsic Mathematica function. Therefore, i is assumed to be an index.

As an example, let us consider the simultaneous analysis of two simple formulas, with some characteristics that help illustrate the features of the package presented here. The input for this example is:

```
f = Sin[x[i]*x[j]] + Cos[y[k]*y[m]];
g = Cos[x[i]*x[j]] + Exp[Sin[x[i]*x[j]]];
formulas = {f, g};

analyzedformulas = ordeqTeX[formulas, {}];
createTeX[fg, analyzedformulas,
  {HoldForm[f], HoldForm[g]}];
```

In this input, the two formulas `f` and `g`, are joined in a single list, `formulas`, which is passed to `ordeqTeX`. The second argument of this call specifies how subexpressions should be grouped according to their functional dependence. In this example, an empty list is specified, meaning that no specific grouping is required.

In procedure `createTeX`, the first argument, `fg`, represents a user-defined name for the set of formulas being implemented. `createTeX` uses this name to prepare a unique label for each subexpression to be used for cross-referencing. The second argument is a list containing several pieces of information about the formulas prepared by procedure `ordeqTeX`. The last argument specifies that the left-hand side of the equations should appear as $f =$ and $g =$. The LaTeX output (slightly edited) is:

```
%
%Formulas for model: fg
%
%
%     if (
%         green(1)   .or.
%         green(2)
%        ) then
%

\smallskip

\begin{dmath}\label{e:fgeqn1}
      w_{2}\left(i,j \right)=
      \sin \left(x\left(i \right)\,
      x\left(j \right) \right)
\end{dmath}
%
%Note symmetry: w$2(j,i)=w$2(i,j)
%=============
%
%
%     end if
%
%
%     if (
%         green(1)
%        ) then
%

\smallskip

\begin{dmath}\label{e:fgeqn2}
      f\left(i,j,k,m \right)=
      \cos \left(y\left(k \right)\,
      y\left(m \right) \right) +
      w_{2}\left(i,j \right)
\end{dmath}
%
%Note symmetry: f(i,j,m,k)=f(i,j,k,m)
%=============
%
%
%Note symmetry: f(j,i,k,m)=f(i,j,k,m)
%=============
%
%
%Note symmetry: f(j,i,m,k)=f(i,j,k,m)
%=============
%
%
%     end if
%
%
%     if (
%         green(2)
%        ) then
%

\smallskip

\begin{dmath}\label{e:fgeqn3}
      g\left(i,j \right)=
      e^{w_{2}\left(i,j \right)} +
      \cos \left(x\left(i \right)\,
      x\left(j \right) \right)
\end{dmath}
%
%Note symmetry: g(j,i)=g(i,j)
%=============
%
%
%     end if
%
%
%The set of formulas was successfully created.
%
%The following lines create the list of symbols.
%
\section*{List of Symbols}
%
%

\subsection*{Roman Letters}
%
$f$          \\
[...]
%

\subsection*{Indexes}
%
$i$          \\
[...]
%
%The list of symbols was successfully created.
```

Note that the LaTeX output contains a variable of the form w_n that is automatically generated to represent an intermediate value. This output also contains several comments that aim at helping authors to discuss the structure of their formulas, should this be desired. The parts of the output flagged with `green(1)` are relevant for the calculation of the first output variable, f, whereas `green(2)` provides a flag for the calculation of g. The output also indicates the existence of symmetry. For instance, it indicates that variables w_2 and g are symmetrical with respect to permutations of the indexes i and j, and that variable f is symmetrical with respect to some permutations of its indexes.

Compilation with LaTeX produces the following output:

$$w_2\left(i,j\right) = \sin\left(x\left(i\right)\ x\left(j\right)\right) \tag{4}$$

$$f\left(i,j,k,m\right) = \cos\left(y\left(k\right)\ y\left(m\right)\right) + w_2\left(i,j\right) \tag{5}$$

$$g\left(i,j\right) = e^{w_2(i,j)} + \cos\left(x\left(i\right)\ x\left(j\right)\right) \tag{6}$$

List of Symbols

Roman Letters

f
g
w_n
x
y

Indexes

i
j
k
m

Note that each variable appearing on the left hand side of Equations 4, 5, and 6 received the correct indexes automatically and that all the variables used in the formulas were included in the list of symbols. The exponential and cosine functions appear in reverse positions in the output compared to the input, as a result of the automatic reordering of expressions performed by Mathematica. Unlike the `prinTeX` command that was designed handle a single formula, the typical use of commands `ordeqTeX` and `createTeX` is in Mathematica sessions where several formulas are derived using computer algebra. In this context, the user has less control of the ordering used by Mathematica to present the formulas. Therefore, even though the formulas are correctly translated into LaTeX, a current limitation is that the formulas may need to be manually edited if some specific order of terms is desired in the LaTeX document.

We successfully tested the procedures discussed here with sets of formulas that are much more complex than those used in these examples. In some cases, especially when there are rather long formulas, a final manual editing step may be necessary to improve layout; the `[layout=RHS]` option of the breqn package proved particularly useful in these cases.

Conclusions

This paper presented new procedures, written in the Mathematica programming language, that automatically generate a representation of formulas in LaTeX with the corresponding list of symbols. There is the option of generating LaTeX code for a single formula or for a set of formulas. In the latter case, a comprehensive analysis of the formula structures allows the identification of common and symmetrical terms. Therefore, if one uses Mathematica as a computational environment for the symbolic and numerical calculations in a given project, it is possible to quickly obtain an exact representation, in LaTeX, of the formulas used and the list of symbols. The procedures may speed up the writing of technical publications and eliminate common sources of error in their preparation.

Acknowledgments

The authors thank Profs. Veronica M.A. Calado and Frederico W. Tavares (Universidade Federal do Rio de Janeiro, Brazil) for their suggestions. The Brazilian agencies CNPq and FAPERJ provided financial support for this research.

Code availability

The procedures developed in this work are available from the authors on request. The procedures were developed and tested using Mathematica 4.1, version 0.94 of the LaTeX package breqn, and Elsevier document classes.

References

Alfradique, M. F., R. O. Espósito, and M. Castier. "Automatic generation of procedures for the simulation of multistage separators using computer algebra". *Chemical Engineering Communications* **189**(5), 657–674, 2002.

Barnett, M. P. "Mathscape — Combining Mathematica and TeX". *TUGboat* **19**(2), 147–156, 1998.

Castier, M. "Automatic implementation of thermodynamic models using computer algebra". *Computers and Chemical Engineering* **23**(9), 1229–1245, 1999.

Dominguez, A., J. Tojo, and M. Castier. "Automatic implementation of thermodynamic models for reliable parameter estimation using computer algebra". *Computers and Chemical Engineering* **26**(10), 1473–1479, 2002.

Frenkel, M., R. D. Chirico, V. V. Oiky, K. N. Marsh, J. H. Dymond, and W. A. Wakeham. "ThermoML — An XML-based approach for storage and exchange of experimental and critically evaluate thermophysical and thermochemical property data. 3. Critically evaluated data, predicted data, and equation representation". *Journal of Chemical and Engineering Data* **49**(3), 381–393, 2004.

Kearfott, R. B. and M. Novoa. "INTBIS, A Portable Interval Newton Bisection Package". *ACM Transactions on Mathematical Software* **16**(2), 152–157, 1990.

Klioner, S. A. "New system for indicial computation and its applications in gravitational physics". *Computer Physics Communications* **115**(2-3), 231–244, 1998.

Piecuch, P. "Maple Symbolic Computation of the Long-Range Many-Body Intermolecular Potentials — 3-Body Induction Forces Between 2 Atoms and A Linear Molecule". *International Journal of Quantum Chemistry* **47**(4), 261–305, 1993.

Sharf, I. "Geometrically non-linear beam element for dynamics simulation of multibody systems". *International Journal for Numerical Methods in Engineering* **39**(5), 763–786, 1996.

Strange, R., F. R. Manby, and P. J. Knowles. "Automatic code generation in density functional theory". *Computer Physics Communications* **136**(3), 310–318, 2001.

Talole, S. E. and S. B. Pradke. "Generating LaTeX documents through Matlab". *TUGboat* **24**(2), 245–248, 2003.

Weinzierl, S. "gTybalt — A free computer algebra system". *Computer Physics Communications* **156**(2), 180–198, 2004.

◇ Marcelo Castier
Escola de Química, Universidade Federal do Rio de Janeiro
Rio de Janeiro, RJ, 21949-900
Brazil
`castier@eq.ufrj.br`

◇ Vladimir F. Cabral
Departamento de Engenharia Química, Universidade Estadual de Maringá
Maringá, PR, 87020-900
Brazil
`vfcabral@yahoo.com.br`