The Polynom Package

Copyright 2000–2004
Carsten Heinz < `cheinz@gmx.de` >

2004/08/12 Version 0.16

Abstract

The polynom package implements macros for manipulating polynomials. For example, it can typeset polynomial long divisions and synthetic divisions (Horner's scheme), which can be shown step by step. The main test case and application is the polynomial ring in one variable with rational coefficients. *Please note that this is work in progress. Multivariate polynomials are* currently *not supported.*

1 Introduction

Donald Arseneau has contributed a lot of packages to the TEX community. In particular, he posted macros for long division on `comp.text.tex`, which were also published in the TUGboat [1] and eventually as `longdiv.tex` on CTAN. The polynom package allows to do the job with polynomials, see figure 1. There you can also see an example of Horner's scheme for synthetic division.

$$
\begin{array}{rrrrr}
 & & & \multicolumn{2}{r}{X^2+2X+2} \\ \cline{2-5}
X-1) & X^3 & +X^2 & & -1 \\
 & -X^3 & +X^2 & & \\ \cline{2-3}
 & & 2X^2 & & \\
 & & -2X^2 & +2X & \\ \cline{3-4}
 & & & 2X & -1 \\
 & & & -2X & +2 \\ \cline{4-5}
 & & & & 1
\end{array}
\qquad
\begin{array}{c|cccc}
 & 1 & 1 & 0 & -1 \\
1 & & 1 & 2 & 2 \\ \hline
 & 1 & 2 & 2 & 1
\end{array}
$$

`\polylongdiv{X^3+X^2-1}{X-1}`   `\polyhornerscheme[x=1]{x^3+x^2-1}`

Figure 1: Polynomial long division and synthetic division. The commands both are able to generate partial output, see polydemo.pdf in fullscreen mode.

Figures 2 and 3 show applications of polynomial division. On the one hand the Euclidean algorithm to determine a greatest common divisor of two polynomials,

$$\begin{aligned} X^4-2X^3+2X^2-2X+1 &= \left(X^3+X^2-X-1\right)\cdot\left(X-3\right)+\left(6X^2-4X-2\right)\\ X^3+X^2-X-1 &= \left(6X^2-4X-2\right)\cdot\left(\tfrac{1}{6}X+\tfrac{5}{18}\right)+\left(\tfrac{4}{9}X-\tfrac{4}{9}\right)\\ 6X^2-4X-2 &= \left(\tfrac{4}{9}X-\tfrac{4}{9}\right)\cdot\left(\tfrac{27}{2}X+\tfrac{9}{2}\right)+0 \end{aligned}$$

```
\polylonggcd {(X-1)(X-1)(X^2+1)} {(X-1)(X+1)(X+1)}
```

Figure 2: Euclidean algorithm with polynomials; the last nonzero remainder is a greatest common divisor. In the case here, it is uniquely determined up to a scalar factor, so $X-1$ and $\frac{4}{9}X-\frac{4}{9}$ are both greatest common divisors

```
\polyfactorize {(X-1)(X-1)(X^2+1)}
```

$$\left(X^2+1\right)\left(X-1\right)^2$$

```
\polyfactorize {2X^3+X^2-7X+3}
```

$$2\left(X-\tfrac{1}{2}\right)\left(X+\tfrac{1}{2}+\frac{\sqrt{13}}{2}\right)\left(X+\tfrac{1}{2}-\frac{\sqrt{13}}{2}\right)$$

```
\polyfactorize {120X^5-274X^4+225X^3-85X^2+15X-1}
```

$$120\left(X-1\right)\left(X-\tfrac{1}{2}\right)\left(X-\tfrac{1}{3}\right)\left(X-\tfrac{1}{4}\right)\left(X-\tfrac{1}{5}\right)$$

Figure 3: Factorizations of some polynomials

and on the other the factorization of a polynomial with at most two nonrational zeros. This should suffice for many teaching aids.

2 Hints

As the examples show, the commands get their data through mandatory and optional arguments. Polynomials are entered as you would type them in math mode:[1] you may use `+`, `-`, `*`, `\cdot`, `/`, `\frac`, `(`, `)`, natural numbers, symbols like `e`, `\pi`, `\chi`, `\lambda`, and variables; the power operator `^` with integer exponents can be used on symbols, variables, and parenthesized expressions. Never use variables in a nominator, denominator or divisor.

The support of symbols is very limited and there is neither support of functions like $\sin(x)$ or $\exp(x)$, nor of roots or exponents other than integers, for example $\sqrt{\pi}$ or e^x. For teaching purposes this shouldn't be a major drawback. Particularly because there is a simple workaround in some cases: the package doesn't look at symbols closely, so define a function like e^x or 'composed symbol' like $\sqrt{\pi}$ as a symbol. Take a look at figure 4 for an example.

Optional arguments are used to specify more general options (and also for the evaluation point for Horner's scheme). The options are entered in key=value fashion using the keyval package [3]. The available options are listed in the respective

[1]The scanner is based on the scanner of the `calc` package [2]. Read its documentation and the implementation part here if you want to know more.

$$
\begin{array}{l}
\bigl(\quad e^x x^3 - e^x x^2 + e^x x - e^x\bigr) \,/\, \bigl(x-1\bigr) = e^x x^2 + e^x \\
\underline{\;-\,e^x x^3 + e^x x^2\;} \\
\qquad\qquad\qquad\; e^x x - e^x \\
\qquad\qquad\quad \underline{-\,e^x x + e^x} \\
\qquad\qquad\qquad\qquad\quad 0
\end{array}
$$

```
\newcommand\epowerx{e^x}
\[\polylongdiv{\epowerx x^3-\epowerx x^2+\epowerx x-\epowerx}{x-1}\]
```

Figure 4: Avoiding problems with e^x. Be particularly careful in such cases. *You* have to take care of the correct result *since* the package does the computation. And by the way, it's always good to keep an eye on plausibility of the results

sections below.

3 Commands

3.1 `\polyset{`⟨*key=value list*⟩`}`

Keys and values in optional arguments affect only that particular operation. `\polyset` changes the settings for the rest of the current environment or group. This could be a single figure or the whole document. Almost every key described in this manual is allowed — just try it and you'll see. Table 5 lists all keys, which are not connected to a particular command. An example is

```
\polyset{vars=XYZ\xi,  % make X, Y, Z, and \xi into variables
         delims={[}{]}}% nongrowing brackets
```

Note that is essential to use `vars`-declared variables only. The package can't guess your intention and `\polylongdiv{\zeta^3+\zeta^2-1}{\zeta-1}` would divide a constant by a constant without the information ζ being a variable.

`vars=`⟨*token string*⟩	make each token a variable
`delims={`⟨*left*⟩`}{`⟨*right*⟩`}`	define delimiters used for printing parenthesized expressions

Table 5: General keys. Default for `vars` is `Xx`. The key `delims` has in fact an optional argument which takes invisible versions of the left and right delimiter. The default is `delims=[{\left.}{\right.}]{\left(}{\right)}`

3.2 `\polylongdiv[`⟨*key=value list*⟩`]`⟨*polynomial a*⟩⟨*polynomial b*⟩

The command prints the polynomial long division of a/b. Applicable keys are listed in table 6. Of course, `vars` and `delims` can be used, too.

`stage=`⟨*number*⟩	print long division up to stage ⟨*number*⟩ (starting with 1)
`style=A`|`B`|`C`	define output scheme for long division, refer polydemo.pdf
`div=`⟨*token*⟩	define division sign for `style=C`, default is $\div$

Table 6: Keys and values for polynomial long division. `style=A` requires `stage=`$3\times$ (#quotient's summands)$\,+\,1$ to be carried out fully. The other styles `B` and `C` need one more stage if the remainder is nonzero

3.3 `\polyhornerscheme[`⟨*key=value list*⟩`]`⟨*polynomial*⟩

The command prints Horner's scheme for the given polynomial with respect to the specified evaluation point. Note that the latter one is entered as a key=value pair in the form ⟨*variable*⟩`=`⟨*value*⟩. Table 7 lists other keys and their respective values.

3.4 `\polylonggcd[`⟨*key=value list*⟩`]`⟨*polynomial a*⟩⟨*polynomial b*⟩

The command prints equations of the Euclidean algorithm used to determine the greatest common divisor of the polynomials a and b, refer figure 2.

3.5 `\polyfactorize[`⟨*key=value list*⟩`]`⟨*polynomial*⟩

The command prints a factorization of the polynomial as long as all except two roots are rational, see figure 3.

3.6 Low-level commands

To tell the whole truth, the commands above don't need the polynomials typed in verbatim. The internal representation of polynomials can be stored as replacement texts of control sequences and such control sequences can take the role of verbatim polynomials. This is also the case for ⟨*a*⟩ and ⟨*b*⟩ in table 8, but each ⟨$cs_{\ldots}$⟩ must be a control sequence, in which the result is saved.

The command in table 8 can be used for low level calculations, and in particular to store polynomials for later use with the high-level commands. For example one could write the following.

⟨*variable*⟩`=`⟨*value*⟩	The definition of the evaluation point is *mandatory*!
`stage=`⟨*number*⟩	print Horner's scheme up to stage ⟨*number*⟩ (starting with 1)
`tutor=true|false`	turn on and off tutorial comments
`tutorlimit=`⟨*number*⟩	illustrate the recent ⟨*number*⟩ steps
`tutorstyle=`⟨*font selection*⟩	define appearance of tutorial comments
`resultstyle=`⟨*font selection*⟩	define appearance of the result
`resultleftrule=true|false`	control rules left to, right to, and at the
`resultrightrule=true|false`	bottom of the result
`resultbottomrule=true|false`	
`showbase=false|top|middle|bottom`	define whether and in which row the base (the value) is printed
`showvar=true|false`	print or suppress the variable name (additionally to the base)
`showbasesep=true|false`	print or suppress the vertical rule
`equalcolwidth=true|false`	use the same width for all columns or use their individual widths
`arraycolsep=`⟨*dimension*⟩	space between columns
`arrayrowsep=`⟨*dimension*⟩	space between rows
`showmiddlerow=true|false`	print or suppress the middle row

Table 7: Keys and values for Horner's scheme. Don't use `showmiddlerow=false` with `tutor=true`.

```
\polyadd\polya {(X^2+X+1)(X-1)-\frac\pi2}{0}% trick
\polymul\polyb {X-1}{1}              % another trick
Let’s see how to divide \polyprint\polya{} by \polyprint\polyb.
  \[\polylongdiv\polya\polyb\]
```

$\langle cs_{a+b}\rangle \leftarrow a+b$	`\polyadd`$\langle cs_{a+b}\rangle\langle a\rangle\langle b\rangle$
$\langle cs_{a-b}\rangle \leftarrow a-b$	`\polysub`$\langle cs_{a-b}\rangle\langle a\rangle\langle b\rangle$
$\langle cs_{ab}\rangle \leftarrow a\cdot b$	`\polymul`$\langle cs_{ab}\rangle\langle a\rangle\langle b\rangle$
$\langle cs_{a/b}\rangle \leftarrow \lfloor a/b\rfloor$ `\polyremainder` $\leftarrow a \bmod b$	`\polydiv`$\langle cs_{a/b}\rangle\langle a\rangle\langle b\rangle$
$\langle cs_{\mathrm{gcd}}\rangle \leftarrow \gcd(a,b)$	`\polygcd`$\langle cs_{\mathrm{gcd}}\rangle\langle a\rangle\langle b\rangle$
print polynomial a	`\polyprint`$\langle a\rangle$

Table 8: Low-level user commands

4 Acknowledgments

I wish to thank Ludger Humbert, Karl Heinz Marbaise, and Elke Niedermair for their tests and error reports.

References

[1] Barbara Beeton and Donald Arseneau.
Long division.
In Jeremy Gibbons’ *Hey — it works!*, TUGboat 18(2), June 1997, p. 75.

[2] Kresten Krab Thorup, Frank Jensen, and Chris Rowley.
The `calc` package, Infix notation arithmetic in LaTeX, 1998/07/07.
Available from CTAN: `macros/latex/required/tools`.

[3] David Carlisle.
The keyval package, 1999/03/16.
Available from CTAN: `macros/latex/required/graphics`.