polexpr root localization examples

Jean-François Burnol

To access the reference documentation:
`texdoc` polexpr.html

The package provides a parser `\poldef` of algebraic polynomial expressions.

Once defined, a polynomial is usable by its name either as a numerical function in `\xintexpr/\xinteval`, or for additional polynomial definitions, or as argument to the package macros.

This document illustrates root localization via usage of macros such as `\PolToSturm` and `\PolSturmIsolateZeros` which implement the Sturm theorem:

- Root localization based on Sturm theorem was added at release `0.4` (2018/02/16).
- Ability to find all rational roots was added at release `0.7.2` (2018/12/09).

As of `0.8` (2021/03/29), polexpr is usable with Plain TEX and not only with LATEX. The examples here use most of the time a syntax which works with both.

Copying-pasting from `pdf` the example source may lose formatting. Formerly, they were included verbatim in the `html` documentation. Here they are both rendered verbatim and got executed during the LATEX run which created this `pdf` file, with the output shown after the source code.

Regarding how polynomial coefficients are printed on the typeset page by `\PolTypeset`:

- The default for `\PolTypesetOne` is to use `\xintTeXsignedFrac` with LATEX, `\xintTeXsignedOver` with Plain. See the xintexpr documentation for a description of what these macros do. A sensible definition is:
  ```
  \def\PolTypesetOne#1{\PolDecToString{\xintREZ{#1}}}%
  ```
  It means to use decimal notation, with perhaps a trailing denominator if the argument is a fraction, and will suppress trailing zeros after the decimal mark.
- As these are expandable macros, they are usable to redefine `\PolToExprCmd` as well:
  ```
  \def\PolToExprCmd#1{\PolDecToString{\xintREZ{#1}}}%
  ```
  This will customize the output of `\PolToExpr` (which a priori is destined for writes to external files but may also be used on the typeset page).

With `\xintverbosetrue` in the TEX source extra information relative to the internal data manipulated by the macros will be written to the `.log` file.

> Package macros related to root localization create (user-level) new polynomials, or numeric variables, via a naming scheme using the given `<sturmname>` as prefix. It is thus advisable to keep this `<sturmname>` name-space separate from the one used to name polynomial or scalar variables.

Regrettably all examples here use the condemnable `\PolToSturm{f}{f}` practice which means that internally defined polynomials will use as prefix the original polynomial name. This merge of namespaces may cause overwriting previously defined data and may lead to hard-to-debug problems.

1 A first example

In this example the polynomial is square-free.

```
  \poldef f(x) := x^7 - x^6 - 2x + 1;
%
  \PolToSturm{f}{f}
  \PolSturmIsolateZeros{f}
  The \PolTypeset{f} polynomial has \PolSturmNbOfIsolatedZeros{f} distinct real
  roots which are located in the following intervals:
  \PolPrintIntervals{f}
```

The $x^7 - x^6 - 2x + 1$ polynomial has 3 distinct real roots which are located in the following intervals:

$$\begin{array}{rcccl} -2 & < & Z_1 & < & -1 \\ 0.4 & < & Z_2 & < & 0.5 \\ 1 & < & Z_3 & < & 2 \end{array}$$

```
  Here is the second root with ten more decimal digits:
  \PolRefineInterval[10]{f}{2}
  $$\PolSturmIsolatedZeroLeft{f}{2}<Z_2<\PolSturmIsolatedZeroRight{f}{2}$$
```

Here is the second root with ten more decimal digits:

$$0.49623869487 < Z_2 < 0.49623869488$$

```
  And here is the first root with twenty digits after decimal mark:
  \PolEnsureIntervalLength{f}{1}{-20}
  $$\PolSturmIsolatedZeroLeft{f}{1}<Z_1<\PolSturmIsolatedZeroRight{f}{1}$$
```

And here is the first root with twenty digits after decimal mark:

$$-1.07196788410802660342 < Z_1 < -1.07196788410802660341$$

```
  The first element of the Sturm chain has degree $\PolDegree{f_0}$. As
  this is the original degreee $\PolDegree{f}$ we know that $f$ is square free.
  Its derivative is up to a constant \PolTypeset{f_1} (in this example
  it is identical with it).
  \PolToSturm{f_1}{f_1}\PolSturmIsolateZeros{f_1}%
  The derivative has \PolSturmNbOfIsolatedZeros{f_1} distinct real
  roots:
  \PolPrintIntervals[W]{f_1}%
  \PolEnsureIntervalLengths{f_1}{-10}%
  Here they are with ten digits after decimal mark:
  \PolPrintIntervals[W]{f_1}
```

The first element of the Sturm chain has degree 7. As this is the original degreee 7 we know that f is square free. Its derivative is up to a constant $7x^6 - 6x^5 - 2$ (in this example it is identical with it). The

derivative has 2 distinct real roots:

$$\begin{array}{rcccl} -0.8 & < & W_1 & < & -0.7 \\ 1.06 & < & W_2 & < & 1.07 \end{array}$$

Here they are with ten digits after decimal mark:

$$\begin{array}{rcccl} -0.7113598308 & < & W_1 & < & -0.7113598307 \\ 1.0653438848 & < & W_2 & < & 1.0653438849 \end{array}$$

```
\PolDiff{f_1}{f''}
\PolToSturm{f''}{f''}
\PolSturmIsolateZeros{f''}
The second derivative is \PolTypeset{f''}.
It has \PolSturmNbOfIsolatedZeros{f''} distinct real
roots:
\PolPrintIntervals[X]{f''}%
Here is the positive one with 20 digits after decimal mark:
\PolEnsureIntervalLength{f''}{2}{-20}%
$$X_2 = \PolSturmIsolatedZeroLeft{f''}{2}\dots$$
```

The second derivative is $42x^5 - 30x^4$. It has 2 distinct real roots:

$$\begin{array}{rcccl} & & X_1 & = & 0 \\ 0.7 & < & X_2 & < & 0.8 \end{array}$$

Here is the positive one with 20 digits after decimal mark:

$$X_2 = 0.71428571428571428571\ldots$$

2 A degree four polynomial with nearby roots

Notice that this example is a bit outdated as `0.7` release has added `\PolSturmIsolateZeros**{<sturmname>}` which would find exactly the roots. The steps here retain their interest when one is interested in finding isolating intervals for example to prepare some demonstration of dichotomy method.

```
\PolDef{Q}{(x-1.050001)(x-1.105001)(x-1.110501)(x-1.111051)}
\PolTypeset{Q}
\PolToSturm{Q}{Q} % it is allowed to use same prefix for Sturm chain
\PolSturmIsolateZeros{Q}
\PolPrintIntervals{Q}
```

$x^4-4376554\cdot 10^{-6}x^3+7181524404656\cdot 10^{-12}x^2-5236452059410679654\cdot 10^{-18}x+1431546080701127890651551\cdot 10^{-24}$

$$\begin{array}{rcccl} 1.05 & < & Z_1 & < & 1.06 \\ 1.10 & < & Z_2 & < & 1.11 \\ 1.110 & < & Z_3 & < & 1.111 \\ 1.111 & < & Z_4 & < & 1.112 \end{array}$$

```
\PolRefineInterval*{Q}{1}
\PolRefineInterval*{Q}{2}
\PolRefineInterval*{Q}{3}
\PolRefineInterval*{Q}{4}
```

```
\PolPrintIntervals{Q}
```

$$\begin{array}{rcccl} & & Z_1 & = & 1.050001 \\ 1.105 & < & Z_2 & < & 1.106 \\ 1.1105 & < & Z_3 & < & 1.1106 \\ 1.11105 & < & Z_4 & < & 1.11106 \end{array}$$

```
\PolEnsureIntervalLengths{Q}{-6}
\PolPrintIntervals{Q}
% finds here all roots exactly
```

$$\begin{array}{rcl} Z_1 & = & 1.050001 \\ Z_2 & = & 1.105001 \\ Z_3 & = & 1.110501 \\ Z_4 & = & 1.111051 \end{array}$$

3 The degree nine polynomial with 0.99, 0.999, 0.9999 as triple roots

Define a user command (xinttools is loaded automatically by polexpr):

```
\def\showmultiplicities#1{% #1 = "sturmname"
\xintFor* ##1 in {\xintSeq{1}{\PolSturmNbOfIsolatedZeros{#1}}}\do{%
    The multiplicity is \PolSturmIsolatedZeroMultiplicity{#1}{##1}
    \PolSturmIfZeroExactlyKnown{#1}{##1}%
    {at the root $x=\PolSturmIsolatedZeroLeft{#1}{##1}$}
    {for the root such that
    $\PolSturmIsolatedZeroLeft{#1}{##1}<x<\PolSturmIsolatedZeroRight{#1}{##1}$}
    \par
}}%
\PolDef{f}{(x-0.99)^3(x-0.999)^3(x-0.9999)^3}
\def\PolTypesetOne#1{\PolDecToString{\xintREZ{#1}}}
\PolTypeset{f}\par
```

$$x^9-8.9667x^8+35.73400293x^7-83.070418400109x^6+124.143648875193123x^5-123.683070924326075877x^4+82.1492603975530756178 91x^3-35.07602992699900159127007x^2+8.736407873331464836867 1733x-0.967100824643585986488103299$$

```
\PolToSturm{f}{f}% it is allowed to use "polname" as "sturmname" too
\PolSturmIsolateZerosAndGetMultiplicities{f}% use the "sturmname" here
% or \PolSturmIsolateZeros*{f} which is exactly the same, but shorter..

\showmultiplicities{f}
```

The multiplicity is 3 at the root $x = 0.99$
The multiplicity is 3 at the root $x = 0.999$
The multiplicity is 3 at the root $x = 0.9999$

It is fun to modify only a tiny bit the polynomial and see if polexpr survives:

```
\PolDef{g}{f(x)+1e-27}
\PolTypeset{g}\par
\PolToSturm{g}{g}
\PolSturmIsolateZeros*{g}

\showmultiplicities{g}
```

$$x^9-8.9667x^8+35.73400293x^7-83.070418400109x^6+124.143648875193123x^5-123.683070924326075877x^4+82.1492603975530756178 91x^3-35.07602992699900159127007x^2+8.736407873331464836867 1733x-0.967100824643585986488103298$$

The multiplicity is 1 for the root such that $0.98 < x < 0.99$
The multiplicity is 1 for the root such that $0.9991 < x < 0.9992$
The multiplicity is 1 for the root such that $0.9997 < x < 0.9998$

This means that the multiplicity-3 roots each became a real and a pair of complex ones. Let's see them better:

```
\PolEnsureIntervalLengths{g}{-10}

\showmultiplicities{g}
```

The multiplicity is 1 for the root such that $0.9899888032 < x < 0.9899888033$
The multiplicity is 1 for the root such that $0.9991447980 < x < 0.9991447981$
The multiplicity is 1 for the root such that $0.9997663986 < x < 0.9997663987$

4 A degree five polynomial with three rational roots

```
\poldef Q(x) :=  1581755751184441 x^5
               -14907697165025339 x^4
               +48415668972339336 x^3
               -63952057791306264 x^2
               +46833913221154895 x
               -49044360626280925;

\PolToSturm{Q}{Q}
  \def\PolTypesetCmdPrefix#1{\allowbreak\xintiiifSgn{#1}{}{+}{+}}%
  $Q_0(x) = \PolTypeset{Q_0}$
\PolSturmIsolateZeros**{Q}
\PolPrintIntervals{Q}

$Q_{norr}(x) = \PolTypeset{Q_norr}$
```

$Q_0(x) = 1581755751184441x^5 - 14907697165025339x^4 + 48415668972339336x^3 - 63952057791306264x^2 + 46833913221154895x - 49044360626280925$

$$\begin{aligned} Z_1 &= 833719/265381 \\ Z_2 &= 165707065/52746197 \\ Z_3 &= 355/113 \end{aligned}$$

$Q_{norr}(x) = x^2 + 1$

Here, all real roots are rational. Let's get their decimal expansion too:

```
\begingroup
  % print decimal expansion of the found roots
  \def\PolPrintIntervalsPrintExactZero
              {\xintTrunc{20}{\PolPrintIntervalsTheLeftEndPoint}\dots}
  \PolPrintIntervals{Q}
\endgroup % we localized the modified \PolPrintIntervalsPrintExactZero
```

$$\begin{aligned} Z_1 &= 3.14159265358107777120\ldots \\ Z_2 &= 3.14159265358979340254\ldots \\ Z_3 &= 3.14159292035398230088\ldots \end{aligned}$$

5 A Mignotte type polynomial

```
\PolDef{P}{x^10 - (10x-1)^2}%
```

```
  \PolTypeset{P}              % prints it in expanded form
  \PolToSturm{P}{P}           % we can use same prefix for Sturm chain
  \PolSturmIsolateZeros{P}    % finds 4 real roots
  This polynomial has \PolSturmNbOfIsolatedZeros{P} distinct real roots:
  \PolPrintIntervals{P}%
```

$x^{10} - 100x^2 + 20x - 1$ This polynomial has 4 distinct real roots:

$$\begin{aligned} -2 &< Z_1 < -1\\ 0.09 &< Z_2 < 0.10\\ 0.1 &< Z_3 < 0.2\\ 1 &< Z_4 < 2 \end{aligned}$$

Let us refine the second and third intervals to separate the corresponding roots:

```
  \PolRefineInterval*{P}{2}% will refine to 0.0999990 < Z_2 < 0.0999991
  \PolRefineInterval*{P}{3}% will refine to 0.100001 < Z_3 < 0.100002
  \PolPrintIntervals{P}%
```

$$\begin{aligned} -2 &< Z_1 < -1\\ 0.0999990 &< Z_2 < 0.0999991\\ 0.100001 &< Z_3 < 0.100002\\ 1 &< Z_4 < 2 \end{aligned}$$

Let us now get to know all roots with 10 digits after decimal mark:

```
  \PolEnsureIntervalLengths{P}{-10}%
  \PolPrintIntervals{P}% now all roots are known 10 decimal digits after mark
```

$$\begin{aligned} -1.8024470510 &< Z_1 < -1.8024470509\\ 0.0999990000 &< Z_2 < 0.0999990001\\ 0.1000010000 &< Z_3 < 0.1000010001\\ 1.7523477264 &< Z_4 < 1.7523477265 \end{aligned}$$

Finally, we display 20 digits of the second root:

```
  \PolEnsureIntervalLength{P}{2}{-20}% makes Z_2 known with 20 digits after mark
  $$\PolSturmIsolatedZeroLeft{P}{2}<Z_2<\PolSturmIsolatedZeroRight{P}{2}$$
```

$$0.09999900004999650028 < Z_2 < 0.09999900004999650029$$

6 The Wilkinson polynomial

See `https://en.wikipedia.org/wiki/Wilkinson%27s_polynomial`.

```
  %\xintverbosetrue % for the curious...

  \poldef f(x) := mul((x - i), i = 1..20);

  \def\PolTypesetCmdPrefix#1{\allowbreak\xintiiifSgn{#1}{}{+}{+}}%
  \def\PolTypesetOne#1{\xintDecToString{#1}}%

  \noindent\PolTypeset{f}

  \PolToSturm{f}{f}
  \PolSturmIsolateZeros{f}
  \PolPrintIntervals{f}
  % \vfill\eject
```

```
% This page is commented out because it takes about 30s on a 2GHz CPU
% \poldef g(x) := f(x) - 2**{-23} x**19;

% \PolToSturm{g}{g}
% \noindent\PolTypeset{g_0}% integer coefficient primitive polynomial

% \PolSturmIsolateZeros{g}
% \PolEnsureIntervalLengths{g}{-10}

% \PolPrintIntervals*{g}
```

$$x^{20}-210x^{19}+20615x^{18}-1256850x^{17}+53327946x^{16}-1672280820x^{15}+40171771630x^{14}-756111184500x^{13}$$
$$+11310276995381x^{12}-135585182899530x^{11}+1307535010540395x^{10}-10142299865511450x^{9}+63030812099294896x^{8}$$
$$-311333643161390640x^{7}+1206647803780373360x^{6}-3599979517947607200x^{5}+8037811822645051776x^{4}$$
$$-12870931245150988800x^{3}+13803759753640704000x^{2}-8752948036761600000x+2432902008176640000$$

$$\begin{aligned}
Z_1 &= 1\\
Z_2 &= 2\\
Z_3 &= 3\\
Z_4 &= 4\\
Z_5 &= 5\\
Z_6 &= 6\\
Z_7 &= 7\\
Z_8 &= 8\\
Z_9 &= 9\\
Z_{10} &= 10\\
Z_{11} &= 11\\
Z_{12} &= 12\\
Z_{13} &= 13\\
Z_{14} &= 14\\
Z_{15} &= 15\\
Z_{16} &= 16\\
Z_{17} &= 17\\
Z_{18} &= 18\\
Z_{19} &= 19\\
Z_{20} &= 20
\end{aligned}$$

The first polynomial is handled fast enough, but the modified one `f(x) - 2**-23 x**19` takes about 20x longer.

Its Sturm chain polynomials have integer coefficients with up to 321 digits, whereas (surprisingly perhaps) those of the Sturm chain polynomials derived from `f` never have more than 21 digits ...

Once the Sturm chain is computed and the zeros isolated, obtaining their decimal digits is relatively faster. Here are the ten real roots of `f(x) - 2**-23 x**19` which would be computed by the commented-out code above:

```
Z_1 = 0.9999999999...
Z_2 = 2.0000000000...
Z_3 = 2.9999999999...
Z_4 = 4.0000000002...
Z_5 = 4.9999999275...
Z_6 = 6.0000069439...
Z_7 = 6.9996972339...
Z_8 = 8.0072676034...
Z_9 = 8.9172502485...
```

```
Z_10 = 20.8469081014...
```

7 The second Wilkinson polynomial

```
\poldef f(x) := mul(x - 2^-i, i = 1..20);

%\PolTypeset{f}

\PolToSturm{f}{f}
\PolSturmIsolateZeros**{f}
\PolPrintIntervals{f}
```

$$
\begin{aligned}
Z_{1} &= 0.00000095367431640625\\
Z_{2} &= 0.0000019073486328125\\
Z_{3} &= 0.000003814697265625\\
Z_{4} &= 0.00000762939453125\\
Z_{5} &= 0.0000152587890625\\
Z_{6} &= 0.000030517578125\\
Z_{7} &= 0.00006103515625\\
Z_{8} &= 0.0001220703125\\
Z_{9} &= 1/4096\\
Z_{10} &= 1/2048\\
Z_{11} &= 1/1024\\
Z_{12} &= 1/512\\
Z_{13} &= 1/256\\
Z_{14} &= 1/128\\
Z_{15} &= 0.015625\\
Z_{16} &= 0.03125\\
Z_{17} &= 0.0625\\
Z_{18} &= 0.125\\
Z_{19} &= 0.25\\
Z_{20} &= 0.5
\end{aligned}
$$

This takes more time than the polynomial with 1, 2, .., 20 as roots but less than the latter modified by the `2**-23` tiny change to one of its coefficient.

There is some incoherence in output format which has its source in the fact that some roots are found in branches which can only find decimal roots, whereas some are found in branches which could find general fractions and they use `\xintIrr` before storage of the found root. This may evolve in future.

8 The degree 41 polynomial with -2, -1.9, -1.8, ..., 0, 0.1, ..., 1.9, 2 as roots

```
\PolDef{P}{mul((x-i*1e-1), i=-20..20)}% i/10 is same but less efficient
```

In the defining expression we could have used `i/10` but this gives less efficient internal form for the coefficients (the `10`'s end up in denominators).

```
\begingroup
  \def\PolToExprCmd#1{\PolDecToString{\xintREZ{#1}}}
  \def\PolToExprTermPrefix#1{\newline\xintiiifSgn{#1}{}{+}{+}}
  \def\PolToExprTimes{${}\cdot{}$}
```

```
\ttfamily
\PolToExpr{P}
\endgroup
```
```
x^41
-28.7·x^39
+375.7117·x^37
-2975.11006·x^35
+15935.28150578·x^33
-61167.527674162·x^31
+173944.259366417394·x^29
-373686.963560544648·x^27
+613012.0665016658846445·x^25
-771182.31133138163125495·x^23
+743263.86672885754888959569·x^21
-545609.076599482896371978698·x^19
+301748.325708943677229642930528·x^17
-123655.89876694504346988698445 44·x^15
+36666.178205488400585560820586 4192·x^13
-7607.858213674594456495183800161 28·x^11
+1053.151359186872985088859502237941 76·x^9
-90.6380005918141132650786081964032·x^7
+4.337015638473273668425522182881 28·x^5
-0.094477096842080473549817826508 8·x^3
+0.000591901218138992768541744168 96·x
```

which shows coefficients with up to 36 significant digits...

Stress test: not a hard challenge to xint + polexpr, but be a bit patient!

```
  \PolDef{P}{mul((x-i*1e-1), i=-20..20)}%
  \PolToSturm{P}{S}           % dutifully computes S_0, ..., S_{41}
  % the [1] optional argument limits the search to interval (-10,10)
  \PolSturmIsolateZeros[1]{S} % finds *exactly* (but a bit slowly) all 41 roots!
  \PolPrintIntervals{S}       % nice, isn't it?
% Unfortunately \PolPrintIntervals uses a non-breakable array environment
% But see next section on how to customize \PolPrintIntervals and let it
% allow pagebreaks
```

$$
\begin{aligned}
Z_{1} &= -2\\
Z_{2} &= -1.9\\
Z_{3} &= -1.8\\
Z_{4} &= -1.7\\
Z_{5} &= -1.6\\
Z_{6} &= -1.5\\
Z_{7} &= -1.4\\
Z_{8} &= -1.3\\
Z_{9} &= -1.2\\
Z_{10} &= -1.1\\
Z_{11} &= -1\\
Z_{12} &= -0.9\\
Z_{13} &= -0.8\\
Z_{14} &= -0.7\\
Z_{15} &= -0.6\\
Z_{16} &= -0.5\\
Z_{17} &= -0.4\\
Z_{18} &= -0.3\\
Z_{19} &= -0.2\\
Z_{20} &= -0.1\\
Z_{21} &= 0\\
Z_{22} &= 0.1\\
Z_{23} &= 0.2\\
Z_{24} &= 0.3\\
Z_{25} &= 0.4\\
Z_{26} &= 0.5\\
Z_{27} &= 0.6\\
Z_{28} &= 0.7\\
Z_{29} &= 0.8\\
Z_{30} &= 0.9\\
Z_{31} &= 1\\
Z_{32} &= 1.1\\
Z_{33} &= 1.2\\
Z_{34} &= 1.3\\
Z_{35} &= 1.4\\
Z_{36} &= 1.5\\
Z_{37} &= 1.6\\
Z_{38} &= 1.7\\
Z_{39} &= 1.8\\
Z_{40} &= 1.9\\
Z_{41} &= 2
\end{aligned}
$$

Release `0.5` has *experimental* addition of optional argument `E` to `\PolSturmIsolateZeros`. It instructs to search roots only in interval `(-10^E, 10^E)`. Important: the extremities are *assumed to not be roots*. In this example, the `[1]` in `\PolSturmIsolateZeros[1]{S}` gives some speed gain; without it, it turns out in this case that polexpr would have started with `(-10^6, 10^6)` interval.

Please note that this feature may be removed or modified.

9 Roots of a Chebyshev polynomial

```
\poldef T_0(x) := 1;
\poldef T_1(x) := x;
\catcode`@ 11
\count@ 2
\xintloop
  \poldef T_\the\count@(x) :=
          2x*T_\the\numexpr\count@-1\relax
           - T_\the\numexpr\count@-2\relax;
\ifnum\count@<15
\advance\count@ 1
\repeat
\catcode`@ 12

$$T_{15} = \PolTypeset[X]{T_15}$$
\PolToSturm{T_15}{T_15}
\PolSturmIsolateZeros*{T_15}% "*" as we will want to confirm multiplicity one
% takes time (each next decimal digit is obtained by dichotomy)
\PolEnsureIntervalLengths{T_15}{-20}% ensure 20 decimal digits for each root
```

$$T_{15} = 16384X^{15} - 61440X^{13} + 92160X^{11} - 70400X^{9} + 28800X^{7} - 6048X^{5} + 560X^{3} - 15X$$

Here is now an example of customization. Indeed `\PolPrintIntervals` default uses `array` and thus does not allow page breaks. And it uses `Left < Z < Right` as presentation of roots and we would like here rather `Z = decimal expansion...`.

```
% 0.8.6 adds an internal patch which would allow usage of amsmath environments
% like this:
% \def\PolPrintIntervalsBeginEnv{\begin{align*}}
% \def\PolPrintIntervalsEndEnv{\end{align*}}
% (the problem was that align evaluates twice its contents so global variables
%  need a reset at the end of first pass, which is what 0.8.6 took care of)
%
% Let's simply do this:
\def\PolPrintIntervalsBeginEnv{\begingroup\leftskip3cm\relax}
\def\PolPrintIntervalsEndEnv{\par\endgroup}
%
% The rows are separated by \PolPrintIntervalsRowSeparator which defaults to \\
% with LaTeX and \cr with Plain. (prior to 0.8.6 it was hardcoded)
\def\PolPrintIntervalsRowSeparator{\\[\jot]}
% And we enter math mode manually at each row, copying pasting from package
% defaults with some added mathon/mathoff:
\def\PolPrintIntervalsKnownRoot{%
   $\PolPrintIntervalsPrintMultiplicity\quad
   \PolPrintIntervalsTheVar_{\PolPrintIntervalsTheIndex}=%
   \PolPrintIntervalsPrintExactZero
   $
}%
\def\PolPrintIntervalsUnknownRoot{%
   $\PolPrintIntervalsPrintMultiplicity\quad
   \xintifSgn{\PolPrintIntervalsTheLeftEndPoint}%
     {\xintifSgn{\PolPrintIntervalsTheRightEndPoint}
```

```
       {\PolPrintIntervalsTheVar_{\PolPrintIntervalsTheIndex}=%
        \PolPrintIntervalsPrintRightEndPoint\dots}%
      {0>\PolPrintIntervalsTheVar_{\PolPrintIntervalsTheIndex}>%
        \PolPrintIntervalsPrintLeftEndPoint}%
      {\impossibleA}}%
    {\xintifSgn{\PolPrintIntervalsTheRightEndPoint}
      {\impossibleB}%
      {\impossibleC}%
      {0<\PolPrintIntervalsTheVar_{\PolPrintIntervalsTheIndex}<%
        \PolPrintIntervalsPrintRightEndPoint}}%
    {\xintifSgn{\PolPrintIntervalsTheRightEndPoint}
      {\impossibleD}%
      {\impossibleE}%
      {\PolPrintIntervalsTheVar_{\PolPrintIntervalsTheIndex}=%
        \PolPrintIntervalsPrintLeftEndPoint\dots}}%
   $
}%
\PolPrintIntervals{T_15}
```

(mult. 1) $Z_1 = -0.99452189536827333692\ldots$

(mult. 1) $Z_2 = -0.95105651629515357211\ldots$

(mult. 1) $Z_3 = -0.86602540378443864676\ldots$

(mult. 1) $Z_4 = -0.74314482547739423501\ldots$

(mult. 1) $Z_5 = -0.58778525229247312916\ldots$

(mult. 1) $Z_6 = -0.40673664307580020775\ldots$

(mult. 1) $Z_7 = -0.20791169081775933710\ldots$

(mult. 1) $Z_8 = 0$

(mult. 1) $Z_9 = 0.20791169081775933710\ldots$

(mult. 1) $Z_{10} = 0.40673664307580020775\ldots$

(mult. 1) $Z_{11} = 0.58778525229247312916\ldots$

(mult. 1) $Z_{12} = 0.74314482547739423501\ldots$

(mult. 1) $Z_{13} = 0.86602540378443864676\ldots$

(mult. 1) $Z_{14} = 0.95105651629515357211\ldots$

(mult. 1) $Z_{15} = 0.99452189536827333692\ldots$