The Polycal Package

Tatsuya SATO

2014/01/12 Version 0.01b

Abstract

The polycal package adds calculus macros to polynom package. It can differentiate or integrate polynomials.

1 Introduction

The polynom package manipulates polynomials. It supports multiplication, division, getting GCD of polynomials with Euclidean algorithm, factorization with coefficients in rational numbers. But it lacks in calculus operation, and substitution commands. Polycal package adds these commands to polynom package.

For one variable polynomial $P(x)$, substitution $P(a)$ can be obtained as a remainder of division with $x - a$. Also dviding with $(x - a)^2$, one can obtain $P'(a)(x-a)+P(a)$ as a remainder, and can get $P'(a)$. But polynom package can't deal with multivariate polynomial enough, so sometimes divsion fails. One can't specify main variable. So I implemented direct substitution commands.

2 Acknowledgments

I wish to thank Carsten Heinz, and Hendri Adriaens for creation and maintainance of polynom package.

3 Usage

See the manual of polynom package for a basic usage of original commands. This manual explains newly added commands.

3.1 Definition of polynomials

First, specify variables you'll use.

```
\polyset{vars=xab}
```

Define some polynomials using `\polydefine`.

`\polydefine` ⟨*macro*⟩`{`⟨*polynomial*⟩`}`

```
\polydefine\fx{x^2} \polydefine\gx{(a+b)x-ab}
```

Polynomials are set to macros `\fx` and `\gx`. You can print them using `\polyprint`:

```
\[ f(x)=\polyprint\fx,\quad\quad  g(x)=\polyprint\gx. \]
```

$$f(x) = x^2, \qquad g(x) = bx + ax - ab.$$

Never print `\fx` or `\gx` directly.

3.2 Differentiation

You can differentiate polynomials using `\polydiff`.

`\polydiff` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*polynomial*⟩`}`

```
\polydiff\Dfx{x}\fx
\[ \frac{df(x)}{dx}=\polyprint\Dfx. \]
```

$$\frac{df(x)}{dx} = 2x.$$

If you define `\x` as `\polydefine\x{x}` , you can write this operation as follows.

```
\polydiff\aa{x}{x^2}   \polydiff\bb{x}\fx
\polydiff\cc\x{x^2}    \polydiff\dd\x\fx
\[\polyprint\aa, \polyprint\bb, \polyprint\cc, \polyprint\dd.\]
```

$$2x, 2x, 2x, 2x.$$

You can differentiate with another variable:

```
\polydiff\Dag{a}\gx \polydiff\Dbg{b}\gx
\[ \frac{\partial g(x,a,b)}{\partial a} =\polyprint\Dag,\quad
   \frac{\partial g(x,a,b)}{\partial b} =\polyprint\Dbg. \]
```

$$\frac{\partial g(x,a,b)}{\partial a} = x - b, \quad \frac{\partial g(x,a,b)}{\partial b} = x - a.$$

But in this manual, $g(x)$ will be treated as one variable function.

```
\polydiff\Dgx{x}\gx  \[ \frac{dg(x)}{dx}=\polyprint\Dgx. \]
```

$$\frac{dg(x)}{dx} = b + a.$$

3.3 Integration

Now we will calculate area between $y = f(x)$ and $y = g(x)$.

```
\polysub\hx\gx\fx
\[ h(x)=g(x)-f(x)=\polyprint\hx. \]
```

$$h(x) = g(x) - f(x) = -x^2 + bx + ax - ab.$$

We will integrate this function.

`\polyint` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*polynomial*⟩`}`

This macro caclulates indefinite integral of ⟨*polynomial*⟩ from ⟨*variable*⟩$= 0$.

```
\polyint\Hx{x}\hx \[ H(x)=\int_0^x h(t)dt= \polyprint\Hx. \]
```

$$H(x) = \int_0^x h(t)dt = -\frac{1}{3}x^3 + \frac{1}{2}bx^2 + \frac{1}{2}ax^2 - abx.$$

3.4 Substitution

And definite integral can be obtained by substitution.

`\polysubst` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*polynomial-1*⟩`}{`⟨*polynomial-2*⟩`}`

This macro substitute ⟨*polynomial-1*⟩ for ⟨*variable*⟩ in ⟨*polynomial-2*⟩.

```
\polysubst\Ha{x}{a}\Hx \polysubst\Hb{x}{b}\Hx  \polysub\S\Hb\Ha
\[ S=\int_a^bh(x)dx=H(b)-H(a)=\polyprint\S. \]
```

$$S = \int_a^b h(x)dx = H(b) - H(a) = \frac{1}{6}b^3 - \frac{1}{6}a^3 + \frac{1}{2}a^2b - \frac{1}{2}ab^2.$$

Unfortunately, factorization of multivariate polynomial is not currently supported, so this can not be arranged to $S = \dfrac{(b-a)^3}{6}$ automatically.

Let's try another substitution. If you substitute $b = a^2 - 2$ in S, then

```
\polysubst\SS{b}{a^2-2}\S
\[ b=a^2-2\ \  \Rightarrow\ \  S=\polyprint\SS=\polyfactorize\SS. \]
```

$$b = a^2 - 2 \ \ \Rightarrow \ \ S = \frac{1}{6}a^6 - \frac{1}{2}a^5 - \frac{1}{2}a^4 + \frac{11}{6}a^3 + a^2 - 2a - \frac{4}{3} = \frac{1}{6}\left(a+1\right)^3\left(a-2\right)^3.$$

`\polysubstnum` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*rational number*⟩`}{`⟨*polynomial*⟩`}`

To substitute a number, there's another command. Compared with `\polysubst`, it is based on different algorithm, but not so fast. Experimental.

`\polysubstsqrt` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*rational number*⟩`}{`⟨*1-variable polynomial*⟩`}`

This macro substitute $\sqrt{\langle \textit{rational number} \rangle}$ for ⟨*variable*⟩. Note that polynomial should be one variable. Let substitute $a = \sqrt{5}$ in $S = \frac{1}{6}\left(a+1\right)^3\left(a-2\right)^3$.

```
\polysubstsqrt\SSS{a}{5}\SS \[ S=\polyprint\SSS. \]
```

$$S = \left(12 + \frac{-16}{3}\sqrt{5}\right).$$

To eliminate parentheses, set `\polyset{delims={}{}}`.

```
\[\polyset{delims={}{}}S=\polyprint\SSS. \]
```

$$S = 12 + \frac{-16}{3}\sqrt{5}.$$

4 Implementation

Let's start with identification.

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{polycal}[2014/01/11 v0.01b (TS)]
\RequirePackage{polynom}
```

4.1 The user commands

`\polydefine` ⟨*macro*⟩`{`⟨*polynomial*⟩`}`

Transform ⟨*polynomial*⟩ to internal representation, and set it to ⟨*macro*⟩.

```
\newcommand*\polydefine[2]{\pld@GetPoly#1{}{#2}}
```

`\polydiff` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*polynomial*⟩`}`

Differentiate ⟨*polynomial*⟩ with respect to ⟨*variable*⟩, and set it to ⟨*macro*⟩.

```
\newcommand*\polydiff[1]{%
  \pld@GetPoly{\pld@polya\pld@polyb}%
  {\pld@DiffPoly#1\pld@polya\pld@polyb
    \ignorespaces}}
```

`\polyint` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*polynomial*⟩`}`

Integrate ⟨*polynomial*⟩ from 0, with respect to ⟨*variable*⟩.

```
\newcommand*\polyint[1]{%
  \pld@GetPoly{\pld@polya\pld@polyb}%
  {\pld@IntPoly#1\pld@polya\pld@polyb
    \ignorespaces}}
```

`\polysubstnum` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*rational number*⟩`}{`⟨*polynomial*⟩`}`

Substitute ⟨*rational number*⟩ for ⟨*variable*⟩ in ⟨*polynomial*⟩.

```
\newcommand*\polysubstnum[1]{%
  \pld@GetPoly{\pld@polya\pld@polyb\pld@polyc}%
  {\pld@SubstNumPoly#1\pld@polya\pld@polyb\pld@polyc
    \ignorespaces}}
```

`\polypower` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*polynomial*⟩`}{`⟨*non negative integer*⟩`}`

Set ⟨*non negative integer*⟩th power of ⟨*polynomial*⟩ to ⟨*macro*⟩.

```
\newcommand*\polypower[1]{%
  \pld@GetPoly{\pld@polya}%
  {\pld@PowerPoly#1\pld@polya}}
```

`\polysubst` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*polynomial-1*⟩`}{`⟨*polynomial-2*⟩`}`

Substitute ⟨*polynomial-1*⟩ for ⟨*variable*⟩ in ⟨*polynomial-2*⟩.

```
\newcommand*\polysubst[1]{%
  \pld@GetPoly{\pld@polya\pld@polyb\pld@polyc}%
  {\pld@SubstPolyPoly#1\pld@polya\pld@polyb\pld@polyc
    \ignorespaces}}
```

`\polysubstsqrt` ⟨*macro*⟩`{`⟨*variable*⟩`}{`⟨*rational number*⟩`}{`⟨*1-variable polynomial*⟩`}`

Substitute non negative square root of ⟨*rational number*⟩ for ⟨*variable*⟩ in ⟨*1-variable polynomial*⟩.

```
\newcommand*\polysubstsqrt[1]{%
  \pld@GetPoly{\pld@polya\pld@polyb\pld@polyc}%
  {\pld@SubstSqrtPoly#1\pld@polya\pld@polyb\pld@polyc
    \ignorespaces}}
```

4.2 Internal routines

The following three macros gets parameter in internal representation of factors.

`\@fifthofsix` `\@secondofthree` `\@thirdofthree`

For example, `\polydefine\X{X}` is equivalent to `\def\X{\pld@R{1}{1}\pld@V{X}{1}}`, and `\@fifthofsix\X` gets variable letter `X`.

```
\providecommand\@fifthofsix[6]{#5}
\providecommand\@secondofthree[3]{#2}
\providecommand\@thirdofthree[3]{#3}
```

4.2.1 Differentiation

`\pld@DiffPoly` ⟨*macro a*⟩⟨*macro b*⟩⟨*macro c*⟩

⟨*macro a*⟩ gets the derivative $\frac{d\langle \textit{macro c}\rangle}{d\langle \textit{macro b}\rangle}$.

```
\def\pld@DiffPoly#1#2#3{%
      \let\polycalc@tempb\@empty%
      \ifx\@empty#3\relax\else
          \expandafter\pld@DiffPoly@\expandafter#2#3+\relax+%
      \fi
      \let#1\polycalc@tempb \pld@Simplify{#1}%
    }
```

`\pld@DiffPoly@` This macro cuts away a monomial in the third argument ⟨*macro c*⟩ of `\pld@DiffPoly`, send it to `\pld@DiffMonom`.

```
\def\pld@DiffPoly@#1#2+{%
  \let\polycalc@tempa\@empty
    \ifx\relax#2 \relax
    \else
        \pld@DiffMonom\polycalc@tempa#1{#2}%
        \pld@ExtendPoly\polycalc@tempb{\polycalc@tempa}%
        \expandafter\pld@DiffPoly@\expandafter#1%
\fi}
```

`\pld@DiffMonom` ⟨*macro a*⟩⟨*macro b*⟩⟨*macro c*⟩

This macro decrease the degree of the monomial ⟨*macro c*⟩ by 1 with respect to variable ⟨*macro b*⟩ and send it to `\pld@DiffMonom@`.

```
\def\pld@DiffMonom#1#2#3{%
    \let#1\@empty
    \edef\pldc@var@letter{\expandafter\@fifthofsix#2}%
    \def\@tempb{#3}\edef\@tempc{\noexpand\pld@V{\pldc@var@letter}{-1}}%
    \pld@MultiplyPoly\@tempa\@tempb\@tempc%
    \def\pldc@CallMono@##1{%
    \expandafter\pld@DiffMonom@\expandafter #1\expandafter##1%
    \@tempa\relax\@empty\@empty\relax\relax}%
    \expandafter\pldc@CallMono@\pldc@var@letter
}
```

`\pld@DiffMonom@` ⟨*macro a*⟩⟨*variable letter*⟩⟨*macro b*⟩`{`⟨*content c*⟩`}{`⟨*content d*⟩`}`

This macro cuts away a factor ⟨*macro b*⟩`{`⟨*content c*⟩`}{`⟨*content d*⟩`}`, of the monomial argument ⟨*macro c*⟩ of `\pld@DiffMonom`. If the factor is a power of ⟨*variable letter*⟩, do $x^{n-1} \rightarrow nx^{n-1}$.

```
\def\pld@DiffMonom@#1#2#3#4#5{%
    \ifx\relax#3
      \relax \expandafter\@gobbletwo
    \else
      \ifx #3\pld@V \if#4#2
            \@tempcnta #5 \advance\@tempcnta 1
            \expandafter\pld@Extend\expandafter#1{\expandafter\pld@R\expandafter{\the\@tempcnta
      \fi\fi
      \pld@Extend#1{\noexpand#3{#4}{#5}}%
    \expandafter\pld@DiffMonom@\expandafter#1\expandafter#2%
    \fi}
```

4.2.2 Integration

`\pld@IntPoly` ⟨*macro a*⟩⟨*macro b*⟩⟨*macro c*⟩

⟨*macro a*⟩ gets the integration $\int_0^{\langle macro\ b\rangle} \langle macro\ c\rangle d\langle macro\ b\rangle$.

```
\def\pld@IntPoly#1#2#3{%
      \let\polycalc@tempb\@empty
      \ifx\@empty#3\relax\else
          \expandafter\pld@IntPoly@\expandafter#2#3+\relax+%
      \fi
    \let#1\polycalc@tempb \pld@Simplify{#1}%
    }
```

`\pld@IntPoly@` This macro cuts away a monomial in the third argument ⟨*macro c*⟩ of `\pld@IntPoly`, send it to `\pld@IntMonom`.

```
\def\pld@IntPoly@#1#2+{%
  \let\polycalc@tempa\@empty
    \ifx\relax#2 \relax
    \else
        \pld@IntMonom\polycalc@tempa#1{#2}%
        \pld@ExtendPoly\polycalc@tempb{\polycalc@tempa}%
        \expandafter\pld@IntPoly@\expandafter#1%
\fi}
```

`\pld@IntMonom` ⟨*macro a*⟩⟨*macro b*⟩⟨*macro c*⟩

This macro increase the degree of the monomial ⟨*macro c*⟩ by 1 with respect to variable ⟨*macro b*⟩ and send it to `\pld@IntMonom@`.

```
\def\pld@IntMonom#1#2#3{%
    \let#1\@empty
    \def\@tempb{#3}\def\@tempc{#2}%
    \pld@MultiplyPoly\@tempa\@tempb\@tempc
    \edef\pldc@var@letter{\expandafter\@fifthofsix#2}%
    \def\pldc@CallMono@##1{%
    \expandafter\pld@IntMonom@\expandafter #1\expandafter##1%
    \@tempa\relax\@empty\@empty\relax\relax}%
    \expandafter\pldc@CallMono@\pldc@var@letter
}
```

`\pld@IntMonom@` ⟨*macro a*⟩⟨*variable letter*⟩⟨*macro b*⟩`{`⟨*content c*⟩`}{`⟨*content d*⟩`}`

This macro cuts away a factor ⟨*macro b*⟩`{`⟨*content c*⟩`}{`⟨*content d*⟩`}`, of the monomial argument ⟨*macro c*⟩ of `\pld@IntMonom`. If the factor is a power of ⟨*variable letter*⟩, do $x^{n+1} \rightarrow \frac{x^{n+1}}{n+1}$.

```
\def\pld@IntMonom@#1#2#3#4#5{%
    \ifx\relax#3 \relax
```

```
      \relax \expandafter\@gobbletwo
    \else
      \ifx #3\pld@V \if#4#2
        \pld@Extend#1{\noexpand\pld@R{1}{#5}}%
      \fi\fi
      \pld@Extend#1{\noexpand#3{#4}{#5}}%
    \expandafter\pld@IntMonom@\expandafter#1\expandafter#2%
    \fi}
```

4.2.3 Substitution of number

`\pld@SubstNumPoly` ⟨*macro a*⟩⟨*macro b*⟩⟨*macro c*⟩⟨*macro d*⟩

Substitute rational number ⟨*macro c*⟩ for variable ⟨*macro b*⟩ in the polynomial ⟨*macro d*⟩, and stores it to ⟨*macro a*⟩.

```
\def\pld@SubstNumPoly#1#2#3#4{%
      \let\polycalc@tempb\@empty
      \ifx\@empty#4\relax\else
          \expandafter\pld@SubstNumPoly@\expandafter#2\expandafter#3#4+\relax+%
      \fi
    \let#1\polycalc@tempb \pld@Simplify{#1}%
    }
```

`\pld@SubstNumPoly@` This macro cuts away a monomial in the fourth argument ⟨*macro d*⟩ of `\pld@SubstNumPoly`, send it to `\pld@SubstNumMonom`.

```
\def\pld@SubstNumPoly@#1#2#3+{%
  \let\polycalc@tempa\@empty
    \ifx\relax#3 \relax
    \else
        \pld@SubstNumMonom\polycalc@tempa#1#2{#3}%
        \pld@ExtendPoly\polycalc@tempb{\polycalc@tempa}%
        \expandafter\pld@SubstNumPoly@\expandafter#1\expandafter#2%
\fi}
```

`\pld@SubstNumMonom` ⟨*macro a*⟩⟨*macro b*⟩⟨*macro c*⟩⟨*macro d*⟩

This macro only replace ⟨*macro b*⟩ with its variable letter, and sends four arguments to `\pld@SubstNumMonom@`.

```
\def\pld@SubstNumMonom#1#2#3#4{%
    \let#1\@empty
    \edef\pldc@var@letter{\expandafter\@fifthofsix#2}%
    \def\pldc@CallMono@##1{%
    \expandafter\pld@SubstNumMonom@\expandafter#1\expandafter##1\expandafter#3\expandafter%
    #4\relax\@empty\@empty\relax\relax}%
    \expandafter\pldc@CallMono@\pldc@var@letter
}
```

`\pld@SubstNumMonom@` ⟨*macro a*⟩⟨*variable letter*⟩⟨*macro b*⟩⟨*macro c*⟩`{`⟨*contents d*⟩`}{`⟨*contents e*⟩`}`

This macro cuts away a factor ⟨*macro c*⟩`{`⟨*content d*⟩`}{`⟨*content e*⟩`}`, of the monomial argument ⟨*macro d*⟩ of `\pld@SubstNumMonom`. If the factor contains ⟨*variable letter*⟩ as variable, this macro substitute the rational number ⟨*macro b*⟩ for that variable.

```
\def\pld@SubstNumMonom@#1#2#3#4#5#6{%
    \ifx\relax#4 \relax \expandafter\@gobbletwo
    \else
      \ifx #4\pld@V
          \if#5#2
             \def\@tempb{\pld@R{-1}{1}}\def\@tempc{#3}%
             \pld@MultiplyPoly\@tempa\@tempb\@tempc
             \edef\@tempb{\noexpand\pld@V{#5}{1}}%
             \pld@ExtendPoly\@tempb\@tempa \pld@Simplify\@tempb
             \def\@tempc{\pld@V{#5}{#6}}%
             \let\pld@stage\maxdimen
             \pld@DividePoly\@tempc\@tempb \pld@Simplify\pld@remainder
             \pld@Extend#1{\pld@remainder}%
          \else
             \pld@Extend#1{\noexpand#4{#5}{#6}}%
          \fi
      \else
          \pld@Extend#1{\noexpand#4{#5}{#6}}%
      \fi
    \expandafter\pld@SubstNumMonom@\expandafter#1\expandafter#2\expandafter#3%
    \fi}
```

4.2.4 Substitution of polynomial

`\pld@SubstPolyPoly` ⟨*macro a*⟩⟨*macro b*⟩⟨*macro c*⟩⟨*macro d*⟩

Substitute polynomial ⟨*macro c*⟩ for variable ⟨*macro b*⟩ in the polynomial ⟨*macro d*⟩, and stores it to ⟨*macro a*⟩.

```
\def\pld@SubstPolyPoly#1#2#3#4{%
      \let\polycalc@tempb\@empty
      \ifx\@empty#4\relax\else
          \expandafter\pld@SubstPolyPoly@\expandafter#2\expandafter#3#4+\relax+%
      \fi
    \let#1\polycalc@tempb \pld@Simplify{#1}%
    }
```

`\pld@SubstPolyPoly@` This macro cuts away a monomial in the fourth argument ⟨*macro d*⟩ of `\pld@SubstPolyPoly`, send it to `\pld@SubstPolyMonom`.

```
\def\pld@SubstPolyPoly@#1#2#3+{%
  \let\polycalc@tempa\@empty
    \ifx\relax#3 \relax
    \else
        \pld@SubstPolyMonom\polycalc@tempa#1#2{#3}%
        \pld@ExtendPoly\polycalc@tempb{\polycalc@tempa}%
```

```
        \expandafter\pld@SubstPolyPoly@\expandafter#1\expandafter#2%
\fi}
```

`\pld@SubstPolyMonom` ⟨*macro a*⟩⟨*macro b*⟩⟨*macro c*⟩⟨*macro d*⟩

This macro only replace ⟨*macro b*⟩ with its variable letter, and sends four arguments to `\pld@SubstPolyMonom@`.

```
\def\pld@SubstPolyMonom#1#2#3#4{%
    \def#1{\pld@R{1}{1}}%
    \edef\@tempb{\expandafter\@fifthofsix#2}%
    \edef\pldc@var@letter{\expandafter\@fifthofsix#2}%
    \def\pldc@CallMono@##1{%
    \expandafter\pld@SubstPolyMonom@\expandafter#1\expandafter##1\expandafter#3\expandafter%
    #4\relax\@empty\@empty\relax\relax}%
   \expandafter\pldc@CallMono@\pldc@var@letter
}
```

`\pld@SubstPolyMonom@` ⟨*macro a*⟩⟨*variable letter*⟩⟨*macro b*⟩⟨*macro c*⟩`{`⟨*contents d*⟩`}{`⟨*contents e*⟩`}`

This macro cuts away a factor ⟨*macro c*⟩`{`⟨*content d*⟩`}{`⟨*content e*⟩`}`, of the monomial argument ⟨*macro d*⟩ of `\pld@SubstPolyMonom`. If the factor contains ⟨*variable letter*⟩ as variable, this macro substitute the polynomial ⟨*macro b*⟩ for that variable.

```
\def\pld@SubstPolyMonom@#1#2#3#4#5#6{%
    \ifx\relax#4 \relax \expandafter\@gobbletwo
    \else
      \ifx #4\pld@V
         \if#5#2
             \edef\@tempa{\number#6}%
             \pld@PowerPoly\@tempb#3\@tempa
             \expandafter\def\expandafter\@tempa\expandafter{#1}%
             \pld@MultiplyPoly#1\@tempa\@tempb
          \else
            \expandafter\def\expandafter\@tempa\expandafter{#1}%
            \edef\@tempb{\noexpand#4{#5}{#6}}%
            \pld@MultiplyPoly#1\@tempa\@tempb
          \fi
      \else
          \expandafter\def\expandafter\@tempa\expandafter{#1}%
          \edef\@tempb{\noexpand#4{#5}{#6}}%
          \pld@MultiplyPoly#1\@tempa\@tempb
      \fi
    \expandafter\pld@SubstPolyMonom@\expandafter#1\expandafter#2\expandafter#3%
    \fi}
```

`\pld@PowerPoly` ⟨*macro a*⟩⟨*macro b*⟩`{`⟨*number a*⟩`}`

⟨*macro a*⟩ gets a ⟨*number a*⟩th power of polynomial ⟨*macro b*⟩. ⟨*number a*⟩ should be a non negative integer.

```
\def\pld@PowerPoly#1#2#3{%
  \ifnum#3=\z@ \def#1{\pld@R{1}{1}}%
  \else
    \@tempcnta#3\advance\@tempcnta-1
    \edef\@tempa{\the\@tempcnta}%
    \pld@PowerPoly#1#2\@tempa
    \let\@tempb#1
    \pld@MultiplyPoly#1\@tempb#2%
  \fi
}
```

4.2.5 Substitution of square root

\pld@SubstSqrtPoly $\langle macro\ a\rangle\langle macro\ b\rangle\langle macro\ c\rangle\langle macro\ d\rangle$

Substitute non negative square root of rational number $\langle macro\ c\rangle$ for variable $\langle macro\ b\rangle$ in the **one variable** polynomial $\langle macro\ d\rangle$, and stores it to $\langle macro\ a\rangle$.

```
\def\pld@SubstSqrtPoly#1#2#3#4{%
  \edef\pldc@var@letter{\expandafter\@fifthofsix#2}%
  \edef\pldc@numer@sqrt{\expandafter\@secondofthree#3}%
  \edef\pldc@denom@sqrt{\expandafter\@thirdofthree#3}%
  \edef\pld@polyd{%
    \noexpand\pld@R{1}{1}\noexpand\pld@V{\pldc@var@letter }{2}%
    +\noexpand\pld@R{-\pldc@numer@sqrt }{\pldc@denom@sqrt }}%
  \let\pld@stage\maxdimen
  \pld@DividePoly#4\pld@polyd
  \let\pldc@main@formula\pld@remainder
  \pld@FDefSqrt{\pldc@numer@sqrt}{\pldc@denom@sqrt}\let\pldc@sqrt@inner\pld@temp
  \def\pldc@sqrt@outer{\pld@R {1}{1}\pld@S {\pldc@sqrt@inner }{1}}%
  \pld@SubstNumPoly#1#2\pldc@sqrt@outer\pldc@main@formula%
}
```

4.3 Miscellany

\pld@IfMonomL@ A bug fix of \pld@IfMonomL@ in polynom.sty.

```
%%%%%% The folowing is a bug fix of \pld@IfMonomL@ in polynom.sty.
\def\pld@IfMonomL@#1\pld@V#2#3#4\@empty#5\pld@V#6#7#8\@empty{%
    \let\pld@next\@empty
          \ifx #6\relax \aftergroup\@secondoftwo
    \else \ifx #2\relax \aftergroup\@firstoftwo
    \else
        \def\pld@va{#2}\def\pld@vb{#6}%
        \ifx\pld@va\pld@vb
            \ifnum#3=#7\relax
                \def\pld@next{\pld@IfMonomL@#4\@empty#8\@empty}%
            \else
                \ifnum#3<#7\relax \aftergroup\@firstoftwo
                            \else \aftergroup\@secondoftwo \fi
            \fi
```

```
        \else
            % \pld@IfVarL#2\relax\@empty#6\relax\@empty
            \pld@IfVarL#2#6%
                {\aftergroup\@firstoftwo}%
                {\aftergroup\@secondoftwo}%
        \fi
    \fi \fi
    \pld@next}
```

⟨/package⟩

5 History

0.01 2014/01/09 (private test version)

- Implemented `\pld@DiffPoly`, `\pld@IntPoly`, `\pld@SubstNumPoly`, `\pld@SubstPlyPoly`, `\pld@SubstSqrtPoly`.

0.01a 2014/01/11

- Fixed a bug in `\pld@IfMonomL@` of original polynom package.

0.01b 2014/01/12

- Wrote .dtx.
- Minor fix in `\pld@SubstSqrtPoly`.

TODO:

- User definable order of variables.
- Rational functions and its derivatives.
- Vector calculus.
- Differential geometric operations.