`pst-func`
plotting special mathematical functions*
v.0.38

Herbert Voß†

November 8, 2004

Contents

*This document was written with `Kile: 1.6a (Qt: 3.1.1; KDE: 3.1.1;` `http://sourceforge.net/projects/kile/`) and the PDF output was build with VTeX/Free (`http://www.micropress-inc.com/linux`)

†Thanks to: Attila Gati, John Frampton and Lars Kotthoff.

1 psPolynomial

The polynomial function is defined as

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3 + \ldots + a_{n-1}x^{n-1} + a_nx^n \tag{1}$$
$$f'(x) = a_1 + 2a_2x + 3a_3x^2 + \ldots + (n-1)a_{n-1}x^{n-2} + na_nx^{n-1} \tag{2}$$
$$f''(x) = 2a_2 + 6a_3x + \ldots + (n-1)(n-2)a_{n-1}x^{n-3} + n(n-1)a_nx^{n-2} \tag{3}$$

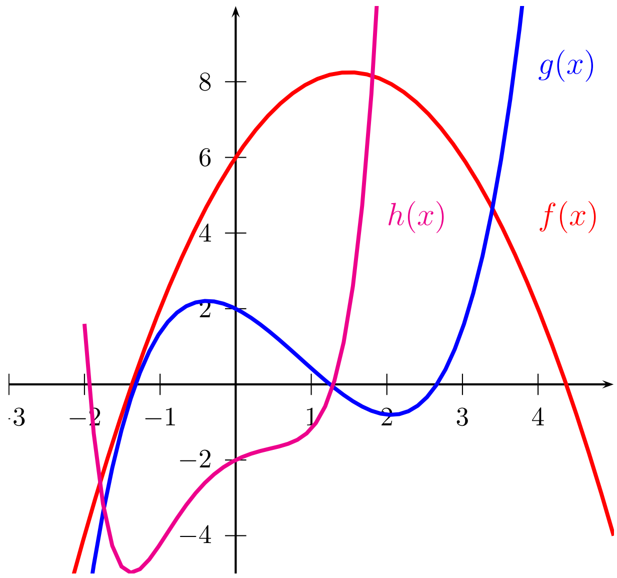
so `pst-func` needs only the coefficients of the polynomial to calculate the function. The syntax is

```
\psPolynomial[<options>]{xStart}{xEnd}
```

There are the following new options:

Name	Value	Default	
`coeff`	`a0 a1 a2 ...`	`0 0 1`	The coefficients must have the order $a_0\ a_1\ a_2 \ldots$ and be separated by **spaces**. The number of coefficients is limited only by the memory of the computer ... The default value of the parameter `coeff` is `0 0 1`, which gives the parabola $y = a_0 + a_1x + a_2x^2 = x^2$.
`Derivation`	`<number>`	`0`	the default is the function itself
`markZeros`	`false|true`	`false`	dotstyle can be changed
`epsZero`	`<value>`	`0.1`	The distance between two zeros, important for the iteration function to test, if the zero value still exists
`dZero`	`<value>`	`0.1`	When searching for all zero values, the function is scanned with this step
`zeroLineTo`	`<number>`	`false`	plots a line from the zero point to the value of the zeroLineTo's Derivation of the polynomial function
`zeroLineStyle`	`<line style>`	`dashed`	the style is one of the for PSTricks valid styles.
`zeroLineColor`	`<color>`	`black`	any valid xolor is possible
`zeroLineWidth`	`<width>`	`0.5\pslinewidth`	

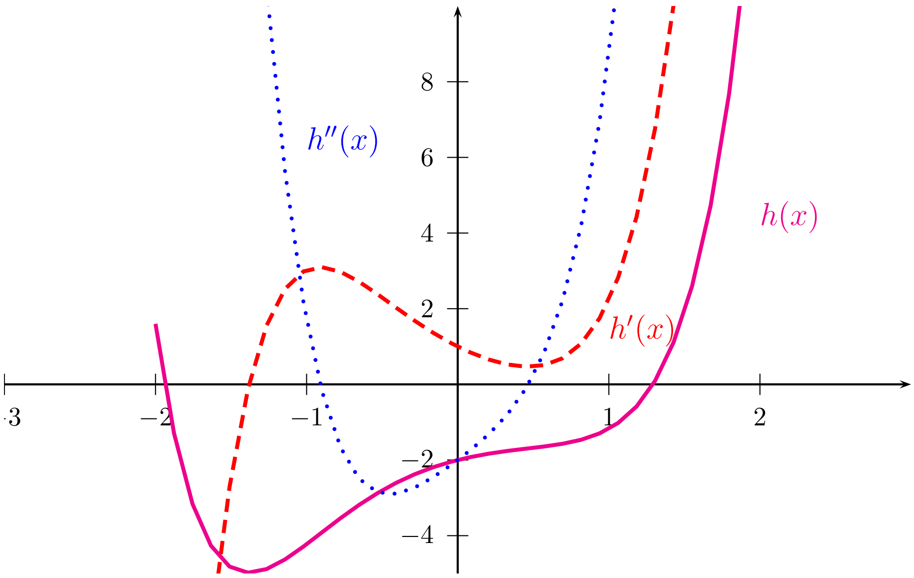
The above parameter are only valid for the `\psPolynomial` macro, but can also be set in the usual way with `\psset`.



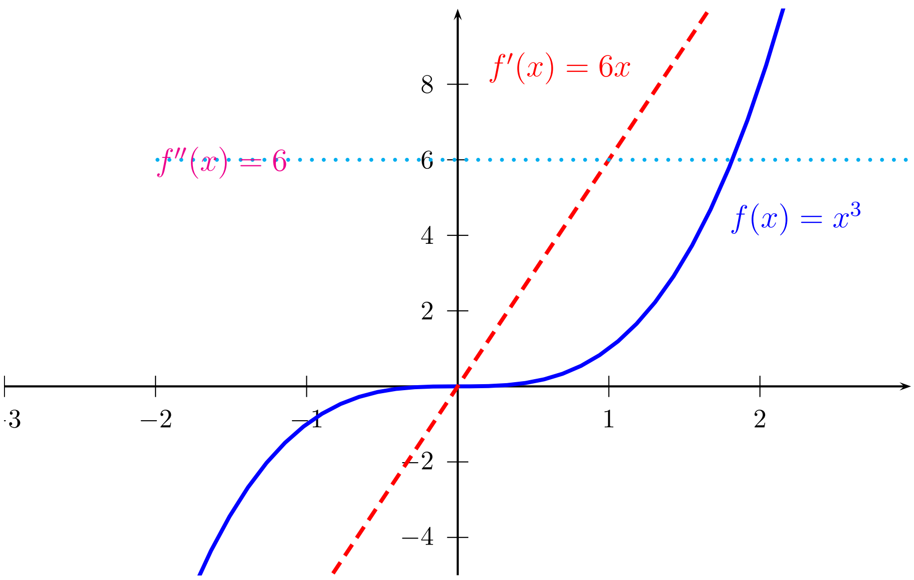

```
{\psset{yunit=0.5cm,xunit=1cm}
\begin{pspicture*}(-3,-5)(5,10)
  \psaxes[Dy=2]{->}(0,0)(-3,-5)(5,10)
  \psset{linewidth=1.5pt}
  \psPolynomial[coeff=6 3 -1,linecolor=red]{-3}{5}
  \psPolynomial[coeff=2 -1 -1 .5 -.1 .025,linecolor=blue]{-2}{4}
  \psPolynomial[coeff=-2 1 -1 .5 .1 .025 .2 ,linecolor=magenta]{-2}{4}
  \rput[lb](4,4){\textcolor{red}{$f(x)$}}
  \rput[lb](4,8){\textcolor{blue}{$g(x)$}}
  \rput[lb](2,4){\textcolor{magenta}{$h(x)$}}
\end{pspicture*}
}
```

The plot is easily clipped using the star version of the **pspicture** environment, so that points whose coordinates are outside of the desired range are not plotted. The plotted polynomials are:

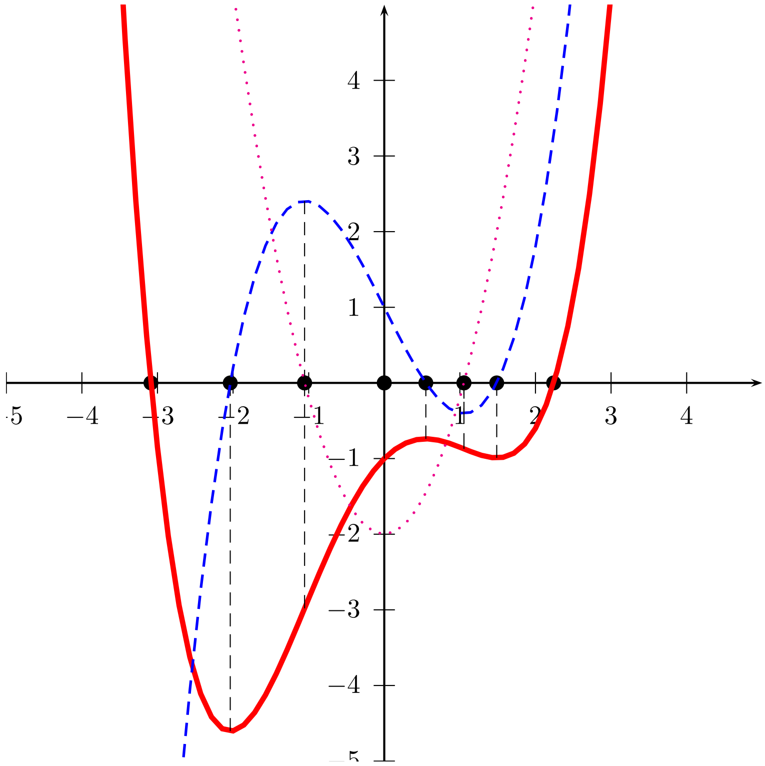
$$f(x) = 6 + 3x - x^2 \tag{4}$$
$$g(x) = 2 - x - x^2 + 0.5x^3 - 0.1x^4 + 0.025x^5 \tag{5}$$
$$h(x) = -2 + x - x^2 + 0.5x^3 + 0.1x^4 + 0.025x^5 + 0.2x^6 \tag{6}$$



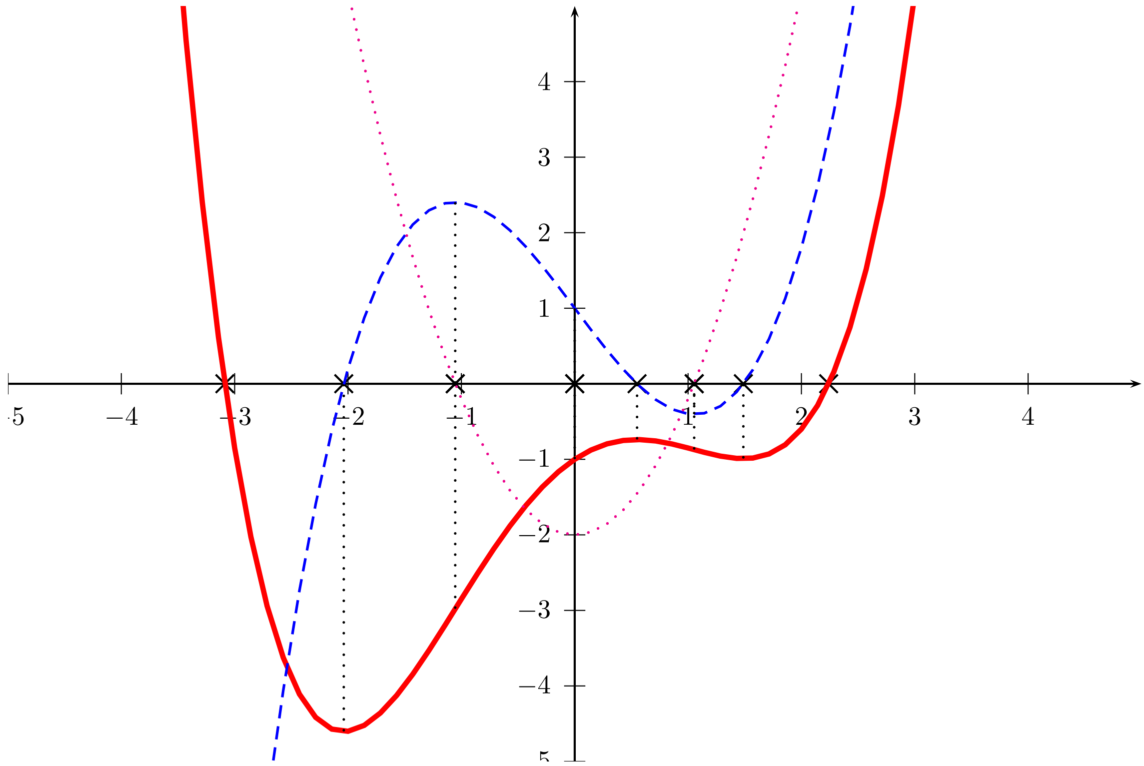

```
\psset{yunit=0.5cm,xunit=2cm}
\begin{pspicture*}(-3,-5)(3,10)
  \psaxes[Dy=2]{->}(0,0)(-3,-5)(3,10)
  \psset{linewidth=1.5pt}
  \psPolynomial[coeff=-2 1 -1 .5 .1 .025 .2 ,linecolor=magenta]{-2}{4}
  \psPolynomial[coeff=-2 1 -1 .5 .1 .025 .2 ,linecolor=red,%
    linestyle=dashed,Derivation=1]{-2}{4}
  \psPolynomial[coeff=-2 1 -1 .5 .1 .025 .2 ,linecolor=blue,%
    linestyle=dotted,Derivation=2]{-2}{4}
  \rput[lb](2,4){\textcolor{magenta}{$h(x)$}}
  \rput[lb](1,1){\textcolor{red}{$h^{\prime}(x)$}}
  \rput[lb](-1,6){\textcolor{blue}{$h^{\prime\prime}(x)$}}
\end{pspicture*}
```



```
\psset{yunit=0.5cm,xunit=2cm}
\begin{pspicture*}(-3,-5)(3,10)
  \psaxes[Dy=2]{->}(0,0)(-3,-5)(3,10)
  \psset{linewidth=1.5pt}
  \psPolynomial[coeff=0 0 0 1,linecolor=blue]{-2}{4}
  \psPolynomial[coeff=0 0 0 1,linecolor=red,%
    linestyle=dashed,Derivation=2]{-2}{4}
  \psPolynomial[coeff=0 0 0 1,linecolor=cyan,%
    linestyle=dotted,Derivation=3]{-2}{4}
  \rput[lb](1.8,4){\textcolor{blue}{$f(x)=x^3$}}
  \rput[lb](0.2,8){\textcolor{red}{$f^{\prime}(x)=6x$}}
  \rput[lb](-2,5.5){\textcolor{magenta}{$f^{\prime\prime}(x)=6$}}
\end{pspicture*}
```




```
\begin{pspicture*}(-5,-5)(5,5)
  \psaxes{->}(0,0)(-5,-5)(5,5)%
  \psset{dotscale=2}
  \psPolynomial[markZeros,linecolor=red,linewidth=2pt,coeff
    =-1 1 -1 0 0.15]{-4}{3}%
  \psPolynomial[markZeros,linecolor=blue,linewidth=1pt,linestyle=dashed,%
    coeff=-1 1 -1 0 0.15,Derivation=1,zeroLineTo=0]{-4}{3}%
  \psPolynomial[markZeros,linecolor=magenta,linewidth=1pt,linestyle=dotted,
    %
    coeff=-1 1 -1 0 0.15,Derivation=2,zeroLineTo=0]{-4}{3}%
  \psPolynomial[markZeros,linecolor=magenta,linewidth=1pt,linestyle=dotted,
    %
    coeff=-1 1 -1 0 0.15,Derivation=2,zeroLineTo=1]{-4}{3}%
\end{pspicture*}
```




```
\psset{xunit=1.5}
\begin{pspicture*}(-5,-5)(5,5)
  \psaxes{->}(0,0)(-5,-5)(5,5)%
  \psset{dotscale=2,dotstyle=x,zeroLineStyle=dotted,zeroLineWidth=1pt}
  \psPolynomial[markZeros,linecolor=red,linewidth=2pt,coeff=-1 1 -1 0 0.15]{-4}{3}%
  \psPolynomial[markZeros,linecolor=blue,linewidth=1pt,linestyle=dashed,%
    coeff=-1 1 -1 0 0.15,Derivation=1,zeroLineTo=0]{-4}{3}%
  \psPolynomial[markZeros,linecolor=magenta,linewidth=1pt,linestyle=dotted,%
    coeff=-1 1 -1 0 0.15,Derivation=2,zeroLineTo=0]{-4}{3}%
  \psPolynomial[markZeros,linecolor=magenta,linewidth=1pt,linestyle=dotted,%
    coeff=-1 1 -1 0 0.15,Derivation=2,zeroLineTo=1]{-4}{3}%
\end{pspicture*}
```

2 psFourier

A Fourier sum has the form:

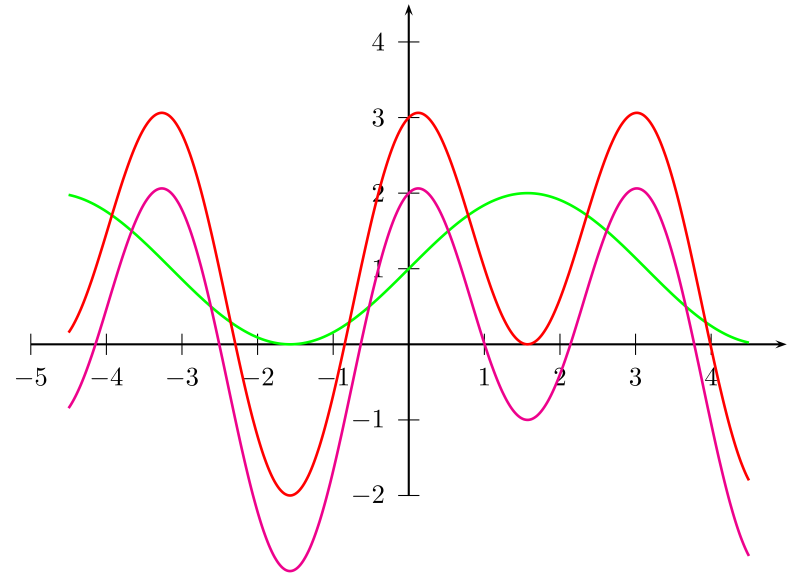
$$s(x) = \frac{a_0}{2} + a_1 \cos \omega x + a_2 \cos 2\omega x + a_3 \cos 3\omega x + \ldots + a_n \cos n\omega x \quad (7)$$

$$+ b_1 \sin \omega x + b_2 \sin 2\omega x + b_3 \sin 3\omega x + \ldots + b_m \sin m\omega x \quad (8)$$

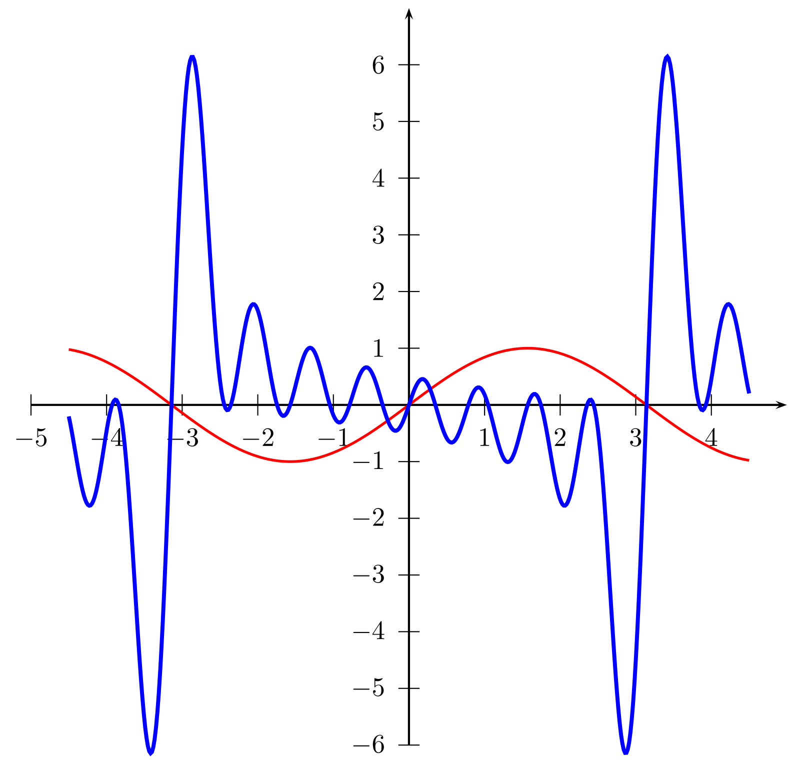
The macro `psFourier` plots Fourier sums. The syntax is similiar to `psPolynomial`, except that there are two kinds of coefficients:

```
\psPolynomial[cosCoeff=a0 a1 a2 ..., sinCoeff=b1 b2 ...]{xStart}{xEnd}
```

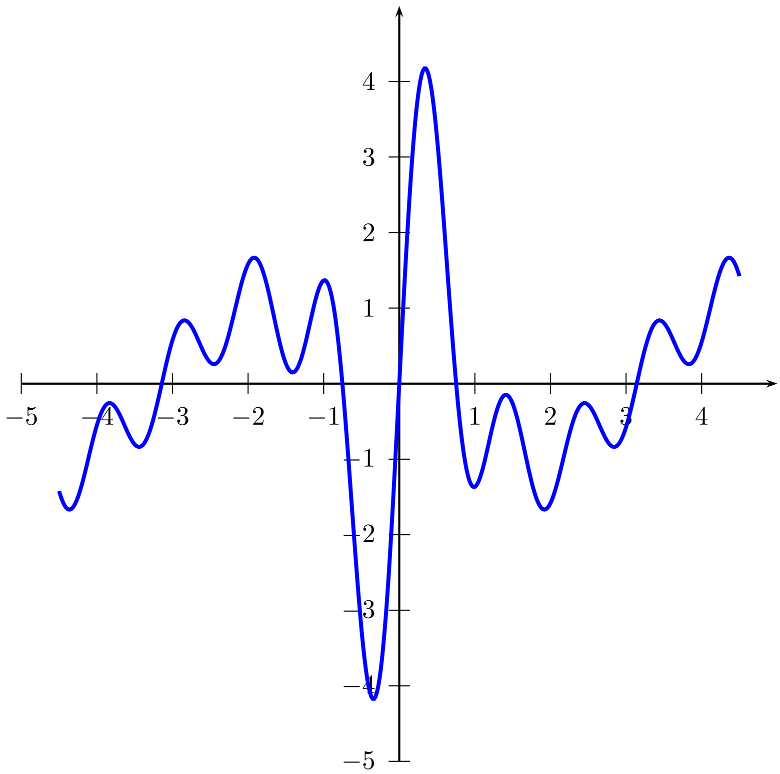
The coefficients must have the orders a_0 a_1 a_2 ... and b_1 b_2 b_3 ... and be separated by **spaces**. The default is `cosCoeff=0,sinCoeff=1`, which gives the standard `sin` function. Note that the constant value can only be set with `cosCoeff=a0`.




```
\begin{pspicture}(-5,-3)(5,5.5)
\psaxes{->}(0,0)(-5,-2)(5,4.5)
\psset{plotpoints=500,linewidth=1pt}
\psFourier[cosCoeff=2, linecolor=green]{-4.5}{4.5}
\psFourier[cosCoeff=0 0 2, linecolor=magenta]{-4.5}{4.5}
\psFourier[cosCoeff=2 0 2, linecolor=red]{-4.5}{4.5}
\end{pspicture}
```



```
\psset{yunit=0.75}
\begin{pspicture}(-5,-6)(5,7)
\psaxes{->}(0,0)(-5,-6)(5,7)
\psset{plotpoints=500}
\psFourier[linecolor=red,linewidth=1pt]{-4.5}{4.5}
\psFourier[sinCoeff= -1 1 -1 1 -1 1 -1 1,%
 linecolor=blue,linewidth=1.5pt]{-4.5}{4.5}
\end{pspicture}
```




```
\begin{pspicture}(-5,-5)(5,5.5)
\psaxes{->}(0,0)(-5,-5)(5,5)
\psset{plotpoints=500,linewidth=1.5pt}
\psFourier[sinCoeff=-.5 1 1 1 1 ,sinCoeff=-.5 1 1 1 1 1,%
 linecolor=blue]{-4.5}{4.5}
\end{pspicture}
```

3 psBessel

The Bessel function of order n is defined as

$$J_n(x) = \frac{1}{\pi}\int_0^{\pi}\cos(x\sin t - nt)\mathrm{d}t \tag{9}$$

$$= \sum_{k=0}^{\infty}\frac{(-1)^k\left(\frac{x}{2}\right)^{n+2k}}{k!\Gamma(n+k+1)} \tag{10}$$

The syntax of the macro is

`\psBessel[options]{order}{xStart}{xEnd}`

There are two special parameters for the Bessel function, and also the settings of many `pst-plot` or `pstricks` parameters affect the plot.

```
\def\psset@constI#1{\edef\psk@constI{#1}}
\def\psset@constII#1{\edef\psk@constII{#1}}
\psset{constI=1,constII=0}
```

These two "'constants"' have the following meaning:

$$f(t) = constI \cdot J_n + constII$$

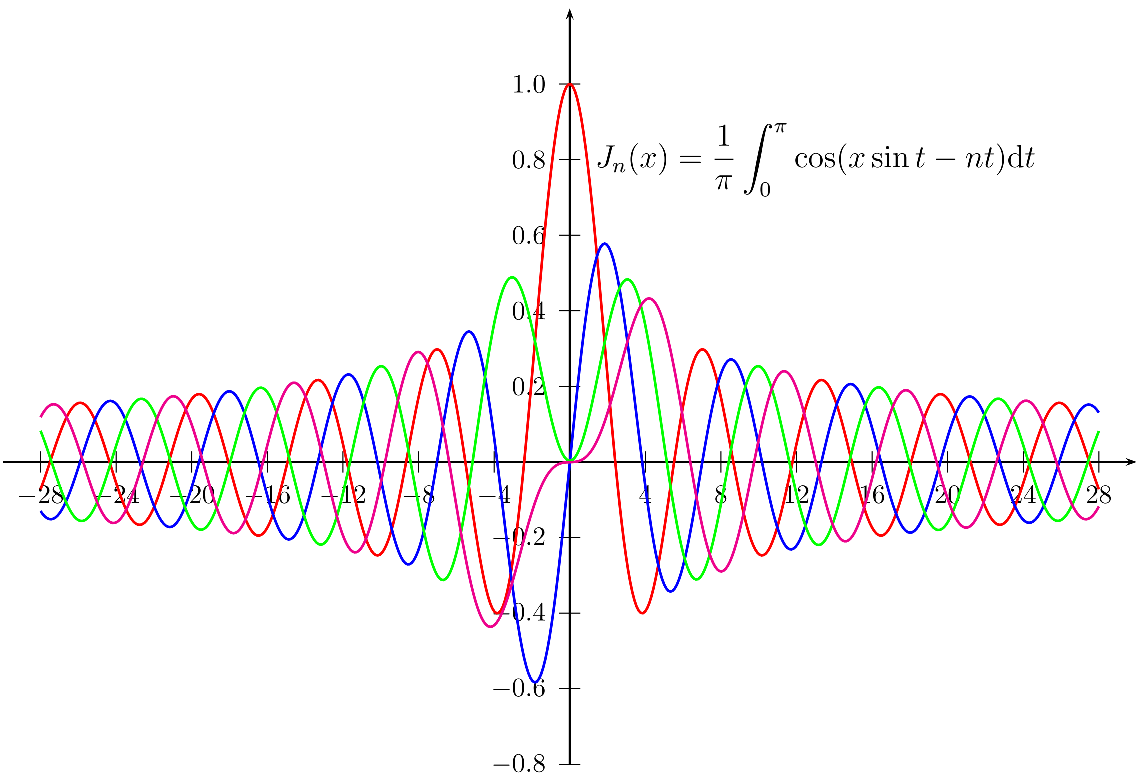
where $constI$ and $constII$ must be real PostScript expressions, e.g.:

```
\psset{constI=2.3,constII=t k sin 1.2 mul 0.37 add}
```

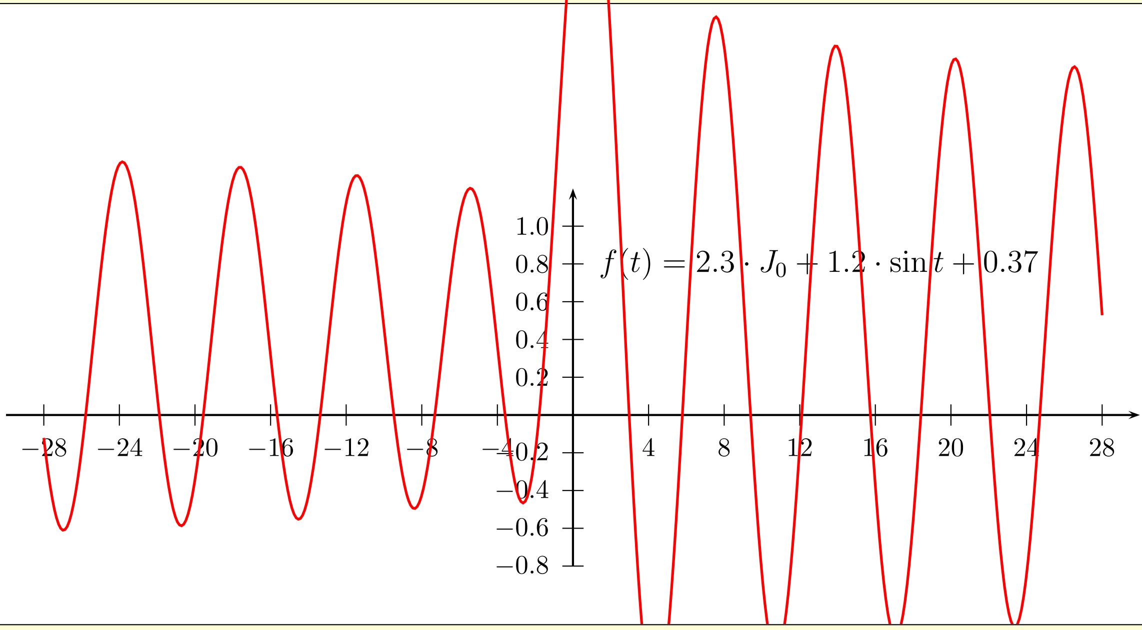
The Bessel function is plotted with the parametricplot macro, this is the reason why the variable is named `t`. The internal procedure `k` converts the value t from radian into degrees. The above setting is the same as

$$f(t) = 2.3 \cdot J_n + 1.2 \cdot \sin t + 0.37$$

In particular, note that the default for `plotpoints` is 500. If the plotting computations are too time consuming at this setting, it can be decreased in the usual way, at the cost of some reduction in graphics resolution.



```
{
\psset{xunit=0.25,yunit=5}
\begin{pspicture}(-13,-.85)(13,1.25)
\rput(13,0.8){%
 $\displaystyle J_n(x)=\frac{1}{\pi}\int_0^\pi\cos(x\sin t-nt)\mathrm{d}t$%
}
\psaxes[Dy=0.2,Dx=4,xyLabel=\footnotesize]{->}(0,0)(-30,-.8)(30,1.2)
\psset{linewidth=1pt}
\psBessel[linecolor=red]{0}{-28}{28}%
\psBessel[linecolor=blue]{1}{-28}{28}%
\psBessel[linecolor=green]{2}{-28}{28}%
\psBessel[linecolor=magenta]{3}{-28}{28}%
\end{pspicture}
}
```




```
{
\psset{xunit=0.25,yunit=2.5}
\begin{pspicture}(-13,-.85)(13,2)
\rput(13,0.8){%
 $\displaystyle f(t) = 2.3 \cdot J_0 + 1.2\cdot \sin t + 0.37$%
}
\psaxes[Dy=0.2,Dx=4,xyLabel=\footnotesize]{->}(0,0)(-30,-.8)(30,1.2)
\psset{linewidth=1pt}
\psBessel[linecolor=red,constI=2.3,constII={t k sin 1.2 mul 0.37 add
  }]{0}{-28}{28}%
\end{pspicture}
}
```

4 psGauss

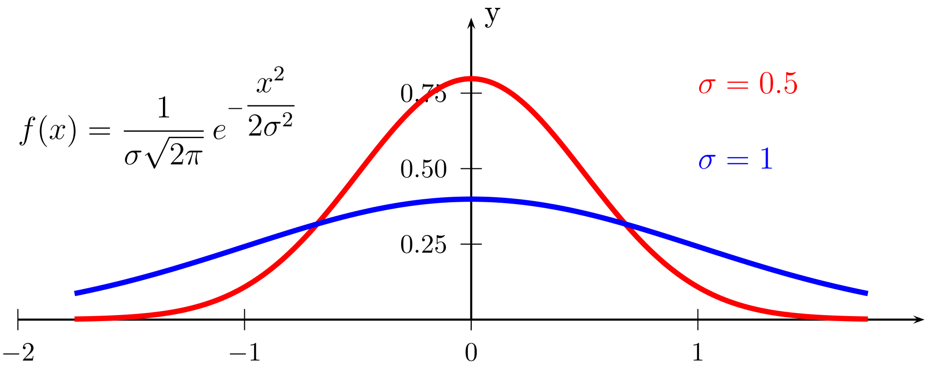
The Gauss function is defined as

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\dfrac{x^2}{2\sigma^2}} \tag{11}$$

The syntax of the macro is

```
\psGauss[options]{xStart}{xEnd}
```

where the only new parameter is `sigma=<value>`, which can also be set in the usual way with `\psset`. It is significant only for the `psGauss`-macro. The default is `0.5`.




```
\psset{yunit=4cm,xunit=3}
\begin{pspicture}(-2,0)(2,1)
%  \psgrid[griddots=10,gridlabels=0pt, subgriddiv=0]
  \psaxes[xyLabel=\footnotesize,Dy=0.25]{->}(0,0)(-2,0)(2,1)
  \uput[-90](6,0){x}\uput[0](0,1){y}
  \rput[lb](1,0.75){\textcolor{red}{$\sigma =0.5$}}
  \rput[lb](1,0.5){\textcolor{blue}{$\sigma =1$}}
  \rput[lb](-2,0.5){$f(x)=\dfrac{1}{\sigma\sqrt{2\pi}}\,e^{-\dfrac{x^2}{2\sigma{}^2}}$}
  \psGauss[linecolor=red, linewidth=2pt]{-1.75}{1.75}%
  \psGauss[sigma=1, linecolor=blue, linewidth=2pt]{-1.75}{1.75}
\end{pspicture}
```

5 Credits

Denis Girou | Manuel Luque | Timothy Van Zandt

References

[1] Denis Girou. Présentation de PSTricks. *Cahier GUTenberg*, 16:21–70, April 1994.

[2] Michel Goosens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Graphics Companion*. Addison-Wesley Publishing Company, Reading, Mass., 1997.

[3] Laura E. Jackson and Herbert Voß. Die Plot-Funktionen von `pst-plot`. *Die TeXnische Komödie*, 2/02:27–34, June 2002.

[4] Nikolai G. Kollock. *PostScript richtig eingesetzt: vom Konzept zum praktischen Einsatz*. IWT, Vaterstetten, 1989.

[5] Herbert Voß. *Chaos und Fraktale selbst programmieren: von Mandelbrotmengen über Farbmanipulationen zur perfekten Darstellung*. Franzis Verlag, Poing, 1994.

[6] Herbert Voß. Die mathematischen Funktionen von PostScript. *Die TeXnische Komödie*, 1/02, March 2002.

[7] Timothy van Zandt. *PSTricks - PostScript macros for generic TEX*. http://www.tug.org/application/PSTricks, 1993.

[8] Timothy van Zandt. *multido.tex - a loop macro, that supports fixed-point addition*. CTAN:/graphics/pstricks/generic/multido.tex, 1997.

[9] Timothy van Zandt. *pst-plot: Plotting two dimensional functions and data*. CTAN:graphics/pstricks/generic/pst-plot.tex, 1999.

[10] Timothy van Zandt and Denis Girou. Inside PSTricks. *TUGboat*, 15:239–246, September 1994.