Macros in PsTricks

Herman Jaramillo
GXT ION—
2105 Citywest Blv # 800, Houston, TX 77042

October 28, 2012

Contents

1 Introduction **1**

2 Riemann Integral **2**
2.1 Theoretical Concepts 2
2.2 Macro Implementation 2

3 Drawing a vector field of an ordinary differential equation **6**
3.1 Theory 6
3.2 Macro Implementation 7
3.3 Example 8

4 Draw an arrow given two points and size **9**

1 Introduction

This is a tutorial to build macros in PSTricks. The first two examples are taken from this document [1]

The first example is about constructing a Riemmann sum. The second example is about the vector field of an ordinary differential equation of order one. Finally I show a small macro to plot a vector when the direction is given at two points, the first of the two points is the origin, and the magnitude is also supplied. This is an important function to plot vectors for basic physics, and unfortunatelly I can not find it in the PsTricks libraries. The simple application of this example is a free diagram of a simple pendulum.

[1] Two applications of macros in PSTricks, by Le Phuong Quan, Cantho University, Vietman. http://www.tex.ac.uk/tex-archive/info/pstricks_calcnotes/two_apps.pdf

2 Riemann Integral

In this section we draw a Riemann sum with centered boxes for the function

$$x - \frac{\cos x}{2} + 2$$

along the domain $[0, 8]$ with boxes of width 0.5.

2.1 Theoretical Concepts

Assume a function f which is bounded in the closed interval $[a, b]$. A partition P of the interval $[a, b]$ is a collection of $n + 1$ points $x_0 = a < x_1 < x_2 < \cdots < x_{n-1} < x_n = b$. The size of the partition is defined as

$$\|P\| = \max_1^n (x_i - x_{i-1}) \max_1^n \Delta x_i$$

The Riemann integral for f in the interval $[a, b]$ is defined as

$$\int_a^b f(x)dx = \lim_{\|P\|\to 0} \sum_{i=1}^{n} f(\xi_i)\Delta x_i,$$

where $\xi_i \in [x_{i-1}, x_i]$. Without loss of generality we assume that Δx_i is constant and this simplifies the drawing.

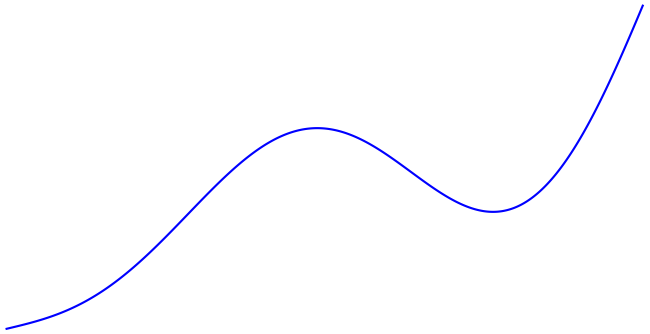
2.2 Macro Implementation

The macro will have 9 arguments. They ares

- #1 Start of x axis.
- #2 End of x axis.
- #3 Function to plot.
- #4 Number of repetitions for multido (here 16). Used in all multido cycles.
- #5 multido initial value for multido for cicle of step lines.
- #6 increment for multido for cicle of step lines. Initial and increment for cicle of vertical lines.
- # increment for multido on vertical lines. Initial for multido on dotted lines.
- #8 step function to the left of the center point
- #9 step function to the right of the center point

The first thing we draw is the function itself. This is done by `\psplot` which draws the graphic. This picks "blue" as its color in the options and takes the first three arguments.

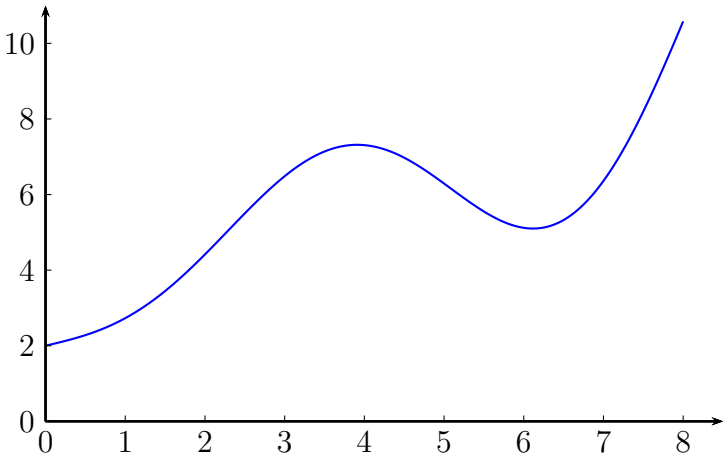
```
\def\RiemannSum#1#2#3#4#5#6#7#8#9
{
     \psplot[linecolor=blue]{#1}{#2}{#3}
}

\begin{center}
\begin{pspicture}(3,0)(4.125,5.5)
    \psset{plotpoints=500, algebraic, dotsize=2.5pt, unit=0.5, xunit=30pt}
     \RiemannSum{0}{8}{x-(x/2)*cos(x)+2}{16}{0.50,0.50}{0.5,0.5}{0.25, 0.50}
          {x+0.25-((x+0.25)/2)*cos(x+0.25)+2}{x-0.25-((x-0.25/2)*cos(x-0.25)+2}
\end{pspicture}
\end{center}
```



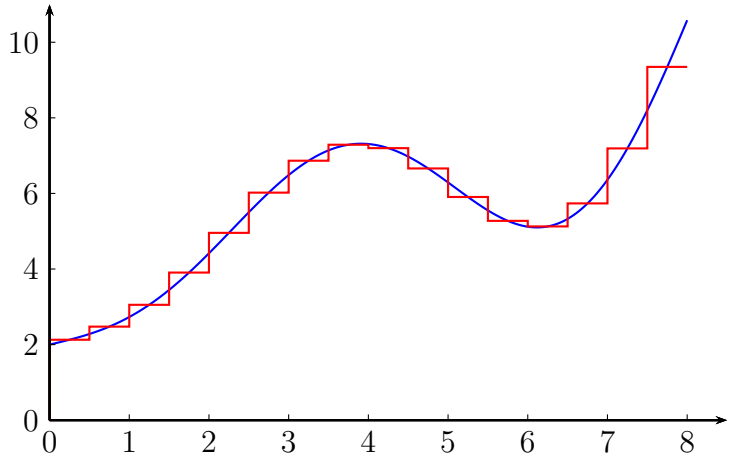
Let us next add the axes on the main program.

```
\begin{center}
\begin{pspicture}(3,0)(4.125,5.5)
    \vspace{2in}
    \psset{plotpoints=500, algebraic, dotsize=2.5pt, unit=0.5, xunit=30pt}
    \RiemannSum{0}{8}{x-(x/2)*cos(x)+2}{16}{0.00,0.50}{0.5,0.5}{0.25,0.50}
      {x+0.25-((x+0.25)/2)*cos(x+0.25)+2}{x-0.25-((x-0.25/2)*cos(x-0.25)+2}
    \psaxes[ticksize=2.2pt, Dy=2, labelsep=4pt]{->}(0,0)(8.5,11)
\end{pspicture}
\begin{center}
```

We will not modify the previous main (driver) routine from now on. Only the Riemann macro.



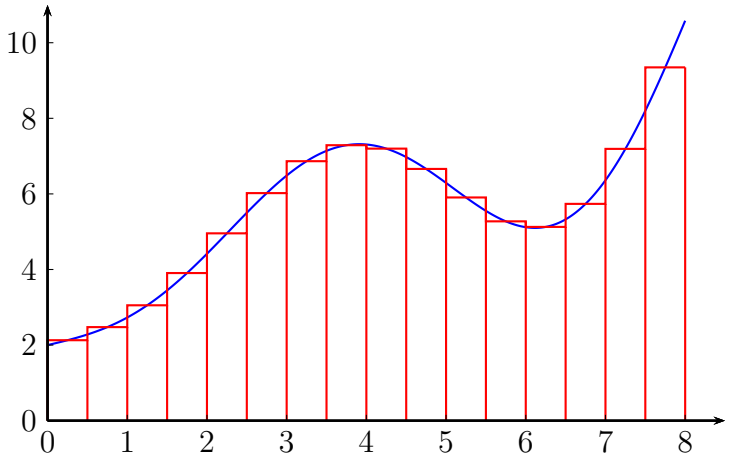
The combination of `\pscustom` and `\multido` serves to draw the segments along the line as follows.




```
\def\RiemannSum#1#2#3#4#5#6#7#8#9
{%
    \psplot[linecolor=blue]{#1}{#2}{#3}
    \pscustom[linecolor=red]
    {%
        \psline{-}(#1,0)(#1,0)
        \multido{
            \ni=#5,\ne=#6
         }{#4}
         {
            \psline(*{\ni} {#8})(*{\ne} {#9})
         }
     }
}
```

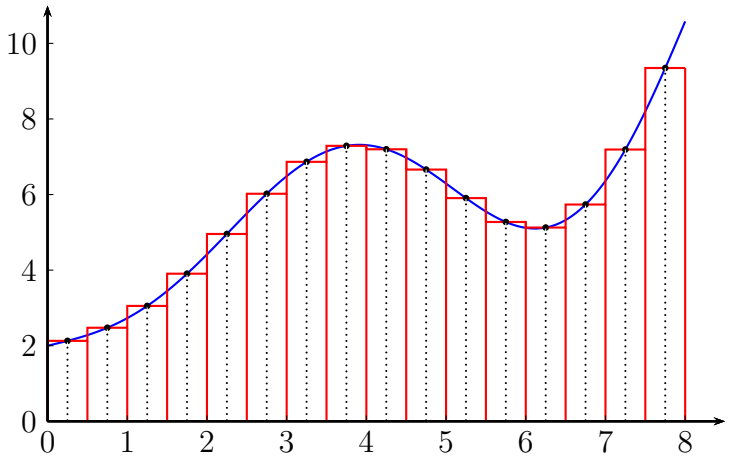
Next we introduce the vertical lines.

```
\def\RiemannSum#1#2#3#4#5#6#7#8#9
{%
    \psplot[linecolor=blue]{#1}{#2}{#3}
    \pscustom[linecolor=red]
    {%
        \psline{-}(#1,0)(#1,0)
        \multido{
            \ni=#5,\ne=#6
         }{#4}
         {
            \psline(*{\ni} {#8})(*{\ne} {#9})
         }
     }
     \multido
     {
        \ne=#6,\nc=#7
     }{#4}
     {
        \psline[linecolor=red](\ne,0)(*{\ne} {#9})
     }
}
```



The final macro is obtained after adding the vertical dotted lines , and function example are given by

```
\def\RiemannSum#1#2#3#4#5#6#7#8#9
{%
    \psplot[linecolor=blue]{#1}{#2}{#3}
    \pscustom[linecolor=red]
    {%
        \psline{-}(#1,0)(#1,0)
        \multido{
            \ni=#5,\ne=#6
         }{#4}
         {
            \psline(*{\ni} {#8})(*{\ne} {#9})
         }
     }
     \multido
     {
        \ne=#6,\nc=#7
     }{#4}
     {
        \psdot(*{\nc} {#3})
        \psline[linestyle=dotted,dotsep=1.5pt](\nc,0)(*{\nc} {#3})
        \psline[linecolor=red](\ne,0)(*{\ne} {#9})
    }
}
```



3 Drawing a vector field of an ordinary differential equation

3.1 Theory

We consider the equation

$$\frac{dy}{dx} = f(x, y). \tag{3.1}$$

We want to draw the tangent vector at each point in a grid. If (x_0, y_0) is a point that satisfies the equation 3.1 and the slope along the solution curve is k, then we say that $k = f(x_0, y_0)$. The tangent vectors paint the vector field and this provides information about the trajectories for the solutions of equation 3.1. We will only worry about the trajectory of the field and not about the actual amplitude. So all tangent vectors are drawn with the same magnitude. In this sense the solution is kinematic, no dynamic. An extension of the macro will include the amplitude $f(x, y)$ of each vector making the vector lengths dynamic. Also, there is not a unique solution unless a boundary (or initial if time is the independent variable) condition is specified.

The method used by Phuong Quan, is based on a grid of points (x_i, y_i), $i = 1, \cdots, m$, $j = 1, \cdots, n$, and the use of polar coordinates. That is he writes

$$\begin{aligned} x &= x_i + r\cos\varphi \\ y &= y_i + r\sin\varphi \end{aligned}$$

we consider $k = f(x_i, y_i) = \tan\varphi$ finite and so we can take $-\pi/2 < \varphi < \pi/2$. From $\sin^2\varphi + \cos^2\varphi = 1$ and $\sin\varphi = k\cos\varphi$,

$$k^2\cos^2\varphi + \cos^2\varphi = 1,$$

so

$$\cos\varphi = \frac{1}{\sqrt{1+k^2}} \qquad \text{and} \qquad \sin\varphi = \frac{k}{\sqrt{1+k^2}}.$$

As explained above all vectors will be consider of equal magnitude (which destroys the dynamics but preserves the kinematics or trajectories). The magnitude assumed is $r = 1/2$. So each vector $\boldsymbol{v}_i$ connects the two points

$$(x_i, y_j), \qquad \left(x_i + \frac{\cos\varphi}{2}\ ,\ y_j + \frac{\sin\varphi}{2}\right) = \left(x_i + \frac{1}{2\sqrt{1+k^2}}\ ,\ y_j + \frac{k}{2\sqrt{1+k^2}}\right).$$

The parametric representation for this segment of line is given, as a function of the paramer t by

$$\begin{aligned} x &= x_i + \frac{t}{2\sqrt{1+k^2}} \quad t \in [0,1] \\ y &= y_i + \frac{tk}{2\sqrt{1+k^2}} \quad t \in [0,1] \end{aligned}$$

3.2 Macro Implementation

We use the macro `\parametricplot` to implement the macro for the vector field. This has the follwing syntax

$$\texttt{\parametricplot}[\text{settings}]\{t_{min}\}\{t_{max}\}\{x(t)|y(t)\}.$$

For the implementation we use the rectangular region

$$R = [a,b] \times [c,d].$$

The grid is confined to points

$$a + 0.25 \leq x_i \leq b - 0.25, \qquad c + 0.25 \leq y_j \leq d - 0.25.$$

With respect to the indices i and j,

$$x_1 = a + 0.25 \qquad y_1 = c + 0.25 \qquad \Delta x = \Delta y = 0.5.$$

We need two loops over i and j, with $0 \leq i \leq m = \lfloor b - a \rfloor$ and $0 \leq j \leq n = \lfloor d - c \leq \rfloor$. The variable declaration for each loop is

$$\begin{aligned} &\texttt{\nx}= \text{initial value} + \text{increment} = x_1 + \Delta x, \\ &\texttt{\ny}= \text{initial value} + \text{increment} = y_1 + \Delta y. \end{aligned}$$

where `\nx` and `\ny` are replaced by x_i and y_i The calling sequence (pseudocode) is given by

```
\multido{y_j = y_1 + Δy}{2n}
{
    \multido{x_i = x_1 + Δx}{2m}
    {
        \parametricplot[settings] {0}{1}
        {
            x_i + t/(2√(1+f(x_i,y_i)^2)) | y_j + t f(x_i,y_j)/(2√(1+f(x_i,y_j)))
        }
    }
}
```

The actual macro is shown next.

```
\def\vecfld#1#2#3#4#5#6
{%
    \multido{#2}{#4}
    {
        \multido{#1}{#3}
        {
            \parametricplot[algebraic,arrows=->,linecolor=red]{0}{1}
            {
                \nx+((#5)*t)*(1/sqrt(1+(#6)^2))|\ny+((#5)*t)*(1/sqrt(1+(#6)^2))*(#6)
            }
        }
    }
}
```

The vector field macro has 6 arguments that I explain next

- #1 Start of the fast grid dimension (inside multido in x)
- #2 Start of the slow grid dimension (outside multido in y)
- #3 number of iterations in the fast dimension
- #4 number of iterations in the slow dimension
- #5 The scalar r which determines the size of the vector. Here $r = 0.5$.
- #6 Here goes $f(x, y)$ which is the slope function.

3.3 Example

We draw the vector field corresponding to the following differential equation.

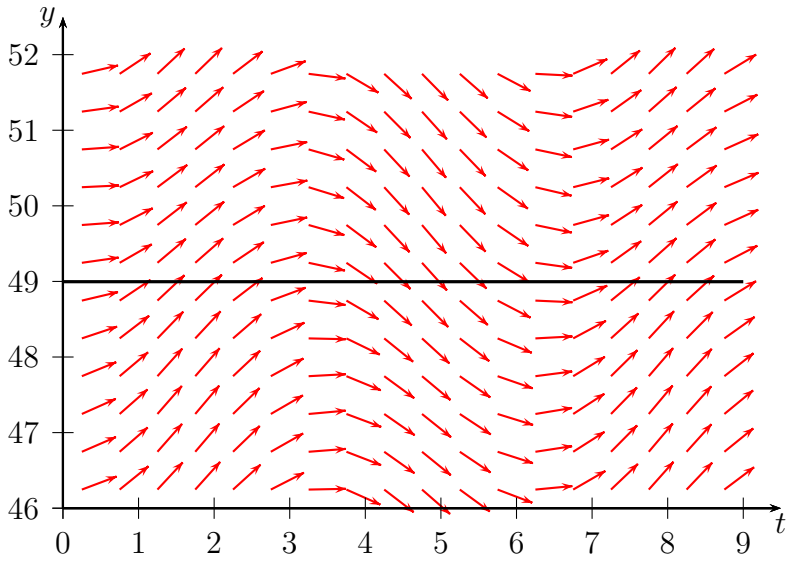
$$\frac{dv}{dt} = \sin(v) - 0.2\cos(t).$$

Note that no boundary (initial) condition is specified, so there is not a unique solution.

The code implementation is next

```
\begin{center}
\begin{pspicture}(0,46)(9.5,52.5)
    \vecfld{\nx=0.25+0.50}{\ny=46.25+0.50}{18}{12}{0.5}{\sin(\nx)-0.2*cos(\ny)}
    \psplot[algebraic,linewidth=1.2pt]{0}{9}{49}
    \psaxes[Dy=1,Dx=1,Oy=46]{->}(0,46)(0,46)(9.5,52.5)
    \rput(9.5,45.8){$t$}\rput(-0.2,52.5){$y$}
    \end{pspicture}
\end{center}
```

and now the figure.



4 Draw an arrow given two points and size

After the previous two macros, this one is piece of cake. Actually I worked copying the two previous macros as a learning excercise.

Let us first study the needed parameters. These are: two points; one origin and the second for direction, and the vector size, for a total of 5 parameteters. The plot as the previous would be a parametric plot. That is if the two points are (x_1, y_1) and $x_2, y_2)$ then the parametric equations of the vector is

$$x = x_1 + r\,t(x_2 - x_1), \qquad y = y_1 + r\,t(y_2 - y_1).$$

where $t \in [0, 1]$.

Let us map the parameters as follows

- # 1 x_1
- # 2 y_1
- # 3 x_2
- # 4 y_2
- # 5 r

The macro should then be

```
\def\vecdraw#1#2#3#4#5#6
{
     \parametricplot[algebraic,arrows=->,linecolor=black]{0}{1}
     {
          #1  + #5*t*(#3 - #1) | #2  + #5*t*(#4 - #2)
     }
}
```

Figure 1 shows an example of its use to draw a simple pendulum.

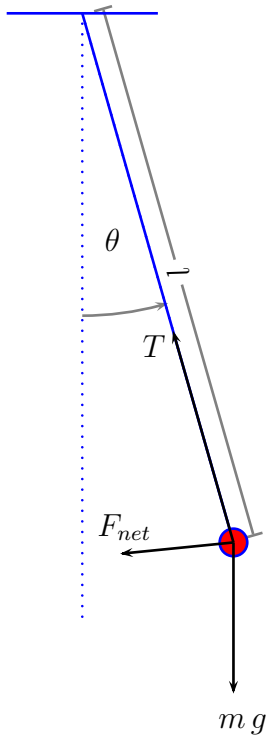


Figure 1: Simple pendulum.

Here the script for the simple pendulum figure. Note the easy use of the `\vecdraw` macro for the three force vectors.

```
\begin{center}
\vspace{1in}
\begin{figure}[!ht]
\vspace{2 in}
\begin{pspicture}(-7,-1)(0,1)
\psset{linecolor=blue, linewidth=1pt, linestyle=dotted, xunit=2.0, yunit=2.0}
\pnode(0,0){A}
\pnode(0,4){B}
\psline(A)(B)
\pnode(1,0.5){A}
\pnode(0,4){B}
\psline(A)(B)
\psset{linecolor=gray, linewidth=1pt,  linestyle=solid}
\degrees[360]
\psarc[arcsepB=2pt]{->}(B){4}{270}{287}
\pcline[offset=-8pt]{|-|}(A)(B)
\lput*{:U}{$l$}
\rput[l](0.15,2.5){$\theta$}
\rput[l](0.4,1.8){$T$}
```

```
\rput[l](0.1,0.6){$F_{net}$}
\rput[l](0.9,-0.7){$m \, g$}
\psset{linecolor=blue, linewidth=1pt, linestyle=solid}
\psline(-0.5,4)(0.5,4)
\psline(0,4)(1,0.5)
\pscircle[fillstyle=solid, fillcolor=red](1,0.5){0.2}
\vecdraw{1}{0.5}{0}{4}{0.4}
\vecdraw{1}{0.5}{-4}{0}{0.15}
\vecdraw{1}{0.5}{1}{0}{2.0}
\end{pspicture}
\vspace{0.5in}
\caption{Simple pendulum.}
\label{simplepen2}
\end{figure}
\end{center}
\normalsize
```