Three applications of macros in PSTricks*

Le Phuong Quan
(Cantho University, Vietnam)
lpquan@ctu.edu.vn

April 3, 2014

Contents

1 Drawing approximations to the area under a graph by rectangles

1.1 Description

We recall here an operation in Calculus. Let $f(x)$ be a function, defined and bounded on an interval $[a, b]$. If f is integrable (in Riemann sense) on $[a, b]$, then its definite integral over this interval is

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

*PSTricks is the original work of Timothy Van Zandt (email address: tvz@econ.insead.fr). It is currently edited by Herbert Voß (hvoss@tug.org).

where $P\colon a = x_0 < x_1 < \cdots < x_n = b$, $\Delta x_i = x_i - x_{i-1}$ is a partition of $[a,b]$, $\xi_i \in [x_{i-1}, x_i]$, $i = 1, 2, \ldots, n$, and $\|P\| = \max\{\Delta x_i\colon i = 1, 2, \ldots, n\}$. Hence, when $\|P\|$ is small enough, we may have an approximation

$$I = \int_a^b f(x)dx \approx \sum_{i=1}^{n} f(\xi_i)\Delta x_i. \tag{1}$$

Because I is independent to the choice of P and ξ_i, we may divide $[a,b]$ into n subintervals with equal length and choose $\xi_i = (x_i + x_{i-1})/2$. Then, I can be approximately seen as the sum of areas of the rectangles with sides $f(\xi_i)$ and Δx_i.

We will make a drawing procedure to illustrate the approximation (1). Firstly, we establish commands to draw the "sum" of rectangles, like the area under piecewise-constant functions (called *step shape*, for brevity). The chosen procedure here obviously includes a combination of the macros `\pscustom` (to *join* horizontal segments, automatically) and `\multido`. In particular, horizontal segments are depicted within the loop `\multido` by

$$\texttt{\textbackslash psplot[}\textit{settings}\texttt{]\{}x_{i-1}\texttt{\}\{}x_i\texttt{\}\{}f(\xi_i)\texttt{\}}$$

Then, `\pscustom` will join these segments altogether with the ending points $(a,0)$, $(b,0)$ to make the boundary of the step shape. Next, we draw the points $(\xi_i, f(\xi_i))$, $i = 1, 2, \ldots, n$, and the dotted segments between these points and the points $(\xi_i, 0)$, $i = 1, 2, \ldots, n$, by

$$\texttt{\textbackslash psdot[algebraic,...](*\{}\xi_i\texttt{\} \{}f(x)\texttt{\})}$$
$$\texttt{\textbackslash psline[algebraic,linestyle=dotted,...]}(\xi_i,0)\texttt{(*\{}\xi_i\texttt{\} \{}f(x)\texttt{\})}$$

where we use the structure `(*{`*value*`} {`$f(x)$`})` to obtain $(\xi_i, f(\xi_i))$. Finally, we draw vertical segments to split the step shape into rectangular cells by

$$\texttt{\textbackslash psline[algebraic,...]}(x_i,0)\texttt{(*\{}x_i\texttt{\} \{}f(x - \Delta x_i/2)\texttt{\})}$$

The process of approximation is depicted in Figure 1.

We now combine the above steps to make a procedure whose calling sequence consists of main parameters a, b, f and n, and dependent parameters x_{i-1}, x_i, ξ_i, $f(\xi_i)$ and $f(x \pm \Delta x_i/2)$. For instance, let us consider approximations to the integral of $f(x) = \sin x - \cos x$ over $[-2, 3]$ in cases of $n = 5$ and $n = 20$. They are given in Figure 2.

In summary, we can make a procedure to illustrate the approximation (1), say `RiemannSum`, whose calling sequence has the form of

$$\texttt{\textbackslash RiemannSum\{}a\texttt{\}\{}b\texttt{\}\{}f(x)\texttt{\}\{}n\texttt{\}\{}x_{\text{ini}}\texttt{\}\{}x_{\text{end}}\texttt{\}\{}x_{\text{choice}}\texttt{\}\{}f(x + \Delta x_i/2)\texttt{\}\{}f(x - \Delta x_i/2)\texttt{\}},$$

where $x_0 = a$ and for each $i = 1, 2 \ldots, n$:

$$x_i = a + \frac{b-a}{n}i, \quad \Delta x_i = x_i - x_{i-1} = \frac{b-a}{n},$$
$$x_{\text{ini}} = x_0 + \Delta x_i, \quad x_{\text{end}} = x_1 + \Delta x_i, \quad x_{\text{choice}} = \frac{x_{\text{ini}} + x_{\text{end}}}{2} = \frac{x_0 + x_1}{2} + \Delta x_i.$$

Note that x_{ini}, x_{end} and x_{choice} are given in such forms to be suitable to variable declaration in `\multido`. They are nothing but x_{i-1}, x_i and ξ_i, respectively, at the step i-th in the loop.

Tentatively, in `PSTricks` language, the definition of `RiemannSum` is suggested to be

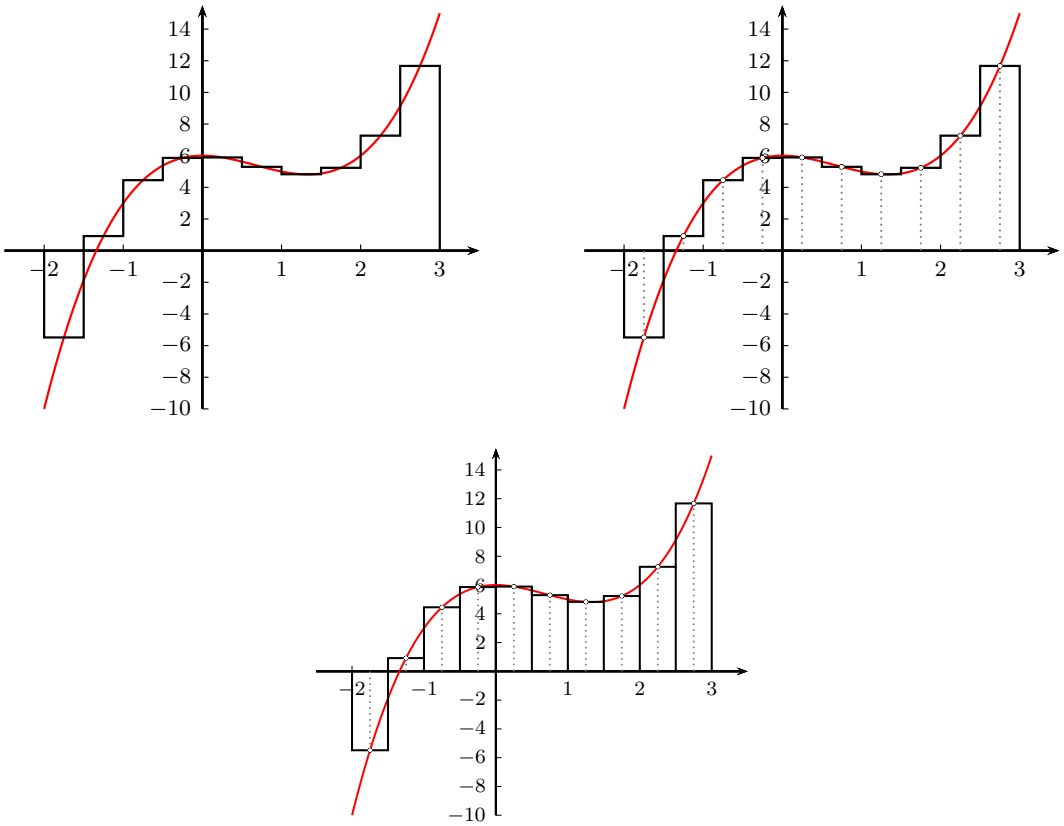

Figure 1: Steps to make the drawing procedure.

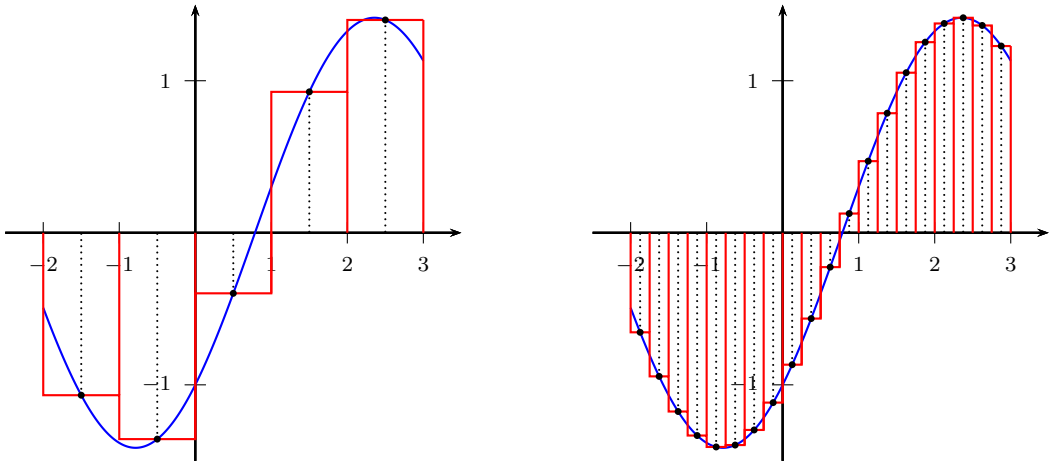

Figure 2: Approximations to the integral of $f(x) = \sin x - \cos x$ over $[-2, 3]$.

```
\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})}}
```

1.2 Examples

We give here two more examples just to see that using the drawing procedure is very easy. In the first example, we approximate the area under the graph of the function $f(x) = x - (x/2)\cos x + 2$ on the interval $[0, 8]$. To draw the approximation, we try the case $n = 16$; thus $x_0 = 0$ and for each $i = 1, \ldots, 16$, we have $x_i = 0.5\,i$, $\Delta x_i = 0.5$, $x_{\text{ini}} = 0.00 + 0.50$, $x_{\text{end}} = 0.50 + 0.50$ and $x_{\text{choice}} = 0.25 + 0.50$.

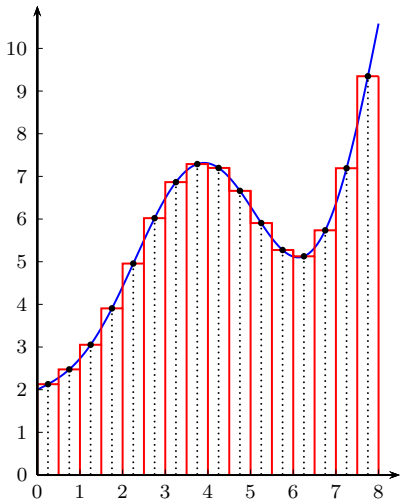

Figure 3: An approximation to the area under the graph of $f(x) = x - (x/2)\cos x + 2$ on $[0, 8]$.

To get Figure 3, we have used the following LaTeX code:

```
\begin{pspicture}(0,0)(4.125,5.5)
\psset{plotpoints=500,algebraic,dotsize=2.5pt,unit=0.5}
\RiemannSum{0}{8}{x-(x/2)*cos(x)+2}{16}{0.00+0.50}{0.50+0.50}{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,labelsep=4pt]{->}(0,0)(8.5,11)
\end{pspicture}
```

In the second example below, we will draw an approximation to the integral of $f(x) = x \sin x$ over $[1, 9]$. Choosing $n = 10$ and computing parameters needed, we get Figure 4, mainly by the

command

```
\RiemannSum{1}{9}{x sin x}{10}{1.00 + 0.80}{1.80 + 0.80}{1.40 + 0.80}
{(x + 0.4) sin(x + 0.4)}{(x - 0.4) sin(x - 0.4)}
```

in the drawing procedure.

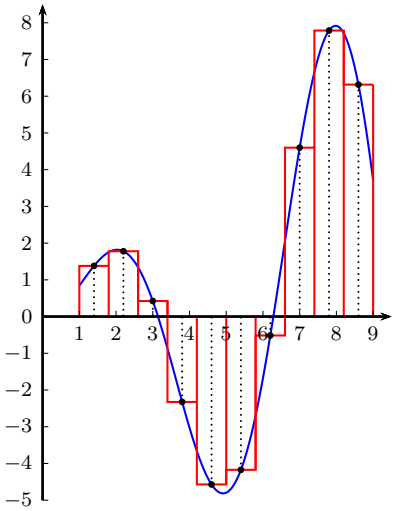

Figure 4: An approximation to the integral of $f(x) = x \sin x$ over $[1, 9]$.

2 Drawing the vector field of an ordinary differential equation of order one

2.1 Description

Let us consider the differential equation

$$\frac{dy}{dx} = f(x, y). \tag{2}$$

At each point (x_0, y_0) in the domain D of f, we will put a vector $\mathbf{v}$ with slope $k = f(x_0, y_0)$. If $y(x_0) = y_0$, then k is the slope of the tangent to the solution curve $y = y(x)$ of (2) at (x_0, y_0). The $\mathbf{v}$'s make a *vector field* and the picture of this field would give us information about the shape of solution curves of (2), even we have not found yet any solution of (2).

The vector field of (2) will be depicted on a finite grid of points in D. This grid is made of lines, paralell to the axes Ox and Oy. The intersectional points of those lines are called *grid points* and often indexed by (x_i, y_j), $i = 0, \ldots, N_x$, $j = 0, \ldots, N_y$. For convenience, we will use polar coordinates to locate the terminal point (x, y) of a field vector, with the initial point at grid point (x_i, y_j). Then, we can write

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

Because $k = f(x_i, y_j) = \tan\varphi$ is finite, we may take $-\pi/2 < \varphi < \pi/2$. From $\sin^2\varphi + \cos^2\varphi = 1$ and $\sin\varphi = k\cos\varphi$, we derive

$$\cos\varphi = \frac{1}{\sqrt{1+k^2}}, \quad \sin\varphi = \frac{k}{\sqrt{1+k^2}}.$$

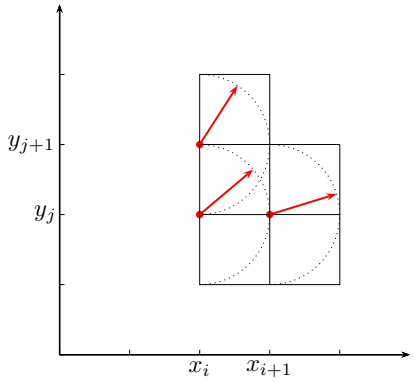

Figure 5: Field vectors on a grid.

The field vectors should all have the same magnitude and we choose here that length to be 1/2, that means $r = 1/2$. Thus, vectors on the grid have their initial and terminal points as

$$(x_i, y_j), \quad \Big(x_i + \frac{1}{2}\cos\varphi, y_j + \frac{1}{2}\sin\varphi\Big),$$

respectively. Hence, we easily get the parametrization of a vector at grid point (x_i, y_j):

$$\begin{aligned} x &= x_i + \frac{t}{2}\cos\varphi = x_i + \frac{t}{2\sqrt{1+k^2}}, \\ y &= y_j + \frac{t}{2}\sin\varphi = y_j + \frac{tk}{2\sqrt{1+k^2}}, \end{aligned}$$

where t goes from 0 to 1, as along the direction of the vector.

Of macros in `PSTricks` to draw lines from their parametrization, we select `\parametricplot`[1] for its fitness. The macro has the syntax

$$\texttt{\textbackslash parametricplot[}\textit{settings}\texttt{]\{}t_{\min}\texttt{\}\{}t_{\max}\texttt{\}\{}x(t)\,\texttt{|}\,y(t)\texttt{\}},$$

where we may use the option `algebraic` to make the declaration of $x(t)$ and $y(t)$ simpler in `ASCII` code.

From the above description of one field vector, we now construct the vector field on a grid within a domain $R = \{(x, y)\colon a \le x \le b,\ c \le y \le d\}$. To determine the grid, we confine grid points to the range

$$a \le x_i \le b, \quad c \le y_j \le d. \tag{3}$$

We start with initial values $x_0 = a$ and $y_0 = c$ to have points (x_i, y_j) with increments $\Delta x = \Delta y = \delta$, corresponding to the length of vectors and the distance between grid points as indicated in Figure 5. Thus, to draw vectors at grid points (x_i, y_j), we need two loops for indices i and j, with $0 \le i \le \lfloor m/\delta \rfloor$, $0 \le j \le \lfloor n/\delta \rfloor$, where $m = b - a$, $n = d - c$. Apparently, these two loops are nested `\multidos`, with variable declaration for each loop as follows

$$\begin{aligned} \texttt{\textbackslash nx} &= \text{initial value} + \text{increment} = x_0 + \Delta x, \\ \texttt{\textbackslash ny} &= \text{initial value} + \text{increment} = y_0 + \Delta y. \end{aligned}$$

[1] This macro is of ones, often added and updated in the package `pstricks-add`, the authors: Dominique Rodriguez (`dominique.rodriguez@waika9.com`), Herbert Voß (`voss@pstricks.de`).

Finally, we will replace \nx, \ny by x_i, y_j in the below calling sequence for simplicity.

Thus, the main command to draw the vector field of the equation (2) on the grid (3) is

$$\texttt{\textbackslash multido}\{y_j = y_0 + \Delta y\}\{\lfloor n/\delta \rfloor\}\Big\{\texttt{\textbackslash multido}\{x_i = x_0 + \Delta x\}\{\lfloor m/\delta \rfloor\}$$
$$\Big\{\texttt{\textbackslash parametricplot}[\textit{settings}]\{0\}\{1\}\Big\{x_i + \frac{t}{2\sqrt{1+\big[f(x_i,y_j)\big]^2}}\Big|\ y_j + \frac{t f(x_i,y_j)}{2\sqrt{1+\big[f(x_i,y_j)\big]^2}}\Big\}\Big\}$$

where we at least use arrows=-> and algebraic for *settings*.

We can combine the steps mentioned above to define a drawing procedure, say \avecfld, that consists of 6 parameters in the order as \nx=$x_0+\Delta x$, \ny=$y_0+\Delta y$, $\lfloor m/\delta \rfloor$, $\lfloor n/\delta \rfloor$, δ and f(\nx,\ny). We may change these values to modify the vector field or to avoid the vector intersection. Such a procedure is suggested to be

```
\def\avecfld#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)}}}}
```

Actually, the procedure \parametricplot is used here only to draw a vector by its parametrization $(x(t), y(t))$, so we can use the structure \curvepnodes in the package pst-node[2] to extract the two ending points of the curve $(x(t), y(t))$ by the command

$$\texttt{\textbackslash curvepnodes[algebraic,plotpoints=2]\{0\}\{1\}\{}x(t)\texttt{|}y(t)\texttt{\}\{P\}},$$

where P is a name of the root of nodes and we just get the two nodes P0, P1 when executing this command. Then, the corresponding vector is drawn by the command

$$\texttt{\textbackslash psline[linecolor=}\textit{settings}\texttt{]\{->\}(P0)(P1)}$$

Therefore, another procedure to draw a vector field may be defined as

```
\def\anothervecfld#1#2#3#4#5#6{%
\multido{#2}{#4}{\multido{#1}{#3}
{\curvepnodes[algebraic,plotpoints=2]{0}{1}
{\nx+((#5)*t)*(1/sqrt(1+(#6)^2))|\ny+((#5)*t)*(1/sqrt(1+(#6)^2))*(#6)}{P}
\psline[linecolor=red]{->}(P0)(P1)}}}
```

2.2 Examples

Firstly, we consider the equation that describes an object falling in a resistive medium:

$$\frac{dv}{dt} = 9.8 - \frac{v}{5}, \tag{4}$$

where $v = v(t)$ is the speed of the object in time t. In Figure 6, the vector field of (4) is given on the grid $R = \{(t, y)\colon 0 \le t \le 9,\ 46 \le v \le 52\}$, together with the graph of the equilibrium solution $v = 49$.

[2]Package authors: Timothy Van Zandt (tvz@econ.insead.fr), Michael Sharpe (msharpe@euclid.ucsd.edu) and Herbert Voß (hvoss@tug.org).

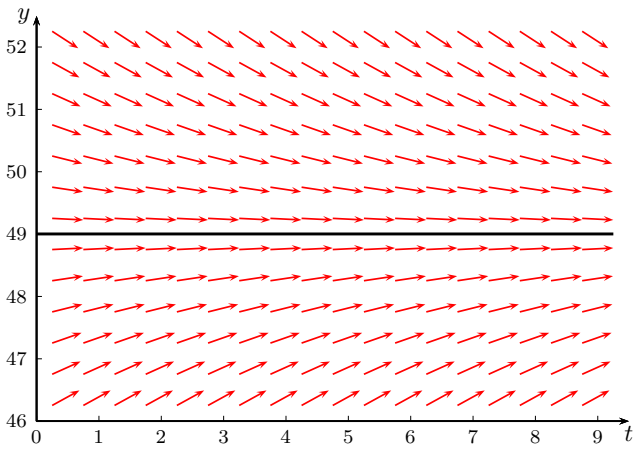

Figure 6: The vector field of (4).

Figure 6 is made of the following LaTeX code:

```
\begin{pspicture}(0,46)(9.5,52.5)
\anothervecfld{\nx=0.25+0.50}{\ny=46.25+0.50}{18}{12}{0.5}{9.8-0.2*\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}
```

Let us next consider the problem

$$\frac{dy}{dx} = x + y, \quad y(0) = 0. \tag{5}$$

It is easy to check that $y = e^x - x - 1$ is the unique solution to (5). We now draw the vector field of (5) and the solution curve[3] on the grid $R = \{(x, y) : 0 \le x \le 3,\ 0 \le y \le 5\}$ in Figure 7.

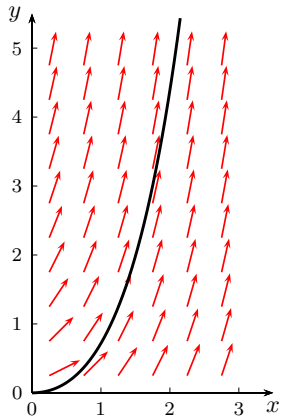

Figure 7: The vector field of (5).

[3]We have used $\mathrm{ch}(1) + \mathrm{sh}(1)$ for the declaration of e, natural base of logarithmic function.

We then go to the logistic equation, which is chosen to be a model for the dependence of the population size P on time t in Biology:

$$\frac{dP}{dt} = kP\Big(1 - \frac{P}{M}\Big), \tag{6}$$

where k and M are constants, respectively various to selected species and environment. For specification, we take, for instance, $k = 0.5$ and $M = 100$. The right hand side of (6) then becomes $f(t, P) = 0.5\,P(1 - 0.01\,P)$. In Figure 8, we draw the vector field of (6) on the grid $R = \{(t, P)\colon 0 \le t \le 10,\ 95 \le P \le 100\}$ and the equilibrium solution curve $P = 100$. Furthermore, with the initial condition $P(0) = 95$, the equation (6) has the unique solution $P = 1900(e^{-0.5t}+19)^{-1}$. This solution curve is also given in Figure 8.

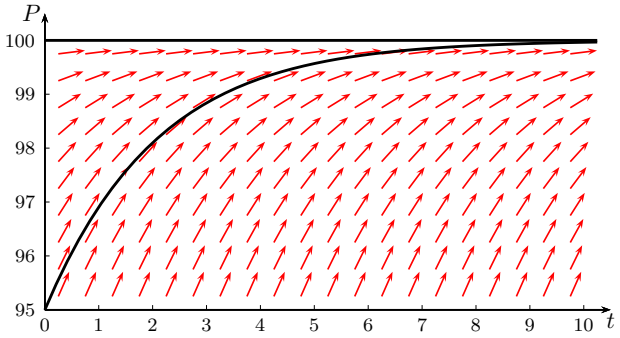

Figure 8: The vector field of (6) with $k = 0.5$ and $M = 100$.

The previous differential equations are all of seperated variable or linear cases that can be solved for closed-form solutions by some simple integration formulas. We will consider one more equation of the non-linear case whose solution can only be approximated by numerical methods. The vector field of such an equation is so useful and we will use the Runge-Kutta curves (of order 4) to add more information about the behaviour of solution curves. Here, those Runge-Kutta curves are depicted by the procedure `\psplotDiffEqn`, also updated from the package `pstricks-add`.

The vector field of the non-linear differential equation

$$\frac{dy}{dx} = y^2 - xy + 1 \tag{7}$$

will be depicted on the grid $R = \{(x, y)\colon\ -3 \le x \le 3, -3 \le y \le 3\}$ and the solutions of Cauchy problems for (7), corresponding to initial conditions

(i) $y(-3) = -1$,

(ii) $y(-2) = -3$,

(iii) $y(-3) = -0.4$,

will be approximated by the method of Runge-Kutta, with the grid size $h = 0.2$. It is very easy to recognize approximate curves, respective to (i), (ii) and (iii) in Figure 9 below.

2.3 Remarks on how to color arrows properly for a vector field

There remains a problem in drawing a vector field. That is coloring arrows. Obviously, their color shade should vary according to their slope and this would give us the picture of domains containing

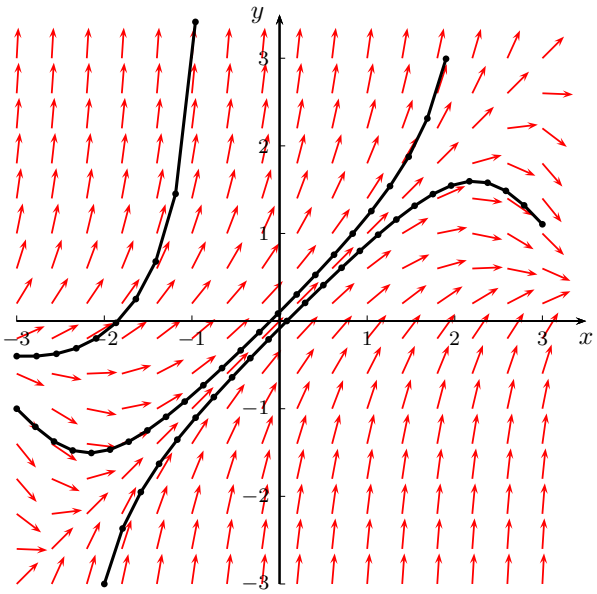

Figure 9: The vector field of (7) and the Runge-Kutta curves.

increase or decrease solutions of a differential equation. In some cases, we even know how large the rate of change of those solutions is in a specific domain.

In Subsection 2.1, we know for the equation (2) that $f(x_i, y_j)$ is right the slope of field vectors at grid points (x_i, y_j), and we will divide these slopes into some number of scales, corresponding to the degree of color shades. Here, we confine our interest to a continuous function $f(x, y)$ in two independent variables on a domain $R = \{(x, y)\colon a \le x \le b,\, c \le y \le d\}$ and choose the scale of 10 degrees. This number of degrees can be changed to any positive integer.

According to the input data from the differential equation (2), the set R and the grid points on it and the value $M = \max\{|f(x_i, y_j)|\colon 0 \le i \le \lfloor m/\Delta x \rfloor,\, 0 \le j \le \lfloor n/\Delta y \rfloor\}$, where $m = b - a$ and $n = d - c$, we can now define the degree of color shade for each arrow in our vector field. It should be an integer n_{ij} such that $n_{ij} = \lfloor 10|f(x_i, y_j)|/M \rfloor$, that is

$$n_{ij}M \le 10|f(x_i, y_j)| < (n_{ij} + 1)M. \tag{8}$$

For finding such an integer, in TEX codes, we need one `\newcount` for it and two `\newdimen` for $f(x_i, y_j)$ and intermediate values to be compared with $|f(x_i, y_j)|$. For more explanation, let us begin with settings `\newcount\intg` (referring (ref.) to "integer"), `\newdimen\slope` (ref. to "slope") and `\newdimen\interm` (ref. to "intermediate values"). Then, the integer n_{ij} at stage (i, j) within the two `\multido` loops can be defined by the recursive macro `\fintg` (ref. to "find the integer") as follows

```
\def\fintg{\interm=Mpt \interm=\intg\interm%
    \ifdim\ifdim\slope<0pt -\fi\slope<\interm\advance\intg by -1\relax
       \else\advance\intg by 1\fintg\fi}
```

where `M` and `\slope` are holding the values M and $f(x_i, y_j)$, respectively. Note that, before running our macro, `\slope` should be multiplied by 10 with the assignment `\slope=10\slope`, as defined in (8). Besides, by simulating the expression of $f(x, y)$, the calculation of $f(x_i, y_j)$ should be declared with operations on `\newcount`s and `\newdimen`s. Then, the integer n_{ij}, which is found at stage

(i, j), should take its degree, say k, from 0 to 10 by its value, suitably associated to the command `\psline[linecolor=red!case-k]{->}(P0)(P1)`. Here, we choose `red` for the main color (it can be changed, of course), and `case-k` will be replaced with an appropriate percentage of `red`. Finally, such a color scale is local and relative, so we can use one more parameter in the procedure to adjust color shades. The old procedures take 6 parameters and the new one will take two more parameters: one for declaration of computing $f(x_i, y_j)$ and the other for adjusting color shades.

Let us take some examples on how to compute $f(x_i, y_j)$ by TEX codes or by commands from the package `calculator`[4]. For a simple polynomial $f(x, y)$, computing $f(x_i, y_j)$ by TEX codes might be facile. Because `\nx` and `\ny` are respectively holding values of x_i and y_j, we need the two corresponding variables `\newdimen\fx` and `\newdimen\fy` to take these values. By assigning `\fx=\nx pt\fy=\ny pt`, we compute $f($`\nx`$,$`\ny`$)$ and assign its value to `\slope`. The declaration of calculations for some cases of $f(x, y)$ is given in the following table.

$f(x, y)$	TEX codes for computing $f($`\nx`$,$`\ny`$)$
$x + y$	`\advance\slope by \fx \advance\slope by \fy`
$1 - xy$	`\advance\slope by -\decimal\fx\fy \advance\slope by 1pt`
$y(3 - y)$	`\advance\slope by -\decimal\fy\fy \advance\slope by 3\fy`
$y^2 - xy$	`\advance\slope by \decimal\fy\fy \advance\slope by -\decimal\fx\fy`

In the table, the command `\decimal`, which is quotative from [5] for producing decimal numbers from dimensions, is put in the preamble using a definition as

```
\def\xch{\catcode`\p=12 \catcode`\t=12}\def\ych{\catcode`\p=11 \catcode`\t=11}
    \xch \def\dec#1pt{#1}\ych \def\decimal#1{\expandafter\dec \the#1}
```

For a transcendental or rational function $f(x, y)$, we may use the package `calculator` for computing $f(x_i, y_j)$. The following table shows how to perform calculations.

$f(x, y)$	The commands from the package `calculator` for computing $f($`\nx`$,$`\ny`$)$
$\sin(y - x)$	`\SUBTRACT{\ny}{\nx}{\sola}\SIN{\sola}{\solb}\slope=\solb pt`

$2xy/(1 + y^2)$	`\SUMfunction{\ONEfunction}{\SQUAREfunction}{\Fncty}`
	`\Fncty{\ny}{\soly}{\Dsoly}\DIVIDE{\Dsoly}{\soly}{\tempa}`
	`\MULTIPLY{\nx}{\tempa}{\tempb}\slope=\tempb pt`

From the old macros `\avecfld` or `\anothervecfld`, we will construct the new one `\vecfldnew` by adding up to the former the two parameters as described above. According to the description of new parameters and of known ones, the calling sequence of `\vecfldnew` may have the form of

`\vecfldnew{\nx=`$x_0 + \Delta x$`}{\ny=`$y_0 + \Delta y$`}{`n_x`}{`n_y`}{`ℓ`}{`$f($`\nx`$,$`\ny`$)$`}{`TEX codes`}{`n_a`}`

where n_a is an estimate value for M and can be adjusted to be greater or less than M. This flexible mechanism might be to increase or decrease the degree of color shades. Finally, `\intg` and `\slope` should be reset to zero at the end of each stage. Now, all materials to make the new macro are ready, and its definition is suggested to be

[4] Package author: Robert Fuster (`rfuster@mat.upv.es`).

```
\def\vecfldnew#1#2#3#4#5#6#7#8{%
\newcount\intg \newdimen\slope \newdimen\interm \newdimen\fx \newdimen\fy
\def\fintg{\interm=#8 \interm=\intg\interm%
    \ifdim\ifdim\slope<0pt -\fi\slope>\interm \advance\intg by 1\fintg\fi}
\multido{#2}{#4}
{\multido{#1}{#3}
{\curvepnodes[algebraic,plotpoints=2]{0}{1}
{\nx+((#5)*t)*(1/sqrt(1+(#6)^2))|\ny+((#5)*t)*(1/sqrt(1+(#6)^2))*(#6)}{P}
#7\slope=10\slope \fintg \ifnum\intg>10\psline[linecolor=red]{->}(P0)(P1)
\else\ifnum\intg=0\psline[linecolor=red!5]{->}(P0)(P1)
\else\multiply\intg by 10\psline[linecolor=red!\the\intg]{->}(P0)(P1)\fi\fi
\intg=0\slope=0pt
}}}
```

If we predefine some scale of degrees, instead of the code `\ifnum\intg>10...\fi\fi`, the structure `\ifcase` can be used as

```
\ifcase\intg
\psline[linecolor=red!5]{->}(P0)(P1)\or
\psline[linecolor=red!10]{->}(P0)(P1)\or
⋮
\psline[linecolor=red]{->}(P0)(P1)\fi
```

The first example is given with two values of n_a to see how different the color shades are between the two cases. The left vector field in Figure 10 is made of the calling sequence

```
\vecfldnew{\nx=-2.00+0.3}{\ny=-2.00+0.3}{14}{14}{0.3}{(\nx)-2*(\ny)}
{\fy=\ny pt \fx=\nx pt \advance\slope by -2\fy \advance\slope by \fx}{7pt}
```

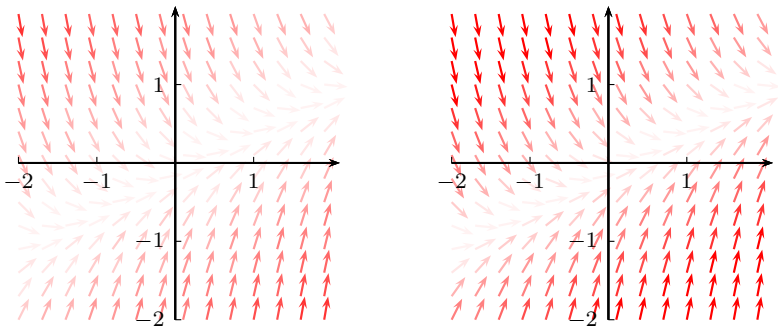

Figure 10: The vector fields of the equation $y' = x - 2y$ with $n_a =$ `7pt` (the left) and $n_a =$ `4pt` (the right)

In Figure 11, the vector fields of the equations $y' = y - x$ and $y' = x(2 - y)$ are respectively drawn by the calling sequences

```
\vecfldnew{\nx=-3.00+0.4}{\ny=-3.00+0.4}{15}{15}{0.35}{(\ny)-(\nx)}
{\fy=\ny pt \fx=\nx pt \advance\slope by -\fx \advance\slope by \fy}{5pt}
```

and

```
\vecfldnew{\nx=-3.00+0.4}{\ny=-3.00+0.4}{15}{15}{0.35}{(\nx)*(2-(\ny))}
{\fy=\ny pt \fx=\nx pt \advance\slope by -\decimal\fx\fy
\advance\slope by 2\fx}{6pt}
```

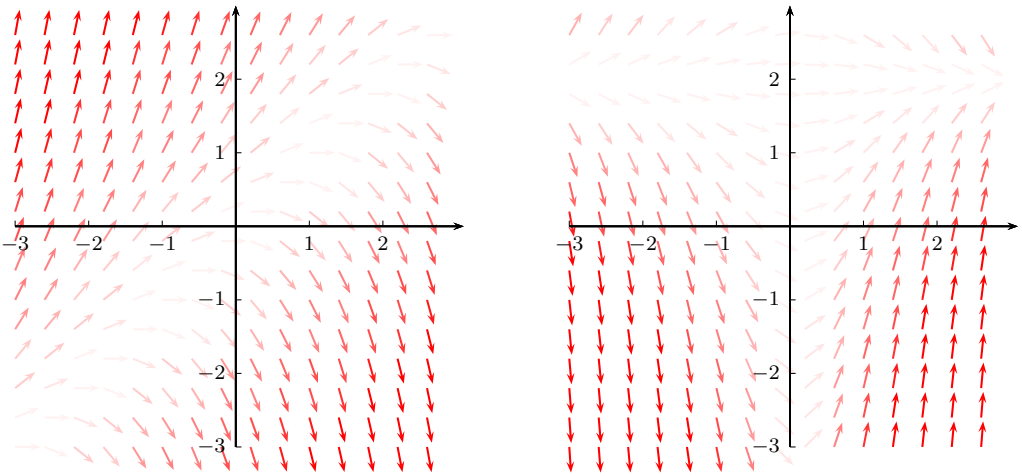

Figure 11: The vector fields of the equation $y' = y - x$ (the left) and $y' = x(2 - y)$ (the right).

Finally, we consider two more examples on vector fields of differential equations $y' = f(x, y)$ containing trigonometric or rational functions on their right side. The calling sequences

```
\vecfldnew{\nx=-3.00+0.4}{\ny=-3.00+0.4}{15}{15}{0.35}{sin(\nx)*cos(\ny)}
{\SIN{\nx}{\tmpa}\COS{\ny}{\tmpb}\MULTIPLY{\tmpa}{\tmpb}{\tmpc}
\slope=\tmpc pt}{0.6pt}
```

and

```
\vecfldnew{\nx=-3.00+0.3}{\ny=-3.00+0.3}{20}{20}{0.3}{2*(\nx)*(\ny)/(1+(\ny)^2)}
{\SUMfunction{\ONEfunction}{\SQUAREfunction}{\Fncty}\Fncty{\ny}{\soly}{\Dsoly}
\DIVIDE{\Dsoly}{\soly}{\tempa}\MULTIPLY{\nx}{\tempa}{\tempb}
\slope=\tempb pt}{2.5pt}
```

respectively result in the vector fields on the left and on the right in Figure 12.

3 Drawing partitions of a simply connected plane domain

3.1 Description

Bézier curves have been used in many different aspects, but mostly in computer graphics to model smooth curves designed by computer programs. We just recall here the vector function of a Bézier curve of degree n with $n+1$ control points whose position vectors are $\mathbf{r}_0$, $\mathbf{r}_1$, ..., $\mathbf{r}_n$. It is the vector function $\mathbf{r}(t)$ in a real variable $t \in [0, 1]$ that is defined as

$$\mathbf{r}(t) = \sum_{i=0}^{n} \binom{n}{i} t^i (1-t)^{n-i} \mathbf{r}_i. \tag{9}$$

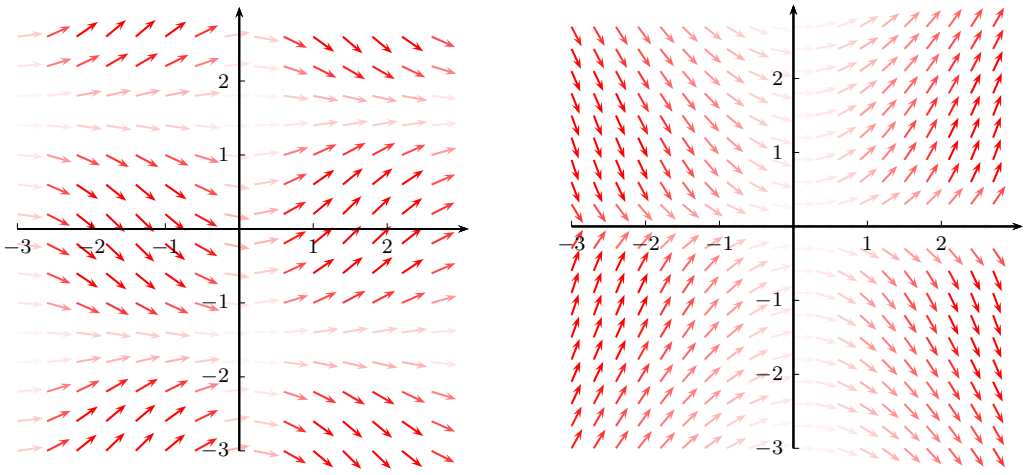

Figure 12: The vector fields of the equation $y' = \sin(x)\cos(y)$ (the left) and $y' = 2xy/(1+y^2)$ (the right).

Since $\mathbf{r}(0) = \mathbf{r}_0$, $\mathbf{r}(1) = \mathbf{r}_n$ and $\mathbf{r}'(0) = n(\mathbf{r}_1 - \mathbf{r}_0)$, $\mathbf{r}'(1) = n(\mathbf{r}_n - \mathbf{r}_{n-1})$, we can take a curve such that it is closed and tangent to a line (Δ) at the point whose position vector is $\mathbf{r}_0$, by letting $\mathbf{r}_n = \mathbf{r}_0$ and the points whose position vectors are $\mathbf{r}_0$, $\mathbf{r}_1$ and $\mathbf{r}_{n-1}$ all be on the line (Δ).

Taking a closed Bézier curve to be the boundary of a plane domain has the great advantage. That is because coordinates of points on the curve are evaluated by polynomials, and the curve has necessary smoothness, beautiful enough for graphic illustrations. `PSTricks` has the procedure `\parametricplot` to draw such a curve. But, an important problem here is how to control the coordinates of its points to construct a procedure for drawing a partition of a simply connected plane domain and coloring (or marking) its cells having common points with the boundary of the domain. We particularly need such a procedure for illustrations of essential notions in measure theory or double integral definition. Because of the limitation in accuracy imposed by the TEX arithmetic, we should mention about 2-dimension vector functions of a Bézier curve of small degree (in fact, as small as possible). According to the above requirements for the closed boundary of a simply connected domain, we will design a family of closed Bézier curves of degree 6, although we may extend their degree to 12.

From the expression of $\mathbf{r}(t)$ in (9), we will construct the following functions and procedures:

- The factorial function `FACTORIAL`.
- The binomial function `BINOMIAL`.
- Polynomials $t^m(1-t)^n$.
- A procedure to compute a sum of 7 terms: values of $\mathbf{r}(t)$ as linear combinations of the Bernstein polynomials
$$\binom{6}{k} t^k (1-t)^{6-k}, \quad k = 0, \ldots, 6.$$

These functions and procedures are macros given by definitions with or without parameters. In the following subsections we will construct them step by step and show how to make connections between them.

3.1.1 Factorial functions and binomial coefficients

To obtain the factorial function, a procedure for multiplying consecutively an integer n with its diminished values will be constructed. For the declared integer variables `\Fa` and `\Fct`, the latter will hold values of the function, and the former will hold factors that are multiplied consecutively and reduced by 1. Namely, if `\Fa` is 0 or 1 then `\Fct` takes 1; else if `\Fa` > 1 then `\Fct` takes the initial value of `\Fa` to begin a loop: diminishing `\Fa` by 1 and updating `\Fct` by multiplying its old value with the new value of `\Fa`. The loop is defined by the control sequence `\Factor` and is ended when `\Fa` is diminished to 1; then, the factorial function obtains its value from the one given back after calling `\FACTORIAL` with one parameter.

```
\newcount\Fa\newcount\Fct
\newcount\tempA
\def\Factor{\ifnum\Fa=1\relax\else\advance\Fa by -1\multiply\Fct by \Fa\Factor\fi}
\def\FACTORIAL#1{\Fa=#1 \ifnum\Fa=0 \Fct=1\relax\else\Fct=\Fa \Factor\fi%
\global\tempA=\Fct}
```

Since TEX limits the largest integer number to $2^{30} - 1$, we can evaluate only to

$$\texttt{\FACTORIAL\{12\}} = 12! = 479001600.$$

Besides, the last value of `\Fct`, which is right the one obtained from calling the function, is assigned to a global integer variable `\tempA` for use in other procedures. Finally, make the calling sequence

```
\FACTORIAL{k}\the\Fct
```

to obtain and show up the value of $k!$. From these values we derive binomial coefficients by the formula

$$\binom{m}{n} = \frac{m!}{n!(m-n)!}.$$

A simple procedure that makes division of $m!$ by $n!$, then of the result by $(m-n)!$ can be performed by the control sequence `\BINOMIAL` with two parameters as follows

```
\newcount\BINOM
\newcount\temp\newcount\tmp
\def\BINOMIAL#1#2{%
\temp=#1\advance\temp by -#2
\FACTORIAL{#1}
\tmp=\tempA
\FACTORIAL{\temp}
\temp=\tempA
\divide\tmp by \temp
\FACTORIAL{#2}
\temp=\tempA
\divide\tmp by \temp\global\BINOM=\tmp}
```

Again, the last value of `\tmp` is assigned to a global integer variable `\BINOM` for later use. Because the vector function of a Bézier curve of degree 6 is only needed, so are the binomial coefficients `\BINOMIAL{6}{`k`}`, $k = 0, 1, \ldots, 6$.

3.1.2 Bernstein polynomials

The construction of Bernstein functions is based on an iterative multiplication of the same value t by a given number m of times, then the process is repeated with the value $1-t$ by a given number n of times. The last result is multiplied by a variable that holds the value of $\binom{m+n}{m}$. We will use an integer variable `\kc` to count times of multiplication for the same factor t or $1-t$. The procedure of iterative multiplication `\xmult` with one parameter m or n is a recursive one that is performed iteratively until `\kc` reaches the value m or n.

```
\newdimen\Xa\newdimen\Yb\newcount\kc
\def\xmult#1{\ifnum\kc<#1\advance\kc by 1\Yb=\decimal\Xa\Yb\xmult{#1}
                                                  \else\relax\fi}
```

Finally, the value of $t^m(1-t)^n$ that is hold by the variable `\Yb` is multiplied by `\BINOM`, the value $\binom{6}{m}$. Thus, Bernstein polynomials are given by the following macro `\BERNSTEIN` with three parameters that hold values of m, n and t, respectively.

```
\newdimen\BSTemp
\def\BERNSTEIN#1#2#3{\Xa=#3pt\kc=0\Yb=1pt\xmult{#1}\kc=0\Xa=-\Xa
\advance\Xa by 1pt\xmult{#2}\BINOMIAL{6}{#1}\global\BSTemp=\BINOM\Yb}
```

Also, values of `\BERNSTEIN` are assigned to a global variable.

3.1.3 Vector functions of a plane Bézier curve of degree 6

We will construct here the vector function of a plane and closed Bézier curve with the control points $M_0(x_0, y_0)$, $M_1(x_1, y_1)$, ..., $M_6(x_6, y_6)$ that is chosen in such a way that the curve is tangent to a line (Δ) at its initial point M_0. According to the mentioned properties of a Bézier curve, the requirements for the curve are satisfied when $M_6 = M_0$ and M_0, M_1, M_5 are all on the line (Δ). In

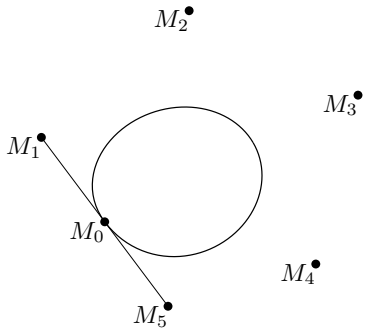

Figure 13: A Bézier curve of degree 6 with the required control points.

case $n = 6$, from (9) we derive the vector function $\mathbf{r}(t)$ with the two components

$$X(t) = \sum_{i=0}^{6} \binom{6}{i} t^i (1-t)^{6-i} x_i, \quad Y(t) = \sum_{i=0}^{6} \binom{6}{i} t^i (1-t)^{6-i} y_i. \tag{10}$$

We first evaluate values of $X(t)$ by the procedures that just have been constructed, with a choice of x_i, $i = 1, \ldots, 6$, from a given x_0 and the requirements of the Bézier curve as in Figure 13. For example, we may take

$$x_1 = x_0 - 1.5, \quad x_2 = x_0 + 2, \quad x_3 = x_0 + 6, \quad x_4 = x_0 + 5, \quad x_5 = x_0 + 1.5, \quad x_6 = x_0.$$

Moreover, we may change the shape of the curve by taking a factor α for $X(t)$. In short, we will make a procedure to evaluate the sum of terms $\binom{6}{i}t^i(1-t)^{6-i}(\alpha x_i)$, $i = 0, 1, \ldots, 6$. In fact, it is a procedure to add up values into a global variable `\XBST` that is called by a macro `\XBC` with three declaring parameters for x_i, t and α. The value of α is used last when being multiplied by a variable that holds $X(t)$. Such a procedure is suggested to be the following macro

```
\newdimen\Xrf\newdimen\Yrf
\newdimen\XBST\newdimen\YBST
\def\XBC#1#2#3{%
\BERNSTEIN{0}{6}{#2}
\Xrf=#1pt
\XBST=\decimal\BSTemp\Xrf
\BERNSTEIN{1}{5}{#2}
\Xrf=#1pt\advance\Xrf by -1.5pt
\advance\XBST by \decimal\BSTemp\Xrf
\BERNSTEIN{2}{4}{#2}
\Xrf=#1pt\advance\Xrf by 2pt
\advance\XBST by \decimal\BSTemp\Xrf
\BERNSTEIN{3}{3}{#2}
\Xrf=#1pt\advance\Xrf by 6pt
\advance\XBST by \decimal\BSTemp\Xrf
\BERNSTEIN{4}{2}{#2}
\Xrf=#1pt\advance\Xrf by 5pt
\advance\XBST by \decimal\BSTemp\Xrf
\BERNSTEIN{5}{1}{#2}
\Xrf=#1pt\advance\Xrf by 1.5pt
\advance\XBST by \decimal\BSTemp\Xrf
\BERNSTEIN{6}{0}{#2}
\Xrf=#1pt
\advance\XBST by \decimal\BSTemp\Xrf
\global\XBST=#3\XBST}
```

In the macro, a control sequence of the form `\decimal\Xdim` is for obtaining the decimal value of a variable `\Xdim` without units (`pt`, by default). This can be defined by

```
\def\xch{\catcode`\p=12 \catcode`\t=12}\def\ych{\catcode`\p=11 \catcode`\t=11}
\xch \def\dec#1pt{#1}\ych \gdef\decimal#1{\expandafter\dec \the#1}
```

Besides, the last line in the definition of `\XBC` (`\global\XBST=#3\XBST`) signifies `\XBST` is adjusted for multiplication by α when holding the current value of the sum $X(t)$. In Table 1, some values of `\XBC` are compared with those of a simpler procedure in Maple[5].

Likewise, from a given y_0, we may have a choice of y_i, $i = 1, \ldots, 6$, as follows

$$y_1 = y_0 + 2, \quad y_2 = y_0 + 5, \quad y_3 = y_0 + 3, \quad y_4 = y_0 - 1, \quad y_5 = y_0 - 2, \quad y_6 = y_0.$$

The macro `\YBC` is constructed in a quite analogous way as for `\XBC` to evaluate values of $Y(t)$ in (10), and the value $\beta Y(t)$ (β is again an adjustment factor) is hold by a global variable `\YBST`.

[5]Maple is a computer algebra system. It was first developed in 1980 by the Symbolic Computation Group at the University of Waterloo in Waterloo, Ontario, Canada. Maple supports numeric and symbolic computations and can be used as a programming language, which resembles Pascal.

	$(-2.4, 0.23, 1.5)$	$(4.2, 0.45, 1.1)$	$(-3.4, 0.7, 1.24)$	$(-0.89, 0.4, 1.5)$	$(5.21, 0.5, 2.15)$
\XBC	-2.41216	8.12482	-0.13675	2.78271	18.75995
Maple	-2.41044	8.13235	-0.13696	2.78628	18.76008

Table 1: A comparison between some values of \XBC and those of a procedure in Maple.

3.1.4 Partition of a simply connected plane domain by a rectangular grid

We consider here a simply connected plane domain D surrounded by a Bézier curve of degree 6 that will be depicted by the procedure \NetDraw. This is called together with its five parameters whose values are assigned to the five local variables with their following meaning:

\CellNum: The number of horizontal and vertical cells of the grid.
\Xref: The value of x_0.
\Yref: The value of y_0.
\Xfact: The value of α.
\Yfact: The value of β.

We denote by R a rectangular domain containing D and having edges parallel to the coordinate axes. A grid of rectangular cells will be depicted on R. According to the given values of x_0, y_0, α and β, the vertices of R can be chosen as the points (\XMin, \YMax), (\XMax, \YMax), (\XMax, \YMin), (\XMin, \YMin), where

$$\texttt{\XMin} = \alpha(\texttt{\Xref} - 1.5), \qquad \texttt{\XMax} = \alpha(\texttt{\Xref} + 6),$$
$$\texttt{\YMin} = \beta(\texttt{\Yref} - 2), \qquad \texttt{\YMax} = \beta(\texttt{\Yref} + 5).$$

Then, the horizontal step size \Xsize and the vertical step size \Ysize may be taken as

$$\texttt{\Xsize} = \frac{\texttt{\XMax} - \texttt{\XMin}}{\texttt{\CellNum}}, \quad \texttt{\Ysize} = \frac{\texttt{\YMax} - \texttt{\YMin}}{\texttt{\CellNum}}.$$

At first, the grid is made by using the structure \multido to draw vertical and horizontal lines whose equations are

$$x = \texttt{\XMin} + i\texttt{\Xsize}, \quad y = \texttt{\YMin} + i\texttt{\Ysize}, \quad i = 0, \ldots, \texttt{\CellNum}.$$

Next, the Bézier curve is depicted by the command

$$\texttt{\parametricplot[algebraic,fillstyle=solid]\{0\}\{1\}\{}\alpha X(t)\,|\,\beta Y(t)\texttt{\}}$$

and it is approximated by the sequence of points (X_i, Y_i), $i = 1, \ldots, N$, where X_i and Y_i are given from the calling sequences

$$\texttt{\XBC\{\XRef\}\{}t_i\texttt{\}\{\Xfact\}}, \quad \texttt{\YBC\{\YRef\}\{}t_i\texttt{\}\{\Yfact\}}.$$

Finally, the most important algorithm is to determine cells that have points in common with the Bézier curve, and to color them. The chosen "filter" method here can be described as follows: for each point (X_i, Y_i) of the approximate sequence, $i = 1, \ldots, N$, we examine cells from left to right in horizontal direction, and from below to above in vertical direction. As soon as being found, the cell containing (X_i, Y_i) is specifically colored by the command \pspolygon[fillstyle=solid,...] passing its four vertices; then, we examine the next point (X_{i+1}, Y_{i+1}), and so on. Actually, a structure of three nested \multido loops is used to perform this filter method, and the loop for counting points (X_i, Y_i) is the most outer one. The algorithm to examine if a cell contains (X_i, Y_i) may have the form of

```
\ifdim\YBST<y\relax\else\advance y by \Ysize
 \ifdim\YBST>y\relax\else
  \ifdim\XBST<x\relax\else\advance x by \Xsize
   \ifdim\XBST>x\relax\else
    \pspolygon[fillstyle=solid,fillcolor=red]
      (x,y)(x+\Xsize,y)(x+\Xsize,y+\Ysize)(x,y+\Ysize)(x,y)\relax
       \fi\fi\fi\fi
```

The single characters "x", "y" and the opertation "+" will be replaced appropriately in the procedure `\NetDraw` below. For increasingly chosen values of N, we may know if the set of cells having points in common with the curve can cover the curve itself? On the other hand, to increase the number of approximate points is corresponding to reduce the increment in the most outer loop `\multido`. For instance, if we take $N = 100$, then the variable `\nz` in the loop should be declared as `\nz` $= 0.00+0.01$ because $t_i \in [0, 1]$.

In summary, to illustrate a partition of a simply connected plane domain, we can apply the following procedure `\NetDraw`. As mentioned above, step by step, `\NetDraw` can: draw the boundary of a plane domain D by a Bézier curve of degree 6, draw a partition of a rectangle R containing D by a grid of rectangular cells, and color cells having points in common with the boundary of D.

```
\def\NetDraw#1#2#3#4#5{%
\newcount\CellNum
\newdimen\XRef
\newdimen\YRef
\newdimen\Xfact
\newdimen\Yfact
\newdimen\XMin
\newdimen\XMax
\newdimen\YMin
\newdimen\YMax
\newdimen\Xsize
\newdimen\Ysize
\newdimen\tempx
\newdimen\tempy
\CellNum=#1
\XRef=#2pt
\YRef=#3pt
\Xfact=#4pt
\Yfact=#5pt
\XMin=\decimal\Xfact\XRef \advance\XMin by -1.50\Xfact
\XMax=\decimal\Xfact\XRef \advance\XMax by 6.00\Xfact
\YMin=\decimal\Yfact\YRef \advance\YMin by -2.00\Yfact
\YMax=\decimal\Yfact\YRef \advance\YMax by 5.00\Yfact
\Xsize=\XMax \advance\Xsize by -\XMin \divide\Xsize by \CellNum
\Ysize=\YMax \advance\Ysize by -\YMin \divide\Ysize by \CellNum
\parametricplot[algebraic,fillstyle=solid,fillcolor=yellow!80,plotpoints=200,
linewidth=0.5pt]{0}{1}{\XBsix{\decimal\XRef}{\decimal\Xfact}|
\YBsix{\decimal\YRef}{\decimal\Yfact}}
```

```
\multido{\nz=0.00+0.005}{200}{\XBC{\decimal\XRef}{\nz}{\decimal\Xfact}
\YBC{\decimal\YRef}{\nz}{\decimal\Yfact}
\multido{\nx=\decimal\XMin+\decimal\Xsize}{\the\CellNum}
{\tempx=\nx pt\multido{\ny=\decimal\YMin+\decimal\Ysize}{\the\CellNum}
{\tempy=\ny pt\ifdim\YBST<\tempy\relax\else\advance\tempy by \Ysize%
              \ifdim\YBST>\tempy\relax\else
              \ifdim\XBST<\tempx\relax\else\advance\tempx by \Xsize
              \ifdim\XBST>\tempx\relax\else
              \pspolygon[fillstyle=solid,fillcolor=blue!90,linecolor=black,
              linewidth=0.2pt]
              (\nx,\ny)(\decimal\tempx,\ny)(\decimal\tempx,\decimal\tempy)
              (\nx,\decimal\tempy)(\nx,\ny)\relax\fi\fi\fi\fi}
}}
\advance\CellNum by 1
\multido{\nx=\decimal\XMin+\decimal\Xsize}{\the\CellNum}
{\psline[linewidth=0.2pt](\nx,\decimal\YMax)(\nx,\decimal\YMin)}
\multido{\ny=\decimal\YMin+\decimal\Ysize}{\the\CellNum}
{\psline[linewidth=0.2pt](\decimal\XMin,\ny)(\decimal\XMax,\ny)}
\parametricplot[algebraic,linecolor=white,plotpoints=200,linewidth=0.5pt]{0}{1}
{\XBsix{\decimal\XRef}{\decimal\Xfact}|\YBsix{\decimal\YRef}{\decimal\Yfact}}}
```

In the definition of `\NetDraw`, the expressions $\alpha X(t)$, $\beta Y(t)$ are declared in the `algebraic` form by the macros `\XBsix`, `\YBsix` with two parameters that hold values of x_0, α and y_0, β, respectively. These macros are given by the following definitions:

```
\def\XBsix#1#2{%
(1-t)^6*(#1)*(#2)+6*t*(1-t)^5*(#1-1.5)*(#2)+15*t^2*(1-t)^4*(#1+2)*(#2)+
20*t^3*(1-t)^3*(#1+6)*(#2)+15*t^4*(1-t)^2*(#1+5)*(#2)+
6*t^5*(1-t)*(#1+1.5)*(#2)+t^6*(#1)*(#2)}
\def\YBsix#1#2{%
(1-t)^6*(#1)*(#2)+6*t*(1-t)^5*(#1+2)*(#2)+15*t^2*(1-t)^4*(#1+5)*(#2)+
20*t^3*(1-t)^3*(#1+3)*(#2)+15*t^4*(1-t)^2*(#1-1)*(#2)+
6*t^5*(1-t)*(#1-2)*(#2)+t^6*(#1)*(#2)}
```

3.2 Examples

Firstly, let us see the effect of change for the shape of a domain when adjusting its boundary by taking different values of α and β. The calling sequence for this purpose may have the form of

```
\parametricplot[algebraic,plotpoints=200,linewidth=0.5pt]{0}{1}
{XBsix{x_0}{α}|\YBsix{y_0}{β}}
```

and its result is given in Figure 14.

The Table 2 below provides three partitions of a domain D with cells that decrease in size. In each case, `\NetDraw` can determine the cells that have points in common with the boundary curve and color them. The calling sequence here takes a simple form

```
\NetDraw{c}{x_0}{y_0}{α}{β}
```

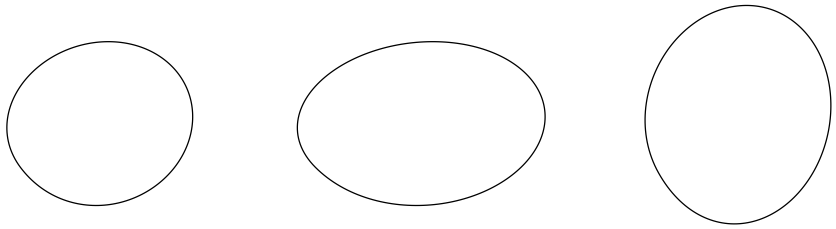

Figure 14: From left to right, corresponding to the couple of values: $\alpha = 2$, $\beta = 2$; $\alpha = 2.5$, $\beta = 2$; $\alpha = 2$, $\beta = 2.5$.

The numerical argument c is assigned to the local variable `\CellNum`. The number of approximate points (X_i, Y_i) for the boundary curve is chosen to be 200, by default, corresponding to the increment $\Delta \mathbf{z} = 0.005$. The number N can be made larger and the filter procedure for determining required cells may be more exact. In Table 2, the Bézier curves are depicted with the choice $x_0 = y_0 = \alpha = \beta = 2$.

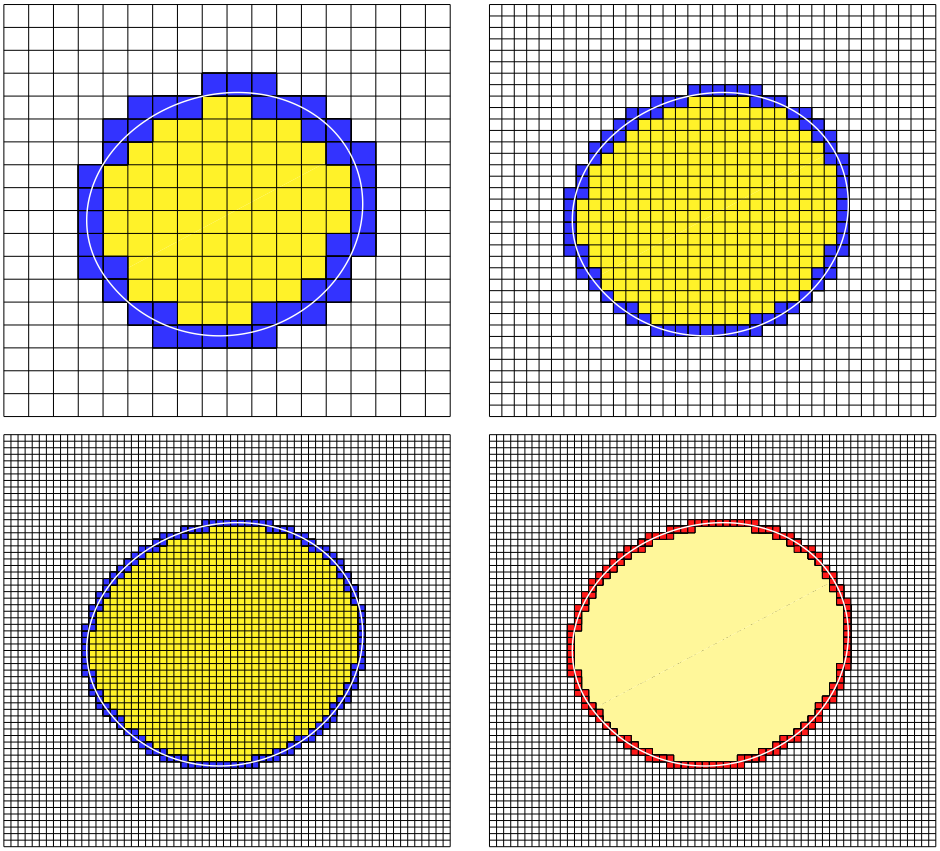

Table 2: The partitions of D with $c = 18$, $c = 36$ and $c = 63$.

Acknowledgment

I am very grateful to

- Timothy Van Zandt, Herbert Voß, Dominique Rodriguez and Michael Sharpe for helping me with their great works on `PSTricks`.
- Hàn Thế Thành for helping me with his pdfLaTeX program.
- Robert Fuster for his very useful package `calculator`.

References

[1] Dominique Rodriguez, Michael Sharpe & Herbert Voß. *`pstricks-add`: Additional Macros for PSTricks*. Version 3.60, http://ctan.org/tex-archive/graphics/pstricks/contrib, 2013

[2] Timothy Van Zandt, Michael Sharpe & Herbert Voß. *`pst-node`: Nodes and node connections*. Version 1.29, http://ctan.org/tex-archive/graphics/pstricks/contrib, 2013

[3] Helmut Kopka & Patrick W. Daly. *Guide to LaTeX*. Addison-Wesley, Fourth Edition, 2004, ISBN 0321173856

[4] Timothy Van Zandt. *User's Guide*. Version 1.5, http://ctan.org/tex-archive/graphics/pstricks/base, 2007

[5] Eitan M. Gurari. *Writing With TeX*, McGraw-Hill, Inc., 1994, ISBN 0-07-025207-6

[6] Robert Fuster. *`calculator-calculus`: Scientific Calculations With LaTeX*. Version 1.0a, http://ctan.org/tex-archive/macros/latex/contrib/calculator, 2012