Enchevêtrements avec LaTeX

Vincent Zoonekynd

zoonek@math.jussieu.fr

19 juin 2000

Résumé

Voici quelques macros qui utilisent Métapost pour réaliser des enchevêtrements — ces diagrammes que l'on rencontre quand on étudie des catégories tressées. Je ne suppose aucune connaissance préalable de Métapost.

1 Installation

Il faut mettre le fichier `tangle.sty` là où LaTeX saura le trouver et le fichier `tangle.mp` là où Métapost saura le trouver. La compilation d'un fichier LaTeX utilisant ces macros peut se dérouler de deux manières différentes.

- Si vous êtes sur une machine Unix (et si vous avez confiance en les macros que vous utilisez), il suffit de compiler deux fois le fichier, en donnant l'option `--shell-escape` au moins la première fois. Cela autorise LaTeX à lancer des commandes Unix, en l'occurrence Métapost.
- Si vous n'êtes pas sur une machine Unix (ou si vous n'avez pas confiance), compilez une première fois, lancez métapost sur le fichier Métapost qui a été créé et relancez LaTeX :

```
latex toto.tex
mpost toto_tangle_output.tex
latex toto.tex
```

La compilation du fichier Métapost peut être *très* longue.

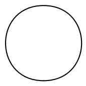
2 Utilisation

Pour charger les macros, il suffit de mettre la ligne

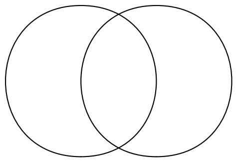
```
\usepackage{tangle}
```

dans l'en-tête du fichier.

Les enchevêtrements sont décrits dans un environement `tangle`. On décrit chaque courbe dans le premier argument de la commande `\c` (pour l'instant, le second reste vide).

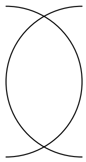


```
$$
\begin{tangle}
  \c{ (0,0) .. (0,1) .. (0,0) }{}
\end{tangle}
$$
```

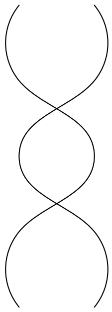


```
$$
\begin{tangle}
  \c{ (0,0) .. (0,2) .. (0,0) }{}
  \c{ (1,0) .. (1,2) .. (1,0) }{}
\end{tangle}
$$
```

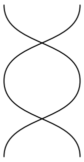
Pour tracer une courbe, on donne les coordonnées (sans unité !) des points de contrôle et on les sépare par deux points (..). On peut en fait mettre n'importe quel code Métapost qui définisse une courbe. Je donne quelques exemples, mais je vous renvoie à une documentation de Métapost pour mieux comprendre.



```
$$
\begin{tangle}
  \c{ (0,0) .. (1,1) .. (0,2) }{}
  \c{ (1,0) .. (0,1) .. (1,2) }{}
\end{tangle}
$$
```



```
$$
\begin{tangle}
  \c{ (0,0) .. (0,1) .. (1,2) .. (0,3) .. (0,4) }{}
  \c{ (1,0) .. (1,1) .. (0,2) .. (1,3) .. (1,4) }{}
\end{tangle}
$$
```

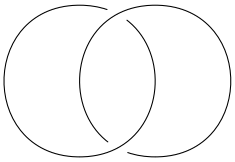


```
$$
\begin{tangle}
  \c{ (0,0){up} .. (1,1) .. (0,2){up} }{}
  \c{ (1,0){up} .. (0,1) .. (1,2){up} }{}
\end{tangle}
$$
```

Il manque quelques exemples.

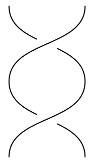
3 Enchevêtrements

C'est le second argument de la commande `\c` qui permet de préciser quelle courbe passe en dessous de quelle autre. Cet argument est une chaine de caractères, un `+` signifie que la courbe passe au dessus, un `-` qu'elle passe en dessous.



```
$$
\begin{tangle}
```

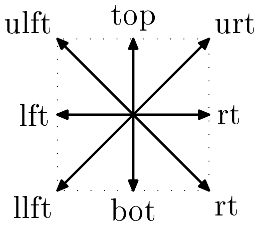
```
  \c{ (0,0) .. (0,2) .. (0,0) }{+-}
  \c{ (1,0) .. (1,2) .. (1,0) }{+-}
\end{tangle}
$$
```



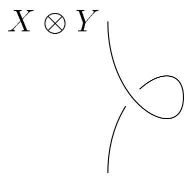
```
$$
\begin{tangle}
  \c{ (0,0){up} .. (1,1) .. (0,2){up} }{+-}
  \c{ (1,0){up} .. (0,1) .. (1,2){up} }{-+}
\end{tangle}
$$
```

4 Labels

La commande `\l` (comme *label*) permet d'écrire le long des courbes, pour donner le nom des objets. Son premier argument, facultatif, est la position du label, `lft` (valeur par défaut), `rt`, `top`, `bot`, `llft`, `lrt`, `ulft` ou `urt`.



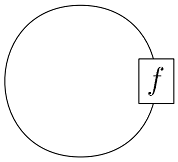
Le second argument est le label, par défaut en mode mathématique (on peut le changer en redéfinissant la commande `\maketanglelabel`).




```
$$
\begin{tangle}
  \c{ (0,0){down} .. (1,-1){up} .. (0,-2){down} }{+-}
  \l[lft]{0}{X \otimes Y}
\end{tangle}
$$
```

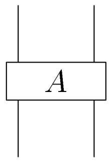
5 Boites

La commande `\b` permet de mettre du texte dans des *boites* sur les courbes, pour désigner des morphismes. Le premier argument est la position sur la courbe (il s'agit de la dernière courbe tracée), le second est le texte à mettre dans la boite. Par défaut, ce texte est en mode mathématique (mais on peut changer ce comportement en redéfinissant la commande `\maketangleboxlabel`). S'il comporte des commandes « fragiles », il convient de les faire précéder de `\protect`.

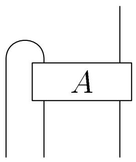
$\operatorname{tr} f =$ 


```
$$
\operatorname{tr} f =
\begin{tangle}
  \c{ (0,0) .. (0,2) .. cycle }{}
  \b{.5}{f}
\end{tangle}
$$
```

Il est possible de tracer des boites « larges », à cheval sur plusieures courbes, pour désigner des morphismes entre produits tensoriels : le premier argument de la commande `\b` contient, séparés par des virgules, la position sur une courbe, suivie du numéro de la courbe, suivi d'une autre position, d'un autre numero de courbe, etc.



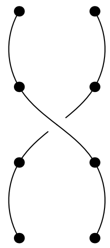

```
$$\begin{tangle}
  \c{ (0,0) -- (0,2) }{}
  \c{ (1,0) -- (1,2) }{}
  \b{.5,0,.5,1}{A}
\end{tangle}$$
```



```
$$\begin{tangle}
  \c{ (0,0) .. (0,1) .. (0,1.3){up} .. (-.5,1.3){down} .. (-.5,0) }{}
  \c{ (1,0) -- (1,2) }{}
  \b{1,0,.5,1}{A}
\end{tangle}$$
```

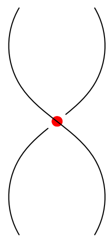
6 Options diverses

La commande `\ShowControlPoints` permet d'afficher les points de contrôle de la figure courrante.



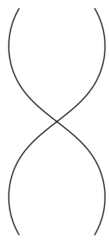
```
$$
\begin{tangle}
  \ShowControlPoints
  \t{d,rd,d}{+}
  \t[1,0]{d,ld,d}{-}
\end{tangle}
$$
```

La commande `\ShowIntersectionPoints` permet d'afficher les points d'intersection.



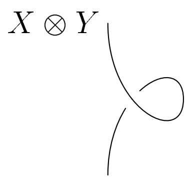
```
$$
\begin{tangle}
  \ShowIntersectionPoints
  \t{d,rd,d}{+}
  \t[1,0]{d,ld,d}{-}
\end{tangle}
$$
```

La commande \DoNotComputeIntersections demande à Métapost de ne pas calculer les points d'intersection, afin d'accélérer le tracé.

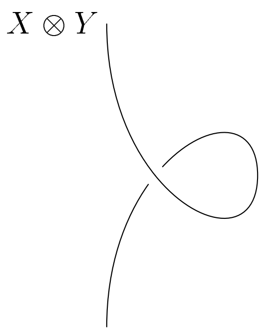


```
$$
\begin{tangle}
  \DoNotComputeIntersections
  \t{d,rd,d}{+}
  \t[1,0]{d,ld,d}{-}
\end{tangle}
$$
```

Il est possible de changer les unités du tracé, à l'aide de la commande \u. Les commandes \ux et \uy permettent de changer séparément l'unité en abscisse et en ordonnée. L'unité par défaut est 1cm.

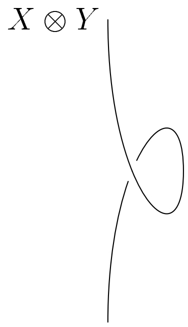



```
$$
\begin{tangle}
  \c{ (0,0){down} .. (1,-1){up} .. (0,-2){down} }{+-}
  \l[lft]{0}{X \otimes Y}
\end{tangle}
$$
```




```
$$
```

```
\begin{tangle}
  \u{2cm}
  \c{ (0,0){down} .. (1,-1){up} .. (0,-2){down} }{+-}
  \l[lft]{0}{X \otimes Y}
\end{tangle}
$$
```

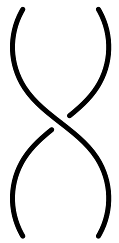



```
$$
\begin{tangle}
  \uy{2cm}
  \c{ (0,0){down} .. (1,-1){up} .. (0,-2){down} }{+-}
  \l[lft]{0}{X \otimes Y}
\end{tangle}
$$
```

Ces commandes ont un effet *local*. Nous verrons dans la section suivante comment rendre ces modifications globales.

7 Si vous connaissez Métapost

La commande `\mp` permet d'insérer du code Métapost quelconque.



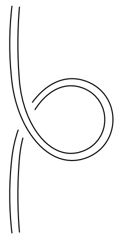
```
$$
\begin{tangle}
  \mp{pickup pencircle scaled 2bp;}
  \t{d,rd,d}{+}
```

```
  \t[1,0]{d,ld,d}{-}
\end{tangle}
$$
```

On peut ainsi rendre globaux les changements de certains paramètres.

```
\mp{ tangle_default_unit_x := 1cm; }
\mp{ tangle_default_unit_y := 1cm; }
\mp{ tangle_default_draw_intersection_points := false; }
\mp{ tangle_default_draw_control_points      := false; }
\mp{ tangle_default_compute_intersections    := true; }
```

On peut aussi utiliser la commande `\mp` pour construire de nouvelles courbes, qui ne rentrent pas le cadre restreint de la commande `\t`. Il faudrait pour cela une documentation des macros Métapost que j'ai écrites...



```
$$\begin{tangle}
  \t{d,dr,u,dl,d}{++--}
  \mp{
    path p,q;
    p:=_tangle_curve[0];
    q:= for i=0 upto length(p)-1:
        ( (point i of p) +
          1mm*unitvector(direction i of p rotated 90)) ..
      endfor (point length(p) of p +
                1mm*unitvector(direction length(p) of p rotated 90));
    curve(q, "++--");
    }
\end{tangle}$$
```

8 Source LaTeX

```
\ProvidesFile{tangle.sty}
\ProvidesPackage{tangle}

%\RequirePackage{verbatim}
\RequirePackage{graphicx}

\newcounter{tangle@n}
\setcounter{tangle@n}{0}
\newwrite \tangle@out
\immediate\openout \tangle@out \jobname_tangle_output.mp
\immediate\write\tangle@out{input tangle;}
\immediate\write\tangle@out{verbatimtex}
\immediate\write\tangle@out{\string\documentclass[12pt]{article}}
\immediate\write\tangle@out{\string\usepackage[T1]{fontenc}}
\immediate\write\tangle@out{\string\usepackage{amsmath,amsfonts}}
\immediate\write\tangle@out{\string\usepackage[T1]{fontenc}}
\immediate\write\tangle@out{\string\begin{document}}
\immediate\write\tangle@out{etex}
\immediate\write\tangle@out{pickup pencircle scaled 1bp;}

\def\maketanglelabel#1{\ensuremath{\displaystyle #1}}
\def\maketangleboxlabel#1{\ensuremath{\displaystyle #1}}

\newenvironment{tangle}{%
  \begingroup
  \csname english\endcsname % Problèmes avec French
  % On pourrait penser résoudre ce problème avec \@gG, mais cela ne
  % marche qu'avec les macros à un argument (ou deux dont un
  % facultatif) :  dans french.doc, on trouve les lignes suivantes.
  %   General code for generating replacement macros for \cite \nocite etc.
  %   \@gG{a string "s" for letting \@"s"@ORI as the original macro}
  %       {original macro name -without backslash}
  %       {string "/" if original macro had no [optional parm] otherwise empty}
  %       {number of other parms} % not implemented now
  \let\mp\relax
  \newcommand{\mp}[1]{\immediate\write\tangle@out{##1}}%
  \let\t\relax
  \newcommand{\t}[3][0,0]{\mp{ curve(##1, "##2", "##3");}}%
  \let\b\relax
  \newcommand{\b}[2]{\mp{ boite(btex \maketangleboxlabel{##2} etex, ##1);}}%
  \let\l\relax
  \newcommand{\l}[3][lft]{\mp{ label.##1(btex \maketanglelabel{##3} etex,
      point ##2 of _tangle_curve[_tangle_n-1]);}}%
  \let\c\relax
  \newcommand{\c}[2]{\mp{ curve( (##1)
      xscaled tangle_unit_x yscaled tangle_unit_y,"##2");}}%
  \let\u\relax
```

```
  \newcommand{\u}[1]{\mp{ tangle_unit_x := ##1; tangle_unit_y := ##1; }}%
  \let\ux\relax
  \newcommand{\ux}[1]{\mp{ tangle_unit_x := ##1; }}%
  \let\uy\relax
  \newcommand{\uy}[1]{\mp{ tangle_unit_y := ##1; }}%
  \let\ShowControlPoints\relax
  \newcommand{\ShowControlPoints}{%
    \mp{tangle_draw_control_points:=true;}}%
  \let\ShowIntersectionPoints\relax
  \newcommand{\ShowIntersectionPoints}{%
    \mp{tangle_draw_intersection_points:=true;}}%
  \let\DoNotComputeIntersections\relax
  \newcommand{\DoNotComputeIntersections}{%
    \mp{tangle_compute_intersections:=false;}}%
  \stepcounter{tangle@n}%
  \IfFileExists{\jobname_tangle_output.\thetangle@n}{%
    \raisebox{-.5\height}{\includegraphics{\jobname_tangle_output.\thetangle@n}}%
    }{}%
  \mp{beginfig(\thetangle@n)}%
  \mp{begintangle}%
  }{%
  \mp{endtangle}%
  \mp{endfig;}%
  \endgroup
  }

\AtEndDocument{%
  \immediate\write\tangle@out{bye;}%
  \immediate\closeout\tangle@out
  \immediate\write18{TEX=latex mpost \jobname_tangle_output}%
  }
```

9 Source Metapost

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% enchevetrements.mp
%%
%% Macros en MetaPost pour tracer des enchevêtrements, comme on en
%% rencontre lors de l'étude des catégories tressées.
%%
%% (c) 2000 Vincent Zoonekynd <zoonek@math.jussieu.fr>
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% BUGS :
%%
%% - Problème lorsque les points d'intersection sont juste au début de
%%   la courbe.
%%
%% - Problème quand les points d'intersection sont trop rapprochés.
%%   Ça marchera néanmoins la plupart du temps...
%%   Il manque des exemples mettant ce problème en évidence.
%%   il faudrait signaler les incohérences, par exemple une courbe qui
%%   passe au dessus puis au dessous en deux points d'intersection trop
%%   rapprochés.
%% - La routine de calcul d'intersection devrait signaler les courbes
%%   _confondues_, donnant lieu à un nombre infini d'intersections.
%% - Dans les exemples, il manque des situations d'erreur (nombre ou
%%   type d'arguments insuffisants, pas de courbe précédente pour la
%%   commande boite(), etc.).
%% - La figure 4 met en évidence un bug dans le calcul des
%%   intersections et dans le tri des points d'intersection.
%%   (je ne l'ai ni trouvé ni corrigé, mais il semble avoir disparu ?)
%% - Je n'ai pas vérifié que la fonction  uniq  fonctionnait
%%   correctement.  La fonction  sort  non plus.
%% - Il manque de la documentation
%% - On devrait pouvoir changer les unités horizontale et verticale
%%   séparément.
%% - Il faudra aussi pouvoir contrôler la largeur des boites contenant
%%   des noms de morphismes, pour pouvoir dessiner des morphismes
%%   entre des produits tensoriels.  la syntaxe pourrait être la
%%   suivante :
%%     boite( btex $f$ etex, 1.5);    % point 1.5 of _tangle_curve[_tangle_n-1]
%%     boite( btex $f$ etex, 1.5, 1); % point 1.5 of _tangle_curve[1]
%%     boite( btex $f \tens g$ etex, 1.5, 1, 1.5, 2);
%%                                    % point 1.5 of _tangle_curve[1] and
%%                                    % point 1.5 of _tangle_curve[2]
%%   Ce sera un peu plus délicat à programmer...
%% - Il n'est pas possible de modifier certaines choses :
%%     · ajouter des commandes à la fin du tracé
```

```
%%     · Rajouter des types d'intersections (par exemple, pour tracer
%%       un cercle.
%%     · Il n'est pas possible de récupérer les résultats des calculs
%%       effectués par mes macros (on peut vouloir jouer avec les
%%       points d'untersection...)
%% - Il manque des instructions de débuggage, du genre
%%   show_intersection_points (qui pourrait afficher à l'écran, dessiner et
%%   numéroter les points d'intersection).
%% - Il ne devrait pas être nécessaire d'exiger que le dernier
%%   caractère de la chaine soit une virgule.
%% - On devrait pouvoir changer le diamètre du cercle utilisé pour
%%   couper les courbes pour _chaque_ point d'intersection.
%% - LaTeX : documentation des macros.  Regarder ce qui se passe si on
%%   met des choses fragiles en argument.
%% - Prévoir d'utiliser psfrag de manière à pouvoir mettre à peu près
%%   n'importe quoi dans les boites.
%%   NON : on n'utilisera pas psfrag, car il ne permet pas d'avoir la
%%   bonne taille pour les boites.
%%   (On pourrait faire les choses de manière très compliquée :
%%   utiliser btex ...  etex pour calculer la taille des boites sans
%%   les tracer en ensuite utiliser psfrag.  C'est très compliqué, n'y
%%   comptez pas.  L'utilisation de psfrag ne sra que très marginale,
%%   je laisse donc à l'utilisateur d'y recourrir, dans les rares cas
%%   où dela est nécessaire.)
%% - Documentation du code
%% - Messages d'erreur :  si on oublie begintangle et endtangle.
%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%
%% En-tête de fichier LaTeX
%%
%% Ces lignes n'ont rien à faire ici, elles doivent figurer dans le
%% fichier principal, à partir duquel vous charger ces macros.
%%
verbatimtex
\documentclass[12pt]{article}
\usepackage[T1]{fontenc}
\usepackage{amsmath}
\begin{document}
etex
%prologues := 2;
%%
%% Macros dont on peut avoir besoin :
%%  - les boites
%%  - la commande TEX, redéfinie pour utiliser LaTeX
%%    (heureusement, elle n'est pas utilisée :  l'utilisateur ne
%%     pourrait pas facilement ajouter des choses dans l'en-tête du
%%     fichier LaTeX...)
%%
```

```
input boxes;
vardef TEX primary s =
  write "verbatimtex"                     to "mptextmp.mp";
  write "\documentclass[12pt]{article}"  to "mptextmp.mp";
  write "\usepackage[T1]{fontenc}"       to "mptextmp.mp";
  write "\usepackage{amsmath,amssymb}"   to "mptextmp.mp";
  write "\begin{document}"               to "mptextmp.mp";
  write "etex"                           to "mptextmp.mp";
  write "btex "&s&" etex"                to "mptextmp.mp";
  write EOF                              to "mptextmp.mp";
  scantokens "input mptextmp"
enddef;
%%
%% Routine de tri
%%
%% (C'est l'algorithme le plus lent que je connaisse, mais c'est le
%% plus facile à programmer :  le tri à bulles.)
%%
%% Premier argument :    nom du tableau
%% Deuxième argument :  nombre d'éléments
%%
vardef sort (suffix t)(expr n) =
  save tmp;
%  show( "SORT begin" );
%  for i=0 upto n-1:  show t[i]; endfor;
  for j=0 upto n-1:
    for i=0 upto n-2:
%      show("comparing "&decimal(i)&" and "&decimal(i+1));
      if t[i]>t[i+1]:
%        show("swapping "&decimal(i)&" and "&decimal(i+1));
        tmp:=t[i+1];
        t[i+1]:=t[i];
        t[i]:=tmp;
      fi;
    endfor;
  endfor;
%  show( "SORT end" );
%  for i=0 upto n-1:  show t[i]; endfor;
enddef;
%%
%% Pour afficher les messages d'erreur
%%
%% Argument :  une liste de choses à afficher, pour l'instant
%%            uniquement des chaines de caractères ou des nombres.
%%
vardef erreur(text t) =
  save s;
  string s;
  s:="";
```

```
  for i=t:
    if string i:  s:=s&i;
    elseif numeric i:  s:=s&decimal(i);
    else:  show("problème dans les arguments de la routine ‘erreur’");
    fi;
  endfor;
  show(s);
enddef;
%%
%% uniq
%%
%% Voici une commande uniq qui enlève les doublons dans les
%% tableaux.  Cela peut se produire quand un point d’intersection est
%% exactement à la jonction de deux petits segments de courbes.
%% Cela risque de se produire quand le point d’intersection est
%% exactement sur l’un des points de contrôle.
%%
%% Premier argument :    nom du tableau
%% Deuxième argument :  nombre d’éléments
%%
%% Ces deux arguments seront modifiés.
%%
vardef uniq (suffix t)(suffix n) =
%      show("initial table:");
%      for j=0 upto n-1:
%        show( "  " & decimal(j) &"  =>  "& decimal(t[j]) );
%      endfor;
  save i; numeric i; i:=0;
  forever:  exitunless (i<n-1);
%    show("comparing " &
%   decimal(i) &
%   " (" &
%   decimal(t[i]) &
%   ") and " &
%   decimal(i+1) &
%   " (" &
%   decimal(t[i+1]) &
%   ")"
% );
    if abs( t[i] - t[i+1] )<0.001:
%      show("removing "&decimal(i+1));
      for j=i+1 upto n-2:
        t[j]:=t[j+1];
      endfor;
      n:=n-1;
%      show("removed "&decimal(i+1));
%      show("new table :");
%      for j=0 upto n-1:
%        show( "  " & decimal(j) &"  =>  "& decimal(t[j]) );
```

```
%      endfor;
    fi;
    i:=i+1;
  endfor;
enddef;
%%
%% Variables globales
%%
%% tangle_N               : nombre de morceaux lors de la découpe des
%%                          courbes (on veut un unique point
%%                          d'intersection ou d'auto-intersection par
%%                          morceau)
%% tangle_unit_x            unité horizontale du tracé
%% tangle_unit_y            unité verticale du tracé
%% tangle_circle_diameter   distance à laquelle un brin disparait lors
%%                          qu'il passe en dessous d'un autre.
%%                          Il faudrait pouvoir modifier cela pour
%%                          chaque point d'intersection.  Mieux :
%%                          l'ordinateur pourrait choisir de le faire
%%                          lui-même si les deux brins se coupant ont
%%                          des tangentes trop proches.
%%
numeric tangle_N, tangle_default_unit_x, tangle_default_unit_y, tangle_circle_diameter;
tangle_N := 50;
tangle_default_unit_x := 1cm;
tangle_default_unit_y := 1cm;
tangle_circle_diameter := 3mm;
boolean tangle_default_draw_intersection_points, tangle_default_draw_control_points;
boolean tangle_default_compute_intersections;
tangle_default_draw_intersection_points := false;
tangle_default_draw_control_points      := false;
tangle_default_compute_intersections    := true;
%%
%% Variables locales, initialisations
%%
def begintangle =
  begingroup;
  numeric _tangle_n;          % Numéro de la prochaine courbe = nombre
  _tangle_n := 0;             % de courbes déjà définies
  path _tangle_curve[];       % Les courbes
  string _tangle_position[]; % Chaines indiquant si la courbe doit
                              % passer au dessus ou au dessous.
  numeric _tangle_intersection_n[]; % Nombre de points d'inetersection
  _tangle_intersection_n[_tangle_n] := 0; % sur l'une des courbes
  numeric _tangle_intersection[][]; % Valeur du paramètre en ces
                                     % points d'intersection
  numeric _tangle_box_n;      % Nombre de boites
  _tangle_box_n:=0;
  numeric tangle_box[];       % Contenu des boites.
```

```
  boolean tangle_draw_intersection_points, tangle_draw_control_points,
       tangle_compute_intersections ;
  tangle_draw_intersection_points := tangle_default_draw_intersection_points;
  tangle_draw_control_points := tangle_default_draw_control_points;
  tangle_compute_intersections := tangle_default_compute_intersections;
  numeric tangle_unit_x, tangle_unit_y;
  tangle_unit_x = tangle_default_unit_x;
  tangle_unit_y = tangle_default_unit_y;
enddef;
%%
%% Les diverses fonctions appelées une fois toutes les courbes
%% définies
%%
def endtangle_compute_intersections =
  for i=0 upto _tangle_n-1:
    for j=0 upto _tangle_n-1:
    if i<>j :
      % Calcul des points d'intersection des courbes i et j
      begingroup;
        save a,b,N,k,l,p,q,r;
        numeric N,k,l;
        path p,q,r,s;
        N:=tangle_N;
        p = _tangle_curve[i];
        q = _tangle_curve[j];
        for k=0 step length(p)/N until length(p):
          for l=0 step length(q)/N until length(q):
            numeric a,b;
            pair A;
            r := (point k of p){direction k of p} ..
                 (point (k+length(p)/N) of p){direction (k+length(p)/N) of p};
            s := (point l of q){direction l of q} ..
                 (point (l+length(q)/N) of q){direction (l+length(q)/N) of q};
            (a,b) = r intersectiontimes s;
            if a <> -1:
              A := r intersectionpoint s;
% begingroup
%   numeric a, b, c, m[];
%   a := k;
%   b := k + length(p)/N;
%   forever:
%     exitunless abs((point a of p) - (point b of p)) > .001mm;
%     m[1] := abs( A - point 1/4[a,b] of p);
%     m[2] := abs( A - point 1/2[a,b] of p);
%     m[3] := abs( A - point 3/4[a,b] of p);
%     if (m[1] < m[2]) and (m[1] < m[3]):
%       b:=1/2[a,b];
%     elseif (m[3] < m[1]) and (m[3] < m[2]):
%       a:=1/2[a,b];
```

```
%       else:
%         c := a;
%         a := 1/4[a,b];
%         b := 3/4[c,b];
%         %(a,b) := ( 1/4[a,b], 3/4[a,b] );
%       fi;
%     endfor;
%            draw fullcircle scaled 4bp shifted point a of p;
%            draw fullcircle scaled 4bp shifted point b of p;
%     show "i:  " &
%          decimal( abs( (point a of p) - (point b of p) )/mm ) &
%          "mm (" &
%          decimal(a) &
%          ", " &
%          decimal(b) &
%          ")";
                _tangle_intersection[i][_tangle_intersection_n[i]] :=
                k + a*length(p)/N;
%     _tangle_intersection[i][_tangle_intersection_n[i]] := a;
          _tangle_intersection_n[i] := _tangle_intersection_n[i] + 1;
% endgroup;
            fi;
          endfor;
        endfor;
      endgroup;
      % Fin du calcul des intersections et i et j
    else:
      % Calcul de l'autointersection de i=j
      begingroup;
        save a,b,N,k,l,p,q,r;
        numeric N,k,l;
        path p,q,r,s;
        N:=tangle_N;
        p = _tangle_curve[i];
        q = _tangle_curve[j];
        for k=0 step length(p)/N until length(p):
          for l=k+2*length(p)/N
              step length(p)/N
              until ( if k=0:  (N-1)*length(p)/N else:  length(p) fi ):
            numeric a,b;
            pair A;
            r := subpath(k,k+length(p)/N) of p;
            s := subpath(l,l+length(p)/N) of p;
            r := (point k of p){direction k of p} ..
                 (point (k+length(p)/N) of p){direction (k+length(p)/N) of p};
            s := (point l of p){direction l of p} ..
                 (point (l+length(p)/N) of p){direction (l+length(p)/N) of p};
            (a,b) = r intersectiontimes s;
            if a <> -1 :
```

```
            % Comme précédemment, on essaye de se ramener d'une
            % coordonnée sur le morceau de courbe à une coordonnée
            % sur la courbe.
            if length(r)=1 :
              _tangle_intersection[i][_tangle_intersection_n[i]] :=
              k + a*length(p)/N;
            else:
              show("SERIOUS PROBLEM (autointersection 1)"); %%%%%%%%%%%%%%
              show(i); show(j); show(a); show( length(r) ); show(r);
            fi;
            _tangle_intersection_n[i] := _tangle_intersection_n[i] + 1;
            % Ne pas oublier l'autre point d'auto-intersection...
            if length(s)=1 :
              _tangle_intersection[i][_tangle_intersection_n[i]] :=
              l + b*length(p)/N;
            else:
              show("SERIOUS PROBLEM (autointersection 2)"); %%%%%%%%%%%%%%
              show(i); show(j); show(a); show( length(s) ); show(s);
            fi;
            _tangle_intersection_n[i] := _tangle_intersection_n[i] + 1;
          fi;
        endfor;
      endfor;
    endgroup;
    % Fin du calcul de l'autointersection de i=j
  fi;
  endfor;
 endfor;
 % On met de l'ordre dans ces tableaux
 for i=0 upto _tangle_n-1:
   sort( _tangle_intersection[i] )( _tangle_intersection_n[i] );
   % debug
   %if( i=2 ):
   %  show("DEBUG");
   %  for k=0 upto _tangle_intersection_n[i]-1:
   %    show( _tangle_intersection[i][k] );
   %  endfor;
   %fi;
   % end debug
   uniq( _tangle_intersection[i] )( _tangle_intersection_n[i] );
 endfor;
enddef;
def _tangle_crop_circle(expr h,p) =
  (fullcircle scaled tangle_circle_diameter shifted( point h of p ))
enddef;
def _tangle_problem_intersection_points_too_near(expr b,p) =
  ( (_tangle_crop_circle(a,p) intersectiontimes p) = (-1,-1) )
enddef;
vardef endtangle_draw_curves =
```

```
save p,a,b;
path p;
numeric a,b;
for i=0 upto _tangle_n-1:
  % Vérification de la longueur de la chaine de caractères décrivant
  % la position de la courbe lors des intersections.
  if( length(_tangle_position[i]) <> _tangle_intersection_n[i] ):
    erreur("curve no ",i," has ",_tangle_intersection_n[i],
           " intersection points whereas the provided string gives ",
           length(_tangle_position[i])," of them");
    show("Superfluous information will not be used, missing information will be assumed to be
    for j=0 upto _tangle_intersection_n[i]-1:
      show(_tangle_intersection[i][j] );
    endfor;
  fi;
  % Débug :  numérotation des points d'intersection
  %if( i=2 ):
  %for j=0 upto _tangle_intersection_n[i]-1:
  %  label.bot( decimal(j),
  %         point _tangle_intersection[i][j] of _tangle_curve[i] );
  %endfor;
  %fi;
  % Dessinons...
  if known _tangle_intersection[i][0] :
    p:= _tangle_curve[i];
    a:= 0;
    b:= _tangle_intersection[i][0];
    p:= subpath(a,b) of p;
    a:= 0; b:= length(p);
    % à revoir
    if _tangle_problem_intersection_points_too_near(b,p)
       and
       ( (substring(0,1) of _tangle_position[i]) = "-" ):
      erreur("Warning:  intersection point too near from the the origin");
    else:
      draw p
           if (substring(0,1) of _tangle_position[i]) = "-":
             cutafter (_tangle_crop_circle(b,p))
           fi;
%          if( i=2 ):
%            draw _tangle_crop_circle(b,p) ;
%          fi;
    fi;
    for j=1 upto _tangle_intersection_n[i]-1:
      p:= _tangle_curve[i];
      a:= _tangle_intersection[i][j-1];
      b:= _tangle_intersection[i][j];
      p:= subpath(a,b) of p;
      a:= 0; b:= length(p);
```

```
          % Il se peut que le morceau de courbe soit trop petit et ne
          % coupe pas le cercle destiné à le réduire encore plus.
          % Dans ce cas, il ne faut pas le tracer du tout (voir figure 21).
          % Il faudrait un message d'erreur quand les points
          % d'intersection sont trop rapprochés et que la position (+ ou
          % -) n'est pas la même pour les deux.
          % à revoir
          if ( _tangle_problem_intersection_points_too_near(a,p)
               or
               _tangle_problem_intersection_points_too_near(b,p)
             ) and (
               ((substring(j,j+1) of _tangle_position[i]) = "-")
               or
               ((substring(j-1,j) of _tangle_position[i]) = "-")
             ):
          erreur("Warning:  intersection point near each other, not drawing curve segment");
          else:
            draw p
                 if (substring(j,j+1) of _tangle_position[i]) = "-":
                   cutafter  (_tangle_crop_circle(b,p))
                 fi
                 if (substring(j-1,j) of _tangle_position[i]) = "-":
                   cutbefore (_tangle_crop_circle(a,p))
                 fi;
%            if( (i=2) and (j=2) ):
%       draw _tangle_crop_circle(b,p);
%       draw _tangle_crop_circle(a,p);
%            fi;
          fi;
        endfor;
        p:= _tangle_curve[i];
        a:= _tangle_intersection[i][_tangle_intersection_n[i]-1];
        b:= length(p);
        p:= subpath(a,b) of p;
        a:= 0; b:= length(p);
        % à revoir
        if _tangle_problem_intersection_points_too_near(a,p)
          and
          ((substring(_tangle_intersection_n[i]-1,_tangle_intersection_n[i])
            of _tangle_position[i]) = "-"):
          erreur("Warning:  intersection point too near the end, skipping  curve segment.");
        else:
          draw p
               if (substring(_tangle_intersection_n[i]-1, _tangle_intersection_n[i])
                   of _tangle_position[i]) = "-":
                 cutbefore (_tangle_crop_circle(a,p))
               fi;
        fi;
      else:
```

```
      draw _tangle_curve[i];
    fi;
  endfor;
enddef;
def endtangle_draw_boxes =
  for i=0 upto _tangle_box_n-1:
    fill bpath _tangle_box[i] withcolor white;
    drawboxed(_tangle_box[i]);
  endfor;
enddef;
def endtangle_draw_control_points =
  for i=0 upto _tangle_n-1:
    for j=0 upto length(_tangle_curve[i]):
      draw point j of _tangle_curve[i] withpen pencircle scaled 4bp;
    endfor;
  endfor;
  for i=0 upto _tangle_box_n - 1:
    draw (_tangle_box[i] .c) withpen pencircle scaled 4bp withcolor .5[green,blue];
  endfor;
enddef;
def endtangle_draw_intersection_points =
  for i=0 upto _tangle_n-1:
    for j=0 upto _tangle_intersection_n[i]-1 :
%      draw subpath(
% _tangle_intersection[i][j]-.05,
% _tangle_intersection[i][j]+.05)
%      of _tangle_curve[i]
%      withpen pencircle scaled 1bp withcolor red;
      draw (point _tangle_intersection[i][j] of _tangle_curve[i])
      withpen pencircle scaled 4bp withcolor red;
    endfor;
  endfor;
enddef;
def endtangle =
  if(tangle_compute_intersections):
    endtangle_compute_intersections;
    if(tangle_draw_intersection_points):
      endtangle_draw_intersection_points;
    fi;
  fi;
  if(tangle_draw_control_points):
    endtangle_draw_control_points;
  fi;
  endtangle_draw_curves;
  endtangle_draw_boxes;
  endgroup;
enddef;
vardef curve(text t) =
  save c,x,y,s,ss,i,p;
```

```
numeric c,x,y,i;
string s,ss;
pair p[];
c := 0;
i:=0;
for a=t:
  if numeric a:
    if i=0:  x:=a;
    elseif i=1:  y:=a;
    else:  erreur("curve:  Wrong type")
    fi;
  elseif string a:
    if i=0:  i:=2; x:=0; y:=0; s:=a;
    elseif (i=1) and (known _tangle_curve[_tangle_n]):  ss:=a;
    elseif i=1:  erreur("curve:  Wrong type (expecting numeric)")
    elseif i=2:  s:=a;
    elseif i=3:  ss:=a;
    else:  erreur("curve:  Too many arguments")
    fi;
  elseif path a:
    _tangle_curve[_tangle_n] := a;
  else:  erreur("curve:  Wrong type")
  fi;
  i:=i+1;
endfor;
if unknown ss:  ss="" fi;
if (unknown _tangle_curve[_tangle_n]) and (unknown s):
  erreur("curve:  Too few arguments");
elseif unknown _tangle_curve[_tangle_n]:
  % On s'assure que la chaine de caractères es termine par une
  % virgule, i.e., que le dernier point de contrôle est bien utilisé.
  if (substring(length(s)-1,length(s)) of s) <> ",":
    s:=s&",";
  fi;
  p[c] := (x,y) xscaled tangle_unit_x yscaled tangle_unit_y;
  for i=0 upto length(s):
    if( substring(i,i+1) of s = "," ):
      c:=c+1; p[c]:=(x,y) xscaled tangle_unit_x yscaled tangle_unit_y;
    elseif( substring(i,i+1) of s = "r" ):          x:=x+1
    elseif( substring(i,i+1) of s = "l" ):          x:=x-1;
    elseif( substring(i,i+1) of s = "u" ):          y:=y+1
    elseif( substring(i,i+1) of s = "d" ):          y:=y-1
    fi;
  endfor;
  _tangle_curve[_tangle_n] :=
    for i=0 upto c-1:  p[i] ..  endfor p[c];
fi;
% On stocke la chaine de caractères contenant la liste de positions
% (au dessus ou en dessous)
```

```
  _tangle_position[_tangle_n] := ss;
  % On incrémente
  _tangle_n := _tangle_n + 1;
  _tangle_intersection_n[_tangle_n] := 0;
enddef;
vardef boite_vieux (expr t,x) =
  save i; numeric i; i:=_tangle_box_n;
  boxit _tangle_box[i] (t);
  _tangle_box[i] .c = point x of _tangle_curve[_tangle_n-1];
  _tangle_box_n := _tangle_box_n + 1;
enddef;
def even(expr a) = (not odd(a)) enddef;
% À l'avenir, il faudrait aussi pouvoir préciser un point quelconque,
% en dehors des courbes.
vardef boite (expr t)(text a) =
  save i, n, c, pos, A, s, G, w, h, dx, dy;
  numeric i, n, c[], pos[], w, h, dx, dy;
  pair A[], G;
  i:=0;
  for s=a:
    if even(i):  n:=i/2; pos[n]:=s; c[n]:=_tangle_n-1;
    else:  c[n]:=s; % Vérifier que la courbe existe...
    fi;
    i:=i+1;
  endfor;
  % Il faudrait vérifier qu'il y avait suffisemment de paramètres,
  % ie, que n est bien défini.
  G := (0,0);
  for i=0 upto n:
    A[i] := point pos[i] of _tangle_curve[c[i]];
    draw A[i] withpen pencircle scaled 4bp withcolor red;
    G := G + (1/(n+1))*A[i];
  endfor;
  i:=_tangle_box_n;
  boxit _tangle_box[i] (t);
  _tangle_box[i] .c = G;
  % Ici, je suppose que t est de type picture (par exemple, btex...etex),
  % mais ce pourrait être une chaîne.
  w := .5xpart( urcorner t - ulcorner t );
  h := .5xpart( urcorner t - lrcorner t );
  dx := defaultdx;
  dy := defaultdy;
  for j=0 upto n:
    dx := max( dx, xpart(A[j]-G) );
    dy := max( dy, ypart(A[j]-G) );
  endfor;
  _tangle_box[i] .dx = dx;
  _tangle_box[i] .dy = dy;
  _tangle_box_n := _tangle_box_n + 1;
```

```
enddef;
```