LuaLaTeX et MetaPost au service de la vulgarisation

Récemment, j'ai participé à l'accueil d'élèves de classe de troisième dans mon laboratoire de recherche en mathématiques. J'ai souhaité les introduire à une partie de mon métier : le calcul scientifique[78]. Nous avons donc, avec un collègue, décidé de construire un exposé autour du calcul numérique de π. Vous pouvez d'ailleurs consulter le support de la présentation à l'adresse :

https://www.ceremade.dauphine.fr/~chupin/exposes/2024/methodesNum.pdf

Ici, je voudrais illustrer comment LuaLaTeX et MetaPost m'ont permis de produire les supports graphiques de nos explications.

LuaTeX et MetaPost avec minim-mp

Comme on peut le constater dans mes divers articles pour la *Lettre*, je suis un fervent utilisateur de MetaPost. Son intégration au moteur LuaTeX me facilite encore son utilisation. J'ai parlé de nombreuses fois du package luamplib qui fournit l'environnement `mplibcode` permettant d'exécuter du code MetaPost dans le source LaTeX.

Dans cet article, je vais monter que l'on peut utiliser un autre package : minim-mp. Celui-ci fait partie du format minim *plain* pour le moteur LuaTeX, format qui implémente le support de nombreuses fonctionnalités de LuaTeX et du format PDF. La plupart d'entre elles sont fournies au travers de packages séparés qui peuvent être utilisés séparément :

minim-mp : pour le support de la bibliothèque Lua `mplib` ;

minim-math : pour la gestion des mathématiques Unicode ;

minim-pdf : pour la gestion des liens hypertextes et les ancrages PDF ;

minim-xmp : pour la gestion des métadonnées PDF.

Les noms de ces packages sont explicites : il s'agit d'être minimal tout en offrant beaucoup d'outils.

C'est au premier d'entre eux, minim-mp, que nous nous intéressons ici. Il permet donc l'intégration de la bibliothèque Lua `mplib` pour le format plain TeX. Ce package TeX fournit la commande de base `\directmetapost` qui permet d'exécuter du code MetaPost directement dans le code LaTeX (un peu comme `\directlua` permet d'exécuter du code Lua). Chaque invocation de la commande `\directmetapost` lance une instance MetaPost indépendante. Ainsi, nous obtenons un dessin MetaPost avec le code suivant :

78. Le calcul scientifique est une discipline aux contours flous, qui regroupe un ensemble de champs mathématiques et informatiques permettant la simulation numérique des phénomènes modélisés par des équations mathématiques.

Exemple 18

code

```
Traçons un cercle : \directmetapost{
beginfig(1);draw fullcircle scaled 1cm;endfig;}.
```

résultat

Traçons un cercle :  .

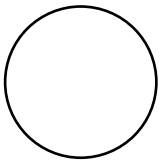
Cette commande produit, pour chaque groupe MetaPost délimité par `beginfig` et `endfig`, des boîtes que nous pouvons manipuler.

Exemple 19

code

```
\scalebox{2}{\directmetapost{beginfig(2);draw
   fullcircle scaled 1cm;endfig;}}
```

résultat



Mais minim-mp n'est pas qu'un package TEX : c'est aussi un package LATEX qui, bien qu'expérimental pour l'instant, est tout à fait utilisable[79]. Celui-ci nous fournit l'environnement `metapost` pour exécuter du code MetaPost.

Exemple 20

```
\begin{metapost}
beginfig(1);
draw (0,0)--(1cm,1cm);
endfig;
\end{metapost}
```

Le package LATEX peut même être chargé avec l'option `luamplib` et fournit alors des commandes et environnements analogues à ceux du package luamplib.

L'environnement `metapost` crée des instances MetaPost indépendantes, mais l'utilisateur peut aisément définir de nouveaux environnements qui, eux, sont persistants. Dans le code suivant, nous créons un environnement `mppersist` qui permet de garder en mémoire les exécutions précédentes des environnements `mppersist`.

79. La rédaction de cet article a d'ailleurs permis de faire remonter quelques dysfonctionnements, très vite corrigés par l'auteur : https://gitlab.com/renkema/minim/-/issues.

Exemple 21

code

```
\newmetapostenvironment{mppersist}
\begin{mppersist}
a:=1cm;
\end{mppersist}

\begin{mppersist}
beginfig(1);
draw (0,0)--(a,a);
endfig;
\end{mppersist}
```

résultat

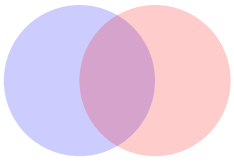
Bien que la documentation de minim-mp manque de détails et d'exemples, ce package présente quelques avantages par rapport à luamplib. Par exemple, la commande `\mpcolor`, qui permet d'utiliser une couleur LaTeX dans le code MetaPost, ne peut, avec luamplib, être utilisée qu'avec l'opérateur `withcolor`. Tandis que minim-mp permet de l'utiliser exactement comme une couleur MetaPost normale. De plus, minim-mp permet d'utiliser des fonctionnalités du format PDF directement depuis MetaPost. Par exemple, alors que MetaPost, utilisé comme programme indépendant de LuaTeX, n'a nativement pas de fonction de transparence[80], minim-mp fournit plusieurs commandes de gestion de la transparence dont l'opérateur `withalpha()`.

Exemple 22

code

```
\begin{metapost}
beginfig(4);
fill fullcircle scaled 2cm withcolor blue withalpha (0.
   2);
fill fullcircle scaled 2cm shifted (1cm,0) withcolor
   red withalpha(0.2);
endfig;
\end{metapost}
```

résultat



Enfin, minim-mp donne accès à de nombreux paramétrages et personnalisations pour qui accepte de se plonger dans ses arcanes.

80. Il faut recourir à des constructions comme l'a proposé Anthony Phan : http://www-math.univ-poitiers.fr/~phan/metalpha.html.

Avec le package animate

Les méthodes numériques de calcul reposent bien souvent sur le calcul approché d'un objet mathématique, soit par discrétisation d'un élément continu, soit par troncature d'une somme infinie. Ainsi, on aura souvent un paramètre fixé, que l'on peut noter N, tel que quand N tend vers l'infini, alors notre méthode numérique (dépendante de ce paramètre N) doit converger vers l'objet mathématique que l'on souhaite calculer. Évidemment, ces concepts ont des définitions mathématiques rigoureuses que nous n'aborderons pas ici.

Une première approche… et $\pi = 4$!

La première méthode numérique que nous avons souhaité présenter aux élèves est une mauvaise méthode [81] qui semble fonctionner et qui semble prouver que $\pi = 4$. On va considérer un quart de cercle, et par symétries, on réalisera sur tout le cercle les opérations décrites. Ainsi, pour N fixé, on divise le rayon (suivant l'axe des abscisses) en N parties égales, puis on projette verticalement (vers le haut) les points obtenus sur le cercle. Une fois ces points obtenus, on les relie, extérieurement au cercle, par des lignes brisées constituées de segments uniquement horizontaux ou verticaux. Le dessin de la figure 20 éclaire cette explication.

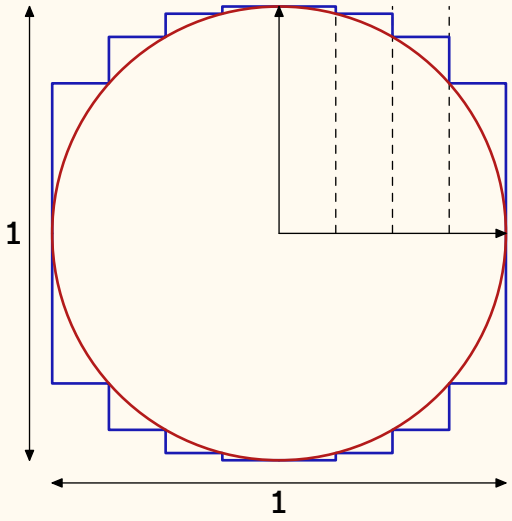


Figure 20 – Illustration de la mauvaise méthode numérique qui semble montrer que $\pi = 4$. Ici $N = 4$.

Utiliser un langage de programmation pour produire ce type d'image permet de paramétrer le dessin, et donc d'illustrer aisément l'évolution d'une méthode numérique.

MetaPost nous permet de programmer plus aisément que TikZ car il est plus proche d'un langage de programmation classique. Voici un code MetaPost, que nous ne détaillerons pas, produisant l'image décrite précédemment :

81. L'intérêt est double : illustrer sur un exemple simple le fait d'avoir une méthode qui semble se rapprocher de ce qu'on souhaite calculer quand le paramètre N tend vers l'infini et illustrer que les choses qui semblent évidentes en mathématiques doivent être considérées rigoureusement.

Exemple 23

```
N:=4; % le paramètre
color DarkBlue,DarkRed,DarkGreen; % trois couleurs
DarkBlue := (0.1,0.1,0.7);
DarkRed := (0.7,0.1,0.1);
DarkGreen := (0.1,0.7,0.1);
beginfig(1);
u:=3cm; % l'unité
pair Point, pointS;
path poly,droite,cercle;
cercle := fullcircle scaled 2u; % on définit le cercle
% on légende les dimensions
drawdblarrow (-1.1u,-u)--(-1.1u,u);
drawdblarrow (-u,-1.1u)--(u,-1.1u);
label.lft(btex 1 etex, (-1.1u,0));
label.bot(btex 1 etex, (0,-1.1u));
if(N=1): % si N=1 alors le polygone est le carré
    draw unitsquare scaled 2u shifted (-u,-u)
    withcolor DarkBlue withpen pencircle scaled 1pt;
else: % sinon
    poly:=(0,u);
    pointS := (0,u);
    for j:=1 upto (N-1): % pour tous les points du
   rayon subdivisé
        % on définit le segment vertical passant par
   le point
        xp := j*u/(N);
        droite := (xp,0)--(xp,u);
        if(N<=10): % on ne trace les pointillés de ces
    segments que pour N<=10
            draw droite dashed evenly;
        fi
        % on définit le point sur le cercle comme
        % l'intersection entre le cercle et le segment
        Point := droite intersectionpoint cercle;
        % rajout du segment vertical au polygone
        poly := poly -- (xp,ypart pointS)--Point;
        pointS := Point; % sauvegarde du point courant
    endfor
    % dernier point
    poly := poly -- (u,ypart pointS)--(u,0);
    % tracé du polygone
    draw poly withcolor DarkBlue withpen pencircle
   scaled
    1pt;
    % et des trois réflexions
    draw poly reflectedabout((0,0),(u,0)) withcolor
   DarkBlue withpen pencircle scaled
    1pt;
```

```
    draw poly reflectedabout((0,0),(0,u)) withcolor DarkBlue withpen pencircle scaled 1pt;
    draw poly rotated 180 withcolor DarkBlue withpen pencircle scaled 1pt;
fi
% on dessine le cercle
draw cercle withcolor DarkRed withpen pencircle scaled 1pt;
% on trace les axes
drawarrow (0,0)--(u,0);
drawarrow (0,0)--(0,u);
endfig;
```

Il est possible d'encapsuler ce code dans une macro LaTeX pour l'appeler simplement avec le package animate et produire ainsi une animation [82].

Exemple 24

```
\newcommand\piquatre[1]{%
\begin{metapost}
N:=#1; % le paramètre
color DarkBlue,DarkRed,DarkGreen; % trois couleurs
DarkBlue := (0.1,0.1,0.7);
DarkRed := (0.7,0.1,0.1);
% ... code MetaPost précédent
endfig;
\end{metapost}
}
```

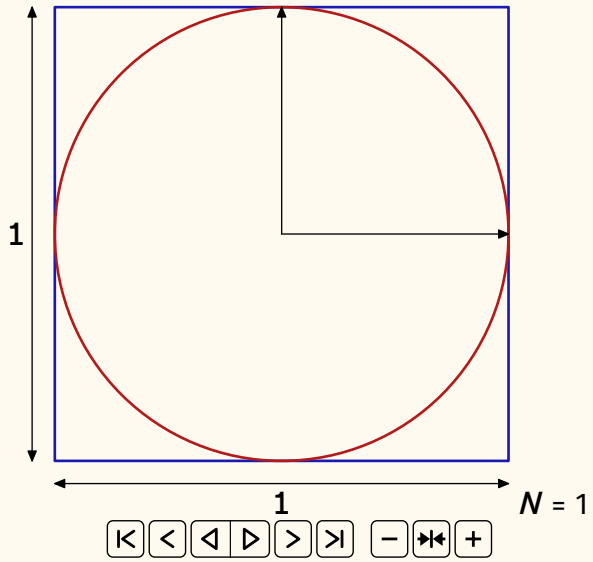
Une fois cette macro définie, il suffit de l'insérer dans l'environnement `animateinline` :

Exemple 25

```
\begin{animateinline}[controls]{1}%
    \multiframe{20}{N=1+1}{%
     \piquatre{\N}%
     $N=\N$
    }%
\end{animateinline}%
```

… pour obtenir le résultat suivant :

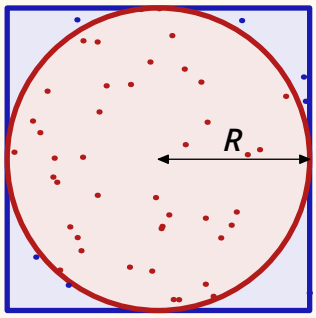
82. Visionnable par exemple avec `Okular` ou `Acrobat Reader`.



La production de ces images est un vrai plaisir mais cette méthode numérique semble montrer que $\pi = 4$! Pourriez-vous expliquer pourquoi cette méthode ne montre pas ce qu'elle semble montrer ?

Un peu d'aléatoire

La dernière méthode numérique[83] que nous avons présentée aux élèves de troisième est celle de Monte Carlo, simple à comprendre : on considère un cercle et son carré circonscrit, et on tire N points aléatoirement dans le carré.



On compte alors les points qui se trouvent à l'intérieur du cercle et on note ce nombre n. On calcule ensuite :

$$\frac{n}{N} \times 4,$$

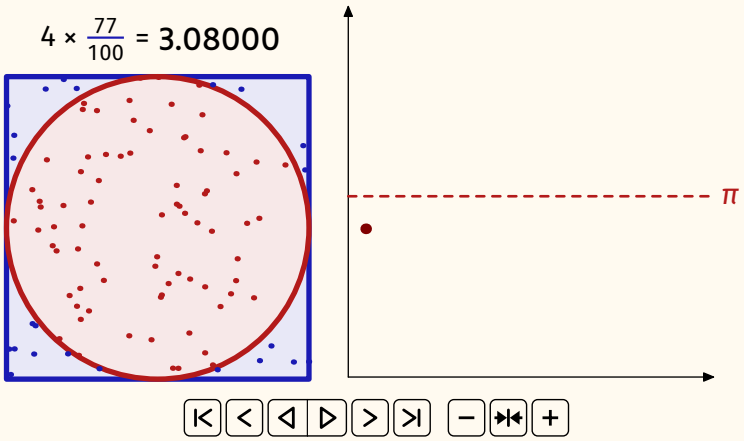
qui donne une approximation de π.

En effet, en mathématiques, un résultat remarquable, la *loi des grands nombres*, garantit que :

$$\text{Probabilité d'être dans le cercle} = \frac{\text{Aire cercle}}{\text{Aire carré}} = \frac{\pi R^2}{(2R)^2} = \frac{\pi}{4}$$

Grâce à LuaLaTeX, MetaPost et animate, nous pouvons illustrer ce phénomène :

83. Avant cela, nous avons présenté la méthode d'Archimède de construction de polygones réguliers inscrits et circonscrits au cercle, mais les codes LaTeX et MetaPost restent très proches de ce qui a déjà été présenté ici.



Pour ce faire, nous définissons un environnement persistant pour que le code MetaPost exécuté dans cet environnement soit visible par les autres environnements.

Exemple 26

```
\newmetapostenvironment[mathmode='double']{
   mpmontecarlo}
```

Il est nécessaire d'ajouter `mathmode='double'` en option de la commande pour utiliser la représentation numérique en double précision et ainsi pouvoir utiliser des nombres entiers très grands (ici, on va en générer 6000).

Ensuite, nous générons de nombreux points dans le carré et définissons la fonction MetaPost qui calcule le rapport entre le nombre de points à l'intérieur du cercle et le nombre de points total. Cela se fait avec le code suivant à inclure dans un environnement `mpmontecarlo` :

Exemple 27

```
numeric monteC[];
pair Points[];

for i:=1 upto 6000:
    Points[i] := (uniformdeviate(4cm)-2cm,
   uniformdeviate(4cm)-2cm);
endfor

vardef proba(expr N)=
    save MC;
    MC := 0;
    for i:=1 upto N:
        if(abs(Points[i])<2cm):
            MC := MC + 1/N;
        fi
    endfor
    (4*MC)
enddef;
```

La figure représentant le nuage de points dans le carré circonscrit au cercle

est obtenue à l'aide du code MetaPost suivant :

Exemple 28

```
color DarkBlue,DarkRed,DarkGreen; % trois couleurs
DarkBlue := (0.1,0.1,0.7);
DarkRed := (0.7,0.1,0.1);
DarkGreen := (0.1,0.7,0.1);
N:=#1; % le paramètre qui proviendra de l'argument d'une commande LaTeX
beginfig(1+#1);
    % le carré
    fill unitsquare scaled 4cm shifted (-2cm,-2cm)
    withcolor 0.9[DarkBlue,white] ;
    draw unitsquare scaled 4cm shifted (-2cm,-2cm)
    withcolor DarkBlue withpen pencircle scaled 2pt;
    % le cercle
    fill fullcircle scaled 4cm withcolor 0.9[DarkRed,white];
    draw fullcircle scaled 4cm withcolor DarkRed
    withpen pencircle scaled 2pt;
    % on va compter les points à l'intérieur
    % et les tracer suivant s'ils sont dedans ou non
    Ni:=0;
    for i:=1 upto (N):
        if(abs(Points[i])<2cm):
            drawdot Points[i] withpen pencircle scaled 2pt withcolor
            DarkRed;
            Ni:=Ni+1;
        else:
            drawdot Points[i] withpen pencircle scaled 2pt withcolor DarkBlue;
        fi
    endfor
    % le label
    string labi,v;
    labi := "$4\times \frac{"&decimal Ni &"}{"&decimal N &"}=$";
    v:=substring (0,7) of decimal proba(N);
    label.lft(textext(labi),(0,2.5cm));
    label.rt(textext(v),(-0.1cm,2.5cm));
endfig;
```

Pour utiliser ce code avec animate, nous l'encapsulons dans une commande LaTeX :

Exemple 29

```
\newcommand\montecarlo[1]{
    \begin{mpmontecarlo}
```

```
    color DarkBlue,DarkRed,DarkGreen; % trois couleurs
    DarkBlue := (0.1,0.1,0.7);
    DarkRed := (0.7,0.1,0.1);
    DarkGreen := (0.1,0.7,0.1);
    N:=#1;
    % code MetaPost précédent
    %...
    \end{mpmontecarlo}
}
```

Avec le code MetaPost suivant, nous procédons de la même manière pour tracer la courbe d'évolution du calcul suivant la valeur de *N* :

Exemple 30

```
beginfig(#1);
    % unités
    uX := 0.23cm;
    uY := 7cm;
    % bornes
    ymin:=2.8;
    ymax:=3.5;
    xmax:=20;
    % axes
    drawarrow (-1uX,ymin*uY)--(xmax*uX,ymin*uY);
    drawarrow (-1uX,ymin*uY)--(-1uX,ymax*uY);
    % la droite y=\pi
    draw ((-1uX,3.1415uY)--(xmax*uX,3.1415uY)) dashed evenly
    withcolor DarkRed withpen pencircle scaled 1pt;
    label.rt(btex $\pi$ etex,(xmax*uX,3.1415uY)) withcolor DarkRed;
    % on fabrique la ligne brisée du graphique
    path plot;
    i:=0;
    for j:=100 step 300 until #1:
        if(j=100):
            plot:=(i*uX,proba(j)*uY);
        else:
            plot:=plot--(i*uX,proba(j)*uY);
        fi
            drawdot (i*uX,proba(j)*uY) withpen pencircle
            scaled 4pt withcolor 0.5[red,black];
        i:=i+1;
    endfor
    % on trace la ligne
    draw plot withcolor 0.5[red,black];
endfig;
```

Et comme précédemment, nous l'encapsulons dans une commande LaTeX :

Exemple 31

```
\newcommand\suite[1]{
    \begin{mpmontecarlo}
        beginfig(#1);
        uX := 0.23cm;
        uY := 7cm;
        % code MetaPost précédent
        % ...
        endfig;
    \end{mpmontecarlo}
}
```

Avec ces commandes LaTeX, nous pouvons donc utiliser le package animate pour produire l'animation présentée ci-dessus.

Exemple 32

```
\begin{center}
  \begin{animateinline}[controls]{1}%
    \multiframe{15}{N=100+300}{%
      \begin{tabular}{cc}
        \montecarlo{\N}&\suite{\N}
      \end{tabular}
    }%
  \end{animateinline}%
\end{center}
```

Conclusion

LaTeX et MetaPost me permettent de produire des choses comme je les imagine et cela très simplement. Pour la plupart de mes productions, les aspects de programmation et de géométrie de ces logiciels me rendent de grands services. Dans les deux exemples que j'ai montrés ici, ce fut en direction de collégiens et de collégiennes, et s'ils n'ont pas fait de retour explicite sur l'aspect des diapositives, ils ont, semble-t-il, apprécié l'exposé : je pense que les supports n'y sont pas pour rien. Au passage, alors que j'avais utilisé luamplib pour produire mes diapositives beamer, j'ai profité de l'écriture de cet article pour essayer minim-mp qui est une excellente alternative permettant, grâce à LuaTeX, d'utiliser du code MetaPost directement dans un source (La)TeX. Vive LaTeX, vive MetaPost !

Maxime Chupin