Le futur de LaTeX est déjà là, il s'appelle LuaLaTeX

Yves Delhaye

9 janvier 2012

Table des matières

0.1 Motivation

Je me suis intéressé à LuaLaTeX suite à la lecture du numéro 54-55 d' octobre 2010 des cahiers Gutenberg consacré à LuaTeX. Ce numéro est à la disposition des membres en téléchargement sur le site de l'association. [2]

Les possibilités perçues m'ont suffisamment motivé pour désinstaller le TeXLive2009. Celui-ci vient en standard avec le système de "packages" d'Ubuntu mais LuaLaTeX n'y est pas joint. L'ajout "à la main" de packages "LuaLaTeX" aboutissait à des résultats très décevants. J'ai donc installé TeXLive2011 en "local" et créé un package local qui m'a permis d'installer mon éditeur préféré "kile" sans occasionner de problème de dépendance. Ceci n'est jamais tout à fait trivial.

Ceux qui connaissent mon passé d'administrateur système solaris savent que je ne me lance dans pareille aventure que si j'estime que le jeu en vaut vraiment la chandelle ! Je suis physicien et enseigne la physique et les mathématiques, LaTeX est mon outil pour rédiger mes cours et mes contrôles. J'utilise de nombreux "packages" que je qualifierais de "non-triviaux" : tikz, sagetex. Je ne peux me permettre que cette chaine de production ne fonctionne plus.

J'ai écrit ce document pour me "clarifier les idées" sur LuaLaTeX. Comme le dit Sherlock Holmes : "Rien n'éclaire autant un problème que de l'exposer à un tiers !"

0.2 Le passé de LuaLaTeX s'appelle PDFLaTeX

Ou : un peu d'histoire de la technologie.

0.2.1 Moteur et format

Quand vous utilisez LaTeX, sans le savoir vous utilisez des moteurs et des formats. Savoir ce que ces mots recouvrent peut vous aider à mieux comprendre l'intérêt de LuaLaTeX.

Un moteur est un programme. Un format est un ensemble de macros utilisé par un moteur. Un même moteur peut charger des formats différents selon le nom avec lequel il est appelé. Les types de fichiers produits différeront aussi selon ce nom.

Les moteurs les plus courants sont pdfTeX, XeTeX et LuaTeX. le moteur TeX n'est plus utilisé que par les sorciers. Je le renseigne néanmoins.

En principe dans le monde "LaTeX", on produit du DVI ou du PDF. De plus le code source peut être du "Plain" [1] ou du LaTeX.

Un petit dessin vaut mieux, paraît-il, qu'un long discours. Ici c'est ce petit tableau 1 page 5 qui résumera la situation. Je reproduis partiellement le tableau trouvé dans les cahiers Gutenberg. [10] Dans ce tableau, aux quatre moteurs correspondent les quatre colonnes. Les deux premières lignes correspondent à du code source "TeX" avec soit une sortie DVI, soit une sortie PDF. Les deux dernières lignes correspondent à du code source "LaTeX" avec soit une sortie DVI, soit une sortie PDF.

Les cellules du tableau renseignent les noms avec lesquels les moteurs sont appelés.

En résumé, lorsque nous compilons un document LaTeX avec la commande

"`latex NOM_DE_FICHIER.tex`", nous invoquons le moteur "pdfTeX" pour qu'il produise du DVI, alors que si nous faisons appel à la commande "`pdflatex NOM_DE_FICHIER.tex`" nous voulons produire un PDF à partir du même fichier source et avec le même moteur.

Tout ceci pour dire que quand je parlerai de LuaTeX, ce sera le plus généralement pour parler uniquement du moteur. Quand je parlerai de LuaLaTeX, je serai plus ambigu. Je parlerai souvent de la commande mais je ferai l'abus de langage de désigner aussi le moteur par LuaLaTeX. C'est le contexte qui vous permettra d'établir la différence.

0.2.2 Lua : un langage de programmation

TeX peut tout faire. Ah bon !

TeX est considéré comme un langage de programmation complet. Il est en principe capable de tout faire.

1. ou PlainTeX

		TEX	pdfTEX	XeTEX	LuaTEX
Plain	DVI	tex	etex		dviluatex
	PDF		pdftex	xetex	luatex
LATEX	DVI		latex		dvilualatex
	PDF		pdflatex	xetex	lualatex

TABLE 1 – Formats et moteurs

Cependant, ceux d'entre nous qui ont déjà essayé de faire un calcul avec TEX ou LATEX (calculer le total des points d'une interrogation et l'afficher dans l'entête du document par exemple) savent à quel point ce genre d'aventure peut tourner rapidement au cauchemar. Nous avons aussi été confronté à des complications extrêmes en essayant de faire des calculs dans des figures Tikz.

L'ouverture du TEXbook de Knuth provoque chez moi une augmentation exponentielle de la consommation d'aspirine : prise de tête garantie !

Bref, LATEX et PDFLATEX manquent d'un langage de script y compris pour les opérations les plus simples.

LuaLATEX se veut, entre autres, une réponse à cela[2].

Lua

Lua est un langage de programmation "normal". Tout qui a fait 3 lignes de Fortran, 2 de C, et 4 de Pascal ou de Python sera capable de l'utiliser. Des boucles, des librairies de calcul, de l'orienté objet...

Ici aussi, un petit exemple rapide avec une boucle "for" sera plus parlant :

Code source n^o 1 Un exemple tout simple en lua

```
for i = 0,10,2 do
  print(i .. "\n")
end
```

Ceci affichera, bien sûr, la liste des nombres pairs entre 0 et 10.

Je fais "tourner" ce programme sur un petit smartphone bon marché.

0.2.3 LuaLATEX

Si LuaTEX, donc, est une extension de TEX et de PdfTEX qui comprend un langage de script, alors LuaLATEX est, tout logiquement, une extension de LATEX et de PdfLATEX qui comprend un langage de script.

Et, de même, si LuaTEX permet, entre autres, de faire des calculs directement dans le source d'un document TEX, LuaLATEX permet, lui aussi, de faire des calculs directement dans le source d'un document LATEX.

Pourquoi choisir LuaLATEX plutôt que LuaTEX ?

Comme expliqué grâce au tableau 1 page 5, LuaTEX est destiné aux fichiers TEX. Certes LuaTEX est capable de tout faire, mais, comme pour la différence entre TEX et LATEX, LuaTEX est plutôt pour les

2. Et beaucoup plus.

grands gourous[3] alors que LuaLaTeX permet d'utiliser ses documents LaTeX quasiment tels quels ! Si, comme moi, vous avez une énorme quantité de documents écrits en LaTeX, "y a pas photo" LuaLaTeX est la solution !

Pourquoi LuaTeX et pas pythonTeX ou javaTeX ?

Je ne suis pas le développeur mais lua présente plusieurs avantages :

1. Il est "embedable". L'interpréteur lua peut être "embarqué" à l'intérieur d'un autre programme.
2. C'est un language "général". Il est capable de faire du calcul scientifique, de traiter du texte[4], de faire de l'"orienté objet"...
3. Il est multiplateforme.
4. Il est assez léger, presque spartiate.
5. Sa syntaxe est claire.

Je connais un projet équivalent : "context". Le gros défaut de "context" est que le language de programmatique choisi est "perl" ! Perl a presque toutes les caractèristiques de lua. Mais, pour moi, sa syntaxe est cryptique ! À nouveau, je suis enseignant de mathématique et de physique et plus informaticien professionnel.

Il est remarquable que les développeurs de context et de LuaLaTeX soient les mêmes presonnes !

Différences entre LuaTeX et lua

Le lua de LuaTeX est modifié par rapport au lua de base. Il s'agit d'adaptations vers TeX. Certaines sorties de lua étaient "indigestes" pour TeX. Certains modules qui sont chargés dynamiquement par lua ont aussi été ajoutés en "statique" à LuaTeX.

0.3 Comment écrire un document LuaLaTeX ?

0.3.1 Compiler un document

Il faut très peu modifier un document LaTeX pour le compiler avec LuaLaTeX. En gros il faut commenter "`\usepackage{inputenc}`" et quelques autres packages. Je suggèrerais d'expérimenter et de mettre en commentaire le premier "package" pour lequel LuaLaTeX "râle", recompiler et recommencer jusqu'à ne plus avoir de message d'erreur.

J'essaye de lister les différences entre LuaLaTeX et LaTeX plus loin[5]

0.3.2 directlua

La commande `\direclua{ CODE CODE CODE }` permet d'appeler lua dans un document LaTeX. Il s'agit d'appeler essentiellement une seule commande lua. Des exemples simples se trouvent plus loin[6].

0.3.3 luacode

L'environnement "`luacode`" permet d'appeler du code lua s'étendant sur plusieurs lignes. Le package "`luacode`" doit être chargé en début de document.

Un exemple est donné dans le code source 4 page 8.

3. australiens, bien sûr !
4. Via, entre autres, les expressions régulières
5. voir 0.5 p. 24
6. 8 p. 14

0.3.4 Un fichier extérieur en lua

le code lua est interprété par LaTeX

LaTeX va interpréter le code lua puis l'envoyer à l'interpréteur lua embarqué de LuaLaTeX. Ceci peut être source de souci. Ainsi, le caractère "%" qui n'est pas un commentaire pour lua, va d'abord être vu par LaTeX pour qui ce caractère et tout ce qui suit doit être ignoré !

Il est plus facile est de placer le code lua dans un fichier externe. De cette manière, le code est passé directement à lua sans passer par LaTeX.

Où placer le fichier lua ?

Ça dépend !

L'avantage des fichiers externes est que l'on peut se constituer des librairies de fonctions utilisables dans tous ses documents : pour faire des tableaux de valeurs ou des graphes de fonctions en Tikz par exemple ! (voir 0.4.6 page 24)

Dans ce cas, il faut que ces fichiers soit accessibles par tous les documents LaTeX et le meilleur endroit est un sous dossier dans son "`texmf`" personnel.

Si par contre, on a du code très spécifique ne concernant que le document courant, alors il faut le mettre dans le même dossier que le document LaTeX sur lequel on travaille.

Contrairement à la commande "`\input{}`" de LaTeX qui parcourt les dossiers connus par LaTeX, la commande "`dofile`" de lua en est incapable seule. Pour que lua "trouve" ce ou ces fichiers, il faut "l'aider" avec une commande ajoutée à lua et n'existant que dans LuaLaTeX : "`kpse.find_file`". En voici un exemple

Code source n^o 2 Utilisation du kpse.find_file

```
\directlua{%
  dofile(kpse.find_file("TblFct.lua"))
}
```

La commande "`kpse.find_file`" va parcourir les dossiers où LaTeX est susceptible de trouver un fichier de style. C'est à dire, dans l'ordre : le dossier courant puis le dossier "texmf" et ses sous-dossiers et, finalement, l'arborescence de l'installation LaTeX.

Manuel Pégourié-Gonnard a été assez gentil pour relire ce texte et me fait remarquer que la commande "`require()`" est elle capable de parcourir l'arborescence LaTeX. La syntaxe serait alors :

Code source n^o 3 require() plutôt que dofile()

```
\directlua{%
  require("TblFct.lua")
}
```

0.4 Quelques exemples simples

0.4.1 Manipulation et accès aux polices de caractères

Les exemples suivants ne sont pas nécessairement les plus simples. La possibilité de modifier les polices de caractère de manière programmatique fait cependant partie des objectifs de premier plan de LuaLaTeX.

Découvrir toutes les polices de caractères sur son ordinateur

Dans cet exemple, je limite le nombre de police à dix pour limiter le place occupée mais aussi le temps de compilation !

J'ai découvert cet exemple sur "stackexchange" [4]

Code source n^o 4 Détection de fontes avec LuaLaTeX

```
\documentclass{article}

\usepackage{fontspec}
\setmainfont{Latin Modern Mono Light}

\usepackage{luacode}

\usepackage[margin=18mm]{geometry}
\parindent=0pt

\usepackage{longtable,makecell}
\renewcommand\arraystretch{2}

\begin{document}
\begin{luacode}
myfonts=dofile(fonts.names.path.localdir..'/otfl-names.lua')
teststring = "Sphinx of black quartz, judge my vow."

tex.print("\\begin{longtable}{ll}\\hline")

for i,v in ipairs(myfonts.mappings) do
  -- Stop early for testing purposes.
  if i > 10 then break end

  tex.print('\\makecell[l]{\\bfseries')
  tex.print(-2, v.familyname)
  tex.print('\\\\[-1ex] \\scriptsize')
  tex.print(-2, v.fontname)
  tex.print('} & \\LARGE\\fontspec{' .. v.fontname .. '}')
  tex.print(-2, teststring)
  tex.print('\\\\ \\hline')
end
tex.print("\\end{longtable}")
\end{luacode}

\end{document}
```

Qui donne le résultat suivant :

GFS Neohellenic Rg GFSNeohellenic-Bold	**Sphinx of black quartz, judge my vow.**
GFS Neohellenic Rg GFSNeohellenic-Italic	*Sphinx of black quartz, judge my vow.*
GFS Neohellenic Rg GFSNeohellenic-BoldItalic	***Sphinx of black quartz, judge my vow.***
GFS Neohellenic Rg GFSNeohellenic-Regular	Sphinx of black quartz, judge my vow.
Linux Libertine Display Slanted O LinLibertineDisplaySlanted0	*Sphinx of black quartz, judge my vow.*
Linux Biolinum O LinBiolinum0	Sphinx of black quartz, judge my vow.
Linux Libertine O LinLibertine0	Sphinx of black quartz, judge my vow.
Linux Libertine Slanted O LinLibertineSlanted0	*Sphinx of black quartz, judge my vow.*
Linux Libertine Slanted O LinLibertineSlanted0B	***Sphinx of black quartz, judge my vow.***
Linux Libertine Initials O LinLibertineI0	S

1

Sortie de code n^o 1: Détection de fontes avec LuaLaTeX

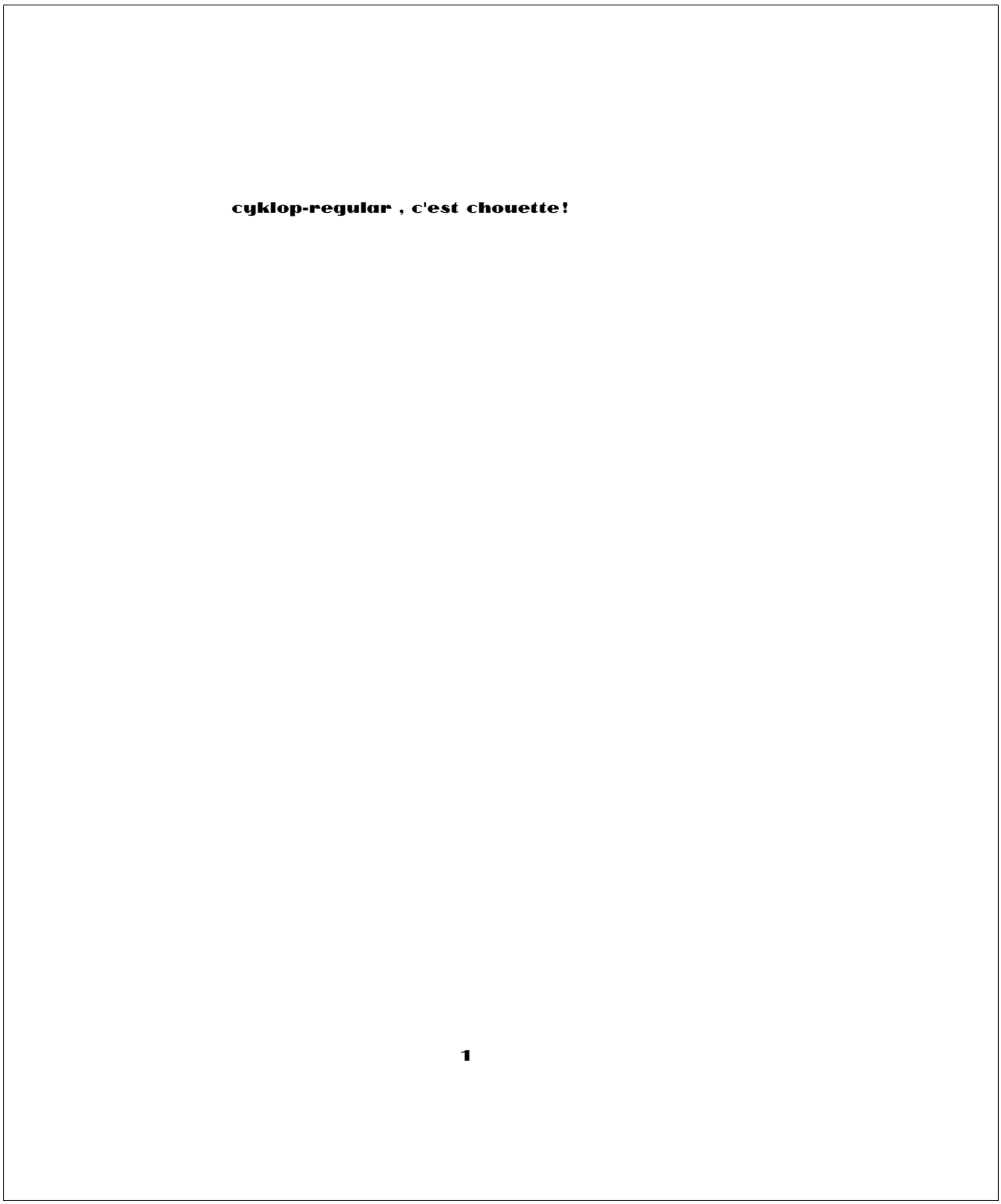
Modifier les fontes par défaut

Une fois les polices de caractères découvertes, il est fort tentant de les utiliser. Les polices ”classiques” de LaTeX sont parfois un peu tristes. Ceci est inspiré du site ”www.cuk.ch”. [9]. Franck Pastor s'y pose la question : ”LuaLaTeX, le futur de LaTeX ?” Je me permets ici de lui répondre !

Code source n^o 5 Les fontes "Cyklop" avec LuaLaTeX

```
% !TEX encoding = UTF-8 Unicode
% !TEX TS-program = LuaLaTeX
\documentclass{article}
\usepackage{fontspec}
\usepackage{luaotfload,luatextra}
\usepackage[frenchb]{babel}
% \usepackage[OT1]{fontenc}

% \setmainfont{Sexsmith}
\setmainfont{cyklop-regular}
% \setmainfont{Holy Smokes}
% \setmainfont{Turmoil BRK}
\begin{document}
% Sexsmith
cyklop-regular
, c'est chouette !
\end{document}
```

Comme prévu :



Sortie de code n^o 2: Les fontes "Cyklop" avec LuaLaTeX

0.4.2 Construction automatique d'un tableau

(d'après l'exemple de Maxime Chupin) [1]

En attendant d'avoir l'autorisation de partager le code lua, je montre ici le code LaTeX et sa sortie.

Il s'agit de faire lire par lua un fichier texte contenant des données sous forme de tableau non formaté et de créer automatiquement et dynamiquement un tableau ou une matrice au point de vue de LaTeX.

Voici les données :

Code source n^o 6 Données du tableau dynamique avec LuaLaTeX

```
2,01 4,78 7,2 7,7 3,4 6,98
3,14 15,95 2,7 6,6 8,31 7,7
```

Le code LaTeX :

Code source n^o 7 Tableau dynamique avec LuaLaTeX

```
\documentclass[a5paper]{article}
\usepackage{luatextra}
% \usepackage{lualatex-math}
\usepackage{tikz}

\directlua{%
  dofile(kpse.find_file("split.lua"))
}

\newcommand{\luatableau}[2][]{%
   \directlua{%
      tableau("#2","#1")
  }%
}

\begin{document}

\begin{tabular}{|c|c|c|c|c|c|}
  \hline
  Titre1& Titre& Titre& Titre4& Titre5& Titre6\\ \hline
  \luatableau{donnees.txt}
  \hline
\end{tabular}


\end{document}
```

Et le résultat :

Titre1	Titre	Titre	Titre4	Titre5	Titre6
2,01	4,78	7,2	7,7	3,4	6,98
3,14	15,95	2,7	6,6	8,31	7,7

1

Sortie de code n^o 3: Tableau dynamique avec LuaLaTeX

0.4.3 Calculs simples

Des calculs simples mais quasi impossibles à faire avec LaTeX seul deviennent très facilement réalisables.

Ce qui suit est aussi inspiré de ”stackexchange”. [6]

Code source n^o 8 Calcul de π avec LuaLaTeX

```
% !TEX encoding = UTF-8 Unicode
% !TEX TS-program = LuaLaTeX
\documentclass[a5paper]{article}
\usepackage{luatextra}
\usepackage[frenchb]{babel}
\thispagestyle{empty}
\begin{document}

Approximation de \(\pi\) :\\
\(\directlua{tex.sprint(math.pi)}\)

Racine carrée de 7 :\\
\(\directlua{tex.sprint(math.sqrt(7))}\)

Cosinus de  \(\pi\) sur quatre:\\
\(\directlua{tex.sprint(math.cos(math.pi / 4))}\)



\end{document}
```

Approximation de π :
3.1415926535898
Racine carrée de 7 :
2.6457513110646
Cosinus de π sur quatre :
0.70710678118655

Sortie de code n^o 4: Calcul de π avec LuaLaTeX

0.4.4 Tableaux de valeurs pour les fonctions

La conjonction des deux points précédents est particulièrement intéressante pour les enseignants de mathématique.

Pour les profs de math, la rédaction de la moindre interrogation où une fonction est utilisée implique de calculer plusieurs valeurs et donc de sortir sa calculatrice... qui justement est :

1. à l'autre étage,
2. "empruntée" par une de mes filles qui avait une interrogation,
3. à plat (comme celles de mes filles),
4. prêtée à un élève qui a oublié de vous la rendre... [7]

7. Barrez les mentions inutiles !

Et s'il y a des absents, il faut refaire une interrogation différente et ressortir sa calculatrice qui justement est ... STOP ! Fort heureusement, il y a LuaLaTeX !

En combinant, la génération de tableau et la capacité d'éffectuer des calculs de LuaLaTeX, plus besoin de calculatrice. voici le code que j'utilise personnelement pour mes interrogations. Il est certes perfectible, mais, comme vous bientôt, je suis encore en train de découvrir toutes les possibilités de LuaLaTeX.

Code source n^o 9 De jolis tableaux de fonction avec LuaLaTeX.

```
tex.print("\\begin{tabular}{c|c}")
tex.print("x & y \\\\\\hline")
for i=-1,5,1.0 do
  tex.print(i .. "&" .. (-i^2 + 6*i - 8)  .. "\\\\");
end
tex.print("\\end{tabular}")
```

Ce qui donne :

x	y
-1	-15
0	-8
1	-3
2	0
3	1
4	0
5	-3

Sortie de code n^o 5: De jolis tableaux de fonction avec LuaLaTeX.

Quelques remarques concernant l'exemple précédent :
- Le séparateur décimal est le point selon la convention anglo-saxonne ;
- Il est possible, mais plus compliqué, de générer un tableau horizontal ;
- la fonction choisie ici est "sans histoire".

Le remplacement du point par la virgule

Si, justement, nous choisissons une fonction avec des discontinuités, la puissance de LuaLaTeX devient flagrante :
- Il est possible de fixer le nombre de chiffre après la virgule ;
- la divison par zéro produit, avec lua, la sortie "inf" ; la substitution de "inf" par le symbole ∞ est triviale ;
- la subsection du point comme séparateur décimal est moins triviale mais tout à fait réalisable.

La substitution du point n'est pas triviale car la commande lua utilisée `string.gsub` emploie les expressions régulières. Or, dans la syntaxe des expressions régulières, le point signifie "n'importe quel caractère". La commande "`string.gsub(UnTexteAvecDesPoints..... ,".","","")`" destinée à remplacer les points par des virgules avec l'espoir d'obtenir "`UnTexteAvecDesPoints,,,,, `" va remplacer *tous* les caractères, y compris les espaces car ce sont aussi des caractères, par des virgules : "`,,,,,,,,,,,,,,,,,,,,,,,,,,`".

La solution consiste à utiliser un caractère d'échapement. Ceci revient à faire en sorte que le point ne soit plus reconnu comme le méta-caractère "remplacement de tout caractère" mais "bêtement" comme un point. Avec lua le caractère d'échapement est "%". Pas de chance, pour LaTeX, c'est la commande de mise en commentaire ! Il faut donc empêcher LaTeX de reconnaître le "%" comme commande de mise en commentaire. La solution consiste de nouveau à utiliser un caractère d'échapement mais pour LaTeX cette fois. Bref, j'utilise la commande "`local pattern = '\%.'`". La variable "pattern" est déclarée locale pour éviter les surprises. Cette plaisanterie m'a coûté quelques heures de frustration !

Code source n^o 10 De PLUS jolis tableaux de fonction avec LuaLaTeX.

```
function round(num, idp)
  local mult = 10^(idp or 0)
  return math.floor(num * mult + 0.5) / mult
end

function FctTbl(i)
  local LaFct = (2/(i+1)) -1
  LaFct = round(LaFct, 3)
  LaFct = string.gsub(LaFct,"inf","$\\infty $")
  local pattern = '\%.'
  LaFct = string.gsub(LaFct,pattern,",")
  return LaFct
end

tex.print("\\begin{tabular}{c|c}")
tex.print("x & y \\\\\\hline")
for i=-4,4,0.5 do
  tex.print(i .. "&" .. FctTbl(i)  .. "\\\\");
end
tex.print("\\end{tabular}")
```

x	y
-4	-1,667
-3.5	-1,8
-3	-2
-2.5	-2,333
-2	-3
-1.5	-5
-1	∞
-0.5	3
0	1
0.5	0,333
1	0
1.5	-0,2
2	-0,333
2.5	-0,429
3	-0,5
3.5	-0,556
4	-0,6

Sortie de code n^o 6: De PLUS jolis tableaux de fonction avec LuaLaTeX.

Encore plus joli : "eval" sous lua = "loadstring"

L'idée "ultime"[8] est de fournir une expression dans le code LaTeX qui produira ce tableau de valeurs. Il s'agit donc de faire un "eval" de cette expression.

Le code LaTeX Voici le code LaTeX que j'utilise.

8. Pour aujourd'hui ! Je me connais.

Code source n^o 11 La commande TableauFonction appelle le code lua.

```
\documentclass{article}
% \usepackage{fontspec}
% \usepackage{luaotfload,luatextra}
\usepackage{verbatim}
\usepackage{amssymb}
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
\directlua{%
  dofile(kpse.find_file("TableauFonction.lua"))
}

\newcommand{\TableauFonction}[1]{%
   \directlua{%
   TblFct("\luatexluaescapestring{(#1)}")
  }%
}
%


\begin{document}

 \TableauFonction{x^2 + 4*x -9 + math.sqrt(x) + 1/x}

\end{document}
```

Il faut "charger" le package "amssymb" pour avoir le symbole ∞

L'évaluation de l'expression mathématique écrite dans la syntaxe lua "`x^2 + 4*x -9 + math.sqrt(x) + 1/x`" est confié via la commande LaTeX"`\TableauFonction{}`" à une fonction lua : "TblFct".

Le code lua Examinons maintenant le fichier "TableauFonction.lua" où la fonction "TblFct" est définie.

Code source n^o 12 Le fichier TableauFonction.lua.

```
require "math"
--
function eval(str)
  local f = assert(loadstring("return" .. str))
  return f()
end

function round(num, idp)
  local mult = 10^(idp or 0)
  return math.floor(num * mult + 0.5) / mult
end

function FctSub(str)
  local str = round(str, 3)  -- manipulation num avant traitememnt de string
--    str = round(str, 3)
  str = string.gsub(str,"-nan","$\\nexists $")
  str = string.gsub(str,"nan","$\\nexists $")
  str = string.gsub(str,"inf","$\\infty $")
--    str = string.gsub(str,"- \\infty","\\infty")
  local pattern = '\%.'
  str = string.gsub(str,pattern,",")
  return str
end


function evalsub(str)
  local fx = eval(str)
  fx =  FctSub(fx)
  return fx
end

--
function TblFct(expr)
--
--    local f = assert(loadstring("return" .. expr))
--
  tex.print("\\begin{tabular}{c|c}")
  tex.print("x & y \\\\\\hline")
  tex.print(" &  \\\\")
--
  for i=-1,5,1.0 do
    x = i   -- ATTENTION x DOIT etre GLOBAL et pas local!
    local fx =  evalsub(expr)
--[[    tex.print(i .. "&" .. f()  .. "\\\\");]]
--[[    tex.print(i .. "&" .. eval(expr)  .. "\\\\");]]
    tex.print(i .. "&" .. fx  .. "\\\\");
  end
--
  tex.print("\\end{tabular}")
--
end
```

Je commence par définir plusieurs fonctions qui vont être finalement imbriquées. Le but est de rendre le code modulable : Je veux pouvoir changer l'ordre des opérations facilement et je trouve que ceci rend le code plus lisible.

Les fonctions lua Examinons ces fonctions.

La fonction ”eval” existe telle quelle dans de nombreux languages de programmation mais pas dans lua. La commande ”loadstring” permet cette évaluation. Il faut concaténer (les deux points ”..”) l'expression (un ”string” : `str`) avec un ”return”. Le ”assert” sert à gérer, très partiellement, les erreurs.

La fonction ”round” sert à gérer les arrondis.

”FctSub” fait des substitutions depuis les sorties lua vers du code LaTeX ”plus joli”. Les substitutions devraient être placées dans un tableau parcouru par une boucle dans une prochaine version. Des ”switch” devraient permettre de faire différentes substitutions selon les conventions de syntaxe mathématique des différents pays.

”evalsub” combine les deux fonctions ”eval” et ”FctSub”

Finalement ”FctTbl” génére le tableau.

Les deux signes moins ”–” sont la mise en commentaire de lua. Je laisse ainsi les traces d'essais divers pour le lecteur curieux.

Le tableau Le tableau suivant a été créé par le code précédent. Il ne faut plus toucher au code lua mais uniquement au code LaTeX pour le produire. Ce qui a l'avantage, entre autres, d'une plus grande lisibilité.

x	y
-1	$\nexists$
0	∞
1	-2
2	4,914
3	14,065
4	25,25
5	38,436

Sortie de code n^o 7: Tableau "dynamique" LuaLaTeX

Le futur La macro LaTeXpeut (et doit) être améliorée pour accepter les valeurs de début, de fin et d'incrément de la boucle "for" de la fonction "TblFct". Des signes moins devant les infinis ou les nan[9] produisent encore des résultats "bizarres". Un tableau de substitutions parcouru par une boucle doit être écrit.

0.4.5 Calculs plus compliqués

(d'après l'exemple de Maxime Chupin) [1]

Des calculs plus compliqués, comme du calcul matriciel, sont relativement simples[10]. Faire faire un ajustement aux moindres carrés à LuaLaTeX et afficher le résultat graphiquement n'est pas plus compliqué que dans l'exemple suivant.

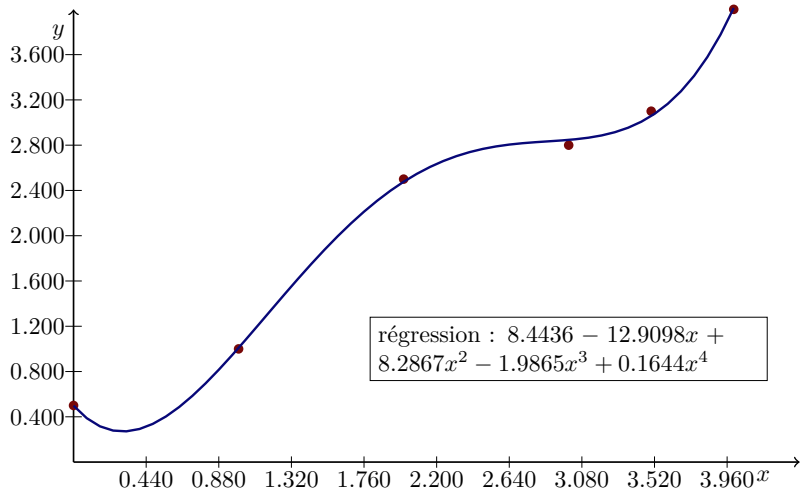
Code source n^o 13 Moindres carrés avec LuaLaTeX

```
\documentclass{article}
\usepackage{luatextra}
% \usepackage{lualatex-math}
\usepackage{tikz}

\directlua{%
  dofile(kpse.find_file("moindre.lua"))
}

\newcommand{\moindreInt}[2]{%
   \directlua{%
      moindre("int","\luatexluaescapestring{#2}","\the\linewidth","\luatexluaescapestring{#
  }%
}

\newcommand{\moindreExt}[2]{%
   \directlua{%
      moindre("Ext","\luatexluaescapestring{#2}","\the\linewidth","\luatexluaescapestring{#
  }%
}

\definecolor{point}{RGB}{120,8,8}
\definecolor{regression}{RGB}{8,8,120}

\begin{document}

\moindreInt{4}{(1,2);(2,2.5);(3,4);(4,4.3);(5,5.5);(4.5,4.6)}

\end{document}
```

Le résultat 8 page 23 est particulièrement joli !

Les points rouges sont les points dont les coordonnées[11] sont fournies dans le code 13 page 22. L'ajustement polynomial dont le degré 4 est demandé dans la commande "`\moindreInt{4}{ ... }` est lui affiché en bleu et le polynôme lui-même est affiché dans une boîte.

9. Not A Number : racine carrée de -1 par exemple !
10. Càd. comme avec n'importe quel langage de programmation "standard".
11. càd. : (1,2) ;(2,2.5) ;(3,4) ;(4,4.3) ;(5,5.5) ;(4.5,4.6)



Sortie de code n^o 8: Ajustement aux moindres carrés avec LuaLaTeX

0.4.6 Scriptages de Tikz

(d'après l'exemple de Maxime Chupin) [1]

Dans les "cahiers Gutenberg", Maxime Chupin décrit comment il ajuste la largeur de l'image tikz en fournissant "`\thepagewidth`" à un script lua. Il utilise également lua pour calculer les coordonnées de points d'une fonction polynomiale et construire un graphe tikz à partir de ces points. Le résultat est présenté dans la figure 8 page 23.

La fonction "eval" qui me sert à construire le tableau devrait aussi permettre de faire un graphe tikz en utilisant la commande "plot" de tikz.

Utiliser la syntaxe LaTeX (ou lua) une seule fois et faire apparaître et le tableau et le graphe doit être possible. Une expérience dans ce sens existe sous "context".

0.4.7 Calculs pour manipuler des fontes

Exemple trouvé sur le web

0.4.8 Autres possibilités

callback

les "callback" sont des fonctions qui redéfinissent des primitives TeX ou LaTeX....

metapost

LuaLaTeX comprend une version embarquée de "Metapost".

0.5 Différence entre un document LaTeX et un document LuaLaTeX

0.5.1 Caveat !

inputenc, c'est fini !

Vive "fontspec" !

Il ne faut plus utiliser "`inputenc`" mais "`fontspec`". L'encodage des caractères accentués se fait via "EU2" et plus via "OT1" comme avec "PDFLaTeX".

Il suffit de mettre "`\usepackage{fontspec}`" dans le préambule et "fontspec" charge "EU2".

Il me semble qu'il faut lancer la commande "`mkluatexfontdb.lua`" qui doit constituer une base de données des polices de caractères présentes sur l'ordinateur.

On peut aussi faire un appel explicite à à fontenc : "`\usepackage[EU2]{fontenc}`" mais ceci est redondant avec "`fontspec`".

microtype non merci

L'utilisation des formules de calcul de tikz avec un babel français posait des problèmes. Les formules de tikz permettant, par exemple, de placer un point systèmatiquement entre deux autres à "x %" de la distance entre ces points utilisent des points d'exclamation "`!`". Dans la typographie française, les points d'exclamation doivent être "décalés". De même, deux "f" successifs seront légèrement déplacés. Comparez "ff" (avec kerning) et "`ff`" (en verbatim, donc sans kerning).

Cette mise au point typographique ne plaît pas du tout à tikz qui provoque une erreur de compilation. L'astuce était d'ajouter ceci au préambule : "`\usepackage[babel=true,kerning=true]{microtype}`".

LuaLaTeX doit régler les problèmes de "microtype". Les problème de "kerning" avec Tikz semblent être réglés.

J'ai, en effet, pu supprimer le "`\usepackage[babel=true,kerning=true]{microtype}`" qui était obligatoire pour éviter le "clash" entre le babel "frenchb" et tikz.

Ceci ne pourrait être que transitoire. Le fait que microtype ne fasse pas de kerning sous LuaLaTeX est plutôt vu comme un défaut à corriger par les développeurs.

0.5.2 Avec LuaLaTeX la vie est belle.

UTF8

Utf8 est l'encodage par défaut de LuaLaTeX. Plus besoin de l'appler via "inputenc". L'encodage "ISO-Latin" peut cependant toujours être utilisé. Je n'ai pas encore investigué dans ce sens.

0.5.3 Autres classes de document

Je n'ai pas encore fait d'essai sérieux avec d'autre classe qu' "article". Mes interregotions et examens étant écrits sous cette classe, elle est prioritaire pour moi.

Beamer

Beamer, en particulier, va devoir être investigué. Les membres de l'équipe LaTeX de l'UREM utilise préférentiellement beamer pour leurs présentations.

0.6 Pistes

0.6.1 Le générateur d'interrogation... again

J'ai depuis longtemps un vieux projet de générateur automatique d'interrogations. [8]

LuaLaTeX devrait permettre un générateur d'interrogation beaucoup plus facile à mettre en œuvre et plus multiplateforme que la solution actuelle avec SAGE.

0.7 Conclusion

0.7.1 Avantages indéniables

La création "automatisée" de tableaux de valeurs est déjà un avantage presque incroyable dont on ne pouvait rêver avec LaTeX seul. L'accès aux fontes ajoute une touche de luxe.

0.7.2 Problèmes et évolutions souhaitées

Il faut apprendre un nouveau language de programmation. Pour moi, c'est un plaisir toujours ambigu. Lua me donne parfois le sentiment d'être un peu "spartiate".

L'intégration avec metapost est, semble t'il, encore limitée.

J'ai aussi un peu de difficulté à trouver de la documentation.

0.7.3 Pour en finir !

Attention l'usage de LuaLaTeX est addictif. Depuis que j'y ai touché, je ne sais plus m'en passer !

0.8 Annexes diverses en attendant

Liste des tableaux

Liste des codes sources

Listes des sorties de code

Bibliographie

[1] Maxime Chupin. Lualatex pour les non-sorciers, deux exemples. *Cahiers GUTenberg*, (54-55), Octobre 2010.

[2] Collectif. Cahiers gutenberg. http://cahiers.gutenberg.eu.org/.

[3] Collectif. L'article wikipedia sur luatex. http://en.wikipedia.org/wiki/LuaTeX.

[4] Collectif. Lualatex : Font table with examples. http://tex.stackexchange.com/questions/23630/lualatex-font-table-with-examples.

[5] Collectif. The programming language lua. http://www.lua.org/.

[6] Collectif. What is a simple example of something you can do with luatex? http://tex.stackexchange.com/questions/70/what-is-a-simple-example-of-something-you-can-do-with-luatex.

[7] Collectif. Wiki luatex. http://wiki.luatex.org.

[8] Yves Delhaye. Générateur d'interrogations. http://www.yvesdelhaye.be/?-Generateur-d-interrogations-.

[9] Franck Pastor. Lualatex, le futur de latex? http://www.cuk.ch/articles/4854.

[10] Manuel Pégourié-gonnard. Un guide pour lualatex. *Cahiers GUTenberg*, (54-55), Octobre 2010.

[11] Will Robertson. Fontspec for luatex. http://latex-alive.tumblr.com/post/643148677/fontspec-for-luatex.