MFLUA 0.5

MFLUA

A fully compatible implementation of METAFONT:

- METAFONT + Lua, as in LuaTEX;
- (still) not a library as `mplib`;
- (still) in PASCALWEB;
- first appearance at BachoTEX meeting 2010.

A METAFONT program can be executed unmodified by MFLUA, giving the same result if MFLUA doesn't use Lua scripts.

MFLUA

Why another implementation of METAFONT ?

- The main output of METAFONT is a *bitmap* version of the draw described by *mathematical and vectorial* instructions;
- but ``*METAFONT works internally with outlines*'' (cubic Bézier curve);
- these outlines can be recorded in the log file, but their post-processing is not easy.

MFLUA can save bitmap and curves into Lua tables and *these* are easier to process.

MFLUA

Do we need the mathematical/parametric type design of METAFONT ?

Let's make an example:

- an OpenType font has 65,536 (or 2^{16}) glyphs;
- a font is part of a *family* (i.e. Serif, Serif Bold, Italic, Italic Bold);
- a family is part of a *super-family* (i.e. Serif, Sans, Handwritten, Monospace)

We can have up to $4\cdot 4\cdot 2^{16}=2^{20}$ glyphs — about *one million*.

Taking a working year of 2000 hours and 8 hours to finish a glyph, a quality font with 2000 glyphs (or a 4-family of 500 glyphs) takes 8·2000/2000 = 8 *man-year*s or 2.5 / 3 years for a team of three persons, covering the *0.19%* of number of the potential glyphs.

MFLUA

The mathematical/parametric type design of METAFONT:

- capture the elements of the font (the style) in a *mathematical* description;
- speed up the the creation of the variants by mean of *parameters*.

But:

- bitmaps alone are not enough — nowadays fonts need curves;
- programming alone is not enough — a designer needs *to see and program*.

MFLUA

MFLUA can:

- work with bitmaps and curves;
- use Lua to connect METAFONT to a Graphics User Interface (GUI) libraries;
- use Lua to export the curves into a font format;
- use Lua to enhance the math of METAFONT.

MFLUA

MFLUA *cannot* replace the designer.

In font design, the designer is the most important part — not the programs used.

Every designer has his own set of tools: MFLUA could be another one.

MFLUA 0.5

MFLUA 0.5 is the first release of MFLUA.

It follows the same rules of MetaPost and LuaTEX:

- SVN repository at https://foundry.supelec.fr/projects/mflua;
- synchronized with TEXLive;
- a set of PASCALWEB `change` files and not a single source.

New in this release:

- `runscript` primitive to execute Lua scripts;
- MFLUAJIT.

MFLUA 0.5

Example of `runscript`:

```
numeric r;
numeric t[];
r:=0;
r:=runscript(
 "return (math.sqrt(5)*math.sqrt(3))"
);
message "DEBUG r=" &  decimal r; message "";
runscript(
 "local t={2.2,3.3,1.1};                  "&
 "table.sort(t);                          "&
 "local s = 't[1]:=%f;t[2]:=%f;t[3]:=%f;' "&
 "return string.format(s,t[1],t[2],t[3])  "
 );
message "DEBUG t[]=" & decimal t[1] & "," &
        decimal t[2] & "," & decimal t[3];
message "";
end.
```

gives:

```
DEBUG r=3.87299
DEBUG t[]=1.1,2.2,3.3
```

MFLUA 0.5

Example of `runscript` and MFLUAJIT:

```
numeric r;
r = runscript(
"local ffi = require('ffi')    " & char(10) &
"ffi.cdef[[                    " & char(10) &
"double     erf( double arg );" & char(10) &
"]]                            " & char(10) &
"return ffi.C.erf(-3)          "
);
message "DEBUG erf(-3)=" & decimal(r);
end
```

gives:

```
DEBUG erf(-3)=-0.99998
```

MFLUA 0.5

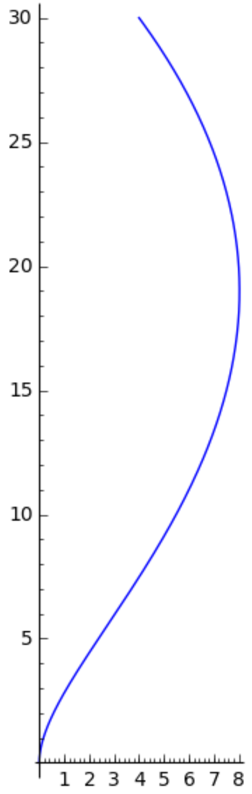
Of course it possible to load a binding
(e.g. from the SWIGLIB project):

```
runscript(
"local cpath =
   './modules/parigp/linux/resources/lib64/?.so'" & char(10) &
"package.cpath = package.cpath .. ';'  .. cpath " & char(10) &
"local pari = require('core')                   " & char(10) &
"pari.pari_init(4000000,500000)                 "
```

In the paper there is an example of implicitization of a cubic Bézier curve $z(t)$ for $-\infty < t < \infty$.

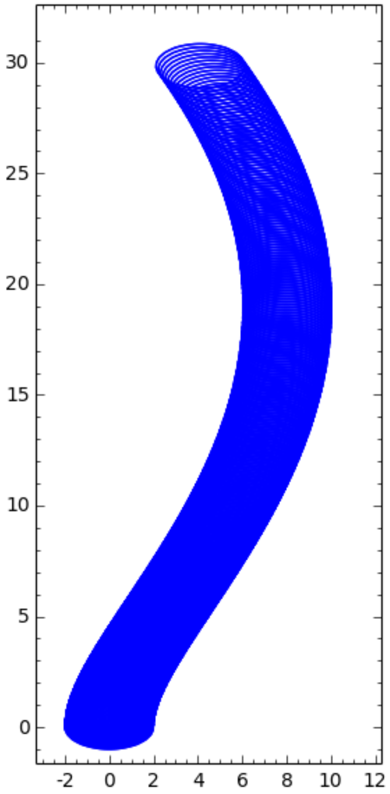
MFLUA 0.5

One problem nice to explore are the outlines of a pen: given a Bézier curve



MFLUA 0.5

an elliptic pen with its center on the curve gives a track



MFLUA 0.5

With SAGEMATH it's possible to have the implicit curve X, Y of the envelope for $-\infty < t < \infty$.

```
┌────────────────────────────────────────────────────────────────────┐
│ Sage Version 6.5, Release Date: 2015-02-17                         │
│ Type "notebook()" for the browser-based notebook interface.        │
│ Type "help()" for help.                                            │
└────────────────────────────────────────────────────────────────────┘
sage: P.<t,x,y> = PolynomialRing(QQ,3,order='lex')
sage: a=2; b=1; px=0; py=0; c1x=0; c1y=5; c2x=15; c2y=15; qx=4; qy=30;
sage: ell = (x-((1-t)^3*px+3*(1-t)^2*t*c1x+3*(1-t)*t^2*c2x+t^3*qx))^2/a^2+(y-
((1-t)^3*py+3*(1-t)^2*t*c1y+3*(1-t)*t^2*c2y+t^3*qy))^2/b^2-1
sage: idcnt = ideal(ell,derivative(ell,t))
sage: idcnt
Ideal (1681/4*t^6 - 1845/2*t^5 + 2925/4*t^4 + 41/2*t^3*x + 450*t^3 - 45/2*t^2*x
- 30*t^2*y + 225*t^2 - 30*t*y + 1/4*x^2 + y^2 - 1, 5043/2*t^5 - 9225/2*t^4
+ 2925*t^3 + 123/2*t^2*x + 1350*t^2 - 45*t*x - 60*t*y + 450*t - 30*y) of
Multivariate Polynomial Ring in t, x, y over Rational Field
sage: envelope =  idcnt.elimination_ideal(t)
```

MFLUA 0.5

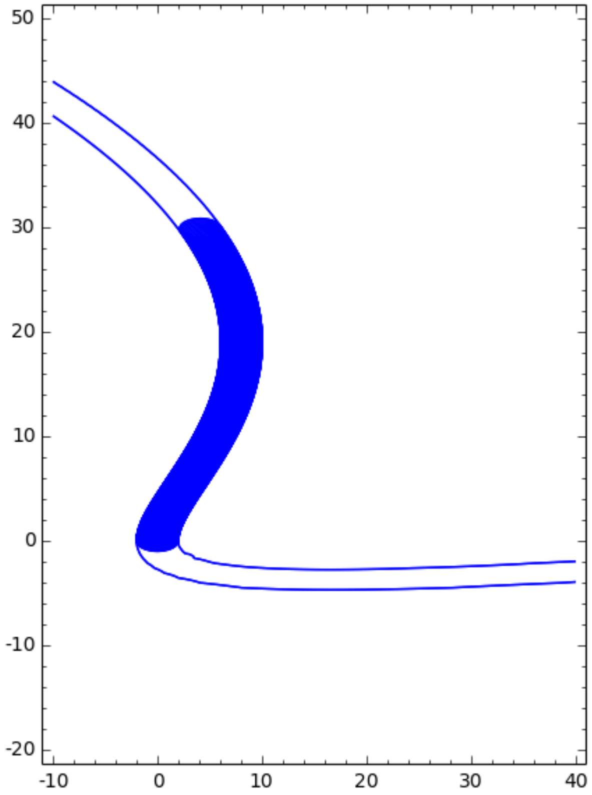
```
sage: envelope
Ideal (32187183890625*x^8 - 914116022493750*x^7*y - 4421802944812500*x^7 -
3206320362675O*x^6*y^3 + 7854960356668125*x^6*y^2 + 42281261393582250*x^6*y
+ 92520167866194375*x^6 + 455297491499850*x^5*y^4 - 2138838463792000*x^5*y^3 +
25827949140435000O*x^5*y^2 + 210397974331334400O*x^5*y - 1262742117879524850*x^5
+ 7984925229121*x^4*y^6 - 807992731394100*x^4*y^5 + 9112652596408013 7*x^4*y^4
- 1099154919606643800*x^4*y^3 - 21305294570209451 37*x^4*y^2 +
9689907949416083790O*x^4*y + 42753587364110826587 9*x^4 + 3918835998825000*x^3*y^6
- 20677832775471600O*x^3*y^5 + 359270957149760580O*x^3*y^4
+ 1097808630451148700O*x^3*y^3 - 1559991395222306616OO*x^3*y^2 -
568401677557565571000*x^3*y - 261860705746712926920O*x^3 + 63879401832968*x^2*y^8
- 6063033053212400*x^2*y^7 + 370309820722113128*x^2*y^6
- 6747050048879526800*x^2*y^5 - 1404651123504004219 2*x^2*y^4 +
4778989807717556908O0*x^2*y^3 + 361384949063715656312 8*x^2*y^2 +
```

MFLUA 0.5

```
3920024076646434548400*x^2*y + 3804377376480379782968*x^2 + 8390584131302400*x*y^8
- 53375030922008000*x*y^7 + 920032985247361840 0*x*y^6 - 2899922813666 60000*x*y^5
- 8748491720270451696 00*x*y^4 - 14788925401534891 40000*x*y^3 -
1954113029275752225600*x*y^2 + 2240001537762688380000*x*y
+ 1049491509914305747440 0*x + 127758803665936*y^10 - 1337629354265600 0*y^9 +
5148423486693303 20*y^8 - 86326305190081680 00*y^7 + 4868618312216128936 0*y^6 +
1505445743365184400 00*y^5 + 1215080186847031906 40*y^4 - 2019611646770171752000*y^3
- 1124512906937757365 0320*y^2 - 1723318757922150086 4000*y
- 2206401301484176382 5936) of Multivariate Polynomial Ring in t, x, y over
Rational Field
```

MFLUA 0.5

The implicit plot of this polynomial is:



MFLUA 0.5

Of course this half of the story (maybe even less):

- is it possibile to translate polynomial description into a set of ``good'' Bézier curves ?
- how does it compare with the linearization of the ellipse of METAFONT and the tracing of POTRACE ?
- is it algebraic geometry / commutative algebra the right tool to use ?

MFLUA 1.0

The next MFLUA 1.0 is planned for TEXLive 2016.

Still missing:

- integration with `kpathsea`;
- a manual like the MetaPost one;
- a small set of C/Lua libraries, possibly part of SWIGLIB.

That’s all !

Thank you Folks !