Strumentazione di METAFONT con Lua

MFLua

- Strumentazione del codice PASCAL-WEB di METAFONT con funzioni Lua (embedding dell'interprete Lua)
- completamente compatibile con METAFONT 2.718281
- https://github.com/luigiScarso/mflua

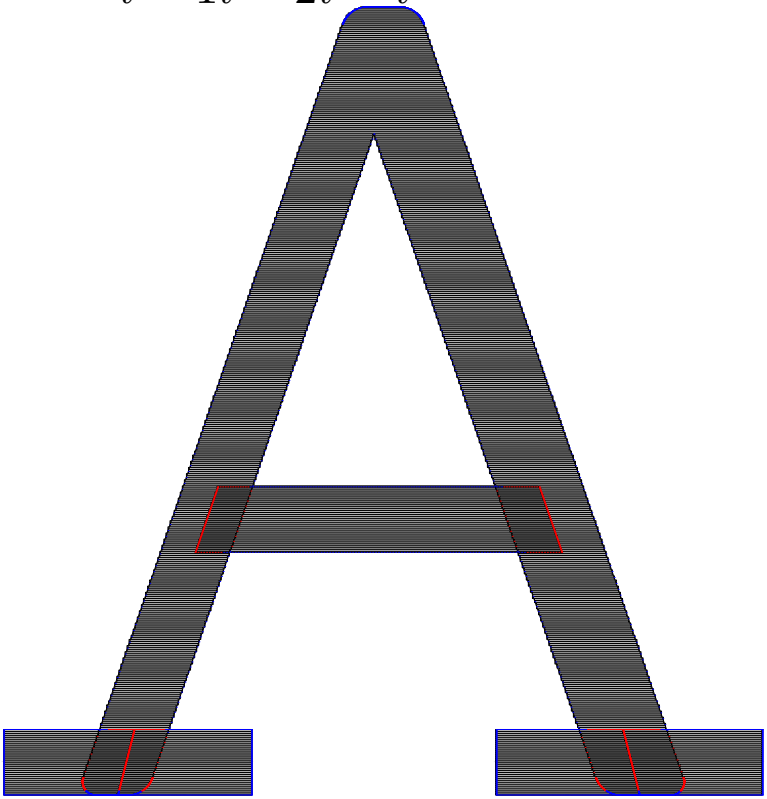
Perché ?

- METAFONT elabora una descrizione ad alto livello di un glifo e produce una immagine bitmap
- le curve del contorno sono rintracciabili nel log (se il tracing è attivo), ma il post-processing è poco agevole
- MFLua salva le curve in tables di Lua, rendendo più agevole il post-processing
- ...for fun...

In pratica MFLua non differisce da METAFONT: il modo ha una risoluzione di 7200dpi (con design size di 10bp significa 1em=1000unità, tipico dei font Type1)

```
mode_def otcff =
  mode_param (pixels_per_inch,3600+3600);
  mode_param (blacker, 0);
  mode_param (fillin, 0);
  mode_param (o_correction, 1);
  mode_common_setup_;
enddef;
```

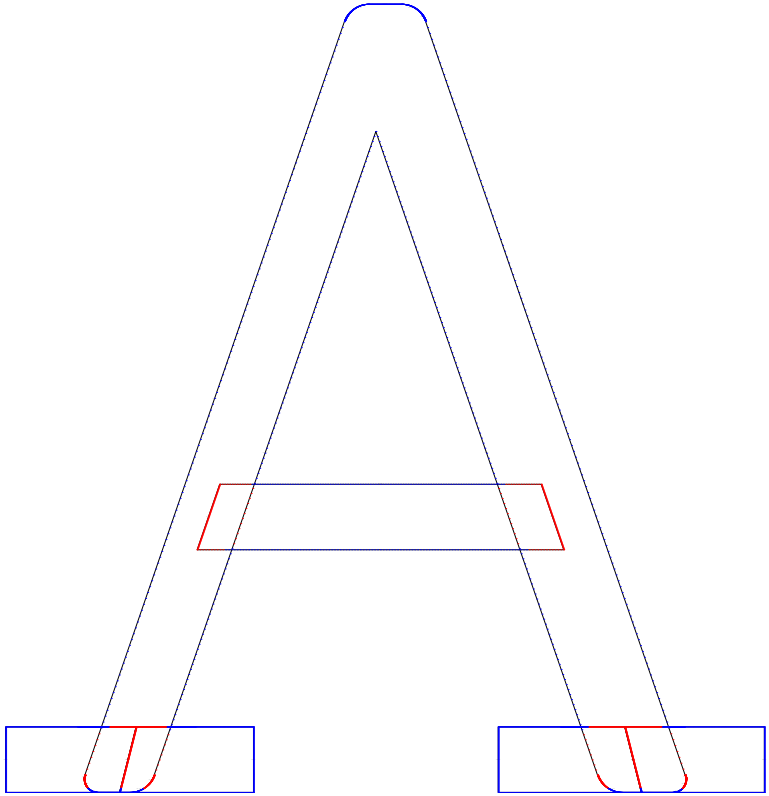
Durante l'esecuzione le funzioni collezionano le curve cubiche p_i, c_{1i}, c_{2i}, q_i e i pixels del glifo.



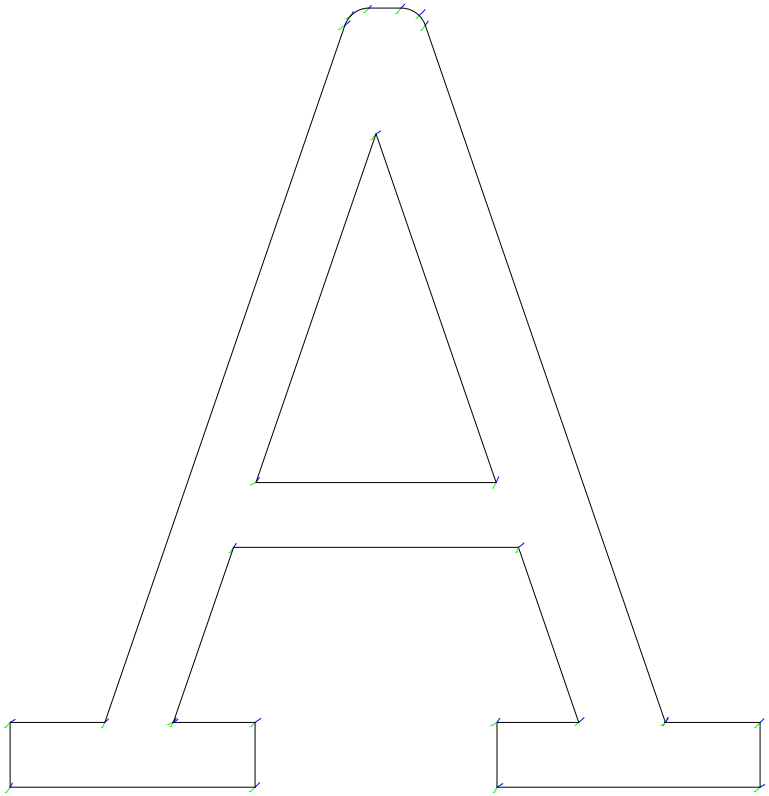
Ciascuna procedura (o “sensore”) chiama a sua volta un file Lua:

```
begin_program.lua
bezier.lua
do_add_to.lua
end_program.lua
end_program_poly_to_bezier.lua
fill_envelope.lua
fill_spec.lua
main_control.lua
make_ellipse.lua
mfluaini.lua
mflua_svg_backend.lua
namelist.lua
offset_prep.lua
parse-log.lua
pen_curves.lua
poly_to_bezier.lua
print_edges.lua
print_path.lua
scan_direction.lua
simplify.lua
skew_line_edges.lua
start_of_MF.lua
tfm.lua
transition_lines.lua
```

Giusto prima che MFLua termini, la funzione Lua `end_program()` “ripulisce” le curve partendo da

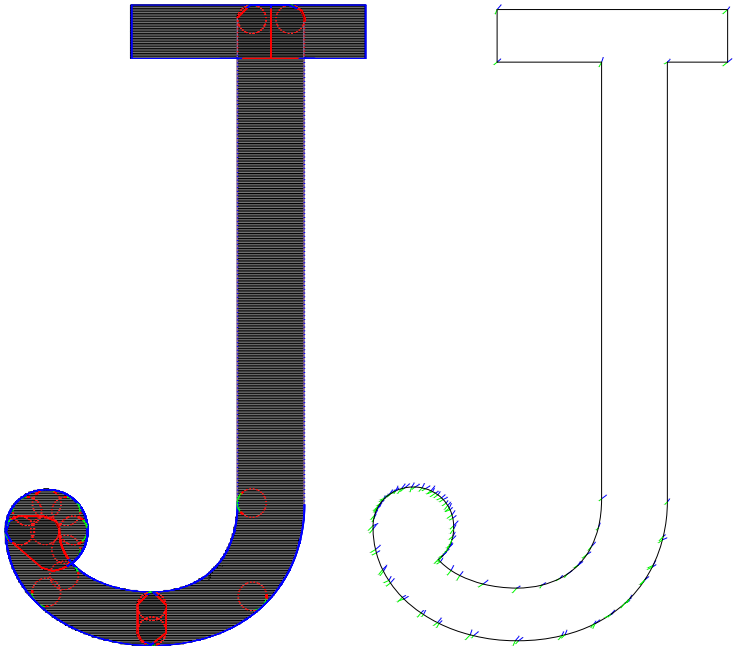


per arrivare a



e produce un font SVG.

L'insieme delle curve di un glifo può essere complicato, specialmente se si usano le penne:



Il programma FontForge può quindi essere usato per convertire un font SVG in un OpenType CFF in uno dei seguenti modi:

- con una tradizionale sessione di editing;
- con uno script FontForge da `end_program()` per mezzo di `os.execute(cmd)`;
- estendendo Lua con un binding per FontForge

Primo risultato:

- Il font SVG `ConcreteOT.svg` da `ccr10.mf` (a 4000 dpi)

- FontForge è stato usato per semplificare le curve and convertire il font SVG in un font OpenType CFF (`ConcreteOT.otf`);

- `ConcreteOT.otf` è il font usato in queste slides, e una versione Type1 è stata usata per l'articolo per il BachoTEX meeting di quest'anno (àèéîòù aggiunti con FontForge per il GUIT meeting)

! 33 exclam	# 35 numbersign	$ 36 dollar	% 37 percent	& 38 ampersand
’ 39 quotesingle	(40 parenleft	) 41 parenright	* 42 asterisk	+ 43 plus
, 44 comma	- 45 hyphen	. 46 period	/ 47 slash	0 48 zero
1 49 one	2 50 two	3 51 three	4 52 four	5 53 five
6 54 six	7 55 seven	8 56 eight	9 57 nine	: 58 colon
; 59 semicolon	= 61 equal	? 63 question	@ 64 at	A 65 A
B 66 B	C 67 C	D 68 D	E 69 E	F 70 F
G 71 G	H 72 H	I 73 I	J 74 J	K 75 K
L 76 L	M 77 M	N 78 N	O 79 O	P 80 P
Q 81 Q	R 82 R	S 83 S	T 84 T	U 85 U

[91 bracketleft	] 93 bracketright	` 96 grave	a 97 a	b 98 b
c 99 c	d 100 d	e 101 e	f 102 f	g 103 g
h 104 h	i 105 i	j 106 j	k 107 k	l 108 l
m 109 m	n 110 n	o 111 o	p 112 p	q 113 q
r 114 r	s 115 s	t 116 t	u 117 u	v 118 v
w 119 w	x 120 x	y 121 y	z 122 z	¡ 161 exclamdown
¯ 175 macron	´ 180 acute	¸ 184 cedilla	¿ 191 questiondown	Æ 198 AE
Ø 216 Oslash	Œ 338 OE	ˇ 711 caron	˘ 728 breve	˚ 730 ring
ˆ 770 uni0302	˜ 771 tildecomb	̇ 775 uni0307	¨ 776 uni0308	̋ 779 uni030B
̷ 823 uni0337	Γ 915 Gamma	Δ 916 Delta	Θ 920 Theta	Λ 923 Lambda

Ψ 936 Psi	Ω 937 Omega	– 8211 endash	— 8212 emdash	‘ 8216 quoteleft
“ 8220 quotedblleft	” 8221 quotedblright	ff 64256 uniFB00	fi 64257 uniFB01	fl 64258 uniFB02
ffi 64259 uniFB03	ffl 64260 uniFB04	□ 983040 .notdef		

Problemi

- `end_program()` troppo complicata
- approssimazione poligonale delle penne: molte curve (gestione complicata, glifo di bassa qualità)

end_program() troppo complicata

- elaborare contorni, inviluppi e penne separatamente
- meno sotto-funzioni, ma più codice

```
1) _remove_useless_curves
2) _simplify_curves
3) _remove_loops
4) _merge_envelopes_and_pens
5) _merge_envelopes_and_contours
6) _simplify_merged_curves
7) _build_cycle
```

- Debug:

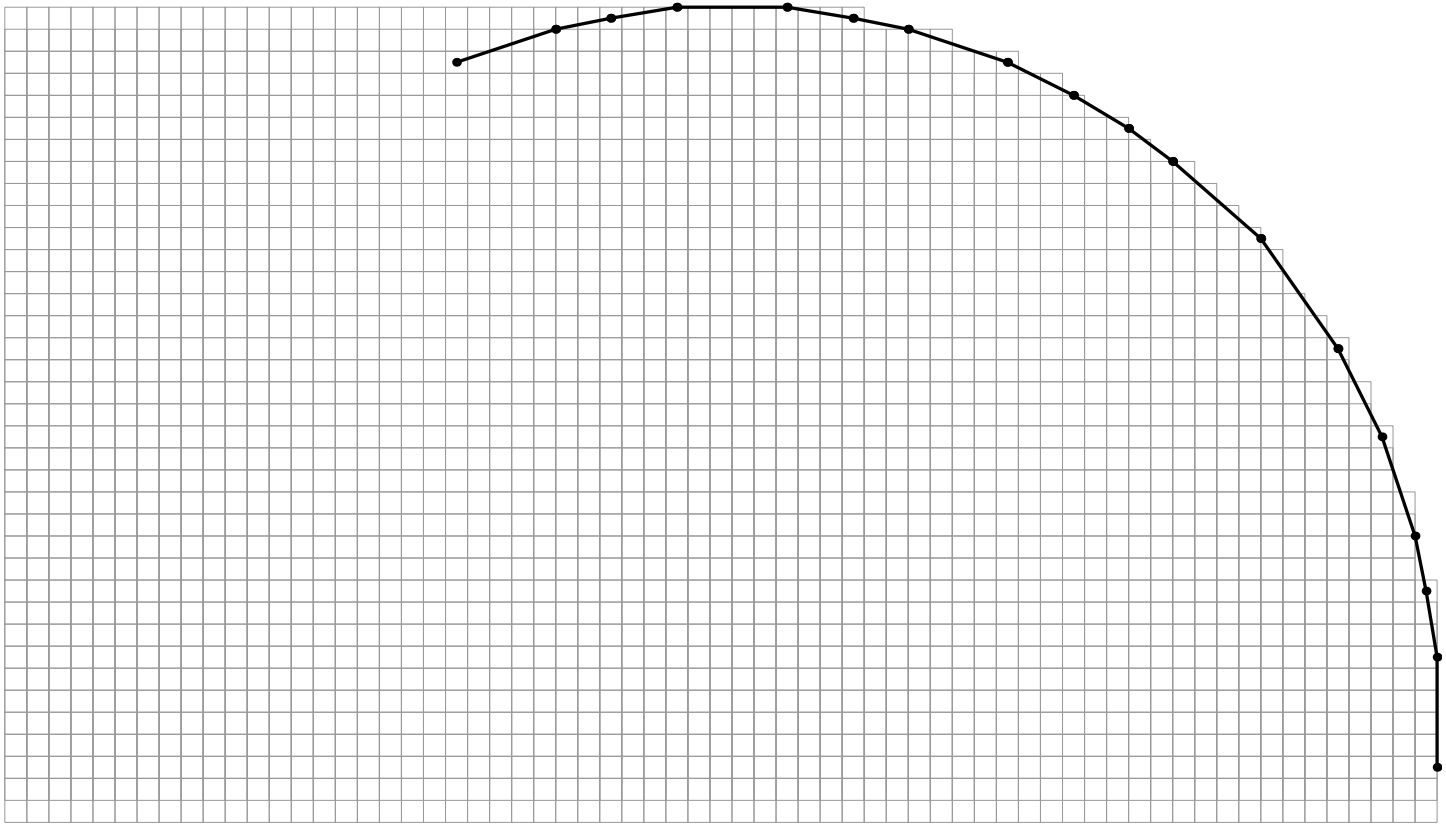
```
_manually_remove(valid_curves_e,{15 }) – can be a function
_dump_curves(final_curves,'final_curves.lua')
```

`end_program()` troppo complicata

- implementati in Lua:
  l'algoritmo di De Casteljau's $(x(t), y(t)$, left(t), right$(t))$;
  l'algoritmo di bisezione di De Casteljau (discretizzazione di una curva);
  intersezione di due cuve;

- correzioni ad-hoc: `_exec_fixes(final_curves, _sf("fix_files/fix_%04d.lua",index))`

Approssimazione poligonale della penna



Approssimazione poligonale della penna

Per produrre la poligonale associata METAFONT prende `majoraxis`, `minoraxis` e l'angolo di rotazione `theta` dalle specifiche della penna chiama la procedura `make_ellipse`.
Mettendo un sensore attorno `make_ellipse` possiamo quindi salvare i 3 parametri in una table di Lua.

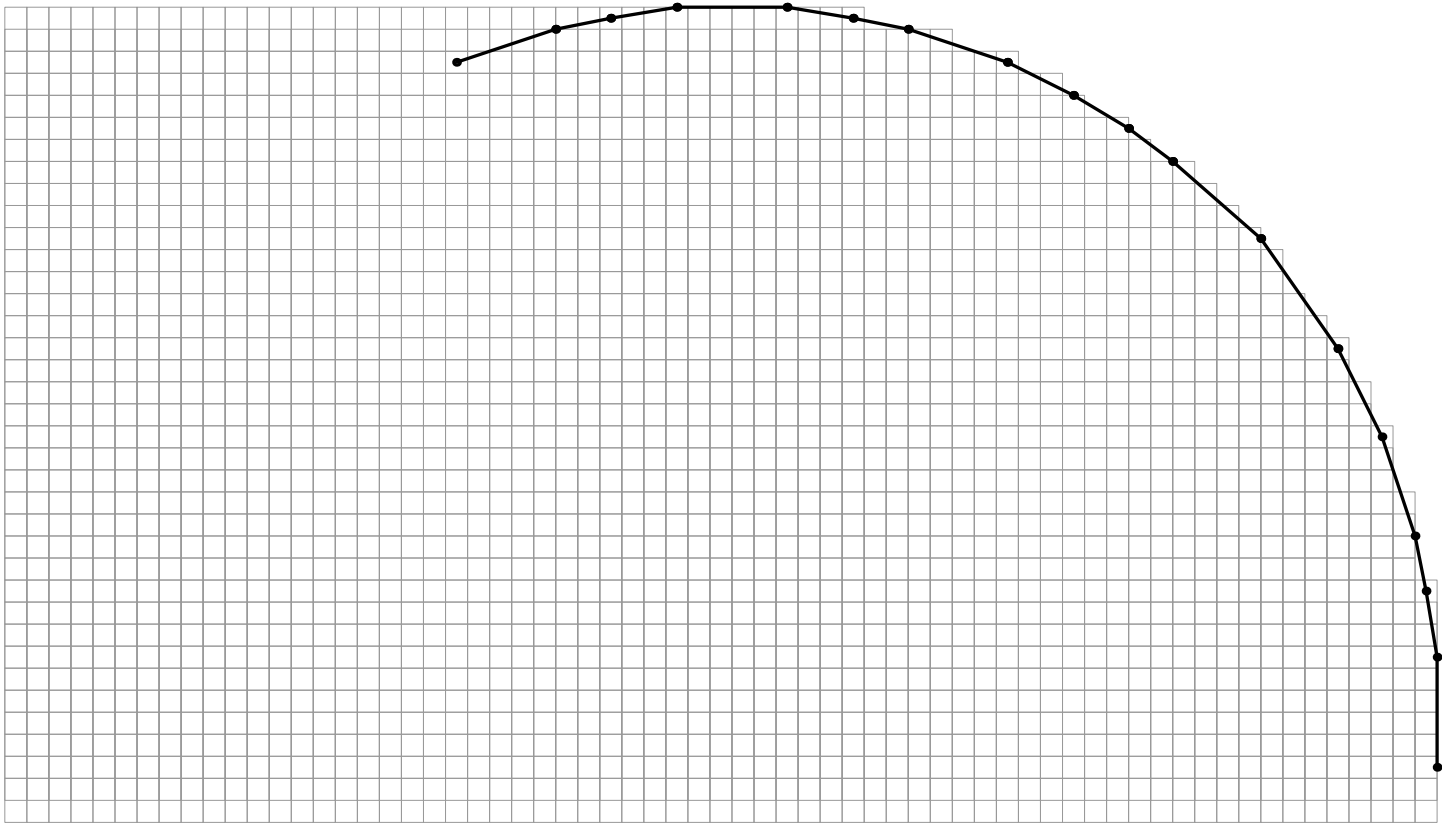
Approssimazione poligonale della penna

Trucchetto: per ogni penna eseguire MFLua sul seguente file:

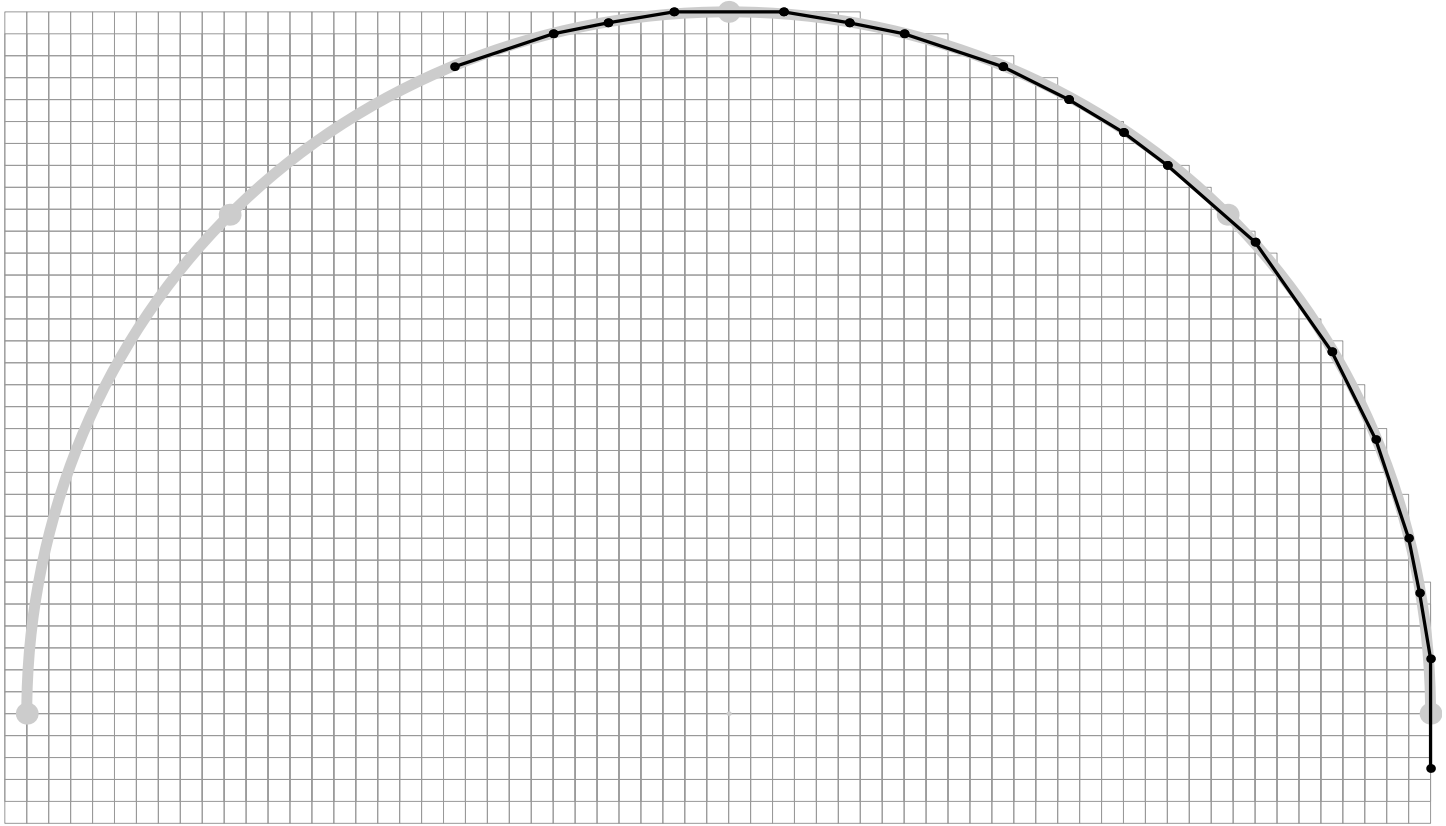
```
batchmode;
fill fullcircle
    xscaled (majoraxis) yscaled (minoraxis) rotated (theta)
shifted (0,0);
shipit;bye.
```

e memorizzare i risultati (i.e. le curve) in un apposito file Lua da leggere in un secondo momento.

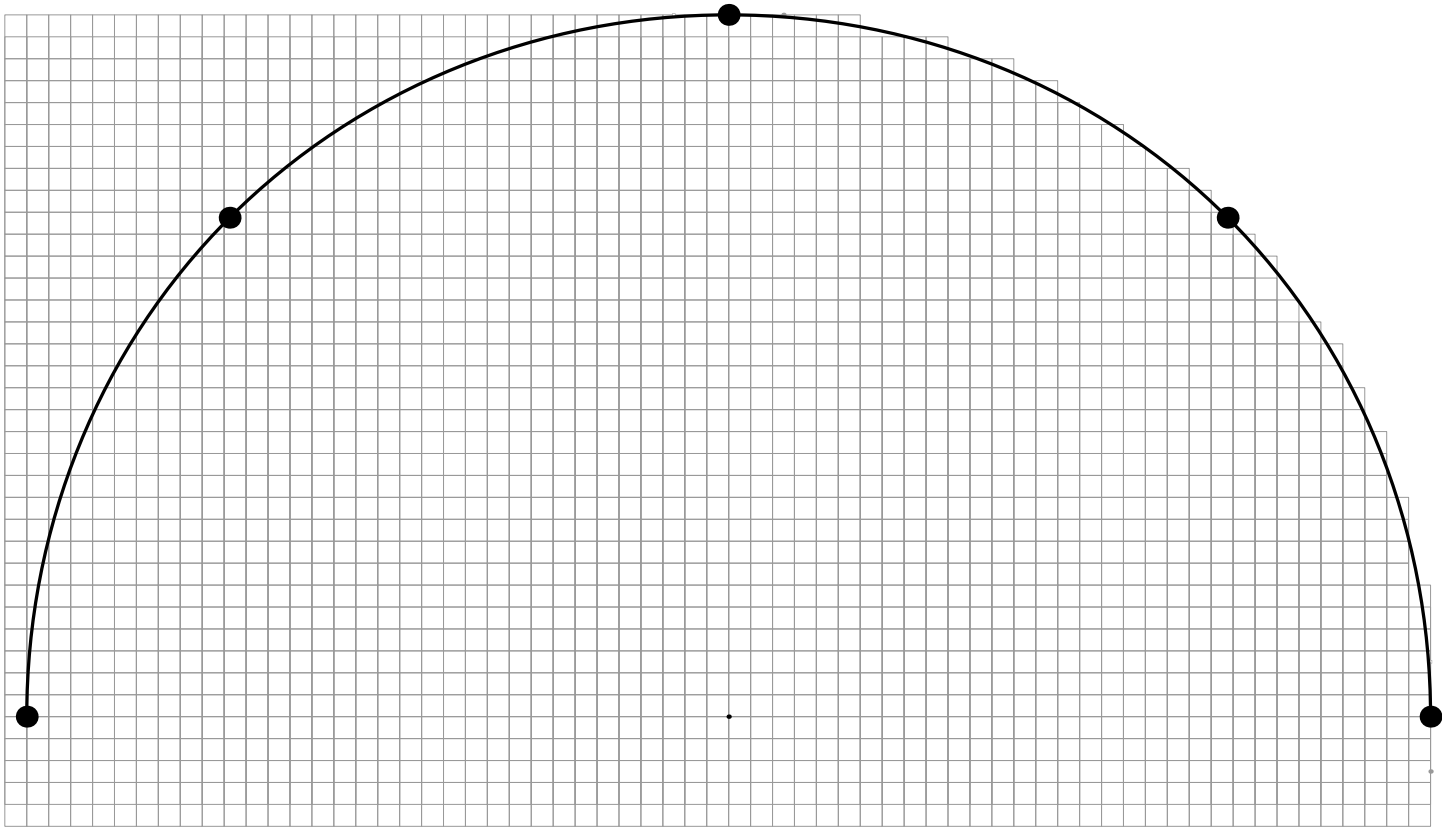
Approssimazione poligonale della penna



Approssimazione poligonale della penna



Approssimazione poligonale della penna



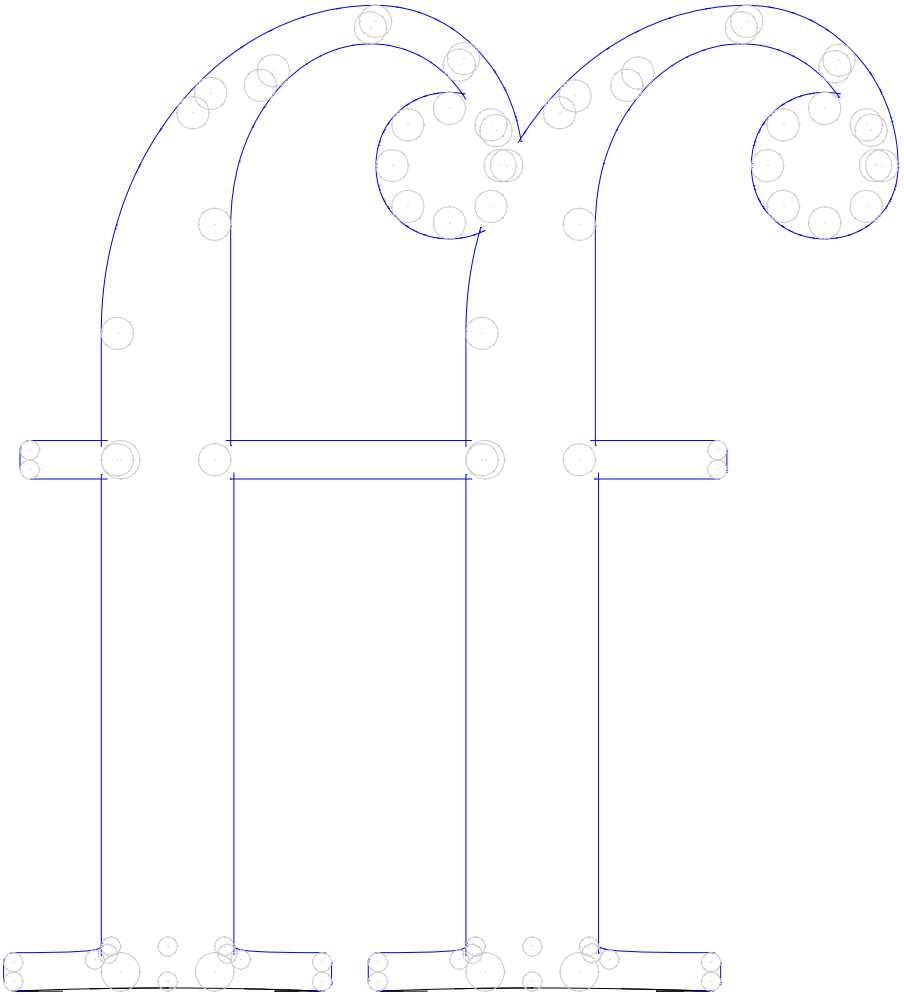
Approssimazione poligonale della penna

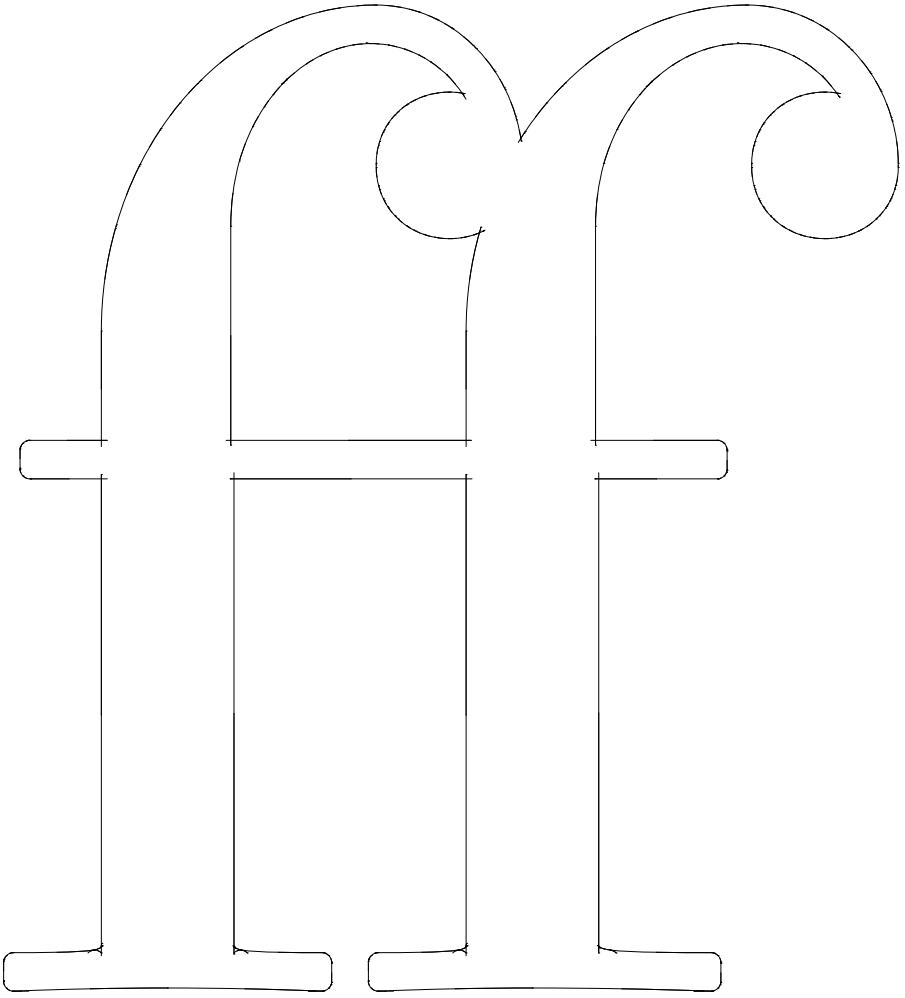
Problemi:

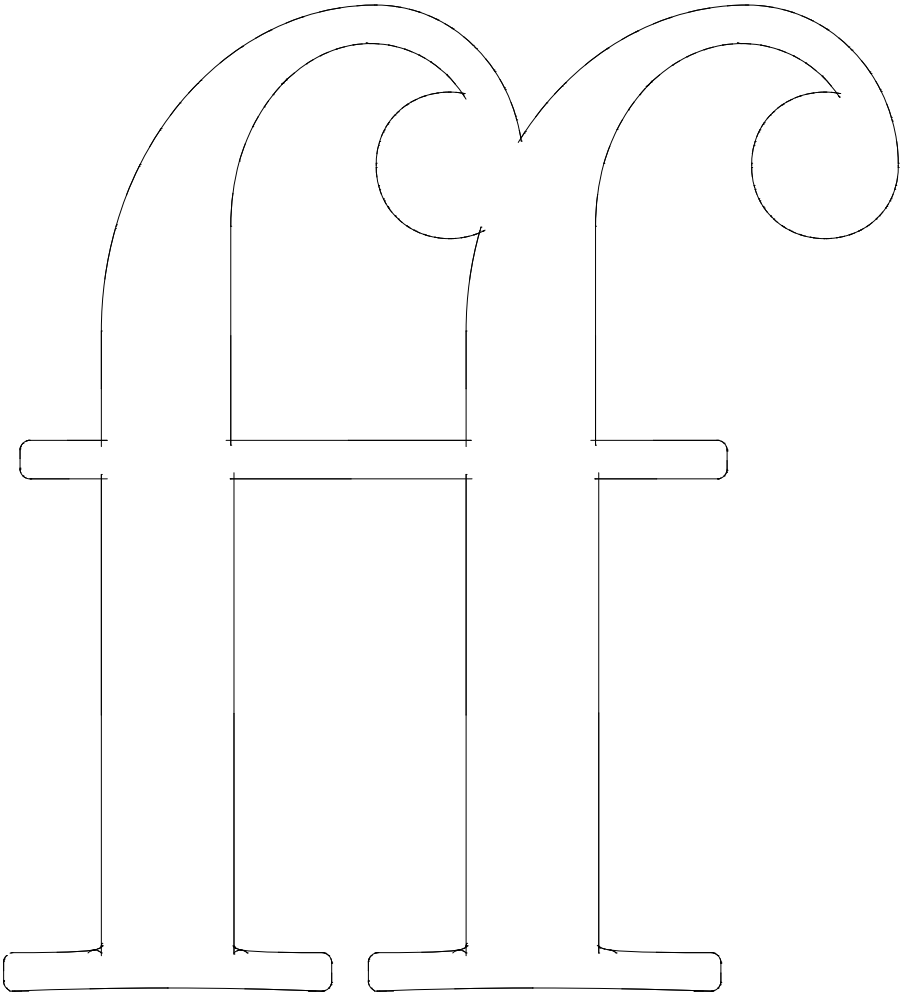
- trovare il punto esatto dove posizionare il centro dell'ellisse;

- gestire le intersezioni.

(cfr. fig. successive)



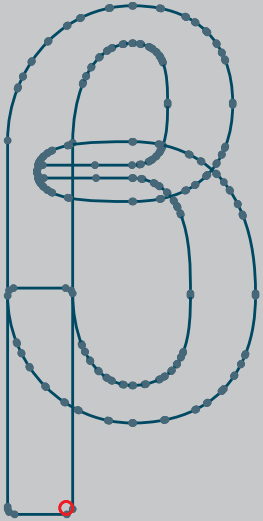




Ancora troppe curve: perché ?

Example 1 – Latin Modern sans serif bold Greek

We wanted to use Knuth's Metafont code for generating the outlines. We hoped that using the *cmssbx10* parameters would yield proper shapes.



We wanted to employ Luigi Scarso's MFLua for finding the outlines. Alas, MFLua finds "raw" outlines, i.e., the envelope created by polygonal pen's vertices; as a result, outlines have too many many nodes; their number could be perhaps automatically reduced, but we finally abandoned that approach.

Example 1 – Latin Modern sans serif bold Greek

We wanted to use Knuth's Metafont code for generating the outlines. We hoped that using the *cmssbx10* parameters would yield proper shapes.

All in all, we retrieved the basic paths (i.e., without pen stroking) from Computer Modern sources, we modified them interactively, then we "expanded strokes" (pen diameters were known from sources), and, finally, we again slightly tuned the resulting outlines manually.

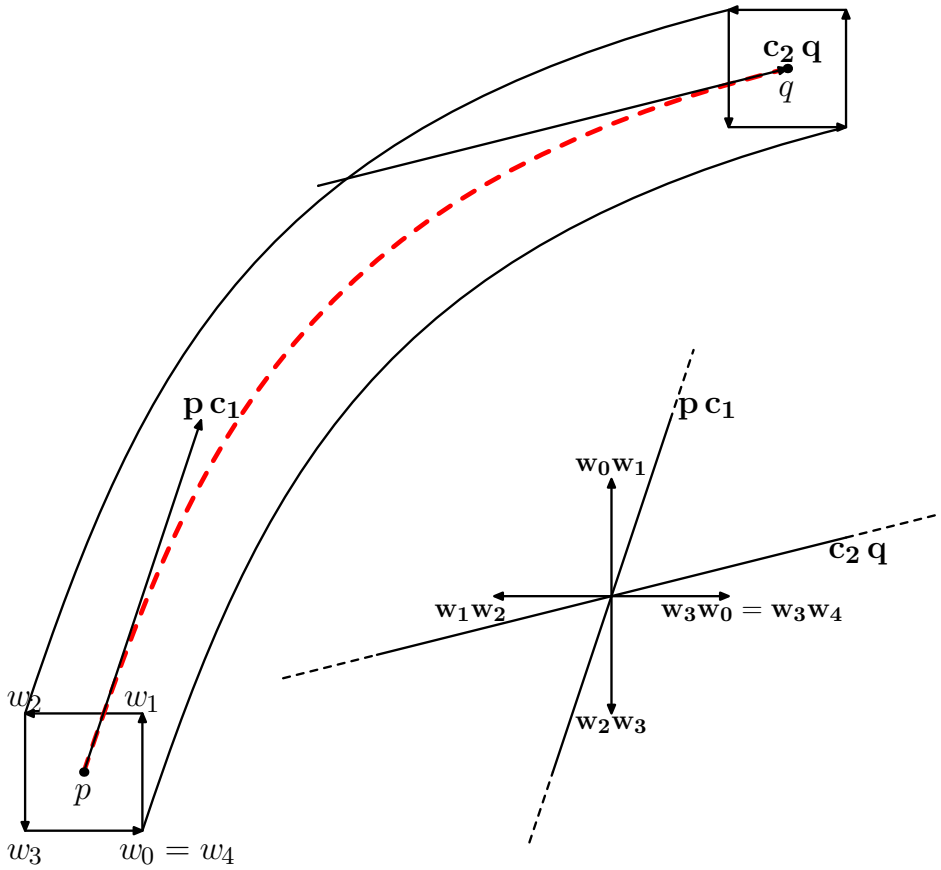
Ancora troppe curve: perché ?

È l'inviluppo della penna:

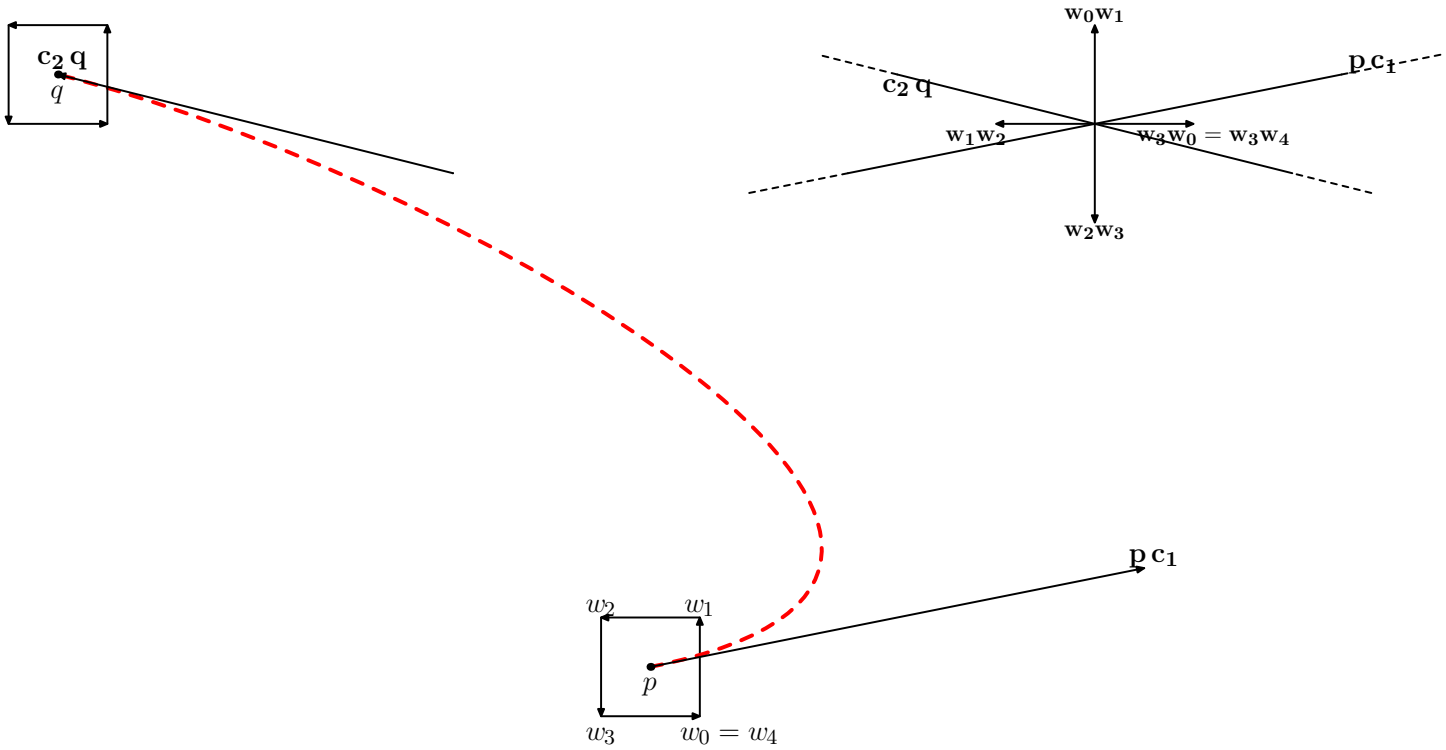
"Given a convex polygon with vertices $\mathbf{w_0}, \mathbf{w_1}, ..., \mathbf{w_{n-1}}, \mathbf{w_n} = \mathbf{w_0}$ a in counterclockwise order ...(and a curve $\mathbf{B(t)}$) the envelope is obtained if we offset $\mathbf{B(t)}$ by $\mathbf{w_k}$ when the curve is travelling in a direction $\mathbf{B'(t)}$ lying between the directions $\mathbf{w_k - w_{k-1}}$ and $\mathbf{w_{k+1} - w_k}$. At times t when the curve direction $\mathbf{B'(t)}$ increases past $\mathbf{w_{k+1} - w_k}$, we temporarily stop plotting the offset curve and we insert a straight line from $\mathbf{B(t)+w_k}$ to $\mathbf{B(t)+w_{k+1}}$; notice that this straight line is tangent to the to the offset curve. Similarly, when the curve direction decreases past $\mathbf{w_k - w_{k-1}}$, we stop plotting and insert a straight line from $\mathbf{B(t) + w_k}$ to $\mathbf{B(t) + w_{k+1}}$; the latter line is actually a "retrograde" step which will not be part of the final envelope under the METAFONT's assumptions. The result of this construction is a continuous path that consist of alternating curves and straight line segments."

METAFONT: The Program, chapter 469.

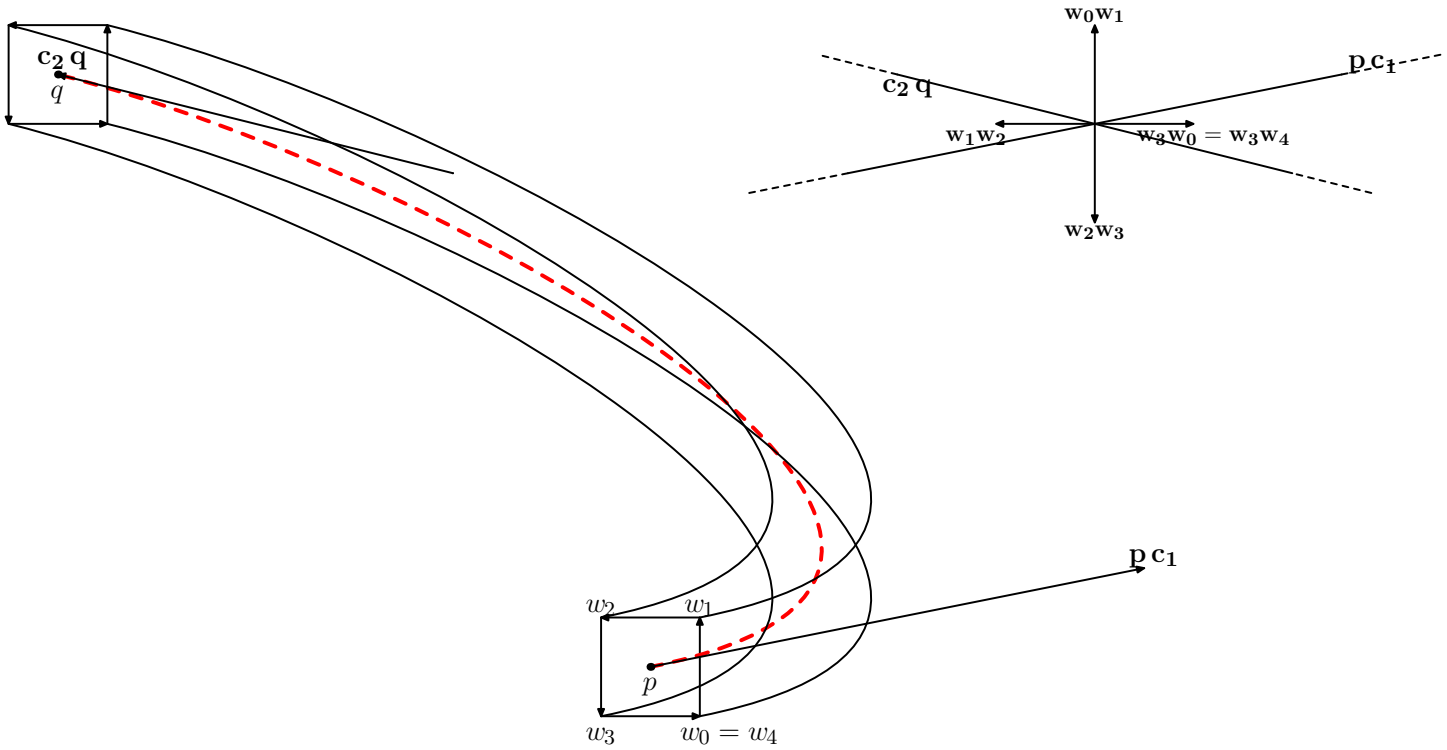
Inviluppo di una penna poligonale



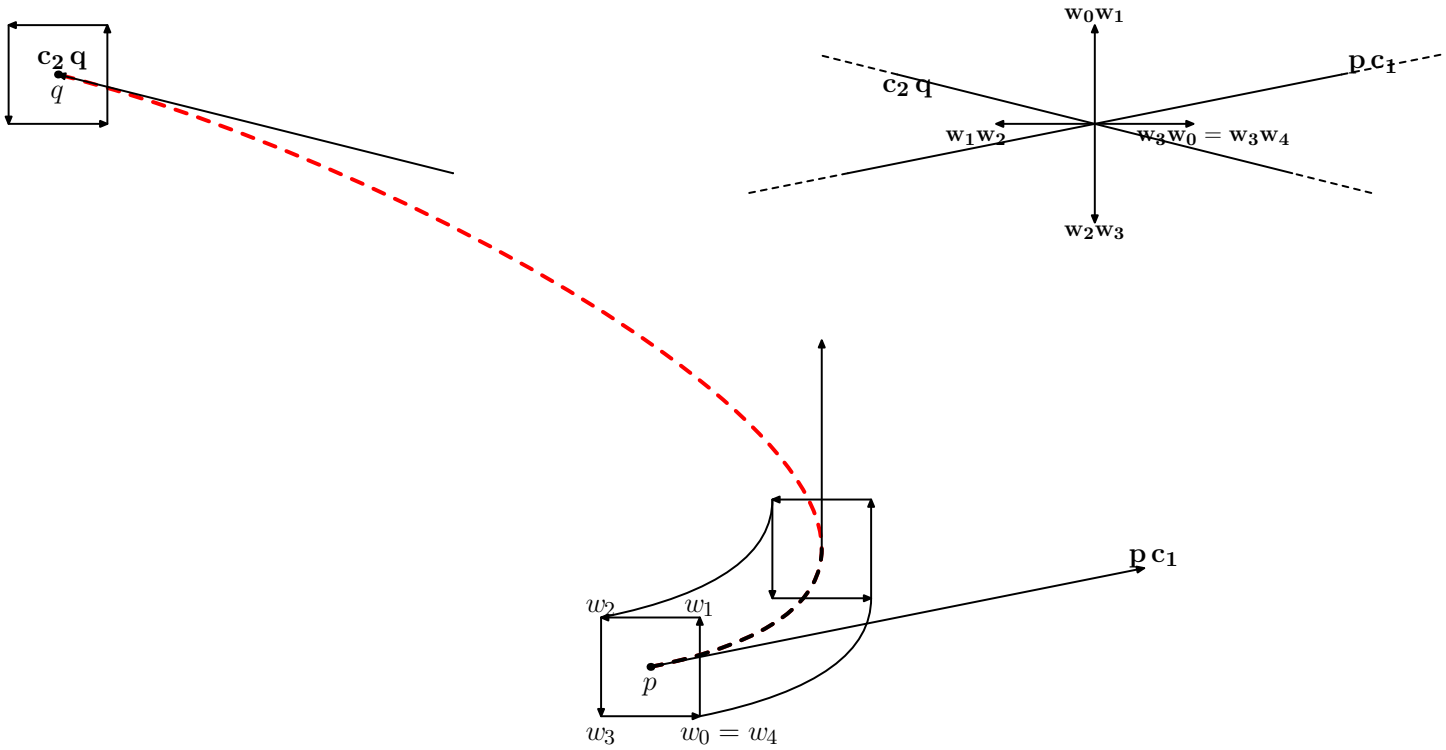
Inviluppo di una penna poligonale



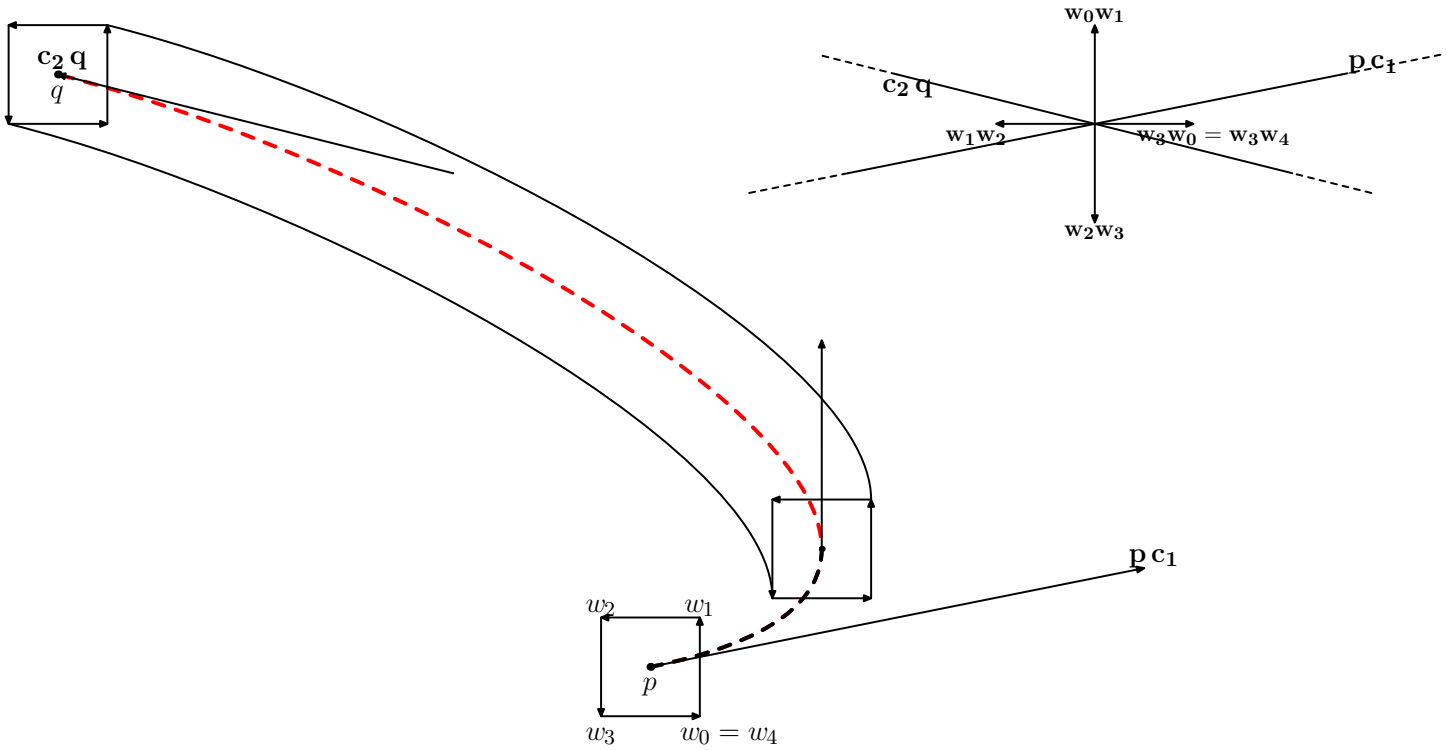
Inviluppo di una penna poligonale



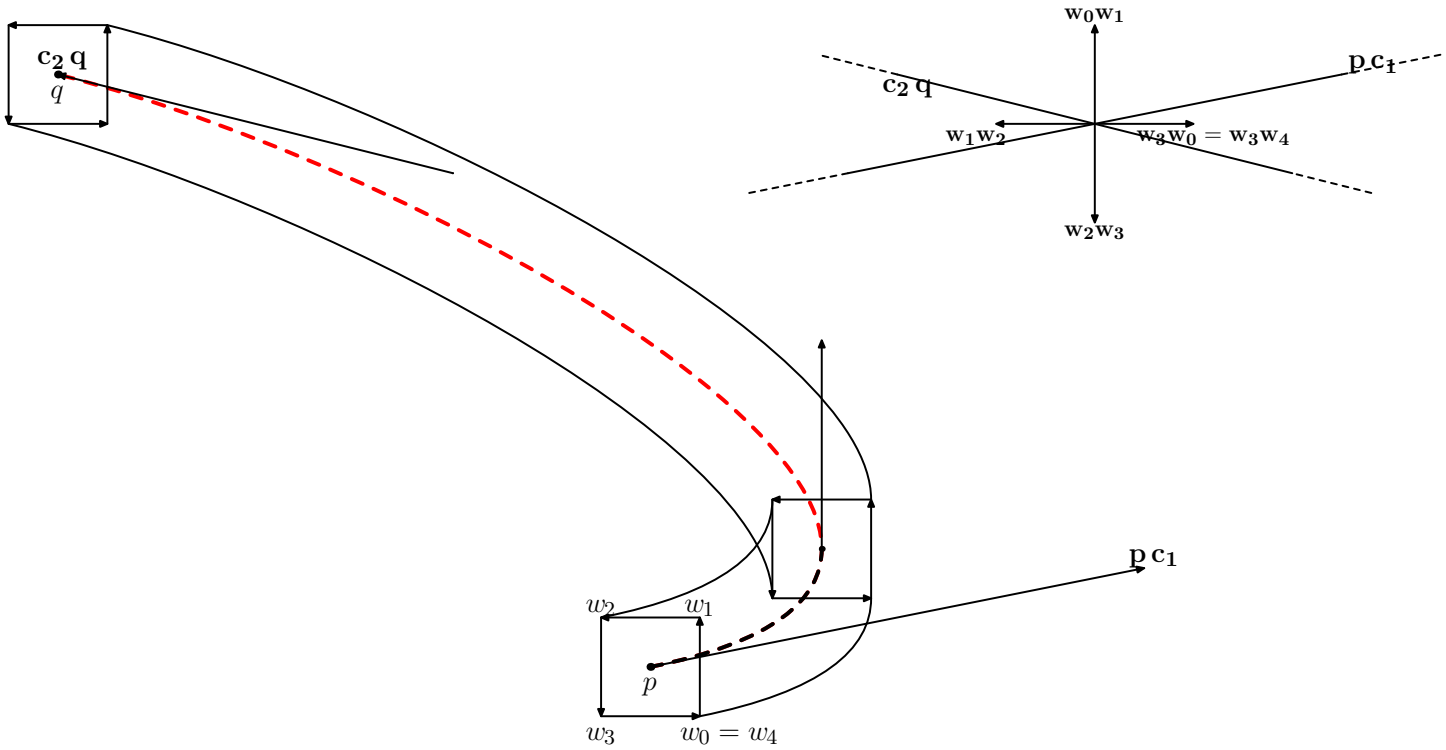
Inviluppo di una penna poligonale



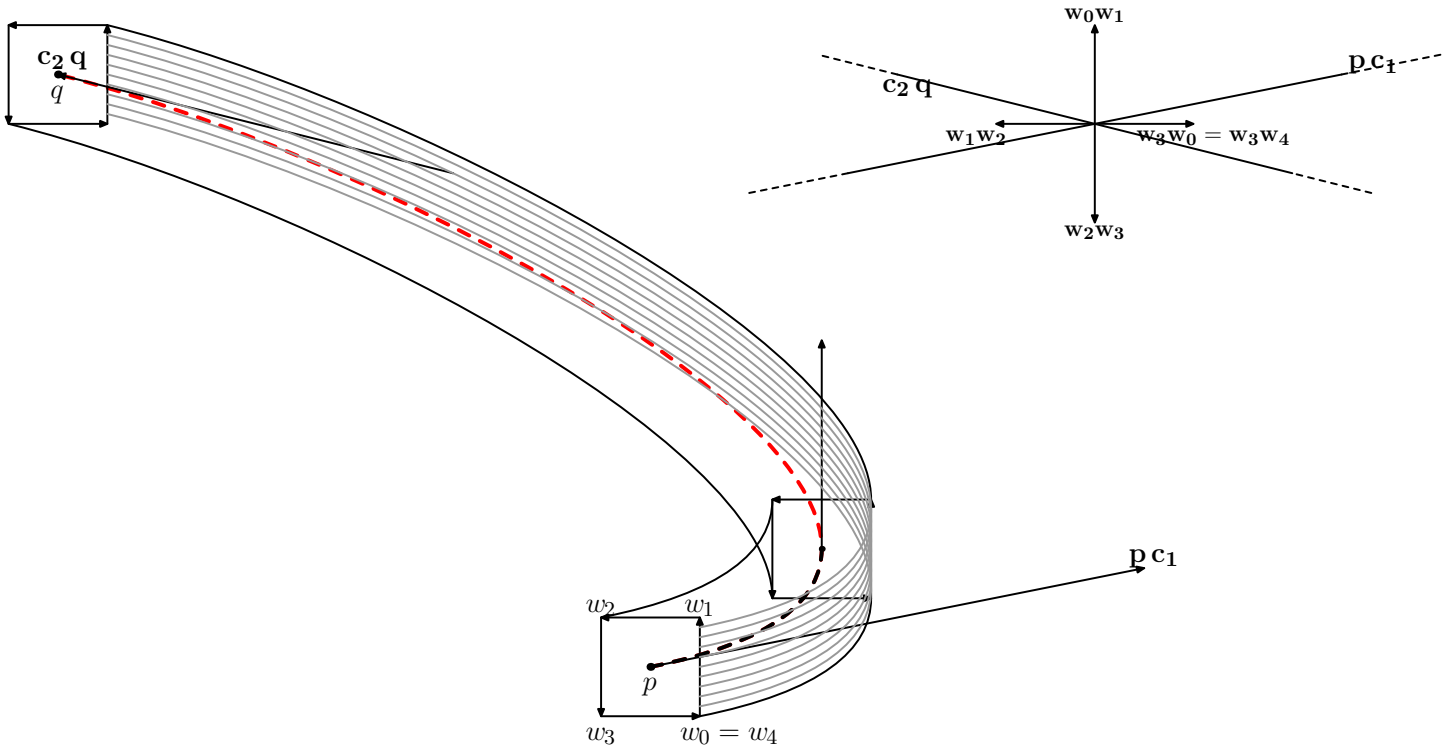
Inviluppo di una penna poligonale



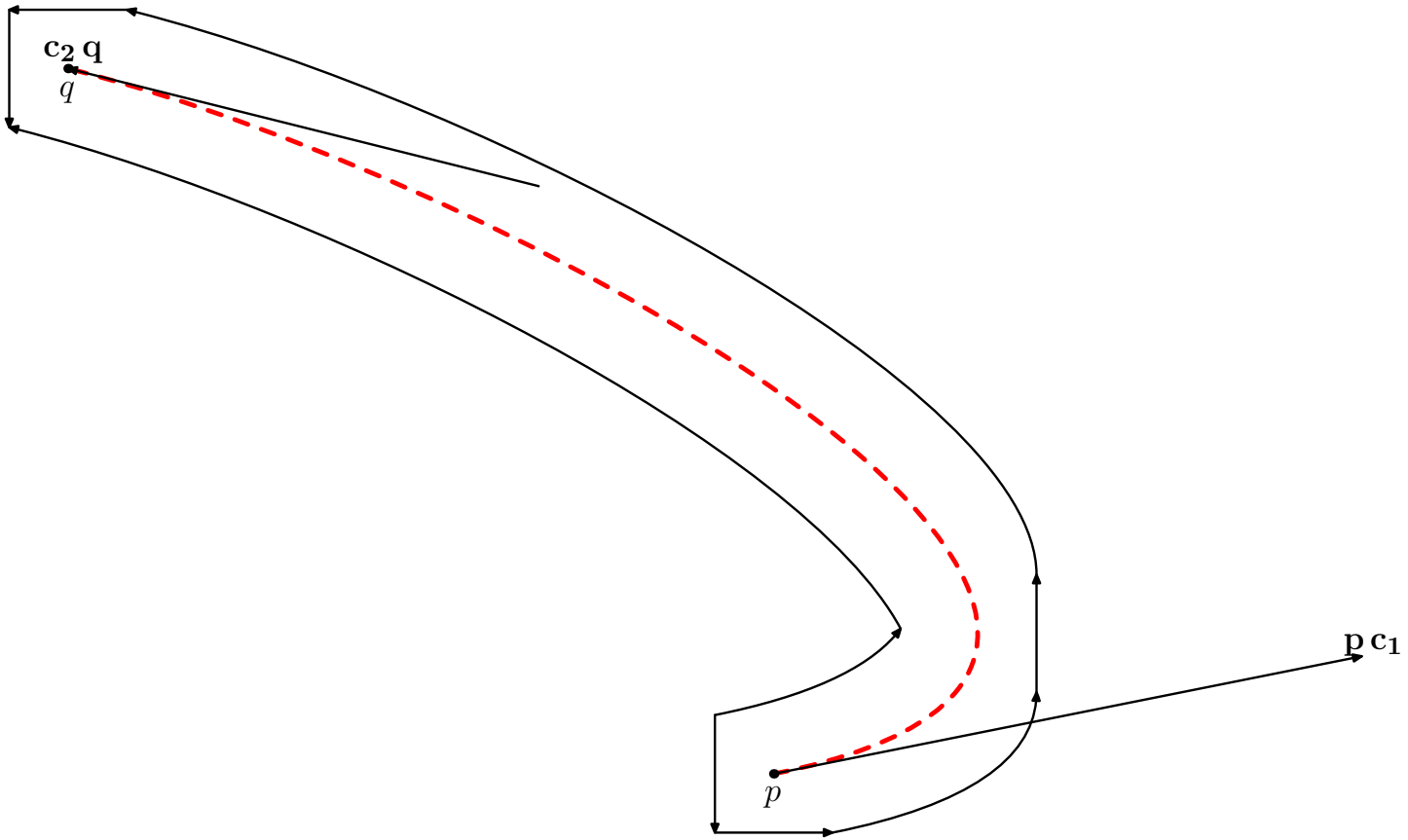
Inviluppo di una penna poligonale



Inviluppo di una penna poligonale



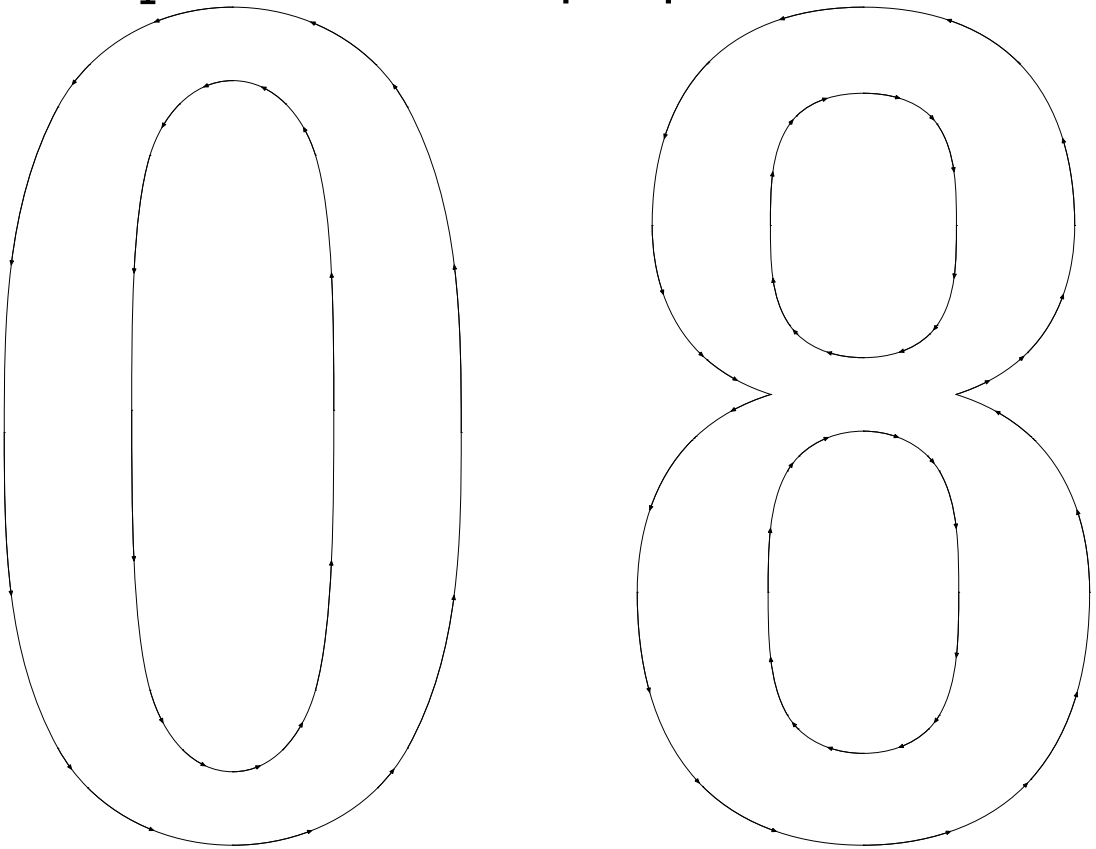
Inviluppo di una penna poligonale



Inviluppo di una penna poligonale

Ancora troppe curve: perché ?

Non è un problema con `penpos`:



Conclusione

- le penne implicano un alto numero di curve e complicano il post-processing;
- se un glifo ha solo contorni il post-processing è più semplice e veloce;
- FontForge (o equivalenti) è indispensabile (come programma o modulo);
- attualmente il lavoro è concentrato `https://github.com/luigiScarso/mflua` su `/tree/master/test/xmssdc10`
  solo per produrre un buon set di outlines (i.e. no OpenType)

Possibili future direzioni

- semplificazione automatica delle curve prodotte dalle penne ;
- calcolo automatico dell'effettivo inviluppo;
- GUI ala FontForge ?
- `libmetafont` in C++ ?
- ...

That's all
Thank you !