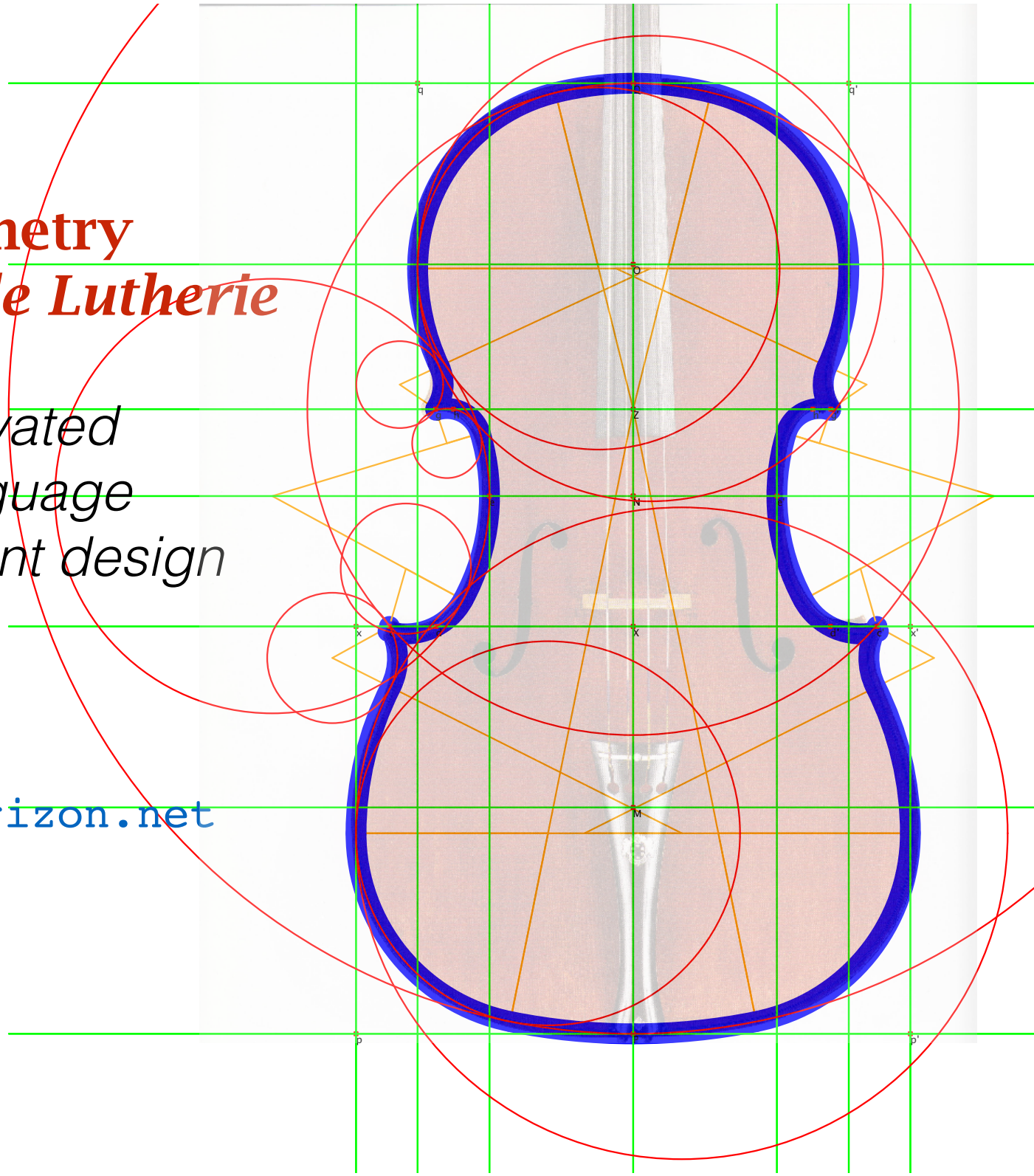
Functional Geometry and the *Traité de Lutherie*

a historically-motivated programming language for string instrument design

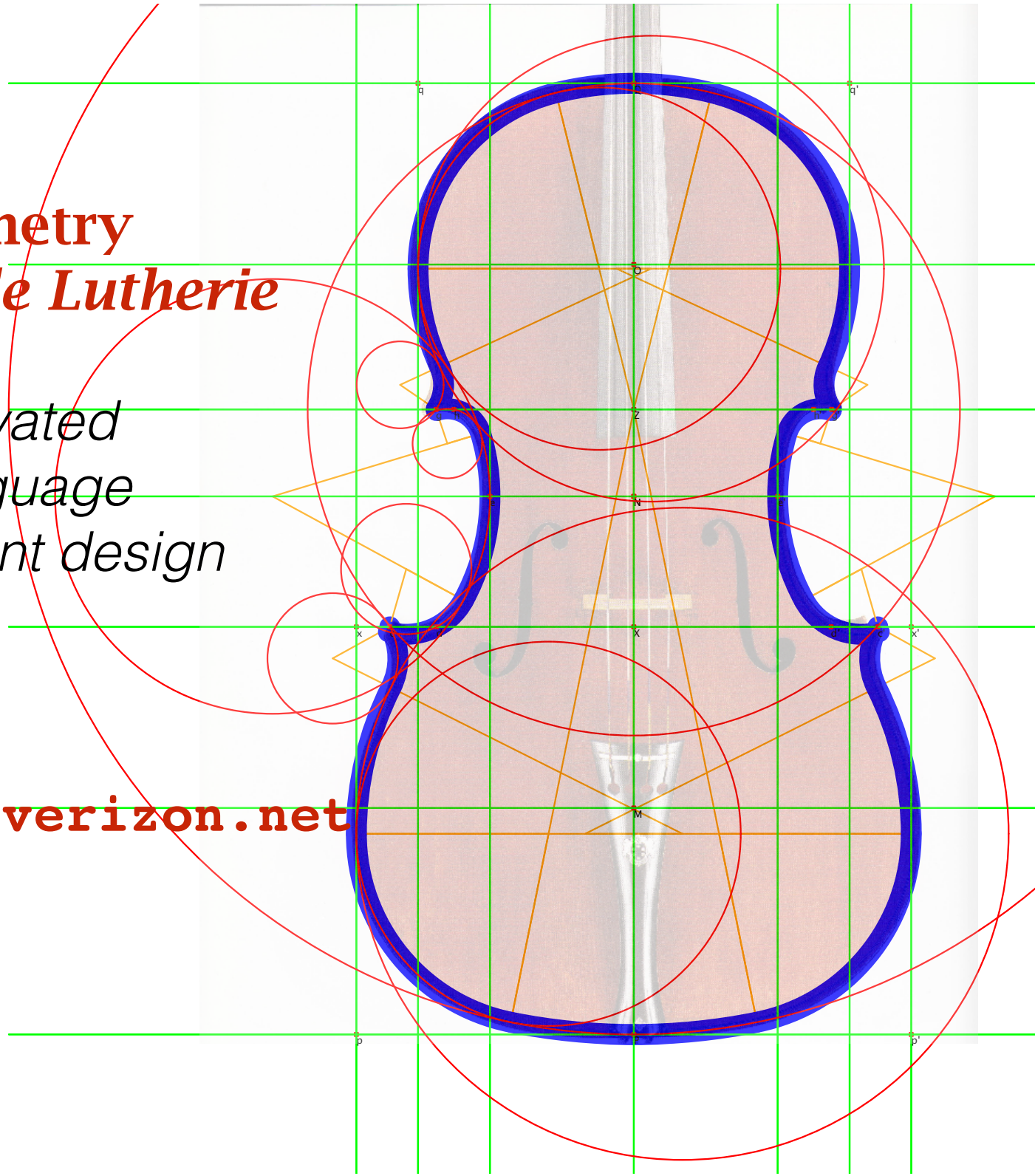


Harry Mairson
harry.mairson@verizon.net

NATIONAL ENDOWMENT FOR THE HUMANITIES
NATIONAL SCIENCE FOUNDATION

Functional Geometry and the *Traité de Lutherie*

a historically-motivated programming language for string instrument design



Harry Mairson

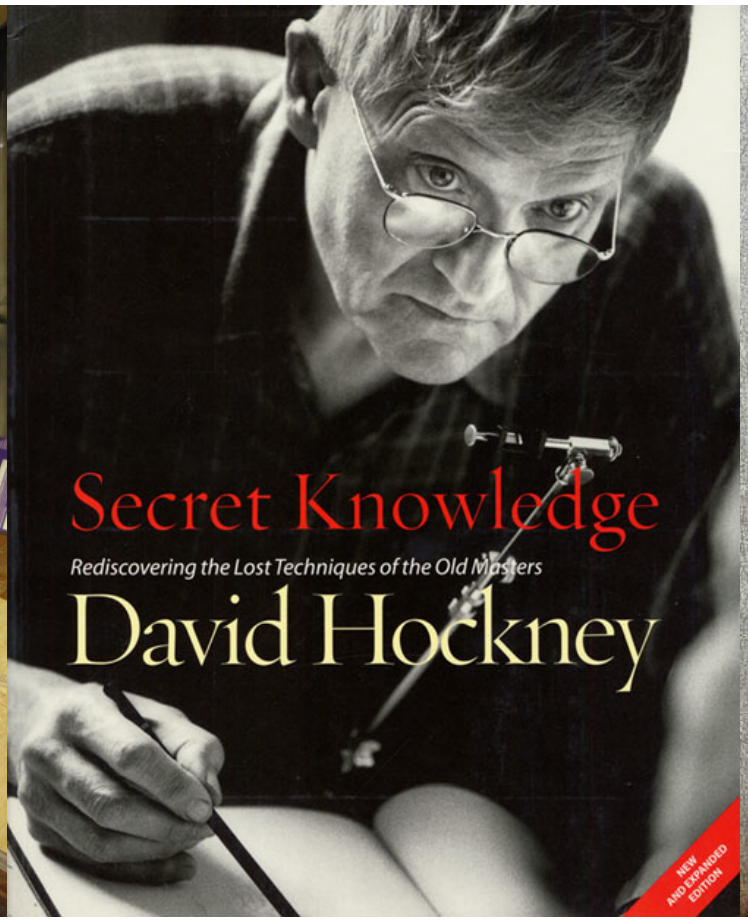
harry.mairson@verizon.net

NATIONAL ENDOWMENT FOR THE HUMANITIES

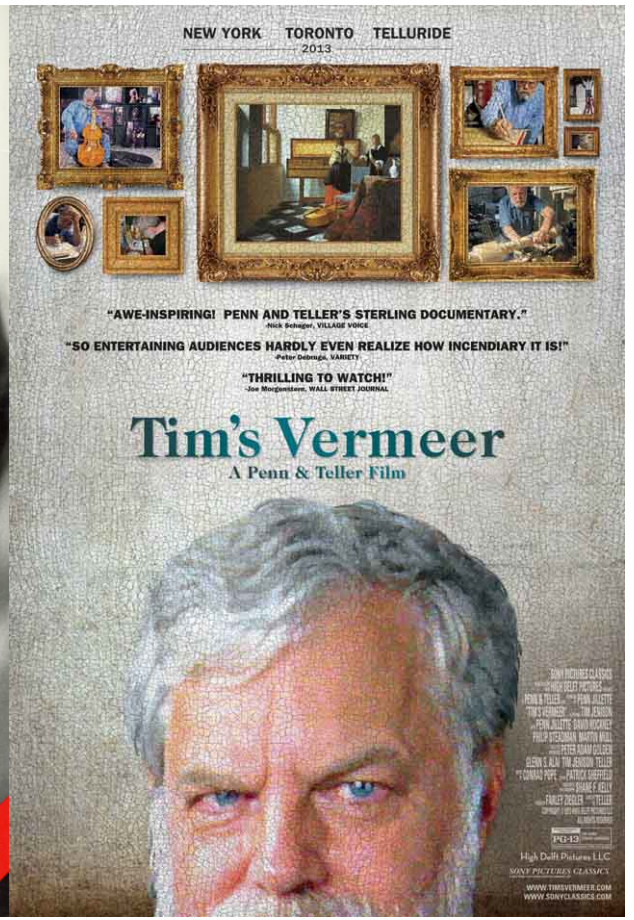
NATIONAL SCIENCE FOUNDATION

I'm an amateur luthier and a computer science professor…

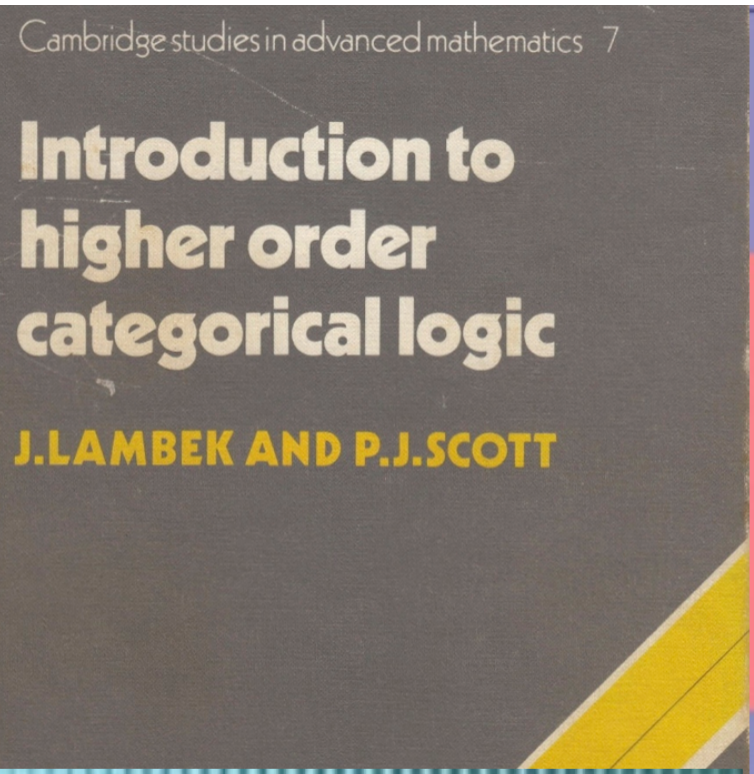
Computational thinking:
"There's a method for everything…"
(that is, an algorithm)

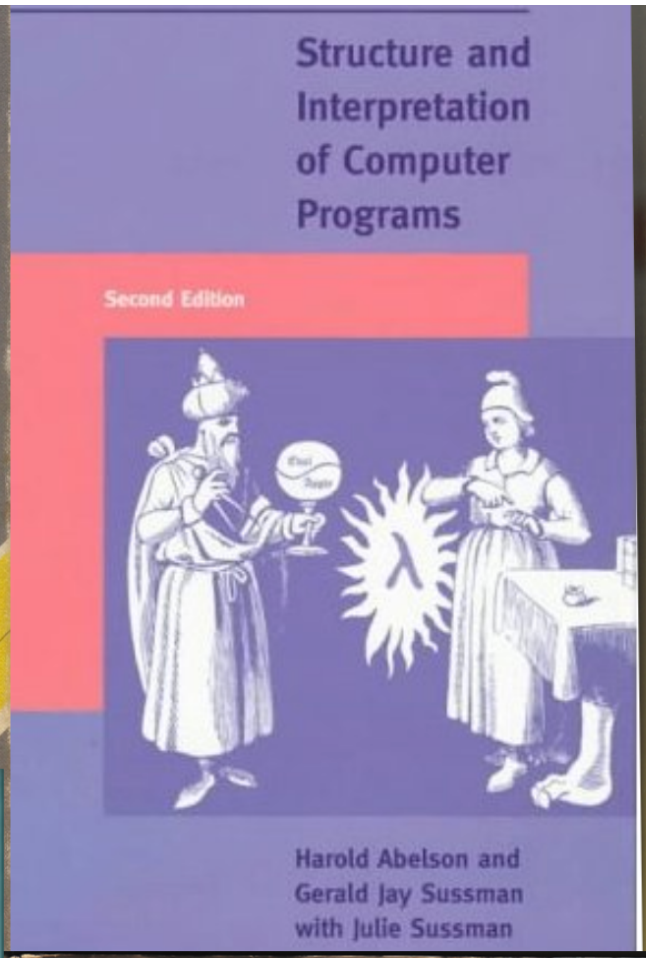


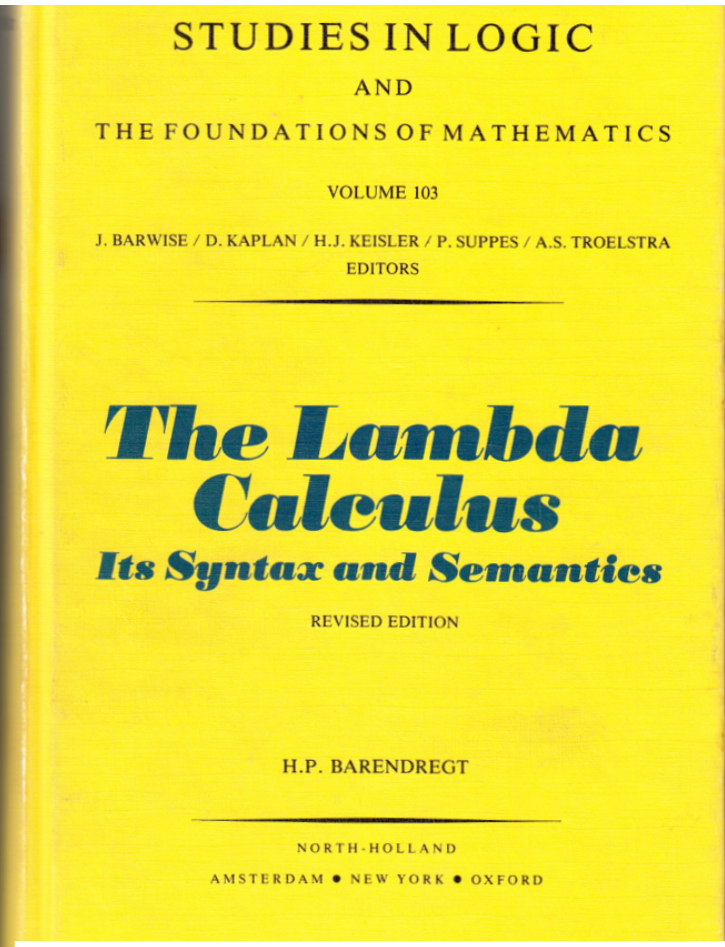
artist



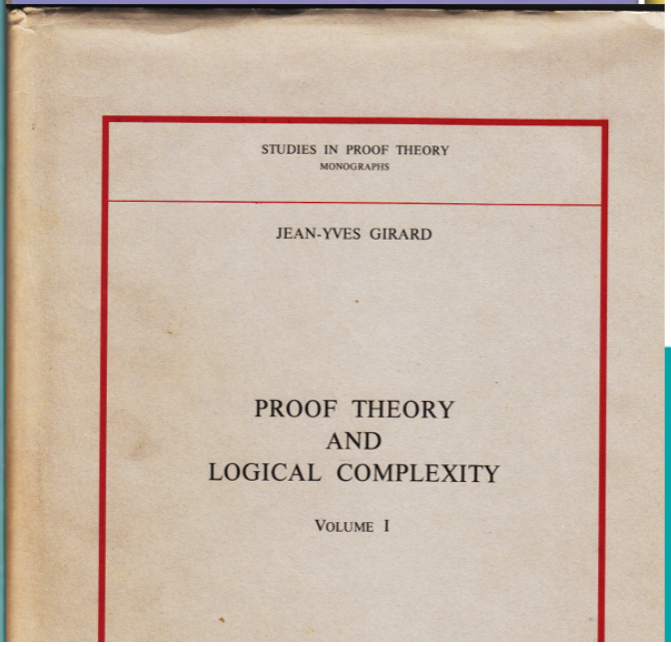
Tim Jenison,
graphics software engineer

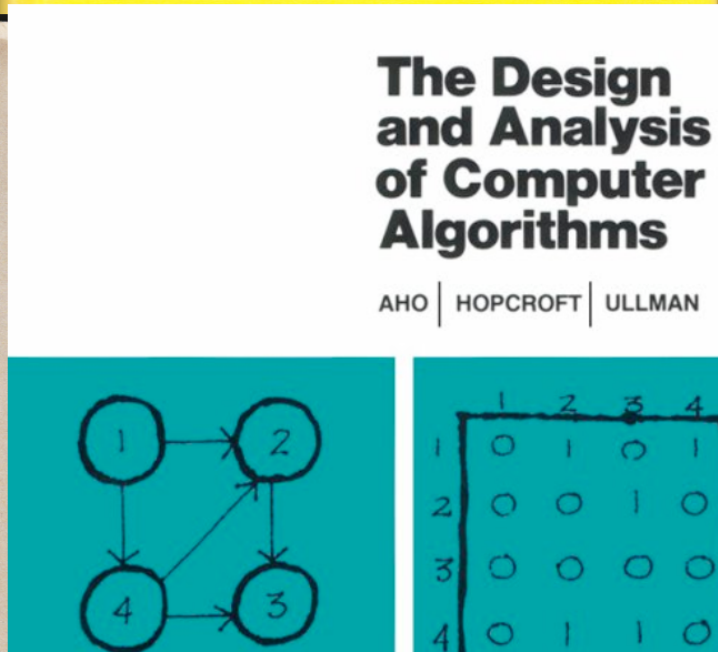

Cambridge studies in advanced mathematics 7
Introduction to higher order categorical logic
J.LAMBEK AND P.J.SCOTT

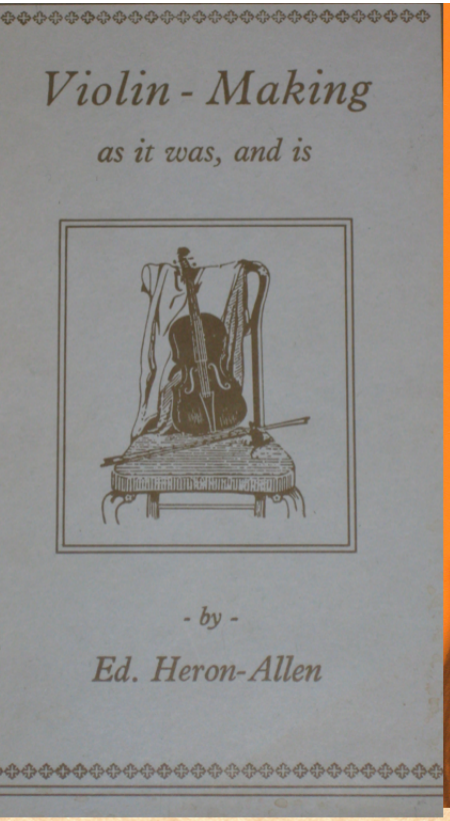

Structure and Interpretation of Computer Programs
Second Edition
Harold Abelson and Gerald Jay Sussman with Julie Sussman

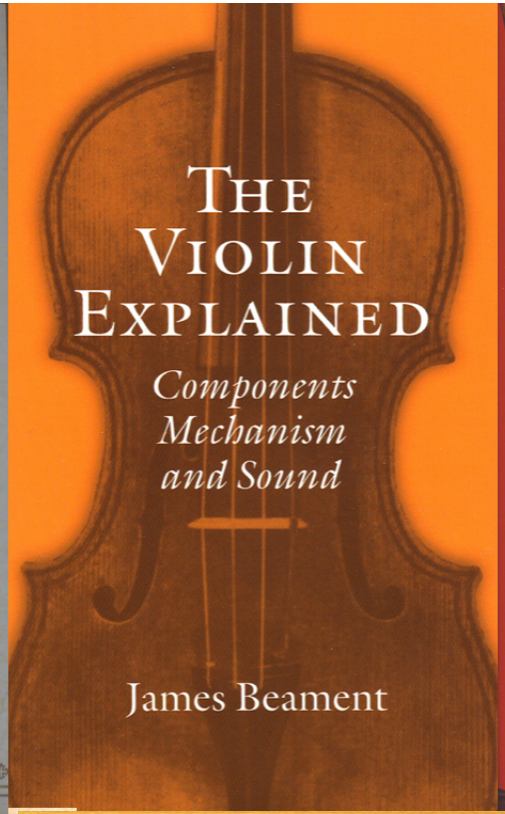

STUDIES IN LOGIC
AND
THE FOUNDATIONS OF MATHEMATICS
VOLUME 103
J. BARWISE / D. KAPLAN / H.J. KEISLER / P. SUPPES / A.S. TROELSTRA
EDITORS
The Lambda Calculus
Its Syntax and Semantics
REVISED EDITION
H.P. BARENDREGT
NORTH-HOLLAND
AMSTERDAM • NEW YORK • OXFORD

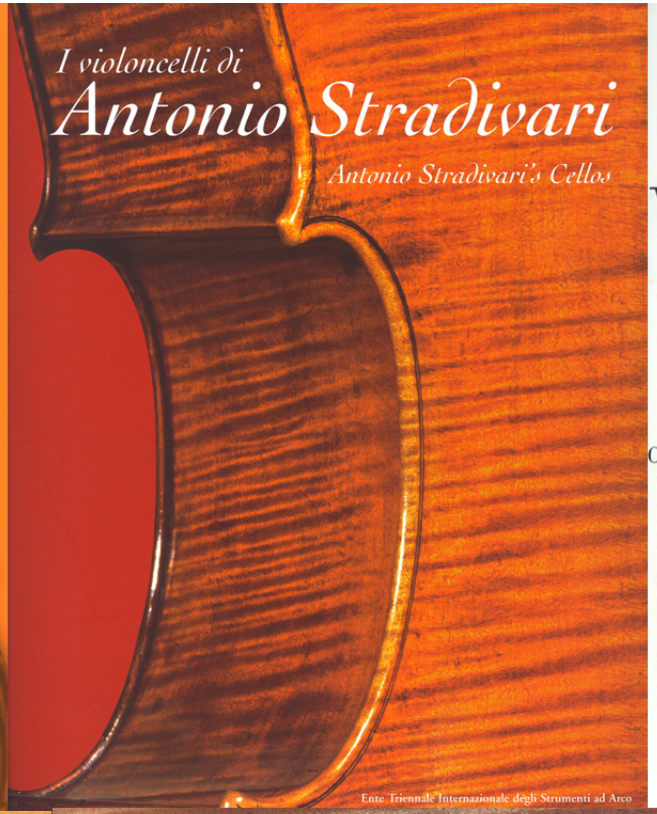

Proofs and Types
Jean-Yves Girard
Yves Lafont
Paul Taylor

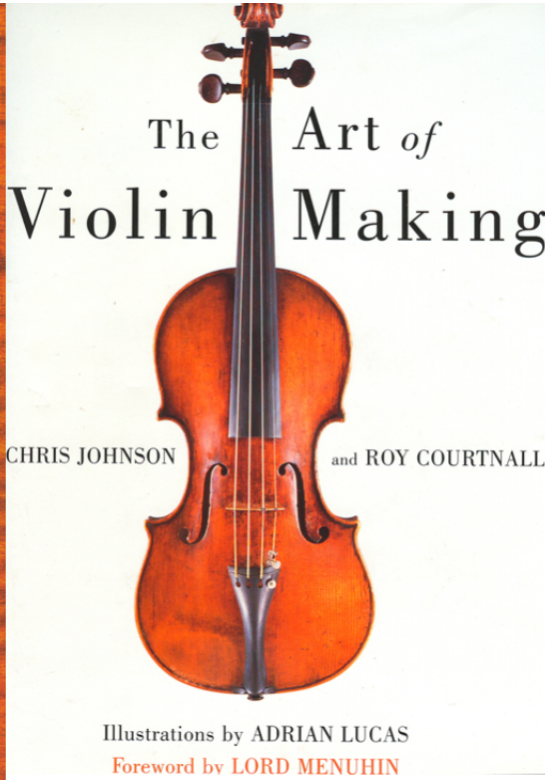

STUDIES IN PROOF THEORY
MONOGRAPHS
JEAN-YVES GIRARD
PROOF THEORY
AND
LOGICAL COMPLEXITY
VOLUME I

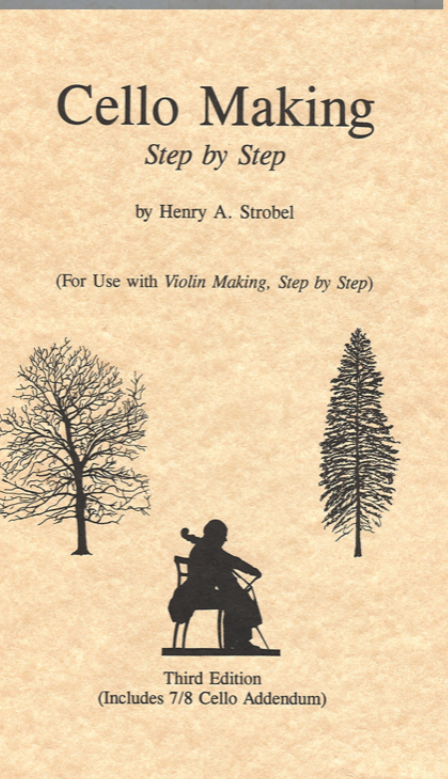

The Design and Analysis of Computer Algorithms
AHO | HOPCROFT | ULLMAN
1
2
4
3
1 2 3 4
1 0 1 0 1
2 0 0 1 0
3 0 0 0 0
4 0 1 1 0

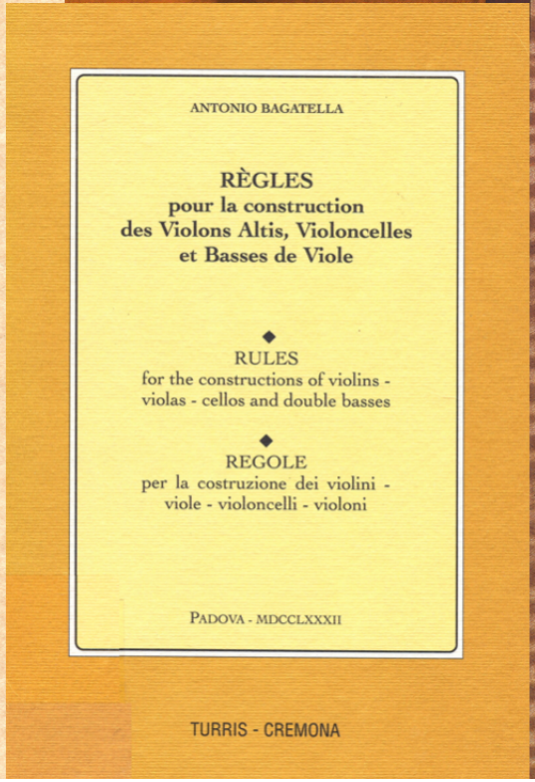

Violin - Making
as it was, and is
- by -
Ed. Heron-Allen

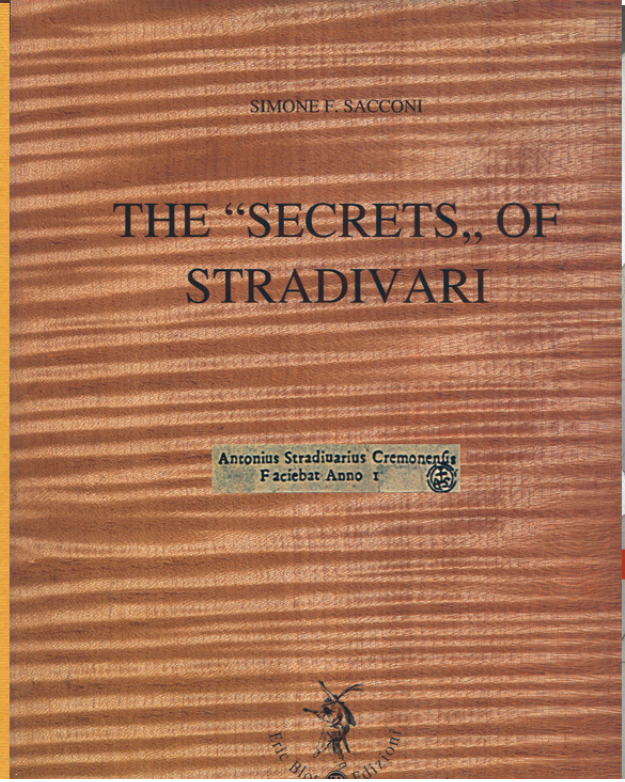

THE VIOLIN EXPLAINED
Components
Mechanism
and Sound
James Beament

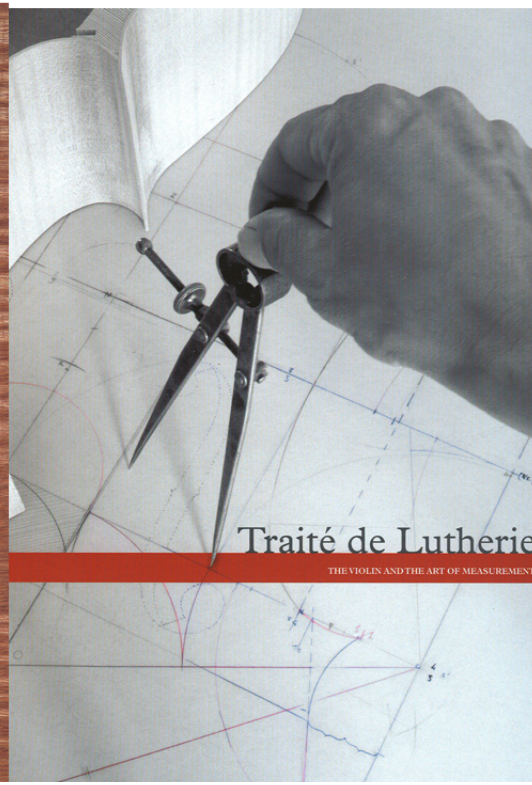

I violoncelli di
Antonio Stradivari
Antonio Stradivari's Cellos
Ente Triennale Internazionale degli Strumenti ad Arco

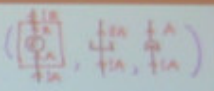

The Art of
Violin Making
CHRIS JOHNSON
and ROY COURTNALL
Illustrations by ADRIAN LUCAS
Foreword by LORD MENUHIN


Cello Making
Step by Step
by Henry A. Strobel
(For Use with Violin Making, Step by Step)
Third Edition
(Includes 7/8 Cello Addendum)


ANTONIO BAGATELLA
RÈGLES
pour la construction
des Violons Altis, Violoncelles
et Basses de Viole
RULES
for the constructions of violins -
violas - cellos and double basses
REGOLE
per la costruzione dei violini -
viole - violoncelli - violoni
PADOVA - MDCCLXXXII
TURRIS - CREMONA

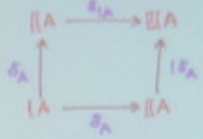

SIMONE F. SACCONI
THE "SECRETS" OF
STRADIVARI
Antonius Stradiuarius Cremonenfis
Faciebat Anno 1

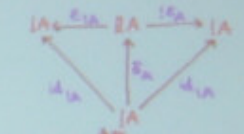

Traité de Lutherie
THE VIOLIN AND THE ART OF MEASUREMENT

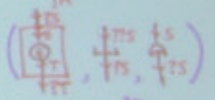
Sharing! The comonad $(!, \delta, \varepsilon) =$ 

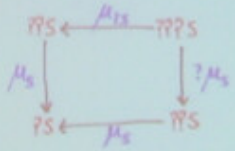
$!: \mathcal{B} \to \mathcal{B}$ a covariant functor

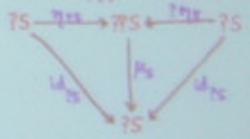
$\delta: ! \Rightarrow !!$
$\varepsilon: ! \Rightarrow 1$
} natural transformations (polymorphic functions)





Unsharing? The monad $(?, \mu, \eta) =$ 





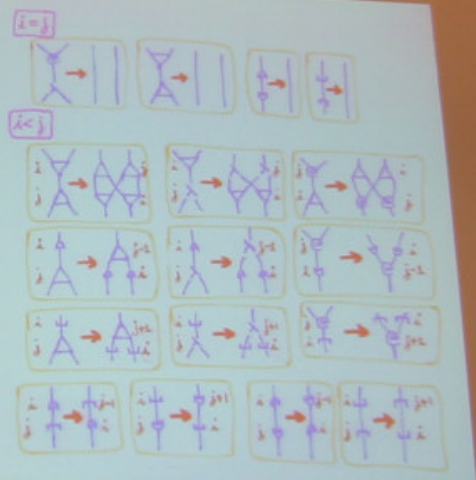
... but what's this got to do with GRAPHS? ...

Optimal reduction

- semantics preserving (what semantics?)
- maximal sharing of β-redexes
- local rules only (clones handled incrementally)

RECALL ... that the index of a node is the number of boxes drawn around it

$\phi -1$ $\psi +1$



Disclaimer:

Some things in today's presentation I…

- Know a lot about
- Have learned a lot about more recently, while climbing the lutherie learning curve
- Sort of know about
- Am unsure about, but am still talking

So bear with me…

Outline:

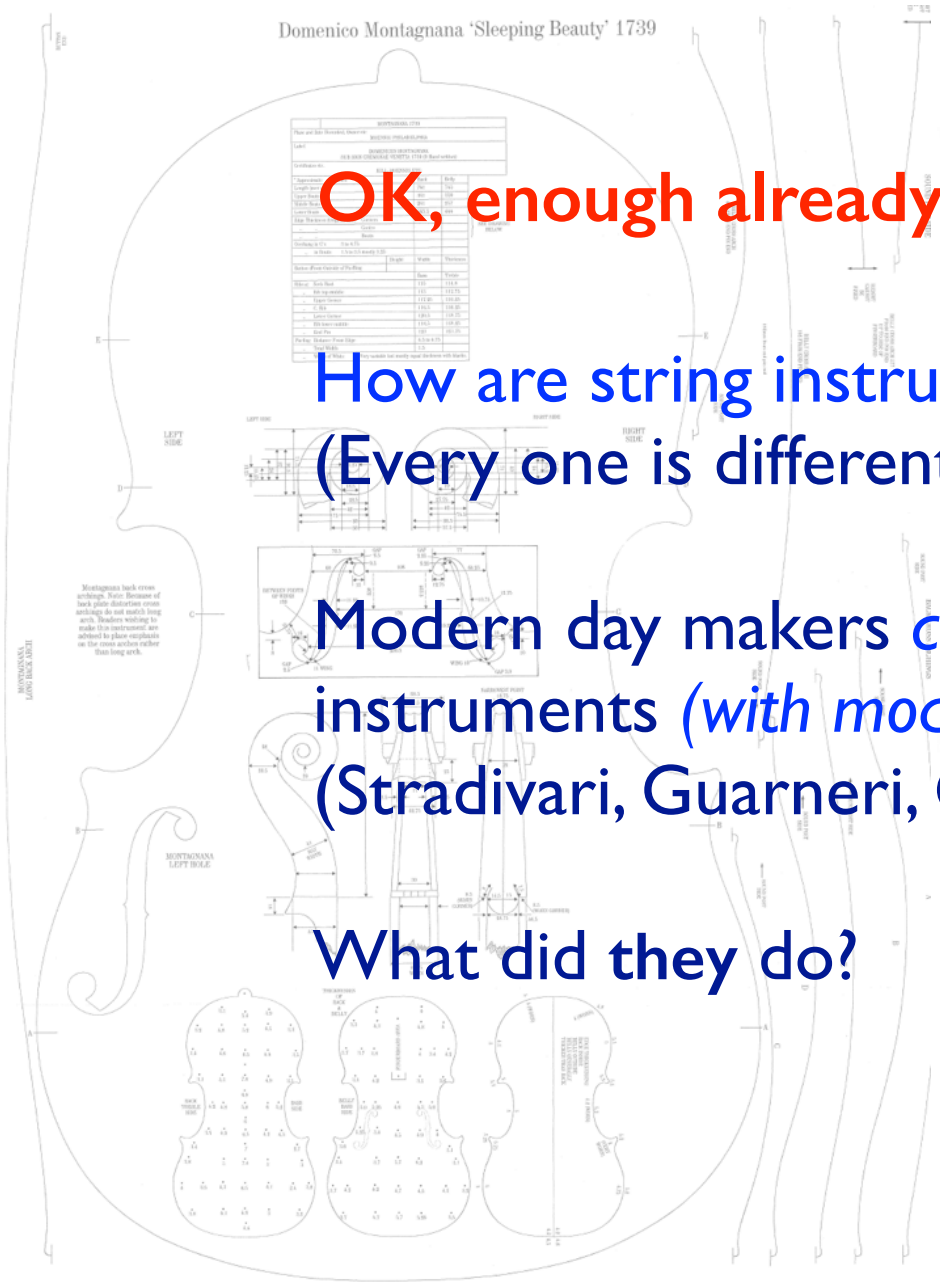
What I'm going to talk about today…

- Some introductory background…

- The software artifact that I've built…
  a programming language for
  François Denis's *Traité de Lutherie.*

- What you could use this software for

- Using this software to facilitate understanding
  better the design of string instruments—-
  "computational art history"
  (speculative…).

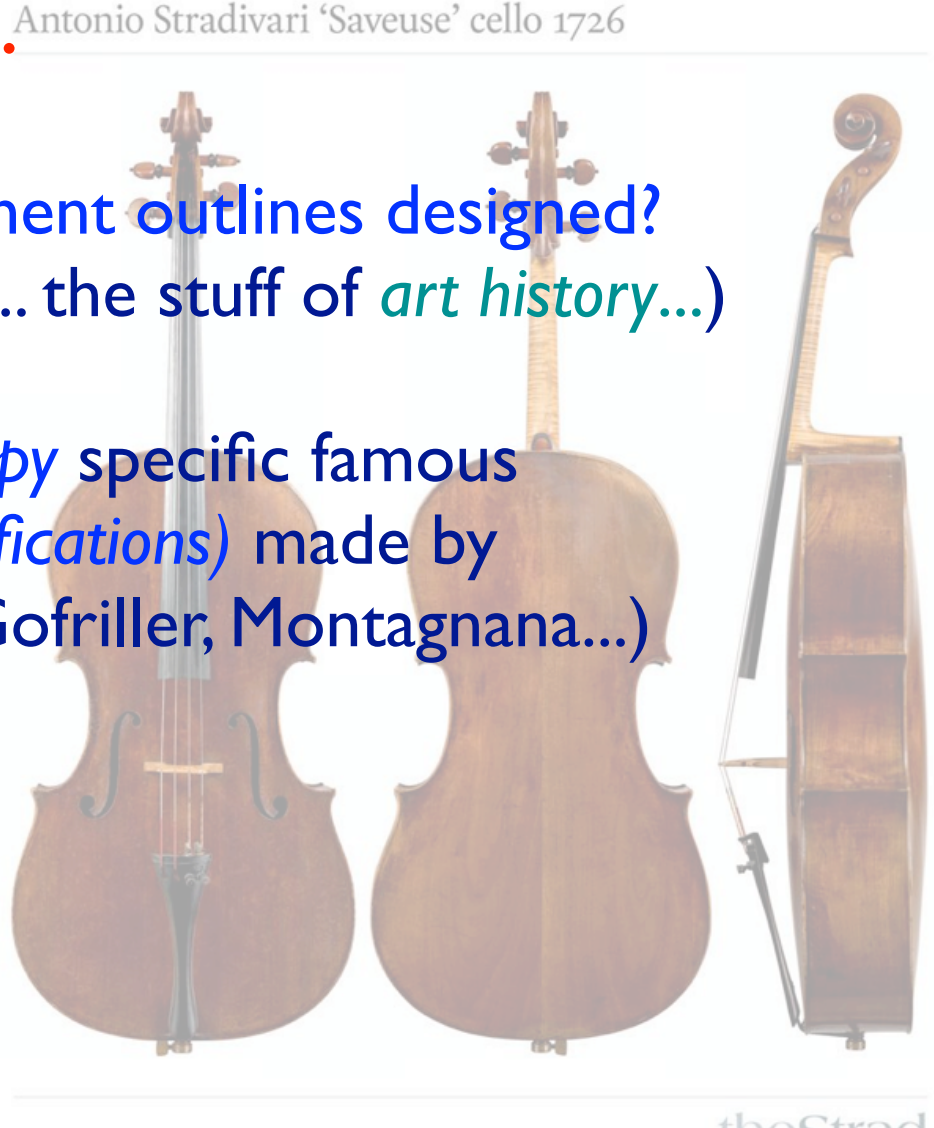


OK, enough already...

How are string instrument outlines designed?
(Every one is different... the stuff of *art history*...)

Modern day makers *copy* specific famous instruments *(with modifications)* made by
(Stradivari, Guarneri, Gofriller, Montagnana...)

What did **they** do?



Violin - Making

as it was, and is

(1885)

- by -

Ed. Heron-Allen

point *b*. Then place the compasses on the point 24, and opening them to *b*, draw the curve *a b a*.

Next set off 2 parts *c*, on each side of the perpendicular, on the horizontal line C. Place the compasses on the point *c*, and opening them to *a*, draw the curves *d d*, from *a* to the horizontal line A.

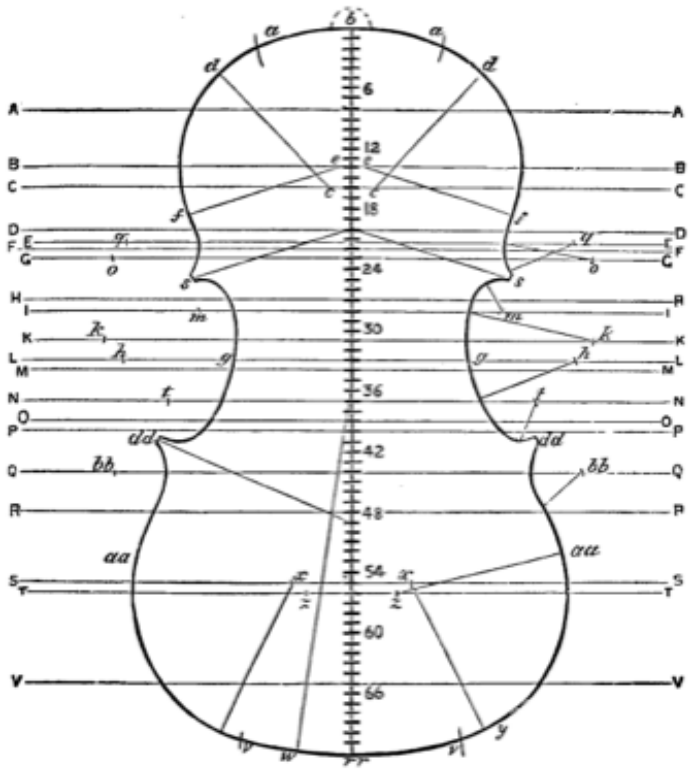


FIG. 79.—Method of drawing an outline mathematically on a given graduated straight line

Now set off one part *e*, on each side of the perpendicular on the line B. Place the compasses on these points, and opening them to the line A, where the curve *d* ends, draw the curves *f* from the line A to that of D. This completes the draught of the upper portion of the instrument without the corners.

Violin - Making

as it was, and is

(1885)

Ed. He

ANTONIO BAGATELLA

RÈGLES
pour la construction
des Violons Altis, Violoncelles
et Basses de Viole

◆

RULES
for the constructions of violins -
violas - cellos and double basses

◆

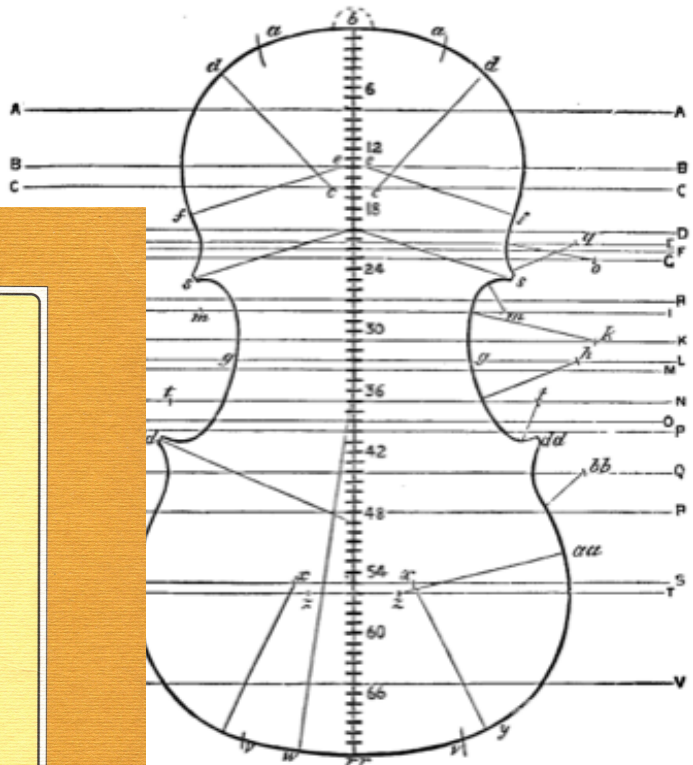
REGOLE
per la costruzione dei violini -
viole - violoncelli - violoni

PADOVA - MDCCLXXXII **(1782)**

TURRIS - CREMONA

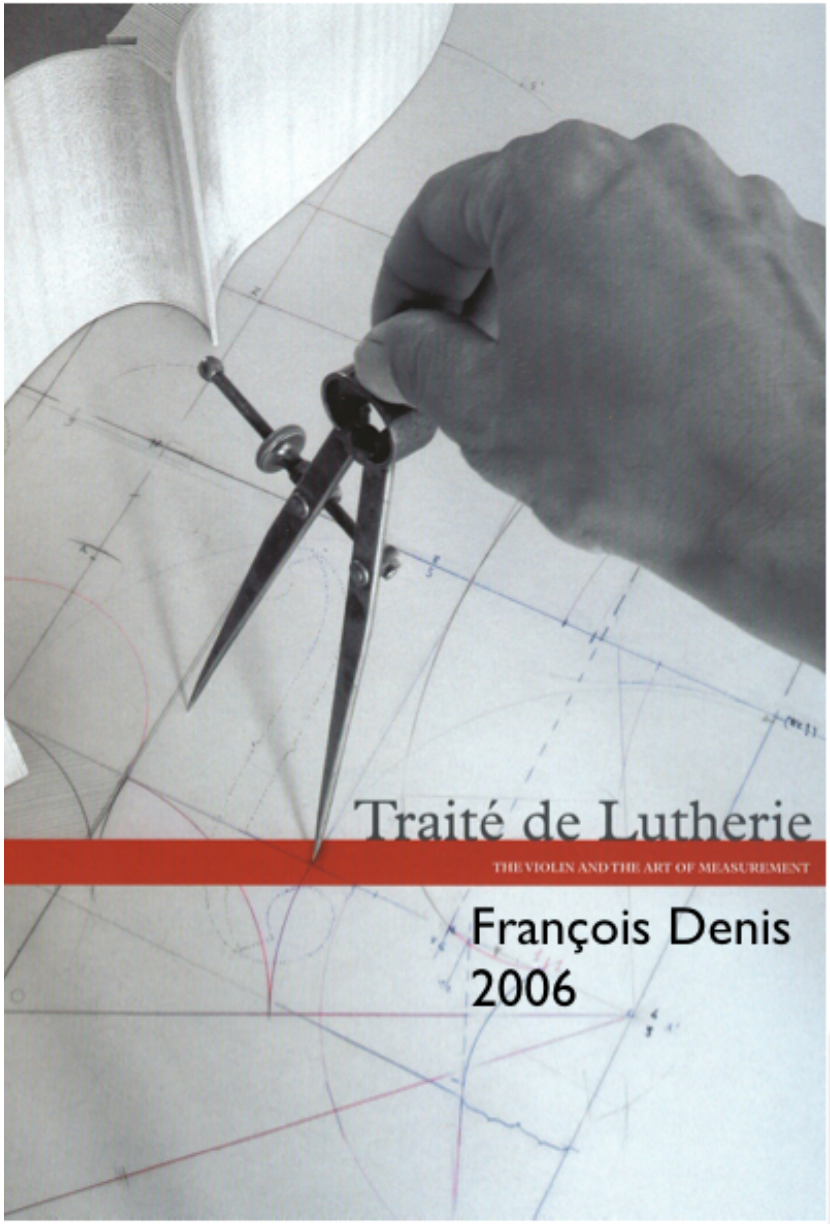
point *b*. Then place the compasses on the point 24, and opening them to *b*, draw the curve *a b a*.

Next set off 2 parts *c*, on each side of the perpendicular, on the horizontal line C. Place the compasses on the point *c*, and opening them to *a*, draw the curves *d d*, from *a* to the horizontal line A.



d of drawing an outline mathematically on a given graduated straight line

off one part *e*, on each side of the perpendicular on
Place the compasses on these points, and opening
line A, where the curve *d* ends, draw the curves *f*
ine A to that of D. This completes the draught of
portion of the instrument without the corners.



And then some time later…

Straightedge and compass:
no graph paper, rulers, Vernier calipers, protractors...

Luthiers did not know math!

Not ***Cartesian***
(how we do everything!)
but rather *Euclidean*...

I wanted to learn how to do this.

...but... done with pencil and paper?

Surely there must be a better way, where you can make mistakes and fix them faster!

Reading *Traité de Lutherie*

Technical writing is hard
(ex.: user's manual for assembling a gas grill…)

Repetition and parameterization
("Repeat the same construction, as in…")
—-with what parameters?

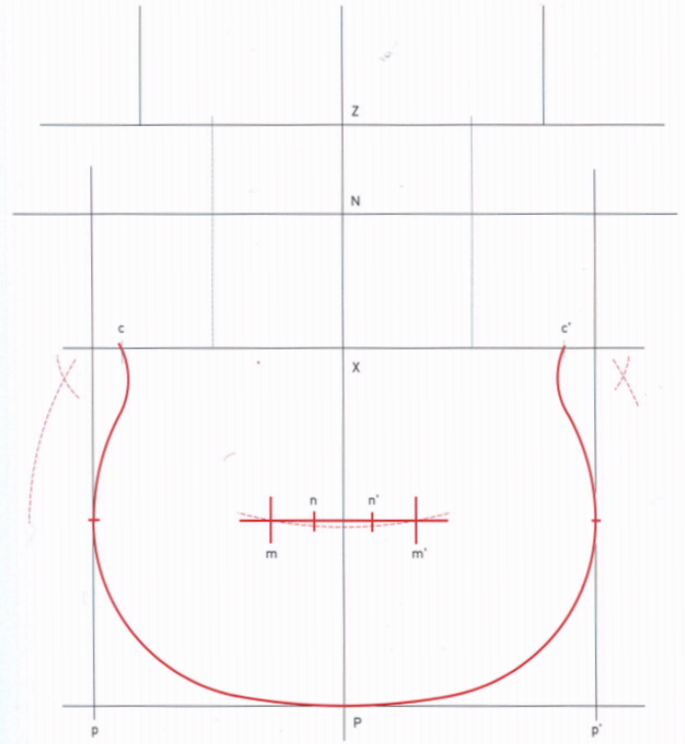
Hierarchy of design
(What depends on what? What changes affect what subsequent constructions?)

This is a problem asking for a ***programming language***,
or--at least--for a ***programming solution***...

A programming language is
a vernacular for expressing
*(computational, algorithmic) **ideas**,*
i.e, computational thinking...about the past

TO DRAW THE REVERSE CURVE OF THE LOWER CORNERS:

- Draw a horizontal line passing through "m" and "m'".
- Place "n" on the horizontal line passing through "m" and "m'" at a distance XZ inside the vertical line passing through "p". Repeat the process symmetrically to place "n'".
- Draw the arc of radius ZX with centre "n" tangent to the vertical passing through "p" and repeat the process symmetrically from "n'".
- MP and ZX are the radii of the arcs tangent to the lower bout width.
- Take an arc radius equal to XZ + XN/2 with centres "n" and "n'" and draw two arcs rising towards the horizontal line passing through X.
- Take an arc radius equal to XN/2, place one point of the compass on "c" and find the centre on the preceding arc of radius XZ + XN/2 (and centre "n"). Draw the lower corner arc whose radius XN/2 is tangent to the arc of radius XZ and centre "n". Repeat the process symmetrically from "c'" and "n'".



Lower arcs	
R1, radius of the arc tangent to the bottom block	R1 = ZP
R2, radius of the flank arc, tangent to the width	R2 = PM = XP/2
R3, radius of the reverse curve arc, tangent to the width	R3 = XZ
R4, radius of the corner reverse curve arc	R4 = XN/2

Table of lower bout arc radius measurements

What I've done so far...

A system of graphics primitives on top of the *Scheme* programming language, *à la* Peter Henderson's *functional geometry* (ca. 1980s)

Language-based, not WYSIWYG (so as to *preserve the luthier's vernacular,* virtually verbatim, as described by Denis).

Points, lines, circles (a programmable straightedge and compass machine)

Geometric objects constructed by intersections
(*no math/numbers*---though the math is of course at the back end)

Beginning catalog of instruments and related analysis

WYSIWYG ("What you see is what you get")
versus a markup language, where you can
see what you did to get what you got

WYSIWYG ("What you see is what you get")
versus a markup language, where you can
see what you did to get what you got

Word processing

WYSIWYG (Microsoft Word)

Markup (LaTeX):

```
Now consider the rational approximation of the subharmonic section.
Let $s+m=1$ where $s$ and $m$ are an approximation of this section:
then a better approximation is $s'=m$ and $m'=s+2m$.
Solving ${s\over m}={{s'}\over{m'}}$ gives $s=1-{1\over{\sqrt{2}}}$,
the smaller part of the section.
```

Now consider the rational approximation of the subharmonic section. Let $s+m=1$ where s and m are an approximation of this section: then a better approximation is $s'=m$ and $m'=s+2m$. Solving $\frac{s}{m}=\frac{s'}{m'}$ gives $s=1-\frac{1}{\sqrt{2}}$, the smaller part of the section.

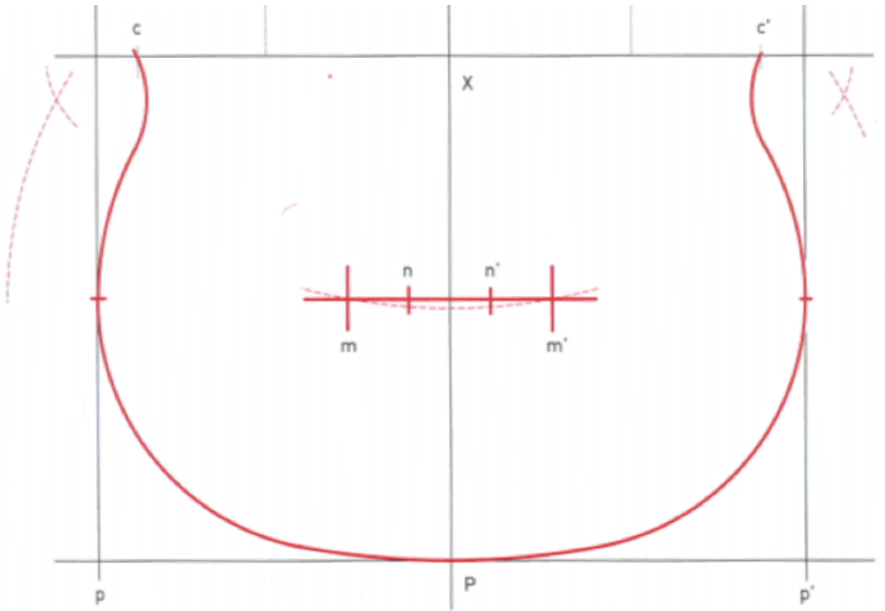
WYSIWYG ("What you see is what you get")
versus a markup language, where you can see what you did to get what you got

String instrument outlines

WYSIWYG (CAD/CAM systems)

Markup: What I'm trying to do here

Instructions that describe an outline (*à la* Denis)



TO DRAW THE REVERSE CURVE OF THE LOWER CORNERS:

- Draw a horizontal line passing through "m" and "m'".
- Place "n" on the horizontal line passing through "m" and "m'" at a distance XZ inside the vertical line passing through "p". Repeat the process symmetrically to place "n'".
- Draw the arc of radius ZX with centre "n" tangent to the vertical passing through "p" and repeat the process symmetrically from "n'".
- MP and ZX are the radii of the arcs tangent to the lower bout width.
- Take an arc radius equal to XZ + XN/2 with centres "n" and "n'" and draw two arcs rising towards the horizontal line passing through X.
- Take an arc radius equal to XN/2, place one point of the compass on "c" and find the centre on the preceding arc of radius XZ + XN/2 (and centre "n"). Draw the lower corner arc whose radius XN/2 is tangent to the arc of radius XZ and centre "n". Repeat the process symmetrically from "c'" and "n'".

WYSIWYG ("What you see is what you get")
versus a markup language, where you can see what you did to get what you got

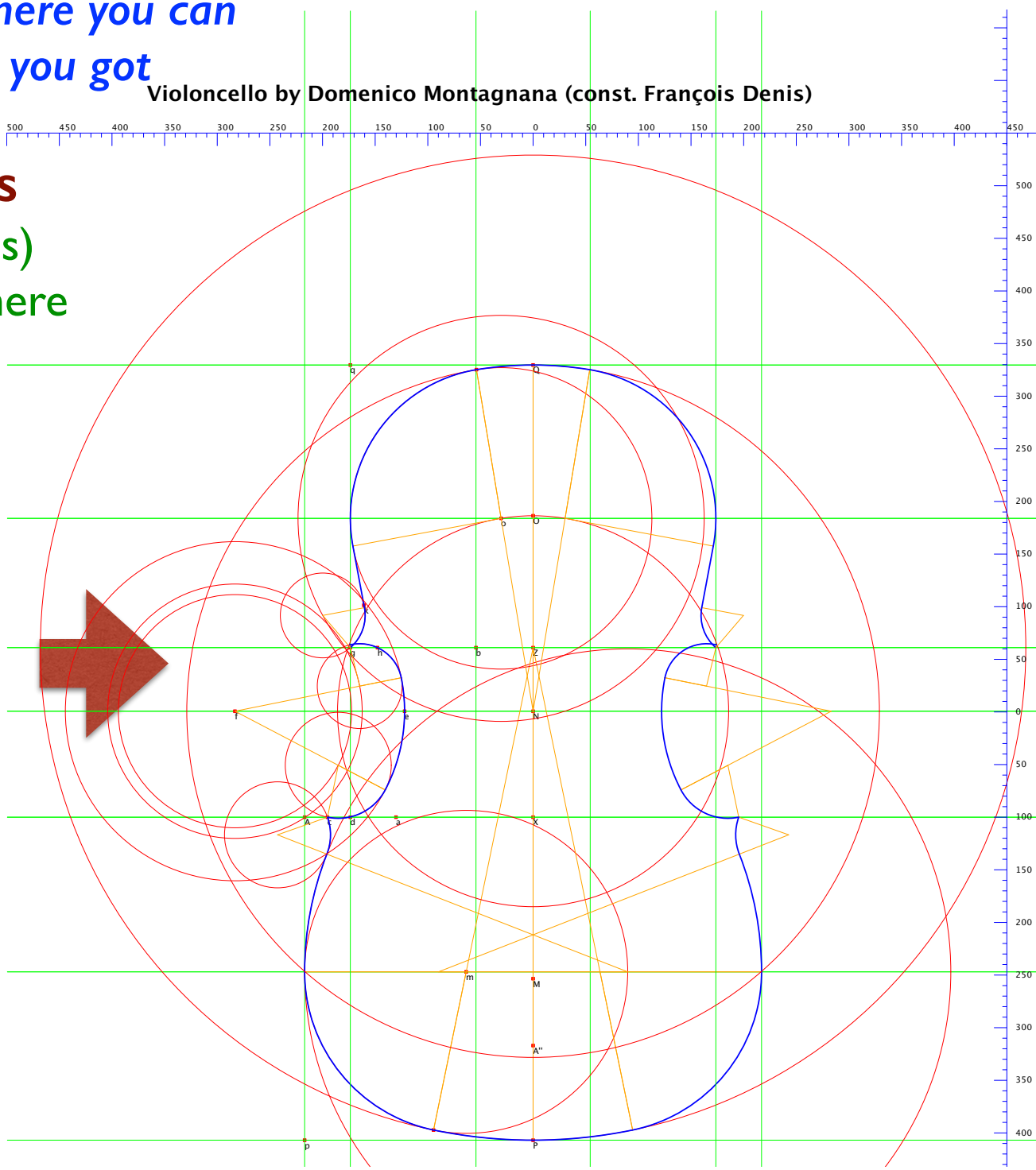
String instrument outlines

WYSIWYG (CAD/CAM systems)

Markup: What I'm trying to do here

Instructions that describe an outline (*à la* Denis)

in effect: the constructional details of his book, *made into executable computer code…*



WYSIWYG ("What you see is what you get")
versus a markup language, where you can see what you did to get what you got

String instrument outlines

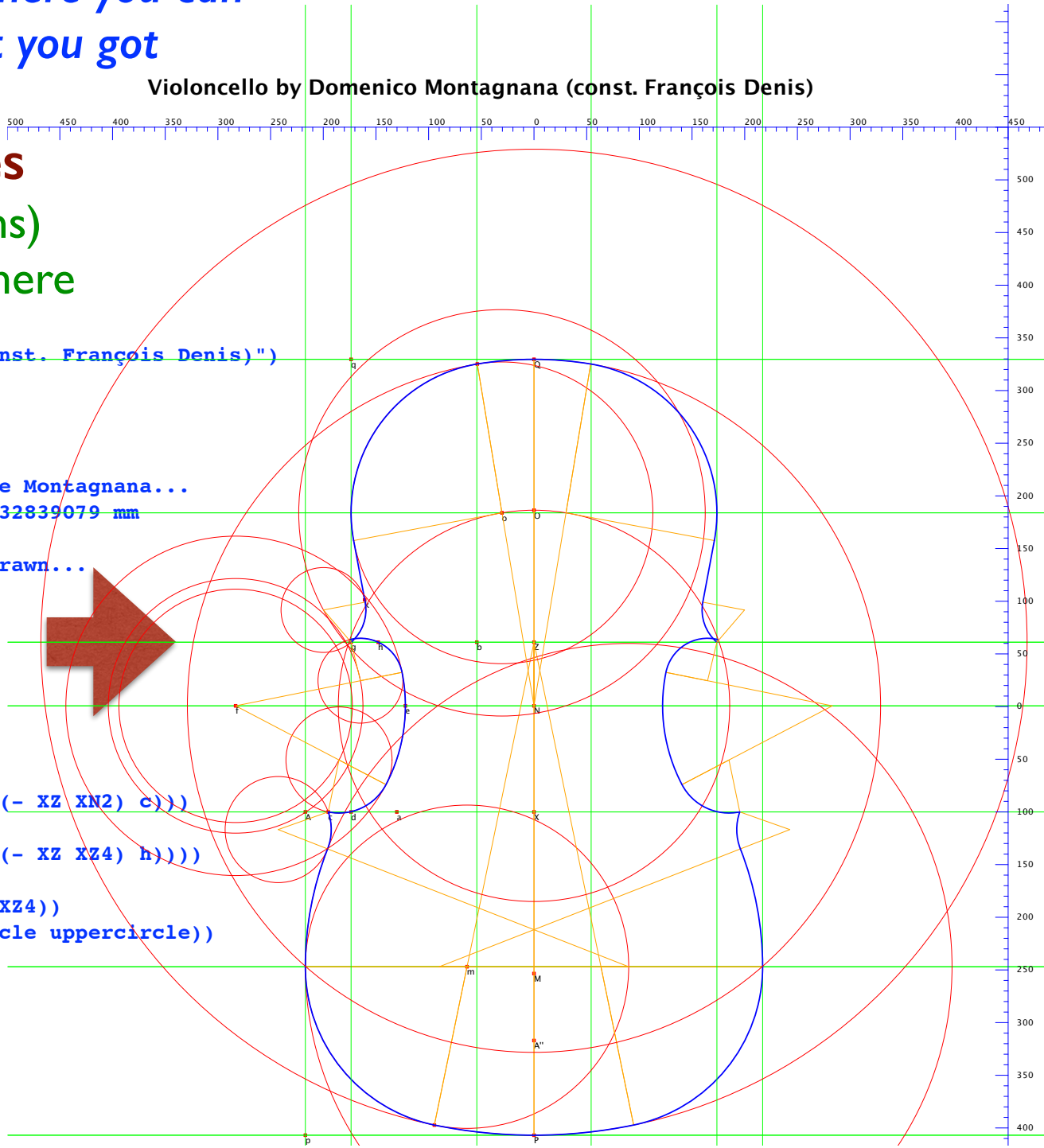
WYSIWYG (CAD/CAM systems)

Markup: What I'm trying to do here

```
(title "Violoncello by Domenico Montagnana (const. François Denis)")

; Cello by Domenico Montagnana

(define (Montagnana)
  (let ((aaprime 434)) ; should be 434mm in the Montagnana...
                       ; body length 736.5224232839079 mm
; LAYOUT OF THE AREA on which the curves are drawn...

…
; Middle bouts

(let* ((f (label "f" (xshift e (- XZ))))
       (fcircle (circle f XZ))
       (XN2 (/ (distance X N) 2))
       (lowercircle
         (upper-circle (reverse-curve fcircle (- XZ XN2) c)))
       (uppercircle
         (lower-circle (reverse-curve fcircle (- XZ XZ4) h))))
  (list f lowercircle uppercircle fcircle
       (circle f (- XZ XN2)) (circle f (- XZ XZ4))
       (make-curve c g (list lowercircle fcircle uppercircle))
```

Instructions that describe an outline (

in effect: the constructional details of his book, executable computer code…



Drawing a triangle or a square is easy with a straightedge and compass.

But how do you draw a pentagon?

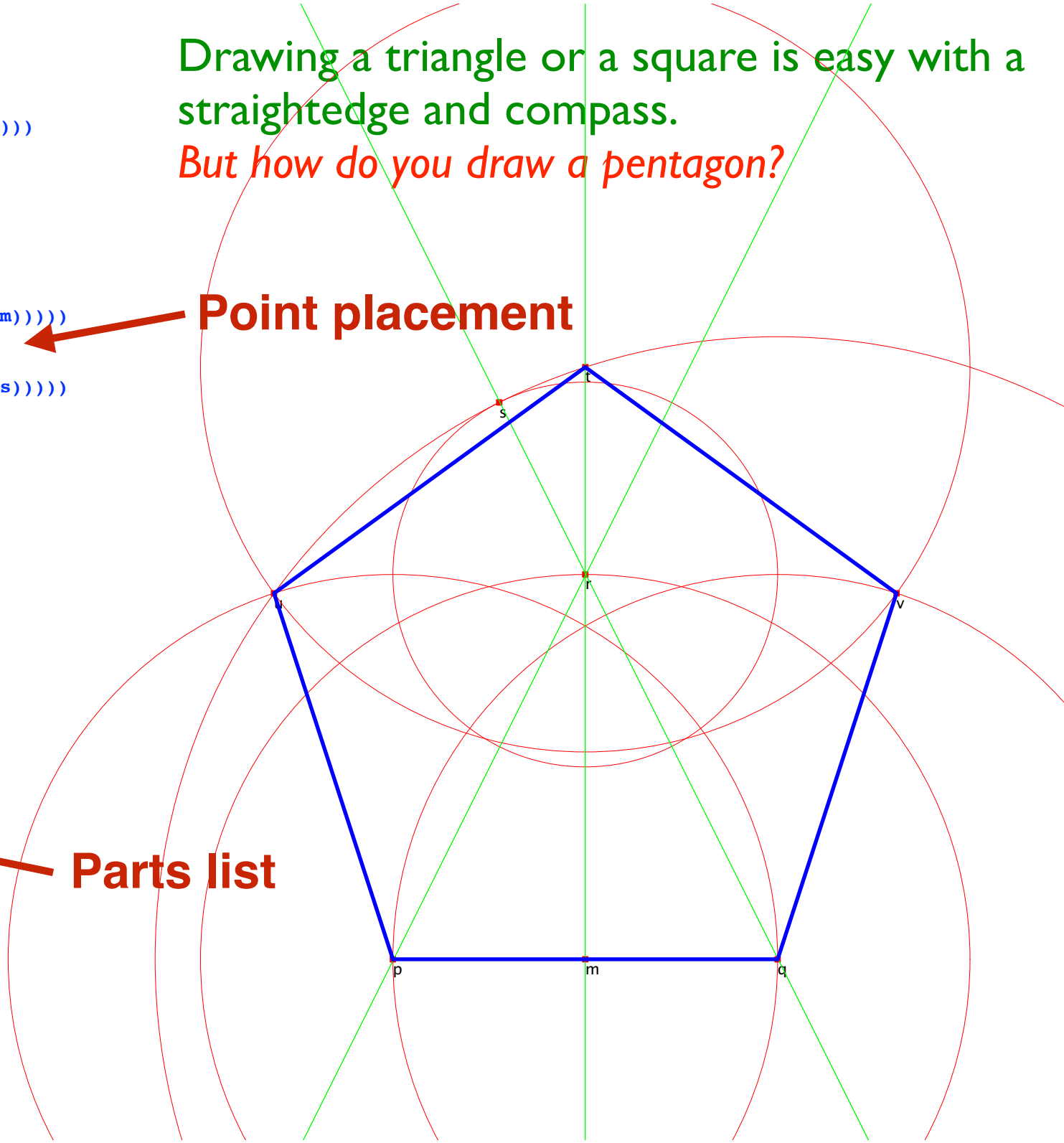
Drawing a triangle or a square is easy with a straightedge and compass.

But how do you draw a pentagon?

```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```

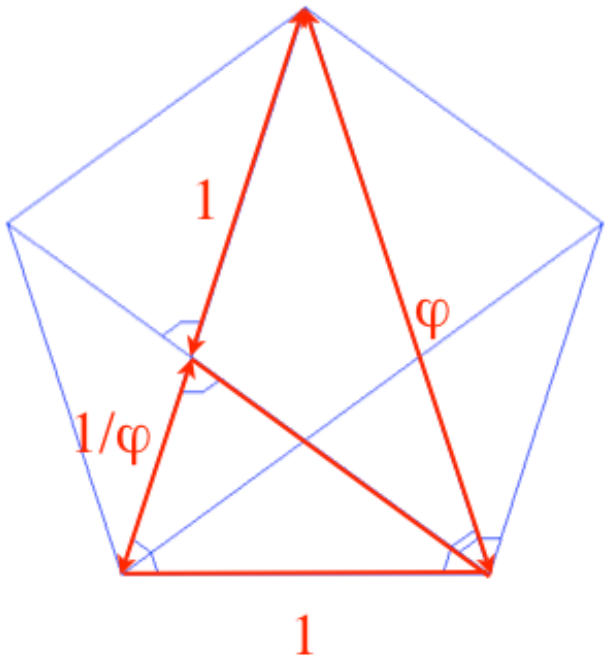
Point placement

Parts list



Pentagons, etc.

(The big red and lower red triangles are similar.)



$$\varphi = 1 + 1/\varphi$$

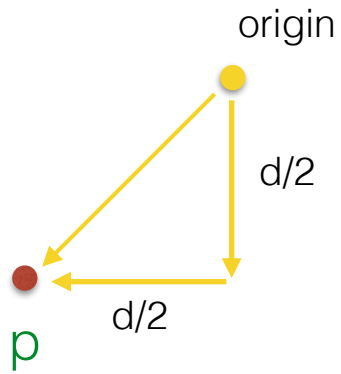
$$\varphi^2 - \varphi - 1 = 0$$

$$\varphi = \tfrac{1}{2}(1 + \sqrt{5})$$

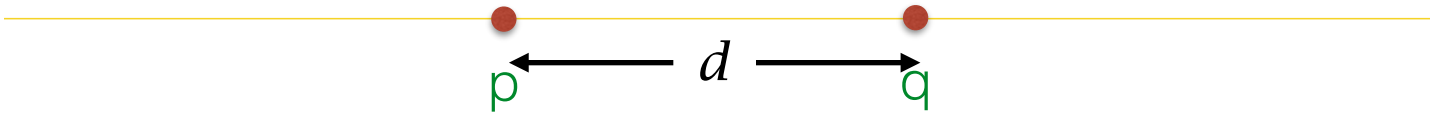
"diameter" of pentagon

“to draw a pentagon with side d,”

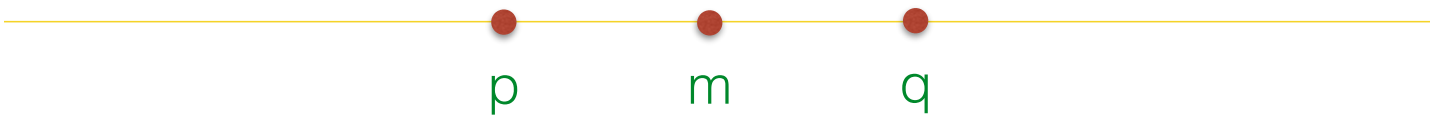
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



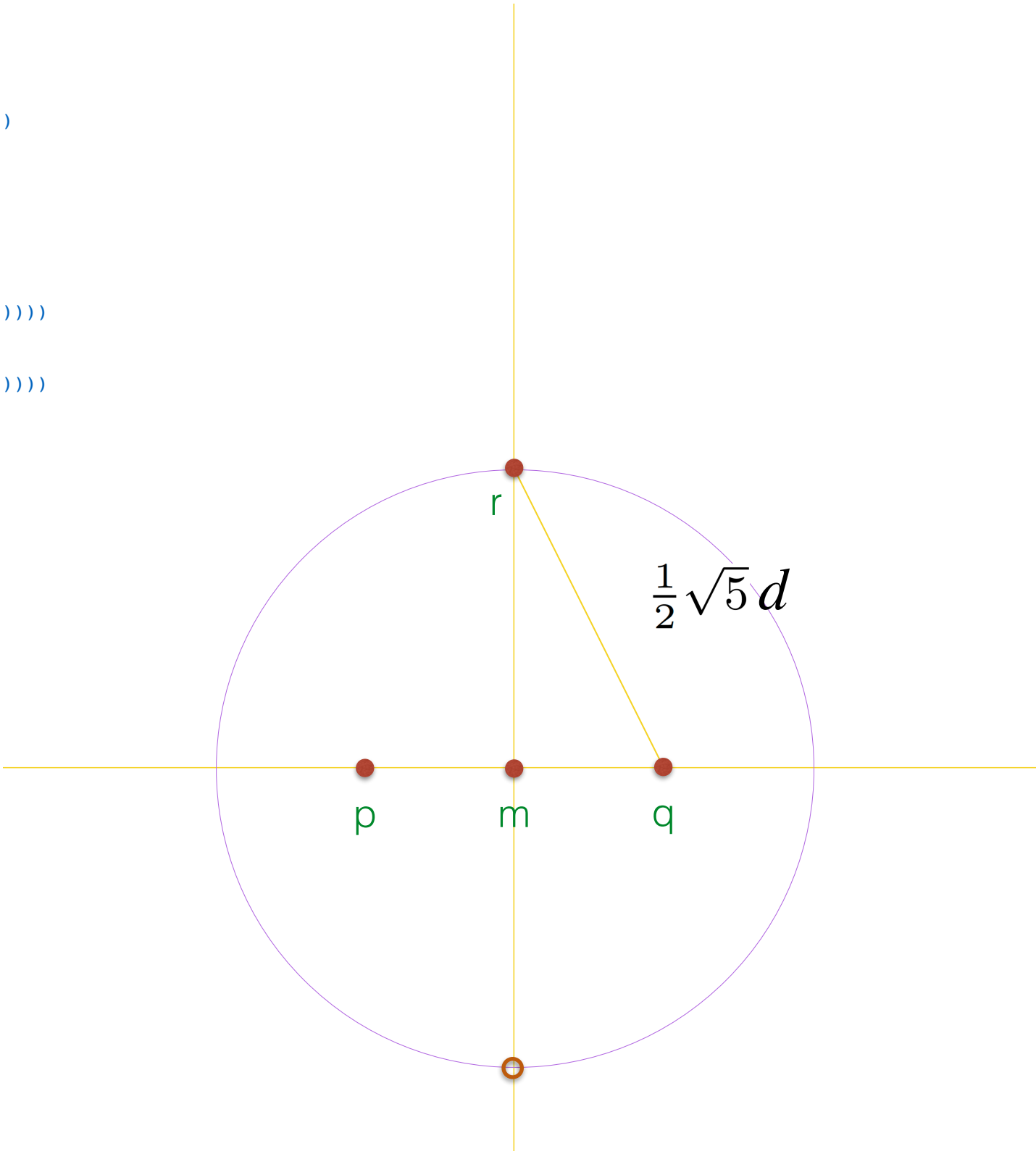
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



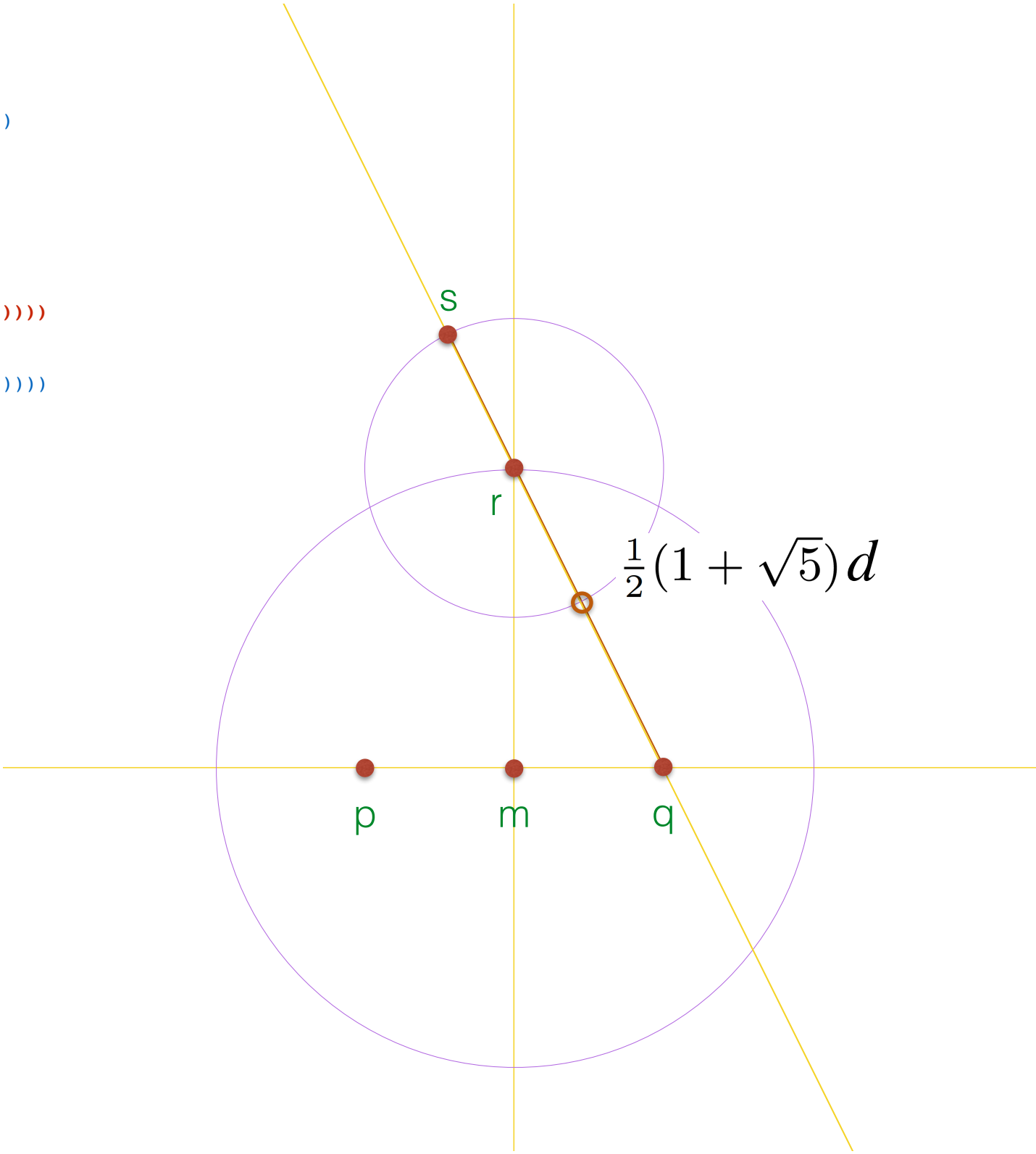
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



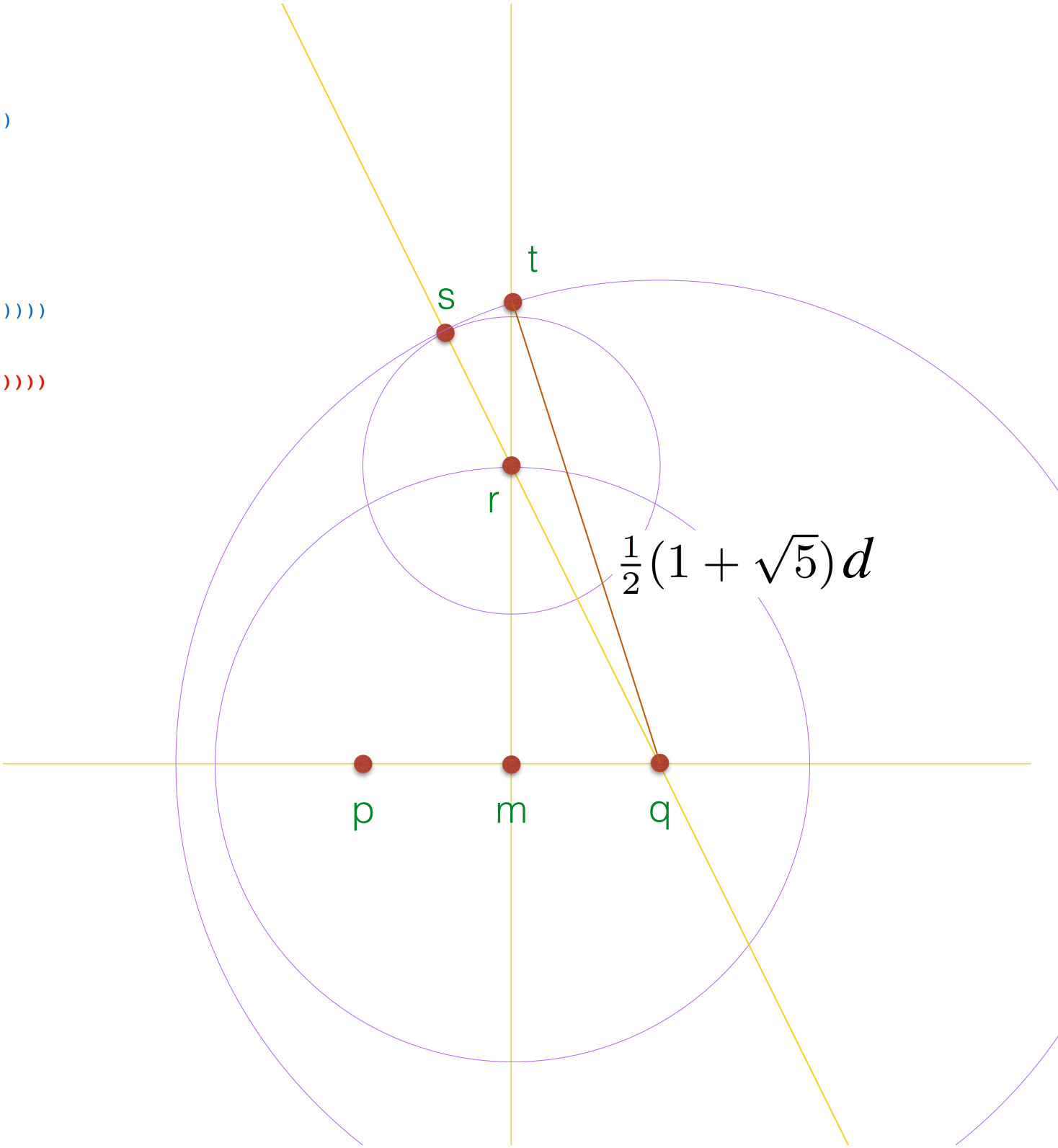
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



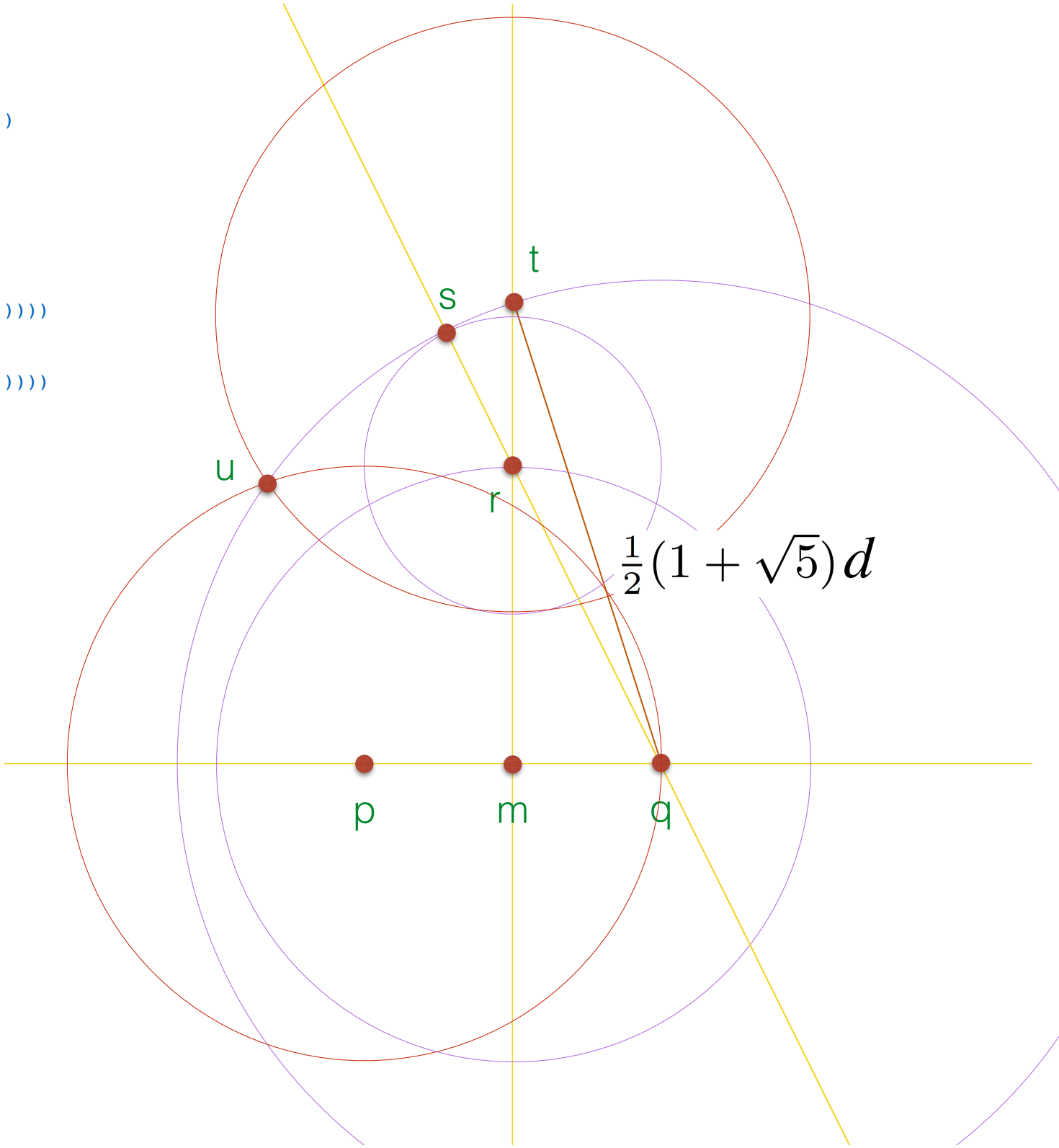
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



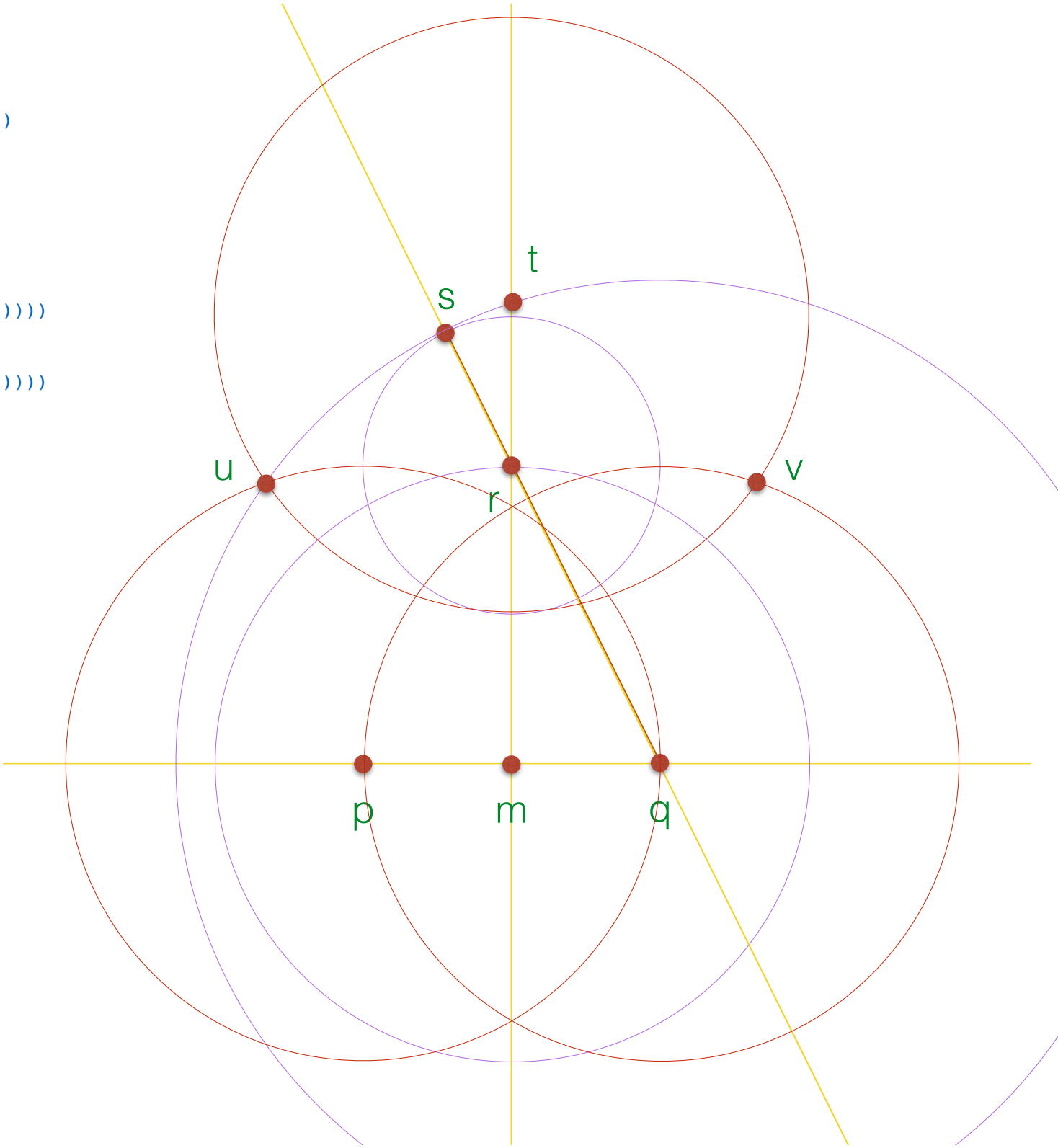
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



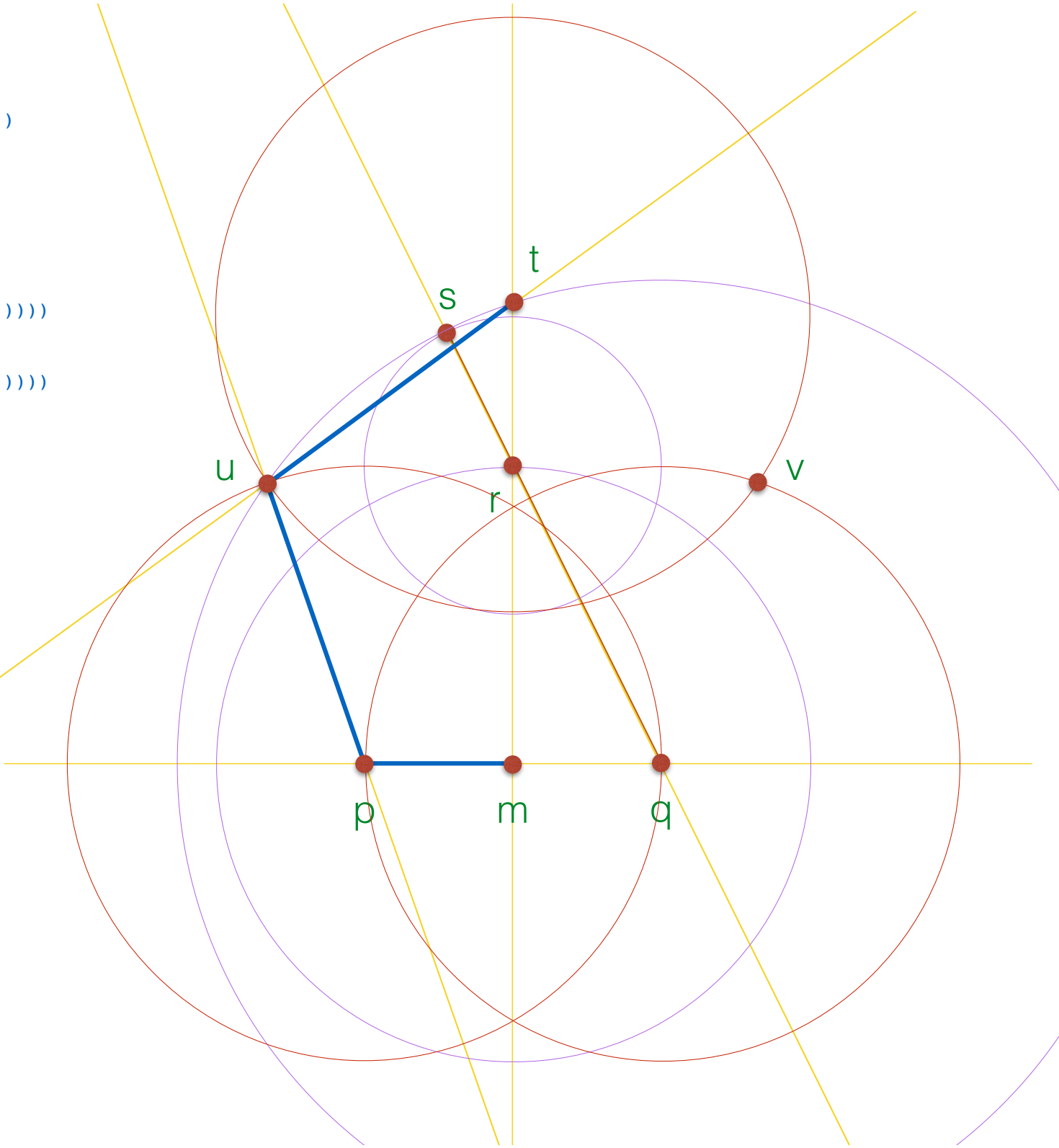
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



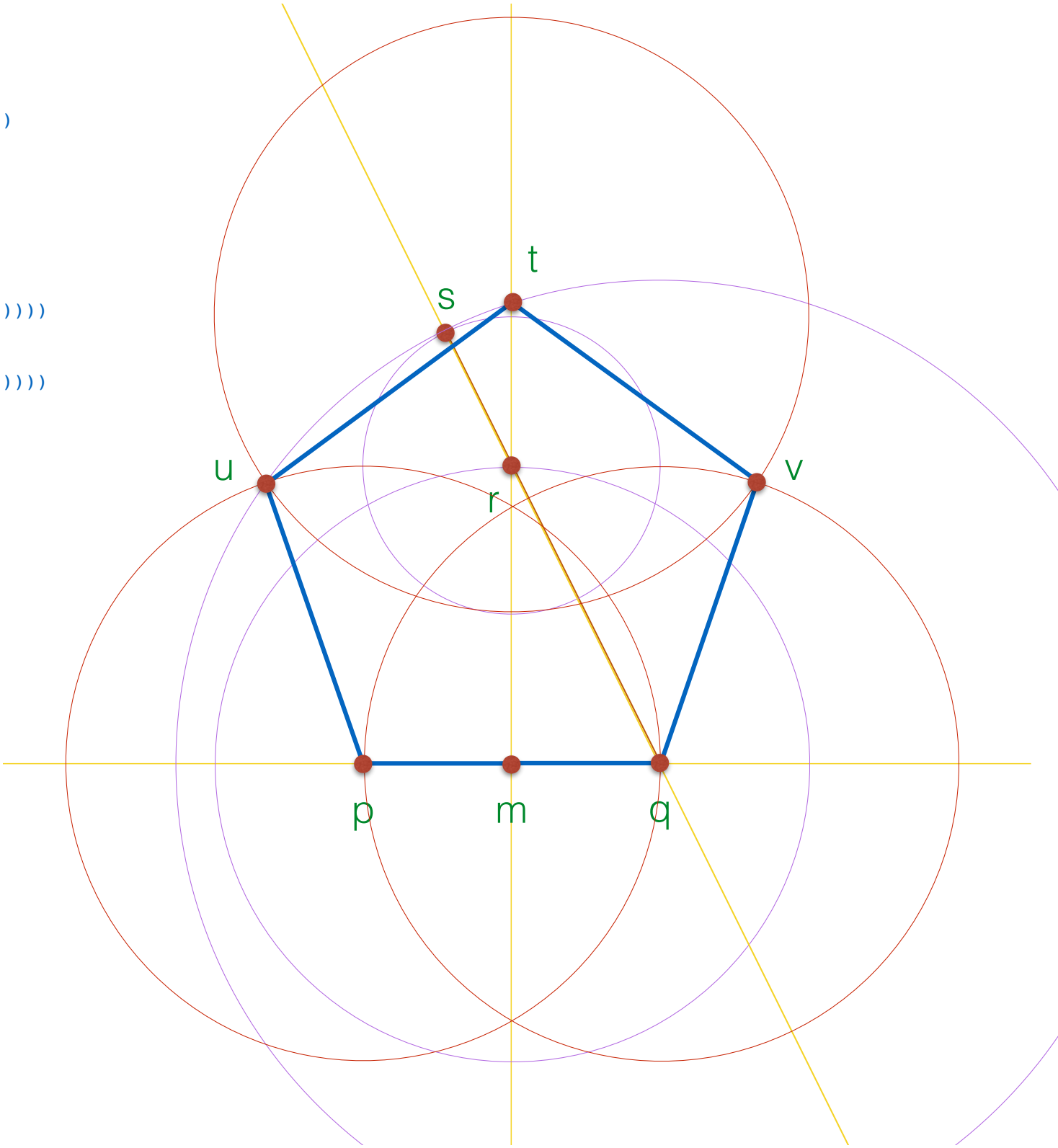
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```

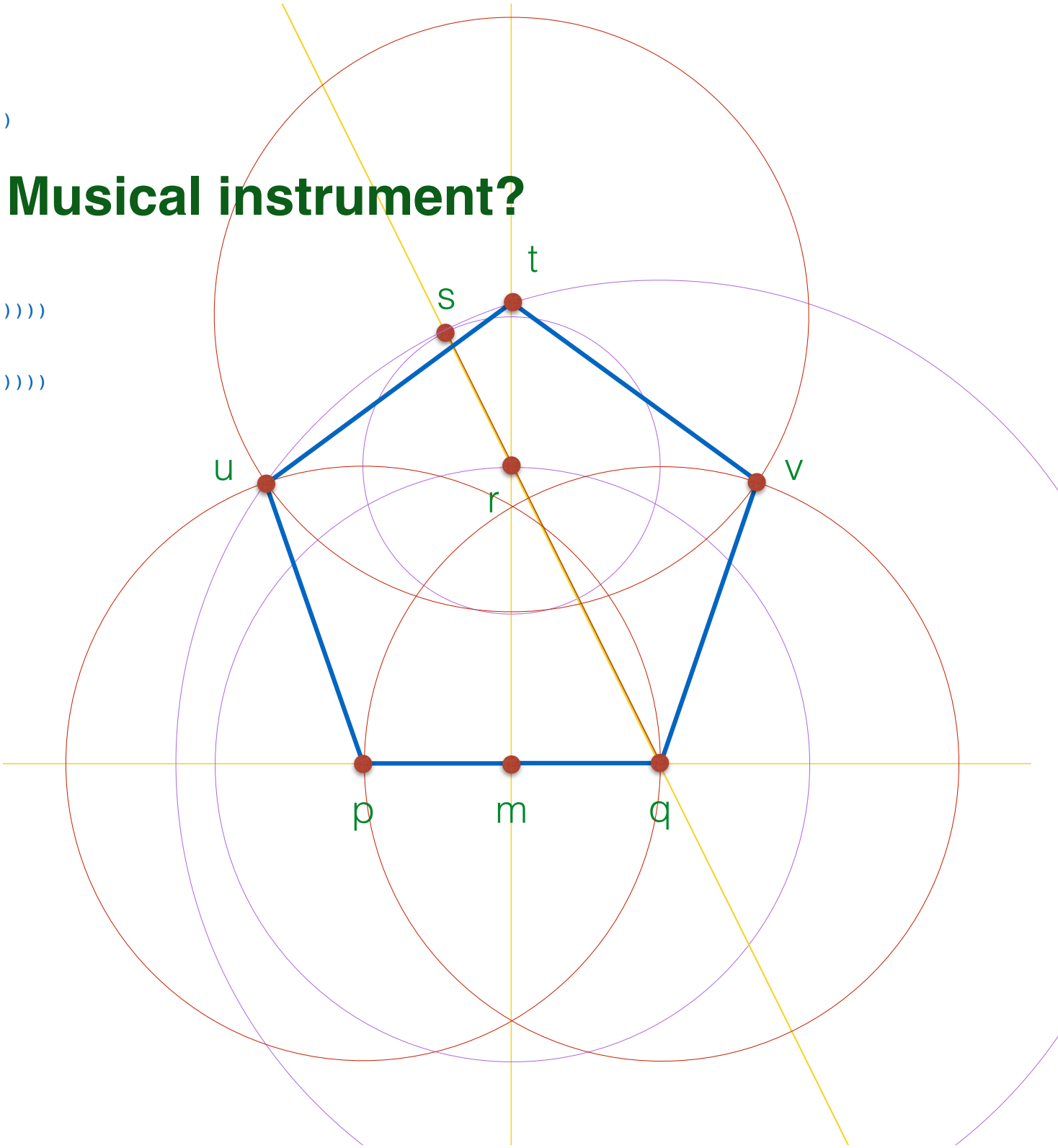


```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```

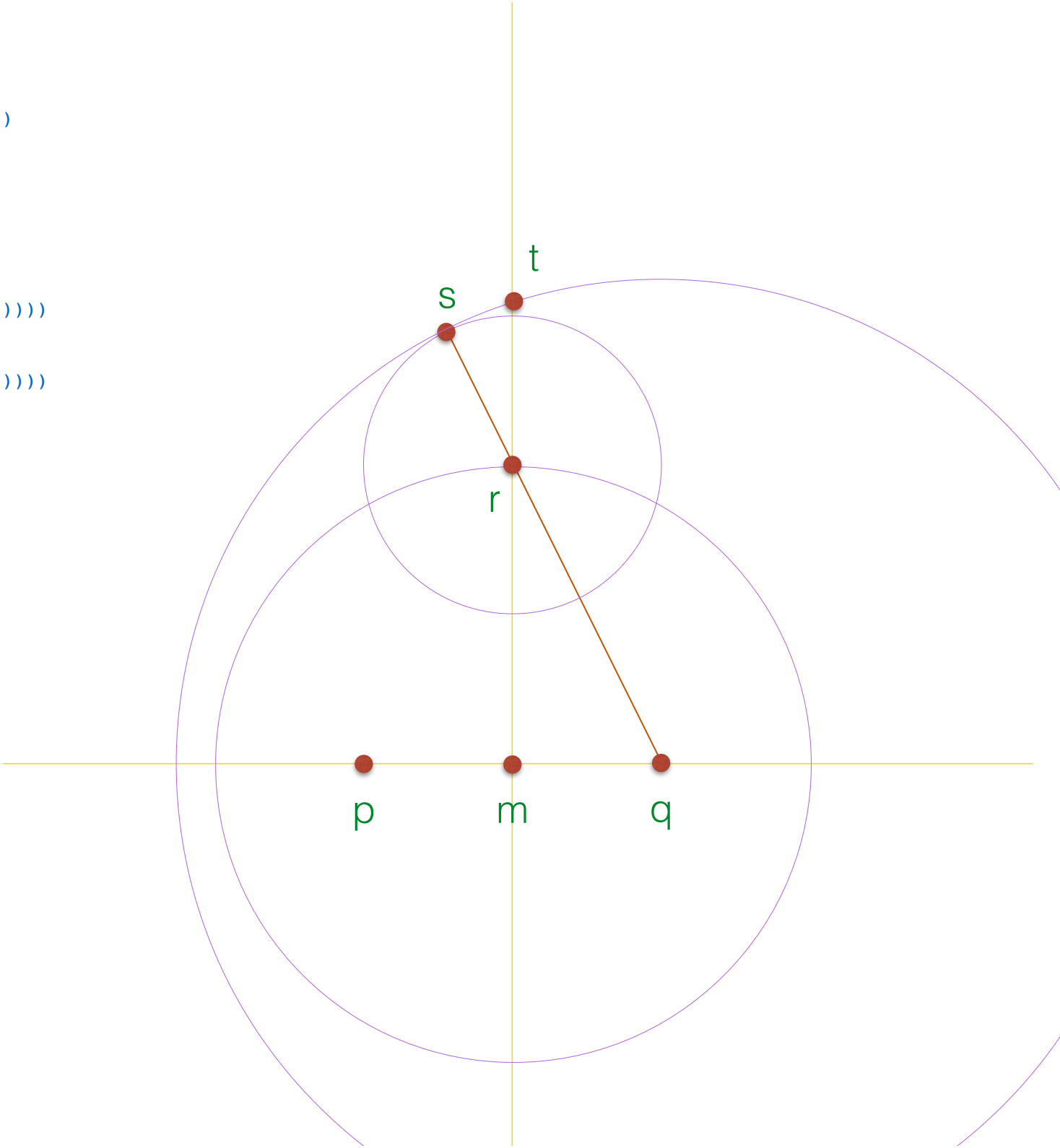


```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```

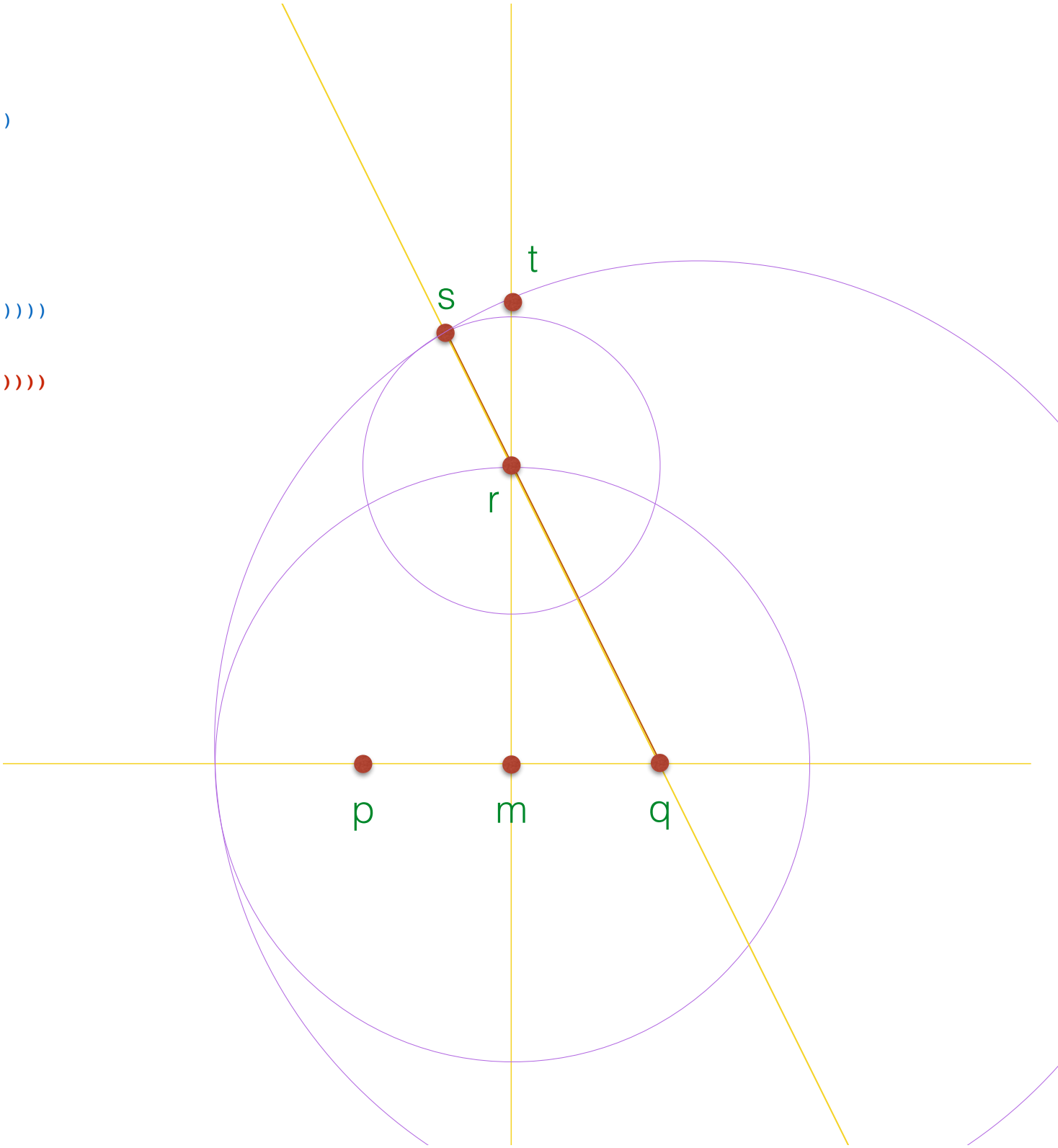
Musical instrument?



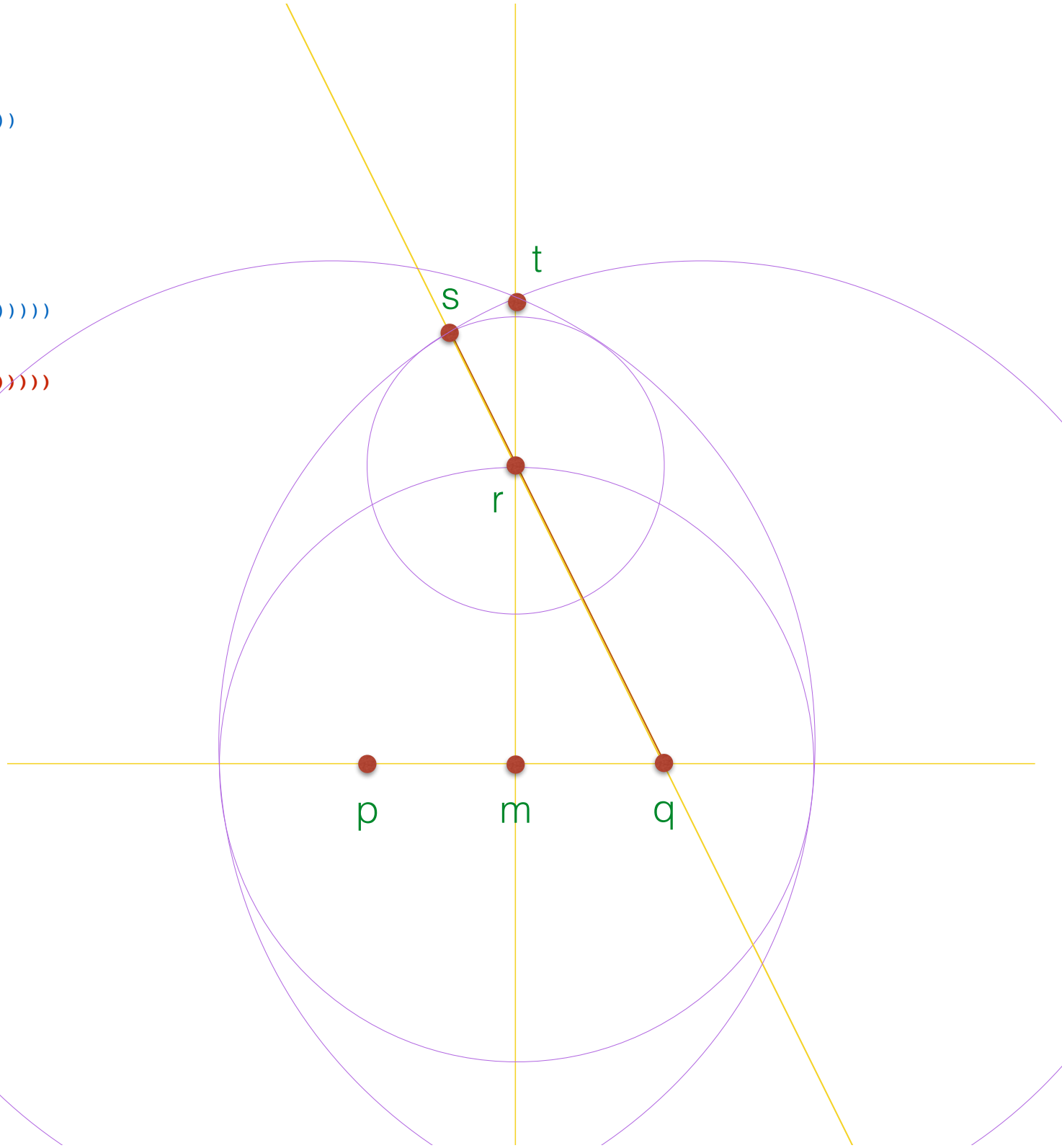
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



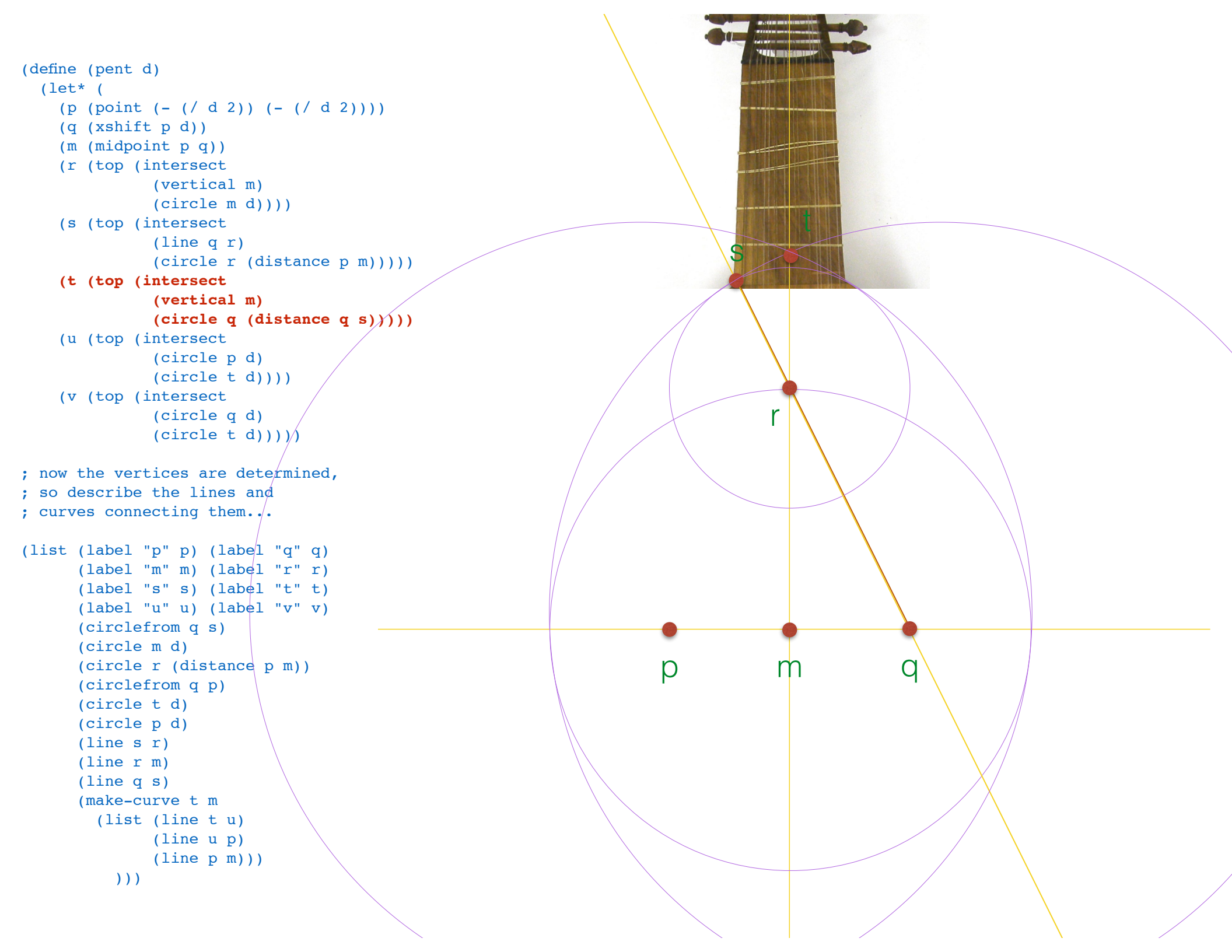
```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```

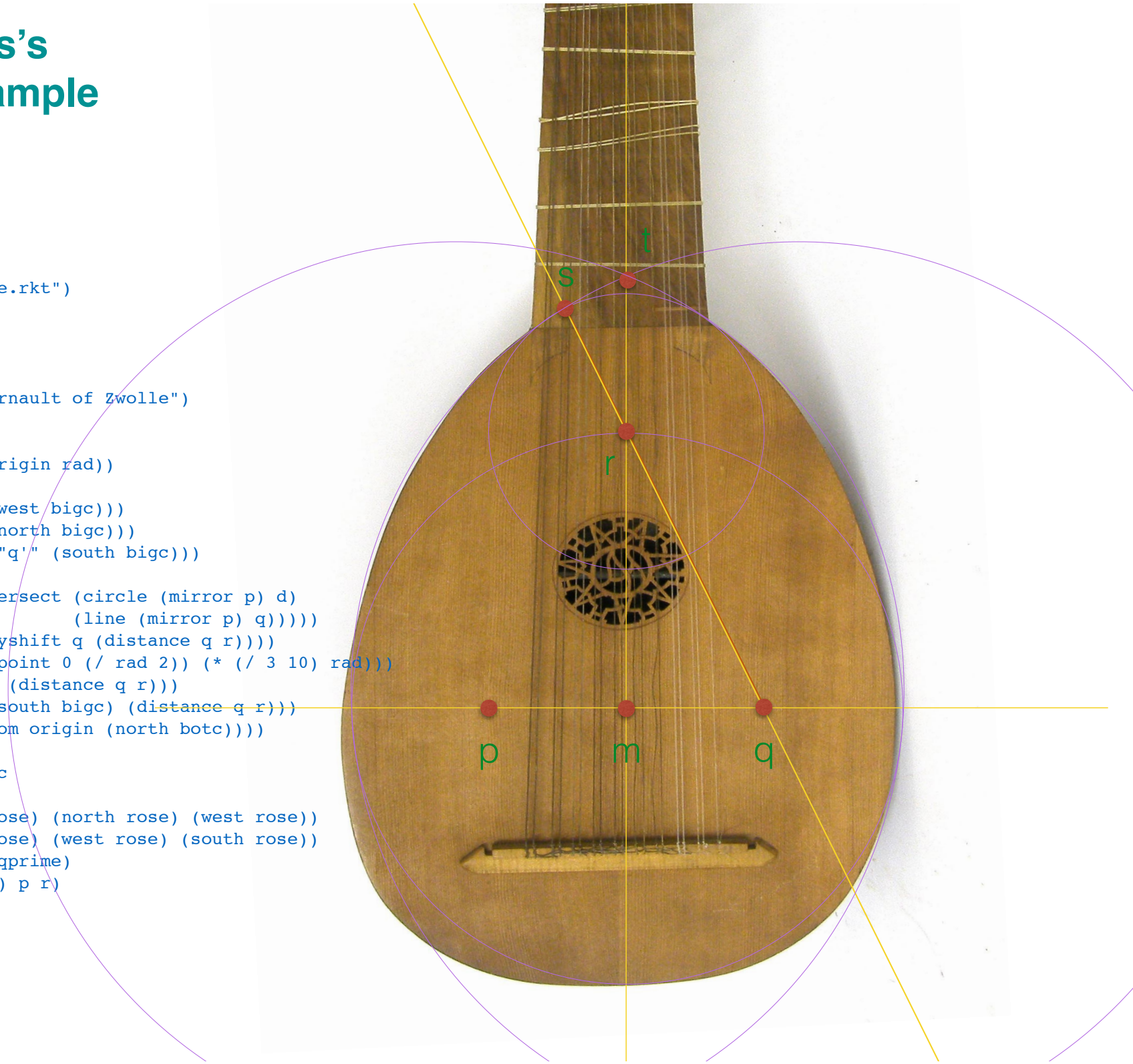


```
(define (pent d)
  (let* (
    (p (point (- (/ d 2)) (- (/ d 2))))
    (q (xshift p d))
    (m (midpoint p q))
    (r (top (intersect
              (vertical m)
              (circle m d))))
    (s (top (intersect
              (line q r)
              (circle r (distance p m)))))
    (t (top (intersect
              (vertical m)
              (circle q (distance q s)))))
    (u (top (intersect
              (circle p d)
              (circle t d))))
    (v (top (intersect
              (circle q d)
              (circle t d)))))

; now the vertices are determined,
; so describe the lines and
; curves connecting them...

(list (label "p" p) (label "q" q)
      (label "m" m) (label "r" r)
      (label "s" s) (label "t" t)
      (label "u" u) (label "v" v)
      (circlefrom q s)
      (circle m d)
      (circle r (distance p m))
      (circlefrom q p)
      (circle t d)
      (circle p d)
      (line s r)
      (line r m)
      (line q s)
      (make-curve t m
        (list (line t u)
              (line u p)
              (line p m)))
        )))
```



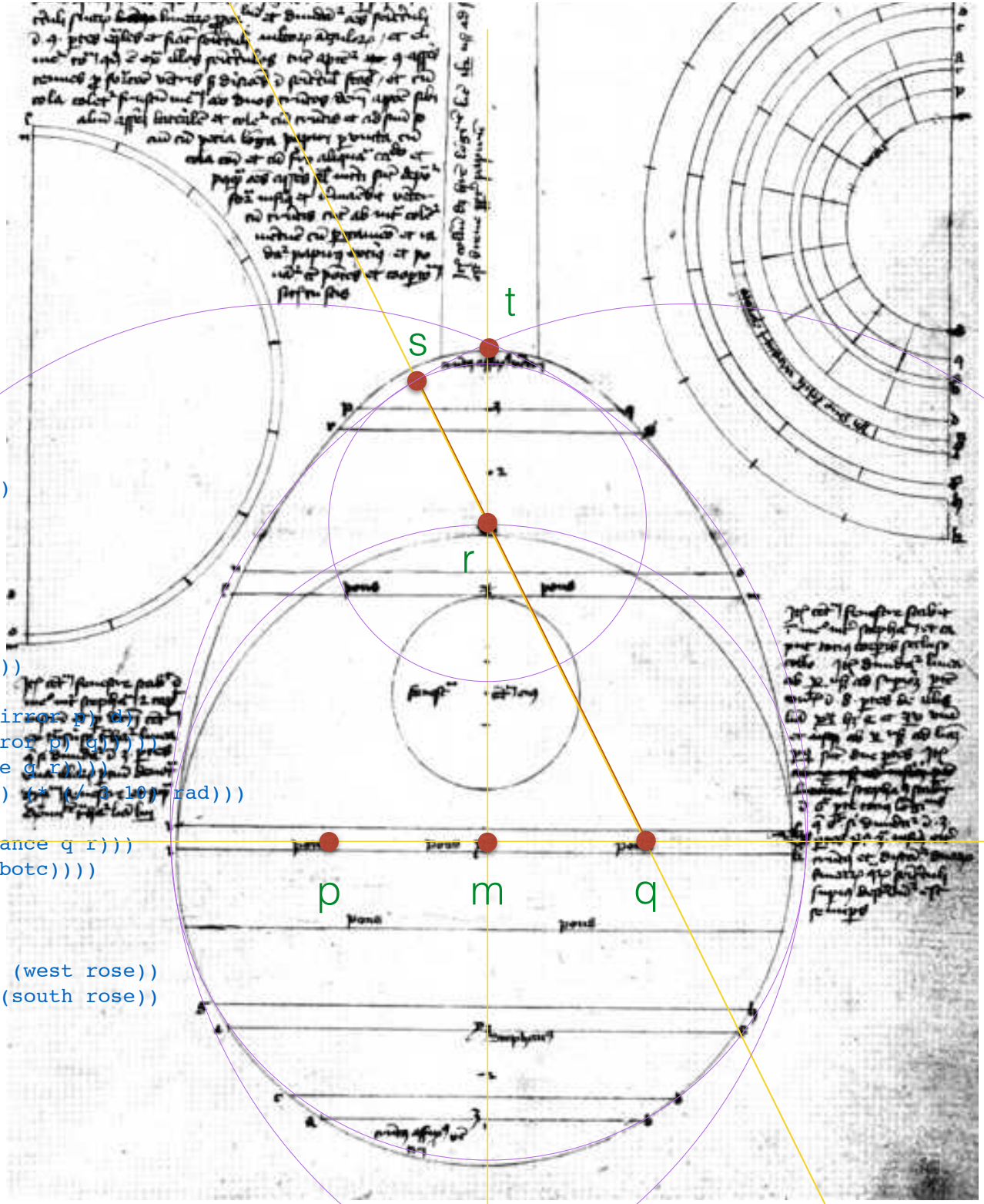
Arnault: Denis's canonical example

```
#lang racket

(require "Geometry-Engine.rkt")

(mirroring #t)
(arcthickness 2)

(title "Lute design by Arnault of Zwolle")

(define (lute rad)
  (let* ((bigc (circle origin rad))
         (d (* 2 rad))
         (p (label "p" (west bigc)))
         (q (label "q" (north bigc)))
         (qprime (label "q'" (south bigc)))
         (r (label "r"
              (left (intersect (circle (mirror p) d)
                               (line (mirror p) q)))))
         (s (label "s" (yshift q (distance q r))))
         (rose (circle (point 0 (/ rad 2)) (* (/ 3 10) rad)))
         (topc (circle q (distance q r)))
         (botc (circle (south bigc) (distance q r)))
         (midc (circlefrom origin (north botc))))
    (list
      bigc topc botc midc
      ;draw the rose:
      (makearc (center rose) (north rose) (west rose))
      (makearc (center rose) (west rose) (south rose))
      (makearc origin p qprime)
      (makearc (mirror p) p r)
      (makearc q r s))))
```



Arnault: Denis's canonical example

```
#lang racket

(require "Geometry-Engine.rkt")

(mirroring #t)
(arcthickness 2)

(title "Lute design by Arnault of Zwolle")

(define (lute rad)
  (let* ((bigc (circle origin rad))
         (d (* 2 rad))
         (p (label "p" (west bigc)))
         (q (label "q" (north bigc)))
         (qprime (label "q'" (south bigc)))
         (r (label "r"
              (left (intersect (circle (mirror p) d)
                               (line (mirror p) q)))))
         (s (label "s" (yshift q (distance q r))))
         (rose (circle (point 0 (/ rad 2)) (* (/ 3 10) rad)))
         (topc (circle q (distance q r)))
         (botc (circle (south bigc) (distance q r)))
         (midc (circlefrom origin (north botc))))
    (list
      bigc topc botc midc
      ;draw the rose:
      (makearc (center rose) (north rose) (west rose))
      (makearc (center rose) (west rose) (south rose))
      (makearc origin p qprime)
      (makearc (mirror p) p r)
      (makearc q r s))))
```

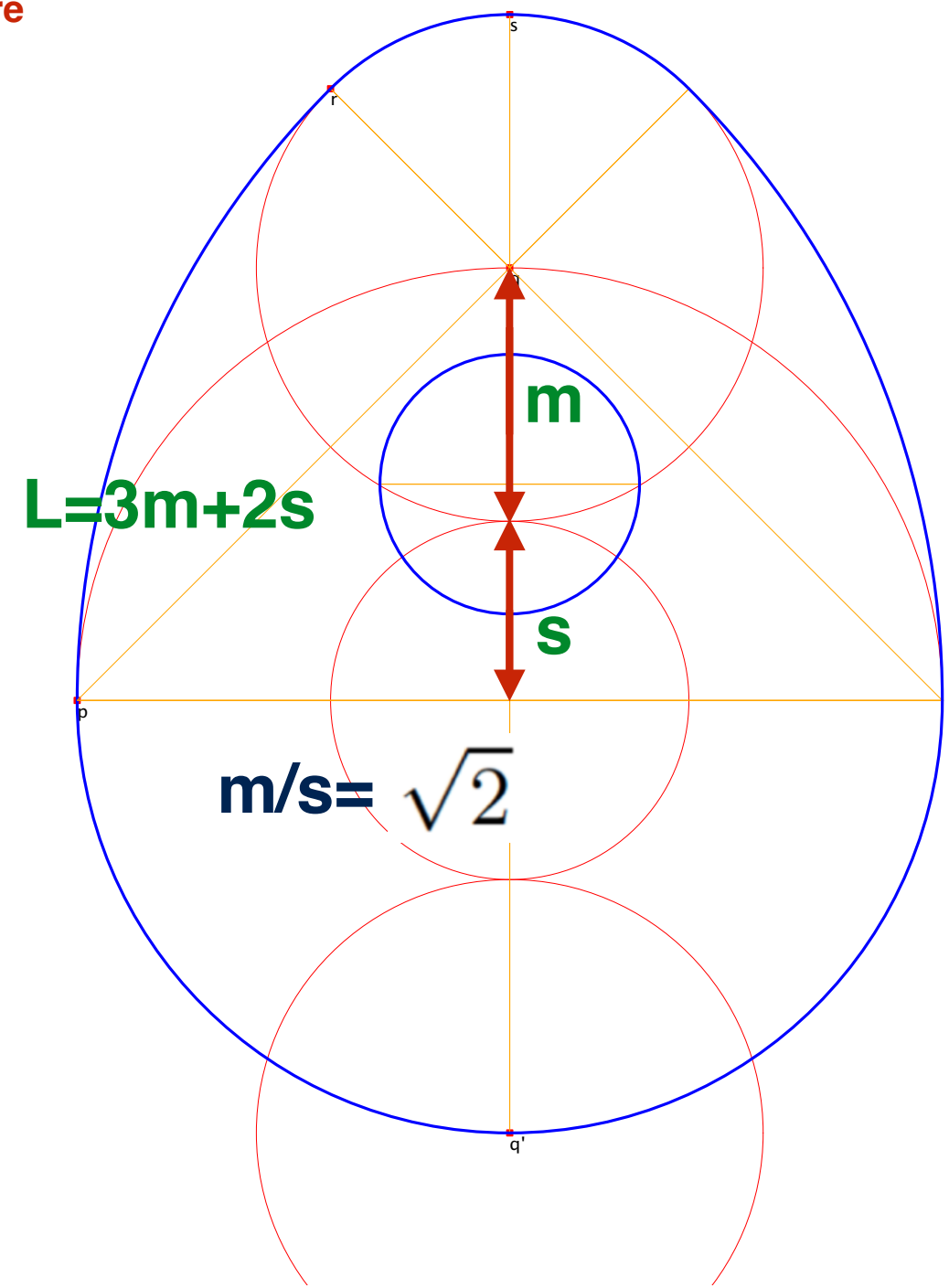


[Bibliothèque Nationale, 1440]

Arnault: Denis's canonical example

written version of oral tradition
ruler/compass construction
rational approximation
division of the square

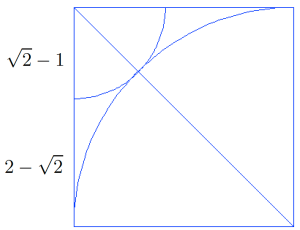
```
#lang racket

(require "Geometry-Engine.rkt")

(mirroring #t)
(arcthickness 2)

(title "Lute design by Arnault of Zwolle")

(define (lute rad)
  (let* ((bigc (circle origin rad))
         (d (* 2 rad))
         (p (label "p" (west bigc)))
         (q (label "q" (north bigc)))
         (qprime (label "q'" (south bigc)))
         (r (label "r"
              (left (intersect (circle (mirror p) d)
                               (line (mirror p) q)))))
         (s (label "s" (yshift q (distance q r))))
         (rose (circle (point 0 (/ rad 2)) (* (/ 3 10) rad)))
         (topc (circle q (distance q r)))
         (botc (circle (south bigc) (distance q r)))
         (midc (circlefrom origin (north botc))))
    (list
      bigc topc botc midc
      ;draw the rose:
      (makearc (center rose) (north rose) (west rose))
      (makearc (center rose) (west rose) (south rose))
      (makearc origin p qprime)
      (makearc (mirror p) p r)
      (makearc q r s))))
```



Arnault: Denis's canonical example

ruler/compass construction
rational approximation
division of the square

```
(define (approximate-lute s m L)
  (let* ((N (+ (* 2 s) (* 3 m)))
         (r (* L (/ (+ m s) N)))
         (q (label "q" (yshift origin
                               (minus r))))
         (p (label "p" (yshift q L)))
         (cmain (circle (yshift q r) r))
         (ctop (circle (north cmain)
                       (distance (north cmain) p)))
         (cright
           (circle (east cmain) (* r 2))))
(list
  ; bridge
  (apply makeseg
   (intersect
     cmain
     (horizontal (yshift q (* L (/ s N))))))
; sound hole
  (let* ((s (yshift p (* (minus L) (/ (+ s (* 3 m)) (* 2 N)))))
         (rs (/ (distance s (left (intersect (horizontal s) cright)))
                3))
         (sh (circle s rs)))
    (list sh (make-curve q p (list cmain cright ctop))) ))))

(sketch (approximate-lute 12 17 600))
```

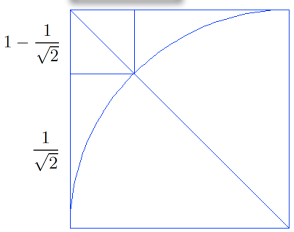


harmonic

m/s= 3/2, 4/3, 7/5, 10/7, 17/12, 24/17...

→ $\sqrt{2}$

(m,s) → (m+2s,m+s)

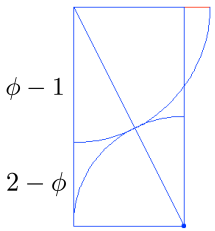


subharmonic

m/s= 3/1, 5/2, 7/3, 12/5, 17/7, 29/12 ...

→ $\sqrt{2}+1$

(m,s) → (2m+s,m)



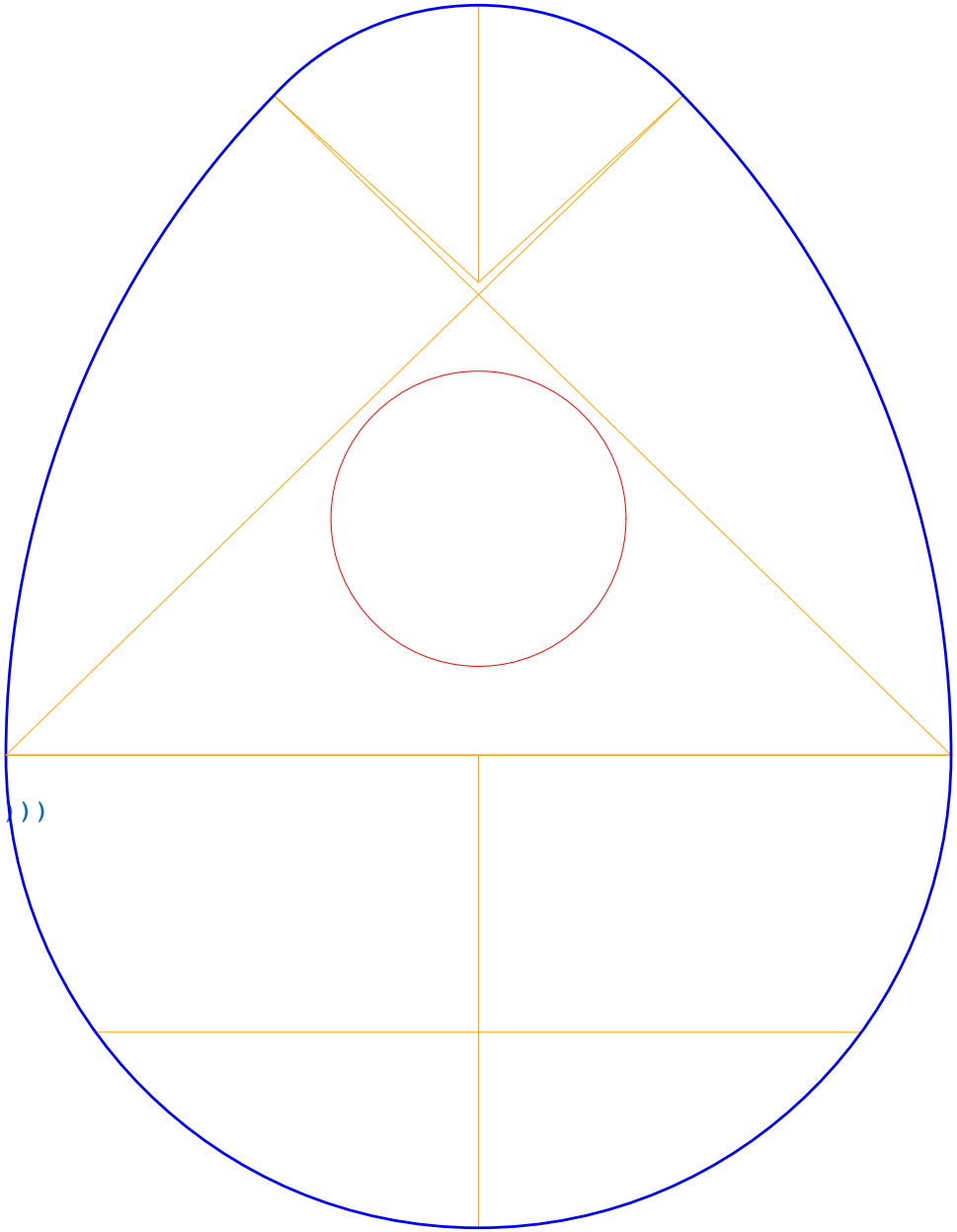
geometric

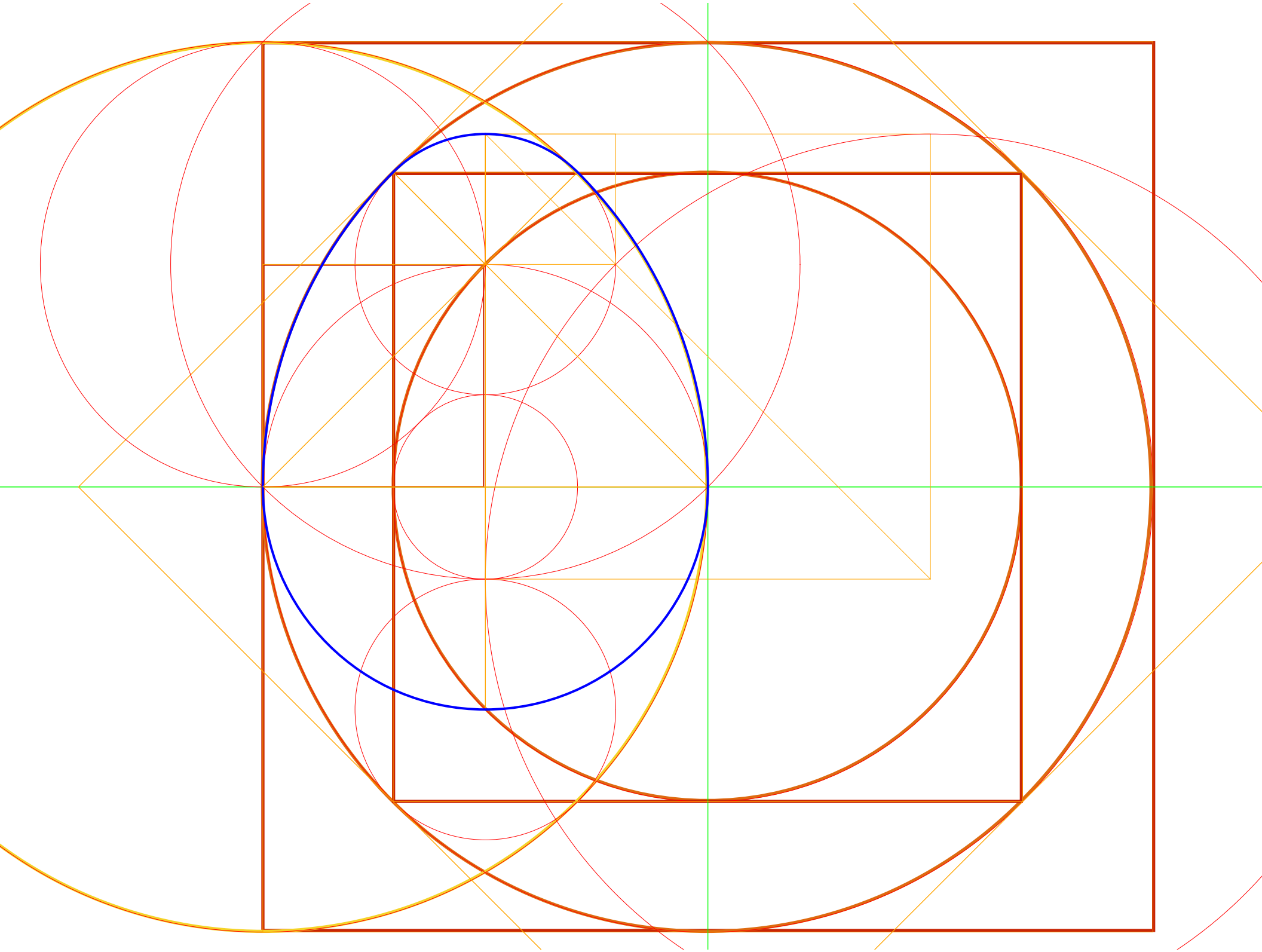
(Fibonacci numbers)

m/s= 3/2, 5/3, 8/5...

→ φ

(m,s) → (m+s,m)





In my programming archives… (and you're free to look, try…)

Code for the proportional constructions in Denis's book of

Violin by Andrea Amati [pp. 114-129]
Violoncello by Josephus (filius Andrea) Guarneri [pp. 182-191]
Violoncello by Domenico Montagnana *(Sleeping Beauty)* [pp. 192-206]

The construction of the code looks "just like" the constructional idiom of Denis's book. *(No new method, just a new delivery…)*

Procedures should abstract over common methods (to be explained…).

It's one thing to follow recipes, yet another to cook…

Code for the proportional constructions of

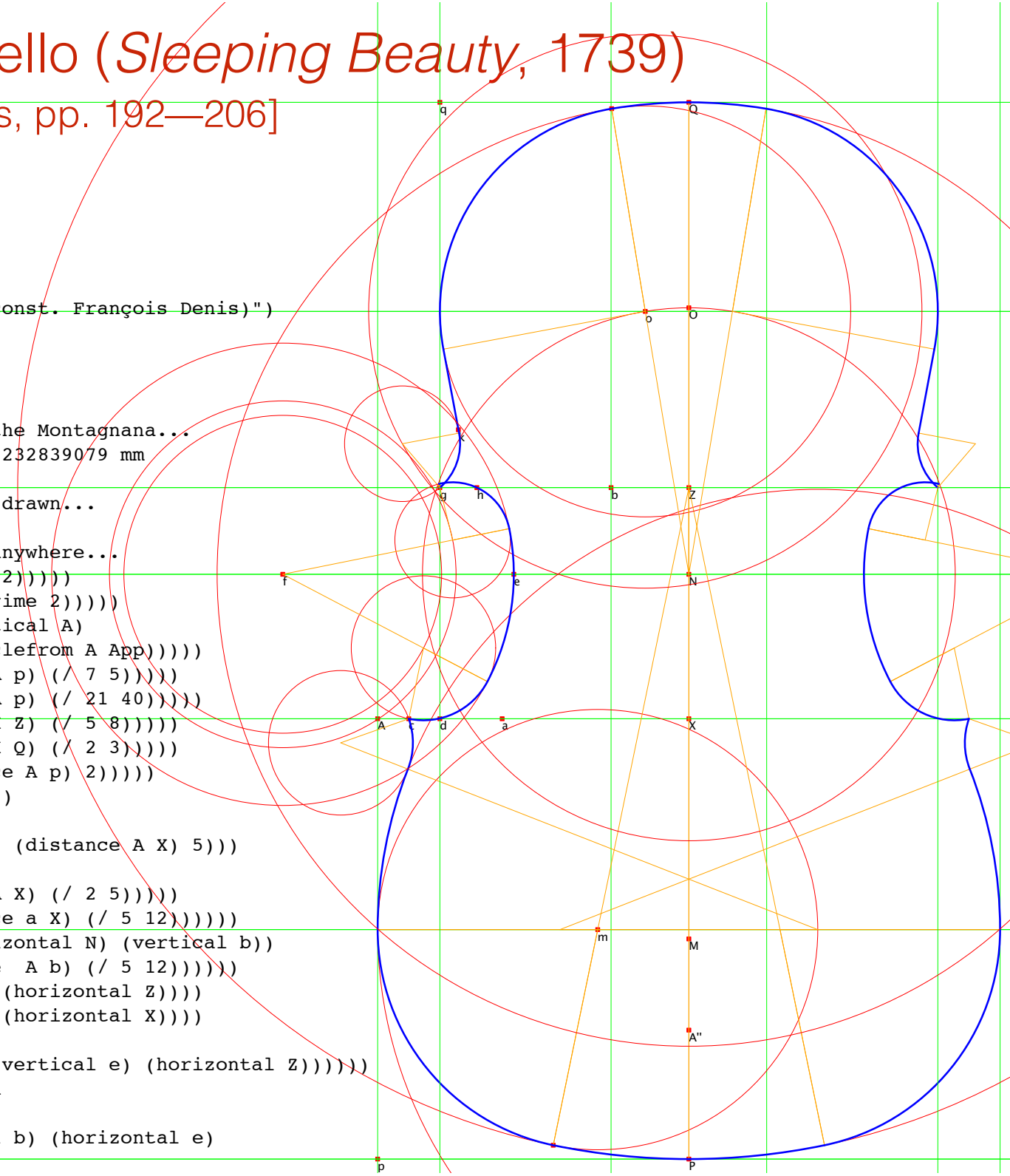
Violoncello by Antonio Stradivari *(Mediceo)*, 1690
Violoncello by Antonio Stradivari *(Cristiani)*, 1700
Violoncello by Antonio Stradivari *(Countess of Stanlein, ex-Paganini)*, 1707

I picked these instruments for a reason (also to be explained…)

Montagnana violoncello (*Sleeping Beauty*, 1739)

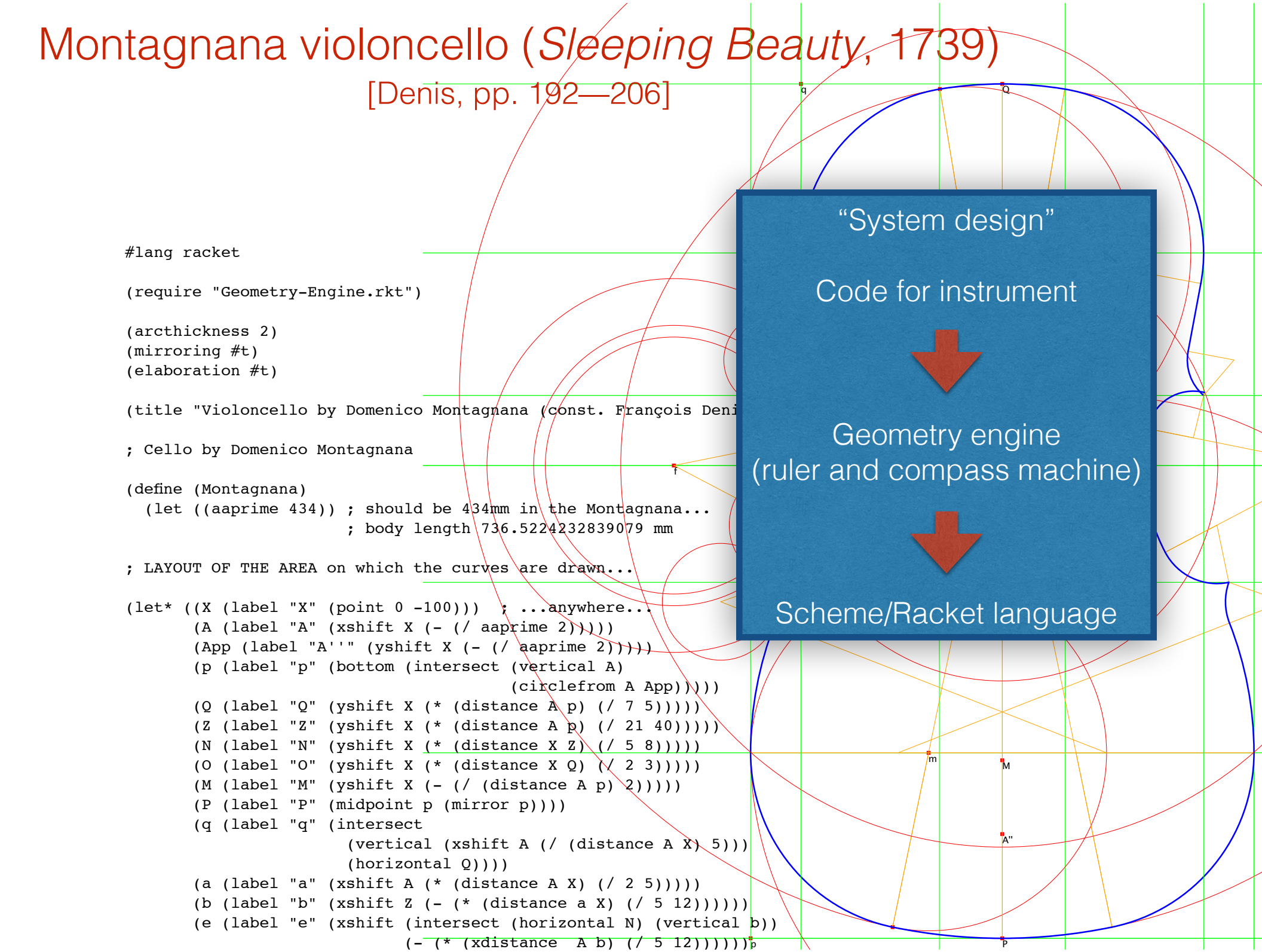
[Denis, pp. 192—206]

```
#lang racket

(require "Geometry-Engine.rkt")

(arcthickness 2)
(mirroring #t)
(elaboration #t)

(title "Violoncello by Domenico Montagnana (const. François Denis)")

; Cello by Domenico Montagnana

(define (Montagnana)
  (let ((aaprime 434)) ; should be 434mm in the Montagnana...
                       ; body length 736.5224232839079 mm

; LAYOUT OF THE AREA on which the curves are drawn...

(let* ((X (label "X" (point 0 -100)))  ; ...anywhere...
       (A (label "A" (xshift X (- (/ aaprime 2)))))
       (App (label "A''" (yshift X (- (/ aaprime 2)))))
       (p (label "p" (bottom (intersect (vertical A)
                                        (circlefrom A App)))))
       (Q (label "Q" (yshift X (* (distance A p) (/ 7 5)))))
       (Z (label "Z" (yshift X (* (distance A p) (/ 21 40)))))
       (N (label "N" (yshift X (* (distance X Z) (/ 5 8)))))
       (O (label "O" (yshift X (* (distance X Q) (/ 2 3)))))
       (M (label "M" (yshift X (- (/ (distance A p) 2)))))
       (P (label "P" (midpoint p (mirror p))))
       (q (label "q" (intersect
                      (vertical (xshift A (/ (distance A X) 5)))
                      (horizontal Q))))
       (a (label "a" (xshift A (* (distance A X) (/ 2 5)))))
       (b (label "b" (xshift Z (- (* (distance a X) (/ 5 12))))))
       (e (label "e" (xshift (intersect (horizontal N) (vertical b))
                             (- (* (xdistance  A b) (/ 5 12))))))
       (g (label "g" (intersect (vertical q) (horizontal Z))))
       (d (label "d" (intersect (vertical q) (horizontal X))))
       (c (label "c" (midpoint A d)))
       (h (label "h" (midpoint g (intersect (vertical e) (horizontal Z))))))
  (list X A App p Q Z N O M P q a b e g d c h
    (vertical p) (vertical b) (horizontal X)
    (horizontal Q) (horizontal P) (horizontal b) (horizontal e)
```



Montagnana violoncello (*Sleeping Beauty*, 1739)

[Denis, pp. 192—206]




```
#lang racket

(require "Geometry-Engine.rkt")

(arcthickness 2)
(mirroring #t)
(elaboration #t)

(title "Violoncello by Domenico Montagnana (const. François Deni

; Cello by Domenico Montagnana

(define (Montagnana)
  (let ((aaprime 434)) ; should be 434mm in the Montagnana...
                       ; body length 736.5224232839079 mm

; LAYOUT OF THE AREA on which the curves are drawn...

(let* ((X (label "X" (point 0 -100)))  ; ...anywhere...
       (A (label "A" (xshift X (- (/ aaprime 2)))))
       (App (label "A''" (yshift X (- (/ aaprime 2)))))
       (p (label "p" (bottom (intersect (vertical A)
                                        (circlefrom A App)))))
       (Q (label "Q" (yshift X (* (distance A p) (/ 7 5)))))
       (Z (label "Z" (yshift X (* (distance A p) (/ 21 40)))))
       (N (label "N" (yshift X (* (distance X Z) (/ 5 8)))))
       (O (label "O" (yshift X (* (distance X Q) (/ 2 3)))))
       (M (label "M" (yshift X (- (/ (distance A p) 2)))))
       (P (label "P" (midpoint p (mirror p))))
       (q (label "q" (intersect
                      (vertical (xshift A (/ (distance A X) 5)))
                      (horizontal Q))))
       (a (label "a" (xshift A (* (distance A X) (/ 2 5)))))
       (b (label "b" (xshift Z (- (* (distance a X) (/ 5 12))))))
       (e (label "e" (xshift (intersect (horizontal N) (vertical b))
                             (- (* (xdistance  A b) (/ 5 12))))))p
```

“System design”

Code for instrument

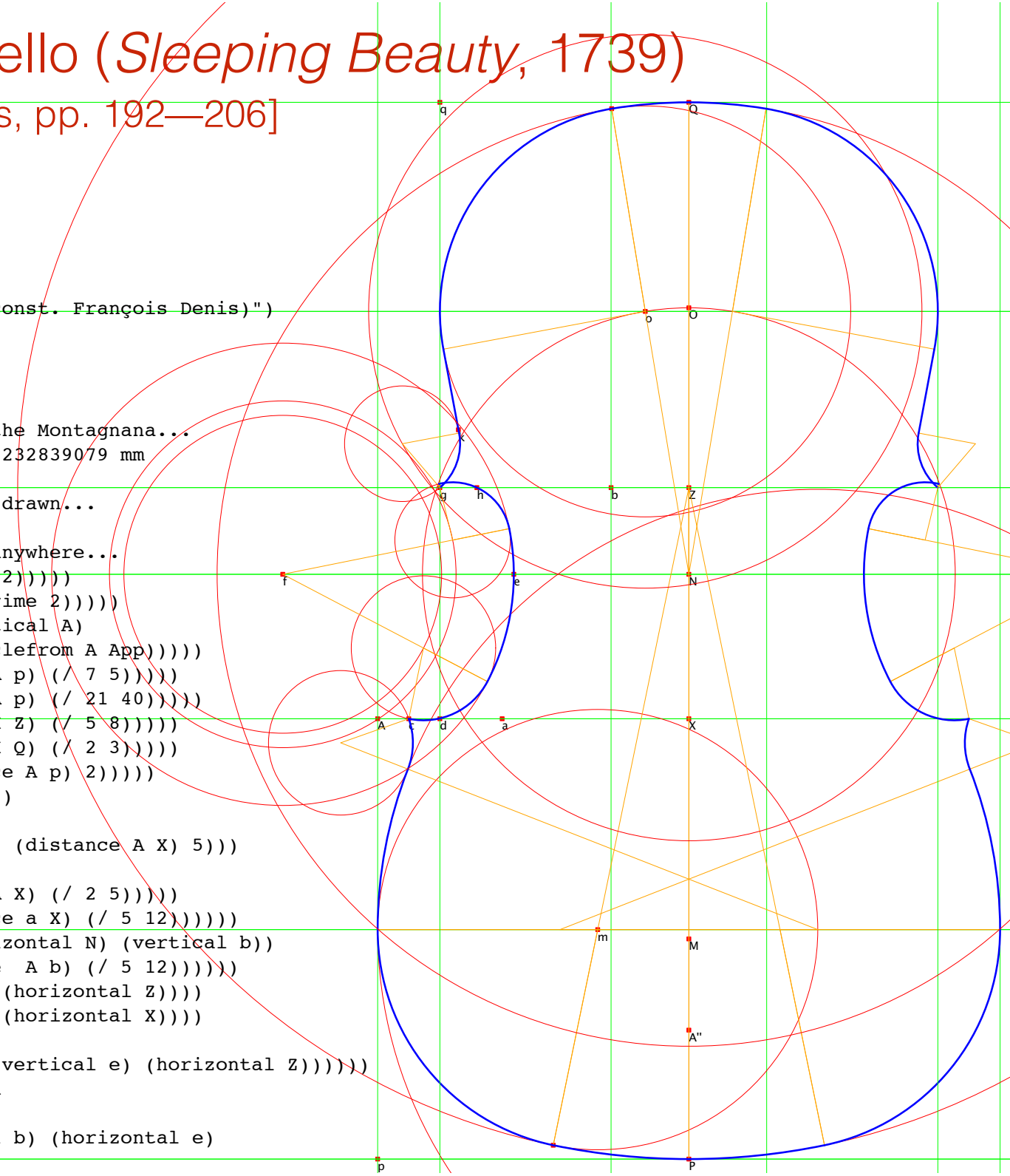
Geometry engine
(ruler and compass machine)

Scheme/Racket language

Montagnana violoncello (*Sleeping Beauty*, 1739)

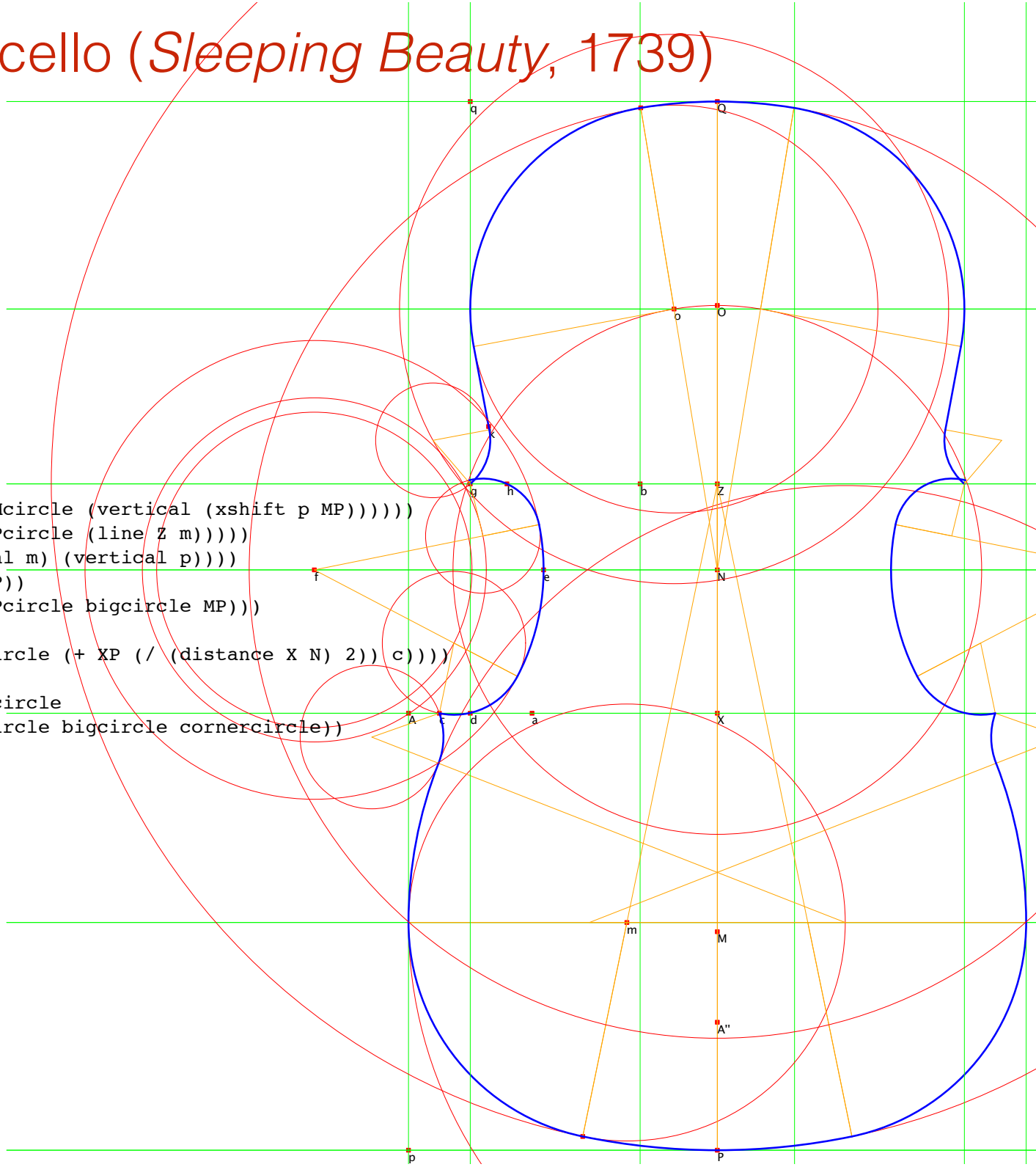
[Denis, pp. 192—206]

```
#lang racket

(require "Geometry-Engine.rkt")

(arcthickness 2)
(mirroring #t)
(elaboration #t)

(title "Violoncello by Domenico Montagnana (const. François Denis)")

; Cello by Domenico Montagnana

(define (Montagnana)
  (let ((aaprime 434)) ; should be 434mm in the Montagnana...
                       ; body length 736.5224232839079 mm

; LAYOUT OF THE AREA on which the curves are drawn...

(let* ((X (label "X" (point 0 -100)))  ; ...anywhere...
       (A (label "A" (xshift X (- (/ aaprime 2)))))
       (App (label "A''" (yshift X (- (/ aaprime 2)))))
       (p (label "p" (bottom (intersect (vertical A)
                                        (circlefrom A App)))))
       (Q (label "Q" (yshift X (* (distance A p) (/ 7 5)))))
       (Z (label "Z" (yshift X (* (distance A p) (/ 21 40)))))
       (N (label "N" (yshift X (* (distance X Z) (/ 5 8)))))
       (O (label "O" (yshift X (* (distance X Q) (/ 2 3)))))
       (M (label "M" (yshift X (- (/ (distance A p) 2)))))
       (P (label "P" (midpoint p (mirror p))))
       (q (label "q" (intersect
                      (vertical (xshift A (/ (distance A X) 5)))
                      (horizontal Q))))
       (a (label "a" (xshift A (* (distance A X) (/ 2 5)))))
       (b (label "b" (xshift Z (- (* (distance a X) (/ 5 12))))))
       (e (label "e" (xshift (intersect (horizontal N) (vertical b))
                             (- (* (xdistance  A b) (/ 5 12))))))
       (g (label "g" (intersect (vertical q) (horizontal Z))))
       (d (label "d" (intersect (vertical q) (horizontal X))))
       (c (label "c" (midpoint A d)))
       (h (label "h" (midpoint g (intersect (vertical e) (horizontal Z))))))
  (list X A App p Q Z N O M P q a b e g d c h
    (vertical p) (vertical b) (horizontal X)
    (horizontal Q) (horizontal P) (horizontal b) (horizontal e)
```



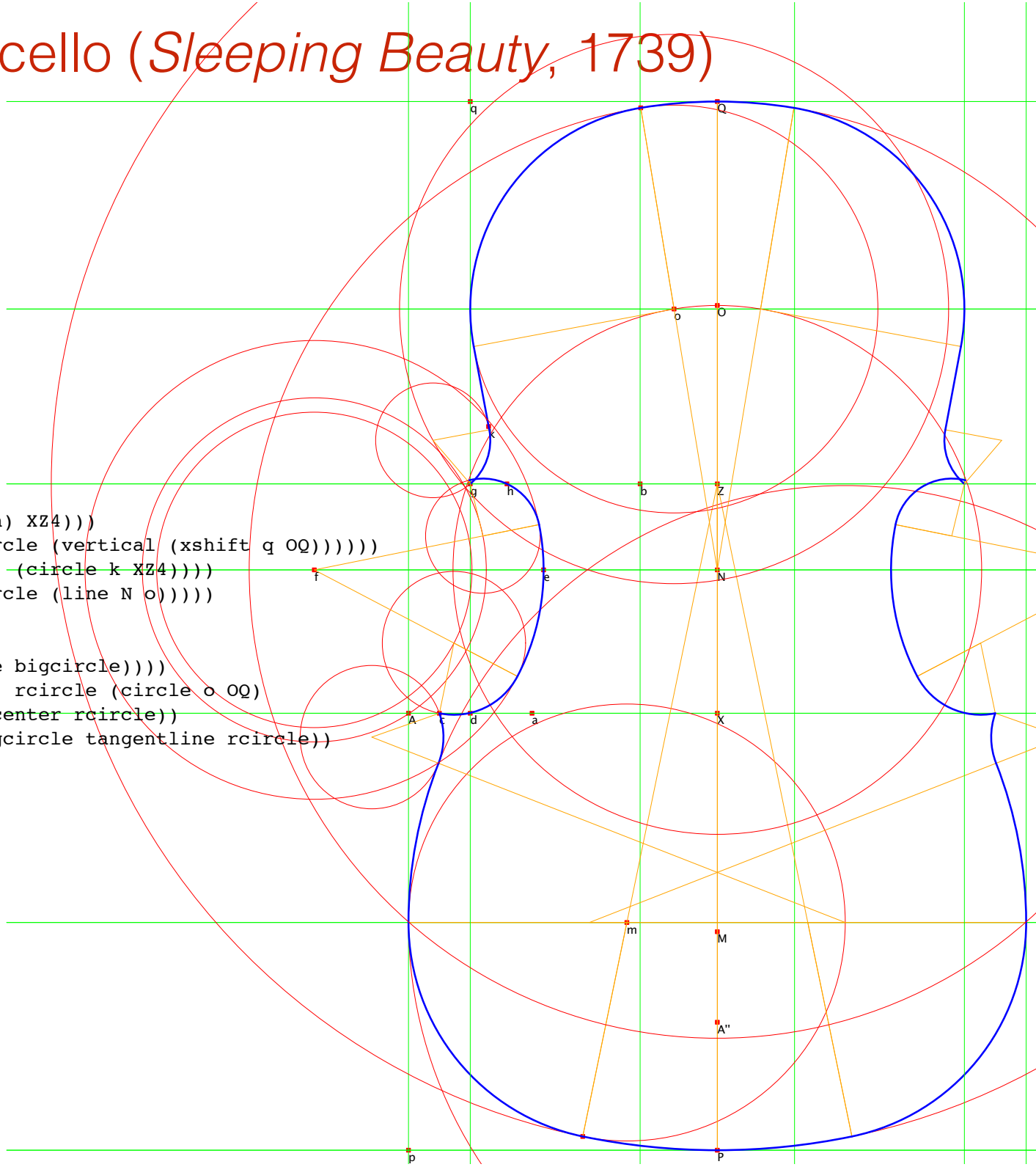
Montagnana violoncello (*Sleeping Beauty*, 1739)

```
; Lower bouts

(let* ((ZPcircle (circlefrom Z P))
       (ZMcircle (circlefrom Z M))
       (XP (distance X P))
       (MP (distance M P))
       (m (label "m" (bottom (intersect ZMcircle (vertical (xshift p MP))))))
       (m2 (label "" (bottom (intersect ZPcircle (line Z m)))))
       (u (label "u" (intersect (horizontal m) (vertical p))))
       (bigcircle (circle (xshift u XP) XP))
       (mcircle (lower-circle (inscribe ZPcircle bigcircle MP)))
       (cornercircle
        (lower-circle (reverse-curve bigcircle (+ XP (/ (distance X N) 2)) c))))
  (list m m2 (horizontal m) ; u
        ZPcircle mcircle bigcircle cornercircle
        (make-curve P c (list ZPcircle mcircle bigcircle cornercircle))
```



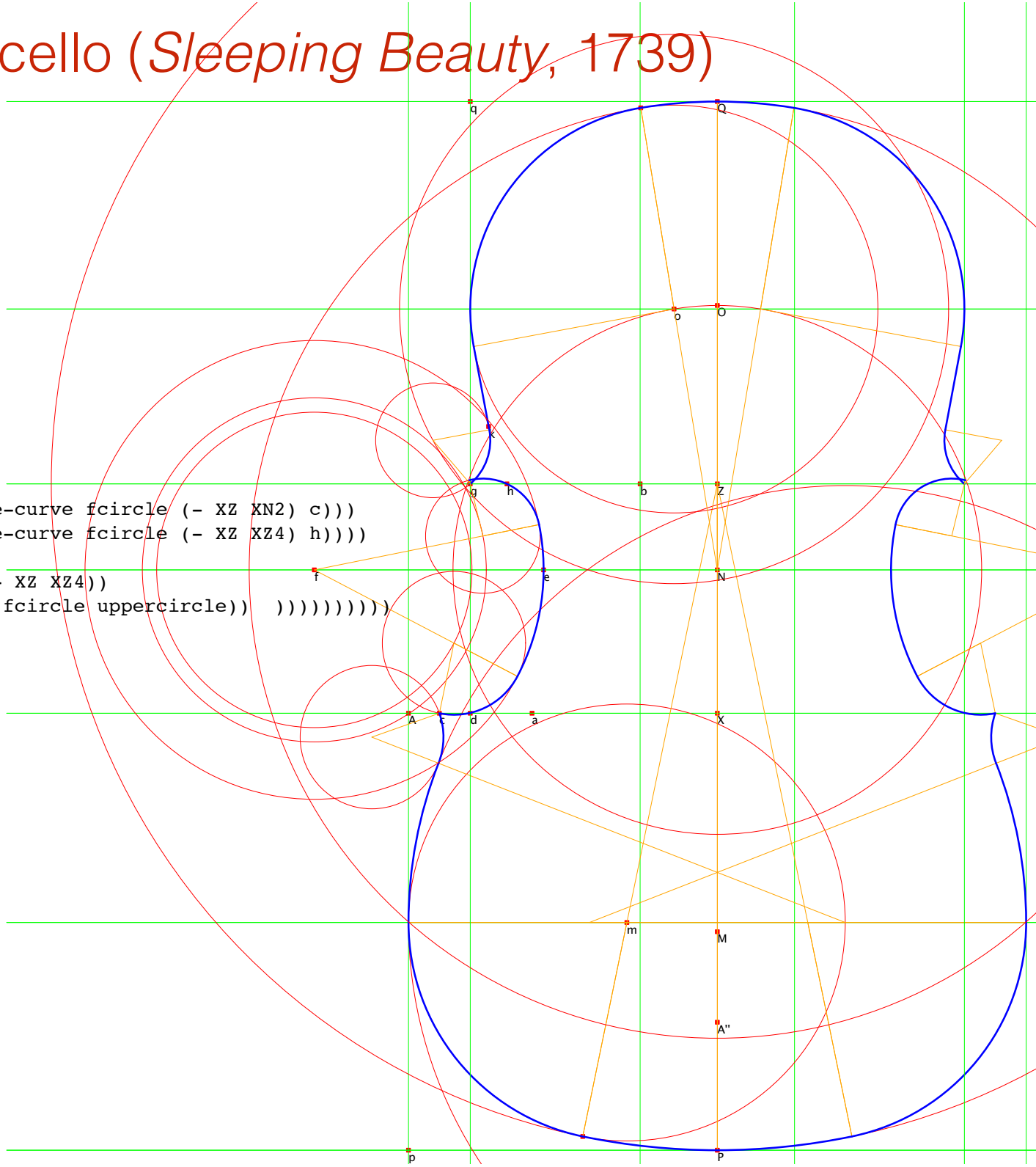
Montagnana violoncello (*Sleeping Beauty*, 1739)

```
; Upper bouts

(let* ((NQcircle (circlefrom N Q))
       (NOcircle (circlefrom N O))
       (OQ (distance O Q))
       (XZ (distance X Z))
       (XZ4 (/ (distance X Z) 4))
       (k (label "k" (yshift (midpoint g h) XZ4)))
       (o (label "o" (top (intersect NOcircle (vertical (xshift q OQ))))))
       (rc (left (intersect (circle g XZ4) (circle k XZ4))))
       (o2 (label "" (top (intersect NQcircle (line N o)))))
       (rcircle (circle rc XZ4))
       (bigcircle (circle o OQ))
       (tangentline (first (tangent rcircle bigcircle))))
  (list k o o2 (horizontal o) (vertical q) rcircle (circle o OQ)
        NQcircle NOcircle (circlefrom o (center rcircle))
        (make-curve Q g (list NQcircle bigcircle tangentline rcircle))
```



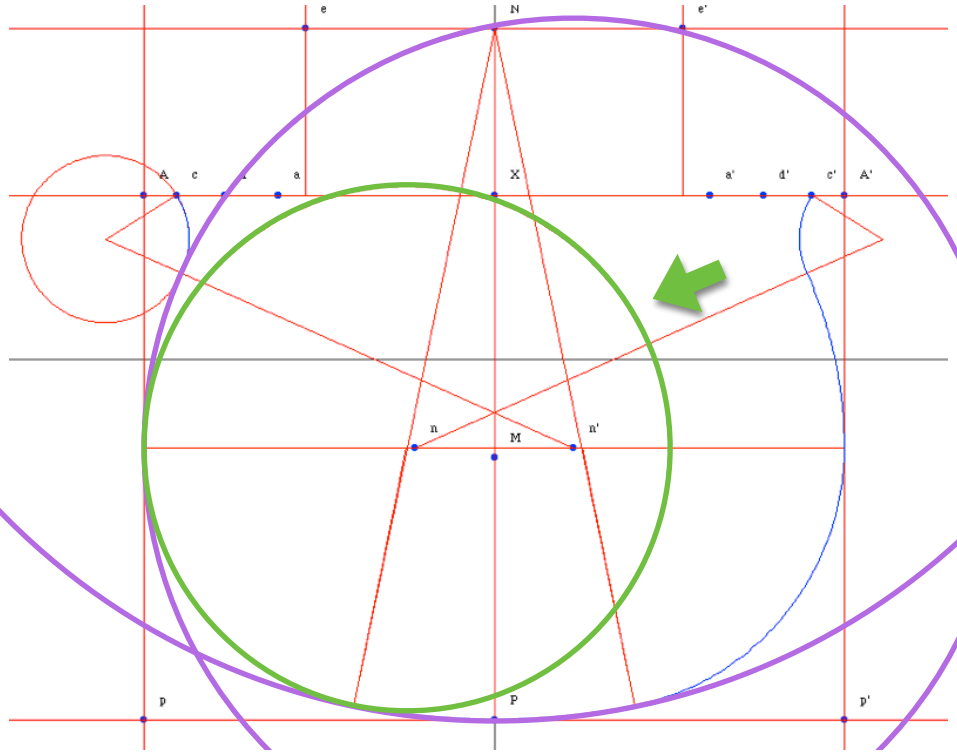
Montagnana violoncello (*Sleeping Beauty*, 1739)

```
; Middle bouts

(let* ((f (label "f" (xshift e (- XZ))))
       (fcircle (circle f XZ))
       (XN2 (/ (distance X N) 2))
       (lowercircle (upper-circle (reverse-curve fcircle (- XZ XN2) c)))
       (uppercircle (lower-circle (reverse-curve fcircle (- XZ XZ4) h))))
  (list f lowercircle uppercircle fcircle
        (circle f (- XZ XN2)) (circle f (- XZ XZ4))
        (make-curve c g (list lowercircle fcircle uppercircle))  ))))))))))

(sketch (Montagnana))

(end-drawing)
```



Montagnana violoncello

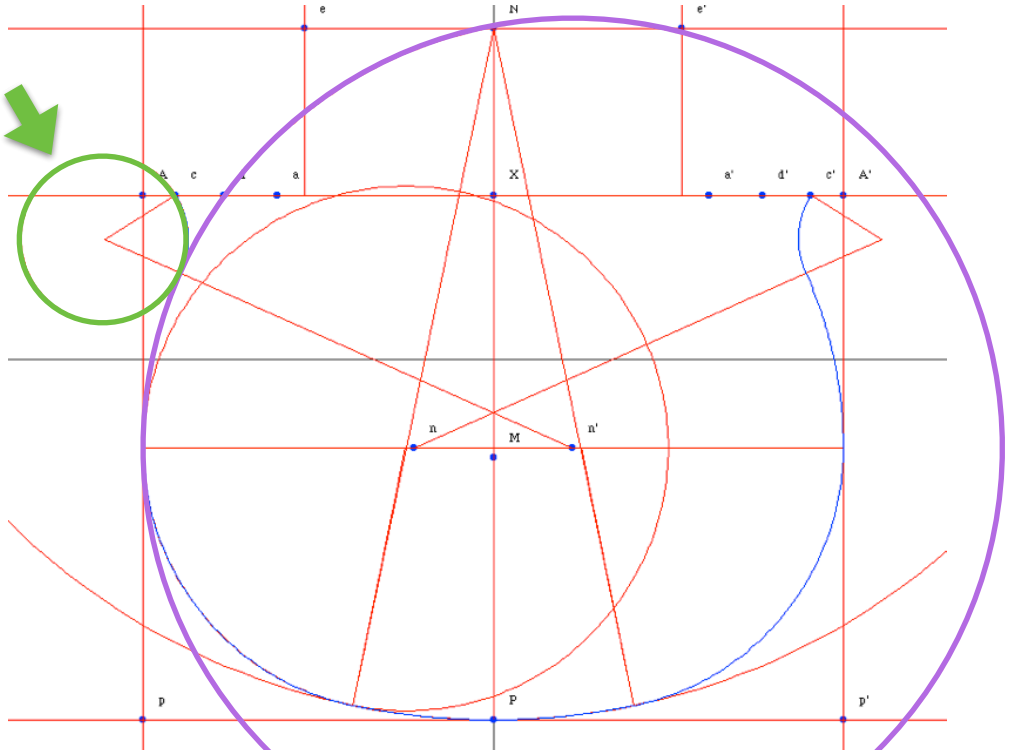
```
; Lower bouts

(let* ((ZPcircle (circlefrom Z P))
       (ZMcircle (circlefrom Z M))
       (XP (distance X P))
       (MP (distance M P))
       (m (label "m" (bottom (intersect ZMcircle (vertical (xshift p MP))))))
       (m2 (label "" (bottom (intersect ZPcircle (line Z m)))))
       (u (label "u" (intersect (horizontal m) (vertical p))))
       (bigcircle (circle (xshift u XP) XP))
       (mcircle (lower-circle (inscribe ZPcircle bigcircle MP)))
       (cornercircle
        (lower-circle (reverse-curve bigcircle (+ XP (/ (distance X N) 2)) c))))
  (list m m2 (horizontal m) ; u
        ZPcircle mcircle bigcircle cornercircle
        (make-curve P c (list ZPcircle mcircle bigcircle cornercircle))
```



Two geometric problems:
inscribed circle
reverse curve

Montagnana violoncello

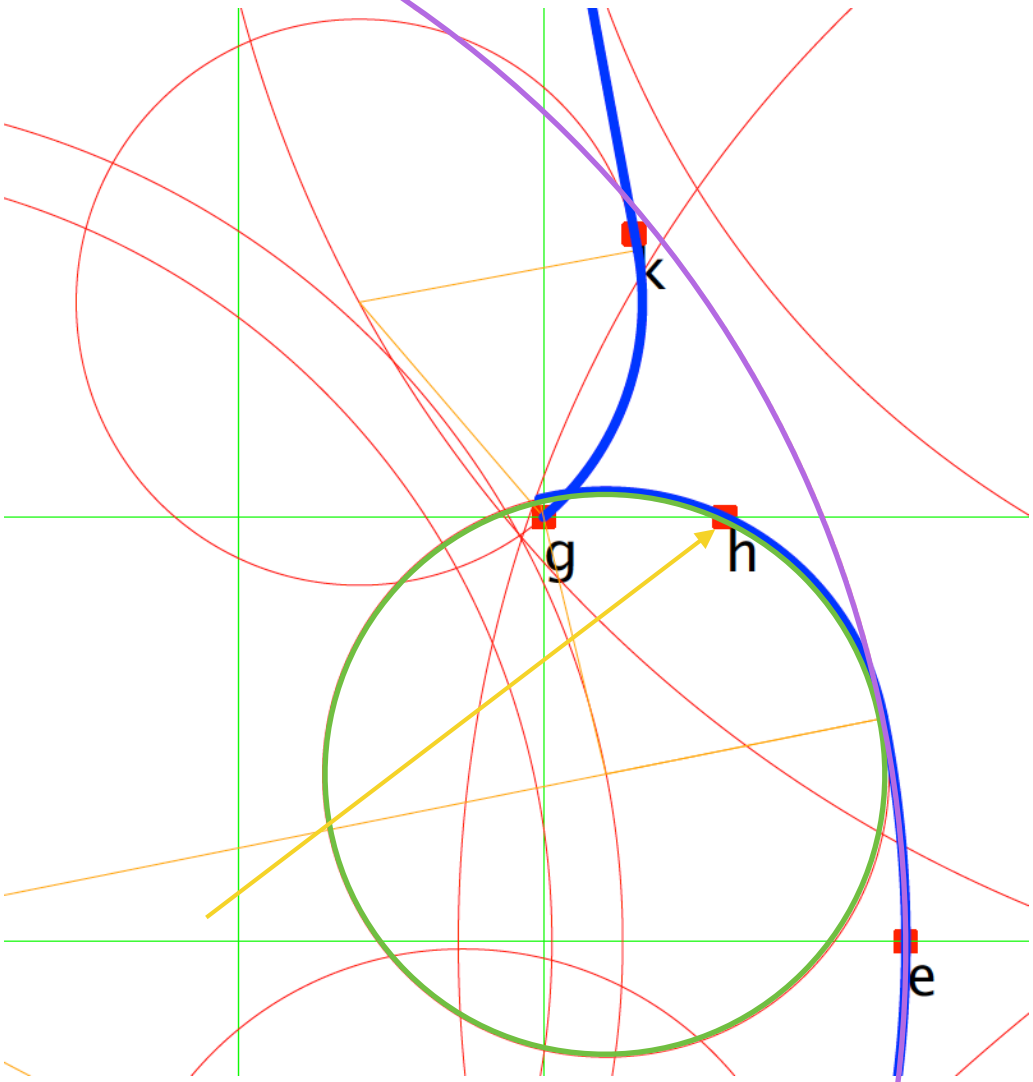
```
; Lower bouts

(let* ((ZPcircle (circlefrom Z P))
       (ZMcircle (circlefrom Z M))
       (XP (distance X P))
       (MP (distance M P))
       (m (label "m" (bottom (intersect ZMcircle (vertical (xshift p MP))))))
       (m2 (label "" (bottom (intersect ZPcircle (line Z m)))))
       (u (label "u" (intersect (horizontal m) (vertical p))))
       (bigcircle (circle (xshift u XP) XP))
       (mcircle (lower-circle (inscribe ZPcircle bigcircle MP)))
       (cornercircle
        (lower-circle (reverse-curve bigcircle (+ XP (/ (distance X N) 2)) c))))
  (list m m2 (horizontal m) ; u
        ZPcircle mcircle bigcircle cornercircle
        (make-curve P c (list ZPcircle mcircle bigcircle cornercircle))
```

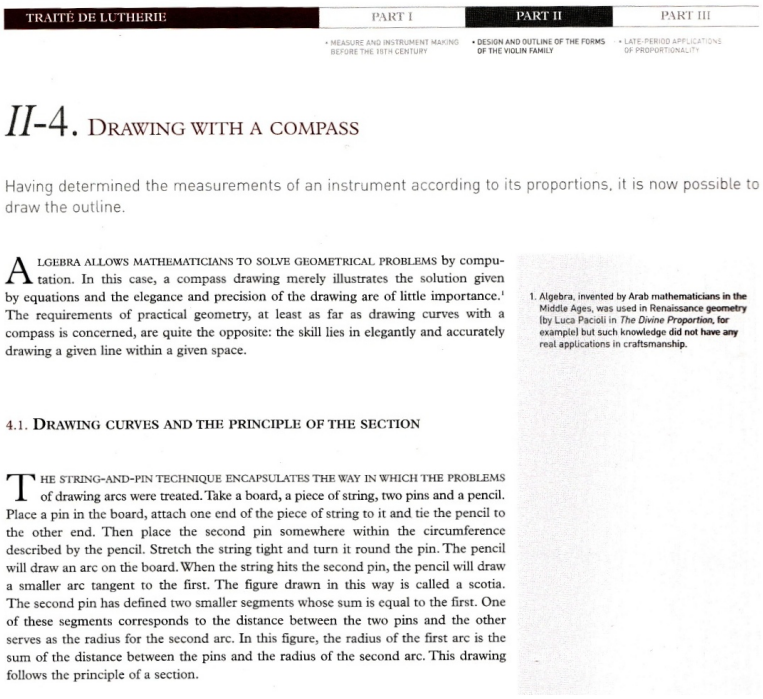


Two geometric problems:
inscribed circle
reverse curve

Reverse curves, inscribed circles: higher-order procedures of instrument design

```
(define (reverse-curve inner-circle outer pt)
  (let* ((outer-radius (- outer (radius inner-circle)))
         (pts (intersect
               (circle pt (abs outer-radius))
               (circle (center inner-circle)
                       (+ outer-radius
                          (radius inner-circle))))))
    (map (lambda (p)
           (circle p (abs outer-radius)))
         pts)))

(define (inscribe c1 c2 r)
  ; circles tangent to insides of c1, c2 of radius r
  (let ((c1p (circle (center c1) (- (radius c1) r)))
        (c2p (circle (center c2) (- (radius c2) r))))
    (map (lambda (pt) (circle pt r))
         (intersect c1p c2p))))
```





TRAITÉ DE LUTHERIE | PART I | PART II | PART III

• MEASURE AND INSTRUMENT MAKING BEFORE THE 18TH CENTURY • DESIGN AND OUTLINE OF THE FORMS OF THE VIOLIN FAMILY • LATE-PERIOD APPLICATIONS OF PROPORTIONALITY

II-4. Drawing with a compass

Having determined the measurements of an instrument according to its proportions, it is now possible to draw the outline.

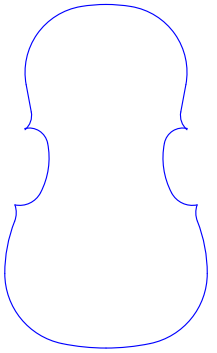
Algebra allows mathematicians to solve geometrical problems by computation. In this case, a compass drawing merely illustrates the solution given by equations and the elegance and precision of the drawing are of little importance.[1] The requirements of practical geometry, at least as far as drawing curves with a compass is concerned, are quite the opposite: the skill lies in elegantly and accurately drawing a given line within a given space.

1. Algebra, invented by Arab mathematicians in the Middle Ages, was used in Renaissance geometry (by Luca Pacioli in *The Divine Proportion*, for example) but such knowledge did not have any real applications in craftsmanship.

4.1. Drawing curves and the principle of the section

The string-and-pin technique encapsulates the way in which the problems of drawing arcs were treated. Take a board, a piece of string, two pins and a pencil. Place a pin in the board, attach one end of the piece of string to it and tie the pencil to the other end. Then place the second pin somewhere within the circumference described by the pencil. Stretch the string tight and turn it round the pin. The pencil will draw an arc on the board. When the string hits the second pin, the pencil will draw a smaller arc tangent to the first. The figure drawn in this way is called a scotia. The second pin has defined two smaller segments whose sum is equal to the first. One of these segments corresponds to the distance between the two pins and the other serves as the radius for the second arc. In this figure, the radius of the first arc is the sum of the distance between the pins and the radius of the second arc. This drawing follows the principle of a section.

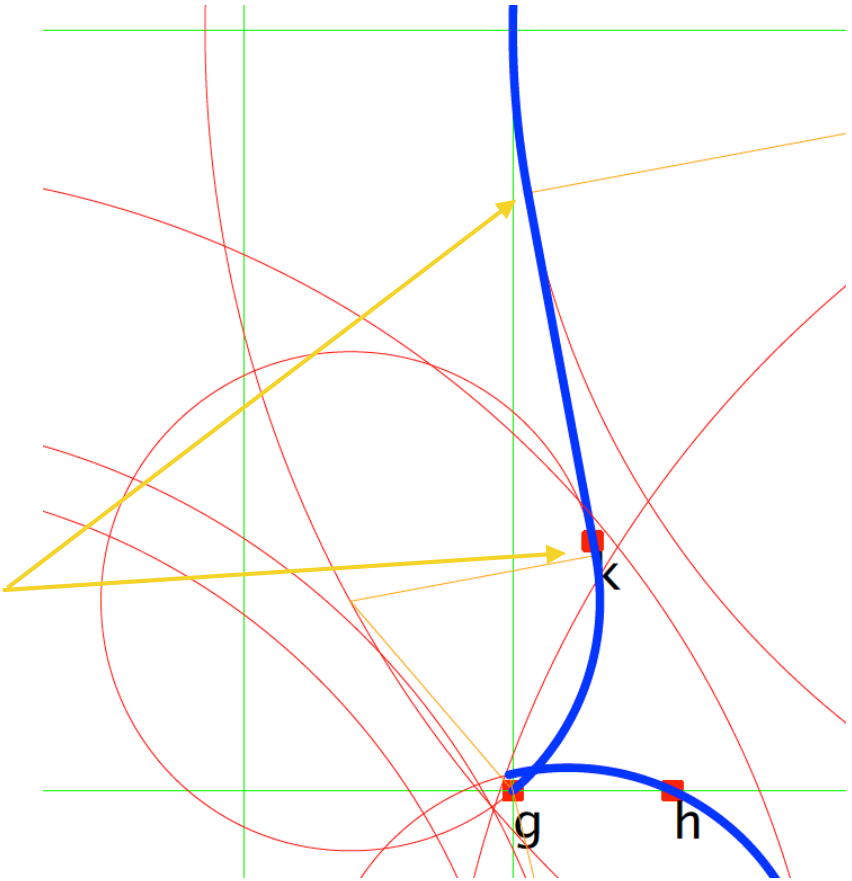
Tangent lines between arcs (example: Montagnana cello)



```
(let* ((tangentline
         (first (tangent rcircle
                         circle o (distance O Q))))))
       (uppertangentpoint
         (intersect tangentline
                    (circle o (distance O Q))))))
```

…and buried in the code for the geometry engine…

```
(define (tangent small big)
  ; finds the two lines tangent to both c1 and c2,
    assuming c1, c2 do not intersect...
  (let* ((d (line (center small) (center big)))
         (p1 (perpendicular d (center small)))
         (p2 (perpendicular d (center big)))
         (a (left (intersect small p1)))
         (b (right (intersect big p2)))
         (i (intersect d (line a b)))
         (j (midpoint i (center big)))
         (pts (intersect big (circle j (distance i j)))))
    (map (lambda (pt) (line i pt)) pts)))
```



...standard constructions (Euclidean geometry is an old subject!), described by code, in the vernacular of string instrument design...

What is any of this good for?

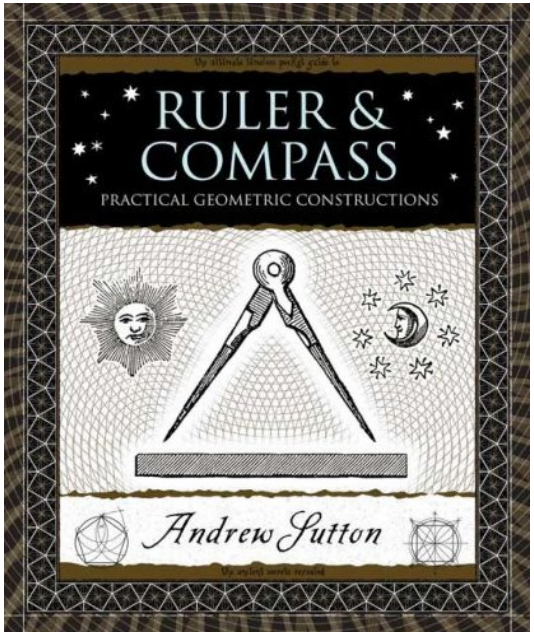
Teaching instrument design, math,
or programming---*without numbers*...

Online repository of famous *designs* (not plans)

Design modification by program alteration
(likely proprietary, unlikely to land in repository!)

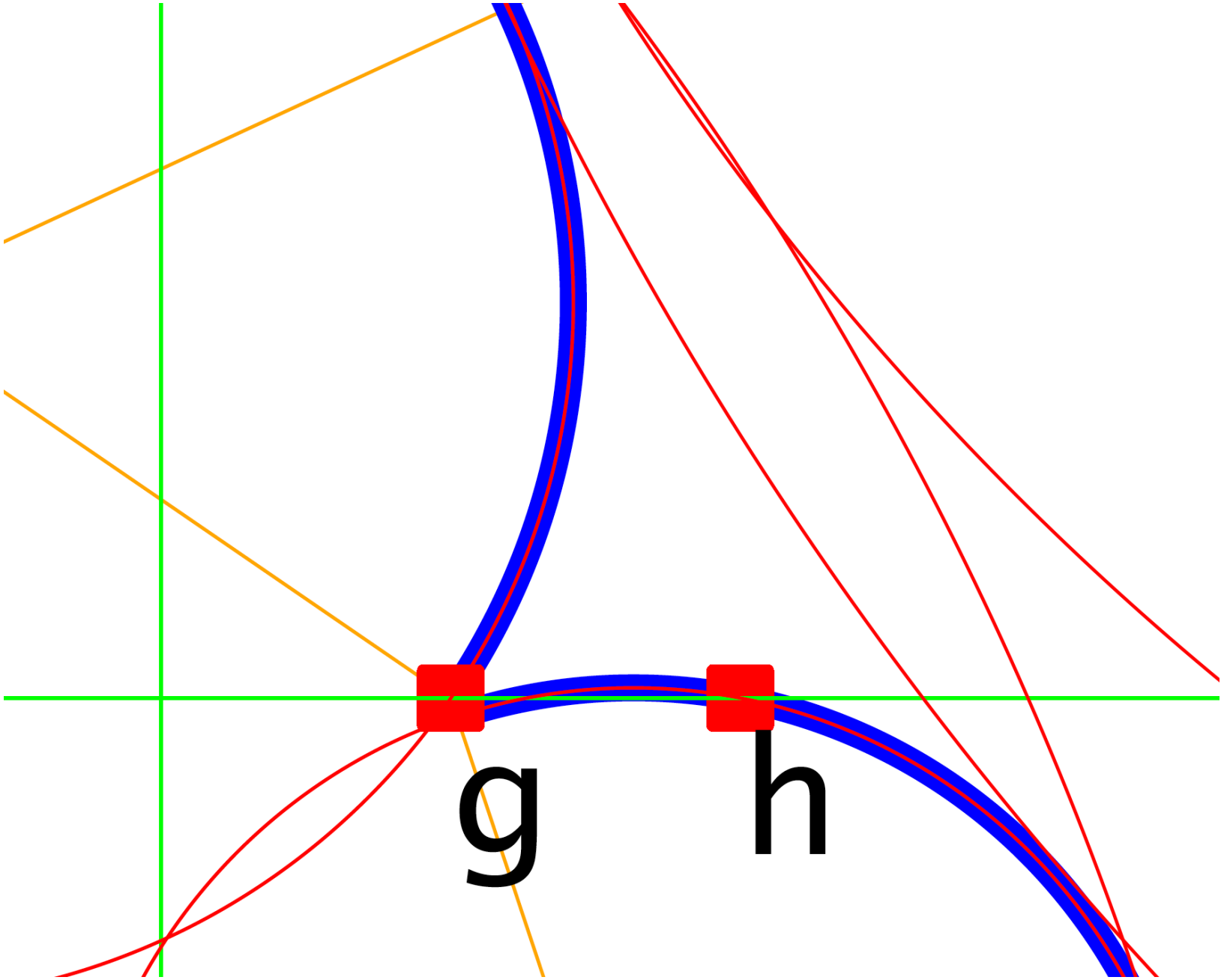
Construction prosthesis (PDF output of forms for constructing molds,
or to drive a fabricator)

Computational art history (study design by geometric construction
method, as well as form): identification of maker, evolution of models
by same or related luthiers...



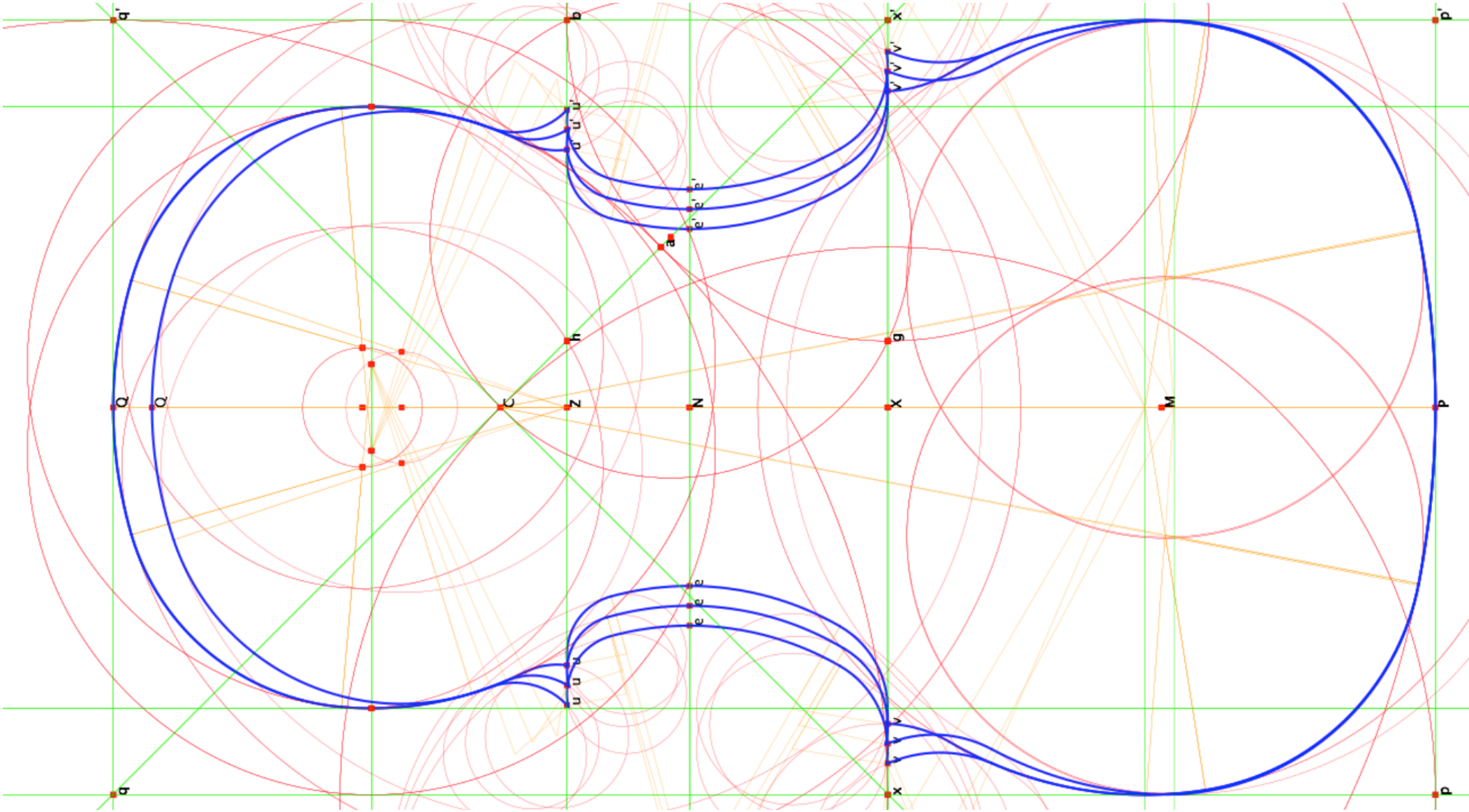
Fabricating a mold

The beauty of PDF and scalable vector graphics…
(printing out accurate, exact-size drawings is easy)



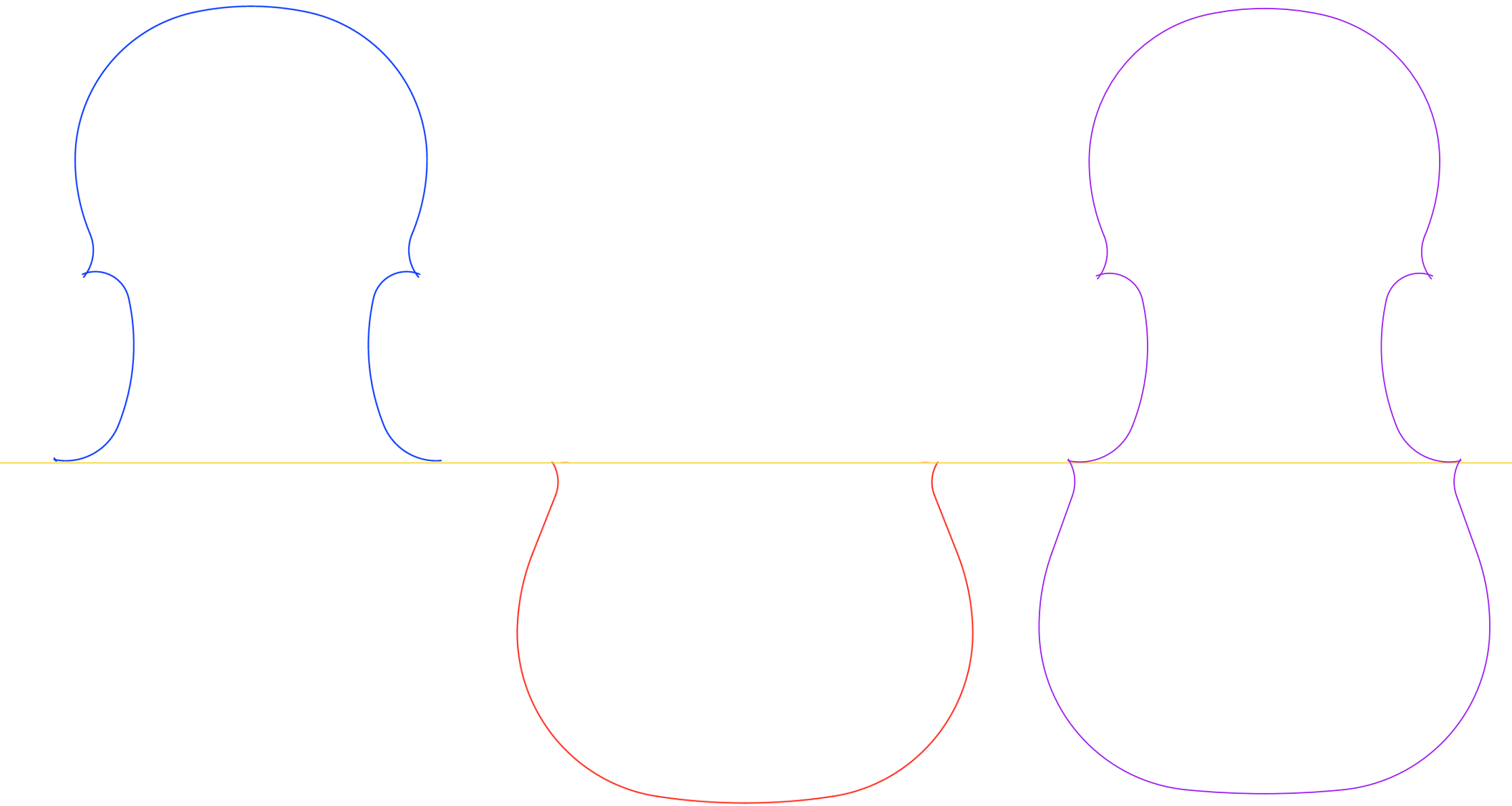
Design modification

Modifying e, e' by -10%, +10%, Q by 5%



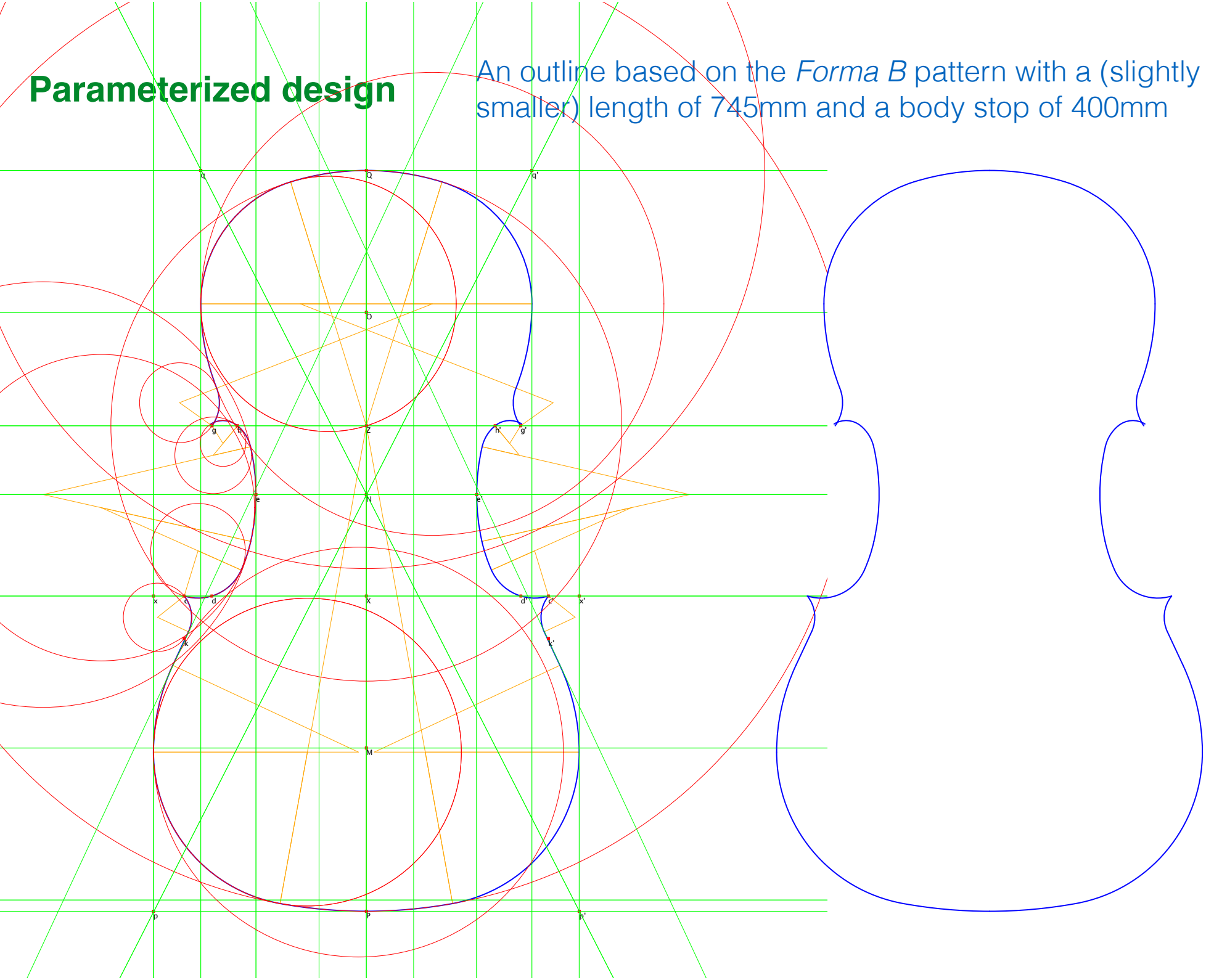
Composite design

Mediceo (in upper two bouts) and *Stanlein* (lower bout)
(scaled, of course)



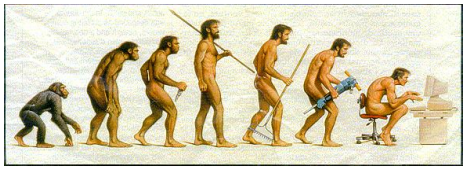
Parameterized design

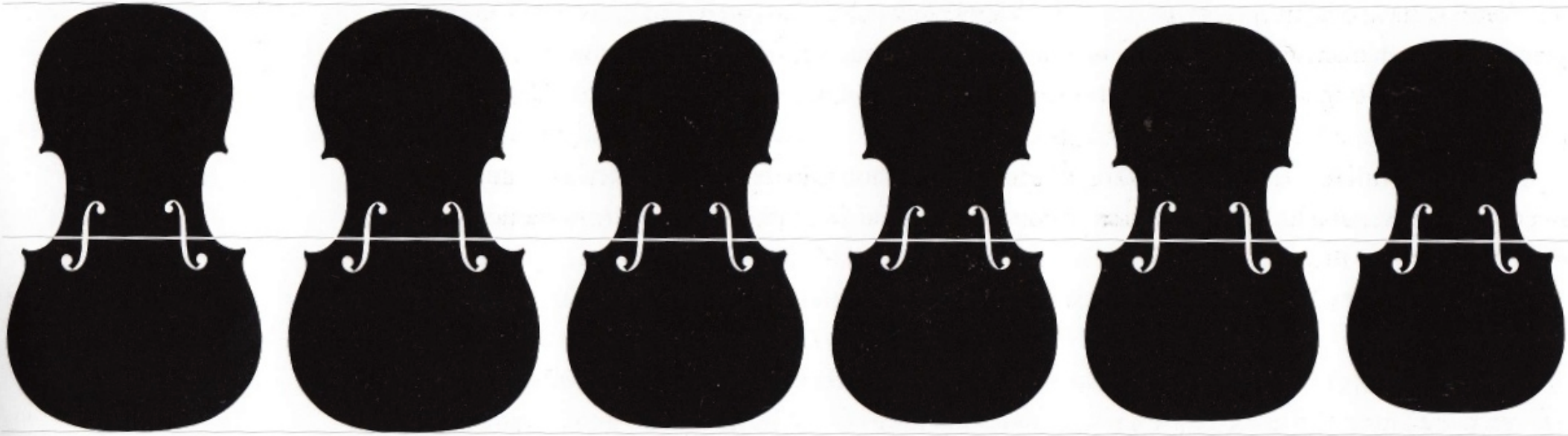
An outline based on the *Forma B* pattern with a (slightly smaller) length of 745mm and a body stop of 400mm



Now… could we use a tool like this to search for something…?

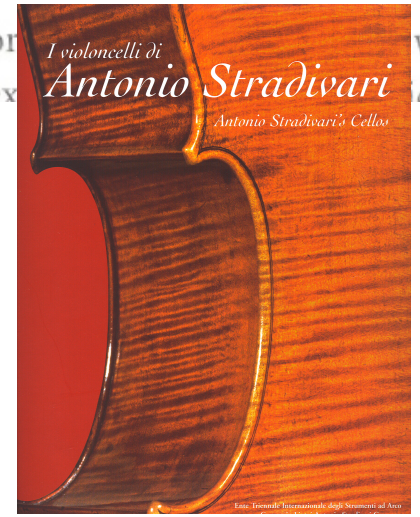
Where did the *forma B* come from?
What is the relevance of proportional,geometric methods?





"Mediceo" 1690 "Cristiani" 1700 "Bass of Spain" 1713 "De Munck" 1730 circa "Pleeth" 1732 circa "Josefowitz" 1732 circa

in 1707, or when he introduced the model considered by so many violin-
makers, ex icians of the past and present to be the ideal cello. Stradivari

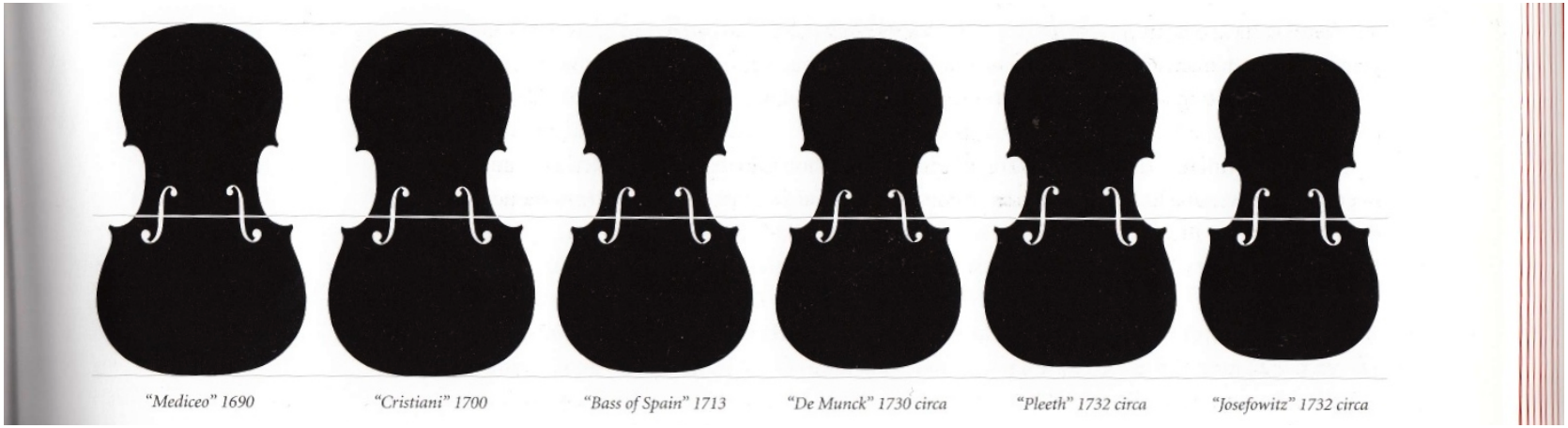


from Bruce Carlson,
The Evolution of the Stradivari Violoncello
2004

Where did the *forma B* come from?
What is the relevance of geometric methods?

"[I]nvention… often results in a loss of know-how." (Denis, p. 23)

"…throughout the century or so of transition, the new concept of measurement was superposed on the former proportional methods, which gradually died out." (p. 207)



Where did the *forma B* come from?
What is the relevance of geometric methods?

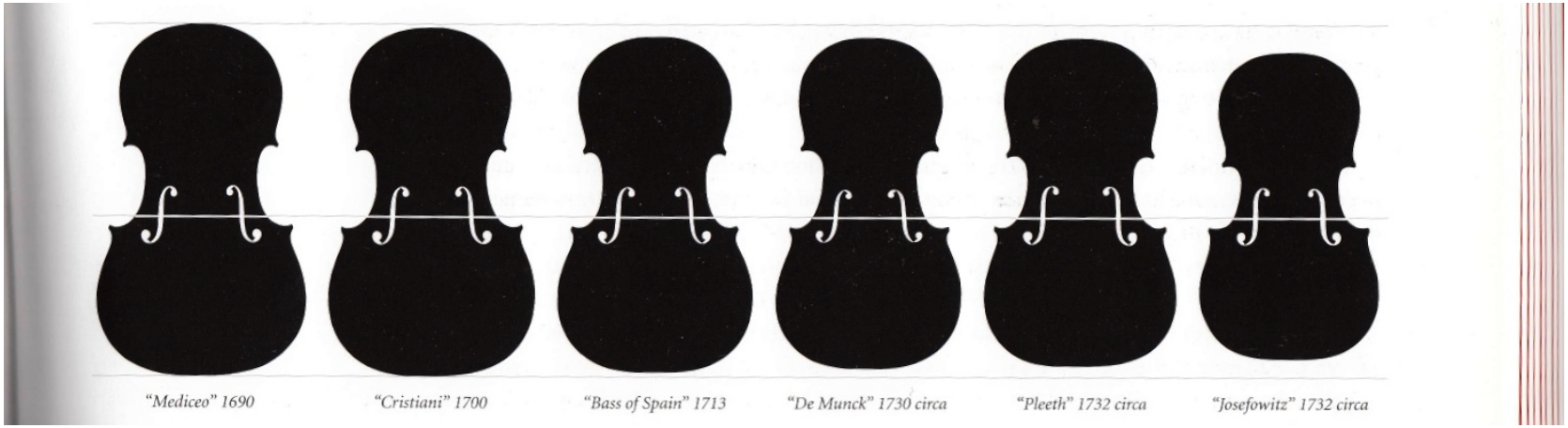
III-2. Stradivari's Inheritance and Particularities

Close scrutiny of Stradivari's work shows a break with tradition alongside the master's classicism.

Why can't you draw a forma B violoncello proportionally, just like Denis drew an Amati violin, or a Montagnana (1739) cello? I wanted to answer this question *in my own terms.*

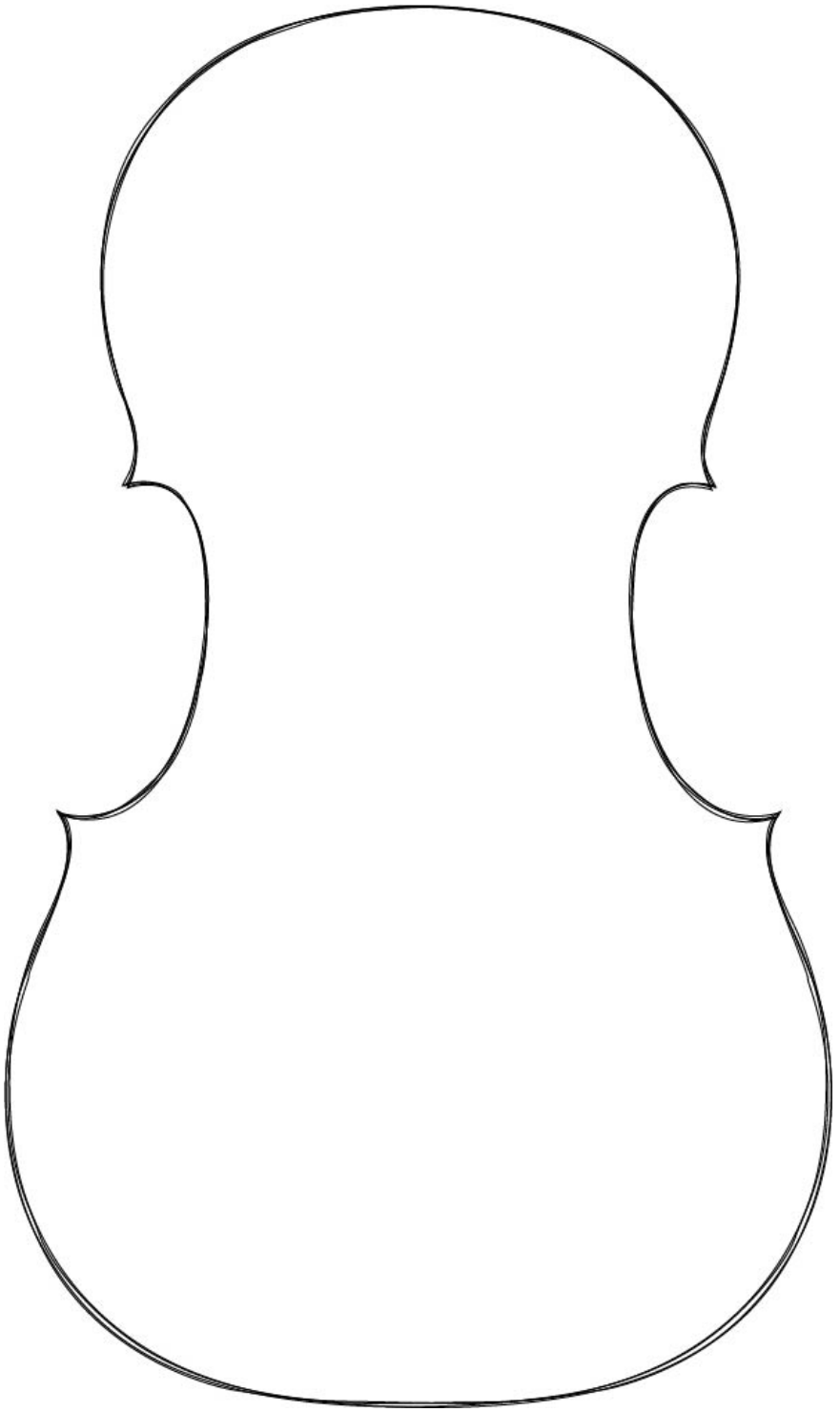
And my answer is: *try and do it, and see how far you get…*

First: try drawing a more Amatisé model (Cristiani, Mediceo) using entirely proportional methods. How difficult is it?



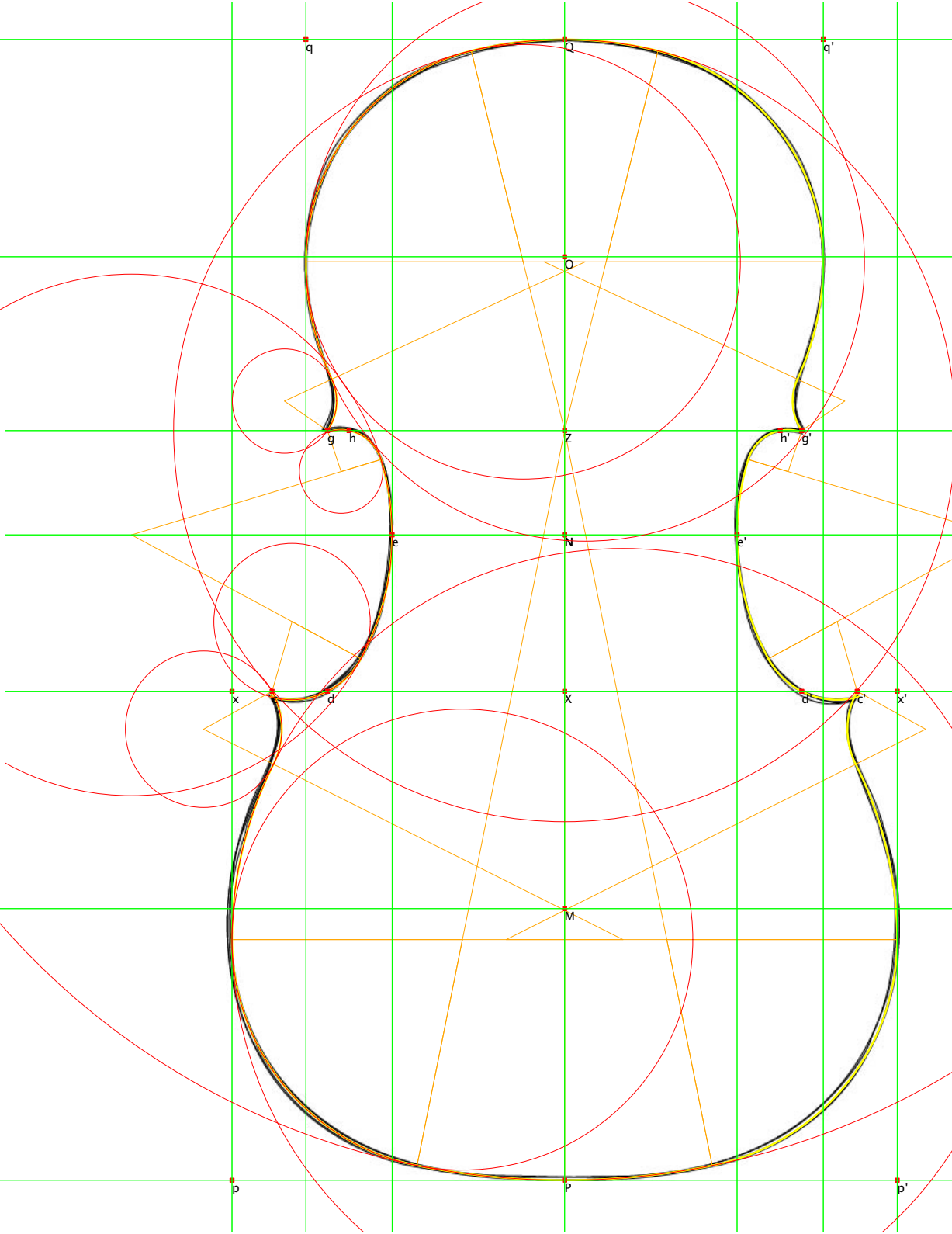
Stradivari violoncello (*Cristiani*, 1700)

tracings taken “by hand” along purfling lines on front and back, reflected, superimposed (4 images)



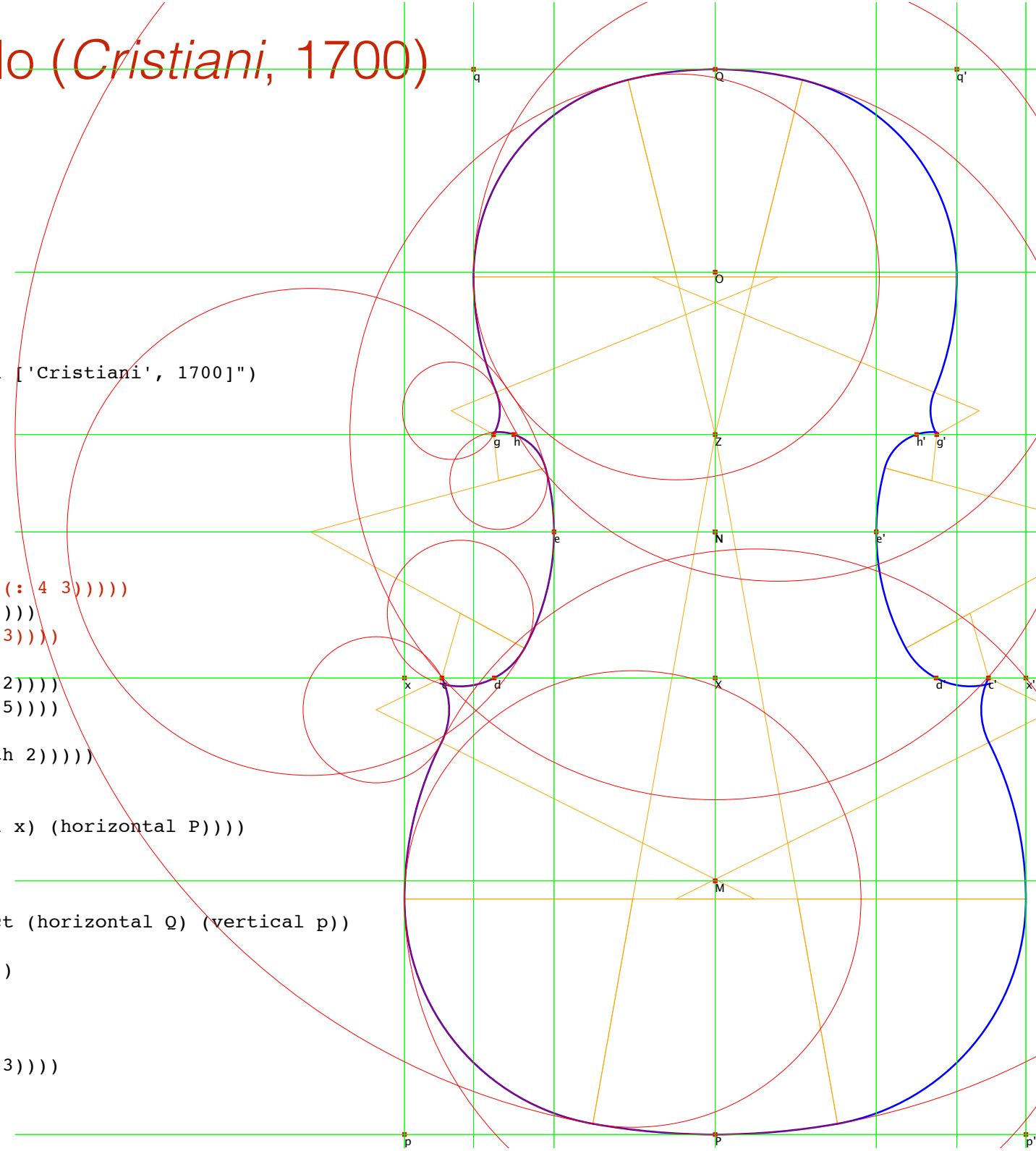
Stradivari violoncello (*Cristiani*, 1700)

tracings taken “by hand” along purfling lines on front and back, reflected, superimposed (4 images)



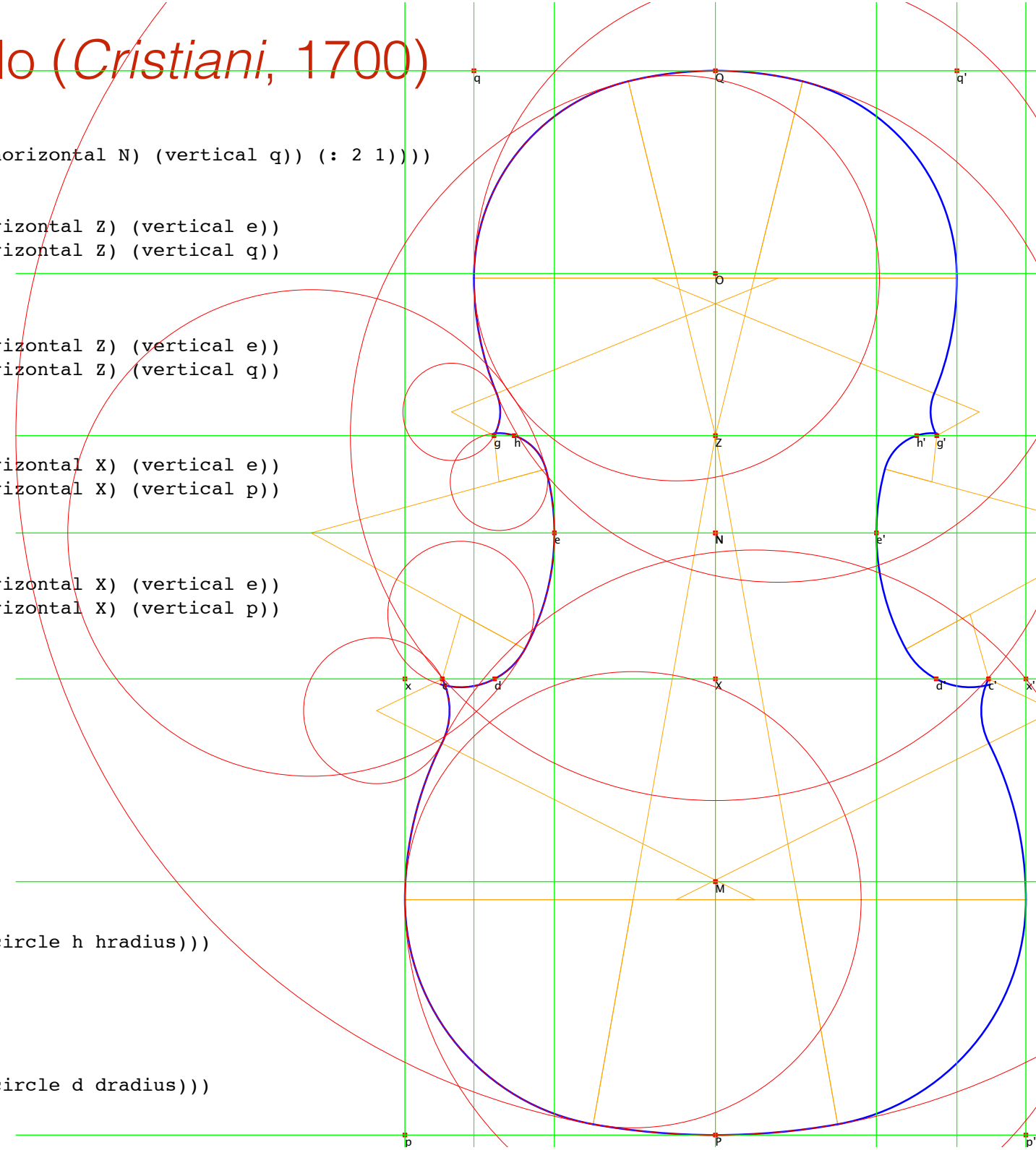
Stradivari violoncello (*Cristiani*, 1700)

```
#lang racket

(require "Geometry-Engine.rkt")

(elaboration #t)
(mirroring #t)
(arcthickness 2)
(arccolor "blue")

(title "Violoncello by Antonio Stradivari ['Cristiani', 1700]")

(define (Cristiani)
(let* (

       (length 760)
       (width (* length (: 7 5)))

       (X (label "X" origin))
       (Q (label "Q" (yshift X (* length (: 4 3)))))
       (P (label "P" (yshift Q (- length))))
       (Z (label "Z" (pointfrom X Q (: 2 3))))

       (O (label "O" (pointfrom Q X (: 1 2))))
       (M (label "M" (pointfrom X P (: 4 5))))

       (x (label "x" (xshift X (- (/ width 2)))))
       (xp (label "x'" (mirror x)))

       (p (label "p" (intersect (vertical x) (horizontal P))))
       (pp (label "p'" (mirror p)))
       (maxlowerwidth (distance p pp))

       (q (label "q" (pointfrom (intersect (horizontal Q) (vertical p))
                                Q
                                (: 2 7))))
       (qp (label "q'" (mirror q)))
       (maxupperwidth (distance q qp))

       (N (label "N" (pointfrom Z X (: 2 3))))
```



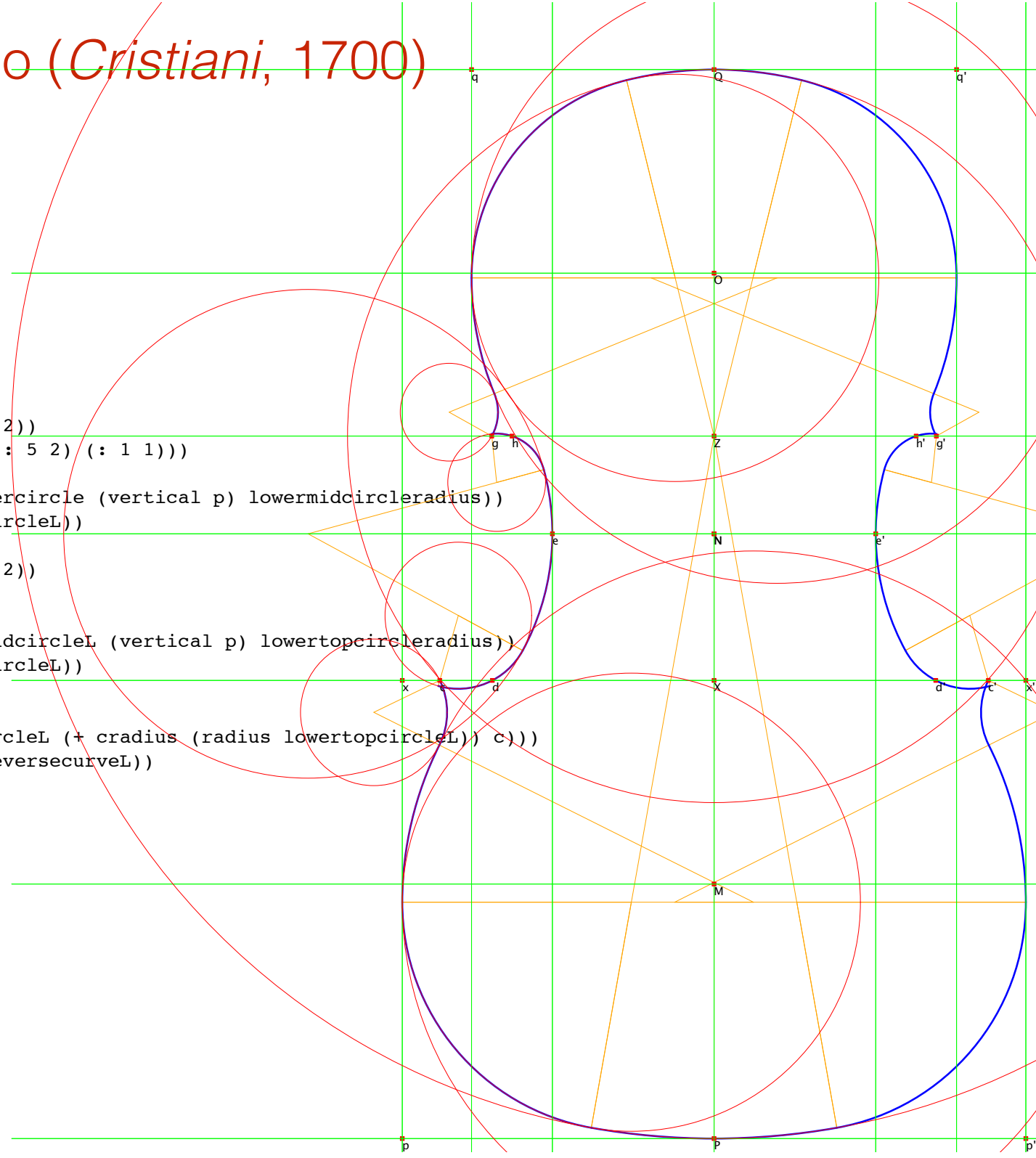
Stradivari violoncello (*Cristiani*, 1700)

```
; middle bout

(e (label "e" (pointfrom N (intersect (horizontal N) (vertical q)) (: 2 1))))
(ep (label "e'" (mirror e)))

(h (label "h" (pointfrom (intersect (horizontal Z) (vertical e))
                         (intersect (horizontal Z) (vertical q))
                         (: 1 1))))
(hp (label "h'" (mirror h)))

(g (label "g" (pointfrom (intersect (horizontal Z) (vertical e))
                         (intersect (horizontal Z) (vertical q))
                         (: 3 1))))
(gp (label "g'" (mirror g)))

(c (label "c" (pointfrom (intersect (horizontal X) (vertical e))
                         (intersect (horizontal X) (vertical p))
                         (: 3 1))))
(cp (label "c'" (mirror c)))

(d (label "d" (pointfrom (intersect (horizontal X) (vertical e))
                         (intersect (horizontal X) (vertical p))
                         (: 2 3))))
(dp (label "d'" (mirror d)))

(fradius (distance X Z))
(f (label "f" (xshift e (- fradius))))
(fp (label "f'" (mirror f)))
(fcircle (circle f fradius))
(fpcircle (mirrorcircle fcircle))

(gradius (/ (distance N Z) 2))

(hradius (* (: 4 1) gradius))

(hcircle (lower-circle (inscribepoint fcircle h hradius)))
(hpcircle (mirrorcircle hcircle))

(cradius (/ (distance X N) 2))
(dradius cradius)

(dcircle (upper-circle (inscribepoint fcircle d dradius)))
(dpcircle (mirrorcircle dcircle))
```



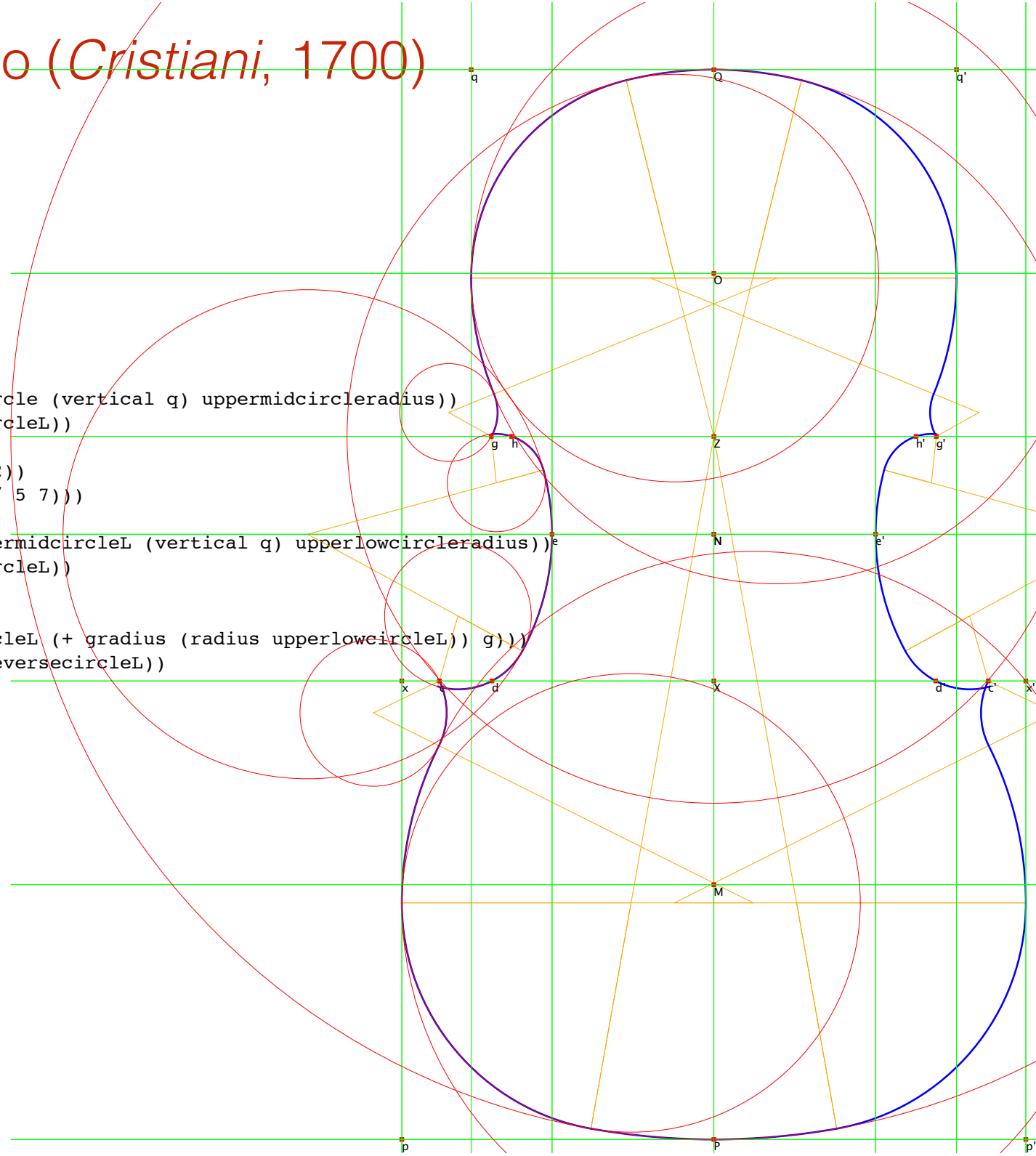
Stradivari violoncello (*Cristiani*, 1700)

```
; lower bout

(lowercircle (circlefrom Z P))

 (lowermidcircleradius (/ (distance X P) 2))
(lowermidcircleradius (* (distance N P) (: 5 2) (: 1 1)))
(lowermidcircleL
  (inscribeinside-circle-line bottom lowercircle (vertical p) lowermidcircleradius))
(lowermidcircleR (mirrorcircle lowermidcircleL))

 (lowertopcircleradius (/ (distance Z P) 2))
(lowertopcircleradius (distance Z Q))
(lowertopcircleL
 (inscribeoutside-circle-line top lowermidcircleL (vertical p) lowertopcircleradius))
(lowertopcircleR (mirrorcircle lowertopcircleL))

(lowerreversecurveL
 (lower-circle (reverse-curve lowertopcircleL (+ cradius (radius lowertopcircleL)) c)))
(lowerreversecurveR (mirrorcircle lowerreversecurveL))
```



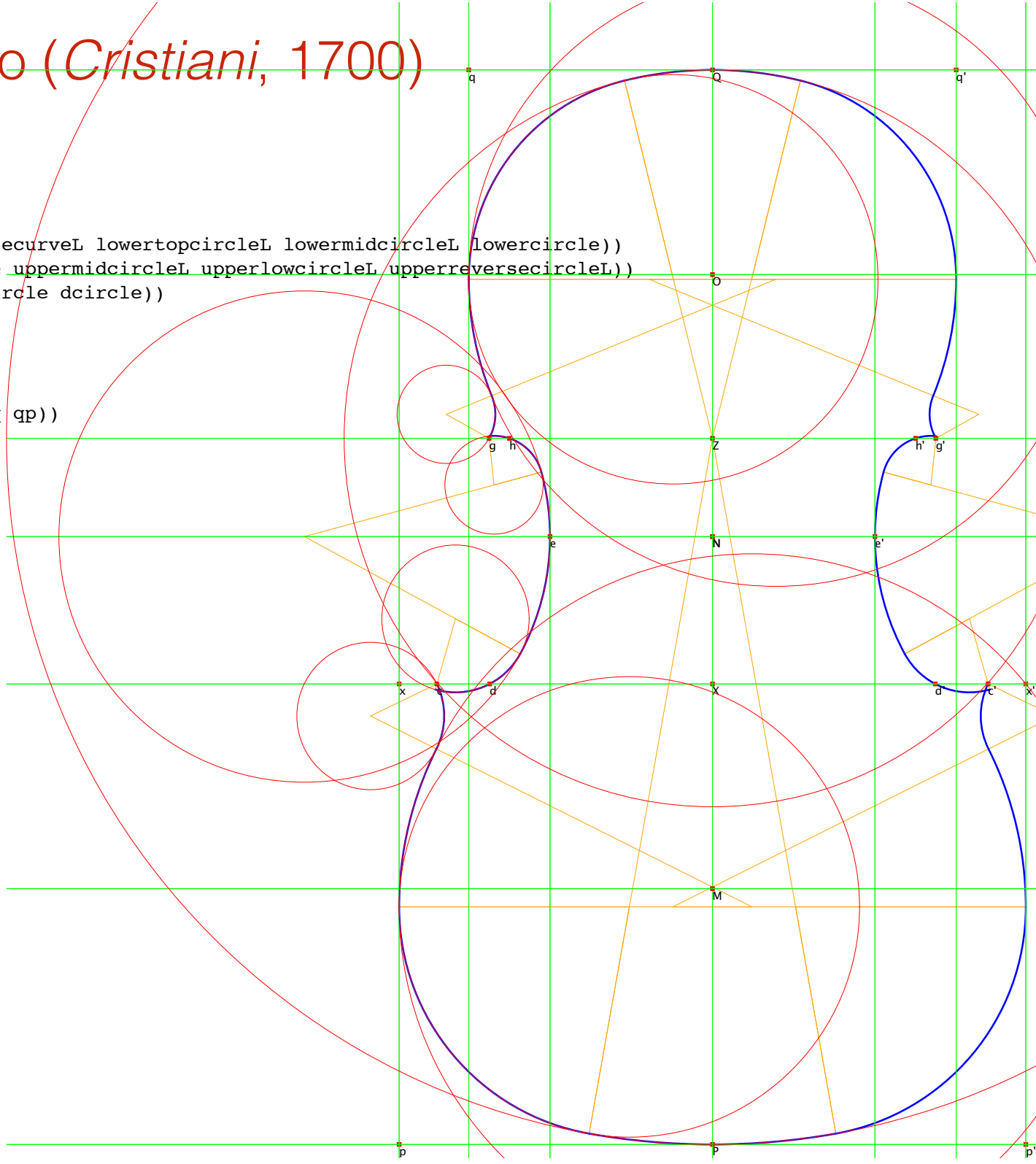
Stradivari violoncello (*Cristiani*, 1700)

```
; upper bout

(uppercircle (circlefrom Z Q))

(uppermidcircleradius (distance O Q))
(uppermidcircleL
  (inscribeinside-circle-line top uppercircle (vertical q) uppermidcircleradius))
(uppermidcircleR (mirrorcircle uppermidcircleL))

 (upperlowcircleradius (/ (distance Q X) 2))
(upperlowcircleradius (* (distance Q Z) (/ 5 7)))
(upperlowcircleL
  (inscribeoutside-circle-line bottom uppermidcircleL (vertical q) upperlowcircleradius))
(upperlowcircleR (mirrorcircle upperlowcircleL))

(upperreversecircleL
 (upper-circle (reverse-curve upperlowcircleL (+ gradius (radius upperlowcircleL)) g)))
(upperreversecircleR (mirrorcircle upperreversecircleL))
```



Stradivari violoncello (*Cristiani*, 1700)

```
(list (make-curve c P (list lowerreversecurveL lowertopcircleL lowermidcircleL lowercircle))
      (make-curve Q g (list uppercircle uppermidcircleL upperlowcircleL upperreversecircleL))
      (make-curve g c (list hcircle fcircle dcircle))

      X Q P Z N x xp q qp p pp N e ep
      g gp h hp c cp d dp
      (map horizontal (list X Q P Z N))
      (map vertical (list p P pp e ep q qp))

      O (horizontal O)
      M (horizontal M)
      lowercircle uppercircle
      upperlowcircleL
      uppermidcircleL
      upperreversecircleL

      lowertopcircleL
      lowermidcircleL
      lowerreversecurveL

      fcircle
      hcircle
      dcircle
      )

      ))


(sketch (Cristiani))

(end-drawing)
```



Stradivari violoncello (*Cristiani*, 1700)

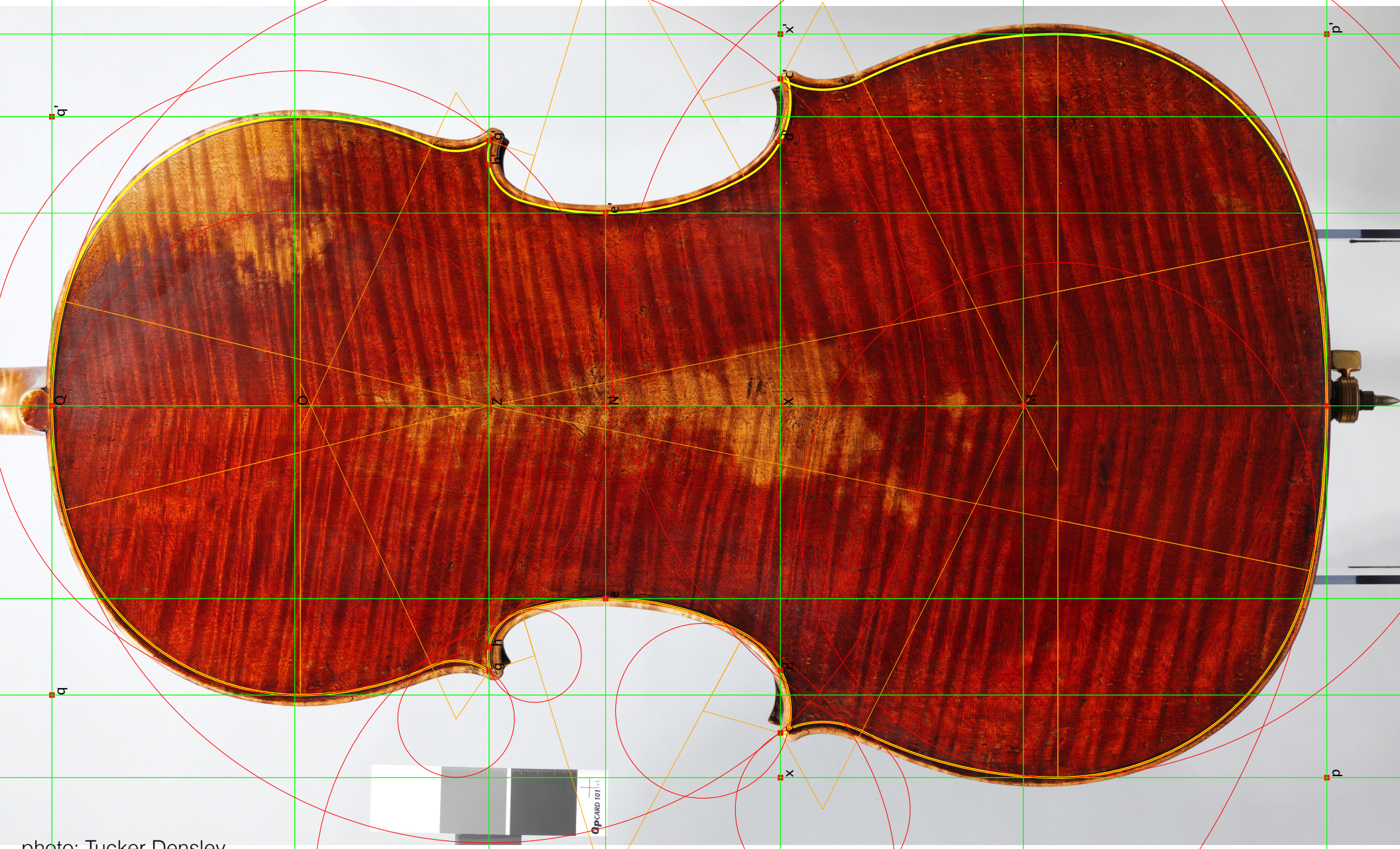


photo: Tucker Densley

Comparing the Stradivari *Castelbarco* (1699, same size) with the idealized *Cristiani*

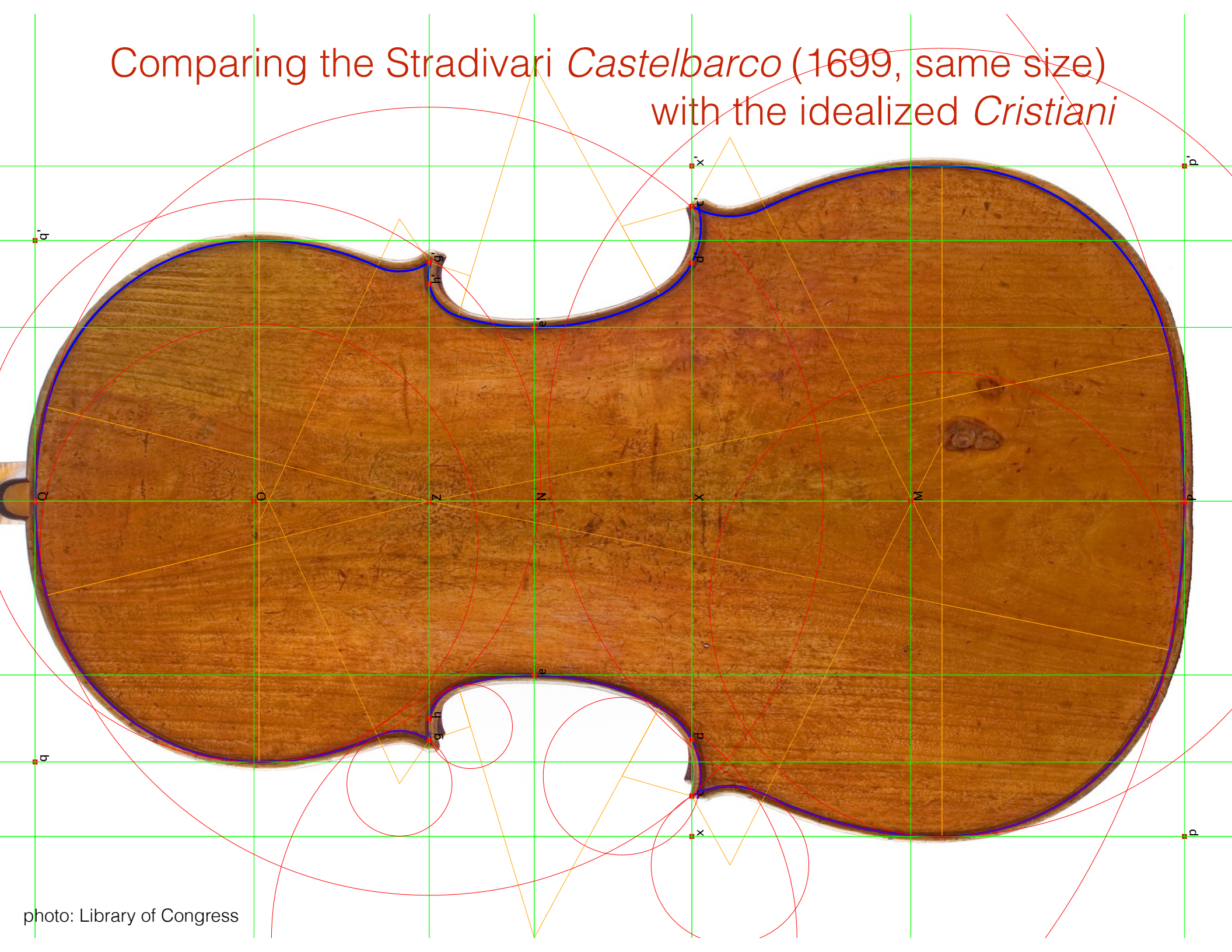
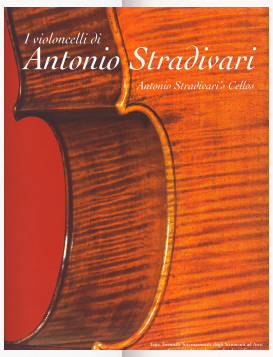
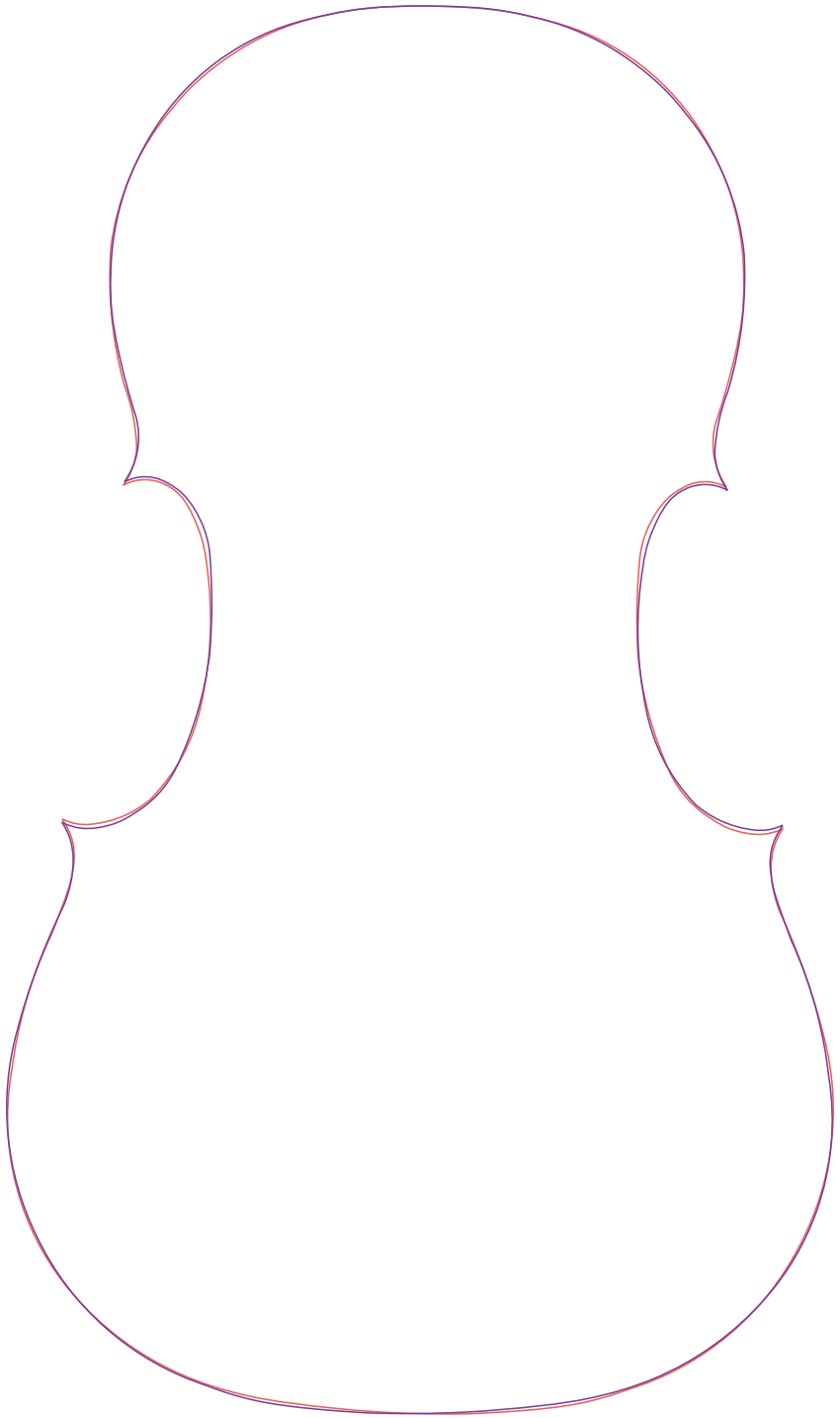


photo: Library of Congress

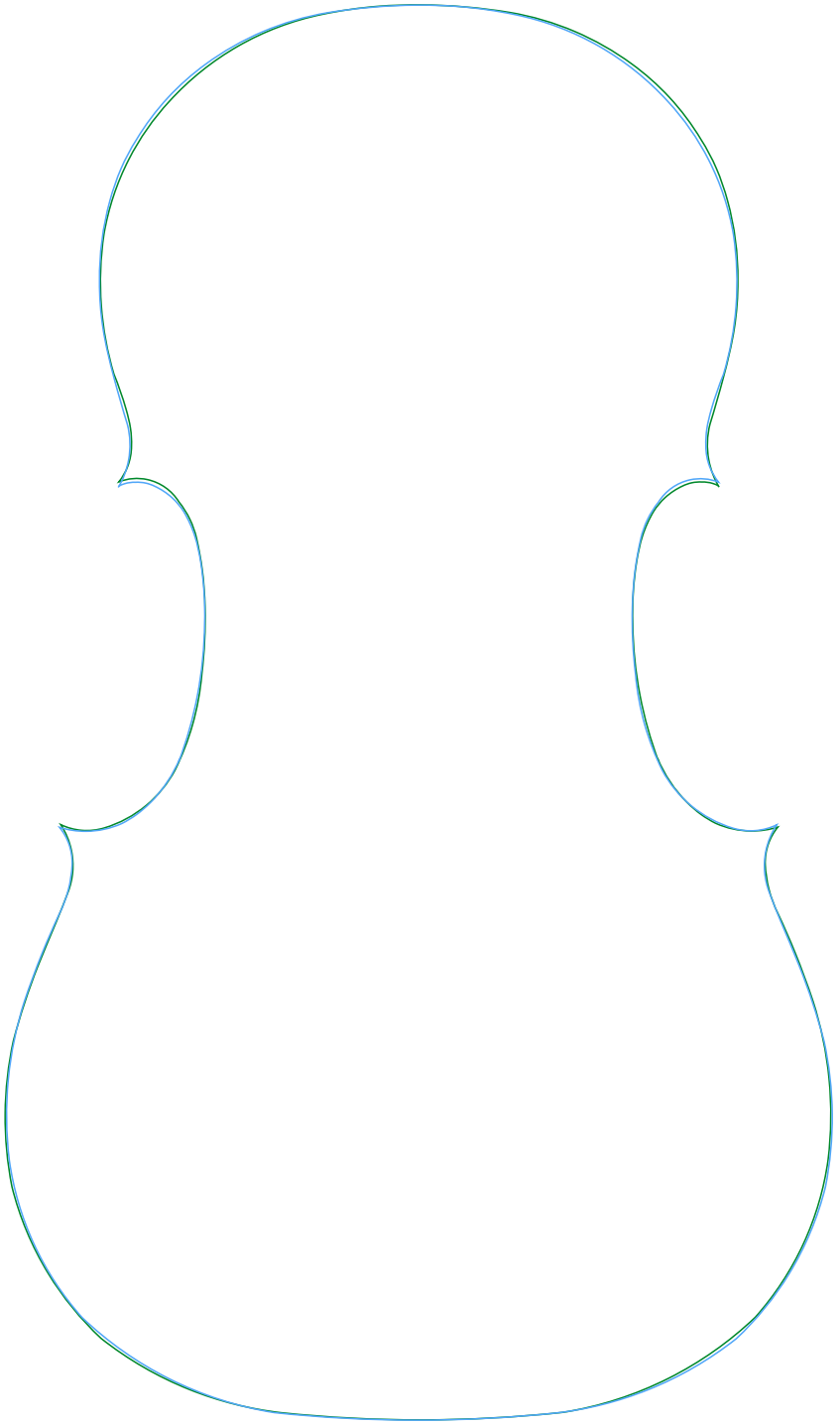
Mediceo, 1690, 792mm



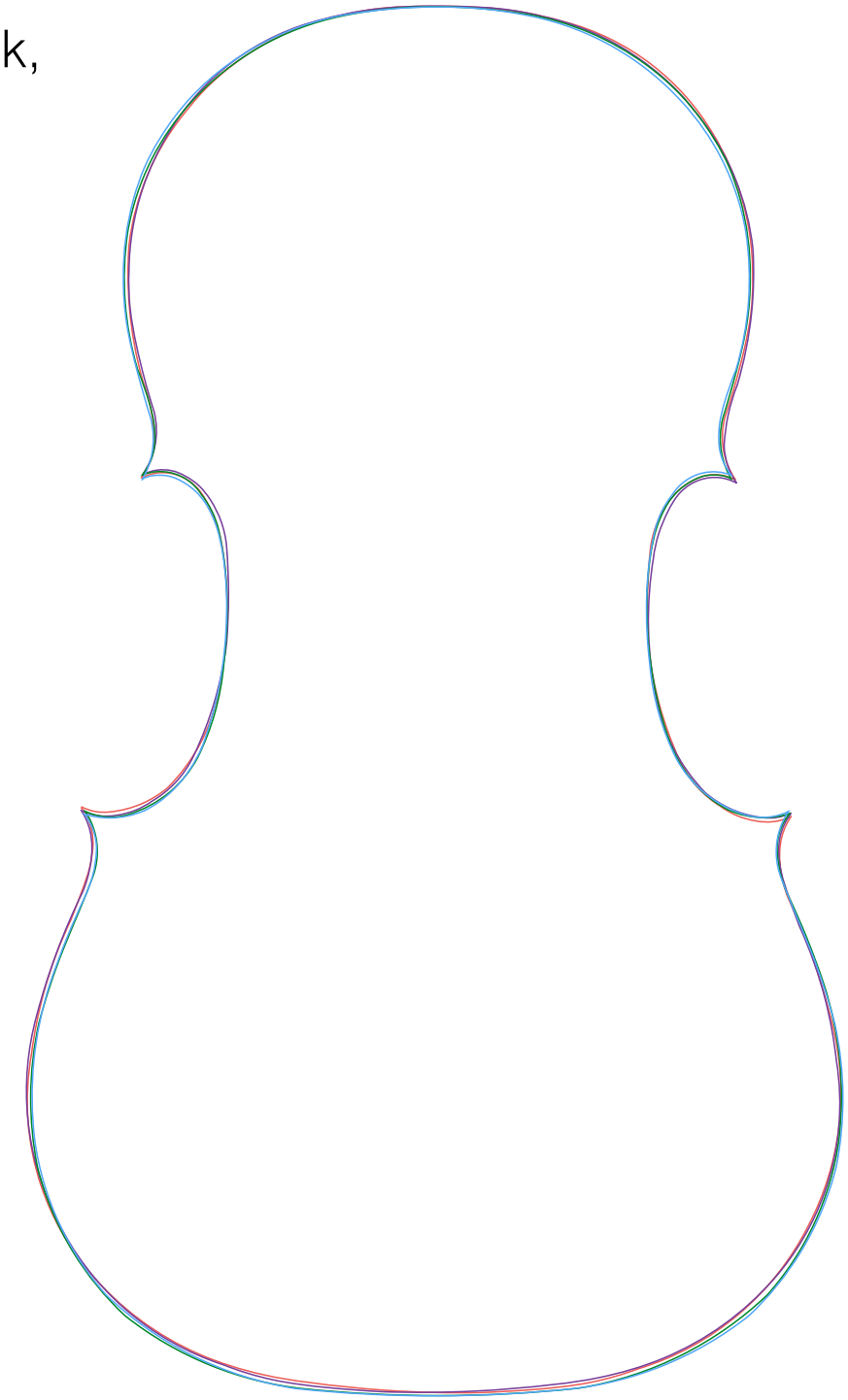
Mediceo front



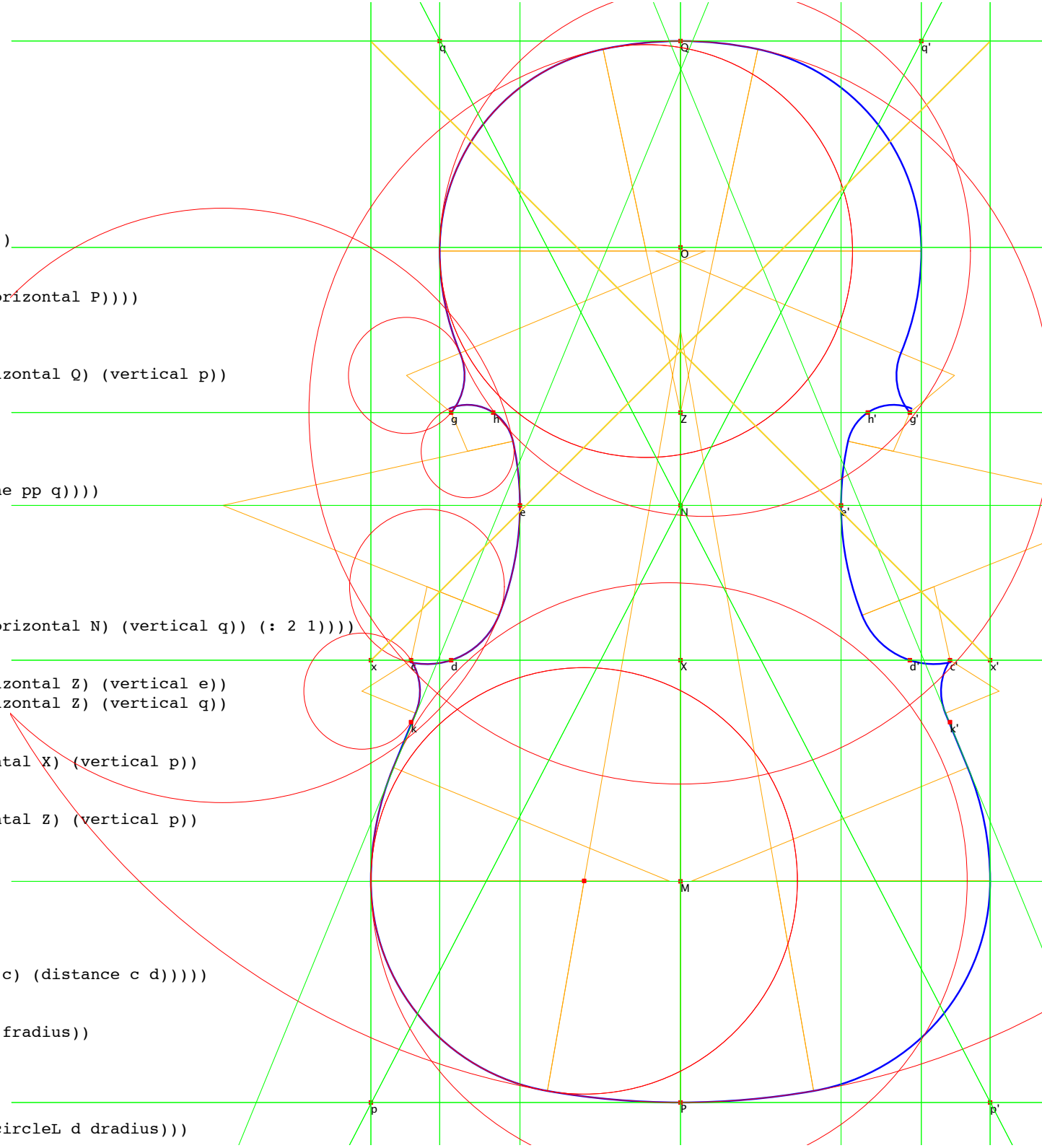
Mediceo back



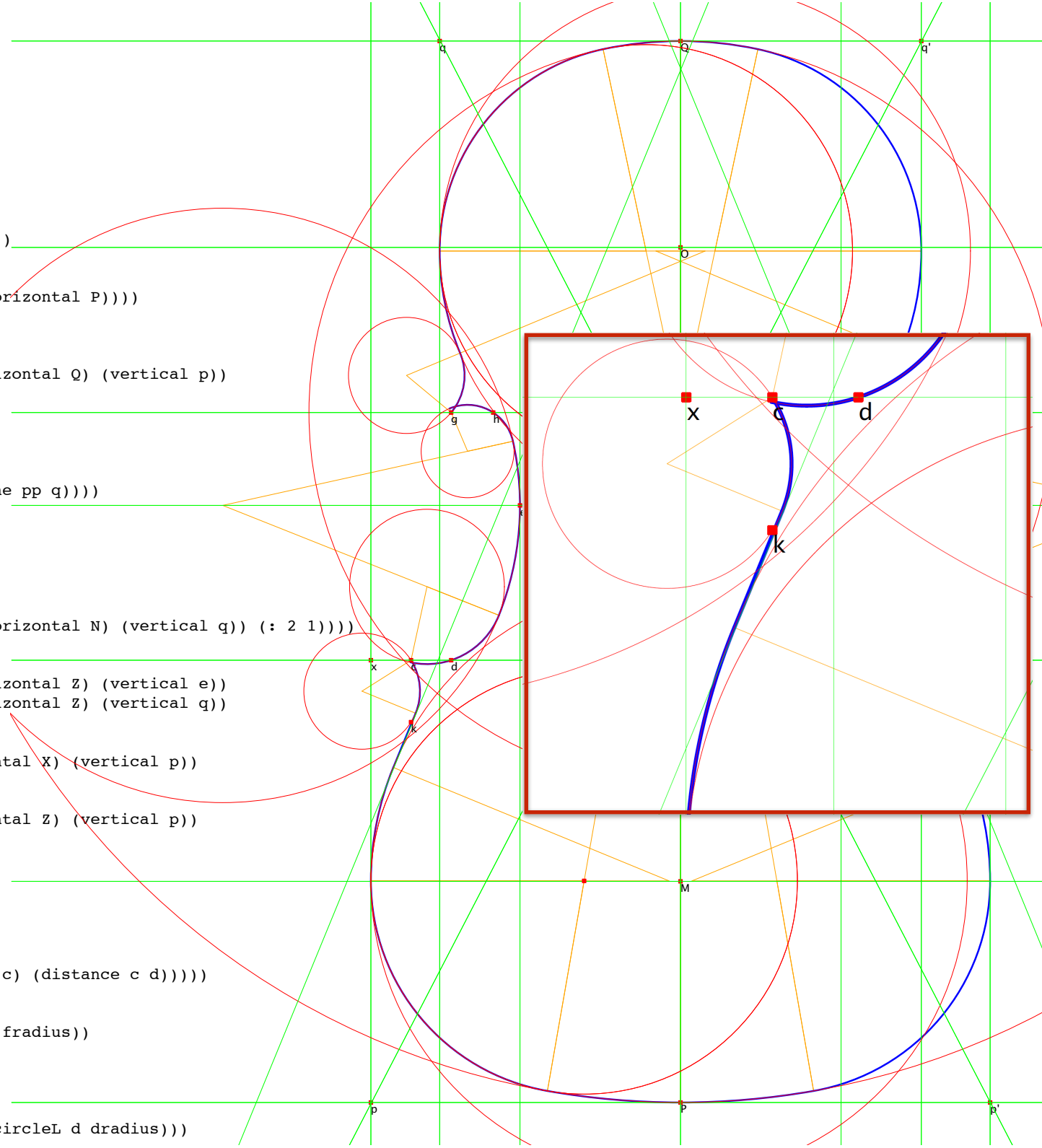
Mediceo front and back, reflected



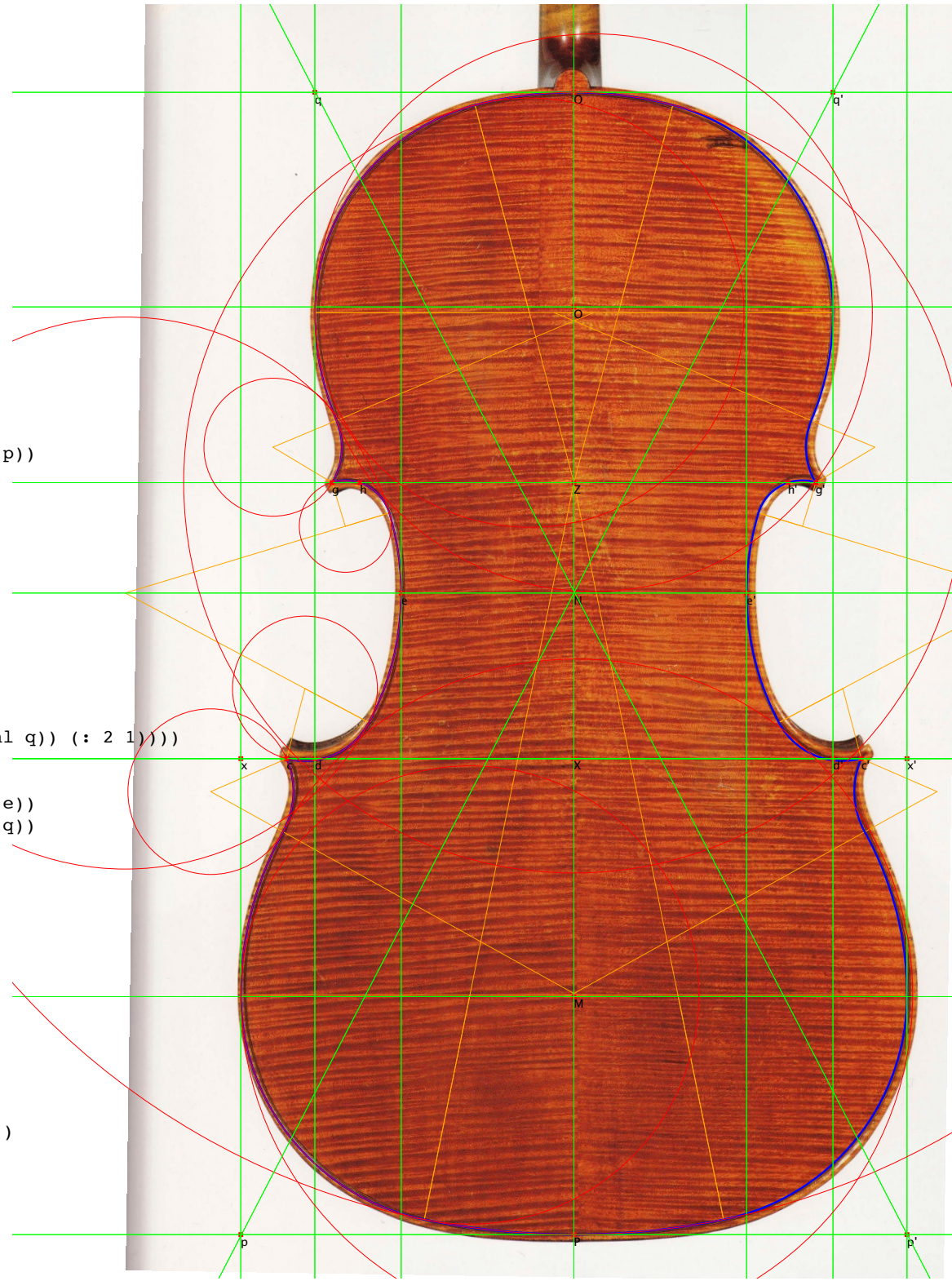
```
(define (Mediceo)
(let* (

       (length 792.5)
       (width (* length (: 7 5)))

       (X (label "X" origin))
       (Q (label "Q" (yshift X width)))
       (P (label "P" (yshift Q (- length))))
       (Z (label "Z" (pointfrom X Q (: 2 3))))

       (x (label "x" (xshift X (- (/ width 2)))))
       (xp (label "x'" (mirror x)))

       (p (label "p" (intersect (vertical x) (horizontal P))))
       (pp (label "p'" (mirror p)))
       (maxlowerwidth (distance p pp))

       (q (label "q" (pointfrom (intersect (horizontal Q) (vertical p))
                                Q
                                (: 2 7))))
       (qp (label "q'" (mirror q)))
       (maxupperwidth (distance q qp))

       (N (label "N" (intersect (line p qp) (line pp q))))
          ; diagonal form: note ZN:ZX = 3:5

       (O (label "O" (pointfrom Q N (: 4 5))))
       (M (label "M" (pointfrom X P (: 1 1))))

; middle bout
       (e (label "e" (pointfrom N (intersect (horizontal N) (vertical q)) (: 2 1))))
       (ep (label "e'" (mirror e)))

       (h (label "h" (pointfrom (intersect (horizontal Z) (vertical e))
                                (intersect (horizontal Z) (vertical q))
                                (: 1 2))))
       (hp (label "h'" (mirror h)))
       (d (label "d" (xshift (intersect (horizontal X) (vertical p))
                             (xdistance q e))))
       (dp (label "d'" (mirror d)))
       (g (label "g" (xshift (intersect (horizontal Z) (vertical p))
                             (xdistance q e))))
       (gp (label "g'" (mirror g)))


       (c (label "c" (midpoint x d)) )
       (cp (label "c'" (mirror c)))

       (foo (write (list 'FOOFOO (- (distance x c) (distance c d)))))

       (fradius (* (distance Z Q) (: 4 1)))
       (fcircleL (circle (xshift e (- fradius)) fradius))
       (fcircleR (mirrorcircle fcircleL))

       (dradius (/ (distance N X) 2))

       (dcircleL (upper-circle (inscribepoint fcircleL d dradius)))
```



```
(define (Mediceo)
(let* (

       (length 792.5)
       (width (* length (: 7 5)))

       (X (label "X" origin))
       (Q (label "Q" (yshift X width)))
       (P (label "P" (yshift Q (- length))))
       (Z (label "Z" (pointfrom X Q (: 2 3))))

       (x (label "x" (xshift X (- (/ width 2)))))
       (xp (label "x'" (mirror x)))

       (p (label "p" (intersect (vertical x) (horizontal P))))
       (pp (label "p'" (mirror p)))
       (maxlowerwidth (distance p pp))

       (q (label "q" (pointfrom (intersect (horizontal Q) (vertical p))
                                Q
                                (: 2 7))))
       (qp (label "q'" (mirror q)))
       (maxupperwidth (distance q qp))

       (N (label "N" (intersect (line p qp) (line pp q))))
          ; diagonal form: note ZN:ZX = 3:5

       (O (label "O" (pointfrom Q N (: 4 5))))
       (M (label "M" (pointfrom X P (: 1 1))))

; middle bout
       (e (label "e" (pointfrom N (intersect (horizontal N) (vertical q)) (: 2 1))))
       (ep (label "e'" (mirror e)))

       (h (label "h" (pointfrom (intersect (horizontal Z) (vertical e))
                                (intersect (horizontal Z) (vertical q))
                                (: 1 2))))
       (hp (label "h'" (mirror h)))
       (d (label "d" (xshift (intersect (horizontal X) (vertical p))
                             (xdistance q e))))
       (dp (label "d'" (mirror d)))
       (g (label "g" (xshift (intersect (horizontal Z) (vertical p))
                             (xdistance q e))))
       (gp (label "g'" (mirror g)))


       (c (label "c" (midpoint x d)) )
       (cp (label "c'" (mirror c)))

       (foo (write (list 'FOOFOO (- (distance x c) (distance c d)))))

       (fradius (* (distance Z Q) (: 4 1)))
       (fcircleL (circle (xshift e (- fradius)) fradius))
       (fcircleR (mirrorcircle fcircleL))

       (dradius (/ (distance N X) 2))

       (dcircleL (upper-circle (inscribepoint fcircleL d dradius)))
```



```
(define (Mediceo)
(let* (

       (length 792.5)
       (width (* length (: 7 5)))

       (X (label "X" origin))
       (Q (label "Q" (yshift X width)))
       (P (label "P" (yshift Q (- length))))
       (Z (label "Z" (pointfrom X Q (: 2 3))))

       (x (label "x" (xshift X (- (/ width 2)))))
       (xp (label "x'" (mirror x)))

       (p (label "p" (intersect (vertical x) (horizontal P))))
       (pp (label "p'" (mirror p)))
       (maxlowerwidth (distance p pp))

       (q (label "q" (pointfrom (intersect (horizontal Q) (vertical p))
                                Q
                                (: 2 7))))
       (qp (label "q'" (mirror q)))
       (maxupperwidth (distance q qp))

       (N (label "N" (intersect (line p qp) (line pp q))))
          ; diagonal form: note ZN:ZX = 3:5

       (O (label "O" (pointfrom Q N (: 4 5))))
       (M (label "M" (pointfrom X P (: 1 1))))

; middle bout
       (e (label "e" (pointfrom N (intersect (horizontal N) (vertical q)) (: 2 1))))
       (ep (label "e'" (mirror e)))

       (h (label "h" (pointfrom (intersect (horizontal Z) (vertical e))
                                (intersect (horizontal Z) (vertical q))
                                (: 1 2))))
       (hp (label "h'" (mirror h)))
       (d (label "d" (xshift (intersect (horizontal X) (vertical p))
                             (xdistance q e))))
       (dp (label "d'" (mirror d)))
       (g (label "g" (xshift (intersect (horizontal Z) (vertical p))
                             (xdistance q e))))
       (gp (label "g'" (mirror g)))


       (c (label "c" (midpoint x d)) )
       (cp (label "c'" (mirror c)))

       (foo (write (list 'FOOFOO (- (distance x c) (distance c d)))))

       (fradius (* (distance Z Q) (: 4 1)))
       (fcircleL (circle (xshift e (- fradius)) fradius))
       (fcircleR (mirrorcircle fcircleL))

       (dradius (/ (distance N X) 2))
```

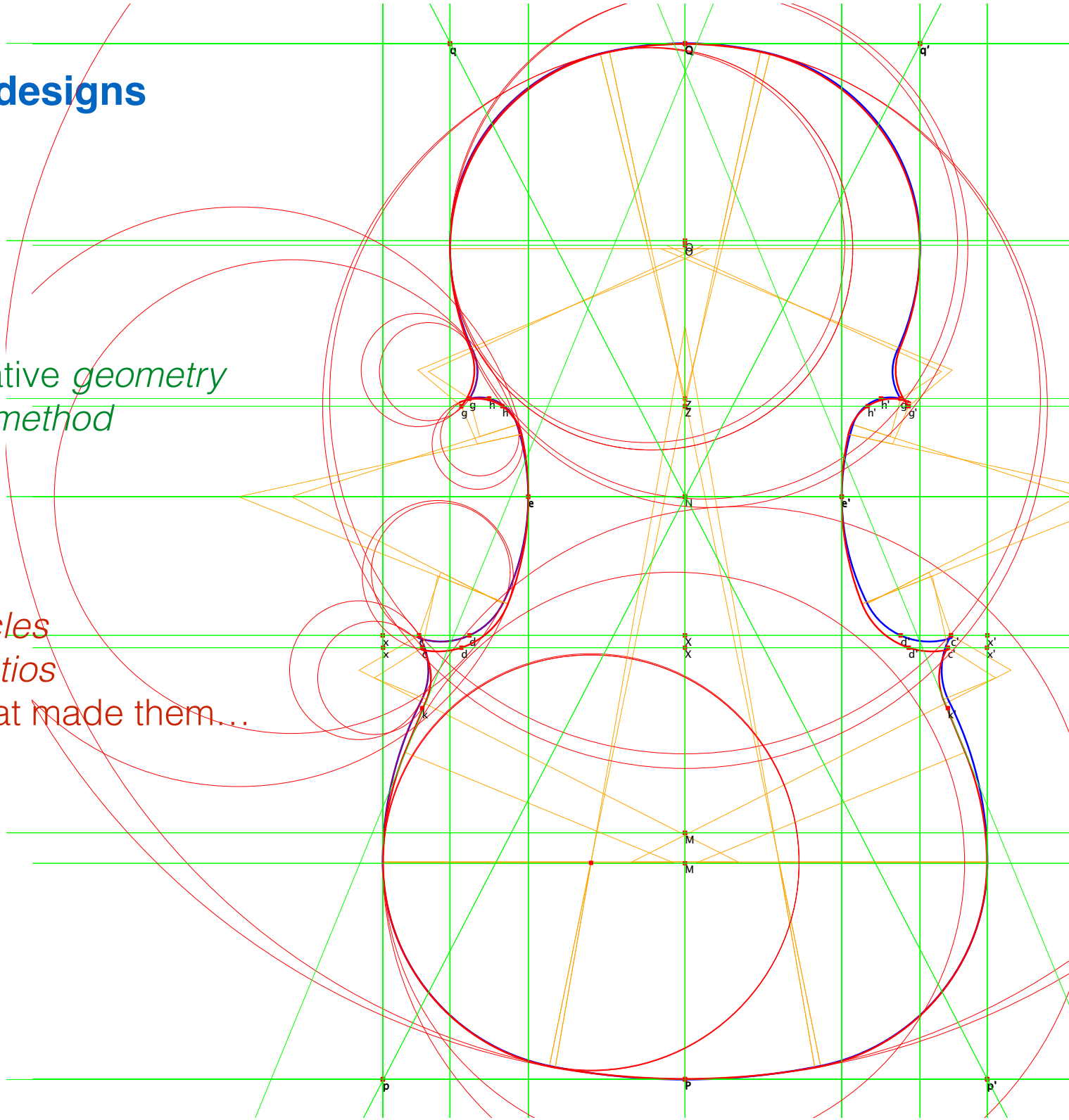


Comparative cello designs

Mediceo Cristiani
(scaled, of course…)

Try and relate comparative *geometry*
and also comparative *method*

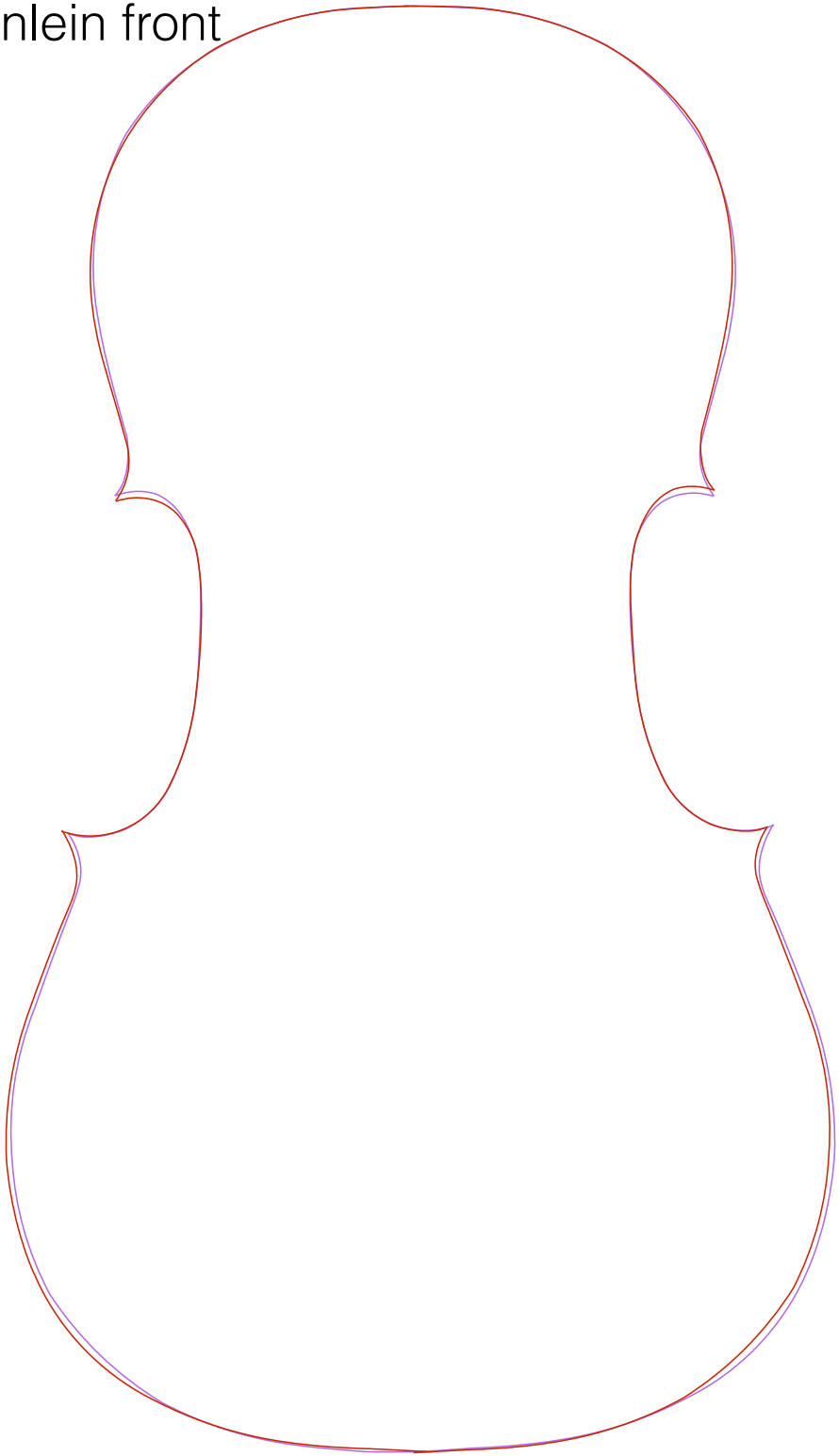
Data visualization:
- the *outline*
- the *underlying circles*
- the *proportional ratios*
  - (the **code**!) that made them…



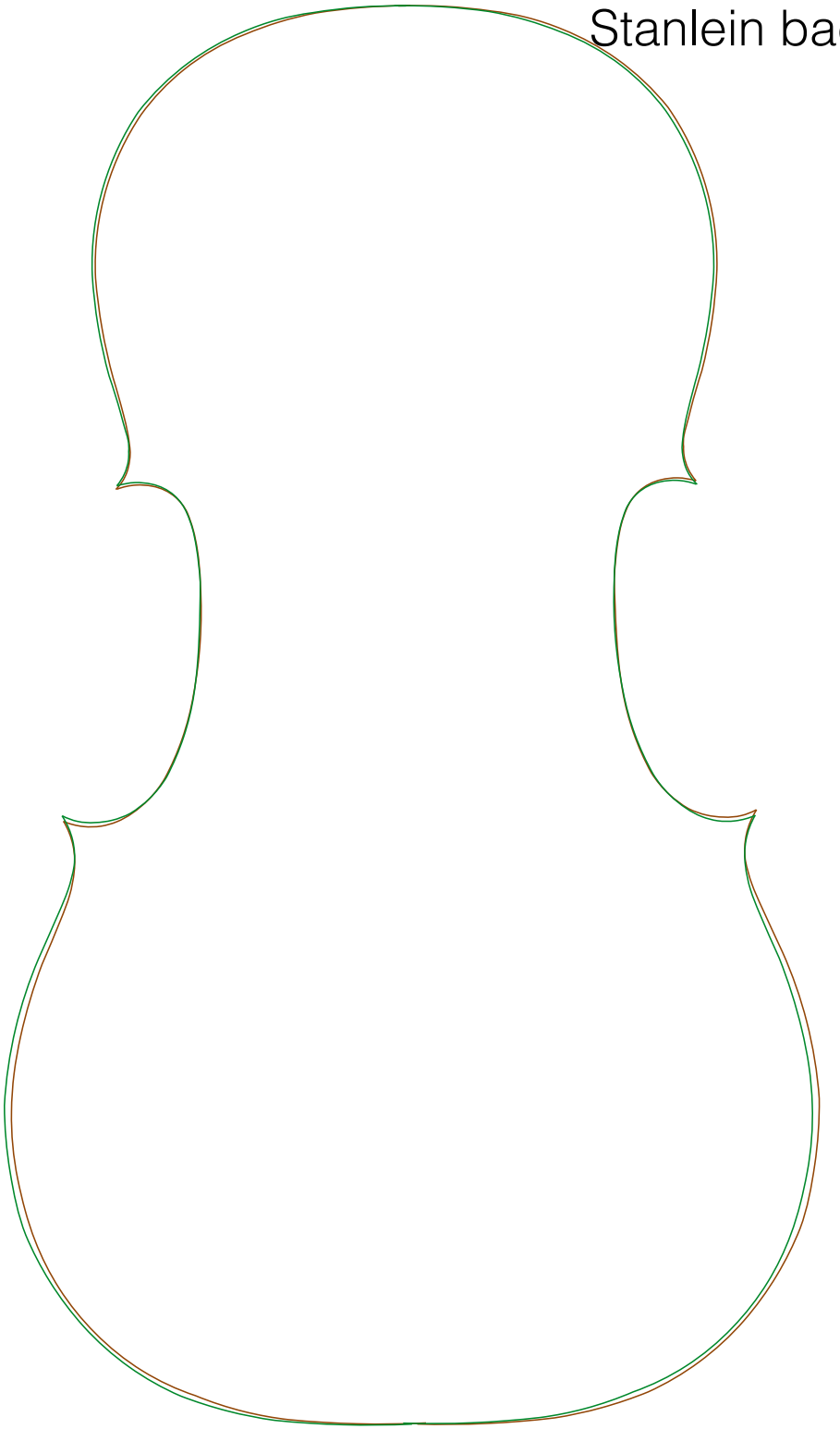
Countess of Stanlein, ex-Paganini 1707, 75?mm

[Reuning brochure]

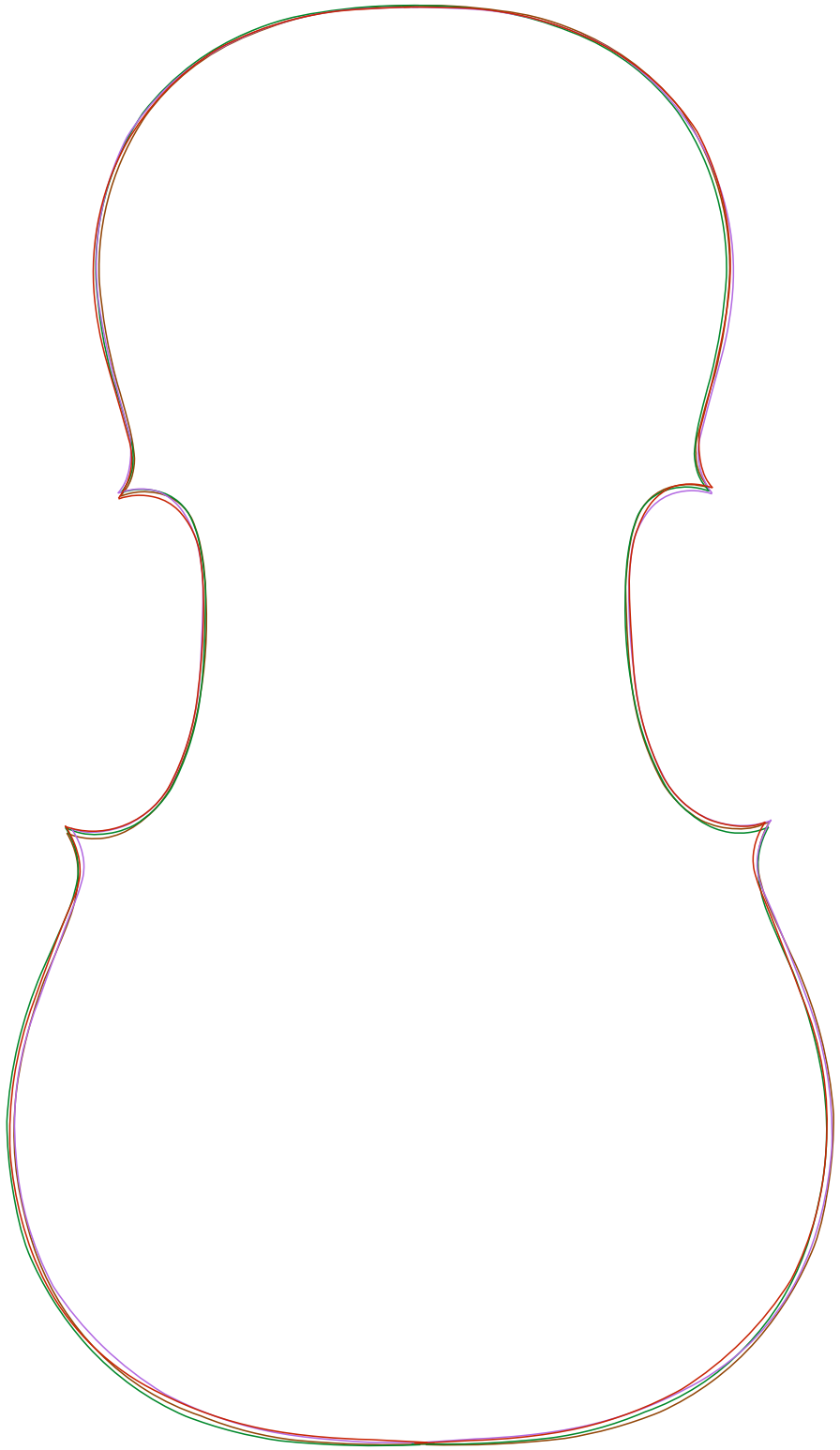
Stanlein front



Stanlein back

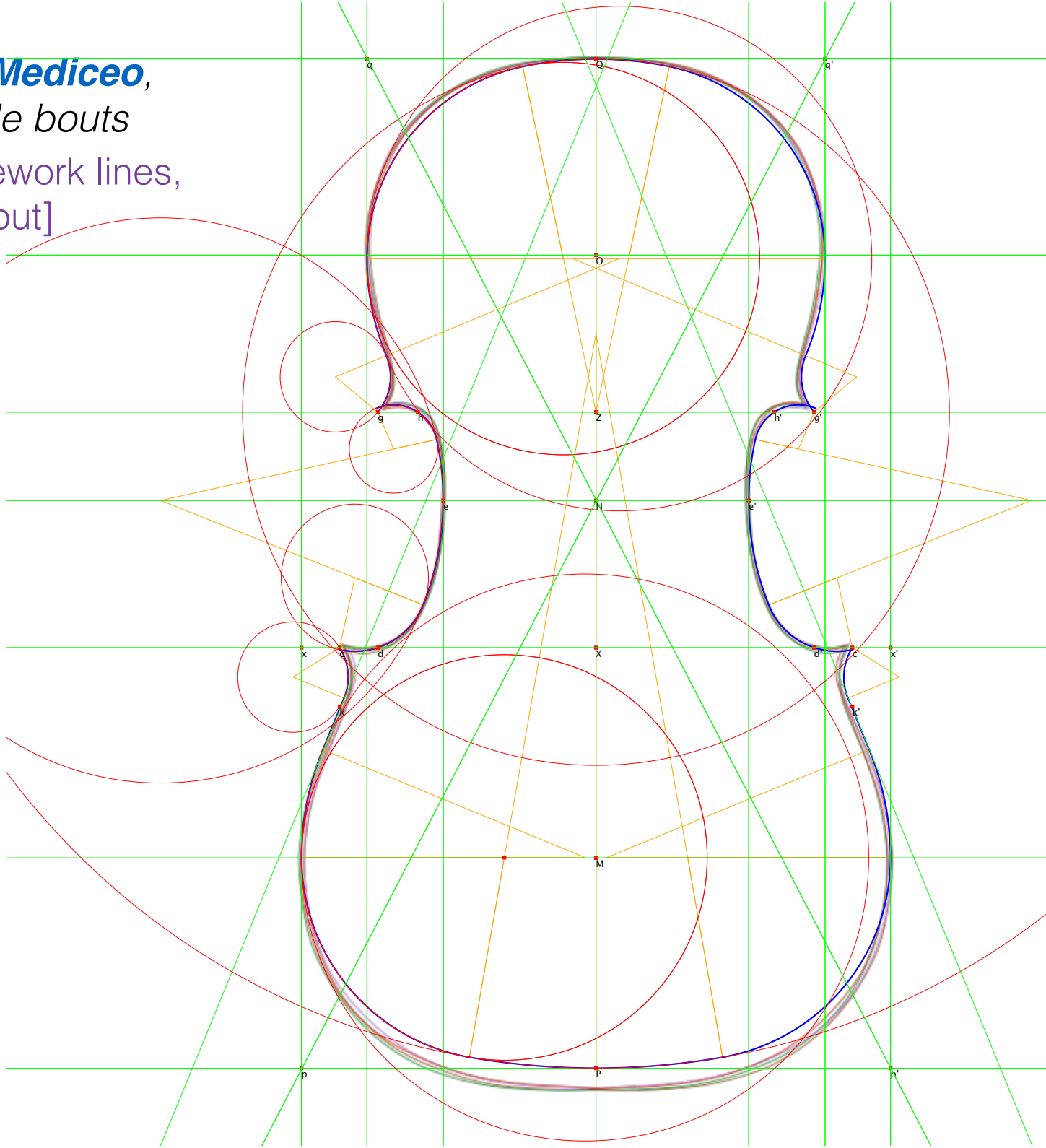


Stanlein front and back, reflected



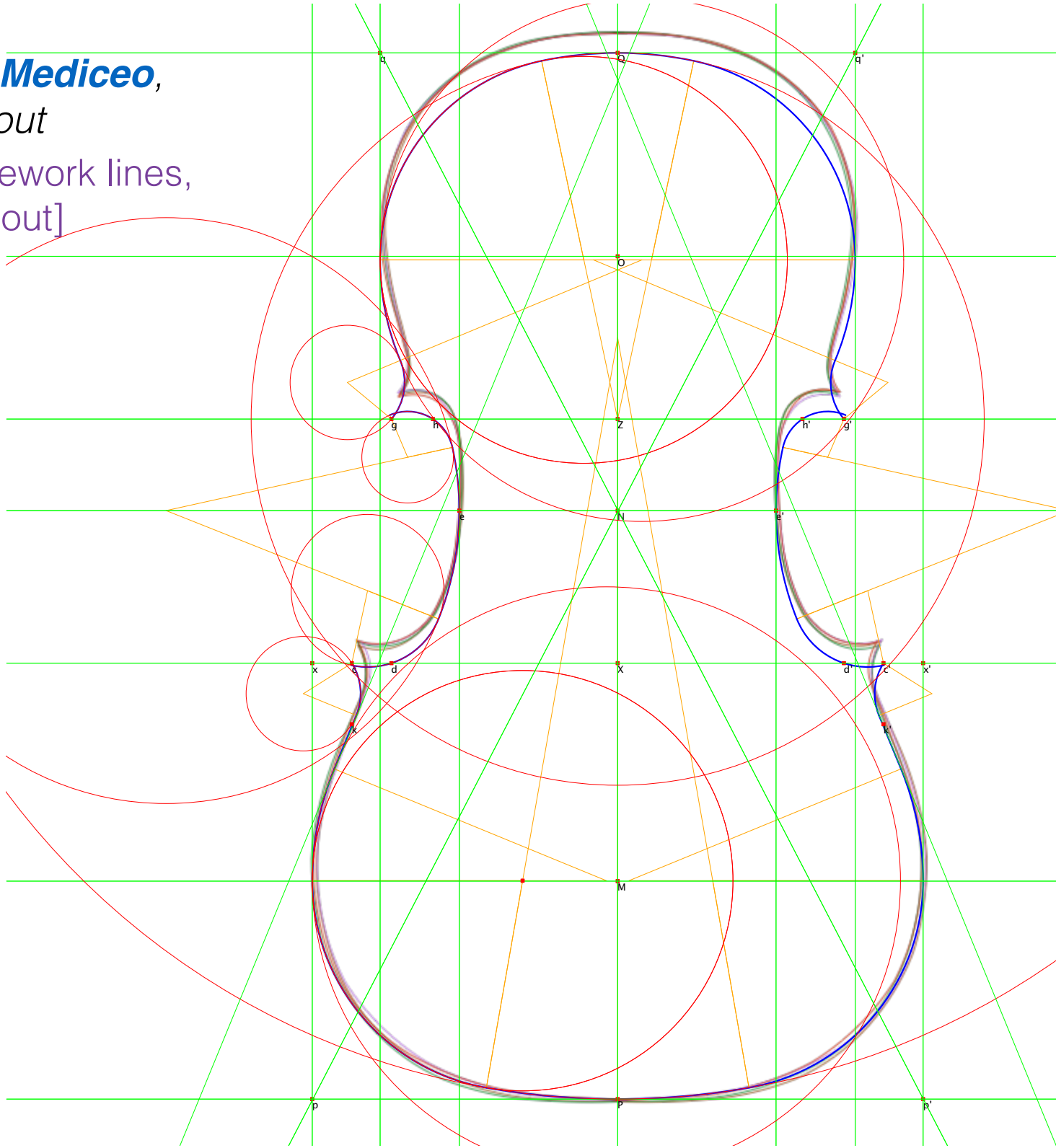
*Stanlein and idealized **Mediceo**, fitted to top and middle bouts*

[note congruence of framework lines, tangent in lower bout]

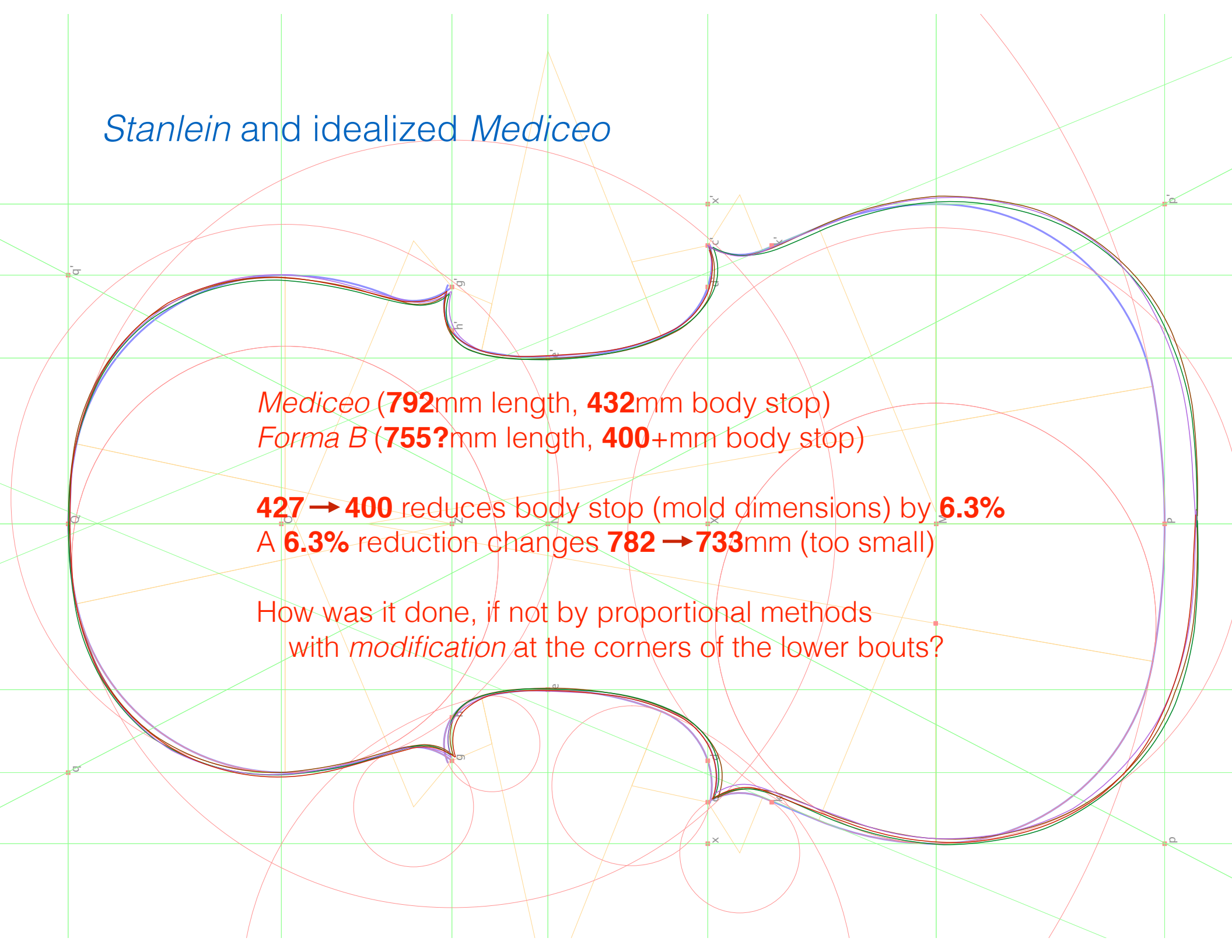


Stanlein and idealized ***Mediceo****,*
fitted to lower bout

[note congruence of framework lines,
tangent in lower bout]



Stanlein and idealized *Mediceo*



Mediceo (**792**mm length, **432**mm body stop)
Forma B (**755?**mm length, **400**+mm body stop)

427 → 400 reduces body stop (mold dimensions) by **6.3%**
A **6.3%** reduction changes **782 → 733**mm (too small)

How was it done, if not by proportional methods
with *modification* at the corners of the lower bouts?

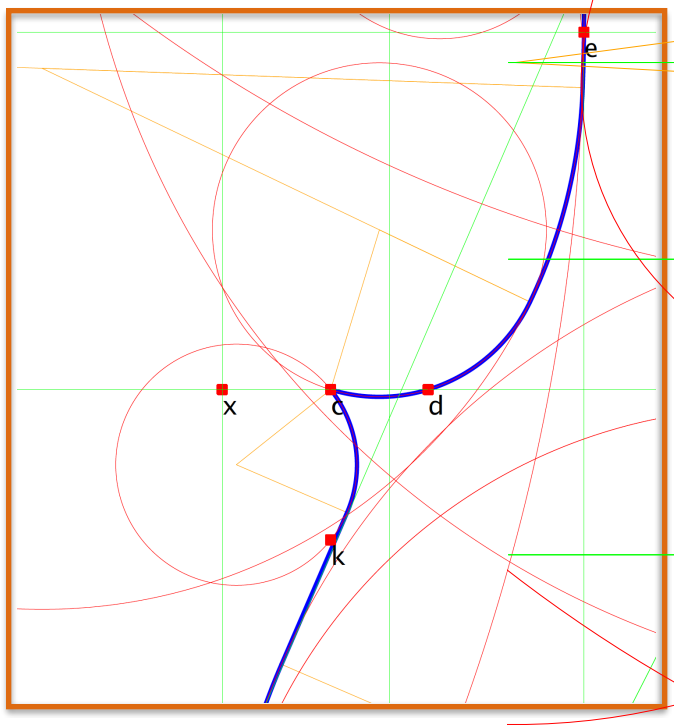
Stanlein via *Mediceo*

code:

Scale the Mediceo down to the appropriate overall dimensions in the top two bouts.

Move the lower bout down ~12mm, and extend tangent lines to connect the dots.

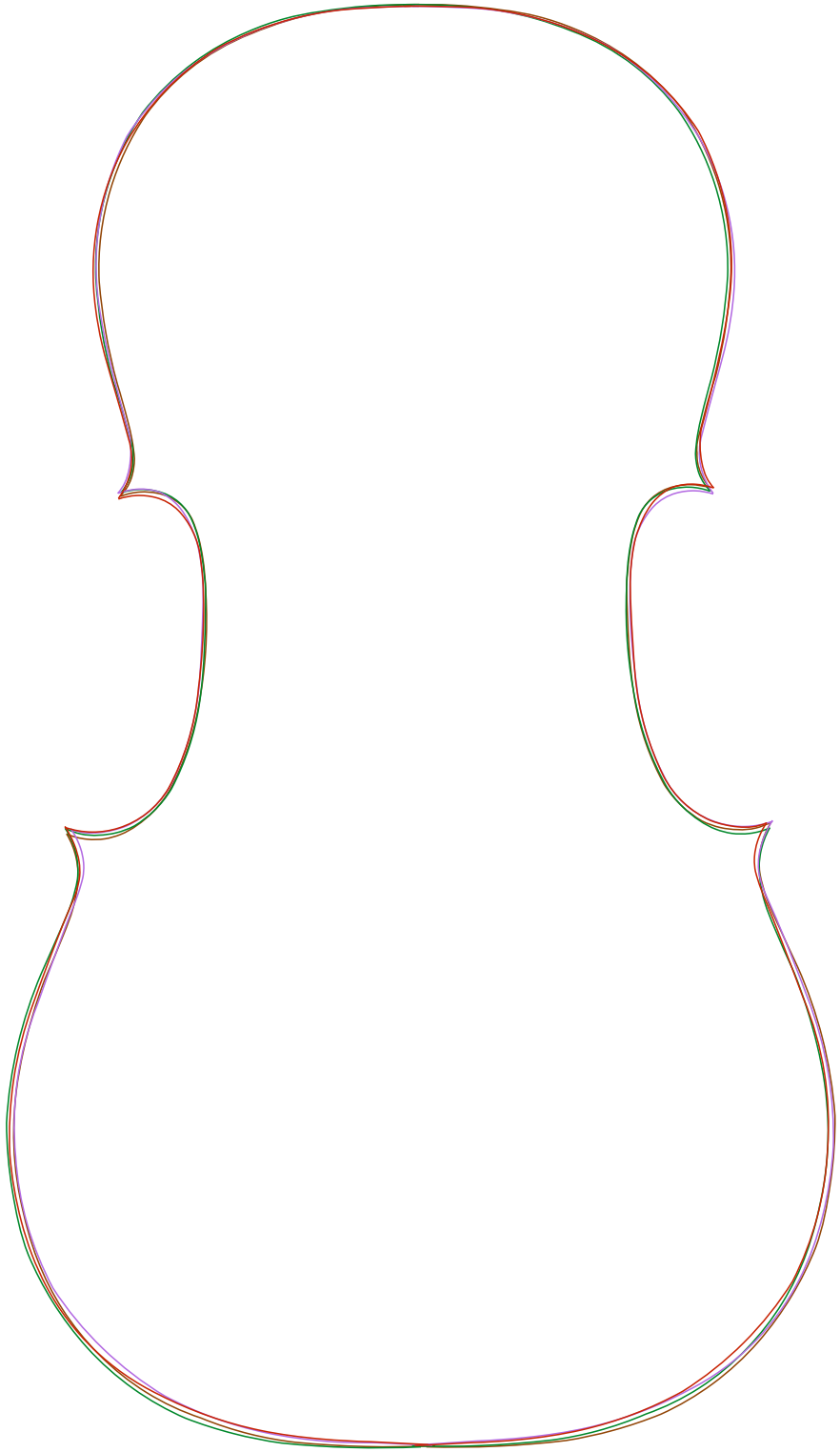
More complex middle bout and upper corner.



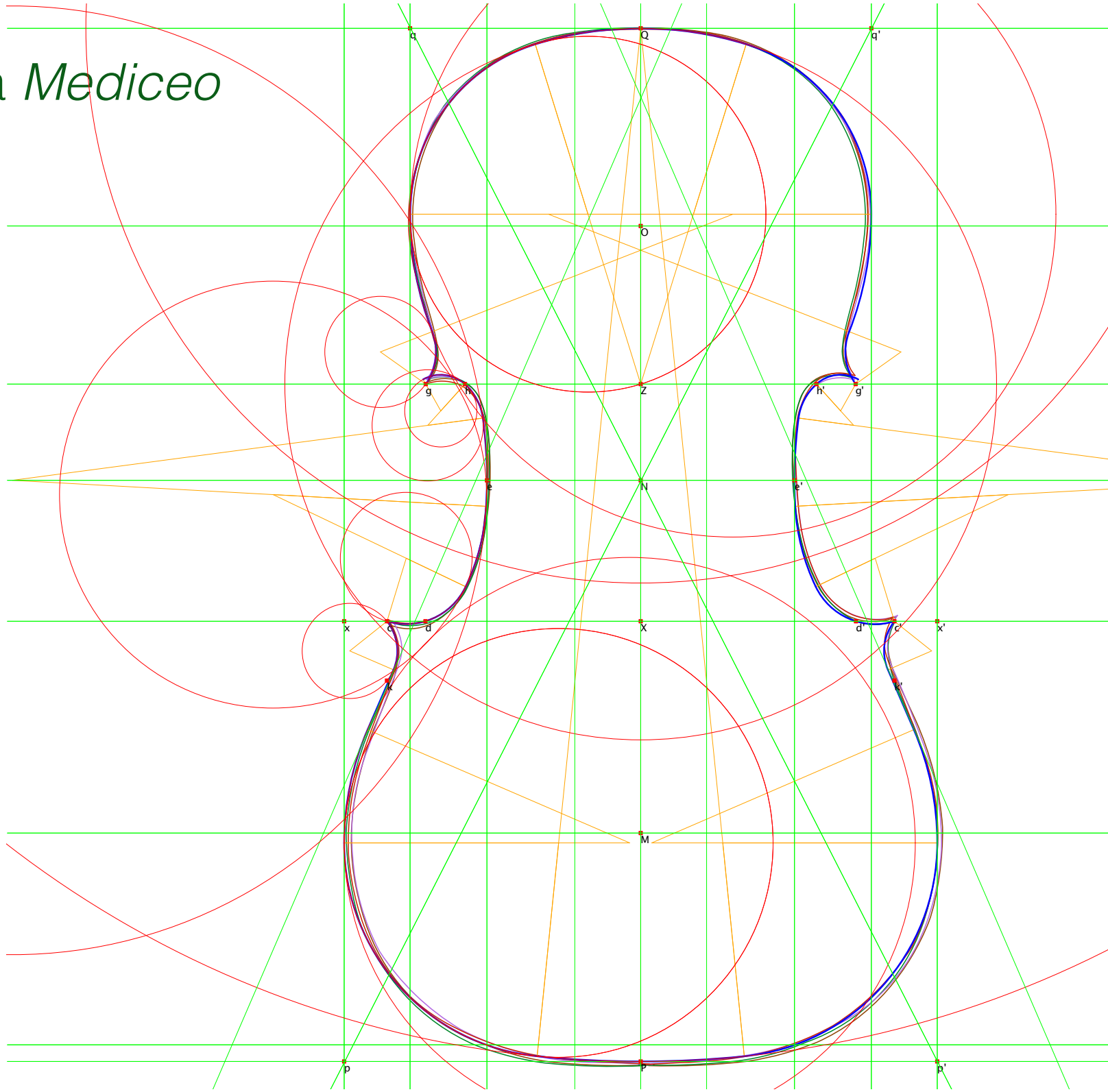
where did this come from?

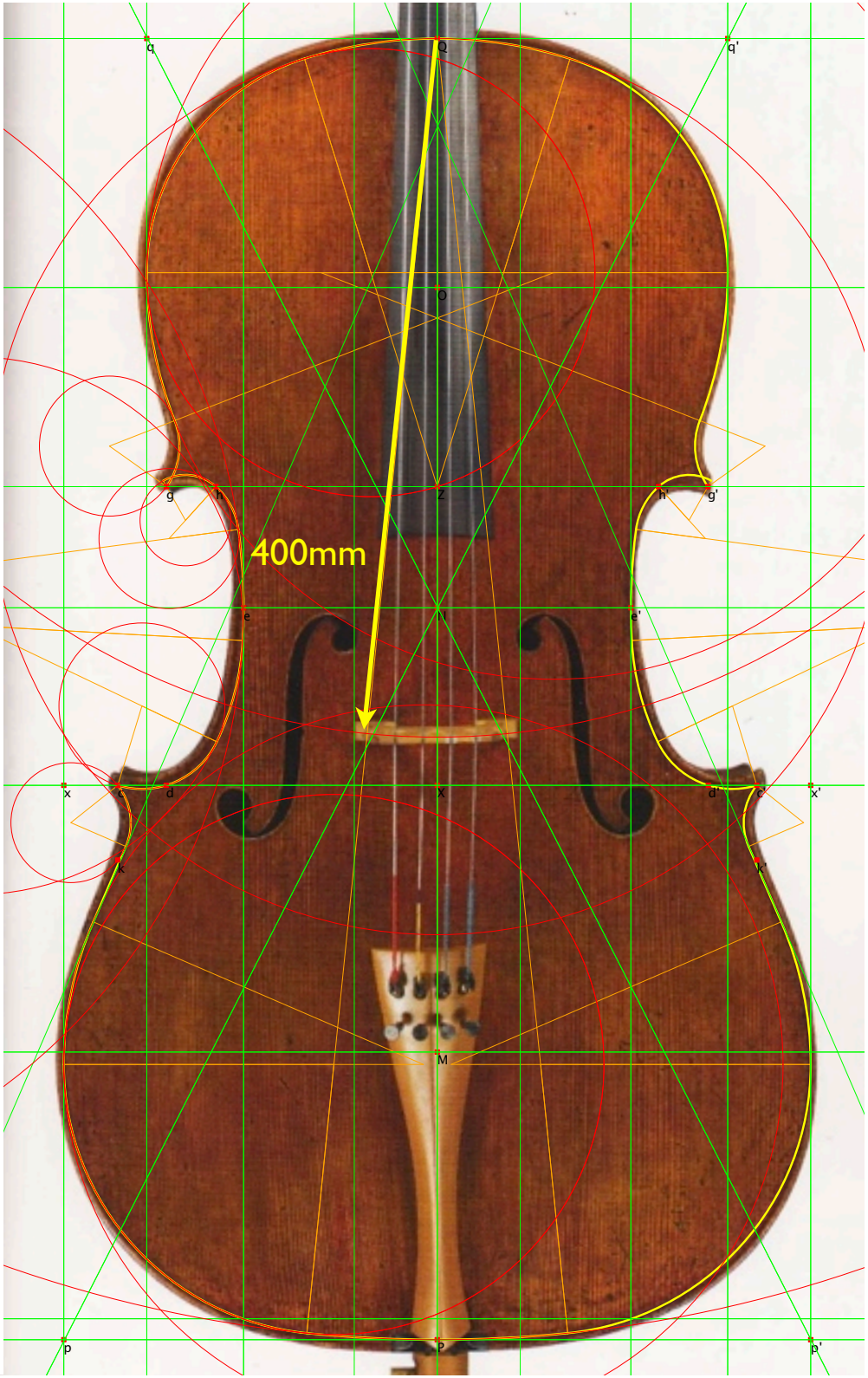
Stanlein via *Mediceo*

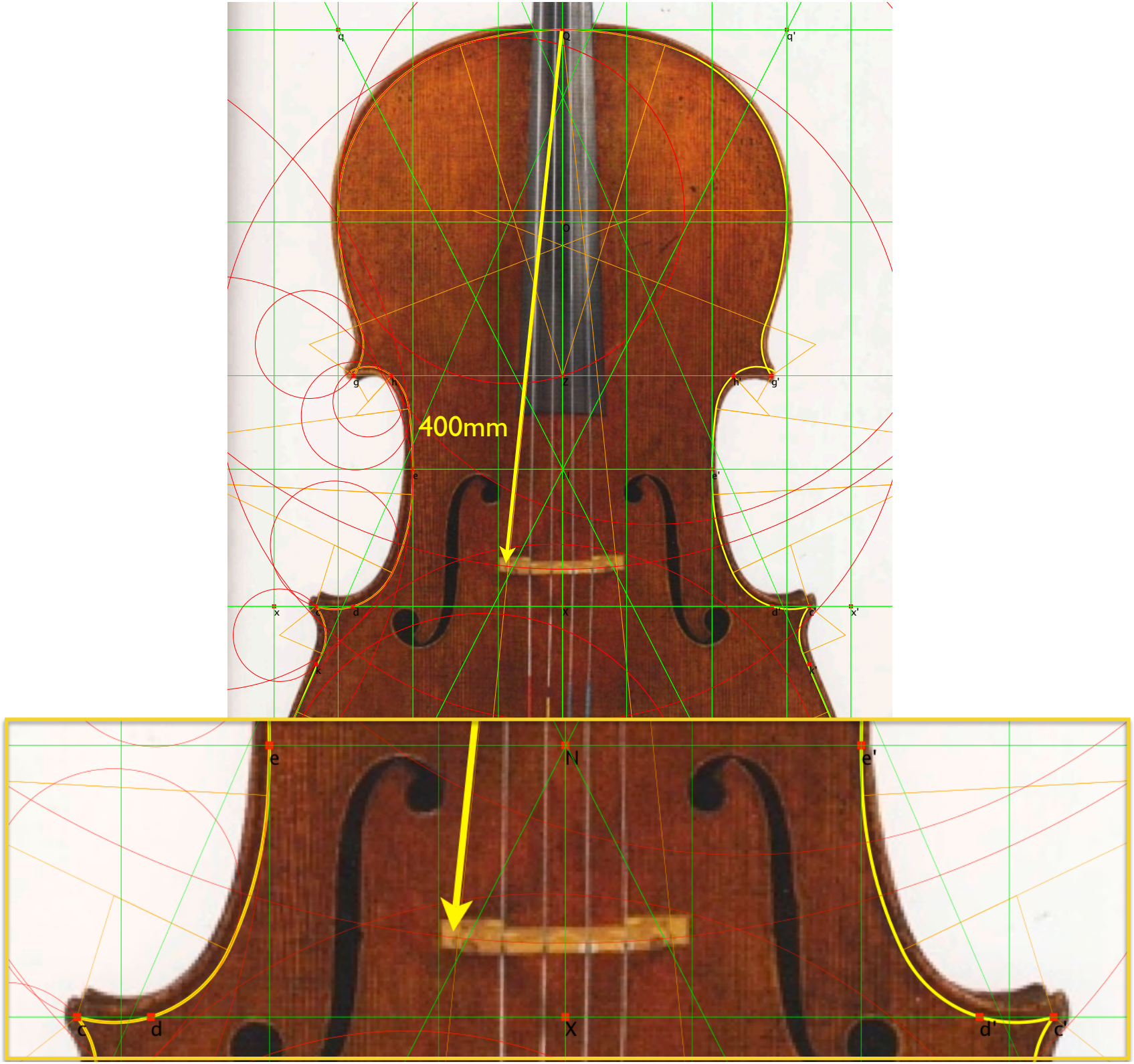
first, the original outline…



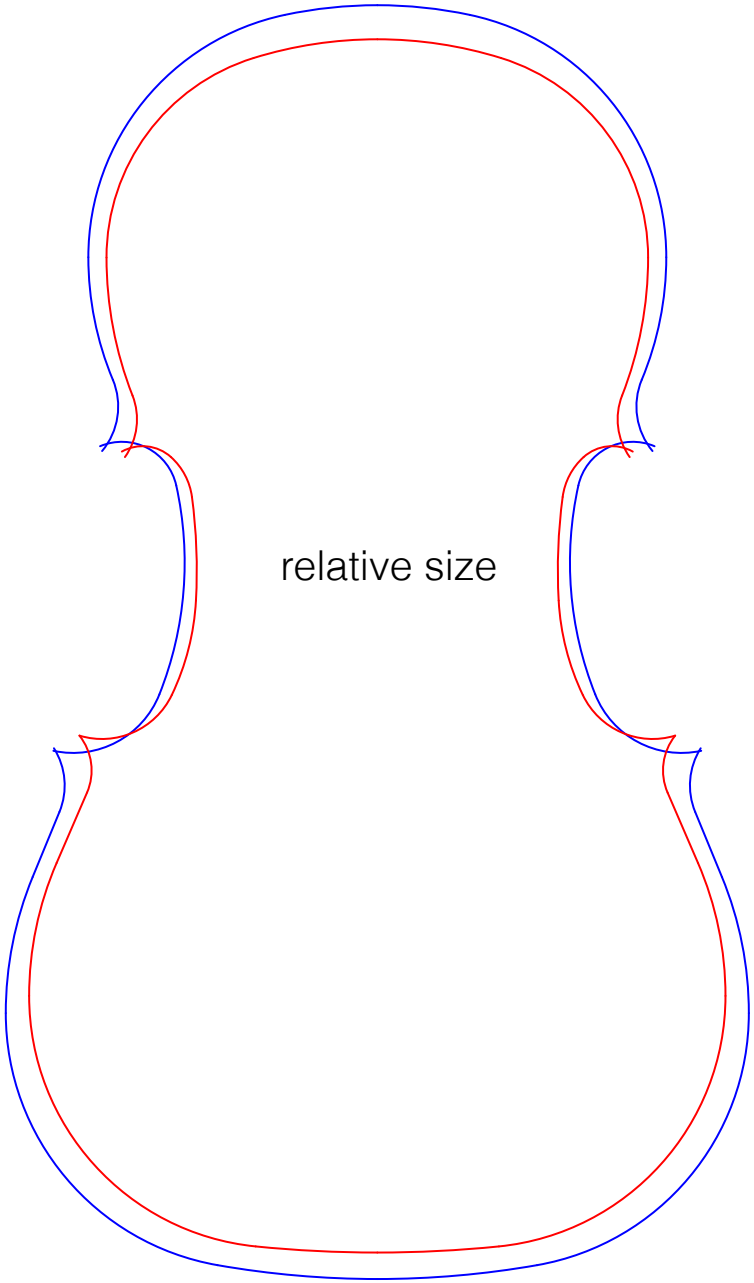
Stanlein via *Mediceo*

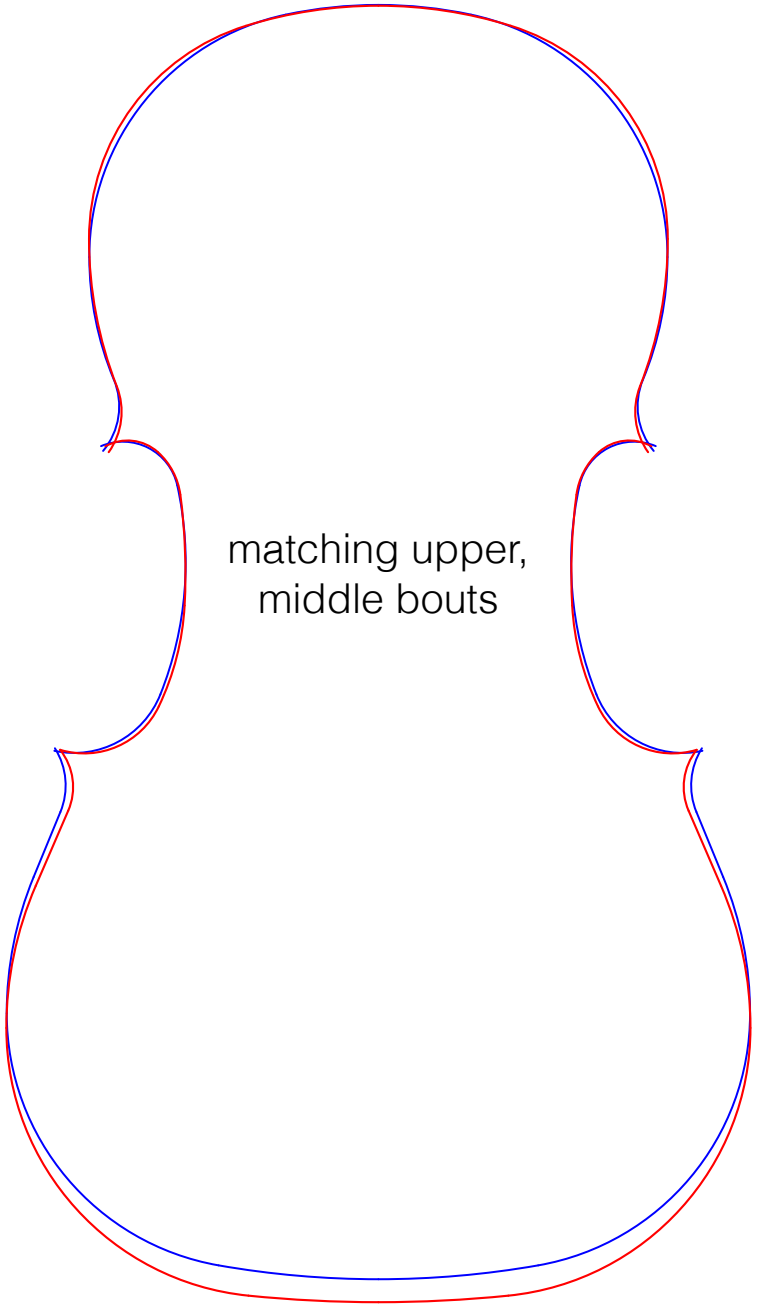



400mm
Q
q
q'
O
Z
g
h
h'
g'
e
e'
I
x
d
X
d'
x'
k
k'
M
p
P
p'

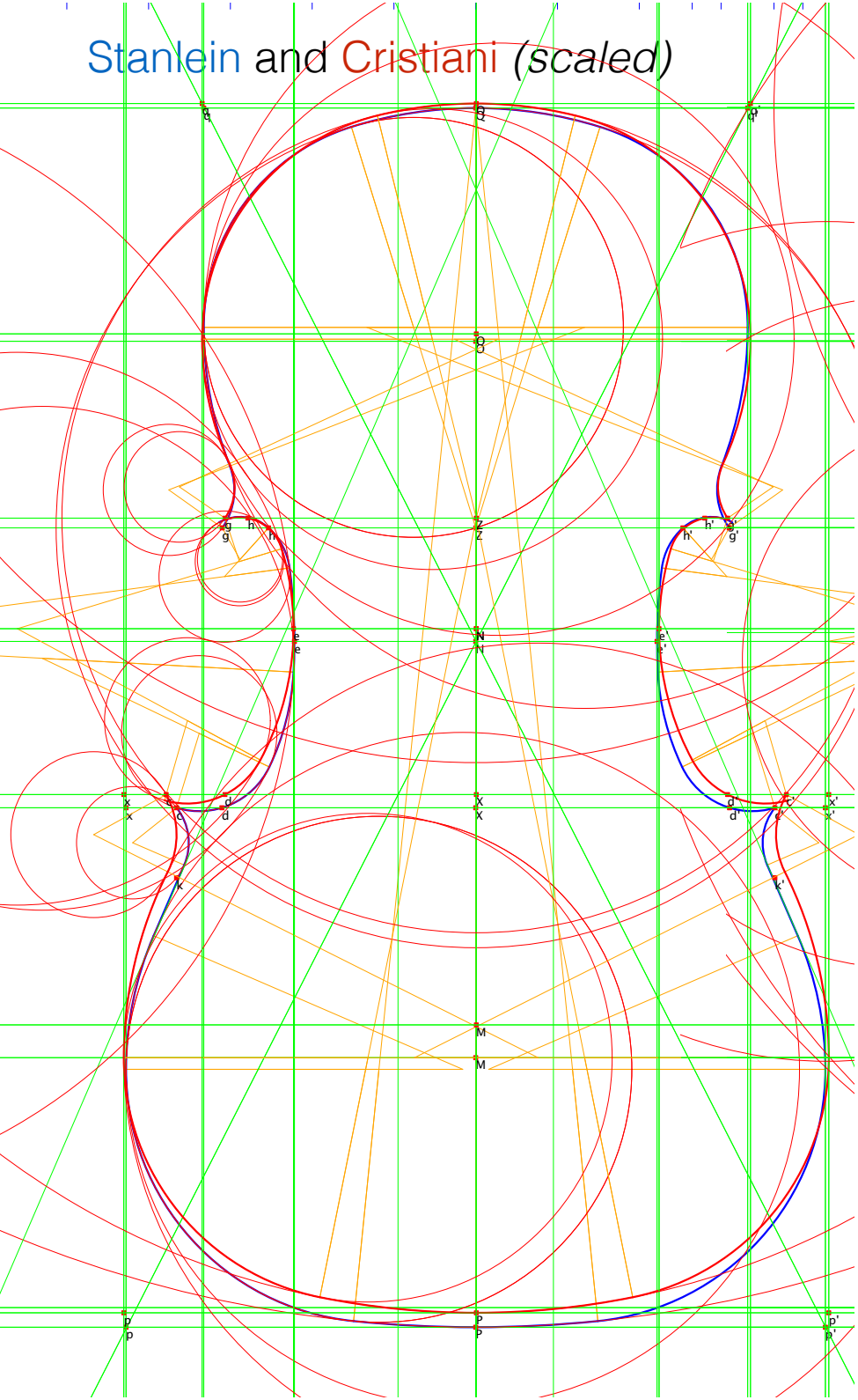

400mm
e
N
e'
c
d
X
d'
c'

Mediceo Stanlein

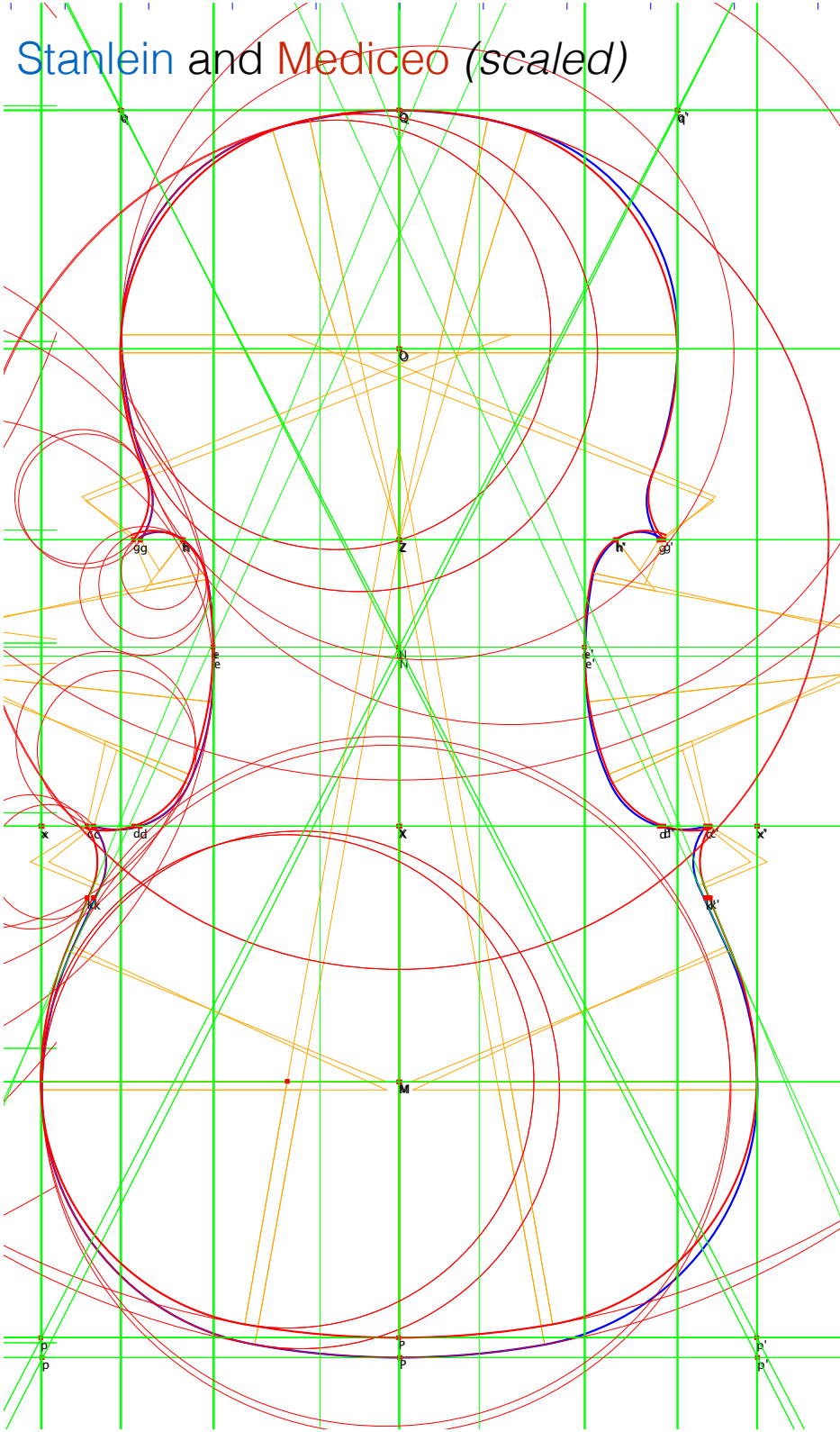


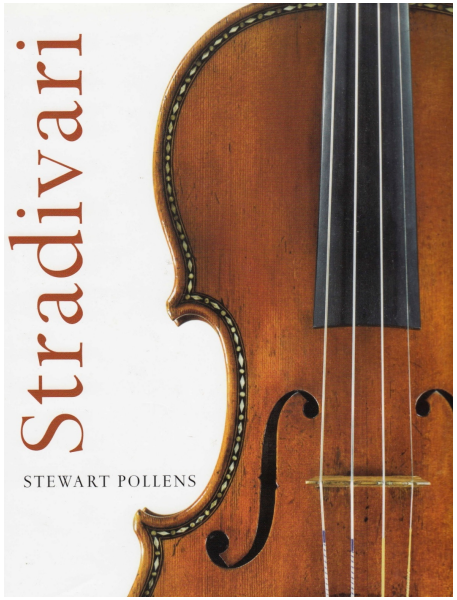


Stanlein and Cristiani *(scaled)*



Stanlein and Mediceo *(scaled)*





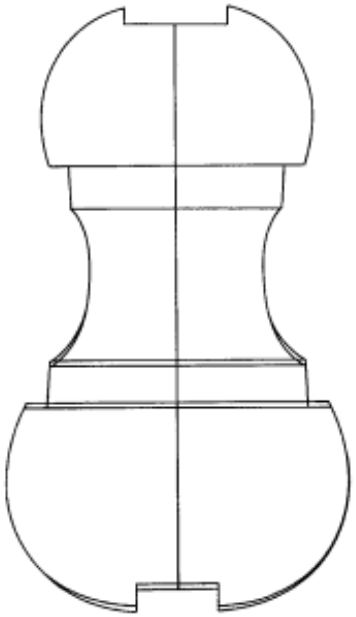


Fig. 3.9 Overlay of tracings of Stradivari's violin forms S and SL (MS nos. 2 and 28).

1707. The top-block recess of the undated S form has been altered in width and height: the plugged recess was once wider and shallower than the present opening. While this might

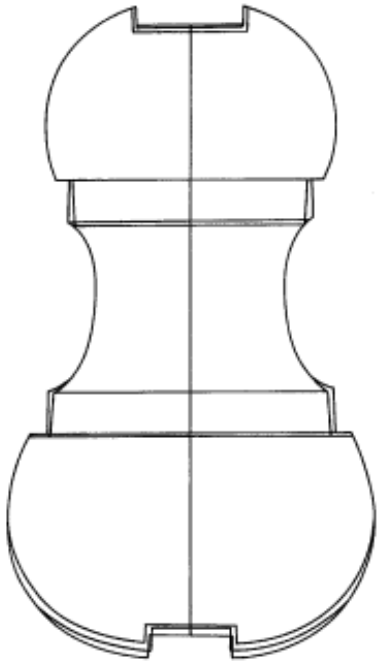
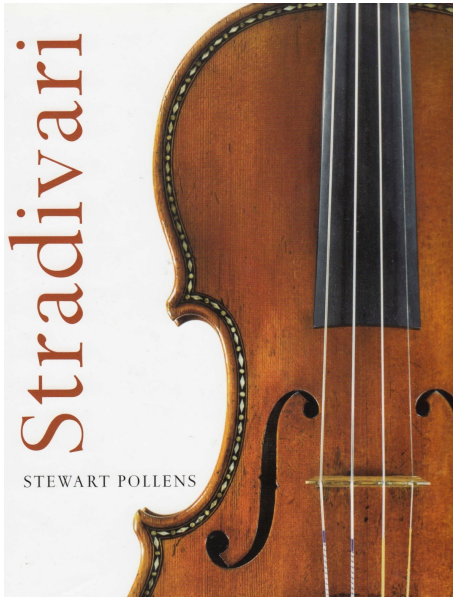


Fig. 3.10 Overlay of tracings of Stradivari's violin forms MB, B, and B (MS nos. 1, 33, and 38).

3.10). The form marked *MB* (possibly the abbreviation for *modello buono*, MS no. 1) is



"The line drawings were made by placing a sheet of translucent plastic film over life-size photographic enlargements of the forms and tracing them. **The images of the forms were not adjusted to different proportions.** I made the drawings the old-fashioned way: with a t-square, triangles, and drafting pens guided by a flexible curve."

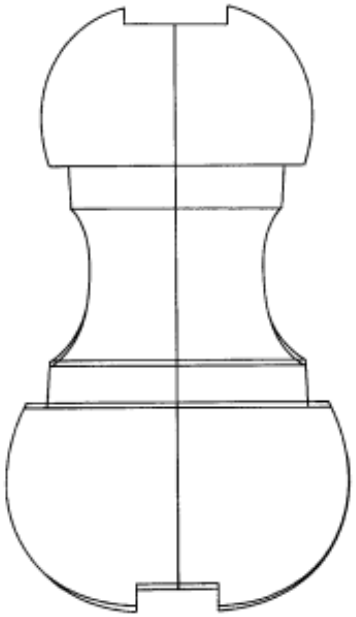


Fig. 3.9 Overlay of tracings of Stradivari's violin forms S and SL (MS nos. 2 and 28).

1707. The top-block recess of the undated S form has been altered in width and height: the plugged recess was once wider and shallower than the present opening. While this might

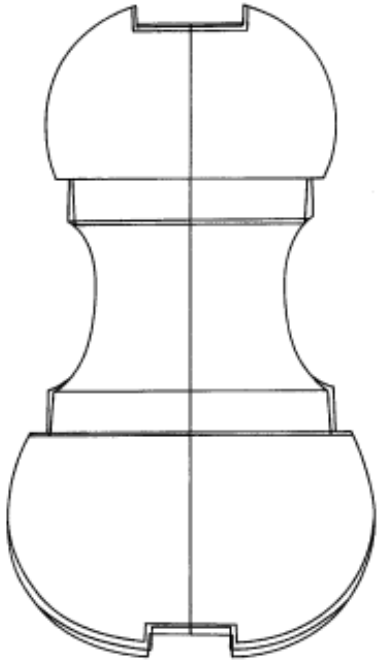
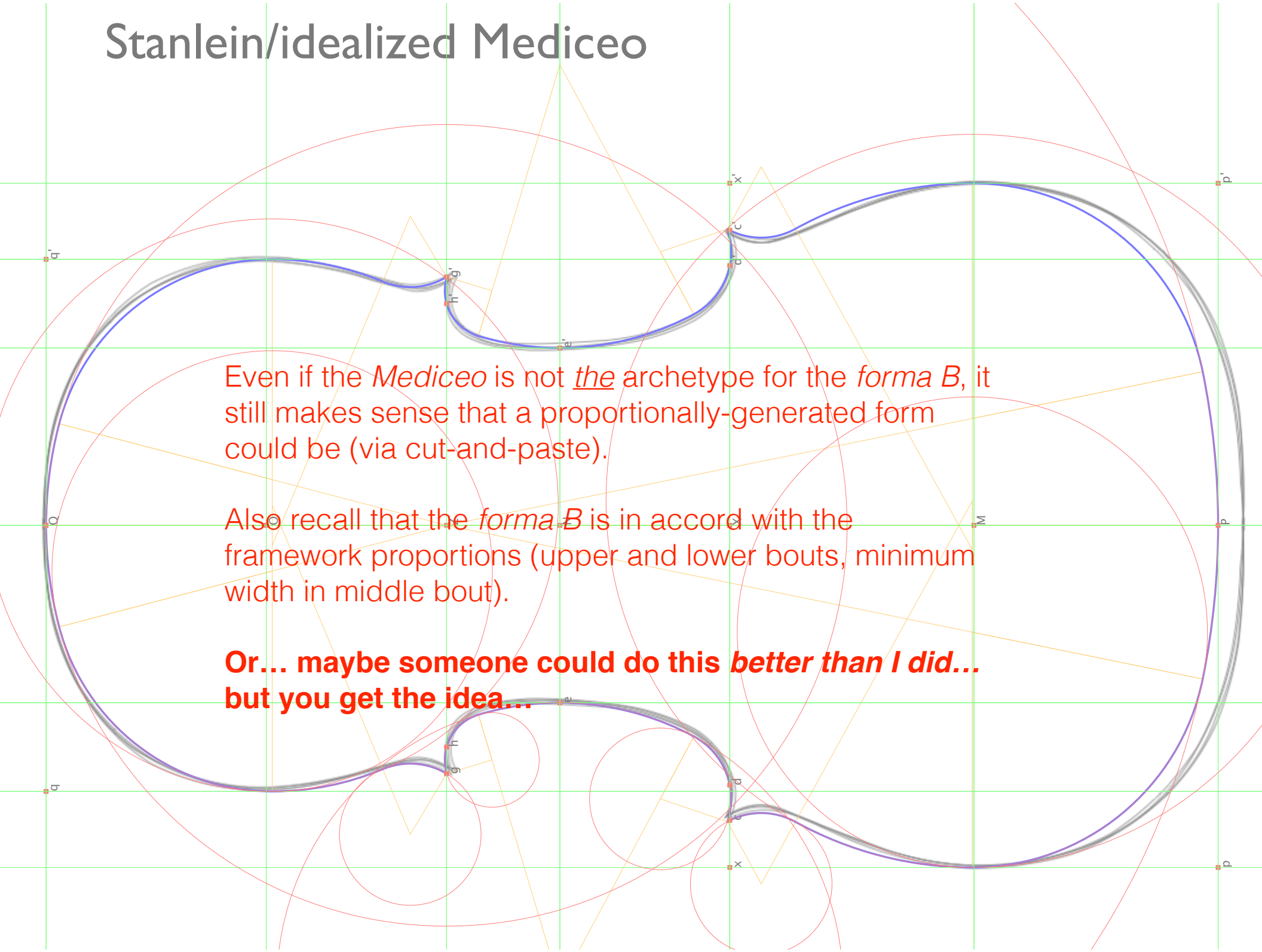


Fig. 3.10 Overlay of tracings of Stradivari's violin forms MB, B, and B (MS nos. 1, 33, and 38).

3.10). The form marked *MB* (possibly the abbreviation for *modello buono*, MS no. 1) is

Stanlein/idealized Mediceo



Even if the *Mediceo* is not *the* archetype for the *forma B*, it still makes sense that a proportionally-generated form could be (via cut-and-paste).

Also recall that the *forma B* is in accord with the framework proportions (upper and lower bouts, minimum width in middle bout).

Or… maybe someone could do this *better than I did…* but you get the idea…

How to…

Come talk to me or email me **(harry.mairson@verizon.net)**

Download and install **Racket** from (for example)
`http://racket-lang.org/download/`

Get my code (I'll send it to you if you want)...

Play around with the constructions (cellos by Guarneri and Montagnana, Stradivari, violin by Amati, lute by Arnault)---try modifying parameters and see what happens...

Pretend you know how to program (follow pattern of preloaded designs). Like making web pages from pirated HTML...

Thanks to...

François Denis

(design,construction)

Marilyn Wallin

(construction)

Curtis Bryant

(identification, restoration, construction)

David Van Horn

(Ph.D. student, now professor at University of Maryland)

Villa Borghese, Rome