# Rigorous interval computations of hyperbolic tetrahedral shapes

Neil Hoffman
joint with Kazuhiro Ichihara (Nihon University), Masahide Kashiwagi (Waseda), Hidetoshi Masai (Tokyo Tech), Shin'ichi Oishi (Waseda), and Akitoshi Takayasu (Waseda)

Max Planck Institut für Mathematik - Bonn
people.mpim-bonn.mpg.de/nhoffman

July 16, 2013

# Outline:

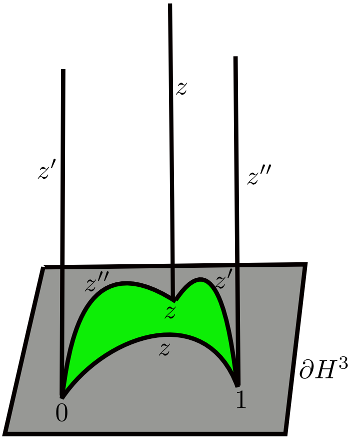
- Hyperbolic gluing equations
- Interval arithmetic
- Two ways to verify Newton's method
- Applications to 3-manifolds

# Main Question

How can we use a computer to rigorously verify a hyperbolic structure on a 3-manifold?

# Hyperbolic ideal tetrahedra

Up to similarity, each ideal tetrahedron in $H^3$ can be parametrized by a complex number.




Here, $z' = \frac{z-1}{z}$ and $z'' = \frac{1}{1-z}$.

# Gluing equations

We want to show that there is a solution the hyperbolic gluing equations.

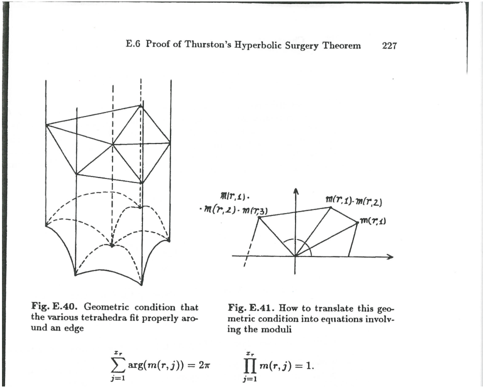





**Fig. E.40.** Geometric condition that the various tetrahedra fit properly around an edge

**Fig. E.41.** How to translate this geometric condition into equations involving the moduli

$$\sum_{j=1}^{z_r} \arg(m(r,j)) = 2\pi \qquad \prod_{j=1}^{z_r} m(r,j) = 1.$$

Figure: Benedetti and Petronio: Lectures on Hyperbolic Geometry page 227

# Three types of gluing equations

These equations are implemented and solved using software like Snappy (based on the Snappea Kernel). For a manifold $M$ with $n$ tetrahedra, $m$ unfilled cusps and $c$ filled cusps, we have:

Edge equations (n of these):

$$\sum_{j=1}^{n}(a_{j,k}\log(z_j) + b_{j,k}\log(\frac{1}{1-z_j}) + c_{j,k}\log(\frac{z_j - 1}{z_j})) = 0 + 2\pi i$$

Cusp equations (2 m of these):

$$\sum_{j=1}^{n}(a_{j,k}\log(z_j) + b_{j,k}\log(\frac{1}{1-z_j}) + c_{j,k}\log(\frac{z_j - 1}{z_j})) = 0 + 0\pi i$$

Dehn surgery equations (c of these):

$$\sum_{j=1}^{n}(a_{j,k}\log(z_j) + b_{j,k}\log(\frac{1}{1-z_j}) + c_{j,k}\log(\frac{z_j - 1}{z_j})) = 0 + 2\pi i$$

Note: here $arg(z) \in (-\pi, \pi]$ and valid solutions must have $arg(z_j) > 0$ for all $j$.

# Dehn Surgery Equations

If a manifold $M$ is obtained from a $p/q$ filling the cusp of $M'$, then we need to solve:

$$p\sum_{j=1}^{n}(a_{j,k}\log(z_k)+b_{j,k}\log(\frac{1}{1-z_j})+c_{j,k}\log(\frac{z_j-1}{z_j}))+$$

$$q\sum_{j=1}^{n}(a_{j,k+1}\log(z_k)+b_{j,k+1}\log(\frac{1}{1-z_j})+c_{j,k+1}\log(\frac{z_{j+1}-1}{z_{j+1}}))=0+2\pi i$$

Here: equation $k$ corresponds to the meridian equation for this cusp and $k+1$ the longitude equation.

# Restated goal

Goal: 1) Show that an approximated solution to the gluing equations is in a small neighborhood of an actual solution.

2) Precisely define small.

# Interval Arithmetic

Basic Idea: A computer can not easily deal with $\log 3$ as an exact number, but it can compute

$$1.098612 \approx \log 3 \in [1.09765625, 1.1015625].$$

Note: in binary that interval is [1.00011001,1.00011010].
We say $x \in [x]$ and $[x] = [\underline{x}, \bar{x}]$.

# Operations in interval arithmetic

- $+ : [a] + [b] = [\underline{a} + \underline{b} - \epsilon, \bar{a} + \bar{b} + \epsilon']$.
- $- : [a] - [b] = [\underline{a} - \bar{b} - \epsilon, \bar{a} - \underline{b} + \epsilon']$.
- $\cdot : [a] \cdot [b] = [min(a_i \cdot b_j) - \epsilon, max(a_i \cdot b_j) + \epsilon']$.
- $\div : [a] \div [b] = [min(\frac{a_i}{b_j}) - \epsilon, max(\frac{a_i}{b_j}) + \epsilon']$ if $0 \notin [\underline{b}, \bar{b}]$

Here $a_i \in \{\underline{a}, \bar{a}\}$ and $b_j \in \{\underline{b}, \bar{b}\}$ and $\epsilon$, $\epsilon'$ account for a processor's rounding error.

# Rounding for basic arithmetic

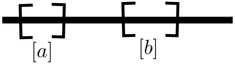
Let $O : \mathbb{R} \to \mathbb{F}$ be a rounding function set be the computer language one uses.
There are four types of rounding $O(x) = x_{approx}$

- $+\infty$ : overestimates the number $O(x) = \lceil x \rceil$
- $-\infty$ : underestimates the number $O(x) = \lfloor x \rfloor$
- nearest : $|x - x_{approx}|$ is smallest
- chopping: $O(x) = \lceil x \rceil$ for $x < 0$ and $\lfloor x \rfloor$ for $x > 0$

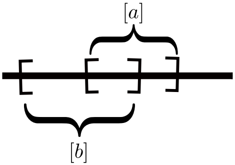
Note: chopping gives best allocation of memory and yields memory overflow errors least often.

# Inequalities

$>, <$ can return *TRUE*, *FALSE*, and *UNDETERMINED*.
Examples:

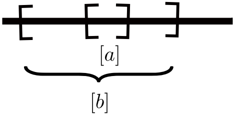



$[a] < [b]$ and $[b] > [a]$




$[a] \not< [b]$ and $[a] \not> [b]$

# Interval inclusion




$$[a] \subset [b]$$

# Approximating functions

To get values of functions we need to give the computer a Taylor (or Maclaurin) series and error bound. For example our program uses log, $\sqrt{}$, $e$ and arctan ($D$ is the domain of convergence).

$$f([z]) = \sum_{k=1}^{n} \frac{f^k([z])}{k!} \text{ for } [z] \subset D$$

and error

$$|E_k([z])| < \frac{M \cdot |[z]|^{k+1}}{(k+1)!}, |f^{k+1}([z])| < M$$

# Exact arithmetic

Given a number field $k$ such that $[k : \mathbb{Q}] = d$, we can represent $k$ as a $d$-dimensional vector space over $\mathbb{Q}$. Here, $\mathbb{Q}[x]/(x^d = a_{d-1}x^{d-1} + \ldots + a_0)$ $a_i \in \mathbb{Q}$.

Recording $\zeta \in \mathbb{Q}$ as $(\zeta_1, \zeta_2, \ldots, \zeta_d)$, $\zeta_i \in \mathbb{Z} \times \mathbb{Z}$. Given enough precision for $\zeta$, it is possible to verify an exact solution to the gluing equations with enough memory (and luck).

Used by snap.

# Advantages of interval arithmetic

- Fast (especially compared to exact arithmetic)
- Uses less memory than exact arithmetic
- Relatively easy to program
- Overwrites the $+.-, \cdot, \div$ functions
- Extends to functions naturally.
- Keeps track of accumulated error by itself
- There are rigorous verification test of Newton's method for interval arithmetic

# Krawczyk Test (statement due to Rump(1983))

**Theorem**

*Given a continuously differentiable $f : D \to \mathbb{R}^{2n}$, $\tilde{x} \in \mathbb{R}^{2n}$, $X = [x_1] \times [x_2] \times ...[x_{2n}]$ with $\vec{0} \in X$ and $\tilde{x} + X \subseteq D$, and $R \in \mathbb{R}^{2n} \times \mathbb{R}^{2n}$. Suppose*

$$S(X, \tilde{x}) = -Rf(\tilde{x}) + \{I - RJ_f(\tilde{x} + X)\}X \subset Int(X).$$

*Then $R$ and all matrices $M \in J_f(\tilde{x} + X)$ are non-singular and there is a unique root $\hat{x}$ of $f$ in $\tilde{x} + S(X, \tilde{x})$.*

Note: here $R \approx J_f^{-1}$.

# Krawczyk Test vs. Kantorovich test

The Krawczyk test requires fewer computations. Consequently, it is faster and requires less memory than the Kantorovich test.

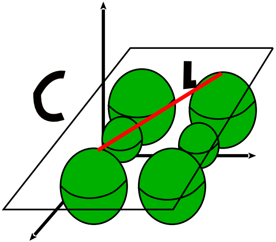
# Implementation

These methods have been used to verify:
the $\leq 5, 6, 7, 8$ tetrahedral censuses (5,6,7 due to Callahan, Hildebrand, and Weeks, 8 due to Thistlethwaite)
and
all but 439 of the manifolds in the closed census (Hodgson and Weeks).

# Implemented in

- MATLAB - big function library, requires a license
- c++ - fast, accurate and free, but hard to use
- python - not fast, not as accurate, but free and easy

# Parabolic length




For $M = \mathbb{H}^3/\Gamma$ with one cusp, we measure the length of a parabolic $p \in \Gamma$ that fixes $\infty$, by measuring it's displacement in the boundary of a maximal horoball.

In general, the length of a parabolic will be the length of its conjugate that fixes $\infty$.

# 6 - Theorem

**Theorem (Agol,Lackenby + Perelman)**

*Let $M$ be a 1-cusped hyperbolic manifold. If $\gamma$ is a parabolic element of length $\geq 6$, then $M(\gamma)$ is hyperbolic.*

Given a solution to the gluing equations, length is a function of the $z_i$.

# Martelli, Petronio, Roukema + $\epsilon$

Martelli, Petronio and Roukema (2012) created an algorithm that used approximate tetrahedral shapes to measure peripheral length.
With $\epsilon$ more work, this can be promoted to a rigorous computation.
This was used to verify 429 of the outstanding 439 closed manifolds have hyperbolic structures.
And is currently being used to identify,

# Currently rigorously Snappy

- Fundamental group presentation, homology, etc.
- *M.is_isometric_to*(*N*) returns TRUE only if a simplicial map is found.

# Further work

- Incorporate into Snappy
- Use these methods to verify topological invariants like: volume, tilt parameter, length spectrum.

Thank you!