The Achilles' Heel of O(3,1)?

William Floyd, Brian Weber, and Jeffrey Weeks

CONTENTS

2000 AMS Subject Classification: Primary 57M50; Secondary 65G50

Keywords: Hyperbolic geometry, computation, numerical error, PSL(2,$\mathbb{C}$), O(3,1)

What's the best way to represent an isometry of hyperbolic 3-space $\mathbb{H}^3$? Geometers traditionally worked in $\mathrm{SL}(2,\mathbb{C})$, but for software development many now prefer the Minkowski space model of $\mathbb{H}^3$ and the orthogonal group O(3,1). One powerful advantage is that ideas and computations in S^3 using matrices in O(4) carry over directly to $\mathbb{H}^3$ and O(3, 1). Furthermore, O(3,1) handles orientation reversing isometries exactly as it handles orientation preserving ones. Unfortunately in computations one encounters a nagging dissimilarity between O(4) and O(3,1): while numerical errors in O(4) are negligible, numerical errors in O(3,1) tend to spiral out of control. The question we ask (and answer) in this article is, "Are exponentially compounded errors simply a fact of life in hyperbolic space, no matter what model we use? Or would they be less severe in $\mathrm{SL}(2,\mathbb{C})$?" In other words, is numerical instability the Achilles' heel of O(3,1)?

1. INTRODUCTION

What's the best way to represent an isometry of hyperbolic 3-space $\mathbb{H}^3$? More often than not, geometers have chosen to work in $\mathrm{SL}(2,\mathbb{C})$. With the advent of widespread computing, however, many geometers have abandoned the Poincaré ball model and switched their thinking to the Minkowski space model of $\mathbb{H}^3$. That is, they visualize $\mathbb{H}^3$ as the set $\{v \in \mathbb{E}^{1,3} | \langle v, v\rangle = -1 \text{ and } v_o > 0\}$, where $\mathbb{E}^{1,3}$ is the Minkowski space defined as $\mathbb{R}^4$ with a metric of signature $(-+++)$. The isometries of $\mathbb{H}^3$ are then just matrices in the orthogonal group $\mathrm{O}(3,1)$.

The orthogonal group $\mathrm{O}(3,1)$ has some powerful advantages over $\mathrm{SL}(2,\mathbb{C})$. First and foremost are the strong ties between the Minkowksi space model of $\mathbb{H}^n$ and the usual picture of an n-sphere as $\{v \in \mathbb{E}^{n+1} | \langle v, v\rangle = +1\}$. Most theorems you prove about the geometry of S^n transfer easily to theorems about $\mathbb{H}^n$, and indeed their proofs transfer almost word-for-word, the only modifications being the introduction of an occasional minus sign. Similarly, most computations that you do in S^n using matrices in $\mathrm{O}(n+1)$ carry over directly to $\mathbb{H}^n$ using matrices in $\mathrm{O}(n,1)$. For example, in computer graphics, modern PCs all come with 3D graphics cards that are hard-wired

to do the matrix operations necessary for real time animations in $\mathbb{E}^3$. These same off-the-shelf graphics cards, when fed transformation matrices in $O(4)$ or $O(3,1)$, will correctly render scenes in S^n or $\mathbb{H}^n$, because the computations are the same [Weeks 01b].

A further advantage of $O(3,1)$ over $SL(2,\mathbb{C})$ is $O(3,1)$'s egalitarian approach to orientation reversing isometries, which it handles exactly as it handles orientation preserving ones. $SL(2,\mathbb{C})$, by contrast, is built for the orientation preserving case, and handles orientation reversing isometries awkwardly, via the complex conjugate, i.e. an orientation reversing isometry is given by $z \mapsto (a\bar{z}+b)/(c\bar{z}+d)$. (This awkwardness may partially explain the shameless neglect of nonorientable 3-manifolds in much of the existing literature.)

Many geometers, especially those with an interest in computing, have embraced $O(3,1)$ for most or all of their work, generally with good results. However, when one gets into the thick of the computations one encounters a nagging dissimilarity between $O(4)$ and $O(3,1)$. While numerical errors in $O(4)$ are generally negligible, numerical errors in $O(3,1)$ have the nasty habit of spiraling out of control, especially if the computation involves matrices with large entries. The question we ask (and answer) in this article is "Are the problems with numerical error the Achilles' heel of $O(3,1)$?" In other words, would the problems with numerical error be less severe in $SL(2,\mathbb{C})$? Or are exponentially accumulating errors simply a fact of life in hyperbolic space, no matter what model we use?

To answer this question, first consider the multiplication of two real numbers x and y. In theory the answer is simply the product xy. But in numerical computations we know the factors only approximately, with errors Δx and Δy. The computed product is $(x+\Delta x)(y+\Delta y) = xy + x\Delta y + y\Delta x + \Delta x\Delta y$. Keeping only the first order error terms and dropping the negligible $\Delta x\Delta y$, the computed product becomes $xy + x\Delta y + y\Delta x$.

Now consider the product of two $n \times n$ matrices M and N. The ik^{th} entry in the product MN is given by the sum

$$\sum_j (M_{ij}+\Delta M_{ij})(N_{jk}+\Delta N_{jk}) \simeq \sum_j \left(M_{ij}N_{jk} + M_{ij}\Delta N_{jk} + \Delta M_{ij}N_{jk}\right).$$

If the original matrices M and N have entries of magnitude 1 (as is the case, for example, with rotation matrices in $O(n)$), then the errors in the computed product are similar to the errors in the factors, and all is well. If, on the other hand, the original matrices have large entries (as is often the case, for example, with matrices in $SL(2,\mathbb{C})$ or $O(3,1)$), then the errors in the computed product are larger than the errors in the factor matrices, in proportion to the magnitude of the factor matrix entries. For example, if the matrices M and N have entries of order 1000, accurate to within an error ϵ, then the product MN will have entries accurate to within $1000 \times 2n\epsilon$. This result isn't so bad if M and N are chosen arbitrarily, because the product MN will have entries of order 1000000, and the fractional error $2000n\epsilon/1000000 = 2n\epsilon/1000$ in the product will be roughly the same as the fractional error $\epsilon/1000$ in the factors. In reality, though, we don't choose our matrices M and N arbitrarily. On the contrary, many computations in hyperbolic space require checking whether the product of several matrices is the identity. In such cases we might, say, multiply two matrices M and N with entries of order 1000 and get a product MN with entries of order 1, yet still have errors in the product of order 1000ϵ! This effect—getting a matrix with small entries but large errors—is what makes numerical computation in hyperbolic space difficult.

If we multiply not two but three matrices, the effect is even more pronounced. Say we need to compute the product of three matrices M, N, and P, each of which has entries of order 1000, with errors of order ϵ. The entries in the partial product MN will, as shown above, have errors of order 1000ϵ, and so the entries in the final product MNP will have errors of order 1000000ϵ. More generally, the size of the errors grows exponentially with the number of matrix multiplications.

Both $O(3,1)$ and $SL(2,\mathbb{C})$ contain matrices with large entries, and both suffer the aforementioned problem with error compounding. The question is, does $O(3,1)$ suffer worse than $SL(2,\mathbb{C})$? As an example, consider the $O(3,1)$ matrix

$$\begin{pmatrix} \cosh d & \sinh d & 0 & 0 \\ \sinh d & \cosh d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

and the corresponding $SL(2,\mathbb{C})$ matrix

$$\begin{pmatrix} e^{d/2} & 0 \\ 0 & e^{-d/2} \end{pmatrix},$$

each of which describes a translation of distance d along a geodesic through the origin in hyperbolic 3-space. The entries in the $O(3,1)$ matrix are of order at most e^d, while those in the $SL(2,\mathbb{C})$ matrix are of order at most $e^{d/2}$. In the next section we'll see that this example is

typical: matrices in $O(3,1)$ with entries of order e^d correspond to matrices in $SL(2,\mathbb{C})$ with entries of order $e^{d/2}$. The difference between e^d and $e^{d/2}$ is enormous for all but the smallest values of d, and we conclude that errors compound much more slowly in $SL(2,\mathbb{C})$ than in $O(3,1)$. Accumulating error really is the Achilles' heel of $O(3,1)$.

2. THE TWO ACTIONS AND THE ESTIMATES

One of the standard models of hyperbolic 3-space is the upper half-space of $\mathbb{R}^3$, considered as the subspace $U = \{z + tj\colon z \in \mathbb{C}, t > 0\}$ of the quaternions, with metric $\frac{ds^2}{t^2}$. The group $SL(2,\mathbb{C})$ of 2×2 complex matrices with determinant 1 acts as isometries on U by

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}(z+tj) = (a(z+tj)+b)(c(z+tj)+d)^{-1}.$$

The kernel of this action is $\{\pm I\}$, and the quotient group $PSL(2,\mathbb{C})$ is isomorphic to the group of orientation-preserving isometries of $\mathbb{H}^3$. See, for example, [Beardon 83, Chapter 4].

Another standard model for $\mathbb{H}^3$ is the hyperboloid model. Consider the Minkowski space $\mathbb{E}^{1,3}$, which as a topological space is just $\mathbb{E}^4$ but which is equipped with the inner product $\langle\ ,\ \rangle$ defined by

$$\langle (v_0,v_1,v_2,v_3),(w_0,w_1,w_2,w_3)\rangle = -v_0w_0 + v_1w_1 + v_2w_2 + v_3w_3.$$

The group $O(3,1)$ acts on $E^{1,3}$ preserving the inner product, and each element of $O(3,1)$ preserves the hyperboloid $\{v \in E^{1,3}\colon \langle v,v\rangle = -1\}$. The hyperboloid model for $\mathbb{H}^3$ is the top sheet $H = \{v \in E^{1,3}\colon \langle v,v\rangle = -1 \text{ and } v_0 > 0\}$ of this hyperboloid, with metric $ds^2 = -dv_0^2 + dv_1^2 + dv_2^2 + dv_3^2$. The set of elements of $O(3,1)$ that preserve H is a subgroup $O^+(3,1)$ of $O(3,1)$ of index 2, and this subgroup is the isometry group of H (and hence of $\mathbb{H}^3$). The group of orientation-preserving isometries of $\mathbb{H}^3$ is the subgroup $SO^+(3,1)$ of $O^+(3,1)$ of elements of $O(3,1)$ with determinant 1.

Before giving the estimates, we first compute the stabilizers of the natural basepoints j and $(1,0,0,0)$ of the two actions. Consider an element

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

of $SL(2,\mathbb{C})$. Then $A(j) = j$ if and only if $(aj+b)(cj+d)^{-1} = j$, which is true if $aj + b = j(cj+d) = -\bar{c} + \bar{d}j$. Hence the stabilizer of j is the special unitary group $SU(2)$. It is easy to check that the stabilizer of $(1,0,0,0)$ in $O(3,1)$ is the image of the monomorphism $\iota : O(3) \to O(3,1)$ defined (with abuse of notation) by $\iota(A)(v_0,v_1,v_2,v_3) = (v_0, A(v_1,v_2,v_3))$.

Lemma 2.1. *The maximum modulus of an entry in a matrix $A \in SL(2,\mathbb{C})$ lies in the interval $[\frac{1}{2}e^{d/2} - 1, e^{d/2} + 1]$, where d is the hyperbolic distance that A moves the natural basepoint j in U.*

Proof: If $d = 0$, then $A \in SU(2)$ and the lemma is clear. Suppose $d > 0$. The matrix

$$\tau(d) = \begin{pmatrix} e^{d/2} & 0 \\ 0 & e^{-d/2} \end{pmatrix}$$

translates the basepoint j in U hyperbolic distance d. By composing $\tau(d)$ with a rotation ρ_1, we can use the product $\rho_1\tau(d)$ to translate the basepoint j to any point that is hyperbolic distance d from j. By pre-composing with another rotation ρ_2, we can express any element A of $SL(2,\mathbb{C})$ as a product $\rho_1\tau(d)\rho_2$. Algebraically, the rotations ρ_1 and ρ_2 are given by matrices in $SU(2)$, as explained above. Hence

$$\begin{aligned} A &= \begin{pmatrix} a & -\bar{b} \\ b & \bar{a} \end{pmatrix}\begin{pmatrix} e^{d/2} & 0 \\ 0 & e^{-d/2} \end{pmatrix}\begin{pmatrix} u & -\bar{v} \\ v & \bar{u} \end{pmatrix} \\ &= \begin{pmatrix} aue^{d/2} - \bar{b}ve^{-d/2} & -a\bar{v}e^{d/2} - \bar{b}\bar{u}e^{-d/2} \\ bue^{d/2} + \bar{a}ve^{-d/2} & -b\bar{v}e^{d/2} + \bar{a}\bar{u}e^{-d/2} \end{pmatrix}, \end{aligned}$$

where $|a|^2 + |b|^2 = 1$ and $|u|^2 + |v|^2 = 1$. It is now straightforward to check that every entry of A has modulus less than $e^{d/2} + 1$, and at least one entry has modulus greater than $\frac{1}{2}e^{d/2} - 1$, thus proving the lemma. □

Lemma 2.2. *The maximum absolute value of an entry in a matrix $A \in SO^+(3,1)$ is $\cosh d$, where d is the hyperbolic distance that A moves the natural basepoint $(1,0,0,0)$ in H.*

Proof: Again we can assume that $d > 0$, since if $d = 0$ then $A \in \iota(O(3))$ and the lemma is clear. The matrix

$$T(d) = \begin{pmatrix} \cosh d & \sinh d & 0 & 0 \\ \sinh d & \cosh d & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

translates the basepoint $(1,0,0,0)$ in H hyperbolic distance d. By composing $T(d)$ with an element of $\iota(O(3))$, we can translate $(1,0,0,0)$ to any point that is hyperbolic distance d from $(1,0,0,0)$. By pre-composing with another element of $\iota(O(3))$, we can express any element

A of $\mathrm{SO}^+(3,1)$ as a product $\rho_1 T(d)\rho_2$, where ρ_1 and ρ_2 are in $\iota(\mathrm{O}(3))$. It is now straightforward to check that the maximum absolute value of each entry of such a matrix $\rho_1 T(d)\rho_2$ is at most $\cosh d$ and the $1,1$ entry is exactly $\cosh d$. □

To use the lemmas, we need to know how we'll compare matrices in the two groups. Since we've only seen one isomorphism between $\mathrm{PSL}(2,\mathbb{C})$ and $\mathrm{SO}^+(3,1)$ given in print, we will make our comparisons with that one and will call it the *standard isomorphism*. The construction of the isomorphism, which is via an action of $\mathrm{SL}(2,\mathbb{C})$ on the space of 2×2 Hermitian matrices, can be found, for example, in [Shafarevich 90, Section 15] and in [Thurston 97, Section 2.6]. An implementation of this isomorphism can be found in the source file matrix_conversion.c of SnapPea [Weeks 01a]. We will not construct the standard isomorphism here, but note that it is easy to see from the construction that it preserves the distance that an element moves the natural basepoint. Hence matrices in $\mathrm{SL}(2,\mathbb{C})$ with entries of order $e^{d/2}$ correspond under the standard isomorphism to matrices in $\mathrm{O}(3,1)$ with entries of order e^d.

3. EXAMPLES

Our expectation from the estimates is that as the translation distances increase the computational errors in taking products in $\mathrm{O}(3,1)$ will be drastically worse than the computational errors in taking the products of the corresponding matrices in $\mathrm{PSL}(2,\mathbb{C})$. But above we just gave upper bounds, and it is conceivable that in actual practice the numerical errors could be much less. We give some examples to illustrate what actually happens. All examples were done in Mathematica[1] Version 4.0.2.0 on a IBM-compatible PC with a Pentium P3 processor and Windows 2000. All but the last example were computed with floating point arithmetic with 16 decimal digits of precision.

One sees dramatic errors just in comparing the product $\tau(d_1)\tau(d_2)$ and the product $T(d_1)T(d_2)$, its image under the standard isomorphism. In particular, consider the product $\tau(d)\tau(-d)$ and the product $T(d)T(-d)$. If d is large, then a computer using floating point arithmetic can no longer distinguish between $\cosh d$ and $\sinh d$, and so the latter product will not be close to the identity. But in some sense this is an unfair comparison, because $\tau(d)$ is diagonal and there is no appreciable error in $\mathrm{PSL}(2,\mathbb{C})$ in taking the product of the diagonal matrices $\tau(d)$ and $\tau(-d)$. So let us consider the product AA^{-1} for a generic matrix A in $\mathrm{PSL}(2,\mathbb{C})$ or $\mathrm{SO}^+(3,1)$.

In order to construct a generic element of $\mathrm{Isom}(\mathbb{H}^3)$, it is convenient to choose generators for the rotation group $\mathrm{Isom}(S^2)=\mathrm{SU}(2)$ and for its image under the standard isomorphism $f\colon \mathrm{PSL}(2,\mathbb{C})\to\mathrm{SO}^+(3,1)$. Given $t\in\mathbb{R}$, let

$$r_1(t)=\begin{pmatrix}\cos t & \sin t\\ -\sin t & \cos t\end{pmatrix}, R_1(t)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \cos 2t & -\sin 2t & 0\\ 0 & \sin 2t & \cos 2t & 0\\ 0 & 0 & 0 & 1\end{pmatrix},$$

$$r_2(t)=\begin{pmatrix}\cos t & i\sin t\\ i\sin t & \cos t\end{pmatrix}, R_2(t)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & \cos 2t & 0 & \sin 2t\\ 0 & 0 & 1 & 0\\ 0 & -\sin 2t & 0 & \cos 2t\end{pmatrix},$$

$$r_3(t)=\begin{pmatrix}e^{ti} & 0\\ 0 & e^{-ti}\end{pmatrix}, \text{ and } R_3(t)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & \cos 2t & \sin 2t\\ 0 & 0 & -\sin 2t & \cos 2t\end{pmatrix}.$$

These matrices describe rotations of S^2 about the x, y, and z axes. To see that they generate all of $\mathrm{Isom}(S^2)$, note that they suffice to take the north pole to any point of S^2, with any desired rotation angle. It is also easy to check that $R_i(t)=f(r_i(t))$ for $i=1,2,3$.

For concreteness we choose a particular family of matrices, parameterized by the distance d that they move the origin. Let

$$A_P(d)=r_1(\pi/3)\tau(d)r_2(4\pi/3),$$

$$B_P(d)=A(d)^{-1}=r_2(-4\pi/3)\tau(-d)r_1(\pi/3),$$

$$A_O(d)=f(A_P(d))=R_1(\pi/3)T(d)R_2(4\pi/3),$$

and

$$B_O(d)=A_O(d)^{-1}=f(B_P(d))=R_2(-4\pi/3)T(d)R_2(\pi/3).$$

Note that neither

$$A_P(8)=\begin{pmatrix}-\frac{e^4}{4}-\frac{3}{4e^4}i & -\frac{\sqrt{3}}{4e^4}-\frac{\sqrt{3}e^4}{4}i\\ \frac{\sqrt{3}e^4}{4}-\frac{\sqrt{3}}{4e^4}i & -\frac{1}{4e^4}+\frac{3e^4}{4}i\end{pmatrix}$$
$$\approx\begin{pmatrix}-13.6495-0.0137367i & -0.0079309-23.6417i\\ 23.6417-0.0079309i & -0.00457891+40.9486i\end{pmatrix}$$

[1] A computer software system available from Wolfram Research, Inc., 100 Trade Center Drive, Champaign, IL 61820, USA.

nor

$$A_O(8) = \begin{pmatrix} \cosh(8) & -\frac{1}{2}\sinh(8) & 0 & \frac{\sqrt{3}}{2}\sinh(8) \\ -\frac{1}{2}\sinh(8) & \frac{1}{4}\cosh(8) & -\frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{4}\cosh(8) \\ \frac{\sqrt{3}}{2}\sinh(8) & -\frac{\sqrt{3}}{4}\cosh(8) & -\frac{1}{2} & \frac{3}{4}\cosh(8) \\ 0 & -\frac{\sqrt{3}}{2} & 0 & -\frac{1}{2} \end{pmatrix}$$

$$\approx \begin{pmatrix} 1490.48 & -745.239 & 0 & 1290.79 \\ -745.239 & 372.62 & -0.866025 & -645.396 \\ 1290.79 & -645.396 & -0.5 & 1117.86 \\ 0 & -0.866025 & 0 & -0.5 \end{pmatrix}$$

is close to a diagonal matrix. For a fixed integer d, $A_P(d)$, $A_O(d)$, $B_P(d)$, and $B_O(d)$ are first computed symbolically, and then are converted to floating point matrices before computing $A_P(d)B_P(d)$ and $A_O(d)B_O(d)$.

Consider taking the product of two $n \times n$ matrices M and N. Recall that the ik^{th} entry in the product MN is given by the sum

$$\begin{aligned} &\sum_j (M_{ij} + \Delta M_{ij})(N_{jk} + \Delta N_{jk}) \\ &\quad \simeq \sum_j (M_{ij}N_{jk} + M_{ij}\Delta N_{jk} + \Delta M_{ij}N_{jk}) \\ &\quad = \sum_j \left(M_{ij}N_{jk} + M_{ij}N_{jk}\frac{\Delta N_{jk}}{N_{jk}} + M_{ij}N_{jk}\frac{\Delta M_{ij}}{M_{ij}} \right). \end{aligned}$$

Hence the absolute error in each entry of the product MN cannot exceed an upper bound on the order of

$$\mathrm{Max}(M_{ij})\,\mathrm{Max}(N_{ij})\,\epsilon_{max},$$

where ϵ_{max} is the greatest fractional error occurring in the entries of M and N. For optimal data, ϵ_{max} roughly equals the machine precision 5×10^{-16}, but in practice it can be much greater, because errors accumulate as we pass data from one step to the next of a multi-step computation.

Applying the results of the preceding paragraph to the special case that $M = A_P(d)$ and $N = B_P(d)$, we get an error bound on the order of

$$\beta_P(d) = e^{d/2}e^{d/2}10^{-16} = e^d 10^{-16},$$

while for $M = A_O(d)$ and $N = B_O(d)$, we get an error bound on the order of

$$\beta_O(d) = e^d e^d 10^{-16} = e^{2d}10^{-16}.$$

Figure 1 plots the error estimates $\beta_P(d)$ and $\beta_O(d)$ and compares them to the actual errors

$$\mathrm{err}_P(d) = \mathrm{Max}(|A_P(d)B_P(d) - I_2|)$$

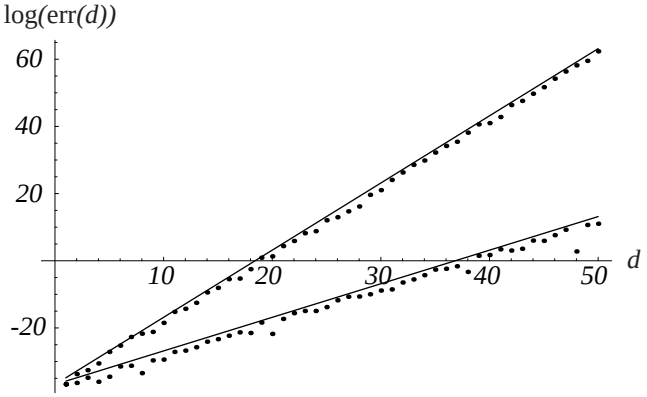

FIGURE 1. The logs of the actual errors $\log(\mathrm{err}_P(d))$ and $\log(\mathrm{err}_O(d))$ for the product AA^{-1} closely track the theoretical estimates $\log(\beta_P(d))$ and $\log(\beta_O(d))$.

and

$$\mathrm{err}_O(d) = \mathrm{Max}(|A_O(d)B_O(d) - I_4|)$$

for $d = 1, \ldots, 50$. The actual errors track the theoretical estimates fairly closely, and it is clear from the data that the numerical error is much greater for the product in $\mathrm{SO}^+(3,1)$.

Now consider the error in evaluating the product of matrices corresponding to a relator R in a discrete subgroup of $\mathrm{Isom}(\mathbb{H}^3)$. We take a very special case of a relator and consider the products

$$R_P(d) = ((A_P(d)A_P(d))B_P(d))((B_P(d)B_P(d))A_P(d))$$

and

$$R_O(d) = ((A_O(d)A_O(d))B_O(d))((B_O(d)B_O(d))A_O(d)),$$

where $B_P(d) = A_P(d)^{-1}$ and $B_O(d) = A_O(d)^{-1}$ are the same as in the previous example. For a fixed integer d we first compute A_P, A_O, B_P, and B_O symbolically, then convert them to floating point matrices before evaluating the relations $R_P(d)$ and $R_O(d)$. Since $R_P(d)$ and $R_O(d)$ should be the identity, we get estimates of error from

$$\mathrm{err}_P(d) = \mathrm{Max}(|R_P(d) - I_2|) \quad \text{and}$$
$$\mathrm{err}_O(d) = \mathrm{Max}(|R_O(d) - I_4|).$$

Table 1 shows some computations of $\mathrm{err}_P(d)$ and $\mathrm{err}_O(d)$ for $d = 1, \ldots, 14$.

As predicted by the theoretical upper bounds, the errors in the $\mathrm{O}(3,1)$ products are much greater than the errors in the $\mathrm{PSL}(2,\mathbb{C})$ products. Even though the hyperbolic translation distance d is relatively small, in $\mathrm{O}(3,1)$ the error quickly becomes unmanageable while the corresponding errors in $\mathrm{PSL}(2,\mathbb{C})$ stay small.

d	$\mathrm{err}_P(d)$	$\mathrm{err}_O(d)$
1	3.7×10^{-16}	4.6×10^{-16}
2	1.3×10^{-15}	3.8×10^{-14}
3	4.8×10^{-15}	1.2×10^{-12}
4	4.3×10^{-14}	9.2×10^{-11}
5	1.4×10^{-13}	1.6×10^{-8}
6	5.2×10^{-13}	3.8×10^{-7}
7	2.1×10^{-11}	6.7×10^{-5}
8	1.4×10^{-10}	9.5×10^{-4}
9	2.5×10^{-9}	8.6×10^{-2}
10	1.2×10^{-8}	2.1×10^{0}
11	3.3×10^{-8}	1.6×10^{2}
12	6.8×10^{-7}	1.2×10^{4}
13	1.1×10^{-6}	1.3×10^{5}
14	1.4×10^{-5}	5.1×10^{7}

TABLE 1. Computations of $\mathrm{err}_P(d)$ and $\mathrm{err}_O(d)$.

Lemma 2.1, Lemma 2.2, and the first example suggest that $\mathrm{err}_P(d)$ might be comparable to $\mathrm{err}_O(d/2)$. Table 2 gives the ratio $\mathrm{err}_O(d/2)/\mathrm{err}_P(d)$. for $d = 2, 4, 6, 8, 12, 14$.

Recall from the proof of Lemma 2.1 that a matrix $A \in \mathrm{SL}(2, \mathbb{C})$ that translates the natural basepoint j in U hyperbolic distance d can be written as a product $\rho_1 \tau(d) \rho_2$ for some matrices ρ_1 and ρ_2 in SU(2). We next consider the effect of changing the matrices ρ_1 and ρ_2. We fix the translation distance d to be 8. Given integers i_1, i_2, i_3, i_4, i_5, and i_6 in $\{1, 2, 3, 4, 5\}$, we let $\rho_1 = r_1(2\pi i_1/5) r_2(2\pi i_2/5) r_3(2\pi i_3/5)$, $\rho_2 = r_1(2\pi i_4/5) r_2(2\pi i_5/5) r_3(2\pi i_6/5)$, $A_P = \rho_1 \tau(8) \rho_2$, $A_O = f(A_P)$, $B_P = A_P^{-1}$, and $B_O = A_O^{-1}$, where A_O, B_P, and B_O are each computed as the product of seven appropriate matrices corresponding to the decomposition of A_P as a sevenfold product. Each of A_P, A_O, B_P, and B_O are first computed symbolically and are then converted to floating point matrices. The products $R_P = ((A_P A_P) B_P)((B_P B_P) A_P)$ and $R_O = ((A_O A_O) B_O)((B_O B_O) A_O)$ would be identity matrices if we worked with infinite precision. Let $\mathrm{err}_P = \mathrm{Max}(|R_P - I_2|)$ and $\mathrm{err}_O = \mathrm{Max}(|R_O - I_4|)$. Figure 2 shows the scatter plot of points $(\log(\mathrm{err}_P), \log(\mathrm{err}_O))$ for the 5^6 choices of the i_j's. Note that the largest value of err_P is eight orders of magnitude smaller than the smallest value of err_O.

d	$\mathrm{err}_O(d/2)/\mathrm{err}_P(d)$
2	0.36
4	0.90
6	2.4
8	0.64
10	1.4
12	0.56
14	4.8

TABLE 2. The ratio $\mathrm{err}_O(d/2)/\mathrm{err}_P(d)$.

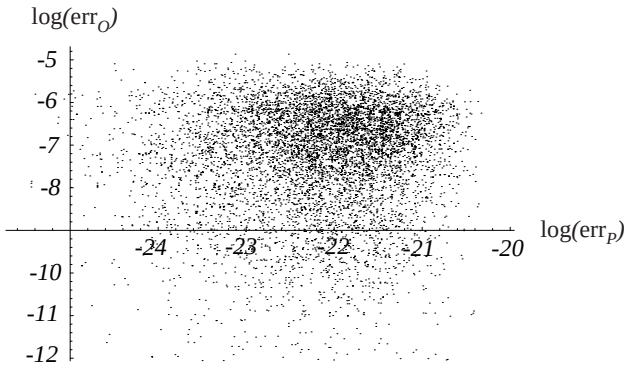

FIGURE 2. A scatter plot of the points $(\mathrm{err}_P, \mathrm{err}_O)$ for the products $((AA)A^{-1})((A^{-1}A^{-1})A)$.

In our example, ($\rho_1 = r_1(\pi/3)$ and $\rho_2 = r_2(4\pi/3)$), $(\log(\mathrm{err}_P(8)), \log(\mathrm{err}_O(8))) \approx (-22.3392, -6.74756)$. This is in the dense central portion of the scatter plot of Figure 2. The scatter plot is clustered tightly enough that we continue to restrict our attention to that one example, i.e. $A_P(d) = r_1(\pi/3)\tau(d)r_2(4\pi/3)$ and $A_O(d) = f(A_P) = R_1(\pi/3)T(d)R_2(4\pi/3)$. This time we consider the effect on $R_P(d)$ and $R_O(d)$ of changing d. For $d \in \{1, 2, \ldots, 50\}$, we compute the points $(d, \log(\mathrm{err}_P(d)))$ and find that the best linear fit for them is $g_1(d) = -41.1529 + 2.15727d$. We then compute the points $(d, \log(\mathrm{err}_O(d)))$ and find that the best linear fit for them is $g_2(d) = -54.2391 + 5.37681d$. For each n, $\log(\mathrm{err}_O(d)) > \log(\mathrm{err}_P(d))$. The two collections of data points and the graphs of g_1 g_2 are shown in Figure 3. Note that $\log(\mathrm{err}_O(d))$ is growing significantly faster than $\log(\mathrm{err}_P(d))$. Furthermore, the slopes are much greater than 1 for g_1 and 2 for g_2. This is because each example is a sixfold product of floating point matrices, and so there is much greater error accumulation than would occur for a single product.

In order to investigate the effect of increasing the numerical precision, in the next example we no longer work with Mathematica's floating point arithmetic. We first compute $A_P(8)$, $A_O(8)$, $B_P(8)$, and $B_O(8)$ symbolically, and then use the **SetPrecision** command in Mathematica to set their precision at 100 decimal digits. Then, given an integer n with $8 \leq n \leq 60$, we multiply each of their elements by a uniformly distributed pseudorandom number, with 100 decimals of precision, in the in-

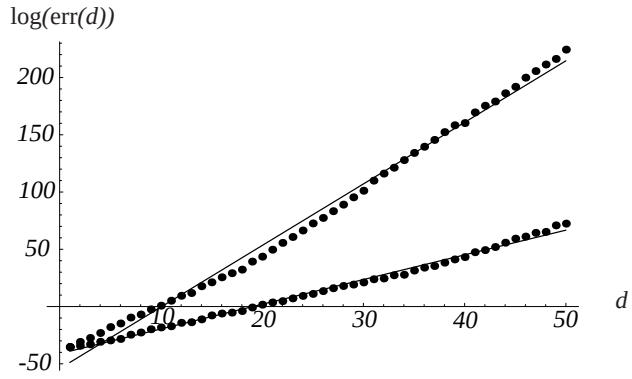

FIGURE 3. The best linear fits for $\log(\mathrm{err}_P(d))$ and $\log(\mathrm{err}_O(d))$.

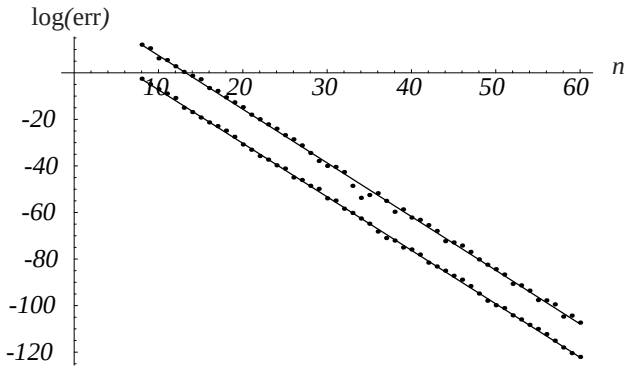

FIGURE 4. The effects of increasing the numerical precision on the logarithm of the error.

terval $[1 - \frac{1}{2}10^{-n+1}, 1 + \frac{1}{2}10^{-n+1}]$. We then compute $\log(\mathrm{err}_P(8))$ and $\log(\mathrm{err}_O(8))$, plot each one as a function of n, and find the best linear fit for each plot. The best linear fits are $h_1(n) = 15.6801 - 2.29848n$ for $\log(\mathrm{err}_P(8))$ and $h_2(n) = 30.5171 - 2.30692n$ for $\log(\mathrm{err}_O(8))$. Figure 4 shows the plots and the graphs of the linear fits for both examples. Since the slopes of the two lines are close to each other, for $n \in [8, 60]$ we expect the ratio $\mathrm{err}_O(8)/\mathrm{err}_P(8)$ to have roughly the same order as $e^{30.5171-15.6801} \approx 2.77733 \times 10^6$. In fact, for each integer n with $8 \leq n \leq 60$ the ratio $\mathrm{err}_O(8)/\mathrm{err}_P(8)$ is in the interval $[1.1 \times 10^5, 1.5 \times 10^7]$. So while increasing the numerical precision predictably improves the accuracy, it does not change the relative degree to which the computations are more accurate in $\mathrm{PSL}(2, \mathbb{C})$ than they are in $\mathrm{SO}^+(3, 1)$.

ACKNOWLEDGEMENTS

This paper arose out of an undergraduate research project of the second author in the Mathematics Department at Virginia Tech. The first two authors gratefully acknowledge support of NSF grant DMS-9971783 and of an REU Supplement to that grant. The third author thanks the MacArthur Foundation for its support. We thank Christopher Beattie and Silvio Levy for helpful comments and suggestions.

REFERENCES

[Beardon 83] A. F. Beardon, *The Geometry of Discrete Groups.* Springer-Verlag, New York Heidelberg Berlin, 1983.

[Shafarevich 90] I. R. Shafarevich, *Algebra I. Encyclopaedia of Mathematical Sciences, v. 11.*, Springer-Verlag, Berlin Heidelberg New York, 1990.

[Thurston 97] W. P. Thurston, *Three-Dimensional Geometry and Topology, Vol. 1*, Princeton University Press, Princeton, 1997.

[Weeks 01a] J. R. Weeks, *SnapPea: A computer program for creating and studying hyperbolic 3-manifolds*, available from http://www.northnet.org/weeks/, 2001.

[Weeks 01b] J. R. Weeks, "Computer graphics in curved spaces," preprint, 2001.

William Floyd, Department of Mathematics, Virginia Tech, Blacksburg, VA 24061 (floyd@math.vt.edu)

Brian Weber, 3700 Richmond Lane NW, Apt. B, Blacksburg, VA 24060 (brweber@vt.edu)

Jeffrey Weeks, 15 Farmer Street, Canton, NY 13617 (weeks@northnet.org)

Received October 12, 2001; accepted in revised form January 7, 2002.