5-2005

A Family of Matrix Representations of the Figure Eight Knot Group

Marvalisa M. Payne
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a thesis written by Marvalisa M. Payne entitled "A Family of Matrix Representations of the Figure Eight Knot Group." I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Morwen Thistlethwaite, Major Professor

We have read this thesis and recommend its acceptance:

David Anderson, Conrad Plaut

Accepted for the Council:
Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a thesis written by Marvalisa M. Payne entitled "A Family of Matrix Representations of the Figure Eight Knot Group". I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Mathematics.

Morwen Thistlethwaite

Major Professor

We have read this thesis
and recommend its acceptance:

David Anderson

Conrad Plaut

Accepted for the Council:

Anne Mayhew

Vice Chancellor and
Dean of Graduate Studies

(Original signatures are on file with official student records.)

A Family of Matrix Representations of the Figure Eight Knot Group

A Thesis Presented for the Master of Science Degree

The University of Tennessee, Knoxville

Marvalisa M. Payne

May 2005

Dedication

I dedicate this thesis to the memory of my mother the late Evelyn Bernice Tollette Mealing. She died three days after my 19th birthday. My memory of my mother has been an indelible influence in my life and continues to guide me though my journey in life.

Acknowledgments

I thank Dr. Thistlethwaite, my major advisor, for his patience and commitment to the completion of this thesis. Dr. Thistlethwaite was quintessential in helping me develop a deeper understanding of mathematics, in particular how to see interconnections between many different realms of mathematics. I would also like to thank Dr. Anderson and Dr. Plaut for serving on my committee. Their willingness to engage me rigorously in my pursuit to master the concepts of abstract mathematics has been an integral part of my success. I acknowledge my undergraduate professors at Augusta State University for introducing me into the realm of true mathematics.

I am especially thankful for my aunt Mary Sue Tollette for her divine influence and for always believing in my abilities although at times I may have faltered.

I am very grateful for the love and support of my husband Phillip and our three daughters. Their unconditional support, encouragement, and understanding has sustained me throughout this journey.

Abstract

An exact computation is given of a new 1-parameter family of $SL(4,\mathbb{R})$ representations of the figure eight knot group.

Table of Contents

Chapter One

Background and Statement of Main Result

Fundamental Group

Let X be a topological space and let x_0 be a point of X. A path in X that begins and ends at x_0 is called a *loop based at* x_0. The set of path homotopy classes of loops based at x_0, with the operation $*$ is called the *fundamental group of* X *relative to the base point* x_0 [6]. It is denoted by $\pi_1(X, x_0)$. The elements of are path homotopy equivalence classes of loops based at x_0. The group operation is

$$[f]\ [g] = [f\ *\ g]$$

where

$$(f * g)\, t = \begin{cases} f(2t) & 0 \le t \le \frac{1}{2} \\ g(2t-1) & \frac{1}{2} \le t \le 1 \end{cases}$$

Given two loops f and g based at x_0, the product $f * g$ is always defined and is a loop based at x_0.

Covering Map

Let $p : E \to B$ be continuous and surjective. If every point b of B has a neighborhood U that is *evenly covered* by p, *i.e.* each component of $p^{-1}(U)$ is mapped homeomorphically by p onto U, then p is called a *covering map*, and E is said to be a *covering space of* B [6].

For example, the map $f : \mathbb{R} \to S^1$ given by $f(t) = e^{2\pi i t}$ is a covering map, wrapping the real line around the circle. The preimage of a small open arc of the circle is a collection of open intervals of the real line, offset by multiples of 2π.

Another important example is the covering of the torus by the Euclidean plane, given by the map $g : \mathbb{R}^2 \to S^1 \times S^1,\ g(t, u) = (e^{2\pi i t}\,,\, e^{2\pi i u})$. Here each unit square $[m\,,\, m+1] \times [n\,,\, n+1]\ \ (m, n \in \mathbb{Z})$ in the plane is wrapped around the torus.

Covering Transformation

Let $p : E \to B$ be a covering map. A *covering transformation of* p is a homeomorphism $\tau : E \to E$ such that $p \circ \tau = p$. The set of covering transformations forms a group under composition. In the case where the covering space E is simply connected, the group of covering transformations is isomorphic to the fundamental group of the base space B [6].

For the above example $f : \mathbb{R} \to S^1$, $f(t) = e^{2\pi it}$, the group of covering transformations is isomorphic to the additive group of integers $\mathbb{Z}$, and consists of translations $\tau_n : x \mapsto x + n$ $(n \in \mathbb{Z})$.

The group of covering transformations of $g : \mathbb{R}^2 \to S^1 \times S^1$, $g(t, u) = (e^{2\pi it}, e^{2\pi iu})$ is isomorphic to $\mathbb{Z} \times \mathbb{Z}$, and consists of translations $\tau_{m,n} : (x, y) \mapsto (x + m, y + n)$ $(m, n \in \mathbb{Z})$. The map $\tau_{m,n}$ shifts points of the plane m units in the x-direction n units in the y-direction.

In both these examples, since the covering transformations are Euclidean isometries, the base space inherits a metric from the standard Euclidean metric on the covering space [9].

Hyperbolic Geometry

Hyperbolic geometry is an example of a "non-Euclidean" geometry. In hyperbolic geometry, lines, distance and motion all behave differently than their Euclidean counterparts. To illustrate these abstract concepts we have different models of hyperbolic space just as we use many different kinds of map projections of the earth. For example, we use a map projection to represent the curved surface of the earth in a flat plane. Likewise, we use the Poincaré, upper half space, Klein-Beltrami and Minkowski models to illustrate hyperbolic space. For the purposes of this paper, I will discuss the upper half space model of 3-dimensional hyperbolic space $\mathbb{H}^3$. This model, attributed to Henri Poincaré, may be generalized to represent hyperbolic space in any number of dimensions.

Upper Half Space Model of $\mathbb{H}^3$

In this model, $\mathbb{H}^3$ is the set of all points $(x, y, t) \in \mathbb{R}^3$ for which $t > 0$. The plane $t = 0$, compactified by adding a point at infinity, is called the *boundary* of $\mathbb{H}^3$ and is homeomorphic to the 2-sphere S^2. We regard the

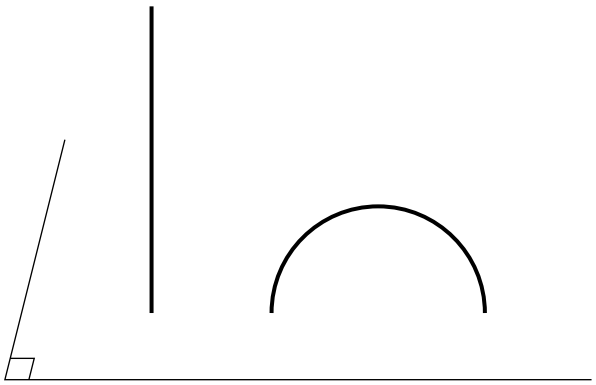

Figure 1: Geodesics in $\mathbb{H}^3$

boundary of $\mathbb{H}^3$ as being the extended complex plane. The metric ds of $\mathbb{H}^3$ is given by the formula

$$ds^2 = \frac{dx^2 + dy^2 + dt^2}{t} \ .$$

Informally, near a point (x, y, t) the hyperbolic metric is the standard Euclidean metric scaled by the factor $1/t$.

The geodesics in this model are either Euclidean half circles with centers located on the plane $z = 0$ or vertical lines (see Figure 1). Thus the shortest route between points P and Q in hyperbolic space is along the unique geodesic containing these points.

Möbius Transformations and Isometries of $\mathbb{H}^3$

A *Möbius transformation* of the extended complex plane $\mathbb{C} \cup \{\infty\}$ is a transformation of form

$$T(z) = \frac{az + b}{cz + d} \qquad (a, b, c, d \in \mathbb{C} \ , \ ad - bc \neq 0) \ .$$

We note that since a fraction in unchanged if we multiply numerator and denominator by the same non-zero number k, $T(z)$ is unaltered if we replace a, b, c, d by ka, kb, kc, kd respectively. In particular, we may assume without loss of generality that $ad - bc = 1$.

The Möbius transformation $T(z)$ is associated in a natural way to its matrix of coefficients

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} ,$$

where the entries a, b, c, d are only determined up to multiplication by a non-zero constant. Composition of Möbius transformations then corresponds to multiplication of the associated matrices. The set of all Möbius transformations forms a group $\mathcal{M}$ under composition. It follows from this discussion that the group $\mathcal{M}$ is isomorphic to the quotient group $SL(2,\mathbb{C})/H$, where $SL(2,\mathbb{C})$ is the group of all 2×2 matrices over the field of complex numbers with determinant 1, and where

$$H = \left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} , \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix} \right\} .$$

The quotient group $SL(2,\mathbb{C})/H$ is called the *projective special linear group over* $\mathbb{C}$, and is denoted $PSL(2,\mathbb{C})$. We may regard it as the group of equivalence classes of 2×2 matrices over $\mathbb{C}$ with determinant 1, where each matrix is declared to be equivalent to its negative.

By definition each Möbius transformation acts on the boundary of $\mathbb{H}^3$, but the action may be extended in a natural way to $\mathbb{H}^3$ itself, giving a direct isometry of $\mathbb{H}^3$ [5]. All direct isometries of $\mathbb{H}^3$ arise in this manner. For example, the transformation $z \mapsto z + 1$, represented by the matrix

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} ,$$

induces the isometry

$$(x, y, t) \mapsto (x + 1, y, t)$$

of $\mathbb{H}^3$.

The Figure Eight Knot and the Riley Homomorphism

The *figure eight knot* is the knot in S^3 shown in Figure 2.

Let us denote the figure eight knot by K. We shall now describe the fundamental group of its complement $S^3 - K$. If we take the basepoint x_0 of $S^3 - K$ to be a point in front of the diagram, then $\pi_1(S^3 - K)$ is generated by

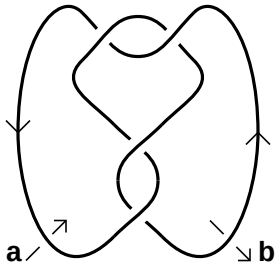

Figure 2: the Figure Eight Knot

the equivalence classes of loops a , b which travel from x_0 to the knot, then loop underneath the knot as indicated in the diagram, and finally travel back to x_0 [8]. Let us denote the inverses of a , b by A , B respectively. Then the generators a , b are subject to the single relation $ABabaBAbab = 1$. Therefore we have the following presentation for $\pi_1(S^3 - K)$:

$$\pi_1(S^3 - K) = \langle a\,,\, b \mid ABabaBAbab = 1 \rangle \ .$$

In the 1970's R. Riley [7] discovered an injective homomorphism of $\pi_1(S^3 - K)$ into $PSL(2, \mathbb{C})$, which we shall denote ϕ_0:

$$\phi_0(a) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \quad , \quad \phi_0(b) = \begin{bmatrix} 1 & 0 \\ w & 1 \end{bmatrix} \ ,$$

where $w = \left(\frac{1}{2}\right)(-1 + \sqrt{3}i)$.

This is sufficient to determine the homomorphism ϕ_0, since the group $\pi_1(S^3 - K)$ is generated by the elements a , b. In the above formula we are abusing notation slightly by writing matrices instead of equivalence classes of matrices.

The image of the Riley homomorphism is a group G of direct isometries of $\mathbb{H}^3$. It corresponds to a covering map $p : \mathbb{H}^3 \to S^3 - K$, for which G is the group of covering transformations. It also gives a complete hyperbolic metric on $S^3 - K$. The situation is very similar to that for the covering map $\mathbb{R}^2 \to S^1 \times S^1$ described earlier, though is harder to visualize. Also, just as the Euclidean plane $\mathbb{R}^2$ was tiled by squares each of which wrapped

around the torus under the covering map, hyperbolic space $\mathbb{H}^3$ is tiled by polyhedra each of which "wraps" around the figure eight knot complement. In the language of group actions, $S^3 - K$ is the orbit space of $\mathbb{H}^3$ under the action of the group G.

Deformations of Homomorphisms and 4×4 Matrices

Let ϕ_1 , ϕ_2 be homomorphisms of a group G to a group H. We shall say that ϕ_1 , ϕ_2 are *equivalent* if there exists an element $h \in H$ such that $h^{-1}\, \phi_1(g)\, h = \phi_2(g)$ for each $g \in G$.

W. Thurston [9] proved that there exists a 2-parameter family of *deformations* of the Riley homomorphism ϕ_0, i.e. a family of homomorphisms $\phi_{u,v} : \pi_1(S^3 - K) \to PSL(2, \mathbb{C})$:

$$\phi_{u,v}(a) = \begin{bmatrix} f_{11}(u,v) & f_{12}(u,v) \\ f_{21}(u,v) & f_{22}(u,v) \end{bmatrix} \quad , \quad \phi_{u,v}(b) = \begin{bmatrix} g_{11}(u,v) & g_{12}(u,v) \\ g_{21}(u,v) & g_{22}(u,v) \end{bmatrix} \ ,$$

where each function f_{ij} , g_{ij} is continuous on some neighborhood of 0, and where $\phi_{0,0}$ is the Riley homomorphism. Furthermore, for $(u, v) \neq (0, 0)$, the homomorphism $\phi_{u,v}$ is not equivalent to ϕ_0, *i.e.* the matrices $\phi_{u,v}(a)$, $\phi_{u,v}(b)$ cannot be obtained by conjugating $\phi_0(a)$, $\phi_0(b)$ by some matrix.

We cannot find any new essential deformations of $\phi_0 : \pi_1(S^3 - K) \to PSL(2, \mathbb{C})$; however, we might hope to be find new essential deformations of ϕ_0 composed with an embedding of $PSL(2, \mathbb{C})$ in some larger group.

It happens that $PSL(2, \mathbb{C})$ is isomorphic to $SO^+(3, 1)$, a subgroup of index 4 of the *Lorentz group* $O(3, 1)$. The group $O(3, 1)$ consists of 4×4 matrices satisfying the following condition:

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}^{-1} = \begin{bmatrix} a_{11} & -a_{21} & -a_{31} & -a_{41} \\ -a_{12} & a_{22} & a_{32} & a_{42} \\ -a_{13} & a_{23} & a_{33} & a_{43} \\ -a_{14} & a_{24} & a_{34} & a_{44} \end{bmatrix} .$$

$SO^+(3, 1)$ is the subgroup consisting of matrices that (i) have determinant 1, and (ii) preserve the *light cone* $\{(x_1, x_2, x_3, x_4)\, ;\, x_1^2 = x_2^2 + x_3^2 + x_4^2$ and $x_1 \geq 0\}$. $SO^+(3, 1)$ is the natural way of viewing the direct isometries of $\mathbb{H}^3$ as presented in the Minkowski model.

Since the matrices in $SO^+(3,1)$ have determinant 1, $SO^+(3,1)$ is a subgroup of the group $SL(4,\mathbb{R})$ consisting of all 4×4 matrices over $\mathbb{R}$ with determinant 1. Let ψ_0 be the composite

$$\pi_1(S^3 - K) \underset{\phi_0}{\longrightarrow} PSL(2,\mathbb{C}) \underset{\rho}{\longrightarrow} SO^+(3,1) \underset{i}{\hookrightarrow} SL(4,\mathbb{R}) \ ,$$

where ρ is the isomorphism mentioned above and i is inclusion.

We are now ready to state the main result of this paper.

Theorem. There exists a 1-parameter family ψ_u of essential deformations into $SL(4,\mathbb{R})$ of the homomorphism ψ_0, that do not arise from Thurston's deformations. More precisely, these deformations are not conjugate to deformations of form $i \circ \rho \circ \phi_{u,v}$.

Chapter Two

Construction of the Deformations of Ψ_0

Summary of the Construction

The construction proceeds in several stages. The complete programs are presented, with documentation, as appendices to this paper.

1. Convert the matrices $\phi_0(a)$, $\phi_0(b)$ to 4×4 matrices in $SO^+(3,1)$.

2. Use Newton's method to find a small number of suitable nearby homomorphisms.

3. For each homomorphism found in Step 2, apply the LLL algorithm [3] to guess exact expressions for the matrix entries.

4. Use polynomial interpolations to obtain a unifying formula for the separate homomorphisms, thus obtaining 4×4 matrices $\psi_v(a)$, $\phi_v(b)$ with entries that are radical expressions in a parameter v.

5. Verify, using the formal algebra capabilities of Maple$^{\text{TM}}$ [4] , that the matrices $\psi_v(a)$, $\phi_v(b)$ satisfy the relation in the presentation of $\pi_1(S^3 - K)$.

Stage 1: Converting from $PSL(2,\mathbb{C})$ to $SO^+(3,1)$

This step of the procedure is well-known (see for example [10]), so we only describe it briefly.

Points in 4-dimensional space can be thought of as 2×2 Hermitian matrices. Let us denote this vector space V. The following four Hermitian matrices form a basis for V:

$$e_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \ , \ e_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \ , \ e_3 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \ , \ e_4 = \begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix} \ .$$

Let $u \in SL(2,\mathbb{C})$ represent $[u] \in PSL(2,\mathbb{C})$, and let $v \in V$. The assignment $v \mapsto u^T v u$ defines an endomorphism of V, and the required matrix $\rho([u])$ is simply the matrix of this endomorphism with respect to the basis

$\{e_1\,,\,e_2\,,\,e_3\,,\,e_4\}$. Since $u^T v u = (-u)^T v(-u)\,,$ ρ is well-defined on equivalence classes. Checking that ρ really is an isomorphism is routine.

Stage 2: Using Newton's Method to Find Probable Nearby Homomorphisms

This stage involved some improvisation, but was justified by the fact that in the end it produced the desired result.

Thurston's deformations all have the property that the traces of the matrices $(i \circ \rho \circ \phi_{u,v})(a)\,,\,(i \circ \rho \circ \phi_{u,v})(b)$ are distinct from 4, for all pairs (u, v) distinct from $(0, 0)$. Therefore if we can find homomorphisms ψ' near ψ_0 for which the traces of $\psi'(a)\,,\,\psi'(b)$ are equal to 4 to within a large number of decimal places, we shall have evidence that our desired deformations do exist.

Throughout the rest of this paper, we express the fact that two matrices are inverses of one another by denoting one in lower case and the other using the corresponding upper case letter, *e.g.* $A_0 = a_0^{-1}$. Let $a_0 = \psi_0(a)\,,\, b_0 = \psi_0(b)$. Then, since ψ_0 is a homomorphism, $a_0\,,\, b_0$ satisfy the relation $A_0 \cdot B_0 \cdot a_0 \cdot b_0 \cdot a_0 \cdot B_0 \cdot A_0 \cdot b_0 \cdot a_0 \cdot b_0 = 1$. To simplify computation, we take advantage of the fact that the set of 4×4 matrices has an algebra structure to rewrite this relation as

$$A_0 \cdot B_0 \cdot a_0 \cdot b_0 \cdot a_0 \;=\; (B_0 \cdot A_0 \cdot b_0 \cdot a_0 \cdot b_0)^{-1} \;\;,$$

or

$$A_0 \cdot B_0 \cdot a_0 \cdot b_0 \cdot a_0 \;-\; B_0 \cdot A_0 \cdot B_0 \cdot a_0 \cdot b_0 \;=\; 0 \;\;.$$

The plan is to perturb the matrices $a_0\,,\, b_0$ slightly, and then use these perturbed matrices as a starting point for Newton's method, hoping to converge to different matrices $a_1\,,\, b_1$ satisfying the same relation, *i.e.*

$$A_1 \cdot B_1 \cdot a_1 \cdot b_1 \cdot a_1 \;-\; B_1 \cdot A_1 \cdot B_1 \cdot a_1 \cdot b_1 \;=\; 0 \;\;.$$

Since each matrix has 16 entries, the system of equations has 32 unknowns. Our single matrix equation gives us 16 equations in these unknowns, so our system is underdetermined, and at each iteration of the Newton process we expect there to be infinitely many solutions. However, using the QR decomposition [1] of the Jacobian matrix we can find a solution of minimal

norm, and will use this solution to adjust the matrices to the starting point of the next iteration.

Since we wish to find solutions of a certain kind, we add six equations to the list, making 22 in all. These extra equations provide the following constraints:

1. Constrain the determinant of a_1 to 1.

2. Constrain the determinant of b_1 to 1.

3. Constrain the trace of a_1 to 4.

4. Constrain the trace of b_1 to 4.

5. Constrain the trace of $A_1 \cdot B_1$ to 3.

6. Constrain the trace of $a_1 \cdot b_1 \cdot b_1$ to some rational number k close to 4. In practice we chose $k = 4 + 1/(99 + n)$ for $n = 1, 2, \ldots, 11$.

The purpose of the third and fourth constraints is to ensure that we avoid Thurston's deformations. Since the trace of $a_0 \cdot b_0 \cdot b_0$ is equal to 4, the sixth constraint ensures that we obtain a homomorphism not equivalent to ψ_0. The fifth constraint is redundant, but improved the performance of Newton's method for some reason.

Separate applications of Newton's method for the 11 values of k stated above yielded 11 solutions to the equation

$$A_1 \cdot B_1 \cdot a_1 \cdot b_1 \cdot a_1 \; - \; B_1 \cdot A_1 \cdot B_1 \cdot a_1 \cdot b_1 \; = \; 0 \;\; ,$$

accurate to 500 decimal places. In order to extract algebraic information out the matrix entries it was necessary to conjugate them to an improvised "normal form". This was done in three stages:

1. Conjugate so that one of the generators is diagonal. Unfortunately, neither a_1 nor b_1 turned out to be diagonalizable, so the generating set was changed to $\{c_1 = a_1 \cdot b_1 \cdot b_1 \;\; , \;\; d_1 = a_1 \cdot b_1\}$, and the new generators c_1, d_1 were simultaneously conjugated to matrices c_2, d_2 where d_2 was diagonal. It was observed that d_1 had four distinct eigenvalues, of form λ_1, $1/\lambda_1$, λ_2, $1/\lambda_2$, where λ_1 is real and λ_2 lies on the unit circle of the complex plane. The matrix d_2 was chosen as

$$\begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & 1/\lambda_1 & 0 & 0 \\ 0 & 0 & \lambda_2 & 0 \\ 0 & 0 & 0 & 1/\lambda_2 \end{bmatrix} \;\; ,$$

where $|\lambda_1| > 1$ and λ_2 lies on the upper unit semicircle.

2. In order to "standardize" the matrix c_2, while keeping d_2 diagonal, c_2 was further conjugated by a diagonal matrix so as to obtain entries equal to 1 immediately above the diagonal. Specifically, let

$$k_1 = 1/(c_2)_{1,2} \ , \ \ k_2 = 1/((c_2)_{1,2}(c_2)_{2,3}) \ , \ \ k_3 = 1/((c_2)_{1,2}(c_2)_{2,3}(c_2)_{3,4}) \ ,$$

and define

$$c_2' \ = \ m^{-1} \cdot c_2 \cdot m \ , \quad \text{where} \quad m_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & k_1 & 0 & 0 \\ 0 & 0 & k_2 & 0 \\ 0 & 0 & 0 & k_3 \end{bmatrix} \ .$$

Then c_2' has form

$$\begin{bmatrix} * & 1 & * & * \\ * & * & 1 & * \\ * & * & * & 1 \\ * & * & * & * \end{bmatrix} \ ,$$

whereas the diagonal matrix d_2 commutes with the conjugating matrix m_1.

3. We could work with the matrices c_2', d_2 constructed in the previous step, but they have a drawback in that their entries are not real. A final conjugation was performed to correct this, the specific conjugating matrix being the matrix m_2 with columns

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \ , \quad \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \ , \quad c_2' \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \ , \quad c_2' \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \ .$$

The end result of this series of three conjugations was a pair of matrices c_3, d_3, with real entries and of form

$$c_3 = \begin{bmatrix} 0 & 0 & * & * \\ 0 & 0 & * & * \\ 1 & 0 & * & * \\ 0 & 1 & * & * \end{bmatrix} \ , \quad d_3 = \begin{bmatrix} \lambda_1 & 0 & * & * \\ 0 & 1/\lambda_1 & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{bmatrix} \ .$$

The first two columns of d_3 are determined by the fact that the first two elements of our new basis are $(1\,,\,0\,,\,0\,,\,0)$, $(0\,,\,1\,,\,0\,,\,0)$, and the first two

columns of c_3 are determined by the fact that the last two elements of our new basis are the images of the first two elements under c_2'.

Stage 3: Obtaining Radical Expressions for the Matrix Entries

The next step in the computation is to guess a field F containing all entries of the matrices c_3, d_3 produced in Stage 2. We hope that F is of small degree over the field of rationals $\mathbb{Q}$. We emphasize that nothing is certain at this stage as everything is built upon numerical data.

Since the eigenvalue λ_1 of d_3 is an entry of d_3, the field F must contain λ_1. Recall that in Stage 2 we used Newton's method to find numerical solutions for $k = 1, 2, \ldots, 11$, where

$$\mathrm{trace}(a_1 b_1 b_1) = \mathrm{trace}(c_1) = \mathrm{trace}(c_3) = 4 + 1/(99 + k) \ .$$

According to Maple, for each of the 11 values of k the characteristic polynomial of d_3 is

$$1 - 3x + ux^2 - 3x^3 + x^4 \ ,$$

where

$$u = 1 - \frac{1}{(99 + k)(100 + k)} \ .$$

Also, by formal algebra Maple manages to express the real eigenvalue λ_1 in terms of u as

$$\lambda_1 = \frac{1}{4}\left(3 + \sqrt{17 - 4u} + \sqrt{10 - 4u + 6\sqrt{17 - 4u}}\right) \ .$$

This suggests strongly that we take u to be the parameter for our conjectured family of homomorphisms, and that we take as candidate field F the field of degree 4 over $\mathbb{Q}(u)$ generated by λ_1. Let us define

$$\alpha = \sqrt{17 - 4u} \quad , \quad \beta = \sqrt{10 - 4u + 6\alpha} \ .$$

Then we may take $\{1, \alpha, \beta, \alpha\beta\}$ as basis for F as a vector space over the field $\mathbb{Q}(u)$.

We wish to express each matrix entry as a linear combination of these basis elements, for each of the 11 rational values of u taken in Stage 2.

Maple does not have this capability built in, so Dr. Thistlethwaite took the numerical data output by Stage 2 and computed the coefficients of the linear combinations using the *lindep()* function of the number theory package Pari [2]. The *lindep()* function is based on the *LLL algorithm* [3] for guessing a polynomial with integer coefficients, of which a given floating-point number is an approximate root. For example, if one gives LLL the number $x = 1.41421356237309504880$, which is a close approximation to $\sqrt{2}$, and asks it to find a polynomial of degree at most 2 for which x is an approximate root, it will return the polynomial $x^2 - 2$.

In the present context, for each matrix entry μ and each of the 11 values of u for which we have data, the Pari function *lindep()* returned integers t_1 , t_2 , t_3 , t_4 , t_5 such that

$$t_1 \cdot 1 \ + \ t_2 \cdot \alpha \ + \ t_3 \cdot \beta \ + \ t_4 \cdot \alpha\beta \ + \ t_5 \cdot \mu \ = \ 0 \ ,$$

whence

$$\mu = -\frac{1}{t_5}(t_1 \cdot 1 \ + \ t_2 \cdot \alpha \ + \ t_3 \cdot \beta \ + \ t_4 \cdot \alpha\beta) \ .$$

Stage 4: Interpolating the Integer Coefficients

For each matrix entry μ and each of the 11 values of the parameter u , we now have integer coefficients t_i $(i = 1, 2, 3, 4, 5)$ of a linear dependence between the numbers 1 , α , β , $\alpha\beta$, μ . We would like to unify this data so that each coefficient t_i is a polynomial in the parameter u . The Maple function *PolynomialInterpolation()* performed this task very efficiently.

We now have each matrix entry expressed as a linear combination of the basis elements 1 , α , β , $\alpha\beta$, where each coefficient $-t_i/t_5$ $(1 \le i \le 4)$ is a quotient of two polynomials over u. These expressions are fairly complicated, and are given in Appendix 3. However, we cannot yet be certain that these expressions are correct, as they were obtained from numerical data by means of a sequence of guesses.

Stage 5: Exact Verification of the Group Relation

Let c_4 , d_4 denote the matrices obtained in Stage 4. These matrices have entries that are exact expressions in the parameter u . We expect that they define a homomorphism ψ_u of the figure eight knot group into $SL(4, \mathbb{R})$ for

each value of u, but in order to prove this we need to check that the defining relation of the figure eight knot group has been preserved.

Recall that in Stage 2 we defined new generators

$$c_1 \;=\; a_1 \cdot b_1 \cdot b_1 \quad , \quad d_1 \;=\; a_1 \cdot b_1 \;\; .$$

It follows by simple algebra that

$$a_1 \;=\; d_1 \cdot C_1 \cdot d_1 \quad , \quad b_1 \;=\; D_1 \cdot c_1 \;\; ,$$

and that the group relation

$$A_1 \cdot B_1 \cdot a_1 \cdot b_1 \cdot a_1 \cdot B_1 \cdot A_1 \cdot b_1 \cdot a_1 \cdot b_1 \;=\; 1$$

is equivalent to the relation

$$c_1 \cdot D_1 \cdot C_1 \cdot d_1 \cdot d_1 \cdot d_1 \cdot C_1 \cdot D_1 \cdot c_1 \;=\; 1 \;\; ,$$

or

$$c_1 \cdot D_1 \cdot C_1 \cdot d_1 \cdot d_1 \;-\; C_1 \cdot d_1 \cdot c_1 \cdot D_1 \;=\; 0 \;\; .$$

Therefore, in order to check that the assignment

$$a \mapsto d_4 \cdot C_4 \cdot d_4 \;\; , \;\; b \mapsto D_4 \cdot c_4$$

defines a homomorphism ψ_u of $\pi_1(S^3 - K)$ to $SL(4, \mathbb{R})$ *independently of the parameter* u, it is sufficient to check that

$$c_4 \cdot D_4 \cdot C_4 \cdot d_4 \cdot d_4 \;-\; C_4 \cdot d_4 \cdot c_4 \cdot D_4 \;=\; 0 \;\; .$$

Maple was assigned this task, and confirmed that the identity does indeed hold. Furthermore, Maple confirmed that the traces of $d_4 C_4 d_4$, $D_4 c_4$ are exactly equal to 4, proving that the deformations ψ_u of ψ_0 do not arise from Thurston's deformations. This final step concludes the proof of the Theorem of this paper.

We note that the original homomorphism ψ_0 can be recovered by setting $u = 1$.

References

References

[1] Noble, B., Daniel, J.W., *Applied Linear Algebra*, Prentice Hall, 1988.

[2] Cohen, H., *Pari/Gp*, Free Computer Algebra System with emphasis on Number Theory, available for download from http://pari.math.u-bordeaux.fr/ .

[3] Lenstra, A. K., Lenstra, H. W., and Lovasz, L., *Factoring Polynomials with Rational Coefficients*, Math. Ann. **261**, 515-534, 1982.

[4] *Maple 9.5*, Computer Algebra System, http://www.maplesoft.com/ .

[5] Maskit, B., *Kleinian Groups*, Springer, 1987.

[6] Munkres, J.R., *Topology* (Second Edition), Prentice Hall, 2000.

[7] Riley, R., *An Elliptic Path from Parabolic Representations to Hyperbolic Structures*, Topology of Low-Dimensional Manifolds, Proceedings, Sussex 1977 (Ed. R. Fenn).

[8] Rolfsen, D., *Knots and Links*, AMS Chelsea, vol. 346.H, 2003.

[9] Thurston, W., *The Geometry and Topology of Three-Manifolds*, Lecture Notes, Princeton University, 1980.

[10] Weeks, J., *SnapPea*, Program for Creating and Studying Hyperbolic Manifolds, available for download from http://www.geometrygames.org/SnapPea/ .

Appendices

Appendix 1

Maple Script for Stages 1, 2 of the Computation

```
# This program calculates an approximation to a deformed representation
# of the group of the figure-eight into SL(4,R). It takes the SO(3,1)
# representation determined by the complete hyperbolic structure, perturbs
# it by a small random vector, and then uses the Newton method to converge
# to a deformed representation.  We constrain the matrices representing
# the standard parabolic generators so that their traces remain equal to 4.
# Because we wish to apply the LLL algorithm to the output, computations
# must be accurate to at least 500 decimal places.

with(LinearAlgebra):
interface(echo=1, rtablesize=32):

# cx, dx are 4x4 matrices of unknowns
cx := Matrix([[c11,c12,c13,c14],[c21,c22,c23,c24],[c31,c32,c33,c34],[c41,c42,c43,c44]]):
dx := Matrix([[d11,d12,d13,d14],[d21,d22,d23,d24],[d31,d32,d33,d34],[d41,d42,d43,d44]]):

# cy, dy are "flattened" versions of the above

cy := Vector([c11,c12,c13,c14,c21,c22,c23,c24,c31,c32,c33,c34,c41,c42,c43,c44]):
dy := Vector([d11,d12,d13,d14,d21,d22,d23,d24,d31,d32,d33,d34,d41,d42,d43,d44]):

mycharpoly := x -> simplify( Determinant(x - Q*IdentityMatrix(4)) ):

ccharpol := mycharpoly(cx):
dcharpol := mycharpoly(dx):
cdcharpol := mycharpoly(cx.dx):

cpol0 := coeff(ccharpol, Q, 0):
cpol1 := coeff(ccharpol, Q, 1):
dpol0 := coeff(dcharpol, Q, 0):
dpol1 := coeff(dcharpol, Q, 1):
cdpol1 := coeff(cdcharpol, Q, 1):
cddpol3 := -tr4(cx.dx.dx):

# Function to compute the trace of a 4x4 matrix

tr4 := x -> x[1,1] + x[2,2] + x[3,3] + x[4,4]:

# Zero function for initializing matrices

f := (i,j) -> 0:

# Procedure to convert an SL(2,C) matrix to an SO(3,1) matrix

Sl2cToSo31 := proc( Sl2cmatrix )

local u, e1, e2, e3, e4, x1, x2, x3, x4, So31matrix,
x11, x12, x13, x14, x21, x22, x23, x24, x31, x32, x33, x34, x41, x42, x43, x44;

u := Sl2cmatrix:
```

```
e1 := Matrix([[1,0],[0,1]]):
e2 := Matrix([[1,0],[0,-1]]):
e3 := Matrix([[0,1],[1,0]]):
e4 := Matrix([[0,I],[-I,0]]):

x1 := u.e1.HermitianTranspose(u):
x2 := u.e2.HermitianTranspose(u):
x3 := u.e3.HermitianTranspose(u):
x4 := u.e4.HermitianTranspose(u):

x11 := Re((x1[1,1] + x1[2,2])/2):
x21 := Re((x1[1,1] - x1[2,2])/2):
x31 := Re(x1[1,2]):
x41 := Im(x1[1,2]):

x12 := Re((x2[1,1] + x2[2,2])/2):
x22 := Re((x2[1,1] - x2[2,2])/2):
x32 := Re(x2[1,2]):
x42 := Im(x2[1,2]):

x13 := Re((x3[1,1] + x3[2,2])/2):
x23 := Re((x3[1,1] - x3[2,2])/2):
x33 := Re(x3[1,2]):
x43 := Im(x3[1,2]):

x14 := Re((x4[1,1] + x4[2,2])/2):
x24 := Re((x4[1,1] - x4[2,2])/2):
x34 := Re(x4[1,2]):
x44 := Im(x4[1,2]):

So31matrix :=
Matrix([[x11,x12,x13,x14],[x21,x22,x23,x24],[x31,x32,x33,x34],[x41,x42,x43,x44]]):

return So31matrix:

end proc:

# This procedure performs a Newton iteration.  The linear system is
# underdetermined, so we use the QR decomposition of the Jacobian to
# find an optimal solution.  The procedure accepts approximations to
# the two generators, and returns better approximations.

NewtonIteration := proc( c1t, d1t )

local c1, d1, C1, D1, cdsubs, resid, ldrel, rdrel, drel,
jac, row, i, j, k, c1charpol, d1charpol, c1d1charpol, c1d1d1charpol,
fac, rh, Q1, R1, trsl, dc1, dd1, dc, dd, nc, nd:

# Move c1t, d1t to local variables c1, d1 and compute their inverses

c1 := c1t:
d1 := d1t:
C1 := MatrixInverse(c1):
D1 := MatrixInverse(d1):
```

```
# cdsubs is the list of substitution rules for evaluating at c1, d1

cdsubs := c11=c1[1,1], c12=c1[1,2], c13=c1[1,3], c14=c1[1,4],
c21=c1[2,1], c22=c1[2,2], c23=c1[2,3], c24=c1[2,4],
c31=c1[3,1], c32=c1[3,2], c33=c1[3,3], c34=c1[3,4],
c41=c1[4,1], c42=c1[4,2], c43=c1[4,3], c44=c1[4,4],
d11=d1[1,1], d12=d1[1,2], d13=d1[1,3], d14=d1[1,4],
d21=d1[2,1], d22=d1[2,2], d23=d1[2,3], d24=d1[2,4],
d31=d1[3,1], d32=d1[3,2], d33=d1[3,3], d34=d1[3,4],
d41=d1[4,1], d42=d1[4,2], d43=d1[4,3], d44=d1[4,4]:

# The 4x4 matrix resid is the residual on evaluating the relator at c1, d1;
# this is what the procedure is trying to cancel out.

resid := c1.d1.c1.D1.C1 - d1.c1.D1.C1.D1:

# The 4x4 matrix drel is the differential of the relator at c1, d1;
# we simplify as we build it up term by term.

ldrel := Matrix(4,4,f):
rdrel := Matrix(4,4,f):

ldrel := ldrel + cx.d1.c1.D1.C1:
ldrel := ldrel + c1.dx.c1.D1.C1:
ldrel := ldrel + c1.d1.cx.D1.C1:
ldrel := ldrel + c1.d1.c1.(-D1.dx.D1).C1:
ldrel := simplify(ldrel + c1.d1.c1.D1.(-C1.cx.C1)):

rdrel := rdrel + dx.c1.D1.C1.D1:
rdrel := rdrel + d1.cx.D1.C1.D1:
rdrel := rdrel + d1.c1.(-D1.dx.D1).C1.D1:
rdrel := rdrel + d1.c1.D1.(-C1.cx.C1).D1:
rdrel := simplify(rdrel + d1.c1.D1.C1.(-D1.dx.D1)):

drel := simplify(ldrel - rdrel):

# The next step is to assemble the 20x32 Jacobian matrix jac;
# the first 16 rows of jac are coefficients of cij, dij in entries of drel.

jac := Matrix(22,32,f):

row := 0;

for i from 1 to 4 do for j from 1 to 4 do
row := row+1:
for k from 1 to 16 do jac[row,k] := coeff(drel[i,j], cy[k]) end do:
for k from 1 to 16 do jac[row,16+k] := coeff(drel[i,j], dy[k]) end do:
end do end do:

# The last four rows of jac are designed to constrain the two determinants to 1
# and the coefficient of degree 1 in each characteristic polynomial to -4.

for i from 1 to 16 do jac[17,i] := subs(cdsubs, diff(cpol0, cy[i])) end do:
for i from 1 to 16 do jac[18,i] := subs(cdsubs, diff(cpol1, cy[i])) end do:
for i from 1 to 16 do jac[19,16+i] := subs(cdsubs, diff(dpol0, dy[i])) end do:
for i from 1 to 16 do jac[20,16+i] := subs(cdsubs, diff(dpol1, dy[i])) end do:
```

```
for i from 1 to 16 do jac[21,i] := subs(cdsubs, diff(cdpol1, cy[i])) end do:
for i from 1 to 16 do jac[21,16+i] := subs(cdsubs, diff(cdpol1, dy[i])) end do:
for i from 1 to 16 do jac[22,i] := subs(cdsubs, diff(cddpol3, cy[i])) end do:
for i from 1 to 16 do jac[22,16+i] := subs(cdsubs, diff(cddpol3, dy[i])) end do:

# Next we assemble the 20x1 right-hand-side matrix, rhs.
# The first 16 entries of rhs are the negatives of the entries of resid,
# and the last four entries are the negatives of the residuals for
# the appropriate coefficients of characteristic polynomials.
# Note that the degree 1 coefficients are the negatives of the traces of C1, D1.

rh := Matrix(22,1,f):

for i from 1 to 4 do for j from 1 to 4 do rh[4*(i-1)+j, 1] := -resid[i,j] end do end do:

c1charpol := Determinant(c1 - Q*IdentityMatrix(4)):
d1charpol := Determinant(d1 - Q*IdentityMatrix(4)):
c1d1charpol := Determinant(c1.d1 - Q*IdentityMatrix(4)):
#c1d1d1charpol := Determinant(c1.d1.d1 - Q*IdentityMatrix(4)):

rh[17,1] := -Determinant(c1) + 1:
rh[18,1] := tr4(C1) - 4:
rh[19,1] := -Determinant(d1) + 1:
rh[20,1] := tr4(D1) - 4:
rh[21,1] := -coeff(c1d1charpol, Q, 1) - 3:
rh[22,1] := tr4(c1.d1.d1) - (4+1/103):

# We use the R matrix from the QR decomposition of jac to find an optimal
# solution to the linear system.

(Q1,R1) := QRDecomposition(Transpose(jac)):

trsl := Transpose(jac).MatrixInverse(Transpose(R1).R1).rh:

# print("jac ", evalf(jac, 10)):
# print("R1 ", evalf(R1, 10)):
# print("rh ", evalf(rh, 10)):
# print("trsl ", evalf(trsl, 10)):

# The actual matrix differentials are dc1, dd1

dc1 := Matrix(4,4,f):
dd1 := Matrix(4,4,f):

for i from 1 to 4 do for j from 1 to 4 do dc1[i,j] := trsl[4*(i-1)+j, 1] end do end do:
for i from 1 to 4 do for j from 1 to 4 do dd1[i,j] := trsl[4*(i-1)+j+16, 1] end do end do:

# nc, nd are norms of differentials dc1, dd1, and are used to see whether
# we need a further iteration.

nc := Norm(dc1):
nd := Norm(dd1):

# Update c1, d1 and adjust determinants
```

```
# If nc, nd are too big we need to adjust by an appropriate fraction of dc1, dc2

fac := max( 1, round( (nc+nd)*5 ) ):
print( "fac ", fac ):

# Adjust matrices

c1 := c1 + dc1/fac:
d1 := d1 + dd1/fac:

# Fix determinants of c1, d1

c1 := c1 / Re(Determinant(c1)^(1/4)):
d1 := d1 / Re(Determinant(d1)^(1/4)):

print("dc1 ", evalf(dc1, 5)):
print("dd1 ", evalf(dd1, 5)):
print("trc ", evalf(tr4(c1),100)):
print("trd ", evalf(tr4(d1),100)):
print("trcd ", evalf(tr4(c1.d1),100)):
print("trcdd ", evalf(tr4(c1.d1.d1),100)):

return c1, d1, nc, nd:

end proc:

#------------------------------------------------------------------------------#

# Start of main part of program

# Set numerical precision to 2.5 times required accuracy

Digits := 100:

# Enter SL(2,C) generators of figure-eight knot group

a0 := Matrix([[1,1],[0,1]]):
b0 := Matrix([[1,0],[(-1+sqrt(3)*I)/2, 1]]):

# Convert generators to matrices in SL(4,R);
# c0, d0 are the matrices a1, b1 in the text

c0 := evalf(Sl2cToSo31(a0),200):
d0 := evalf(Sl2cToSo31(b0),200):

# Compute inverses

C0 := MatrixInverse(c0):
D0 := MatrixInverse(d0):

# To check relator, uncomment this statement
# print(c0.d0.c0.D0.C0 - d0.c0.D0.C0.D0):
```

```
# Apply small "random" perturbation to each generator

pertc := Matrix([[0,1,0,0],[0,0,1,0],[0,0,0,1],[0,0,0,0]]):
pertd := Matrix([[2,1,0,0],[2,-1,1,5],[3,-4,-3,3],[3,-4,2,1]]):

c1 := c0 + (1/10^8)*pertc:
d1 := d0 + (1/10^8)*pertd:

# Adjust determinant to 1

c1 := c1 / Re(Determinant(c1)^(1/4)):
d1 := d1 / Re(Determinant(d1)^(1/4)):

# Perform Newton iteration until increments have norm less than required accuracy

for count from 1 to 100000 do
print("count:  ", count):
(c1, d1, nc, nd) := NewtonIteration(c1, d1):

Digits := max(100, round( -3*log(min(nc,nd)) ) ):
print("digits ", Digits):

if(nc < 10^(-100) and nd < 10^(-100)) then break end if:

end do:

# Change generators to u1 = c1.d1.d1, v1 = c1.d1;
# u1, v1 are the matrices c1, d1 in the text

u1 := c1.d1.d1:
v1 := c1.d1:

(eigval, eigvec) := Eigenvectors(v1):

u2 := MatrixInverse(eigvec).u1.eigvec:
v2 := MatrixInverse(eigvec).v1.eigvec:

print("u2", evalf(u2, 10)):
print("v2", evalf(v2, 10)):

conj := Matrix(4,4,f):

conj[1,1] := 1:
conj[2,2] := 1/u2[1,2]:
conj[3,3] := 1/(u2[1,2]*u2[2,3]):
conj[4,4] := 1/(u2[1,2]*u2[2,3]*u2[3,4]):

u2a := MatrixInverse(conj).u2.conj:

print("u2a", evalf(u2a, 10)):

vec1 := <1,0,0,0>:
vec2 := <0,1,0,0>:
```

```
conj := <vec1 | vec2 | u2a.vec1 | u2a.vec2>:

u3 := MatrixInverse(conj).u2a.conj:
v3 := MatrixInverse(conj).v2.conj:

U3 := MatrixInverse(u3):
V3 := MatrixInverse(v3):
```

Appendix 2

Maple Script for Stages 4, 5 of the Computation

```
with(CurveFitting):
with(LinearAlgebra):

# udata, vdata contain the coefficients of entries of the matrices u, v,
# as linear combinations of 1, beta, gamma, beta*gamma; here
# beta = sqrt(17 - 4*z) and gamma = sqrt(10 - 4*z + 6*beta) , namely
# the quantities involved in the real roots of the characteristic polynomial
# of u, 1 - 3*Q + z*Q^2 - 3*Q^3 + Q^4 .
# The Newton program was run for y = trace(v) = 4 + 1/k
# (k = 100, 101, ...  , 110).
# It was observed that the middle coefficient z was 1 - y^2/(y + 1) ;
# at the moment this is all guesswork, as the source of the data is
# a collection of floating-point numbers, but it will be verified later.

udata := Matrix(4,4):
vdata := Matrix(4,4):

# umat, vmat will be exact versions of u, v, with entries that are functions
# of y .  These entries are formed by evaluating the above linear
# combinations.  At the end of the program we'll check that the matrices
# umat, vmat do satisfy the defining relation of the figure-eight knot group,
# independently of y .

umat := Matrix(4,4):
vmat := Matrix(4,4):

# The data for each matrix entry is a 5x11 matrix.  Each row consists of the
# coefficients a, b, c, d, e of a linear relation
# a.1 + b.beta + c.gamma + d.beta.gamma + e.w = 0 , where w is the
# matrix entry.  The 11 rows correspond to the 11 values of y dealt with
# in the Newton program.

for i from 1 to 4 do for j from 1 to 4 do udata[i,j] := Matrix(5,11) end do end do:
for i from 1 to 4 do for j from 1 to 4 do vdata[i,j] := Matrix(5,11) end do end do:

# Next we read in udata , vdata from the file lin_all .

#read("C:/Documents and Settings/Marvalisa Payne/My Documents/Thesis/v44");

read("lin_all"):

# We shall now fit a polynomial curve to each set of 11 values of a ;
# this is repeated for b, c, d, e , thus getting 5 polynomial curves for
# each matrix entry.  If the polynomials have degree less than 11 - 1 = 10 ,
# they'll be convincing (though they will be checked rigorously later).
# Maple will do polynomial interpolation on a sequence of points, given
# the sequence of x-coordinates and the sequence of y-coordinates.

xdata := Vector(11):
ydata := Vector(11):
```

```
# Recall that the middle coefficient z is 1 - y^2/(y+1) .

z := 1 - y^2/(y+1):

# We have to spell gammma with an extra "m" so as not to confuse with the
# Euler number.

beta := sqrt(17 - 4*z):
gammma := sqrt(10 - 4*z + 6*beta):

# Each matrix entry is a combination of 1, beta, gammma, beta*gammma .

basis := [1, beta, gammma, beta*gammma]:
pol := Vector(5):

# With 20/20 hindsight, multiply the results by suitable monomials so as
# to get genuine polynomials.  These monomials are conveniently placed
# in matrices umult, vmult.

umult := [[0,0,2*y^6,2*y^5], [0,0,4*y^7,2*y^6], [0,0,2*y^4,4*y^6], [0,0,2*y^7,2*y^4]]:

for row from 1 to 4 do for col from 3 to 4 do
for gen from 1 to 5 do

for i from 1 to 11 do xdata[i] := i end do:
for i from 1 to 11 do ydata[i] := udata[row,col][gen,i] end do:

pol[gen] := PolynomialInterpolation(xdata,ydata,x):

pol[gen] := simplify( subs( x = 1/y - 99, pol[gen] )*umult[row,col] ):

#print(row, col, gen, pol[gen] ):

end do:

umat[row, col] := 0:
for i from 1 to 4 do umat[row, col] := simplify(umat[row, col] - pol[i]*basis[i]/pol[5]) end
do:

end do end do:

vmult := [[0,0,8*y^6,2*y^6], [0,0,4*y^7,8*y^6], [0,0,2*y^5,4*y^6], [0,0,4*y^8,2*y^5]]:

for row from 1 to 4 do for col from 3 to 4 do
for gen from 1 to 5 do

for i from 1 to 11 do xdata[i] := i end do:
for i from 1 to 11 do ydata[i] := vdata[row,col][gen,i] end do:

pol[gen] := PolynomialInterpolation(xdata,ydata,x):

pol[gen] := simplify( subs( x = 1/y - 99, pol[gen] )*vmult[row,col] ):
```

```
print(row, col, gen, pol[gen] ):

end do:

vmat[row, col] := 0:
for i from 1 to 4 do vmat[row, col] := simplify(vmat[row, col] - pol[i]*basis[i]/pol[5]) end
do:

end do end do:

# umat[1,1] , umat[2,2] are the two real roots of the characteristic polynomial
# of the 4x4 matrix u (this is how we guessed that beta, gamma would be needed
# to express the matrix entries).  Also, vmat[3,1] = vmat[4,2] = 1, since the
# third and fourth basis vectors were the images under v of the two chosen
# eigenvectors of u.

umat[1,1] := (3 + beta + gammma)/4:
umat[2,2] := (3 + beta - gammma)/4:
vmat[3,1] := 1:
vmat[4,2] := 1:

# We now have exact matrices umat, vmat, containing a parameter y.
# Next we get Maple to construct their inverses.  Since determinants
# are 1, we can use the adjoint function of Maple.

Umat := simplify(Adjoint(umat)):
Vmat := simplify(Adjoint(vmat)):

# Simplify all matrix entries

for i from 1 to 4 do for j from 1 to 4 do
umat[i,j] := simplify(evala(Expand(umat[i,j]))):
vmat[i,j] := simplify(evala(Expand(vmat[i,j]))):
Umat[i,j] := simplify(evala(Expand(Umat[i,j]))):
Vmat[i,j] := simplify(evala(Expand(Vmat[i,j]))):
end do end do:

r11 := vmat:
r12 := simplify(r11.Umat):
r13 := simplify(r12.Vmat):
r14 := simplify(r13.umat):
r15 := simplify(r14.umat):

r21 := Vmat:
r22 := simplify(r21.umat):
r23 := simplify(r22.vmat):
r24 := simplify(r23.Umat):

rel := simplify(r15 - r24):
```

Appendix 3

Exact Matrix Entries

Let

$$u = 3 - v - \frac{1}{v} \quad , \quad \alpha = \sqrt{17 - 4u} \quad , \quad \beta = \sqrt{10 - 4u + 6\alpha} \ .$$

Each matrix entry is of form

$$-\frac{1}{t_5}\,(t_1 \ + \ t_2\alpha \ + \ t_3\beta \ + \ t_4\alpha\beta) \ ,$$

where t_i is a polynomial in v $(1 \le i \le 5)$. The table below gives the five polynomials t_i for each matrix entry in columns 3, 4.

Entries of the Matrix c_4 :

$(1\,,\,3)$

$$\begin{aligned} t_1 &= -4(1 - 11v + v^2)(1 + v + v^2)(2 - v + 2v^2) \\ t_2 &= 0 \\ t_3 &= v(6 - 17v - 23v^2 - 17v^3 + 6v^4) \\ t_4 &= -v(2 - 15v - v^2 - 15v^3 + 2v^4) \\ t_5 &= 8v(1 - 11v + v^2)(1 + v + v^2) \end{aligned}$$

$(1\,,\,4)$

$$\begin{aligned} t_1 &= -2(-1 + v)(1 + v + v^2)(-1 - 5v + 3v^2)(4 + 5v + 4v^2) \\ t_2 &= 2(-1 + v)v(1 + v + v^2)(-9 - 11v - 3v^2 + 2v^3) \\ t_3 &= -v(1 + v)(2 + v)(-3 - v + v^2)(4 + 5v + 4v^2) \\ t_4 &= v(1 + v)(-2 - 22v - 13v^2 - 17v^3 + 9v^4) \\ t_5 &= 8v(1 - 11v + v^2)(1 + v + v^2)(4 + 5v + 4v^2) \end{aligned}$$

$(2\,,\,3)$

$$\begin{aligned} t_1 &= 2(-1 + v)v(1 + v + v^2)(4 + 5v + 4v^2) \\ t_2 &= -2(-1 + v)(1 + v + v^2)(2 + 2v + 3v^2) \\ t_3 &= -(1 + v)(1 - v + 3v^2)(4 + 5v + 4v^2) \\ t_4 &= v(1 + v)(1 + 2v)(3 + v + v^2) \\ t_5 &= 8(1 + v + v^2)(-1 - v - 2v^2 + v^3) \end{aligned}$$

$(2\,,\,4)$

$$\begin{aligned} t_1 &= -4(1 - 11v + v^2)(1 + v + v^2)(2 - v + 2v^2) \\ t_2 &= 0 \\ t_3 &= -v(6 - 17v - 23v^2 - 17v^3 + 6v^4) \\ t_4 &= v(2 - 15v - v^2 - 15v^3 + 2v^4) \\ t_5 &= 8v(1 - 11v + v^2)(1 + v + v^2) \end{aligned}$$

$(3\,,\,3)$
$$\begin{aligned} t_1 &= -4(3+v)(1-11v+v^2)(1+v+v^2) \\ t_2 &= 0 \\ t_3 &= -4+11v+26v^2+18v^3+3v^4 \\ t_4 &= -v(-3-14v-2v^2+v^3) \\ t_5 &= 8(1-11v+v^2)(1+v+v^2) \end{aligned}$$

$(3\,,\,4)$
$$\begin{aligned} t_1 &= 2(1+v+v^2)(4+5v+4v^2)(-2+11v+18v^2-10v^3+v^4) \\ t_2 &= -2(-1+v)v(1+v+v^2)(8-15v-11v^2+3v^3) \\ t_3 &= v(4+5v+4v^2)(-7-8v-11v^2-4v^3+3v^4) \\ t_4 &= -v(-4-11v-2v^2+v^3+10v^4-5v^5+2v^6) \\ t_5 &= -8v(1-11v+v^2)(1+v+v^2)(4+5v+4v^2) \end{aligned}$$

$(4\,,\,3)$
$$\begin{aligned} t_1 &= 2(5+v)(1+v+v^2)(4+5v+4v^2) \\ t_2 &= 2(-1+v)(1+v+v^2)(-4-11v-8v^2+2v^3) \\ t_3 &= (4+5v+4v^2)(2+4v+3v^2-v^3+v^4) \\ t_4 &= -(v(2+v)(-2-v+v^2+3v^3)) \\ t_5 &= -8(1+v+v^2)(-1-v-2v^2+v^3) \end{aligned}$$

$(4\,,\,4)$
$$\begin{aligned} t_1 &= -4(3+v)(1-11v+v^2)(1+v+v^2) \\ t_2 &= 0 \\ t_3 &= 4-11v-26v^2-18v^3-3v^4 \\ t_4 &= v(-3-14v-2v^2+v^3) \\ t_5 &= 8(1-11v+v^2)(1+v+v^2) \end{aligned}$$

Entries of the Matrix d_4 :

$(1\,,\,3)$
$$\begin{aligned} t_1 &= -6(1-11v+v^2)(1+v+v^2)(2+v+3v^2) \\ t_2 &= -2v(3+v)(1-11v+v^2)(1+v+v^2) \\ t_3 &= -v(1+2v)(-3-8v-13v^2-10v^3+v^4) \\ t_4 &= -3v(1-8v-5v^2-10v^3+v^4) \\ t_5 &= 8v(1-11v+v^2)(1+v+v^2) \end{aligned}$$

$(1\,,\,4)$
$$\begin{aligned} t_1 &= -2(-1+v)(-1-6v+v^2)(1+v+v^2)(4+5v+4v^2) \\ t_2 &= 6v(1+v+v^2)(3+3v-v^2+v^3) \\ t_3 &= -3v(4+5v+4v^2)(-2-2v+v^3) \\ t_4 &= v(-2-22v-20v^2-22v^3+v^4+2v^5) \\ t_5 &= 8v(1-11v+v^2)(1+v+v^2)(4+5v+4v^2) \end{aligned}$$

$(2\,,\,3)$

$$\begin{aligned} t_1 &= 0 \\ t_2 &= -4(1+v+v^2)(-1-v-2v^2+v^3) \\ t_3 &= -(4+5v+4v^2)(1-v+2v^2+v^3) \\ t_4 &= 3v(1+3v+2v^2+v^3) \\ t_5 &= 8(1+v+v^2)(-1-v-2v^2+v^3) \end{aligned}$$

$(2\,,\,4)$

$$\begin{aligned} t_1 &= -6(1-11v+v^2)(1+v+v^2)(2+v+3v^2) \\ t_2 &= -2v(3+v)(1-11v+v^2)(1+v+v^2) \\ t_3 &= v(1+2v)(-3-8v-13v^2-10v^3+v^4) \\ t_4 &= 3v(1-8v-5v^2-10v^3+v^4) \\ t_5 &= 8v(1-11v+v^2)(1+v+v^2) \end{aligned}$$

$(3\,,\,3)$

$$\begin{aligned} t_1 &= 6(1-11v+v^2)(1+v+v^2) \\ t_2 &= -2(1-11v+v^2)(1+v+v^2) \\ t_3 &= -(-1-3v+v^2)(2-v+2v^2) \\ t_4 &= 3v(-1-3v+v^2) \\ t_5 &= -8(1-11v+v^2)(1+v+v^2) \end{aligned}$$

$(3\,,\,4)$

$$\begin{aligned} t_1 &= 2(2+v)(-1+7v)(1+v+v^2)(4+5v+4v^2) \\ t_2 &= -2v(2+v)(1+v+v^2)(-3+7v-18v^2+2v^3) \\ t_3 &= v(2+v)(4+5v+4v^2)(-3-2v-5v^2+v^3) \\ t_4 &= -(v(2+v)^2(-1+v^2+3v^3)) \\ t_5 &= -8v(1-11v+v^2)(1+v+v^2)(4+5v+4v^2) \end{aligned}$$

$(4\,,\,3)$

$$\begin{aligned} t_1 &= -2(-3+v)(2+v)(1+v+v^2)(4+5v+4v^2) \\ t_2 &= -2(-2+v)(2+v)(1+3v)(1+v+v^2) \\ t_3 &= (2+v)(1+2v)(4+5v+4v^2) \\ t_4 &= -v(2+v)(-3-2v+2v^3) \\ t_5 &= -8(1+v+v^2)(-1-v-2v^2+v^3) \end{aligned}$$

$(4\,,\,4)$

$$\begin{aligned} t_1 &= 6(1-11v+v^2)(1+v+v^2) \\ t_2 &= -2(1-11v+v^2)(1+v+v^2) \\ t_3 &= (-1-3v+v^2)(2-v+2v^2) \\ t_4 &= -3v(-1-3v+v^2) \\ t_5 &= -8(1-11v+v^2)(1+v+v^2) \end{aligned}$$

Vita

Marvalisa Mealing Payne was born Marvalisa Jennette Mealing in Detroit, Michigan on October 22, 1963. In 1986, she received a Bachelor of Art degree in Sociology from Oberlin College in Oberlin, Ohio. In 1987, she began working for the Internal Revenue Service (IRS) as a tax examining assistant. There she met her husband Phillip. From 1990 through 1997, Marvalisa worked as a paralegal. She was a paralegal for the Federal Deposit Insurance Corporation (FDIC) in Knoxville, Tennessee and in Atlanta, Georgia; she was a paralegal for the law firm of Jackson and Kelly in Charleston, West Virginia. In 2000, Marvalisa, her husband and their three daughters moved to Hambach, Germany which is roughly 2.5 miles north of Schweinfurt. After her return to the States in 2002, she received her Bachelor of Science degree in Mathematics from Augusta State University in 2003.

Marvalisa began her graduate education at the University of Tennessee in 2003. While at Tennessee, she worked as a graduate teaching assistant and teaching associate in the 2003/2004 and 2004/2005 school years respectively. Marvalisa completed the requirements for a Master of Science degree in Mathematics from the University of Tennessee in May 2005.