

Trains Help

Overview

This is an implementation of Bestvina and Handel's algorithm for determining train tracks of surface homeomorphisms (*Topology* **34** (1995), 109-140). It works for orientation-preserving homeomorphisms of punctured closed orientable surfaces only.

An isotopy class of surface homeomorphisms is represented by the induced graph map g on a graph G embedded in the punctured surface X , which deformation retracts onto G . The algorithm works by repeatedly modifying G until either it has found an explicit reducing curve for the isotopy class, or it has found a graph map $g:G\rightarrow G$ which is the simplest possible. If the transition matrix for g has spectral radius 1, then the isotopy class is finite order: otherwise, the isotopy class is pseudo-Anosov, and $g:G\rightarrow G$ can be used to construct a Thurston train track and train track map.

The starting graph map can be entered [explicitly](#): this is the only option when the surface X has non-zero genus. For isotopy classes on punctured disks, the starting graph map can be constructed from a [braid](#) (the mapping class group of the n -punctured disk is isomorphic to the n -braid group mod its centre), or from a collection of periodic orbits of Smale's [horseshoe](#) map (in this case the isotopy class is that of the horseshoe acting on the disk punctured at these periodic orbits). Graph maps can also be [loaded](#) from (and [saved](#) to) disk. In addition, there is a limited [batch](#) file capability, in which a sequence of isotopy classes in an external file is processed.

The Bestvina-Handel algorithm can then be run on the graph map, either in its [totality](#), or for a given number of [steps](#). In addition, there is a limited [batch](#) file capability, in which a sequence of isotopy classes in an external file is processed.

The program provides the following information about graph maps:

At any stage

- The [graph map](#) itself, and (in the case of graph maps constructed from braids/horseshoe orbits) how it is [embedded](#) in the surface.
The [transition matrix](#) of the graph map, and its [characteristic polynomial](#).

For pseudo-Anosov isotopy classes, once the algorithm has run

Details of the [gates and infinitesimal edges](#) at each vertex.

Information about the [singularities](#) of the invariant foliations.

In the horseshoe orbit case, the [topological train track type](#).

For reducible isotopy classes, once the algorithm has run

Details of the [reducing curves](#): either an invariant subset of edges or (if there is an efficient fibred surface) the gates and infinitesimal edges at each vertex, with at least one vertex not being connected by infinitesimal edges.

New features

Many new features have been added to the program in the last few months (to June 2007). Among the main ones are:

The program now keeps track of the [Embedding](#) of graphs, when the initial graph map is entered as a braid or horseshoe orbit collection.

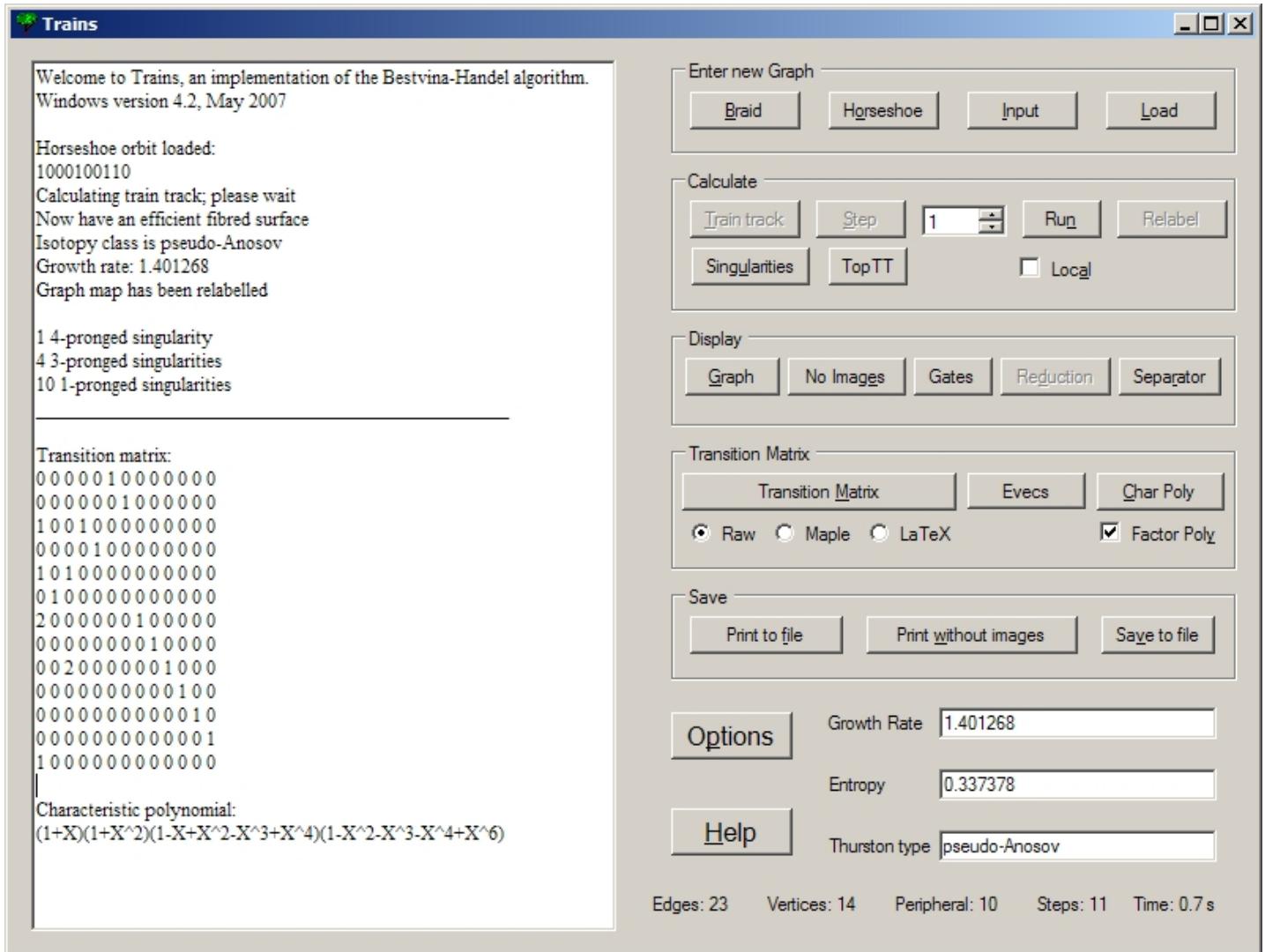
It is possible to obtain [transition matrices](#), together with their [characteristic polynomials](#) and the [eigenvectors](#) corresponding to their dominant eigenvalues.

A summary of the [singularity structure](#) of pseudo-Anosovs can be generated.

Many new commands are available in [batch files](#). Batch files can also be entered directly, rather than loaded from disk.

Please note that these new commands are only available in the Windows version of the program. I am no longer maintaining other versions.

The Main Window



The main window is divided into two parts: the [output window](#) on the left, and the button panel on the right. The main window can be resized, provided that enough space is left for the button panel.

The components of the button panel

Note that keyboard shortcuts are provided for most of the buttons: press ALT with the underlined character in the button caption.

[Braid](#) [Horseshoe](#) [Input](#) [Load](#)
[Train track](#) [Step](#) [Run](#) [Relabel](#)
[Singularities](#) [Top TT](#)
[Graph](#) [No Images](#) [Gates](#) [Reduction](#) [Separator](#)
[Transition Matrix](#) [Evecs](#) [Char Poly](#)
[Print to file](#) [Print without images](#) [Save to file](#)
[Options](#)

The *Growth rate* box displays the spectral radius of the transition matrix of the graph map currently held in memory, provided this matrix is irreducible. The *Entropy* box displays the natural log of the growth rate.

The *Thurston type* box displays the Thurston type of the graph map currently held in memory, provided that the algorithm has been run in its entirety.

The bottom line of the button panel displays the number of edges, vertices, and peripheral edges in the current graph map, together with the number of algorithm steps which have been run since the last time the [Train track](#) or [Step](#) buttons were pressed, and the amount of time taken by the most recent operation (buttons in the *Enter new graph*, *Calculate*, and *Transition matrix* panels are timed, those in the *Display* and *Save* panels are not).

The output window

The output window is where the output of the program is displayed.

You can edit its contents, but any such editing will be ignored by the program: the window has no input function.

You can print a horizontal line across the output window (to separate different calculations) by pressing the **Separator** button.

Right-clicking on the output window brings up a context menu with three options:

Save session: saves the contents of the output window to a text file.

Change font.

Clear window.

How graph maps are represented

The graphs used in the Bestvina-Handel algorithm are embedded as spines of a punctured surface, and the graph maps are those induced on these graphs by homeomorphisms of the punctured surface. If you enter a graph/graph map which does not arise in this way, then the program will produce unpredictable results.

In fact, the algorithm only requires one piece of information about the graphs beyond their abstract graph structure, namely the cyclic order of edges around each vertex.

The program therefore maintains the following information about the graph and graph map:

A list of directed edges, each labelled with a positive integer.

A list of vertices, each labelled with a positive integer.

For each edge, its initial and final vertices, and its image edge-path. This image is given as a sequence of non-zero integers, the positive entries representing edges traversed in accordance with their orientation, the negative entries representing edges traversed in the opposite direction.

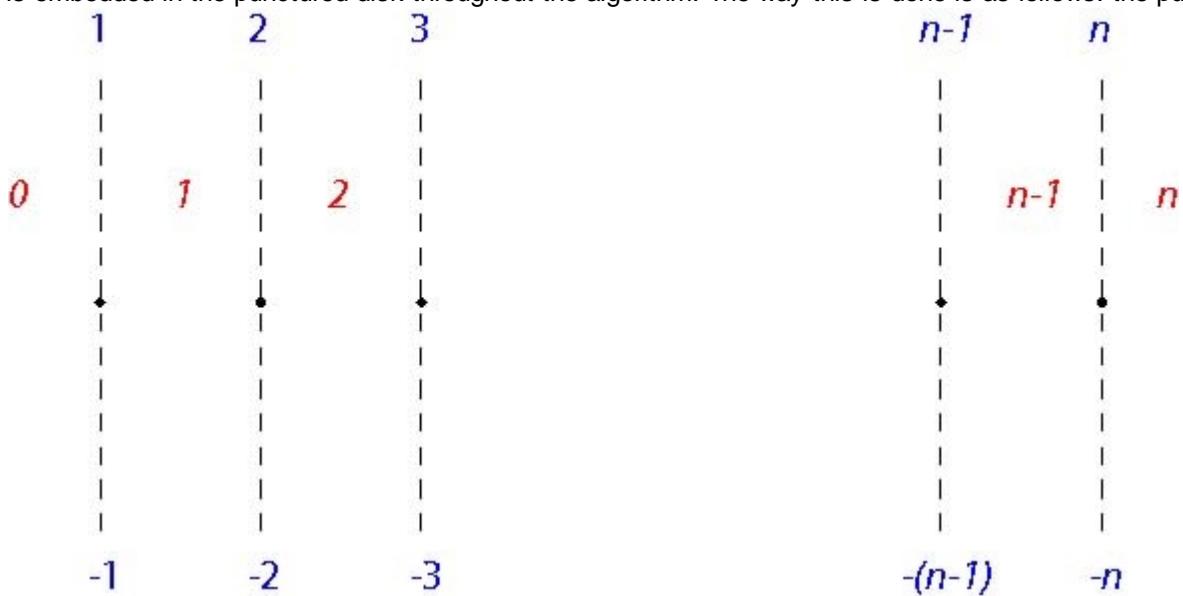
For each vertex, its image vertex, and a list of edges around the vertex in cyclic (anticlockwise) order. This list consists of non-zero integers, positive entries representing initial segments of edges and negative entries representing final segments of edges.

In addition the program maintains information about the way that graphs are embedded in the punctured disk, when isotopy classes are entered using the [Braid button](#) or the [Horseshoe button](#) (although this information is not required for the algorithm itself). See the help page on [Embedding information](#) for details.

See the [Graph map information example](#) for further clarification.

Embedding Information

When a graph map is entered using the [Braid button](#) or the [Horseshoe button](#), the program keeps track of how the graph is embedded in the punctured disk throughout the algorithm. The way this is done is as follows: the punctured disk



is divided into $n+1$ regions, labelled (in red) 0 through n , by (dotted) arcs through the punctures - these arcs should be regarded as having their endpoints on the boundary of the disk. The upper and lower parts of the i th arc are labelled i and $-i$ respectively.

The program keeps track:

For each vertex, which of the (red) regions it is located in.

For each edge, which of the (dotted) arc segments it crosses between its initial and its final vertex.

Of course, it would be very nice if the program could use this information to draw a picture of the graph, but I haven't been able to do this.

See the [graph map information example](#) for clarification.

Entering graph maps

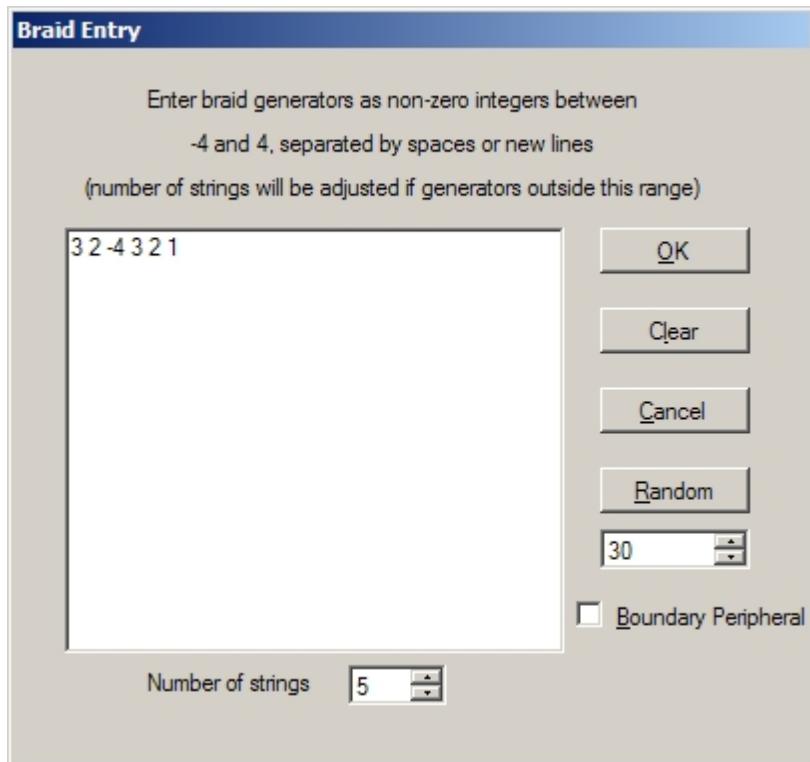
[Entering isotopy classes on the punctured disk as braids](#)

[Entering isotopy classes on the punctured disk as collections of horseshoe periodic orbits](#)

[Entering graph maps explicitly](#) (this is the only way of working with higher genus surfaces).

[Loading graph maps from disk](#)

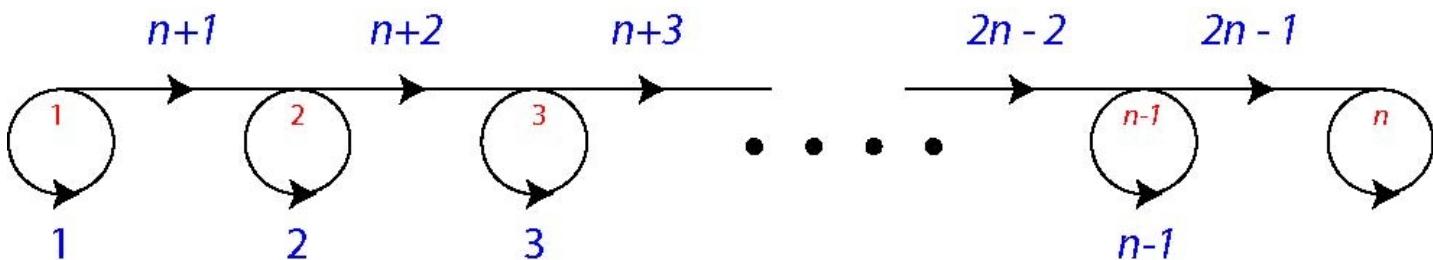
The Braid Button



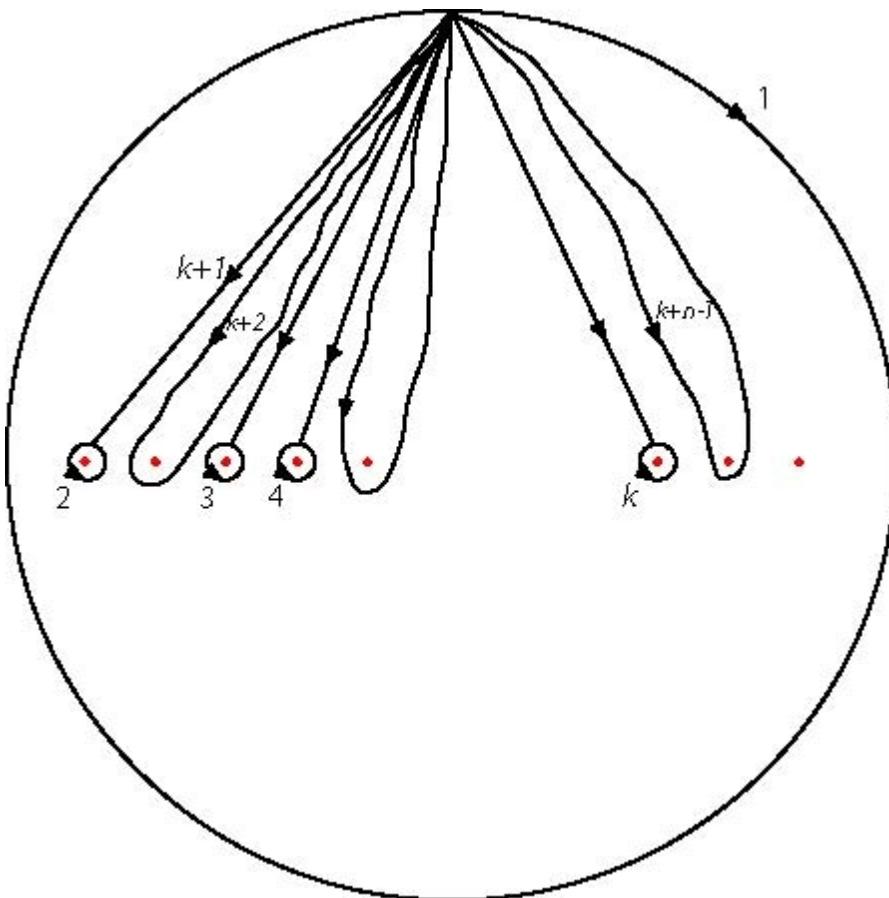
Pressing the **Braid** button brings up the Braid Entry dialog box shown above: in the example above, the 5-braid $\sigma_3\sigma_2\sigma_4^{-1}\sigma_3\sigma_2\sigma_1$ is being entered. The braid is entered as a sequence of non-zero integers (representing the Artin braid generators and their inverses), separated by any number of spaces and new lines.

Please Note: by default, the convention used for the braid generators is that of Birman's book *Braids, links and mapping class groups*: that is, σ_i represents an anti-clockwise swapping of punctures i and $i+1$. This is the opposite of the standard convention. There is an [option](#) to use the standard convention instead.

When the braid is submitted (by pressing **OK**), assuming it is valid, a graph map is loaded describing the action of the isotopy class on the n -punctured disk corresponding to the braid on the standard graph depicted below (the edge labels are given in blue, and the vertex labels in red).



Checking the **Boundary Peripheral** check box gives an alternative initial graph, in which there is a peripheral loop surrounding the boundary of the disk. This graph is as shown below:



Here there is a peripheral loop (labels 2 through k) about each puncture which is not in the orbit of the rightmost puncture. A single main edge surrounds each puncture which is in the orbit of the rightmost puncture, except for the rightmost puncture itself.

Pressing the **Clear** button clears the braid input window.

Pressing the **Cancel** button closes the Braid Entry dialog box without loading a new graph map.

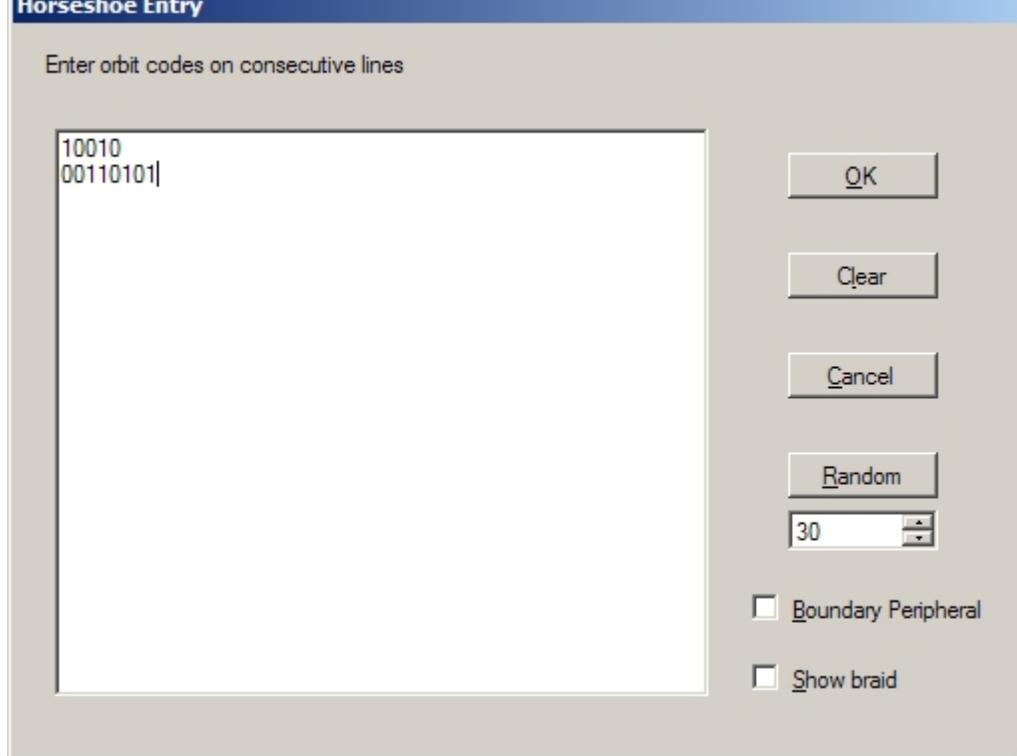
Pressing the **Random** button adds the number of braid generators given in the edit box below it to the end of the braid input window. These generators are picked randomly subject to the condition that there are no "obvious" cancellations, i.e. g is never followed immediately by $-g$.

The braid entered will be interpreted as having the **number of strings** given at the bottom of the dialog box, unless it makes no sense, in which case the number of strings is increased to the smallest number for which the braid does make sense. (For example, if the braid 3 2 -4 3 2 1 is entered when the number of strings is set to 3, then the number of strings will be increased to 5 (and a warning message is issued in the [output window](#)).

The Horseshoe Button

Horseshoe Entry

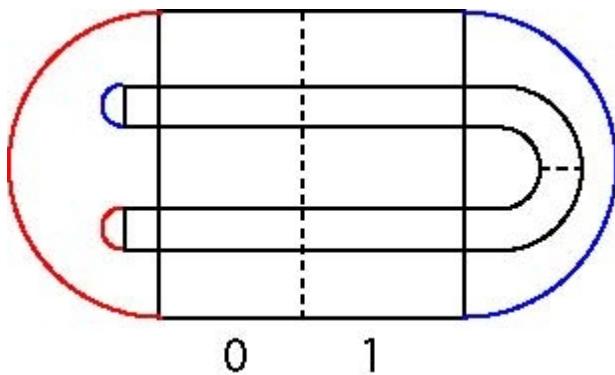
Enter orbit codes on consecutive lines



Pressing the **Horseshoe** button brings up the Horseshoe Entry dialog box shown above: in the example above, the pair of horseshoe periodic orbits with codes 10010 and 00110101 is being entered.

When the horseshoe orbit collection is submitted (by pressing **OK**), the corresponding horseshoe braid is calculated, and a graph map is loaded as described in the [Braid Button](#) help page.

The model of the horseshoe shown below is used:



An **error message** is given if any of the codes describes an orbit whose period is less than the length of the code (e.g. 1010), or if two of the codes describe the same orbit (e.g. 10010 and 00101).

Checking the **Boundary Peripheral** check box gives an alternative initial graph, in which there is a peripheral loop surrounding the boundary of the disk (see the [Braid button](#) help page for details of this graph).

Pressing the **Clear** button clears the horseshoe orbit input window.

Pressing the **Cancel** button closes the Horseshoe Entry dialog box without loading a new graph map.

Pressing the **Random** button adds a random string with the number of symbols given in the edit box below the button at the end of the input window.

If the **Show braid** checkbox is checked, then the horseshoe braid is shown in the output window when the orbit collection is submitted. **Note** that this braid is given in Artin generators using the Birman convention, even if the *Standard convention option* has been selected. (see the [Braid Button](#) help page for details).

The Input Button

Pressing the **Input** button makes it possible to enter a graph and graph map explicitly. This is the only way to use the program to study examples on surfaces other than the punctured disk.

Full details are given in the [explicit graph map entry examples](#).

The program performs some rudimentary sanity checking on graph maps entered this way, but doesn't perform a complete check that they can be realised by the action of an isotopy class on a spine of the surface. **If the graph map you enter cannot be thus realised, the output of the program will be unpredictable.**

The Load Button

Pressing the **Load** button brings up a file open dialog box, enabling the user to load a graph map which was previously saved using the [Save Button](#).

Running the algorithm

[Running the algorithm in its entirety](#)

[Running a specified number of steps of the algorithm](#)

[Batch file capability](#)

The "Train Track" button

Pressing the **Train track** button runs the Bestvina-Handel algorithm on the graph map currently held in memory. On completion the graph map is [relabelled](#).

The algorithm can be interrupted by pressing the [interrupt](#) button.

The Step Button

Pressing the **Step** button runs the Bestvina-Handel algorithm on the graph map currently held in memory for the specified number of steps, or until it terminates. If the algorithm does terminate, the graph map is [relabelled](#).

The algorithm can be interrupted by pressing the [interrupt](#) button.

Note that, because of the way the algorithm is implemented, many of the steps clean up after themselves by performing other steps. These supplementary steps are not displayed, and are not included in the step count.

The Interrupt Button

This button appears when the algorithm is running, as a result of pressing either the [Train track](#) or the [Step](#) button.

Pressing it interrupts the algorithm at the end of the current step (if the graph map is very complicated, it may take a long time for this step to be completed).

The Run Button

The program has a limited capability to run batch files, that is to read a file (probably computer-generated) consisting of a list of braids/horseshoe orbits to be processed consecutively.

Pressing the **Run** button causes the program to prompt for and run a batch file, with default extension .btt. If the **Local** checkbox is checked, a dialog box will instead be displayed for entry of the batch file.

Each command in the batch file must be on a single line (the exception is the **BR** command, for which the list of braid generators can be split over several lines), and only one command can appear on each line.

The following commands are recognised within batch files:

%

Anything following % on any line is ignored.

TO *filename*

Specifies the file to which subsequent output is directed. The command **TO CON** directs output to the output window. (Note that the output will not appear until the batch file has been run in its entirety.)

Default: **TO CON**

OUT *outputSpecifier*

Tells the program what output to produce for each isotopy class processed. The *outputSpecifier* is a combination of the symbols **t**, **g**, **b**, **d**, **p**, **s**, **/**, and **.** These direct the program to output, respectively:

t: The Thurston type.

g: The growth rate.

b: The braid/horseshoe orbit itself.

d: Full details of the graph map as produced by the [Graph button](#).

p: The characteristic polynomial of the main edge transition matrix.

m: The main edge transition matrix, in the chosen format (see the **RAW**, **LATEX**, and **MAPLE** commands).

s: In the pseudo-Anosov case, the singularity structure.

/: A new line.

.. A space.

Default: OUT b/t.g/

STR *n*

Tells the program that subsequent braids should be considered as being on *n* strings. The command **STR AUTO** tells the program to assume that each braid is on the least number of strings for which it makes sense.

Default: **STR AUTO**

PREC *n*

Tells the program that subsequent floating point values (growth rates) should be output to *n* decimal places. This does not affect the internal precision with which calculations are carried out. If *n*<0 then the precision is set to 0, and if *n*>14 then the precision is set to 14.

Default: PREC 12

BR *braid*

Specifies a braid, given as a sequence of non-zero integers **terminated with 0**. The program computes the train track of the braid, and then outputs the information specified by **OUT** to the location specified by **TO**.

RANDOMBR *number_of_braids* *number_of_strings* *number_of_generators*

The program processes *number_of_braids* random braids with the given number of strings and generators. The number of strings must be at least 3, the number of braids cannot exceed 100000, and the number of generators cannot exceed 1000.

HS *horseshoe_periodic_orbit*

Specifies a horseshoe periodic orbit, given as a word in the symbols 0 and 1. The program computes the train track of the corresponding horseshoe braid, and then outputs the information specified by **OUT** to the location specified by **TO**.

RANDOMHS *number_of_orbits* *period*

The program processes *number_of_orbits* random horseshoe orbits of the given *period*. The period must be at least 3, and the number of orbits cannot exceed 100000.

SAVE *filename*

Saves the current graph map to the specified file, with default extension .grm.

PRINT *text_to_print*

Simply echoes the *text_to_print* wherever the output has been directed.

BOUNDARYPERIPHERAL, BOUNDARYNONPERIPHERAL

Tells the program whether or not to make the boundary of the disk a peripheral loop when setting up initial graph maps.

Default: BOUNDARYNONPERIPHERAL

Synonyms: BP, BNP

SHORTSING, LONGSING

Sets the output of singularity structure to short or long format, respectively (see the [Singularities Button](#) help page for more details).

Default: LONGSING

Synonyms: SS, LS

IFPA, IFFO, IFRED, IFRESET

These commands enable you to tell the program to ignore braids/horseshoe orbits which aren't of specified Thurston type. For example, IFPA ignores all but pseudo-Anosov examples. If this is followed (or preceded) by IFFO, then it will ignore all but pseudo-Anosov and finite order examples. IFRESET resets to the initial state, in which output is always generated, but the program is ready to accept further restrictive IF commands.

FACTOR, NOFACTOR

Tells the program whether or not to factorise characteristic polynomials.

Default: FACTOR

BEEP

Generates a beep. This can be useful for tracking progress through the batch file. See also the [Beep option](#).

RAW, LATEX, MAPLE

Sets the output format for transition matrices.

[Examples](#)

The Relabel Button

Each edge and vertex in the graph is assigned an integer label. When algorithm operations are applied to the graph, any new edges and vertices created are assigned labels which have not previously been used. This means that after several steps have been carried out, the largest edge and vertex labels can be much greater than the number of edges and vertices. Pressing the **relabel** button causes the labels to be reassigned, so that the labels used are consecutive integers starting with 1. Moreover, the edge labels are so assigned that peripheral edges have the lowest labels (and are labelled in order of the puncture to which they are associated), followed by preperipheral edge, and finally main edges.

Once the algorithm terminates, the graph is automatically relabelled.

Getting information about graph maps

[Getting detailed information about the graph and graph map](#)

[Getting information about the singularity structure of a pseudo-Anosov](#)

Getting information about the [transition matrix](#), its [Characteristic polynomial](#), and its leading [eigenvectors](#)

The Graph Button

Pressing the **Graph** button gives detailed information about the graph and graph map: it provides the following information:

The vertices of the graph. For each vertex the image vertex is given, together with a list of edges emanating from the vertex in anti-clockwise cyclic order.

The edges of the graph. For each edge the following information is given: initial and final vertices; type (main, peripheral, or pre-peripheral); the number of the puncture it is associated with in the peripheral case; and its image edge path.

If the algorithm has been run, this information is supplemented by the results of the algorithm. The Thurston type of the isotopy class is given, and

If the class is reducible, and this has been detected because the transition matrix for the main edges is reducible, a collection of edges corresponding to an invariant subgraph is given (see also the help page on the [Reduction button](#)).

If reducibility has been detected because there is an efficient fibred surface with a vertex at which not all of the gates are connected by infinitesimal edges, or in the pseudo-Anosov case, a list of gates and infinitesimal edges at each vertex is given.

The same information can be saved to file using the [Print to file button](#).

See the [Graph map information example](#) for clarification.

The "No Images" Button

Pressing the **No Images** button causes the same information as produced by the [Graph Button](#) to be shown in the output window, except that edge images are omitted. This is useful in cases where the growth rate of the graph map is very large, and hence the edge images are very long.

The Gates Button

When there is an efficient fibred surface, pressing the **Gates** button gives a list of gates and infinitesimal edges at each vertex. See the [graph map information example](#) for more details.

The Reduction Button

If the graph map held in memory has been found to represent a reducible isotopy class, then pressing the **reduction** button will give details of reducing curves. There are two possibilities:

If reducibility has been detected because the main edge transition matrix is reducible, an invariant subset of the main edges is given. See example 1 of the [reducible examples](#).

If reducibility has been detected because there is an efficient fibred surface, with a vertex at which the gates are not all connected by infinitesimal edges, then details of the gates and infinitesimal edges at each vertex are given. See example 2 of the [reducible examples](#).

The TopTT Button

Pressing the **TopTT** button displays the topological train track type of a horseshoe periodic orbit of pseudo-Anosov braid type.

See *Experimental Math* **11** (2002), 271-288 for more details.

The Singularities Button

If the graph map held in memory has been found to represent a pseudo-Anosov isotopy class, then pressing the **singularities** button gives information about the singularities of its invariant foliations. The following information is provided:

The number of singularities with each number of prongs.

The orbits of the singularities.

The locations of the singularities. A p -pronged singularity at vertex n arising because there are p infinitesimal edges joining the p gates there is denoted (n) . Other singularities are given in the format $[\text{edge-path}]$, where the edge-path is a list of oriented edges of the graph. Walking around this edge-path, the complementary region of the train track in which the singularity is contained is on your right.

See the end of the [graph map information example](#) for clarification.

There is an [option](#) to display less complete singularity information: only the number of singularities with each number of prongs.

The "Transition Matrix" Button

Pressing the **Transition Matrix** button displays the transition matrix of the graph map in the output window. It can be displayed in raw format (just the numerical entries), or in *Maple* or *LaTeX* formats (suitable to be pasted into a *Maple* worksheet or a *LaTeX* file respectively).

There is an [option](#) to create a matrix corresponding only to the main edges of the graph, or to use all of the edges of the graph.

The image of each edge is encoded in a *column* of the transition matrix: the rows and columns correspond to the edges of the graph in the same sequence as given by their labelling. For example, the following graph map (the peripheral edges 1 through 4 have been omitted)

Edge number 5 from vertex 1 to vertex 4:

Type: Main

Image is: -5 -1 5 -2 6 3 7 -4 -7 -3 -6 2 -5

Path (1 \rightarrow 1):

Edge number 6 from vertex 4 to vertex 2:

Type: Main

Image is: 5 -2 6 3 7

Path (1 \rightarrow 2): -2

Edge number 7 from vertex 2 to vertex 3:

Type: Main

Image is: -7 -3 -6 2 -5 1 5 -2 -5 -1 5 -2 6

Path (2 \rightarrow 3): 3

yields the transition matrix

```
3 1 4
2 1 2
2 1 1
```

The "Char Poly" Button

Pressing the **Char Poly** button displays the Characteristic Polynomial of the [transition matrix](#) of the graph map in the output window. The polynomial can be displayed factored or unfactored.

There is an [option](#) to calculate the Characteristic Polynomial of the matrix corresponding only to the main edges of the graph, or to use all of the edges of the graph.

The Evecs Button

Provided that the [transition matrix](#) of the graph map is irreducible, pressing the **Evecs** button displays its column and row eigenvectors corresponding to its Perron-Frobenius eigenvalue. Note that only the eigenvector entries corresponding to the main edges of the graph are given.

The conventions used mean that, in the pseudo-Anosov case, the entries of the column eigenvector give the unstable measures of arcs across Markov boxes transverse to the unstable foliation, and the entries of the row eigenvector give the stable measures of arcs across Markov boxes transverse to the stable foliation.

Saving information to disk

[Saving a graph map in human-readable format](#)

[Saving a graph map in machine-readable format](#)

The "Print to file" Button

Pressing the **Print to file** button saves full details of the current graph map, as generated by the [Graph button](#), to a specified file.

The "Print without images" Button

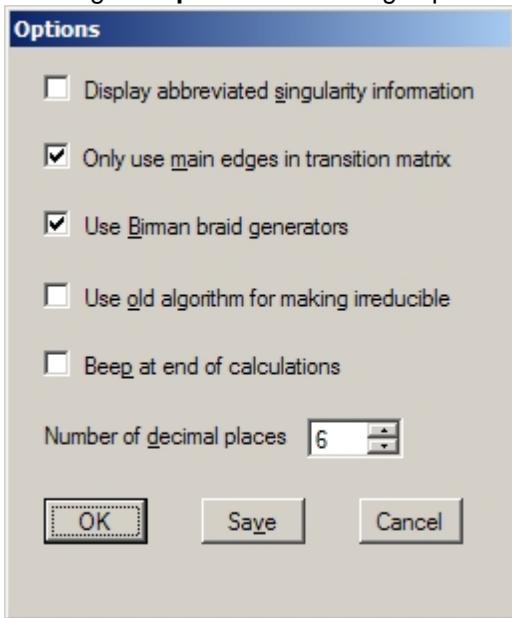
Pressing the **Print without images** button saves details of the current graph map, omitting edge images (as generated by the [No Images button](#)), to a specified file.

The "Save to File" Button

Pressing the **Save to file** button saves the graph map to disk so that it can later be [reloaded](#) into the program.

Setting options

Pressing the **Options** button brings up the options dialog box:



Pressing the **OK** button sets the selected options and closes the dialog box.

Pressing the **Save** button sets the selected options, closes the dialog box, and saves the options to be used when the program is run in future (the options are saved in the windows registry).

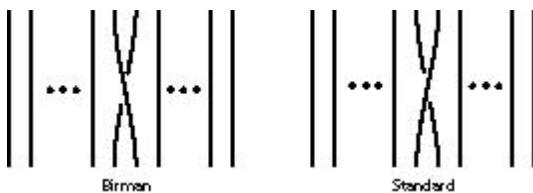
Pressing the **Cancel** button closes the dialog box without changing any options.

The options are as follows:

Display abbreviated singularity information. When this is selected, pressing the [Singularities Button](#) only provides a summary of the number of singularities which each number of prongs, instead of more detailed information.

Only use main edges in transition matrix. This option controls whether the [Transition Matrix](#) and [Char Poly](#) buttons take account of all edges in the graph (main, peripheral, and preperipheral), or of the main edges only. Note that the [Evecs Button](#) only ever causes eigenvector entries corresponding to the main edges to be displayed.

Use Birman braid generators. This option enables you to choose the convention used for the Artin braid generators. The generator $\backslash\sigma_i$ has the following meanings in the two conventions:



Use old algorithm for making irreducible. In an earlier version of the program, the algorithm step *Making irreducible* would consume well over half the running time for complicated examples. A new version of this step reduces this time substantially, but I haven't been able to prove that it always works: it should never give incorrect final answers, but may cause the program to go into an infinite loop. Selecting this option causes the old, slow and safe, version of this step to be used instead. Please [contact](#) me if you find any examples where the two versions behave differently (except insofar as running time is concerned).

Beep at end of calculations. When this option is selected, there is a beep after running a batch file, or after using the *Train track* or *Step* buttons, if processing took more than 3 seconds. See also the beep [batch file command](#).

Number of decimal places. This option enables you to choose the number of decimal places used in displaying results of floating point calculations. It doesn't affect the number of decimal places used internally in calculations.

Examples

The following examples are available:

[An example showing how to reconstruct the embedding of a graph in the punctured disk, and how to construct a Thurston train track.](#)

[Two examples showing how to obtain reducing curves.](#)

[Two examples showing how to enter graph maps explicitly.](#)

[Three examples illustrating the program's batch file capability.](#)

Graph map information example

Consider the following graph and graph map (obtained from running the algorithm on the horseshoe periodic orbit 1000100):

Graph on surface with 7 peripheral loops:

Vertex number 1 with image vertex 5:

Edges at vertex are: 1 -1 8

Region 1

Vertex number 2 with image vertex 3:

Edges at vertex are: 3 -3 10 -9

Region 2

Vertex number 3 with image vertex 4:

Edges at vertex are: 5 -5 13

Region 4

Vertex number 4 with image vertex 1:

Edges at vertex are: 7 -7 -12

Region 6

Vertex number 5 with image vertex 7:

Edges at vertex are: -2 -8 9 2

Region 1

Vertex number 6 with image vertex 2:

Edges at vertex are: -6 -11 12 6

Region 5

Vertex number 7 with image vertex 6:

Edges at vertex are: -4 14 -13 4

Region 3

Vertex number 8 with image vertex 8:

Edges at vertex are: -10 -14 11

Region 3

Edge number 1 from vertex 1 to vertex 1:

Type: Peripheral about puncture number 1

Image is: 2

Path (1 -> 1): 1 -1

Edge number 2 from vertex 5 to vertex 5:

Type: Peripheral about puncture number 2

Image is: 4

Path (1 -> 1): -2 2

Edge number 3 from vertex 2 to vertex 2:

Type: Peripheral about puncture number 3

Image is: 5

Path (2 -> 2): -3 3

Edge number 4 from vertex 7 to vertex 7:

Type: Peripheral about puncture number 4

Image is: 6

Note that:

There are 7 peripheral loops, labelled 1 through 7, surrounding the (red) punctures 1 through 7 respectively. Each is oriented anti-clockwise.

Each of the vertices (labelled 1 through 8 in pink) is in the region (given by the blue numbers) specified in the output. The edges around each vertex are in the anticlockwise order given in the output. For example, the edges around vertex 7 are: end of edge 4, start of edge 14, end of edge 13, beginning of edge 4, indicated -4 14 -13 4 in the output.

Each of the main edges 7 through 14 crosses the dotted embedding lines (labelled with red numbers) as indicated in the output. For example, edge 11 goes from region 3 to region 5, crossing line 4 followed by line 5, indicated in the output by

Path (3->5) : 4 5

To obtain a Thurston train track, we use the "gates" information.

Vertex 1:

Gates are: {-1}, {8}, {1}

Infinitesimal edges join 1 to 8, -1 to 8

Vertex 2:

Gates are: {-3}, {10, -9}, {3}

Infinitesimal edges join 3 to 10, -3 to 10

Vertex 3:

Gates are: {-5}, {13}, {5}

Infinitesimal edges join 5 to 13, -5 to 13

Vertex 4:

Gates are: {-7}, {-12}, {7}

Infinitesimal edges join -12 to 7, -12 to -7

Vertex 5:

Gates are: {-8, 9}, {2}, {-2}

Infinitesimal edges join -8 to 2, -8 to -2

Vertex 6:

Gates are: {-11, 12}, {6}, {-6}

Infinitesimal edges join -11 to 6, -11 to -6

Vertex 7:

Gates are: {14, -13}, {4}, {-4}

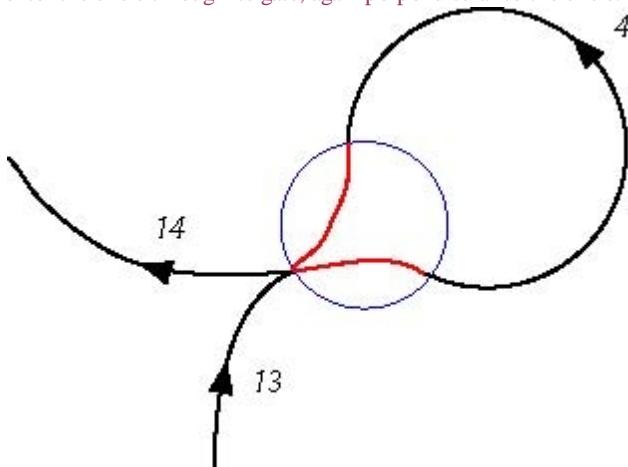
Infinitesimal edges join 4 to 14, -4 to 14

Vertex 8:

Gates are: {-14}, {11}, {-10}

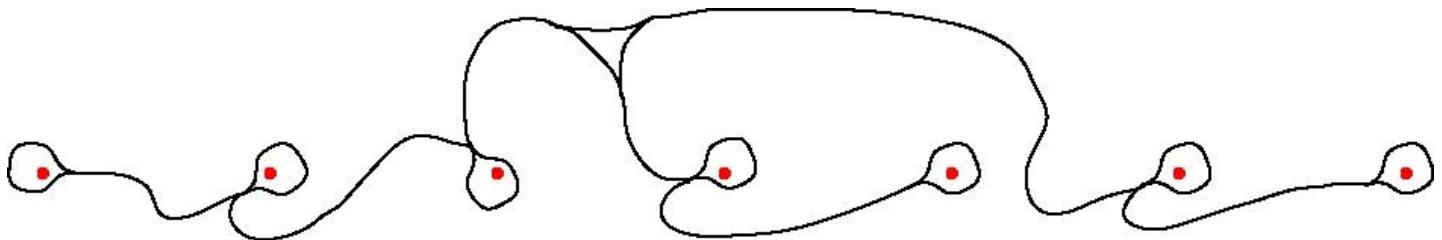
Infinitesimal edges join -14 to -10, -14 to 11, -10 to 11

Surround each vertex with a small circle. Mark a point on this circle for each of the gates specified, in the given cyclic order. Within the circle, join gates with the infinitesimal edges specified (the ends of the infinitesimal edges should be perpendicular to the circle). Let each edge of the graph enter the circle through its gate, again perpendicular to the circle. For example, at vertex 7 we get the following picture:



There are 3 gates on the (blue) circle: one where edges 14 and -13 enter, one where edge 4 enters, and one where edge -4 enters. There are two (red) infinitesimal edges within the circle, one connecting the gates 4 and {14, -13}, the other connecting the gates -4 and {14, -13}.

Carrying out this procedure at each vertex in the graph yields the following Thurston train track:



The singularity information given by the [Singularities button](#) is as follows:

1 4-pronged singularity

[-14 -13 5 13 4 14 11 12 7 -12 6 -11 -10 -9 2 -8 1 8 9 3 10]

1 3-pronged singularity

(8)

7 1-pronged singularities

Period 7 orbit: [-7] -> [-1] -> [-2] -> [-4] -> [-6] -> [-3] -> [-5]

The 4-pronged singularity is on the boundary (coming from the 4 "exterior" cusps in the Thurston train track). As we walk around the train track along the path given (-14 -13 5 13 4 ...) the complementary region with these cusps on its boundary is on our right.

The 3-pronged singularity is at vertex 8, where all 3 gates are connected by infinitesimal edges.

The 7 1-pronged singularities are at the punctures. As we go around each path (e.g. -7), the complementary region containing a cusp corresponding to such 1-prongs is on our right.

Reducible examples

Example 1

Consider the following graph and graph map (obtained from running the algorithm on the horseshoe periodic orbit 100110):

Graph on surface with 6 peripheral loops:

Vertex number 1 with image vertex 5:

Edges at vertex are: 1 -1 7

Region 1

Vertex number 2 with image vertex 3:

Edges at vertex are: 3 -3 9

Region 2

Vertex number 3 with image vertex 1:

Edges at vertex are: 6 -6 -8

Region 5

Vertex number 4 with image vertex 4:

Edges at vertex are: 10 12 11

Region 3

Vertex number 5 with image vertex 6:

Edges at vertex are: -2 -12 -9 2

Region 2

Vertex number 6 with image vertex 7:

Edges at vertex are: -4 -11 8 4

Region 3

Vertex number 7 with image vertex 2:

Edges at vertex are: -5 -10 -7 5

Region 4

Edge number 1 from vertex 1 to vertex 1:

Type: Peripheral about puncture number 1

Image is: 2

Path (1 -> 1): 1 -1

Edge number 2 from vertex 5 to vertex 5:

Type: Peripheral about puncture number 2

Image is: 4

Path (2 -> 2): 2 -2

Edge number 3 from vertex 2 to vertex 2:

Type: Peripheral about puncture number 3

Image is: 6

Path (2 -> 2): -3 3

Edge number 4 from vertex 6 to vertex 6:

Type: Peripheral about puncture number 4

Image is: 5

Path (3 -> 3): -4 4

Edge number 5 from vertex 7 to vertex 7:

Type: Peripheral about puncture number 5

Image is: 3

Path (4 -> 4): -5 5

Edge number 6 from vertex 3 to vertex 3:

Type: Peripheral about puncture number 6

Image is: 1

Path (5 -> 5): -6 6

Edge number 7 from vertex 1 to vertex 7:

Type: Main

Image is: -9

Path (1 -> 4): 2 3 4 5 -5

Edge number 8 from vertex 6 to vertex 3:

Type: Main

Image is: -7

Path (3 -> 5): -4 -5

Edge number 9 from vertex 2 to vertex 5:

Type: Main

Image is: -8

Path (2 -> 2): -2 2

Edge number 10 from vertex 4 to vertex 7:

Type: Main

Image is: 12 -2 -9

Path (3 -> 4): 4

Edge number 11 from vertex 4 to vertex 6:

Type: Main

Image is: 10

Path (3 -> 3):

Edge number 12 from vertex 4 to vertex 5:

Type: Main

Image is: 11

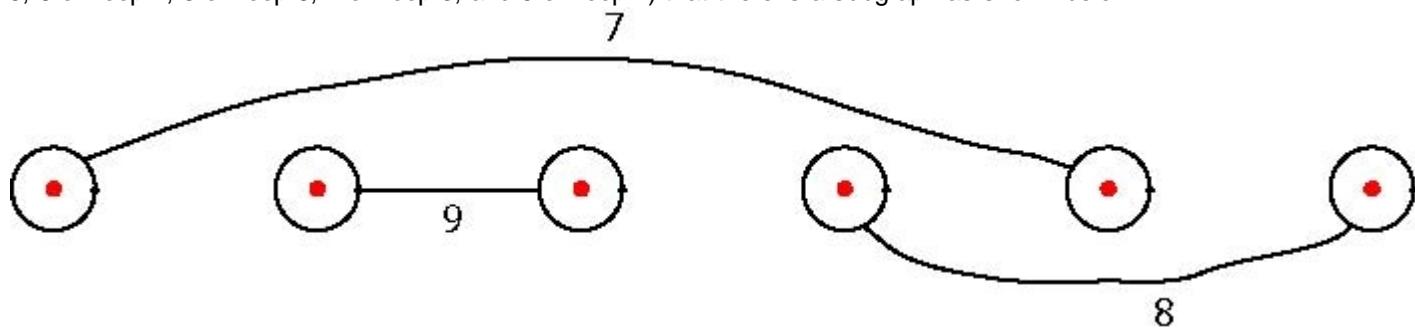
Path (3 -> 2): 3

Reducible Isotopy class

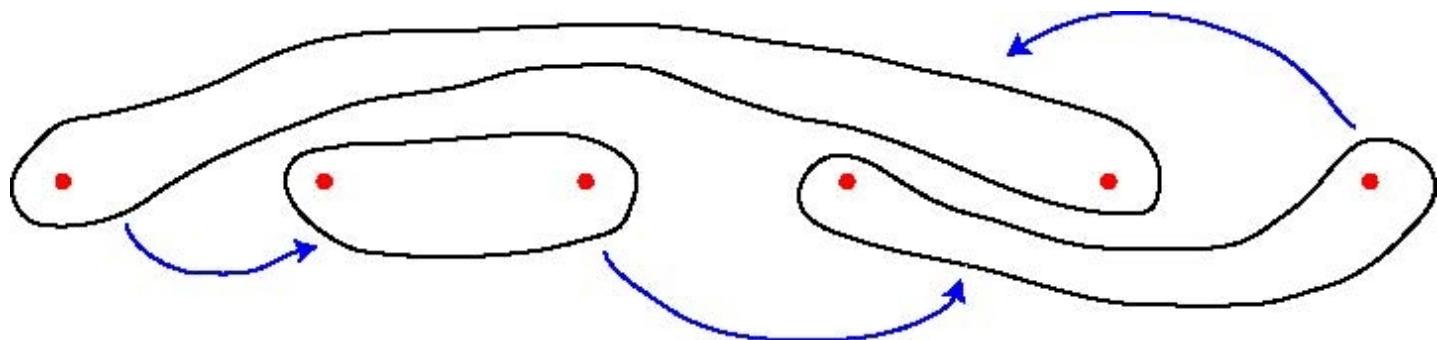
The following main edges and their images constitute an invariant subgraph:

7 8 9

It is clear from the paths of edges 7, 8, and 9 (together with the locations of the vertices: 1 on peripheral loop 1, 7 on loop 5, 6 on loop 4, 3 on loop 6, 2 on loop 3, and 5 on loop 2) that there is a subgraph as shown below:



with edge 7 mapping to edge 9, 9 to 8, and 8 to 7. This gives the following system of reducing curves:



Example 2

Consider the following graph and graph map (obtained from running the algorithm on the 4-braid $\sigma_2^{-1} \sigma_1 \sigma_2 \sigma_3^{-1}$):

Graph on surface with 4 peripheral loops:

Vertex number 1 with image vertex 3:

Edges at vertex are: 1 -1 5

Region 1

Vertex number 2 with image vertex 1:

Edges at vertex are: 3 -3 7

Region 2

Vertex number 3 with image vertex 2:

Edges at vertex are: 4 -4 -6

Region 3

Vertex number 4 with image vertex 4:

Edges at vertex are: 2 -2 -5 -7 6

Region 2

Edge number 1 from vertex 1 to vertex 1:

Type: Peripheral about puncture number 1

Image is: 4

Path (1 -> 1): 1 -1

Edge number 2 from vertex 4 to vertex 4:

Type: Peripheral about puncture number 2

Image is: 2

Path (2 -> 2): 2 -2

Edge number 3 from vertex 2 to vertex 2:

Type: Peripheral about puncture number 3

Image is: 1
Path (2 -> 2): -3 3

Edge number 4 from vertex 3 to vertex 3:
Type: Peripheral about puncture number 4
Image is: 3
Path (3 -> 3): -4 4

Edge number 5 from vertex 1 to vertex 4:
Type: Main
Image is: -6 -5 -1 5
Path (1 -> 2): -2

Edge number 6 from vertex 4 to vertex 3:
Type: Main
Image is: 6 -4 -6 -7
Path (2 -> 3): 3

Edge number 7 from vertex 2 to vertex 4:
Type: Main
Image is: 5
Path (2 -> 2):

Reducible Isotopy class

Vertex 1:

Gates are: {-1}, {5}, {1}

Infinitesimal edges join -1 to 5, 1 to 5

Vertex 2:

Gates are: {-3}, {7}, {3}

Infinitesimal edges join -3 to 7, 3 to 7

Vertex 3:

Gates are: {-4}, {-6}, {4}

Infinitesimal edges join -6 to -4, -6 to 4

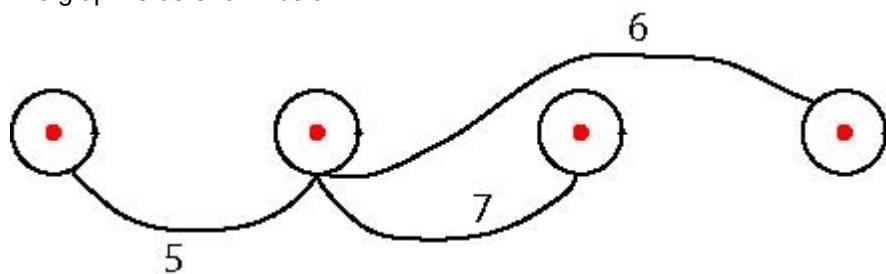
Vertex 4:

Gates are: {-2}, {-5, -7}, {6}, {2}

Infinitesimal edges join -5 to 6

Note that there is only one infinitesimal edge connecting the four gates at vertex 4, so that the gates are not all connected by infinitesimal edges.

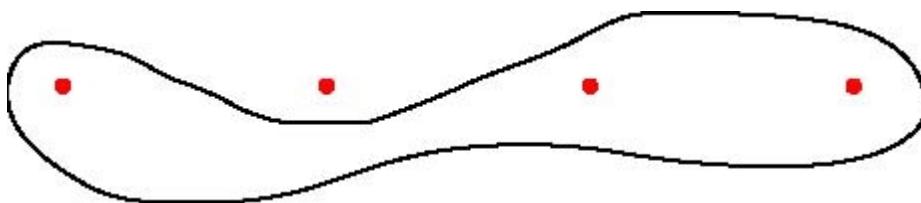
The graph is as shown below:



Adding infinitesimal edges as described in the [graph map information example](#) gives the following picture:



from which we can see that the curve shown below is an invariant reducing curve.



Explicit Graph Map Entry

Example 1

Consider the example on a genus 2 surface given in section 6.1 of Bestvina and Handel's paper (page 129). The edges there labelled a , b , c , and d will here be labelled 1, 2, 3, and 4 respectively.

On pressing the **Input** button, you are prompted for the number of peripheral loops, edges, and vertices in the graph:

Input Graph

Number of peripheral loops	<input type="text" value="0"/>
Number of edges	<input type="text" value="4"/>
Number of vertices	<input type="text" value="1"/>

OK **Cancel**

Next, enter the edges around the (unique) vertex in anticlockwise cyclic order (this has been read off from Figure 14 on page 130).

Vertex Entry

Vertex number 1

Image Vertex	<input type="text" value="1"/>
Edges round vertex	<input type="text" value="1 2 -1 -2 -4 -3 4 3"/>

>> (B) **<< (B)** **Edges** **Submit** **Abort**

Click the **Edges** button to move on to giving information about the edges of the graph. You are prompted to enter the image edge path of each edge: press the **>>** and **<<** buttons to move between edges.

Edge Entry

Edge number 1

Image edge path

>> (F) << (B) Vertices Submit Abort

Edge Entry

Edge number 2

Image edge path

>> (F) << (B) Vertices Submit Abort

Edge Entry

Edge number 3

Image edge path

>> (F) << (B) Vertices Submit Abort

Edge Entry

Edge number 4

Image edge path

>> (F) << (B) Vertices Submit Abort

Finally, hit the **Submit** button and the graph map input is complete.

Example 2

Consider the example on the 4-times punctured sphere given in section 6.2 of Bestvina and Handel's paper (page 136). The edges there labelled a , and c will here be labelled 1, 2, 3, 4, and 5 respectively.

The only substantial difference from example 1 is that there are now peripheral loops in the graph:

Input Graph

Number of peripheral loops	<input type="text" value="3"/>
Number of edges	<input type="text" value="5"/>
Number of vertices	<input type="text" value="3"/>

OK **Cancel**

Thus, when entering each edge, the program needs to know whether or not it is a peripheral edge. If it is, you are also prompted to enter the number of the puncture with which it is associated.

Vertex Entry

Vertex number 1

Image Vertex	<input type="text" value="1"/>
Edges round vertex	<input type="text" value="1 -1 -4"/>

>> (F) **<< (B)** **Edges** **Submit** **Abort**

Vertex Entry

Vertex number 2

Image Vertex	<input type="text" value="2"/>
Edges round vertex	<input type="text" value="2 -2 5 4"/>

>> (F) **<< (B)** **Edges** **Submit** **Abort**

Vertex Entry

Vertex number 3

Image Vertex

Edges round vertex

Edge Entry

Edge number 1

Peripheral Puncture

Image edge path

Edge Entry

Edge number 2

Peripheral Puncture

Image edge path

Edge Entry

Edge number 3

Peripheral Puncture

Image edge path

Edge Entry

Edge number 4

Peripheral

Image edge path

Edge Entry

Edge number 5

Peripheral

Image edge path

Batch file examples

Example 1

The following batch file writes (to hs7.txt) a list of all period 7 horseshoe orbits of pseudo-Anosov braid type, together with their growth rate, the characteristic polynomials of their main edge transition matrices, and their singularity structure (in short format).

```
TO hs7.txt
SHORTSING
OUT b..g/p/s//
```

IFPA

```
HS 1011110
HS 1011010
HS 1001010
HS 1001110
HS 1001100
HS 1000100
HS 1000110
HS 1000010
HS 1000000
```

Example 2

The following batch file processes the three string braid $\sigma_1\sigma_2^{-1}$, the four string braid $\sigma_1\sigma_2\sigma_1\sigma_3$, and the five string braid $\sigma_1\sigma_2\sigma_1\sigma_3$ in turn. For the first example, it displays the braid, Thurston type and growth rate to the output window; for the second, it displays the same information to the file 'test.out', and also saves the resulting train track to the file 'test.grm'; and for the third, it displays full details to 'test.out'

```
PRINT 3-string 1 -2
PRINT -----
BR 1 -2 0
TO test.out
PRINT 4-string 1 -2 3
PRINT -----
BR 1 -2 3 0
SAVE test
STR 5
OUT b/t.g/d//
PRINT 5-string 1 -2 3
PRINT -----
BR 1 -2 3 0
```

Example 3

The following batch file processes 100 random period 16 horseshoe orbits, outputting full information about all those which are reducible to the file hsred16.txt

```
TO hs16red.txt
IFRED
OUT b/d
RANDOMHS 100 16
```

Contact

For more information, or to report errors, contact Toby Hall at t.hall@liv.ac.uk.

Before reporting an error, please make sure that you have read the warnings about explicit graph map input using the [Input](#) button, and about the [Use old algorithm for making irreducible](#) option.