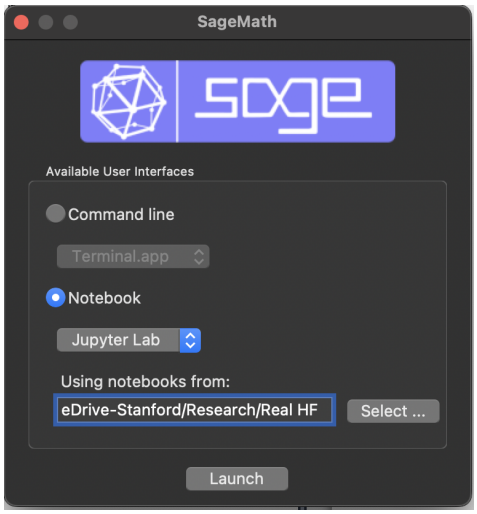
DOCUMENTATION FOR HFR_PYTHON CODE

GARY GUTH AND CIPRIAN MANOLESCU

1. Getting Started

The program is written in a Sage notebook. The simplest way to get started is to install Sage and Jupyter Notebook. Next, download the file HFR_python.ipynb here. Open Sage and choose to open a notebook from the directory containing HFR_python.ipynb.



2. Basic Functionality

Computing the Euler characteristic of the real Heegaard Floer homology of a knot in S^3 is straightforward. To get started, press Shift + Enter to run the first two cells of code. Then, press the "+" icon in the toolbar to create a new cell. Type the following commands:

```
braid_word = [1,-2,1,-2]
H = realHD(braid_word)
H.real_euler_characteristics()
```

Press Shift + Enter again; the output should be:

```
[1,1,-1,1,1]
```

Here, H is an instance of a class "Real_Heegaard_Diagram", which as the name implies, encodes the data of a real Heegaard diagram. The function "realHD" takes as input a braid word w and returns an object recording a real Heegaard diagram for the branched double cover of the braid closure of w. In the example above, H is a real Heegaard diagram object for the double branched cover of the figure eight, 4_1, represented as the closure of the braid $[1, -2, 1, -2]$. The method real_euler_characteristics()

returns the Euler characteristics of the branched double cover of 4_1 in its five (real) Spin^c-structures. There is also a method to compute the total Euler characteristic: type

H.real_total_euler_characteristics()

and press Shift + Enter again to compute the total Euler characteristic. The output should be

3.

3. Real Heegaard Diagrams

The Euler characteristic for $\widehat{HFR}(\Sigma_2(K))$ for a knot in S^3 can be computed combinatorially according to [GM25, Proposition 7.1]. We briefly recall the salient details.

Recall that a real Heegaard diagram for $\Sigma_2(K)$ can be obtained from a Seifert surface F for K which has the property that $S^3 \smallsetminus \nu(F)$ is a handlebody. The boundary of $\nu(F) \cong F \times I$ has an obvious involution with fixed set K. Therefore, we can obtain a real Heegaard diagram for $\Sigma_2(K)$ by choosing alpha curves for $S^3 \smallsetminus \nu(F)$ and reflecting these across K to obtain the beta curves. The genus of the resulting Heegaard splitting is twice that of the Seifert surface F, so $|\mathbb{T}_\alpha \cap M^R|$ grows rather quickly.

Fix a real Heegaard diagram for $\Sigma_2(K)$ with orientable quotient. Number the alpha and beta arcs, and label every intersection point between them. Label the intersection points on the fixed set first: start at the basepoint and follow along the knot labeling intersection points as they appear according to the knot's orientation. The class "Real_Heegaard_Diagram" has attributes (alphas, betas, tau, alpha_int_beta, alpha_int_C), where:

(1) "alphas" is a dictionary whose keys are the indices of the ordered alpha curves whose values are lists consisting of the intersection points between this alpha curve and the beta curves, listed in the order they appear along the alpha curve (following the orientation); "betas" is defined analogously.
(2) The input "tau" is a dictionary encoding the action of τ on the intersection points. Finally, "alpha_int_beta" is a dictionary which associated to the ith intersection point its sign as an intersection point between the alphas and betas.
(3) The input "alpha_int_C" is a dictionary assigning to the ith intersection point its sign as an intersection point between the alpha curves and C (if this point is not on the fixed set, we just assign it the sign of its intersection as a point in $\alpha \cap \beta \subset \Sigma^+$).

From this data, one can compute the Euler characteristic of $\widehat{HFR}(\Sigma_2(K))$. The program generates a list of points in $\mathbb{T}_\alpha \cap M^R$, sorts them into real Spin^c-structures, and computes the associated signed count.

For example, consider the real Heegaard diagram for $\Sigma_2(K)$, where $K = 4_1$ shown in Figure 3.1. The intersection points have been labeled and the alpha and beta curves have been ordered and oriented. Note that the first intersection points are those on the fixed set which are ordered by the location of the basepoint and the orientation of the knot. This data specifies a "Real_Heegaard_Diagram" object.

```
alphas = {0: [0,1,8,7,3], 1: [5,2,4,9,6]}
betas = {0: [0,1,9,7,3], 1: [5,2,4,8,6]}
tau = {0:0,1:1,2:2,3:3,4:4,5:5,6:6,7:7,8:9,9:8}
alpha_int_beta = {0:1,1:-1,2:-1,3:1,4:-1,5:-1,6:1,7:1,8:-1,9:-1}
alpha_int_C = {0:1,1:-1,2:-1,3:-1,4:1,5:1,6:-1,7:1,8:-1,9:-1}

H = Real_Heegaard_Diagram(alphas, betas, tau, alpha_int_beta, alpha_int_C)
```

These objects have a few useful built-in functions. As a sanity check, it can compute the first homology of the branched double cover (it returns its invariant factors), whose product should be the determinant of the knot.

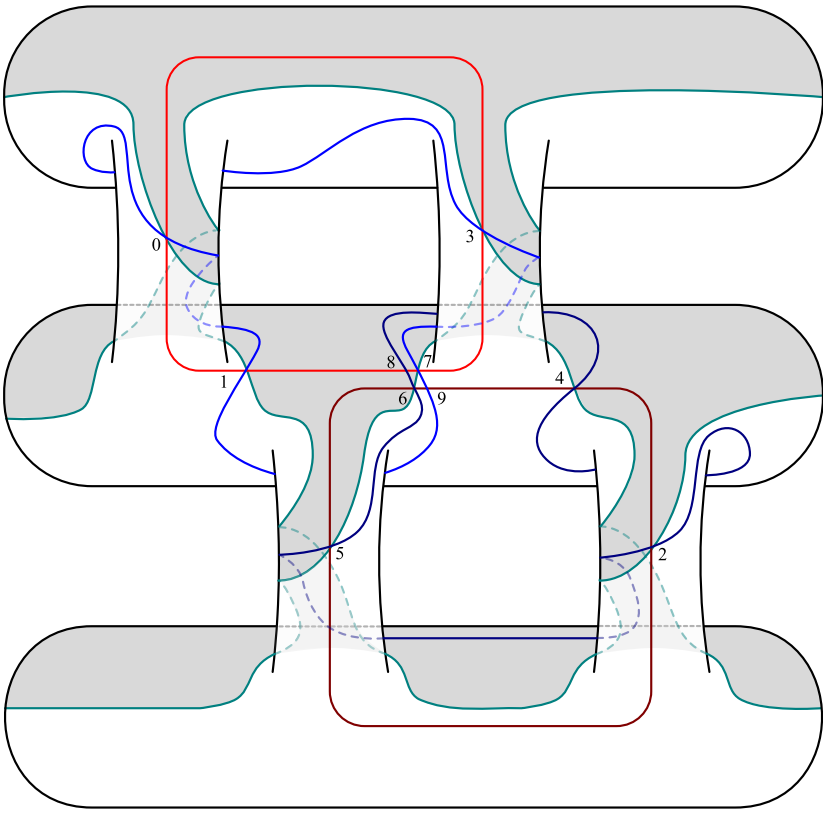


FIGURE 3.1. A labeled real Heegaard diagram for $\Sigma_2(4_1)$.

```
alphas = {0: [0,1,8,7,3], 1: [5,2,4,9,6]}
betas = {0: [0,1,9,7,3], 1: [5,2,4,8,6]}
tau = {0:0,1:1,2:2,3:3,4:4,5:5,6:6,7:7,8:9,9:8}
alpha_int_beta = {0:1,1:-1,2:-1,3:1,4:-1,5:-1,6:1,7:1,8:-1,9:-1}
alpha_int_C = {0:1,1:-1,2:-1,3:-1,4:1,5:1,6:-1,7:1,8:-1,9:-1}

H = Real_Heegaard_Diagram(alphas, betas, tau, alpha_int_beta, alpha_int_C)

H.homology()
```

```
[5]
```

It will also, of course, return the Euler characteristics. The program fixes some arbitrary identification of $\mathrm{Spin}^c(Y)$ with $H_1(Y)$ and returns the Euler characteristics preserving the cyclic ordering.

```
alphas = {0: [0,1,8,7,3], 1: [5,2,4,9,6]}
betas = {0: [0,1,9,7,3], 1: [5,2,4,8,6]}
tau = {0:0,1:1,2:2,3:3,4:4,5:5,6:6,7:7,8:9,9:8}
alpha_int_beta = {0:1,1:-1,2:-1,3:1,4:-1,5:-1,6:1,7:1,8:-1,9:-1}
alpha_int_C = {0:1,1:-1,2:-1,3:-1,4:1,5:1,6:-1,7:1,8:-1,9:-1}

H = Real_Heegaard_Diagram(alphas, betas, tau, alpha_int_beta, alpha_int_C)

H.real_euler_characteristics()
```

```
Starting computation...
Invariant beta points obtained...
Time to prepare for Euler characteristic computation: 0.45 seconds.
Computing Euler characteristics...
Total runtime: 0.45 seconds.
{(0,): -1, (1,): -1, (2,): 1, (3,): -1, (4,): -1}
```

Actually, if $H_1(Y)$ is cyclic, the program will present the Euler characteristics symmetrically.

Our program will also generate the data needed to compute the Euler characteristic of $\widehat{HFR}(\Sigma_2(K))$. It currently takes as input a braid word whose closure is the knot K. Applying Seifert's algorithm to such diagrams yields surfaces whose complements have obvious compressing disks, so, we can extract all the input data just from the braid word. Returning to our example, we can represent $K = 4_1$ as the closure of the braid $[1, -2, 1, -2]$. The function "real_HD" returns a "Real_Heegaard_Diagram" class for K.

```
# 4_1 is the braid closure of the braid [1,-2,1,-2]

H = real_HD([1,-2,1,-2])
print(H.alphas, H.betas,H.tau, H.alpha_int_beta, H.alpha_int_C)
```

```
{2: [3, 8, 7], 3: [2, 4, 9]} {2: [3, 9, 7], 3: [2, 4, 8]} {2: 2, 3: 3, 4: 4, 7: 7, 8: 9, 9: 8} {2: -1, 3: 1, 4: -1, 7: 1, 8: -1, 9: -1}
{2: -1, 3: -1, 4: 1, 7: 1, 8: -1, 9: -1}
```

This data comes from the Heegaard diagram shown in Figure 3.1; the program has removed a few canceling intersection points. Representing K as a braid closure has the advantage that it is easy to choose a collection of compressing disks for the complement of the associated Seifert surface; the disadvantage is that the resulting surface may have needlessly large genus. For particular examples, it is sometimes possible to carefully choose a small free Seifert surface (for instance, the surface for 9_{46} in [GM25, Example 6.5] is genus 1, while applying Seifert's algorithm to this diagram produces a larger genus surface; similarly, representing K as a braid closure also yields a surface of large genus). Though, Seifert's algorithm is not always optimal either.

In general, it would be good to have more efficient methods for computing $|\deg(K)|_{HF}$. As noted above, one wants to find a small genus diagram (which the program does not always do). The second drawback, is that the program cannot pick diagrams with minimal intersections. As noted above, it will eliminate some bigons to reduce the total number of intersections, but it will sometimes choose the alpha curves inefficiently (the total number of intersection point can be reduced by a handle-slide, say, but the program does not know to do this).

For example, for $K = P(-2, 3, 7)$, our program computes $|\deg(K)|_{HF} = 3$ (this took my 2023 Macbook two days). The Seifert genus is 5; hence, the smallest real Heegaard diagram can hope to find will have genus 10 (and indeed, applying Seifert's algorithm to the standard diagram for K yields a free, genus 5 surface.) In this case, our program does indeed construct a genus 10 diagram with many crossings, and so there are an enormous number generators of $\widehat{CFR}(\Sigma_2(K))$.

References

[GM25] Gary Guth and Ciprian Manolescu, *Real Heegaard Floer Homology*, Preprint, ADD (2025).

Department of Mathematics, Stanford University, Building 380, Stanford, California 94305
Email address: gmguth@stanford.edu

Department of Mathematics, Stanford University, Building 380, Stanford, California 94305
Email address: cm5@stanford.edu