Studying Legendrian Invariants with FrontLeg

Shaowei Lin
Department of Mathematics
Stanford University

June 4, 2005

Abstract

The Chekanov-Eliashberg DGA is a powerful tool for helping us to decide if two Legendrian knots are Legendrian isotopic. However, computing it can be an extremely tedious task. FrontLeg is a program created to automate these calculations. In this paper, we examine the algorithm for constructing the DGA and its Poincaré polynomials. We also describe the design, structure and functionalities of FrontLeg, and give suggestions for extension and research.

1 Introduction

A *Legendrian knot* in $\mathbb{R}^3$ is a topological knot where the y-coordinate of any point is given by the slope at the point of the xz-projection of the curve. More precisely, if the knot is given by the embedding

$$f : S^1 \to \mathbb{R}^3, f(t) = (x(t), y(t), z(t))$$

then $y = dz/dx$. A Legendrian *link* is a topological link where each component is Legendrian. Two Legendrian links are said to be *Legendrian isotopic* if there is a smooth isotopy between them through Legendrian links. In this paper, we will primarily be interested in deciding if two given Legendrian links are Legendrian isotopic.

Contact geometry provides a neat and useful language in which to talk about such links. We start by introducing the concept of a contact manifold. At each point p in $\mathbb{R}^3$, pick a 2-dimensional subspace ξ_p of the tangent space $T_p\mathbb{R}^3$. This collection ξ of 2-dimensional subspaces is a plane field. All plane fields can be defined as the kernel of some locally defined 1-form, so let α be the 1-form where $\xi_p = \ker(\alpha_p)$. If $\alpha \wedge d\alpha \neq 0$, we say that ξ is *non-integrable*. In this case, we will also refer to the 1-form α as a *contact form*, and the plane field ξ as a *contact structure*. If ξ is non-integrable, then at every point p, it is impossible to find a smooth 2-dimensional manifold in a small neighborhood of p that is everywhere tangent to ξ.

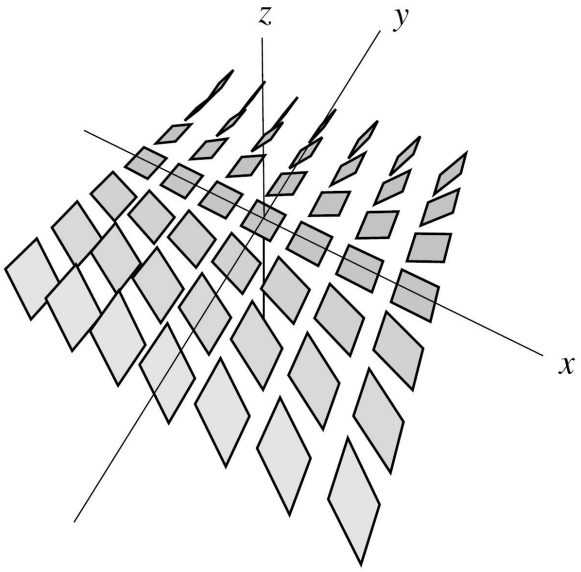

Figure 1: [Et1] The standard contact structure on $\mathbb{R}^3$.

An important example is the *standard* contact form $\alpha = dz - ydx$ on $\mathbb{R}^3$. Its kernel $\ker(\alpha)$ gives the standard contact structure, as shown in Figure 1. Notice that at each point p, the contact plane $\ker(\alpha_p)$ contains every line of slope $dz/dx = y$. A manifold, such as $\mathbb{R}^3$, that is imbued with a contact form and contact structure is called a *contact manifold*. A more complete and precise introduction to contact geometry can be found in [Et1].

We are now ready to define a Legendrian knot in the language of contact geometry: it is a knot that is everywhere tangent to the given contact structure. Legendrian knots can be defined for any contact manifold, but in this paper, we shall investigate only those in the standard contact manifold $\mathbb{R}^3$. Also, to keep the ideas simple, we will often limit our discussion to Legendrian knots, rather than talk generally about Legendrian links. It will be clear to the reader how these ideas can be extended to describe links. In parts where such an extension is ambiguous, we will spell out the necessary details.

1.1 Lagrangian and Front Projections

Like topological knots, Legendrian knots can be studied by looking at their projections onto 2-dimensional planes. Of these projections, two are commonly used: their projection onto the xy plane (the *Lagrangian* projection), and their projection onto the xz plane (the *front* projection). Figure 2 shows the Lagrangian and front projections of the trivial knot.

Let L be a Legendrian knot, and $\pi(L)$ be its Lagrangian projection. From $\pi(L)$, we can obtain the original knot L, up to a translation in the z-direction, by

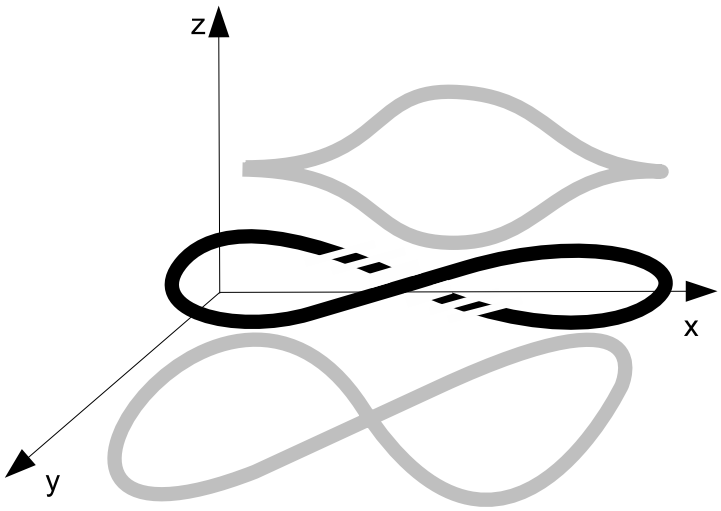

Figure 2: Projections of the trivial knot.

integrating the 1-form ydx along $\pi(L)$ to get the z-coordinate. This is because $dz = ydx$ along the curve. It is, however, difficult to determine by inspection whether a given diagram corresponds to the projection of some Legendrian knot. In [Ch2], Chekanov describes a procedure to accomplish this.

On the other hand, it is much easier to identify front projections of Legendrian knots. One only needs to check that, except at the self-intersection points, there is a unique non-vertical tangent line at each point on the projection. This is because the original Legendrian knot can be obtained by setting the y-coordinate of a point to be the slope of the unique tangent line.

In general, a front projection can have many different types of singularities. For instance, we may have crossings where the projection of the knot intersects itself more than once. Fortunately, since we are only interested in studying Legendrian knots up to isotopy, we can deform the knot slightly so that the singularities of the projection are either semi-cubic cusps or transverse double points. Such a well-behaved projection will be called a *front*, while the corresponding knot will be called a *generic* knot. We will refer to the singularities of fronts as *left* cusps ($\prec$), *right* cusps ($\succ$), and *crossings* ($\times$). Note that at the crossings, we do not need to indicate which is the overlapping knot segment, because the segment with the tangent line of larger slope is the one on top.

Recall that one can perform Reidemeister moves on the projections of topological knots to show that two knots are isotopic. It turns out that there exist similar moves on fronts for demonstrating Legendrian isotopy. Indeed, two generic knots are Legendrian isotopic if and only if one of the fronts can be transformed to the other via a sequence of *Legendrian Reidemeister* moves [Sw]. These moves are illustrated in Figure 3. It is interesting to note that Reide-

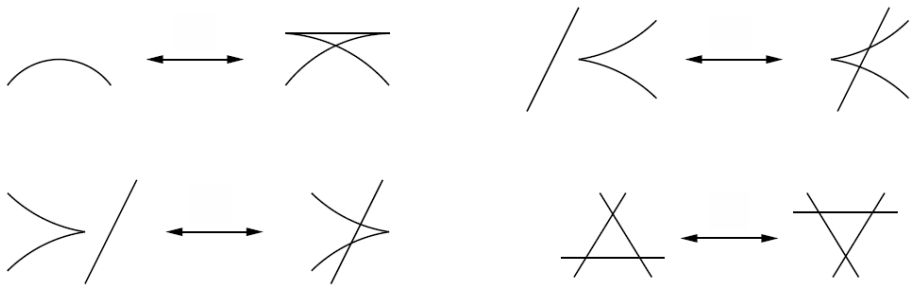

Figure 3: [Ng1] Legendrian Reidemeister moves.

meister moves also exist for the Lagrangian projections of knots. However, the corresponding isotopy theorem is weaker: two knots are Legendrian isotopic *only if* there exist a sequence of Lagrangian Reidemeister moves between their Lagrangian projections [Et1]. The converse is not true because these Reidemeister moves do not preserve certain integral constraints on Lagrangian projections.

1.2 Legendrian Invariants

Given two fronts, the question of whether they represent knots which are Legendrian isotopic is an extremely difficult one. Usually, the first step is to check if the fronts represent knots of the same topological type, since Legendrian isotopic knots are also topologically isotopic. In the unfortunate case that they are, we will need to find either a sequence of Reidmeister moves that shows that the two knots are isotopic, or an invariant that is preserved under Legendrian isotopy but takes different values for the two knots to show that they are non-isotopic.

Two commonly-used *classical* invariants are the rotation number r, and the Thurston-Bennequin number tb. To compute these invariants, we will need to orient the knot. Switching the orientation changes the sign of r but does not change the value of tb. Let Y be the front of an oriented Legendrian knot. Let $c(Y)$ be the total number of cusps. Define $\downarrow(Y)$ to be the number of downward oriented cusps, and $\uparrow(Y)$ the number of upward oriented cusps, as shown in Figure 4 and 5. Also, define $\leftrightarrow(Y)$ to be the number of "left-right" crossings, and $\updownarrow(Y)$ the number of "up-down" crossings, as shown in Figure 6 and 7. Then, we have the following formulas for r and tb:

$$r(Y) = \frac{1}{2}\downarrow(Y) - \frac{1}{2}\uparrow(Y)$$

$$tb(Y) = \leftrightarrow(Y) - \updownarrow(Y) - \frac{1}{2}c(Y)$$

Sometimes, the *Maslov* number $m(Y)$ is used instead of the rotation number. It is given by $m(Y) = 2r(Y)$.

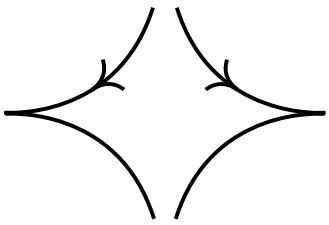

Figure 4: Downward Oriented Cusps.

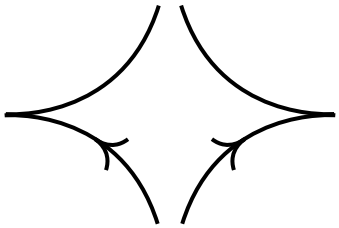

Figure 5: Upward Oriented Cusps.

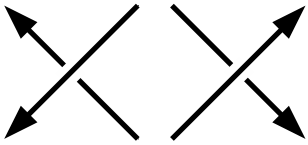

Figure 6: "Left-Right" Crossings.

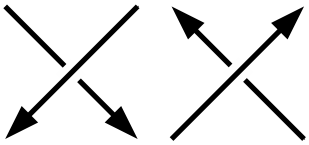

Figure 7: "Up-Down" Crossings.

It turns out that for certain topological types, two knots are Legendrian isotopic if and only if they have the same rotation and Thurston-Bennequin number. This is known to be true for the unknot [EF], torus knots [EH], and the figure eight knot [EH]. However, there are also numerous examples of knots that have the same topological type and classical invariants but are not Legendrian isotopic. Many of these examples were discovered with the help of a new invariant, called the Chekanov-Eliashberg differential graded algebra (DGA). This DGA can be defined for both oriented links, and non-oriented links. For the rest of this paper, we shall limit ourselves to looking at oriented links.

1.3 Chekanov-Eliashberg DGA

The invention of the Chekanov-Eliashberg DGA was motivated by developments in symplectic field theory. In [Ch2], Chekanov describes combinatorially how to construct a differential graded algebra from the Lagrangian projection of a knot. This algebra is invariant up to "stable tame isomorphisms" over Legendrian isotopy. His original algebra was defined over $\mathbb{Z}_2$ with grading over $\mathbb{Z}_m$ where m is the Maslov number of the knot. Etnyre, Ng and Sabloff [ENS] later generalized it to an algebra that is defined over the ring $\mathbb{Z}[t, t^{-1}]$ with grading over $\mathbb{Z}$. We shall be looking at this generalized DGA in this paper.

Ng also went on to investigate how this DGA can be defined for fronts [Ng1]. He developed a procedure to construct, from the front of a knot L, the Lagrangian projection of a knot L' that is isotopic to L. He calls this the *resolution* of the front. This procedure allowed him to translate Chekanov's construction of the DGA for Lagrangian projections into the language of fronts. Ng identified a special class of fronts, called simple fronts, for which the DGA

is easily computable. The next two sections will describe Ng's method of constructing a DGA for the front of a knot and of a link.

1.4 Constructing the DGA for Fronts of Knots

Let Y be the front of a knot L. We call the right cusps and crossings *generators*, ignoring the left cusps. Label these generators $a_1, a_2, \ldots, a_n$. Now, consider the free, non-commutative algebra A with unit 1 that is generated by $a_1, \ldots, a_n$ over the ring $\mathbb{Z}[t, t^{-1}]$. Next, we introduce a grading $deg : A \to \mathbb{Z}$ and differential $\partial : A \to A$ as described below. This gives us a differential graded algebra which we call the Chekanov-Eliashberg DGA for Y. We shall use the symbol a_i to refer interchangeably to both the generator a_i in the front Y and also the generator a_i in the DGA A.

1.4.1 Defining the Degree

The degree is subject to the usual rules $deg(1) = 0$, $deg(ab) = deg(a)+deg(b)$, and $deg(a+b) = \max\{deg(a), deg(b)\}$. The degree of 0 is undefined. In addition, we let the degree of the indeterminate t be $m(Y)$. It remains for us to determine the degree for the generators $a_1, \ldots, a_n$, so that the definition of deg may be extended to the whole of A by the above rules.

Before we do that, let us define the capping path for a generator a_i. If a_i is a crossing, let S be the knot segment at the crossing which has a greater slope. Note that this is also the segment on top. Define the *capping path* P_i to be the directed path that begins at a_i, continues along S in the direction of the orientation of Y, and goes around Y until it returns to a_i. If a_i is a right cusp, define the capping path P_i to be the empty path if a_i is upward oriented, and P_i to be the whole of Y if a_i is downward oriented.

Now, just as in the definition of the rotation number, let $\downarrow(P_i)$ be the number of downward oriented cusps on P_i, and $\uparrow(P_i)$ the number of upward oriented cusps on P_i. Define $m(P_i) = \downarrow(P_i) - \uparrow(P_i)$, which is analogous to our definition of the Maslov number in Section 1.2. We compute the degree of a generator by

$$deg(a_i) = \begin{cases} -m(P_i) + 1 & \text{if } a_i \text{ is a right cusp} \\ -m(P_i) & \text{if } a_i \text{ is a crossing} \end{cases}$$

For a right cusp a_i, the above formula simplifies to give

$$deg(a_i) = \begin{cases} 1 & \text{if } a_i \text{ is an upward oriented cusp} \\ 1 - m(Y) & \text{if } a_i \text{ is a downward oriented cusp} \end{cases}$$

1.4.2 Defining the Differential

We start with Ng's definition of an admissible map [Ng1]. Consider an immersion $f : D^2 \to \mathbb{R}^2$ of the two-dimensional disk onto the plane, such that the boundary of the disk is mapped onto the front Y. We call f an *admissible* map if

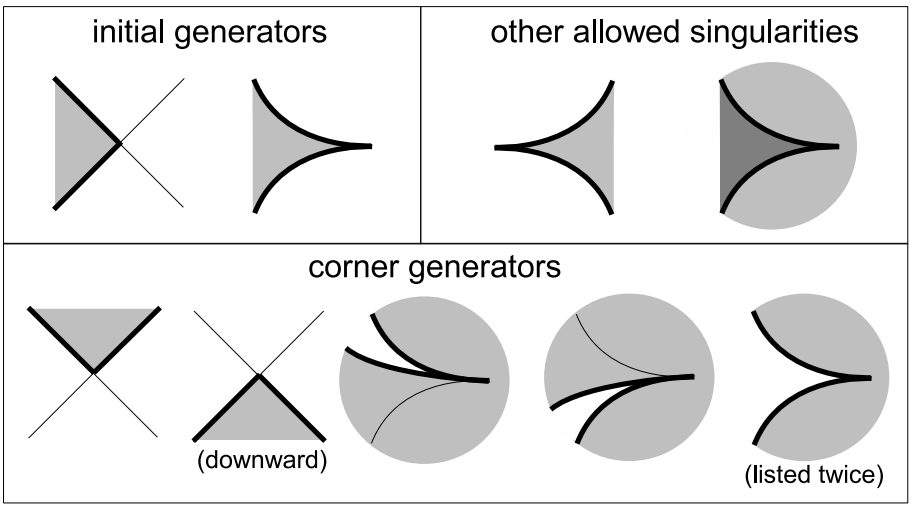

Figure 8: Allowed Singularities.

1. it is smooth except at the cusps and crossings.
2. the image of f near each singularity looks locally like one of the pictures in Figure 8. A darker region indicates that the image overlaps itself.
3. there is exactly one initial generator.

Ng described the DGA in terms of the diffeomorphism classes of the admissible maps. We shall take a slightly different approach, so that it will be easier for us to write a computer algorithm that carries out the construction. Let P be a directed path on Y that begins and ends with the generator a_i. We say that P is an *admissible* path for a_i if

1. there exists an admissible map f that maps the boundary of the two-dimensional disk bijectively to P and has a_i as its initial generator.
2. it begins by traversing upwards from the initial generator a_i, and ends by returning to a_i from below.

We can associate with each admissible path P for a_i a monomial in the algebra A. Let $a_{j_1}, a_{j_2}, \ldots, a_{j_m}$ be the corner generators in the order of traversal on the directed path P. Some corner generators need to be listed twice, as indicated in Figure 8. We count the number of downward corner generators on P which are of even degree. Define $\wedge(P)$ to be $+1$ if this number is even, and -1 if it is odd. Lastly, pick any non-singular point β on Y. Recall that P_j is the capping path for a_j. Now, let $n(P)$ be the signed number of times β is traversed in the direction of the orientation of Y among the paths $P_{j_1}, P_{j_2}, \ldots, P_{j_m}, -P_i$ and $-P$. A path with a negative sign refers to the same path but traversed

in the opposite direction. Note that the value of $n(P)$ does not depend on the choice of β, since the above mentioned paths form a closed cycle on Y. The monomial associated with P is given by

$$\alpha(P) = \wedge(P)t^{n(P)}a_{j_1} \dots a_{j_m}$$

We are finally ready to define the differential for a generator:

$$\partial a_i = \begin{cases} \sum \alpha(P) & \text{if } a_i \text{ is a crossing} \\ 1 + \sum \alpha(P) & \text{if } a_i \text{ is an upward oriented right cusp} \\ t^{-1} + \sum \alpha(P) & \text{if } a_i \text{ is an downward oriented right cusp} \end{cases}$$

where the summation is taken over all admissible paths P with a_i as the initial generator. Also, define $\partial(\mathbb{Z}[t, t^{-1}]) = 0$. The differential is extended to the rest of the algebra by linearity

$$\partial(\lambda a + \mu b) = \lambda\partial(a) + \mu\partial(b) \text{ where } \lambda, \mu \in \mathbb{Z}[t, t^{-1}]$$

and the signed Leibniz rule

$$\partial(ab) = \partial(a)b + (-1)^{deg(a)}a\partial(b).$$

1.5 Constructing the DGA for Fronts of Links

For a front Y of a link L, the construction of the DGA is very similar to that for knots. Here, the DGA is still a free, unital, and non-commutative algebra A generated by $a_1, a_2, \dots, a_n$, one for each of the right cusps and crossings in Y. One main difference is that the algebra is defined over the ring $\mathbb{Z}[t_1, t_1^{-1}, t_2, t_2^{-1}, \dots, t_k, t_k^{-1}]$ rather than $\mathbb{Z}[t, t^{-1}]$, where there is an indeterminate t_i for each of the k link components in L. The definition of the differential, which we will see in a while, is adjusted to incorporate these new indeterminates. Another main difference is that there is more than one way of assigning degrees to the generators, as will be described below.

1.5.1 Defining the Degree

First, for each link component L_i of L, let Y_i be its projection in Y. Pick a non-singular point β_i on each Y_i, which we call the *base point* of Y_i. Also, pick an integer $\rho_i \in \mathbb{Z}$ for each component Y_i, which we call the *grading coefficient* of Y_i. We will now show how to get an assignment of degrees for the generators of Y for each choice of $\{\beta_1, \dots, \beta_k\}$ and $\{\rho_1, \dots, \rho_k\}$.

Again, we define the capping paths, but this time with two paths for each generator a_i. If a_i is a crossing, let the upper knot segment belong to the link component L_u, and the lower knot segment to L_l. Note that it is possible that $L_u = L_l$. Let the capping path P_i^u begin at the base point β_u, follow L_u in the direction of its orientation, pass through the upper knot segment, and end at a_i. Similarly, let P_i^l begin at β_l, follow L_l in the given orientation, pass through

the lower knot segment, and end at a_i. If a_i is a right cusp, then P_i^u and P_i^l are both the path that begins with $\beta_u = \beta_l$, goes in the direction of $L_u = L_l$ and ends with a_i. We define the degree as

$$deg(a_i) = \begin{cases} 1 & \text{if } a_i \text{ is a right cusp} \\ m(P_i^u) - m(P_i^l) + 2\rho_l - 2\rho_u & \text{if } a_i \text{ is a crossing} \end{cases}$$

It turns out that if we arbitrarily fix the base points $\{\beta_1, \ldots, \beta_k\}$ and set $\rho_1 = 0$, we can still get every possible assignment of degrees by just varying the grading coefficients $\{\rho_2, \ldots, \rho_k\}$.

1.5.2 Defining the Differential

We maintain the same definition for an admissible path as with the previous section. Here, the path is allowed to traverse more than one component in the front Y. Again, we associate a monomial with each admissible path P. For each link component L_i, let $n_i(P)$ be the signed number of times that $-P$ passes the base point β_i in the direction of the orientation of L_i. As before, let $\wedge(P)$ be $+1$ or -1 depending on the whether the number of downward corner generators of even degree on P is even or odd. Then, the monomial for P is

$$\alpha(P) = \wedge(P) t_1^{n_1(P)} \ldots t_k^{n_k(P)} a_{j_1} \ldots a_{j_m}$$

The differential is now given by

$$\partial a_i = \begin{cases} \sum \alpha(P) & \text{if } a_i \text{ is a crossing} \\ 1 + \sum \alpha(P) & \text{if } a_i \text{ is a right cusp} \end{cases}$$

Again, we let $\partial(\mathbb{Z}[t_1, t_1^{-1}, \ldots, t_k, t_k^{-1}]) = 0$, and extend the differential to the rest of the algebra by linearity and the signed Leibniz rule.

1.6 Examples and Simple Fronts

The following example of the DGA for a figure eight knot is extracted here from [Ng1] for completeness. Figure 9 shows the front with its generators labelled. The degrees of its generators are all 1, except for a_5 and a_6 which are of degree 0. The differentials of the generators are

$$\begin{aligned} \partial a_1 = \;& 1 + a_6 - t^2 a_6 a_4 a_6 a_7 - t^2(1 - t a_6 a_5) a_3 a_6 a_7 \\ & + t a_6 a_2 (1 - t a_6 - t^2 a_7 a_4 a_6) a_7 \\ \partial a_2 = \;& 1 - t a_5 a_6 \\ \partial a_3 = \;& t^{-1} - a_6 - t a_6 a_7 a_4 \\ \partial a_4 = \;& \partial a_5 = \partial a_6 = \partial a_7 = 0 \end{aligned}$$

Examples of how the above monomials are computed can be found in [Ng1]. As Ng points out, computing the differentials can be an extremely tedious task because one has to ensure that no admissible path was left out.

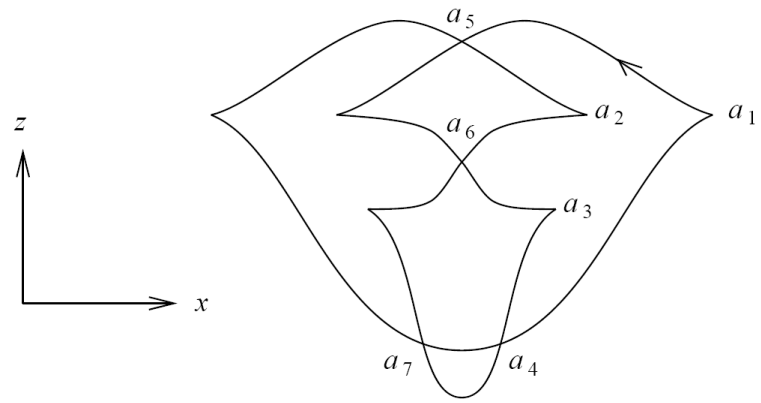

Figure 9: [Ng1] Diagram of a figure eight knot, with its generators labelled.

Fortunately, there exists a class of fronts, called *simple* fronts, for which the differentials can be computed with ease. Such fronts are diffeomorphic to a front where all the right cusps have the same x coordinate. Loosely speaking, the right cusps of simple fronts are all "exposed" rather than "buried" within the front itself. It is not difficult to see that an admissible path for a generator a_i must traverse leftwards, hit a left cusp, and traverse rightwards back to a_i. While it is traversing leftwards or rightwards, it can only change directions at the crossings. The path cannot turn leftwards again at a right cusp after traversing rightwards, because all right cusps in a simple front are initial generators and an admissible path can only have one initial generator.

All fronts can be made simple by using the Legendrian Reidemeister moves. We only need to push all the right cusps rightwards until they are exposed. Figure 10 shows the original figure eight knot front made simple. The shaded regions illustrates the admissible maps and paths corresponding to the monomials $ta_{10}a_5$ in ∂a_1 and $-ta_{10}a_7$ in ∂a_6. The DGA for this front is

$$\begin{aligned}
\partial a_1 &= 1 + a_6 + ta_{10}a_5 \\
\partial a_2 &= 1 - ta_9a_{10} \\
\partial a_3 &= t^{-1} - a_{10} - ta_{10}a_{11}a_8 \\
\partial a_4 &= t^{-1} + a_8a_7 - a_9a_6 - ta_9a_{10}a_5 \\
\partial a_5 &= a_7 + a_{11} + ta_{11}a_8a_7 \\
\partial a_6 &= -ta_{10}a_7 - ta_{10}a_{11} - t^2a_{10}a_{11}a_8a_7 \\
\partial a_7 &= \partial a_8 = \partial a_9 = \partial a_{10} = \partial a_{11} = 0
\end{aligned}$$

The degrees are as follow: a_1, a_2, a_3, a_4, and a_8 are of degree 1; a_7 and a_{11} are of degree -1, and all the rest of the generators are of degree 0.

For an example of the computation of the DGA for a front of a link, the reader may refer to [Ng1].

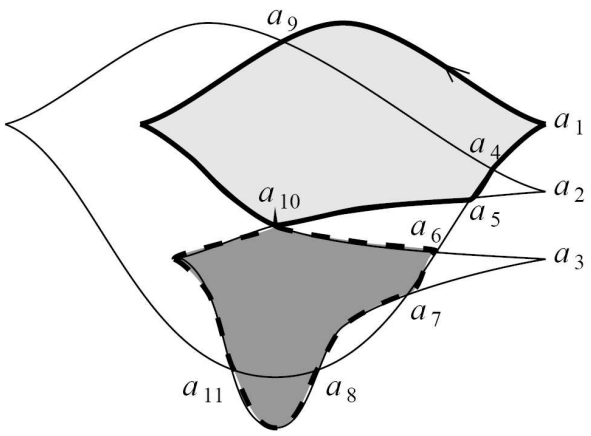

Figure 10: Original figure eight knot made simple.

1.7 Invariance of DGAs

The following two theorems, proved in [Ch2], [ENS] and [Ng1], tell us that the DGAs of Legendrian isotopic knots are not too different from one another. They use the notion of stable tame isomorphism which we will not define in this paper.

Theorem 1.1. *The DGAs of the fronts of Legendrian isotopic knots are stable tame isomorphic.*

Theorem 1.2. *Given the fronts Y_1 and Y_2 of two Legendrian isotopic links, there exists a choice of grading on the generators of Y_1 and Y_2 such that their associated DGA are stable tame isomorphic.*

One can show that the differential ∂ satisfies two important properties [Ch2]. Firstly, it lowers the degree of the element it is acting on by 1. Secondly, $\partial^2 = 0$. The second property implies that $\operatorname{im}\partial \subseteq \ker\partial$. This allows us to study the graded homology ring $H = \ker\partial/\operatorname{im}\partial$. The above two theorems give us the following corollary, also from [Ch2], [ENS] and [Ng1].

Corollary 1.3. *The graded homology of the DGA for a front of a knot is invariant under Legendrian isotopy. For links, the graded homology is invariant up to a choice of grading.*

A few clarifications are needed at this point. Firstly, the choice of grading mentioned in Theorem 1.2 and Corollary 1.3 refers to the choice of grading coefficients $\{\rho_2, \ldots, \rho_k\}$ described in Section 1.5.1. The choice of base points $\{\beta_1, \ldots, \beta_k\}$ is unimportant, since all possible gradings can be achieved by varying the ρ_i's. Secondly, the DGAs derived from different choices of base points in Section 1.5.2 may differ in terms of the powers of the t_i's, but they will all be stable tame isomorphic to one another. Thirdly, Sections 1.4 and 1.5 may

give slightly different constructions for the DGA of the front of a single knot, but the resulting algebras will still be stable tame isomorphic to each other.

The invariance of link DGAs up to stable tame isomorphism opens up a highway in our quest to classify Legendrian knots up to Legendrian isotopy. However, recognizing that two DGAs are isomorphic or that two homology rings are the same is a tough problem by itself. We need tools to help us differentiate them. Ironically, we need to find *invariants* of these invariant DGAs and homology rings.

In [Ch2], Chekanov describes how to create a set of Poincaré polynomials $I(A, \partial)$ for a given knot DGA (A, ∂), and showed that it remains invariant over stable tame isomorphism. Ng extended this construction to create a set of *families* of Poincare polynomials for a link DGA. We shall refer to these polynomials as the *Poincaré-Chekanov polynomials* of a knot and of a link respectively.

1.8 Poincaré-Chekanov polynomials of a Knot

First, given a DGA over $\mathbb{Z}[t, t^{-1}]$, we shall reduce it to a DGA (A, ∂) over $\mathbb{Z}/2$ by mapping t and t^{-1} to 1 and reducing the algebra modulo 2. For any $a \in A$, let $\partial_0 a$ be the constant term in ∂a, i.e. $\partial_0 x \in \mathbb{Z}/2$. Similarly, let $\partial_1 a$ be the linear terms in ∂a, i.e. $\partial_1 a \in A_1$ where A_1 is the vector space generated by $\{a_1, \ldots, a_n\}$ over $\mathbb{Z}/2$.

Next, we give the definition of an augmentation for the new DGA (A, ∂). Let $\varepsilon : \{a_1, \ldots, a_n\} \to \mathbb{Z}/2$ be a function on the generators of A. Define $g : A \to A$ to be the automorphism where $g(a_i) = a_i + \varepsilon(a_i)$ for all generators a_i. We shall demand that g be graded, so we must have $\varepsilon(a_i) = 0$ if $deg(a_i) \neq 0$. Then, we will let the image of the original DGA (A, ∂) under this automorphism be (A, ∂^g) where $\partial^g = g^{-1} \circ \partial \circ g$. We say that ε is an augmentation for (A, ∂) if $(\partial^g)_0 = 0$. Note that $\partial_0^g a$ can be calculated by substituting a_i with $\varepsilon(a_i)$ in ∂a. Thus, ε can be seen as a map from $\{a_1, \ldots, a_n\}$ to the roots of the equations $\partial a_1 = 0, \ldots, \partial a_n = 0$ over $\mathbb{Z}/2$.

Now, let ε be an augmentation for (A, ∂), and g be the corresponding automorphism. Since $\partial_0^g = 0$, we have $(\partial_1^g)^2 = 0$ so we can define the homology $H(A_1, \partial_1^g) = \ker \partial_1^g / \operatorname{im} \partial_1^g$. Because ∂_1^g is linear, this homology is a vector space. Furthermore, the homology breaks into degree homogeneous components. By computing the dimension of each of these components, we can define the Poincaré-Chekanov polynomial

$$P_{(A,\partial^g)}(t) = \sum_{\lambda} \dim(H_\lambda(A_1, \partial_1^g))t^\lambda$$

where $H_\lambda(A_1, \partial_1^g)$ is the degree λ homoegenous component of $H(A_1, \partial_1^g)$, and the summation is taken over all possible degrees of the components.

The Poincaré-Chekanov polynomials $I(A, \partial)$ is the set of all the polynomials $P_{(A,\partial^g)}(t)$ as g varies over the augmentations ε of (A, ∂). The number of augmentations is finite, since there are only a finite number of mappings $\varepsilon : \{a_1, \ldots, a_n\} \to \mathbb{Z}/2$. Hence, $I(A, \partial)$ is a finite set. Frequently, $I(A, \partial)$

contains only one element, but examples where it contains more than one element have been found [MS]. The reader may be interested to know that for the figure eight knot examples in Figure 9 and 10, the set of Poincaré-Chekanov polynomials in both cases is $\{t^{-1} + 2t\}$.

1.9 Poincaré-Chekanov polynomials of a Link

Given a link DGA, one can also use the exact same procedure in the previous section to compute its Poincaré-Chekanov polynomials.However, link DGAs have additional algebraic structure that allows us to compute a slightly different invariant. In [Ng1], a *differential link module* related to the link DGA is defined. This link module gives rise to a family of Poincaré-Chekanov polynomials.

To define this family of polynomials, we consider an augmentation ε which satisfies $\varepsilon(a_i) = 0$ on crossings a_i between different link components of the link L. Let g be the automorphism associated with this augmentation. We say that a crossing is a (j_1, j_2)-generator if its bottom segment belongs to the component L_{j_1} of L and its top segment belongs to the component L_{j_2}. We say that a right cusp is a (j, j)-generator if it belongs to the component L_j. Now let $V_{j_1 j_2}$ be the graded vector space generated by all (j_1, j_2)-generators over $\mathbb{Z}/2$. It can be shown that ∂_1^g preserves each $V_{j_1 j_2}$. Once again, we can compute the rank of each graded component of the corresponding homology to get a Poincaré-Chekanov polynomial $P_{j_1 j_2}$. Note that if we consider the component L_j as a knot by itself, then V_{jj} is the vector space spanned by its generators. Thus, the polynomial P_{jj} is just the Poincaré polynomial for the knot corresponding to the given augmentation ε.

We have the following invariance theorems for the Poincaré polynomials, which are proven in [Ch2] and [Ng1].

Theorem 1.4. *Let L_1 and L_2 be two Legendrian isotopic knots. Let (A_1, ∂_1) and (A_2, ∂_2) be their respective DGAs. Then, $I(A_1, \partial_1) = I(A_2, \partial_2)$.*

Theorem 1.5. *Let L_1 and L_2 be two Legendrian isotopic links. Then, for any given grading and augmentation of the DGA for L_1, there is a grading and augmentation of the DGA for L_2 such that the Poincaré polynomials $P_{j_1 j_2}$ for L_1 and L_2 are equal for all j_1, j_2.*

The polynomials studied in the previous two sections are referred to as first-order Poincaré polynomials, since we looked at the linear part ∂_1^g of the differential ∂^g. These first-order Poincaré polynomials are not the only computable invariants for Legendrian knots. In a similar way, one can compute higher-order Poincaré polynomials, or study the reduction of the DGA over other fields besides $\mathbb{Z}/2$. Other invariants include characteristic algebras [Ng1], and the number of admissible decompositions [Ch3].

Driving this search for new Legendrian invariants is the desire to completely classify knots up to Legendrian isotopy. When a new invariant is found, two questions are frequently asked. Firstly, does this invariant encode information about previously known invariants? Secondly, does the invariant allow us to

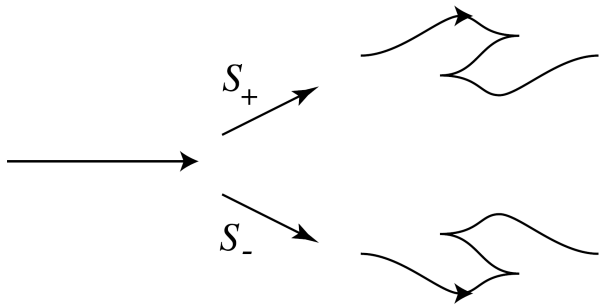

Figure 11: [Et1] Stabilizations of a Legendrian knot.

classify knots of certain topological types? These are often difficult problems by themselves. Legendrian invariants can also be used in helping us to understand other types of knots, such as transversal knots.

1.10 Transversal Knots and N-Copies

Recall that in terms of the language of contact geometry, a knot is Legendrian if it is everywhere tangent to the contact structure ξ. A *transverse* knot is one that is everywhere transverse to ξ. We say that two transverse knots are transversely isotopic if there is a smooth isotopy between them through transverse knots. [Et1] gives an excellent introduction to the basic properties of transverse knots.

It turns out that traverse knots are intimately related to Legendrian knots. We begin by defining the *stabilization* a Legendrian knot L. Take the front of L remove a small segment of it, and replace it with one of the two zigzags shown in Figure 11. If the zigzag is downward oriented, we call this surgical process a positive stabilization and denote the new knot by $S_+(L)$. If the zigzag is upward oriented, we say that it is a negative stabilization and the new knot is denoted by $S_-(L)$. The two processes are not inverses of each other, nor do they commute with each other. Although the new knots $S_+(L)$ and $S_-(L)$ are topologicially isotopic to the original knot L, they are not Legendrian isotopic to L because they have different Thurston-Bennequin and rotation numbers. Interestingly, Fuchs and Tabachnikov showed that given two topologically isotopic knots, one can apply a sequence of positive and negative stabilizations to each knot so that the resulting knots are Legendrian isotopic [FT].

One can get a transverse knot from a Legendrian knot L by displacing it slightly. To be more precise, let $A = S^1 \times [-1, 1]$ be a thin annulus embedded in $\mathbb{R}^3$ such that $S^1 \times \{0\}$ is the knot L, and A is transverse to the contact structure ξ. Then, the transverse knots $S^1 \times \{+\frac{1}{2}\}$ and $S^1 \times \{-\frac{1}{2}\}$ are the transverse *push-offs* of L. This leads us to the following important theorem relating Legendrian and transverse knots [EFM].

Theorem 1.6. *Two Legendrian knots are negatively stably isotopic if and only if their transverse push-offs are tranversely isotopic.*

Here, we say that two Legendrian knots are negatively stably isotopic if they become Legendrian isotopic after a number of negative stabilizations to each of them. The theorem implies that invariants of Legendrian knots which are unaffected by negative stabilization will also be invariants of transverse knots. This implication is an important one since very few transverse invariants are known today.

One place to start in searching for these transverse invariants is the DGA of a Legendrian knot. Unfortunately, the DGA of a knot vanishes when the knot is stabilized. To avoid this problem, one can look at the *N-copy* of a stabilized knot, where N is a positive integer. The N-copy of a knot can be constructed by taking the front of a knot, making N copies of it, and displacing each copy slightly in the z-direction to make a link. The DGA of the N-copy of a stabilized knot does not vanish. This gives us hope of finding some property of the DGA that survives negative stabilizations, so we can use this property to define a new transverse invariant.

2 FrontLeg: Design and Application

The computation of the DGA and the Poincare polynomials of a link can be a tedious task. FrontLeg is a Windows C++ program designed to help with this problem. The program is so named because it is a pun on the words "*Front* Projection of *Leg*endrian Knots". FrontLeg is still in its initial developmental stages but hopefully, it will be powerful enough for most purposes.

2.1 Program Design

The main data structure used to store the front Y of an oriented link L is a directed graph $G = (V, E)$. Here, there is a one to one correspondence between the elements of the vertex set V and the singularities of Y. We can write $V = L \cup R \cup C$ where L, R and C represent the sets of left cusps, right cusps and crossings respectively. The edge set E represents all the directed knot segments that connect the singularities to one another. Each vertex representing a right cusp or a left cusp has 2 edges incident to it, while each vertex representing a crossing has 4 edges incident to it. Note that in particular, G must be a planar graph. Figure 12 shows the graph for the front of the figure eight knot in Figure 9. From now, we shall use the terms *left cusp*, *right cusp* and *crossing* to refer to the vertices that represent them in G.

The table below shows how information about the vertices and edges are stored in the program. This information is sufficient for us to calculate the DGA and Poincaré polynomials of the knot. Here, $I(v)$ is the 2-tuple or 4-tuple representing the edges incident to the vertex v of the graph. For left and right cusps, the first edge of the 2-tuple is the top edge, and the second edge is the bottom one. For crossings, the first edge of the 4-tuple is the edge that is in

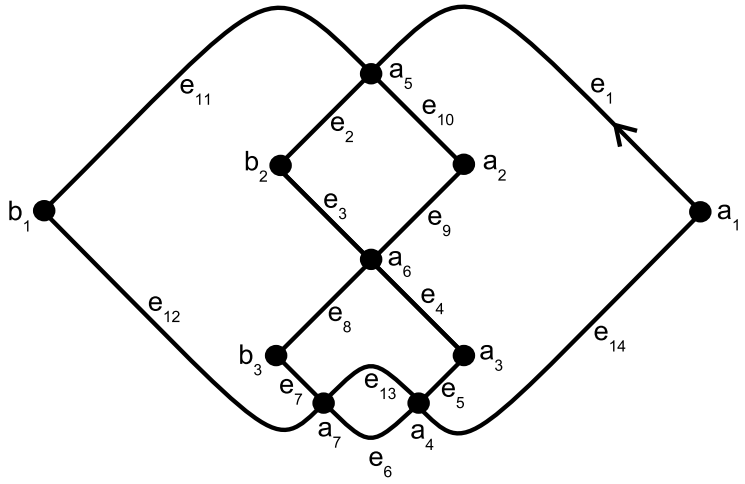

Figure 12: Graph representing the figure eight knot.

the top-right neighborhood of the vertex. The rest of the edges are listed in counter-clockwise order. $I(e)$ is the 2-tuple representing the vertices incident to the edge e of the the graph. Every edge is oriented, and points in the direction of the first vertex of the 2-tuple to the second vertex.

$G = (V, E)$
$V = \{a_1, \ldots, a_7, b_1, b_2, b_3\}$
$E = \{e_1, e_2, \ldots, e_{14}\}$

$I(a_1) = (e_1, e_{14})$
$I(a_2) = (e_{10}, e_9)$
$I(a_3) = (e_4, e_5)$
$I(a_4) = (e_5, e_{13}, e_6, e_{14})$
$I(a_5) = (e_1, e_{11}, e_2, e_{10})$
$I(a_6) = (e_9, e_3, e_8, e_4)$
$I(a_7) = (e_{13}, e_7, e_{12}, e_6)$
$I(b_1) = (e_{11}, e_{12})$
$I(b_2) = (e_2, e_3)$
$I(b_3) = (e_8, e_7)$

$I(e_1) = (a_1, a_5)$
$I(e_2) = (a_5, b_2)$
$I(e_3) = (b_2, a_6)$
$I(e_4) = (a_6, a_3)$
$I(e_5) = (a_3, a_4)$
$I(e_6) = (a_4, a_5)$
$I(e_7) = (a_7, b_3)$

$I(e_8) = (b_3, a_6)$
$I(e_9) = (a_6, a_2)$
$I(e_{10}) = (a_2, a_5)$
$I(e_{11}) = (a_5, b_1)$
$I(e_{12}) = (b_1, a_7)$
$I(e_{13}) = (a_7, a_4)$
$I(e_{14}) = (a_4, a_1)$

2.1.1 Algorithmic Difficulties

Most of the algorithms used in the computation of the Legendrian invariants were easily programmed. However, some interesting difficulties were encountered, and I shall highlight some of them here.

The first difficulty was in computing the DGA of non-simple fronts. The main problem lies in checking if a path P is the boundary of an immersed disk.

To overcome this problem, an algorithm was written to make a simple front from a non-simple one. Then, the DGA of this simple front can be computed easily. This DGA is stable tame isomorphic to the original one. It is still not clear to the author how a program can be written to compute the DGA of a non-simple front directly.

The second difficulty was in checking if a graph represents a simple front. In particular, we want to find out if a given right cusp is "exposed" or "buried"(see Section 1.6)? One solution is this: imagine that you are walking on the edges of the graph, start by traversing along the top edge of the right cusp, and trace a path by always taking a "right turn" at crossings. This path should return to the original right cusp. If this closed path goes clockwise, then the right cusp is buried. If the path goes counter-clockwise, then it is exposed.

This method can also be used to tell us how deep a right cusp is buried, so that we can find the fastest way of pushing it out so as to expose it. First, note that the planar graph divides the plane into regions separated by the edges. Identify the out-most region by the method described in the previous paragraph: tracing its boundary and checking that the trace goes counter-clockwise. We say that this region is of *depth* 0. Now, identify the regions that share an edge with this out-most regions, and define them to be of depth 1. Regions of depth 2 are those currently without a depth value but sharing an edge with regions of depth 1. We continue recursively in a similar fashion to define a depth value for all the other regions. We then expose a right cusp by repeatedly pushing it out into a region of smaller depth. This procedure allows us to create a simple front from any given front.

The third difficulty was in speeding up the search for the augmentations of a DGA. In fact, it is the slowest step in the computation of the Poincaré Chekanov polynomials. This search is slow because the number of possible augmentations grows exponentially with the number of generators. Since fronts of links usually have a lot more generators than those of knots, this can potentially be a serious issue. Fortunately, the augmentations for the fronts of links are defined to be zero at crossings between different components, so the search reduces to finding augmentations for each link component, which is much faster.

Another trick used to speed up this search is to employ forward checking and the most-constrained-variable heuristic. Here is a set-up for the search problem: we are given a system of polynomial equations in the variables $x_1, \ldots, x_n$, and we wish to find solutions of the system in $\mathbb{Z}/2$. The algorithm is as follows: recursively, at each stage, we choose the variable that appears most in the system of equations, and assign to it a value of either 0 or 1. We substitute its value into the system and simplify the system. If there is a contradiction in the system (e.g. $0 = 1$ appears as one of the equations), then the system has no solution for the current assignment of variables and we backtrack. If there is no contradiction, we continue with the next variable assignment.

2.2 Program Structure

There are two main parts to the code of the program. The first is the module for the graphical user interface. This is the main program in FrontLeg. The second is the library for doing computations on the knot. We have a core header file "knot.h" that contains most of the key algorithms. This library is almost stand-alone, except that some lines of code were added to allow it to integrate with the user interface module. Also, some other libraries were written to help speed up the search for augmentations. I hope to integrate all of these external code into the core library someday, so that the library can be completely stand-alone and used for other research purposes.

The flow of the program in computing the Poincaré polynomials can be categorized into distinct steps:

1. Read data file.
2. Make front simple.
3. Compute DGA from simple front.
4. Find augmentations of DGA.
5. Compute Poincaré polynomials from augmentation.

For the reader who is interested in deciphering and editing the source code, it is probably best to understand the code in terms of these basic steps.

FrontLeg accepts files of two different file formats. First, it is able to read SnapPea files [SP]. SnapPea is a freeware program commonly used to study topological knots. SnapPea enthusiasts will be glad to know that their favorite knots import painlessly into FrontLeg. Second, FrontLeg reads FrontLeg files where the knot is stored in terms of its graph notation. This file format will be convenient for researchers who want to generate knot input files for FrontLeg automatically with a computer. Note that currently, FrontLeg is unable to display the images of knots stored using the FrontLeg file format.

2.3 Program Usage

Here are some simple steps to follow for drawing and computing the Legendrian invariants of a link using SnapPea and FrontLeg.

1. **Draw the knot with SnapPea**. FrontLeg converts intersections between line segments into crossings. It turns vertices whose adjacent segments lie to the right of the vertex into left cusps. It identifies vertices whose adjacent segments lie to the left of the vertex as right cusps. No line segment should be vertical. Be careful in drawing the line segments so as not to introduce undesired crossings or cusps. Save the figure as a text file. Actually, FrontLeg does not require the extension of the file to be '.txt' but it is a good practice to always save the file as so.

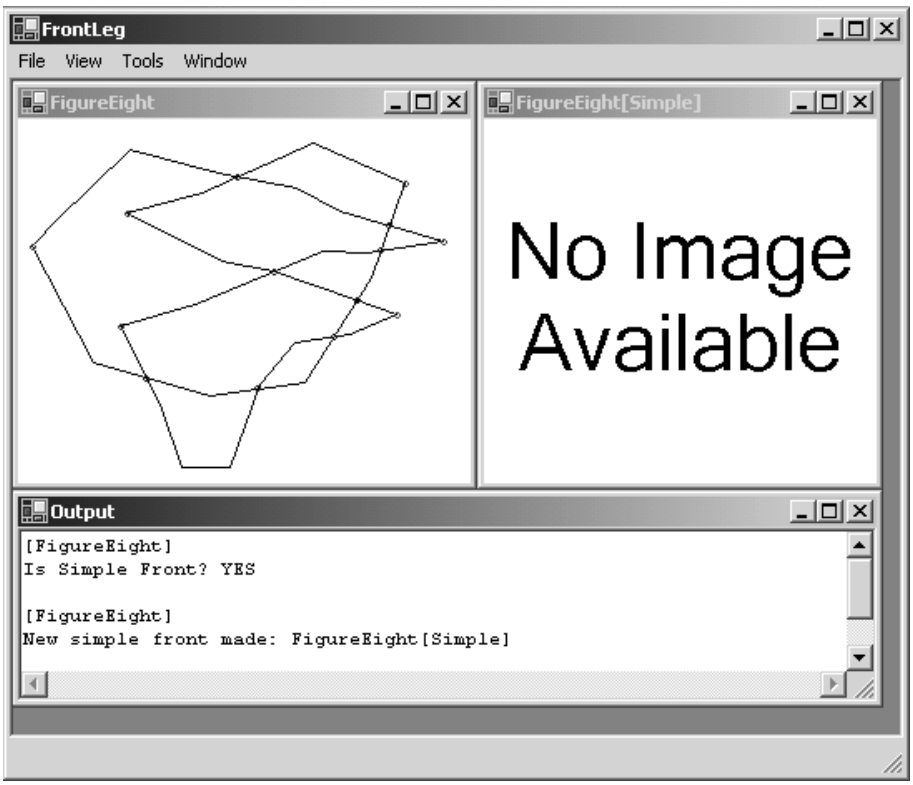

Figure 13: The FrontLeg user interface.

2. **Open the file with FrontLeg**. Select File|Open from the menu bar.

3. **Make a simple front.** You can check if the front is already simple using View|Is Simple Front?. An answer should appear in the Output window. If the knot is not simple, use Tools|Make Simple Front to create a new simple knot. A new window with the words "No Image Available" should appear. This window represents the new knot that was made. Figure 13 shows what you might expect to see on your screen. In general, remember to click on the title bar of the knot to select the knot before choosing an operation from the menu bar.

4. **Calculate the Poincare polynomials.** The menu option can be found under Tools|Poincare Polynomials. FrontLeg prints all the Poincaré polynomials which can be generated by some augmentation, and counts the number of augmentations that produce that polynomial. If the link is a single knot, FrontLeg prints a single polynomial for each augmentation. If the link has more than one component, the program prints the family of polynomials as described in Section 1.9. Figure 14 shows the results of the above computations.

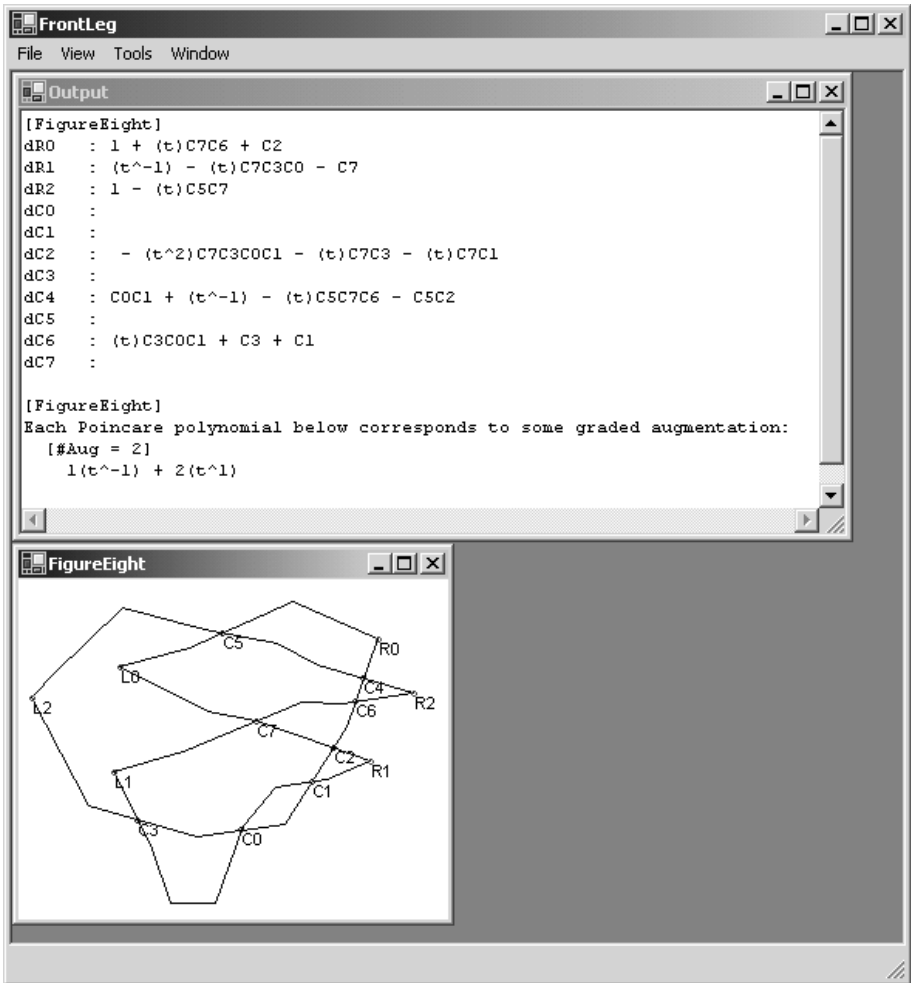

Figure 14: Using FrontLeg to compute the DGA and Poincaré polynomials.

5. **Save the new knot (optional).** To save the newly created simple front in FrontLeg file format, go to File|Save.

Some of the other functions that FrontLeg offers are as follows:

- **Calculating the classical invariants.** FrontLeg computes the Thurston-Bennequin, Maslov and rotation numbers of each component of the link. See the options Thurston- Bennequin, Maslov Number and Rotation Number under View.
- **Calculating the grading.** FrontLeg calculates the degree of each generator in the link. Use the option View|Vertex Degree.
- **Graph notation.** FrontLeg displays the vertices and edges of the graph that represents the knot. It counts the number of cusps, crossings, components and generators. See the option View|Knot Data.
- **Making an N-copy.** FrontLeg makes an N-copy of the selected knot. Choose View|Make N-Copy. The program asks how many copies you would like to make, and creates a new window representing the new knot. Currently, FrontLeg does not display an image of the new N-copy that is made, so the words "No Image Available" will be seen instead.
- **Differential algebra.** FrontLeg computes the DGA of the selected link. See Tools|Differential Algebra.
- **Chekanov Invariants.** FrontLeg computes the Chekanov invariant I, which is discussed in the preprint [Ch1].

The graphical interface of FrontLeg also allows you to label the edges and vertices of the knot. The menu option View|Show\Hide Labels allows you to toggle the showing of the labels on the image of the knot. If the labels are too close together to be read, you can zoom into the image with View|Zoom. Lastly, if the Output window becomes too cluttered, you can click on Window|Clear Output to remove all the text.

3 Suggested Extensions and Research

Software development is a never-ending process. While FrontLeg has the basic features needed for research requiring computation of Chekanov-Eliashberg DGAs and Poincaré polynomials, many other functionalities could be added to make it more useful. Below are some suggested improvements. Of those listed, the last two are probably the most challenging ones.

1. Allow drawing and editing of fronts.
2. Draw smooth image of knot for display in papers.
3. Compute higher-order Poincare polynomials.

4. Find augmentations of DGA over fields other than $\mathbb{Z}/2$.

5. Draw image of N-copies.

6. Draw image of knot given graph notation.

7. Compute DGA of non-simple fronts.

We also realize that knowledge about Legendrian knots is increasing at a dramatic rate, and this program is probably insufficient for most calculational purposes. People are interested in a well-documented library that can be extended and integrated easily into their own programs. Producing such a library is one of the areas that I hope to work on as well.

Most importantly, the completion of a working version of FrontLeg marks the beginning of experimentation to test old conjectures and form new hypotheses. One of the main reasons for the creation of FrontLeg was to study the relationship between the Poincaré-Chekanov polynomials and negative stabilizations. The hope was to find an invariant of Legendrian knots, or their N-copies, that survive this type of stabilization so that it would also be an invariant of transversal knots by Theorem 1.6. This is just one of the many unsolved questions out there in the field of Legendrian knots. Hopefully, FrontLeg will free people from the tedious computations so that they can focus on answering these questions.

4 Acknowledgements

First and foremost, I would like to thank my advisor, Professor Yasha Eliashberg, for his infectious enthusiasm and endless energy in providing brilliant ideas, patient guidance and timely encouragement. I would also like to thank my "brudders", Joel Goh and Sean Wat, fellow Singaporean students here at Stanford, for their support, prayers and jokes that carried me through my high and low points. Last but not least, I thank the Lord for His anointing on this project and His strength through me. I thank Him also for teaching me to dream, and for continually revealing His beauty to me through Mathematics!

References

[Ch1] Yu. Chekanov *Differential Algebra of Legendrian Links.* preprint (1997); revised version (1999).

[Ch2] Yu. Chekanov *Differential Algebra of Legendrian Links.* Invent. Math. **150** (2002), no. 3, 441-483.

[Ch3] Yu. Chekanov *Invariants of Legendrian Knots.* Proceedings of the ICM, Beijing 2002, vol. 2, 385-394.

[EF] Ya. Eliashberg, M. Fraser *Classification of Topologically Trivial Legendrian Knots.* 'Geometry, Topology and Dynamics' (Montreal, PQ, 1995), CRM Proc. Lecture Notes **15** (1998), 17-51.

[EFM] J. Epstein, D. Fuchs, M. Meyer *Chekanov-Eliashberg Invariants and Transverse Approximations of Legendrian Knots.* Pacific J. Math. **201** (2001), no. 1, 89-106.

[EH] J. Etnyre, K. Honda *Knots and Contact Geometry.* arXiv:math.GT/0006112.

[ENS] J. Etnyre, L. Ng, and J. Sabloff *Invariants of Legendrian Knots and Coherent Orientations.* J. Symplectic Geom. **1** (2002), no. 2, 321-367.

[Et1] J. Etnyre *Legendrian and Transversal Knots.* (2003), math.SG/0306256.

[FT] D. Fuchs, S. Tabachnikov *Invariants of Legendrian and Transverse Knots in the Standard Contact Space.* Topology **36** (1997), no. 5, 1025-1053.

[MS] P. Melvin, S. Shrestha *The Nonuniqueness of Chekanov Polynomials of Legendrian Knots.* arXiv:math.GT/0411206.

[Ng1] L. Ng *Computable Legendrian Invariants.* Topology **42** (2003), no. 1, 55-82.

[SP] http://geometrygames.org/SnapPea/.

[Sw] J. Swiatkowski *On the isotopy of Legendrian knots.* Ann. Global Anal. Geom. **10** (1992), 195-207.