# Magma Symbolic Integration Report

Mitchell Holt

February 2024

## Contents

## 1 Introduction and Background

This report describes an incomplete implementation of the Risch procedure for symbolic integration, as given in [1]. Rational integration (see §4) has been implemented completely. Logarithmic integration (see §5) is implemented, but not correct. The tests contain some failing cases. Although exponential integration is partially described in Geddes [1], it has not been implemented. The integration of algebraic functions or other transcendentals (such as error functions) was beyond the scope of the project.

A significant design decision in the code is to allow the *Risch Structure Theorem* [2] to be used to verify that each new logarithmic or exponential elementary field extension is transcendental, as is required. The *Risch Structure Theorem* has not been implemented, but if one could be created, then it would be trivial to add it to the existing code.

## 2 Differential Fields

In this section we describe the intrinsics in `DifferentialFields.m`, which are motivated by *Definition 1*.

**Definition 1** (Elementary Function Field)**.** Let $K = \mathbb{Q}(\alpha_1, \ldots, \alpha_m)$ be a constant differential field (with differential operator $'$) where each $\alpha_i$ is algebraic over $\mathbb{Q}$. Let $x$ be a transcendental symbol over $K$ satisfying $x' = 1$ (in the differential field $(K(x), ')$). In the differential extension field $K(x, \theta_1, \ldots, \theta_n)$, we say each $\theta_j$ $(j = 1, \ldots, n)$ is *elementary* over $K(x, \theta_1, \ldots, \theta_{j-1})$ if $\theta_j$ is *transcendental* over $K(x, \theta_1, \ldots, \theta_{j-1})$ and either:

(i) $\theta_j' = u'/u$ for some $u \in K(x, \theta_1, \ldots, \theta_{j-1})$ (in which case we say $\theta_j$ is *logarithmic*), or

(ii) $\theta_j'/\theta_j = u'$ for some $u \in K(x, \theta_1, \ldots, \theta_{j-1})$ (in which case we say $\theta_j$ is *exponential*).

If each $\theta_j$ is elementary, then we say that $K(x, \theta_1, \ldots, \theta_{j-1})$ is an *elementary function field*.

Ideally, we would represent the elementary function field $K(x, \theta_1, \ldots, \theta_m)$ in Magma as a `RngDiff` whose constant field is either the rational field or an absolute number field, and whose (ordered) generators are $x$, $\theta_1$, $\ldots$, $\theta_m$. However, there are elementary function fields which cannot be constructed with a single call to the existing `DifferentialFieldExtension` intrinsic because it requires the universe of the input sequence to be a multivariate polynomial ring (in our case over $K(x)$), rather than the field of fractions. For example, consider the elementary function field $\mathbb{Q}(x, \log x, \log(\log x))$, where $\log(\log x)$ has derivative $\frac{1}{x \log x} \notin \mathbb{Q}(x)[\log x, \log(\log x)]$.

Therefore, we represent elementary function fields either as $K(x)$, or as $F(\theta)$, where $F$ is an elementary function field and $\theta$ is elementary over $F$. In the case that $F$ is not $K(x)$, using `DifferentialFieldExtension` does not set the generators of the field it returns correctly. Nils Bruin provides a fix for this bug in the case of a logarithmic extension:

```
fld := LogarithmicFieldExtension(F, f); // incorrect generators
fld`Generators := [ fld | c : c in OrderedGenerators(
    UnderlyingField(fld)) ];
```

This does not work for exponential extensions. The intrinsic `IsLogarithmic` assumes that the construction of exponential extensions works as intended (although it currently does not, see §2.1).

## 2.1 Documentation

`IsPolyFractionField(F): RngDiff -> BoolElt`

Checks if `F` is of the form $K(x)$ for some constant field $K$.

`AsFraction(f): RngDiffElt -> FldFunRatElt`

Return the representation of `f` as a fraction inside the underlying field of `Parent(f)`.

`IsPolynomial(f): RngDiffElt -> BoolElt, RngUPolElt`

Let `Parent(f)` be either $K(x)$ for some constant field $K$ or $F(\theta)$ for some elementary function field $F$ with $\theta$ elementary over $F$. Check if `f` is in $K[x]$ or $F[\theta]$ respectively, and return its representation as a univariate polynomial if it is.

`ExtendConstantField(F, C): RngDiff, Fld -> RngDiff`

This is the same as `ConstantFieldExtension`, but works for any elementary function field (not just $K(x)$, which satisfies `IsAlgebraicDifferentialField`).

`IsLogarithmic(F): RngDiff -> BoolElt`

Check if `F` is of the form $F(L)$ for some elementary function field $F$ and elementary function $L$.

`AllLogarithms(F): RngDiff -> SeqEnum`

Give an ordered sequence of all the logarithmic generators of `F`, with the last logarithm in the extension tower being the last to appear in the sequence.

`IsTranscendentalLogarithm(new, logarithms): RngDiffElt, SeqEnum -> SeqEnum`

Let `logarithms` be the logarithmic generators of an elementary function field $F$ and `new` be of the form $\frac{u'}{u}$ for some $u \in F$. Use the *Risch Structure Theorem* to check if $\log u$ is transcendental over $F$. If it is, return the empty sequence. If not, return a sequence $S$ of `< factor, non-zero power >` pairs such that $u = \prod_{(g,k)\in S} g^k$.

`LogarithmicExtension(F, f): RngDiff, RngDiffElt -> RngDiff, SeqEnum, RngDiffElt`

Named parameters:

`logarithms : SeqEnum` with default value `[]`.

Return the differential extension field $F(L)$, where $L$ is logarithmic over $F$ with derivative $f$, noting that it may be the case the $F(L) = F$. Also return all the logarithms of $F(L)$ and the representation of $L$ in $F(L)$. If `logarithms` is non-empty, it is assumed that these are all of logarithmic generators of $F$. Otherwise, the logarithms will be calculated.

`ExponentialExtension(F, f): RngDiff, RngDiffElt -> RngDiff, SeqEnum, RngDiffElt`

Named parameters:

`exponentials : SeqEnum` with default value `[]`.

Return the differential extension field $F(E)$, where $E$ is exponential over $F$ with derivative $fE$, noting that it may be the case the $F(E) = F$. Also return all the logarithms of $F(E)$ and the representation of $E$ in $F(E)$. If `exponentials` is non-empty, it is assumed that these are all the exponential generators of $F$. Otherwise, the exponentals will be calculated.

This function does not currently work, for example the derivative of $\exp x$ is not set correctly in the construction of $\mathbb{Q}(x, \log x, \exp x)$. It seems that the problem may be in the intrinsic DifferentialFieldExtension, which is called by ExponentialFieldExtension.

```
> F<x> := RationalDifferentialField(RationalField());
> G<g> := LogarithmicExtension(F, F!(1/x));
> E<e> := ExponentialExtension(G, G!1);
> Derivative(e);
g
> e eq g;
false
> Generators(E);
[ e ]
> CoefficientRing(E) eq G;
true
```

NameField(~F): RngDiff ->

Named parameters:

FirstExtName : MonStgElt with default value "x".

AlgNumName : MonStgElt with default value "a".

Using sensible defaults, name the transcendental generators of $F$ and the generator of its constant field $K$ (if any) for readable printing of elements of $F$. FirstExtName is the name to be assigned to the (unique) transcendental over the constant field which has derivative 1.

## 3 Integration

There is a single intrinsic inside Integration.m, which handles deciding how to integrate an elementary function and calls the relevant routine, depending on whether the input is a rational function, is logarithmic, or is exponential.

**Definition 2** (Integration)**.** Let $F$ be an elementary function field and $f \in F$. If there exists a finitely and explicitly generated elementary extension $G$ of $F$ and a $g \in G$ with $g' = f$, then we say $g$ is the *elementary antiderivative* or *elementary integral* of $f$ and write $\int f = g$. If no such $g$ or $G$ exist, we say that $f$ has no elementary integral.

## 3.1 Documentation

`ElementaryIntegral(f): RngDiffElt -> BoolElt, RngDiffElt, SeqEnum`

Named parameters:

`all_logarithms` with default value `[]`.

Check if the given elementary function has an elementary integral. If it does, return an integral and a list of all the logarithms needed to generate the smallest extension of the field that the input comes from.

Note that, by *Theorem 2*, the only elementary extensions required to express the integral of an elementary function are logarithmic, so we need only return all the logarithmic generators of the smallest function field (w.r.t. number of logarithmic generators) containing an integral.

# 4 Rational Integration

A *rational function* is an element of $K(x)$, where $K$ is some finitely and explicitly generated algebraic extension of $\mathbb{Q}$. Note that, for rational integration, an elementary integral always exists.

## 4.1 Documentation

`RothsteinTrager(num, denom): RngUPolElt, RngUPolElt -> SeqEnum`

Perform the Rothstein-Trager algorithm on the numerator, denominator input. Return a list of `<constant, logarithm argument>` pairs.

`RationalIntegral(f): RngDiffElt -> RngDiffElt, SeqEnum`

Integrate the given rational function using Hermite's method and the Rothstein-Trager method. Return the integral and all the logarithms generating the (smallest) elementary function field containing it. All algorithms used are as in Geddes [1].

## 5 Logarithmic Integration

Although all the necessary functions for logarithmic integration (that is, the integration of elements of $F(\theta)$ where $F$ is an elementary function field and $\theta$ elementary and logarithmic over $F$) are written, there are various bugs which have not been fixed.

### 5.1 Documentation

`LogarithmicRothsteinTrager(F, num, denom) : RngDiff, RngUPolElt, RngUPolElt -> BoolElt, SeqEnum`

Let $F = G(\theta)$ where $G$ is an elementary function field and $\theta$ is elementary and logarithmic over $G$. Let $\iota : G[\theta] \to G(\theta)$ be the inclusion map. Let `num` and `denom` be polynomials $f, g \in G[\theta]$. Use the logarithmic Rothstein-Trager theorem to find if $\iota(f)/\iota(g)$ has an elementary integral. If it does, return true and an elementary integral (represented as a sequence of `< coefficient, logarithm >` pairs). Otherwise, return false.

`IntegrateLogarithmicPolynomial(f): RngDiffElt -> BoolElt, RngDiffElt, SeqEnum`

Named parameters:

`all_logarithms` with default value `[]`.

Let $F(\theta)$ the parent of `f`, where $F$ is an elementary function field and *theta* is elementary and logarithmic over $F$. Moreover, suppose that $f \in F[\theta]$. If $f$ has an elementary integral, return true, an integral, and a list of all the logarithms appearing in the parent differential field of the solution. Otherwise, return false.

We give a derivation of the algorithm used. Write `f` as $\sum_{i=0}^{m} p_i\theta^i$. Consider theorems 12.2 and 12.5 in Geddes [1]:

**Theorem 1.** *Let $F$ be an elementary function field and $\theta$ be elementary and logarithmic over $F$, with $\theta' = \frac{u'}{u}$ for some $u \in F$. If $f \in F[\theta]$ with $\deg f > 0$, then:*

(i) *$f' \in F[\theta]$.*

(ii) *If the leading coefficient of $f$ is constant, then* $\deg f' = \deg f - 1$.

(iii) *If the leading coefficient of $f$ is* not *constant, then* $\deg f' = \deg f$.

**Theorem 2** (Liouville's Principle)**.** *Let $(F, D)$ be a differential field and $f \in F$. If there exists an elementary extension $(G, E)$ of $F$ and a $g \in G$ such that $Eg = f$, then there exist $v_0, \dots, v_m \in F$ and $c_1, \dots, c_m$ in the constant field of $G$ such that*

$$f = Dv_0 + \sum_{i=1}^{m} c_i \frac{Dv_i}{v_i}. \tag{1}$$

Using *Theorem 2* and *Theorem 1*, one can show that, if there is an elementary integral, we must have

$$\begin{aligned} \sum_{i=0}^{m} p_i \theta^i &= \left( \sum_{i=0}^{m+1} q_i \theta^i + \sum_{j=1}^{n} c_j \log v_j \right)' \\ &= q_{m+1}\theta^{m+1} + \sum_{i=1}^{m} (q_i' + (i+1) q_{i+1}' \theta') \theta^i + q_1 \theta' + q_0 + \sum_{j=1}^{n} c_j \frac{v_j'}{v_j} \end{aligned} \tag{2}$$

for some constants $c_1, \dots, c_n$ that are algebraic over the constant field of $F$, $v_0, \dots, v_n \in F(\theta)$ and $q_0, \dots, q_{m+1} \in F$ with $q_{m+1}$ constant. The part not dependent on $\theta$, namely $q_1\theta' + q_0' + \sum_{j=1}^{n} c_j \frac{v_j'}{v_j}$ comes from the integration of the trailing coefficient $p_0$, which we can compute using a recursive call to integration in $F$. Therefore, our algorithm need only compute the coefficients (except $q_0$) of the logarithmic polynomial $\sum q_i \theta^i$ in the integral.

First we consider the leading two terms. By equating the coefficients of powers of $\theta$ in (2), we have that $p_m = (m+1)q_{m+1}\theta' + q_m'$. We use the notation $\tilde{f} = \int f'$ to make it clear that we have only recovered $f$ up to a constant; that is, $f = \tilde{f} + C$. Then $\tilde{q}_m = \int p_m - \int (m+1) q_{m+1} \theta'$. Because $m+1$ and $q_{m+1}$ are constants, we have

$$\tilde{q}_m = \int p_m - (m+1) q_{m+1} \theta. \tag{3}$$

Moreover, if $\int p_m$ is not elementary or $\int p_m \notin F[\theta]$, then $\int \sum p_i \theta^i$ is not elementary. Again, we apply *Theorem 1* to find that $\int p_m = a\theta + b$ for some $a, b \in F$ with $a$ constant. Then

$$\tilde{q}_m = b + (a - (m+1) q_{m+1}) \theta, \tag{4}$$

and because $\tilde{q}_m \in F$, we must have $a - (m+1) q_{m+1} = 0$ (and hence $q_{m+1} = a/(m+1)$) and $\tilde{q}_m = b$.

The method to calculate the $q_i$-th for $1 \leq i < m$ is very similar to that for the leading two terms, and when we calculate each $q_i$, we also find the constant $C$ to add to $\tilde{q}_{i+1}$ so that $\tilde{q}_{i+1} + C = q_{i+1}$. By equating the coefficients of powers of $\theta$, we now have that

$$p_i = (i+1)(\tilde{q}_{i+1} + C)\theta' + q_i' = (i+1)\tilde{q}_{i+1}\theta' + q_i' + (i+1)C\theta'. \tag{5}$$

Notice that $(i+1)$ and $C$ are constants, so

$$\tilde{q}_i = \int (p_i - (i+1)\tilde{q}_{i+1}\theta') - (i+1)\theta. \tag{6}$$

Similarly to the first two terms, if the integral in (6) is not elementary or not in $F[\theta]$, then $\int \sum p_i \theta^i$ is not elementary. Otherwise, we apply *Theorem 1* to find that $\int (p_i - (i+1)\tilde{q}_{i+1}\theta')$ must be of the form $a\theta + b$ for some $a, b \in F$. By the same reasoning as above, it follows that $C = a/(i+1)$ and $\tilde{q}_i = b$.

`LogarithmicIntegral(f): RngDiffElt -> BoolElt, RngDiffElt, SeqEnum`

Named parameters:

`all_logarithms` with default value `[]`.

Let `Parent(f)` be an elementary function field $F(\theta)$ for some elementary function field $F$ with $\theta$ elementary and logarithmic over $F$. Return if `f` has an elementary integral, an integral, and a list of the logarithms generating the minimal elementary function field (w.r.t number of additional logarithmic generators) containing the integral.

## 6 Additional Notes

### 6.1 Coercion

The intrinsic `IsCoercible` not preserve the derivative of differential ring elements. For example, consider attempts to coerce $\exp x$ into $\mathbb{Q}(x, \log x)$ and $\log x$ into $\mathbb{Q}(x, \exp x)$:

```
> F<x> := RationalDifferentialField(RationalField());
> A<a> := LogarithmicFieldExtension(F, 1/x);
> B<b> := ExponentialFieldExtension(F, 1);
> B<b> := ExponentialFieldExtension(F, F!1);
> IsCoercible(A, b);
true a
> IsCoercible(B, a);
true b
```

I had expected that coercion would respect derivatives, but I do acknowledge that there may be good reasons that it does not.

## 6.2 Construction of Homomorphisms

Magma does not allow homomorphisms to be built from a domain that is a differential field that is not algebraic. For example, we cannot build the inclusion $\mathbb{Q}(x, \log x) \hookrightarrow \mathbb{Q}(x, \log(x+1), \log x)$ which preserves the derivative. An attempt to coerce $\log x$ into the second field gives $\log(x+1)$.

```
> A<a> := LogarithmicFieldExtension(F, 1/(x + 1));
> B<b> := LogarithmicFieldExtension(A, A!(1/x));
> G<g> := LogarithmicFieldExtension(F, 1/x);
> B!g;
a
> inj := hom< G -> B | B!a >;

>> inj := hom< G -> B | B!a >;
               ^
Runtime error in hom< ... >: Cannot build homomorphism from this
   domain
```

Again, there may be good reasons that homomorphism domains should not be allowed to be differential fields that are not algebraic.

## References

[1] K. O. (Keith O.) Geddes, S. R. (Stephen R.) Czapor, G. (George) Labahn, K. O. Geddes, S. R. Czapor, and G. Labahn. *Algorithms for Computer Algebra*. Kluwer Academic, Boston, 1992.

[2] Robert H. Risch. Algebraic properties of the elementary functions of analysis. *American Journal of Mathematics*, 101(4):743–759, 1979.