GSoC Application

Anurag Sharma

17.4.2014

1 Abstract

In symbolic integration one seeks a finite expression for the integral. The Risch algorithm is an algorithm for symbolic integration of any elementary function. Given an elementary function, it will either produce an antiderivative, or prove that none exists. The proposed project aims to continue the work of Aaron Meurer and Chetna Gupta done in 2010 and 2013 GSoC projects respectively, i.e implementing the algorithm from Manuel Bronsteins book, Symbolic Integration I: Transcendental Functions. A successful implementation of this would not only provide concrete algorithms for transcendental functions but would also form the basis for very similar Karr Algorithm, decision procedure for symbolic summation, yet to be implemented in sympy. Since much of the work from the book aforementioned has already been done, the proposal aims at completing the implementation of the sub-algorithms mentioned in the book and go towards implementing Risch Algorithm for algebraic functions which prove to be tougher than the algorithm for transcedental functions.

2 Personal Details

- Name : Anurag Sharma
- Email Id : anurags92@gmail.com
- Alternate Email Id : sharmaan@iitk.ac.in
- IRC : anurags92 on freenode
- Github : anurags92

3 About Me

I am a sophomore Mathematics student at Indian Institute of Technology, Kanpur, India. I have keen interests in Algebra, Field theory and Galois theory. I was first introduced to Differential Algebra while studying for this project and since then I have made major gains in my understanding of Differential Galois theory.

I wish to pursue further studies in Commutative Algebra and Category Theory. In my two years of undergraduate study at my institute I have been exposed to varied topics in math which include Real and Complex Analysis, Abstract Algebra, Universal Algebra, Linear Algebra, Ordinary and Partial Differential Equations etc.

4 Programming Experience

I am proficient in C, C++ have a decent knowledge of Python. All my previous programming experience has been for related to various hackathons organised in our college and course projects. Here is a link to a Gesture recognition applet that was designed by me and my friends for a summer project in our first year.
Link : `http://www.youtube.com/watch?v=q-55tBvrD2g` Apart from the hackathons and development related prject I am also a member of a three member team - MemoryOverflow, which represented IIT Kanpur in ACM ICPC regionals held at IIT Kharagput. My SPOJ handle is anurags92. I have solved 150+ problems on SPOJ and am quite good at writing code in timed contests. If I get selected in GSoC, it would be my first big contribution to an open source software. I had used git earlier for my projects but that was restricted only to basic commands. During my 2 3 months association with the community I have become quite familiar with this version control.

5 Contributions to Sympy

My Contribution to sympy are as follows:

- Implemented a new class of Beta function which was earlier implemented as a subformula of Gamma function
  https://github.com/sympy/sympy/pull/2782

- Improved the efficiency of order_at function from n to log(n) where n is the power of prime in the https://github.com/sympy/sympy/pull/7237

6 Timeline

I have started work on Chetna's pull request :https://github.com/sympy/sympy/pull/2380 I plan to continue this work in community bonding period and first two week of the summers.

1. Week 1-2
   (a) Make a common function for converting the denominator to special denominator for the Parametric Risch Differential Equation and Risch Differential Equation.

   Pull Request 1. (This branch will get merged after this)

2. Week 3-4
   (a) Get Chetna's Cds & is_derive branch merged.
   (b) Implementing Structure theorems for is_derive and is_log_derive in prde.py.

3. Week 5-6
   (a) Solution for Hyper-tangent Case using Cancellation algorithm.
   (b) Writing test cases for the functions written in Week 3-4 and Week 5-6.

   Pull Request 2. (On a different branch)

4. Week 7
   (a) Implementation of Liouvillian Case for Parametric Risch Differential Equation.
   (b) Implementation of Non Linear Case for Parametric Risch Differential Equation.

   Mid-Term Evaluation.

5. Week 8-9

(a) After the tasks mentioned above are finished, I would be able to complete Parametric Logarithmic derivative

(b) Implementation of Non Linear Case for Parametric Risch Differential Equation Heuristic Solver.

6. Week 10

   (a) Implementation of Complete Parametric Logarithmic Complete Solver. Completing this would be a big success for the project and this would probably finish the work in prde.py and hopefully it could then get merged in the master.

   Pull Request 3. (On the same branch as PR 2)

7. Week 11-12

   (a) After finishing prde.py I would like to hook that code with its uses in rde.py and risch.py.

   (b) Functions in rde.py like cancel_exp and cancel_primitive would make use of the work done in Week 1-2 and Week 10.

8. Week 13

   (a) Would review all the pull requests and would add more tests cases for each PR.

   (b) Would like to clean the code in rde.py and remove as many NotImplemented exceptions as possible. Hopefully after Week 10's work this would be routine exercise and would get completed easily.

   (c) Add tests cases for rde.py and check if everything is in good shape.

   Pull Request 4.

7 Theory Relevant to all implementations

A **differential field** is a field F with a differentiation operator D such that for any $f, g \in F$ $D(f+g) = Df + Dg$ and $D(fg) = (Df)g + f(Dg)$.

A $\mathbf{t}$ is a **monomial** over (F, D) if $\mathbf{t}$ is transcendental over F and $D(t)$ is a polynomial in $\mathbf{t}$ with coefficients from F.

Extension of a differential field: We can adjoin new elements $t_1, ..., t_n$ to a differential field (F, D) to get a field $F(t_1, ..., t_n)$. The result is a differential field extension of (F, D) if:

- $D(t_i) \in F(t_1, ..., t_n)$
- D can be extended consistently to $F(t_1, ..., t_n)$.

8 Implementation Detail

Week 1-4

My first target would be to get this PR https://github.com/sympy/sympy/pull/2380 merged.
There are few issues with the PR as of now. Most important of them is a routine similar to is_log_derive_k[t] in prde.py.
"is_log_derive_k[t]" recognises which algebraic field extension can be written in terms of lower field extension.
Example: $Q(x, e^x, e^{log(x)/2})$ has transcedence degree 2 and not 3 because $e^{log(x)/2}$ can be written in terms of x. So we need an algorithm which creates the tower of field extensions which are "irreducible" in a sense. This is effectively done by "is_log_derive_k[t]" for exponential and logarithmic functions. We have to extend the function for hypertangent case.

8.1 Theory

First we build a finite tower $T = F(x, \theta_1, \theta_2, ..., \theta_n)$ of monomial extensions over Q. Each of the θ_i are elementary over F. Now to check whether T is a transcedental extension of Q we have to determine the algebraic structure of each of $f_i = F(x, \theta_1, \theta_2, ..., \theta_{i-1})$ over Q and then apply it recursively for k_{i+1} till we know it for all k_n and then we can check the nature of F over Q.If it is transcedental then only we proceed with integration otherwise raise exception NotImplemented.
Reference [2].

Week 5-7

Cancellation algorithms although proposed last year were not implemented for :

- Liouvillian Case

- Nonlinear Case
- Hypertangent Case

For Liouvillian case [1] describes the algorithm to solve Parametric Risch Differential equation which reduces the problem to a similar in non cancellation cases. I'll follow the text to implement this routine. There is no general algorithm for solving the equation in Nonlinear Case. The algorithm described in [1] works only in a special case where $S^{irr} \neq \phi$. This can also be implemented as shown in the text. For the last Hypertangent Case, the algorithm differs from that of Risch Differential Equation only for one case as discussed below.

8.2 Theory

We reduce the problem of solving Parametric Risch Differential Equation to one of solving the following equation:

$Dq + bq = \sum c_i q_i \rightarrow (1)$.
For hypertangent case $\frac{D(t)}{t^2+1} = \eta$.
Now for $b = b_0$ - ηnt is the case that is not handled by PolyRischDE-CancelTan given in [1]. For this case the algorithm is similar to Nonlinear Cases.

Week 8-10

The task for week 8 onwards is to integrate the different snippets written till now. I plan to first start with completing Parametric Logarithmic Derivative Heuristics Solver and Paramteric Logarithmic Derivative complete solver. Code for heuristic solver is already there in Chetna's branch I will have to make it work with "is_log_derivative" created in Week 2-4. Psuedocode given in [1] will come in handy for this function. If heuristic fails then we move on to solve the equation using structure theorems described in Chapter-9 of [1]. I have not gone over the theory of that particular portion so I can not explain the details of that part right now, but I have a brief idea on how to go about it which I try to explain below.

8.3 Theory

Parametric Logarithimic Derivative Problem aims to solve differential quation of the form:

Given a differential field K of characteristic 0, an hyperexponential monomial θ over K and $f \in K$, we need to find whether there exist integers n,m $\in Z$ with n $\neq$0 such that $nf = \frac{Dv}{v} + \mathrm{m}\frac{D\theta}{\theta}$ has a solution v $\in K$ and also to find the solution if that exists. Now this equation can be solved in two ways. One is employed by heuristic method and the other is based on structure theorems. f will always has an elementary integral over K for Parametric Logarithmic problems. Now let F be the elementary extension over $K(\theta)$ and $g \in F$ such that $f = Dg$. Since (1) has a solution we can write it in the form $nf = \frac{Dv(\theta)^m}{v(\theta)^m}$. This proves that $f = Dg$ is logarithmic over F. Now using a lemma in Chapter 9 of [1] we show that this is true iff there exist $r_i \in Q$ such that $\sum_{i \in L} r_i Dt_i + \sum_{i \in E} r_i \frac{Dt_i}{t_i} = f$. I do not completely understand the proof of this lemma. But we can safely assume that this holds. Now our E and L are the set of exponential and logarithmic extensions respectively which can be extracted with our already present functions. Now we are only left to find the rational solutions of the new equation. This can be done by obtaining a coefficient matrix in C (F is expressed as extension of C).