SymIntegration

Freya · Lasthrim · DS Glanzsche[1]

Berlin-Sentinel Academy of Science, AliceGard

2025 Edition

[1]A thank you or further information

Contents

Preface

For my Wife Freya, and our daughters Solreya, Mithra, Catenary, Iyzumrae and Zefir.

For Lucrif and Znane too along with all the 8 Queens.

For Albert Silverberg, Camus and Miklotov who left TREVOCALL to guard me.

To Nature(Kala, Kathmandu, Big Tree, Sentinel, Aokigahara, Hoia Baciu, Jacob's Well, Mt Logan, etc) and my family Berlin: I have served, I will be of service.

To my dogs who always accompany me working in Valhalla Projection, go to Puncak Bintang or Kathmandu: Sine Bam Bam, Kecil, Browni Bruncit, Sweden Sexy, Cambridge Klutukk, Milan keng-keng, Piano Bludut and more will be adopted. To my cat who guard the home while I'm away with my dogs: London.

The one who moves a mountain begins by carrying away small stones - Confucius

A book to explained how we create C++ library from forking / branching out SymbolicC++ 3.35 to improve its' symbolic integral computation.

a little about GlanzFreya:
Freya the Goddess is (DS Glanzsche') Goddess wife. We get married on Puncak Bintang on November 5th, 2020 after we go back from Waghete, Papua. My wife birthday (Freya the Goddess) is on August 1st. After not working with human anymore since 2019, I (DS Glanzsche) work in a forest, shaping a forest into a natural wonder / like a canvas for painting, and also clean up trashes that are thrown in the forest. Thus after 6 years working with Nature I have found what can makes me waltz to work, it is going to the forest in the morning then go back home again and study / learn science, engineering, computer programming.

Figure 1: *Freya, thank you for everything, I am glad I marry you and I could never have done it without you.*

Figure 2: *I paint her 3 days before Christmas in 2021.*

Chapter 1

Introduction Story

On top of the mountain a heart shaped stone waiting for You, my eyes can't see, my body can't touch, but my heart knows it's You. Forever only You. - ***Glanz to Freya***

This book is written on May 9th, 2025, it is exactly 1 month after I finish with the algorithms and C++ codes for $\int \sin^n x \, dx$, $\int \cos^n x \, dx$, $\int \tan^n x \, dx$, $\int \sec^n x \, dx$, $\int \csc^n x \, dx$.

SymIntegration itself is a C++ library that is branching out from SymbolicC++3.35. We don't really need to start from scratch. The main idea is to improve its symbolic integration codes. Our main focus is to make it able to compute:

1. All kinds of standard integral form (trigonometry, inverse trigonometry, polynomial, transcendental, hyperbolic).
2. The sum, product and divide combination of the standard functions.
3. To be able to compute improper integrals with cases (e.g. computing mean and variance for exponential distribution)

On April 9th, Sentinel, a Nature residing in a big robust tree in VP (Valhalla Projection) forest told me to start creating C++ library that can do integration like what SymPy able to do, or Mathematica. Around April 2025, GiNaC cannot handle integration by parts, and SymbolicC++ also returns $\int \sin^n x \, dx$ with $\frac{1}{n+1} \sin^{n+1}$. So then we decided to branching out / forking from SymbolicC++ version 3.35. Fixing the C++ codes there so the computation will be made right.

We will cover mostly tons of integral problems, how to find its patterns then how to create the C++ code so SymIntegration will be able to return the integral correctly, thus there will be a lot of technical writings after this chapter. The requirement to comprehend SymIntegration mathematical background is you should at least self learn, read by yourself a Calculus book [6]. No need to go to college, it is expensive and only add more loan to your life.

The name is chosen as SymIntegration, not only it has a great Chaldean numerology of 46 (on purpose), a king of success number / Nobel winner number / President / richest person number, but also it is in accordance with the main purpose, to be able to compute all kinds of integration with C++. Which still dominated by SymPy, Mathematica or MATLAB. Mathematica and MATLAB are using license, and I am not working with human anymore, thus I do not have salary / stable income like I used to, so I cannot lean on that (MATLAB / Mathematica), SymPy is

free, can be called from JULIA too besides Python, but then, I read that Game Engine (CryEngine, Genie Engine, Godot, Unreal Engine) that is able to create amazing game with beautiful graphics, animation, real life graphics, you name all the game with best graphics (I can say from Age of Empires, Call of Duty, Crysis, Cyberpunk 2077, Death Stranding, Dota, Grand Theft Auto, Final Fantasy, Metro 2033, Pokemon Go, The Witcher), and I can say it is made with C++. It is the near hardware and one of the fastest programming language if you have a mind on creating something better in the future, like Science and Engineering simulator, e.g. Bullet, Project Chrono, HySys, Autodesk Civil Engineering. Let say if you have time you can create your own Autodesk instead of paying for its license, basically the brain behind them are derivative, integral, statistics and algebra.

Basically we choose C++ so we can create something long lasting in the future, and then I (DS Glanzsche) dedicate everyday of my life to eat and breathe C++ to be able to achieve this stage, hopefully it can be developed further till we can get practical use, like having a simulation for fluid flow, pharmacokinetics (to compute how drugs are metabolized by the body), simulation for human' organ, or to create a stable bridge or skycraper, since the basic of calculus, integral, linear algebra, statistics can be covered nicely by SymIntegration. The front end looks for plot and simulation can be done by third party C++ library like ImGui, MatplotPlusPlus or OpenGL.

Symbolic integration is basically harder than differentiation, differentiation has rigid rules: sum rule, product rule, chain rule, divide rule. While integration has only method of substitution and integration by parts, which back to the method itself. We can obtain different results for integration, but same result for differentiating different functions with respect to the same variable, like this case:

$$D_x \frac{-\cos^2(nx)}{2n} = \cos(nx)\ \sin(nx)$$
$$D_x \frac{\sin^2(nx)}{2n} = \cos(nx)\ \sin(nx)$$

Depending on the method of substitution, which one we substitute, if we reverse the equations above into integral problems, then we will obtain:

$$\int \cos(nx)\ \sin(nx)\ dx = \frac{-\cos^2(nx)}{2n} + C$$
$$\int \cos(nx)\ \sin(nx)\ dx = \frac{\sin^2(nx)}{2n} + C$$

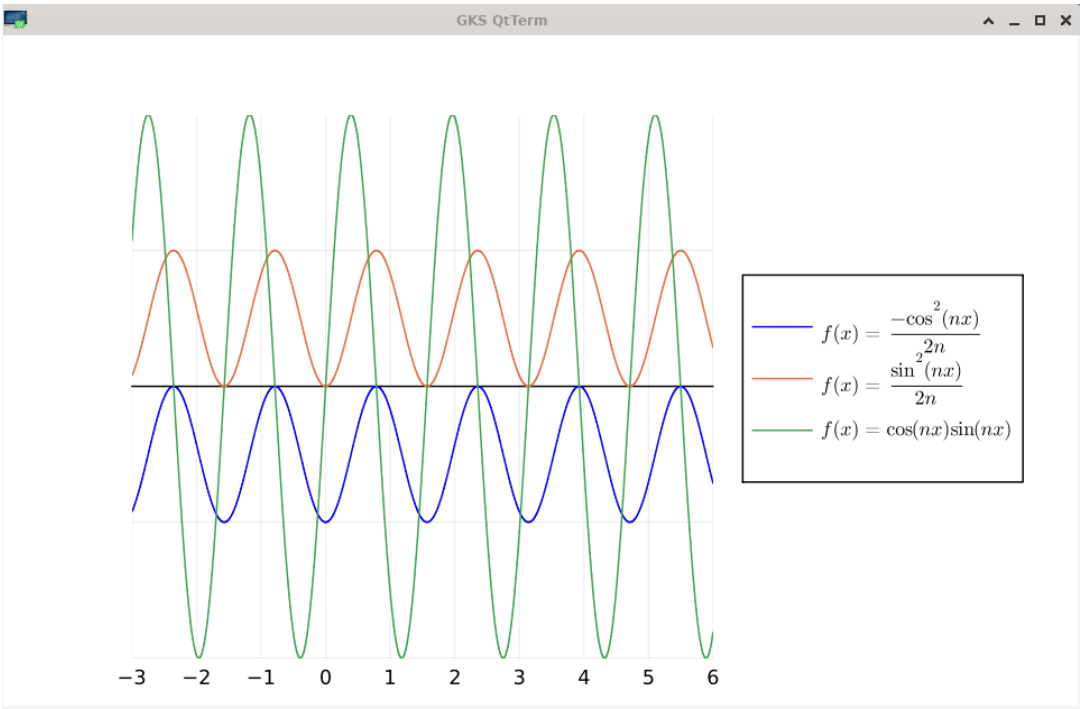

Figure 1.1: *Integrating a function with different method can result in at least two different functions.*

To compute the definite integral of a function in an interval you have to know how it behaves on the entire interval, and to compute the indefinite integral you have to know how it behaves on all intervals. While differentiation is a "local" operation, it computes the derivative of a function at a point you only have to know how it behaves in a neighborhood; distance, speed and acceleration from basic physics is the example of differentiation and integral. Computing integral / antiderivative often requires clever algebraic manipulation and technique and there are no universal rules for integration like there are for differentiation.

Integrals are fundamental in all science and engineering fields for solving problems involving area, volume, and other accumulations, some specific applications of integrals in science and enginnering:

1. To calculate the area of irregular shapes, the volume of solids, and the surface area of objects.
2. To determine center of mass and moments of inertia for various shapes, they are crucial for structural analysis and design.
3. To calculate work done by a variable force, which is essential in mechanics and thermodynamics.
4. To calculate energy, power, and signal properties over time in electrical and computer engineering.
5. To determine stresses and strains analysis in structures under various loads.
6. To calculate fluid flow, pressure distributions, and forces exerted by fluids.

7. To compute probability distributions and statistical calculations.
8. To predict the trajectory of satellites and other objects in motion, ensuring they follow the intended path.
9. To calculate drug accumulation and half-life in a body, it is crucial for understanding how drugs are metabolized and eliminated by the body in Pharmacokinetics.
10. To model how diseases spread through a population with Susceptible-Infectious-Removed (SIR) model.
11. To understand better the complex processes of modeling tumor growth and metastasis.
12. To understand how reactions progress and their rates for reaction kinetics in chemistry.
13. To calculate work done during processes like gas expansion and changes in internal energy or enthalpy. These calculations are crucial for understanding the energetics of chemical reactions and processes.
14. To solve problems related to process design, optimization and understanding reaction kinetics in building a plant in Chemical Engineering.
15. To determine the volume of concrete needed for a structure, the area of a road surface, or the volume of soil required for an excavation in building bridges and structures.
16. To find the deflection (bending) and slope of a beam under load, which is crucial for ensuring structural stability.
17. To determine the velocity of a fluid flow, which is important in designing hydraulic structures like dams and pipelines.
18. To calculate the pressure distribution in fluids, which is essential for understanding the forces exerted on structure by fluids.
19. To understand the behavior of capacitors and inductors. The voltage across a capacitor is the integral of the current flowing through it, and the current through an inductor is the integral of the voltage across it.
20. To compute total charge and energy in Electrical Engineering.
21. To determine the circuit's voltage, current, or other parameters as a function of time by using integration to solve differential equations for many kind of circuit behaviors.

Now we will take a look at the applications of the integration in computer science:

1. Mathematics and computing.
2. Block chain method.
3. Quantum computing.
4. Big data analytics.
5. Neural Networking.
6. Artificial Intelligence.

7. Machine Learning.

Integration, calculus, linear algebra are the brain behind the applications above.

Triple integrals are used in physics, engineering, and even finance. If we get ahold of the C++, we can do the integral computation right, then it is only a matter of presentation, create the plot / simulation / computation for our goal, e.g. to create a stable bridge that can handle earthquake with magnitude of 10 SR. It takes simulation first, before constructing the real product.

The Operating System and Softwares in Use:

1. GFreya OS version 1.8 (based on LFS and BLFS version 11.0 System V books)

2. GCC version 11.2.0 (to compile C++ codes in Box2D)

3. SymIntegration version 1.51

By May 9th, 2025 SymIntegration is able to compute

1. The integral of $\sin(ax+b), \cos(ax+b), \tan(ax+b), \cot(ax+b), \sec(ax+b), \csc(ax+b)$ with $ax+b$ is a polynomial of order 1.

2. The integral of $\frac{1}{(ax+b)}$', '$\frac{1}{ax^2+bx+c}$.

3. The integral and derivative of $\text{asin}\,(ax+b), \text{acos}\,(ax+b), \text{atan}\,(ax+b), \text{acot}\,(ax+b), \text{asec}\,(ax+b), \text{acsc}\,(ax+b)$ with $ax+b$ is a polynomial of order 1.

4. The integral and derivative of all hyperbolic trigonometry functions $\sinh(ax+b), \cosh(ax+b), \tanh(ax+b), \coth(ax+b), \text{sech}\,(ax+b), \text{csch}\,(ax+b)$.

5. The integral with the form of '$x^b e^{ax}, x^b e^x$' with integration by parts method.

6. The integral of $\sin(ax)\cos(bx), \cos(ax)\cos(bx), \sin(ax)\sin(bx)$

We are really appreciate those who spend time to write constructing critics, not hate words without reason that will only add more good luck and karma to me, write their opinions on what should be added in SymIntegration or if there is an incorrect C++ codes or algorithm to compute the integrals. We cannot provide the readers who send nice feedbacks with anything now, but if we have money, when someone send / email me a very good feedbacks then we will send you either cookies, doughnuts or parcel of foods. Feedbacks can be sent to **dsglanzsche@gmail.com**.

Chapter 2

Symbolic Integral Computation with SymIntegration

"The best and most beautiful things in the world cannot be seen or even touched - they must be felt with the heart" - ***Helen Keller***

We will first learn how to create the shared library for C++ library in a very simple and manual way, then we will see the source code that composing this C++ library, SymIntegration.

I. Build and Install SymIntegration in GFreya OS

We are using GFreya OS as it is a custom Linux-based Operating System, your distro your rules. Building applications from scratch instead of using package manager, you learn more from mistakes than just depending on package manager.

To download SymIntegration, open terminal / xterm then type:
git clone https://github.com/glanzkaiser/SymIntegration.git
Enter the directory then type:
cd src
bash dynamiclibrary.sh

or you can also type from the main parent directory of SymIntegration repository:

```
cd src
g++ -fPIC -c *.cpp
g++ -shared -o libsymintegration.so *.o
```

Code 1: *Create dynamic library*

It will create the dynamic library **libsymintegration.so** with a very manual way, instead of using CMake, we are using less space.

Assuming you are using Linux-based OS, you can then copy / move the dynamic library to **/usr/lib**.

Then open terminal and from the current working directory / this repository main directory:

```
cd include
cp -r * /usr/include
```

Code 2: *Move include folder*

When we want to create the shared library it will look for this header files in the default path where the include files are usually located. In Linux OS **usr/include** is the basic / default path.

All files, examples codes and source codes used in this books can be found in this repository:
https://github.com/glanzkaiser/SymIntegration

We are using less, minimal amount of code to create the library, if you compare it with the original SymbolicC++ that has **.configure** and **Makefile** that will be spawn after you configure it, we don' t use any of that.

Two simple commands with **g++** can already make a shared symbolic integration computation library, with limitation still.

If you are using Windows OS or MacOS then you are on your own.

The C++ codes are only located in the **/src** directory , it is designed to contain all the '.cpp' files, then inside the **/include/** we have header files and another folder **/include/symintegral/** is

also a folder that contain header files too.

You can open them one by one, read them and making sense of it based on the function that you are looking for.

There are total of 27 **.cpp** files and 26 **.h** / header files. So it won't take a long time for someone to learn it.

II. Techniques of Integration

We will cover some techniques of integration and also two Fundamental Theorems of Calculus that are very useful in solving integral problems or even differential equations.

Theorem 2.1: The First Fundamental Theorem of Calculus

Let f be continuous on the closed interval $[a, b]$ and let x be a (variable) point in (a, b). Then

$$\frac{d}{dx}\int_a^x f(t)\,dt = f(x)$$

Theorem 2.2: The Second Fundamental Theorem of Calculus

Let f be continuous (hence integrable) on $[a, b]$, and let F be any antiderivative of f on $[a, b]$. Then

$$\int_a^b f(x)\,dx = F(b) - F(a)$$

- Integration by Parts

[SI*] From the analogue of the product rule for differentiation. Suppose we have two functions $f(x)$ and $g(x)$ with

$$\frac{d\,f(x)g(x)}{dx} = f(x)\frac{dg(x)}{dx} + g(x)\frac{df(x)}{dx} \tag{2.1}$$

Integrating both sides and rearranging the terms, we will get the integration by parts formula:

$$\int f(x)\frac{dg(x)}{dx}\,dx = f(x)g(x) - \int g(x)\frac{df(x)}{dx}\,dx \tag{2.2}$$

This formula is only useful if $\int g(x)\frac{df(x)}{dx}\,dx$ or $\int f(x)\frac{dg(x)}{dx}\,dx$ is easier to integrate than $\int f(x)g(x)\,dx$.

The application for integration by part is very useful in statistics, e.g. to be able to compute the mean and variance for exponential distribution.

- Substitution Rule for Indefinite Integral

[SI*] The substitution rule is nothing more than the Chain Rule in reverse.

Theorem 2.3: Substitution Rule for Indefinite Integrals

Let g be a differentiable function and suppose that F is an antiderivative of f. Then

$$\int f(g(x))g'(x)\,dx = F(g(x)) + C$$

Theorem 2.4: Substitution Rule for Definite Integrals

Let g have a continuous derivative on $[a, b]$, and let f be continuous on the range of g. Then

$$\int_a^b f(g(x))g'(x)\,dx = \int_{g(a)}^{g(b)} f(u)\,du$$

where $u = g(x)$.

III. Integration by Parts

- Example: Exponential Distribution

[SI*] The exponential distribution, which is sometimes used to model the lifetimes of electrical or mechanical components, has probability density function

$$f(x) = \begin{cases} \lambda e^{-\lambda x}, & \text{if } 0 \leq x \\ 0, & \text{otherwise} \end{cases}$$

where λ is some positive constant. Compute the mean μ and the variance σ^2.

Solution:

To compute for the mean we will have

$$\begin{aligned} E(X) &= \int_{-\infty}^{\infty} xf(x)\,dx \\ &= \int_{-\infty}^{0} x \cdot 0\,dx + \int_{0}^{\infty} x\lambda e^{-\lambda x}\,dx \end{aligned}$$

We apply integration by parts in the second integral with

$$\begin{aligned} u &= x \\ dv &= \lambda e^{-\lambda x}\,dx \\ du &= dx \\ v &= -e^{-\lambda x} \end{aligned}$$

Thus

$$\begin{aligned} E(X) &= [-x\lambda e^{-\lambda x}]_0^{\infty} - \int_0^{\infty} (-e^{-\lambda x})\,dx \\ &= (-0+0) + \left[-\frac{1}{\lambda}e^{-\lambda x}\right]_0^{\infty} \\ &= \frac{1}{\lambda} \end{aligned}$$

The variance is

$$\begin{aligned} \sigma^2 &= E(X^2) - \mu^2 \\ &= \int_{-\infty}^{\infty} x^2 f(x)\,dx - \left(\frac{1}{\lambda}\right) \\ &= \int_{-\infty}^{0} x^2 \cdot 0 + \int_0^{\infty} x^2\,\lambda e^{-\lambda x}\,dx - \frac{1}{\lambda^2} \\ &= [-x^2 e^{-\lambda x}]_0^{\infty} - \int_0^{\infty} (-e^{-\lambda x})2x\,dx - \frac{1}{\lambda^2} \\ &= (-0+0) + 2\int_0^{\infty} xe^{-\lambda x}\,dx - \frac{1}{\lambda^2} \\ &= 2\frac{1}{\lambda^2} - \frac{1}{\lambda^2} \\ &= \frac{1}{\lambda^2} \end{aligned}$$

[SI*] When they say integration is an art it is true, you need more practice to be able to handle all kinds of integral problems.

If we are facing this kind of equation:

$$f(x) = x\lambda e^{-\lambda x}$$

then what we need to do is learning the pattern

$$\begin{aligned}
\int xe^x \, dx &= (x-1)e^x + C \\
\int x^2e^x \, dx &= (x^2 - 2x + 2)e^x + C \\
\int x^3e^x \, dx &= (x^3 - 3x^2 + 6x - 6)e^x + C \\
\int x^4e^x \, dx &= (x^4 - 4x^3 + 12x^2 - 24x + 24)e^x + C \\
&\vdots
\end{aligned}$$

so for the general function of

$$f(x) = x^n e^x$$

The algorithm will be like this:

```
Symbolic sgn = 1;
int k = 1;
Symbolic integral;
for (int i = 0; i <= n; i++)
{
integral += k * sgn * (x^(n—i)) * exp(x)
sgn = —sgn;
k *= n—i;
}
return integral;
```

The C++ file to handle cases of integration by parts is located in **src/integrate.cpp**

```
Symbolic integrate(const Symbolic &f,const Symbolic
    &x)
{
       list<Equations> eq;
       list<Equations>::iterator i;
       UniqueSymbol a, b, c;
...
...
```

Code 3: *src/integrate.cpp*

Now for the case of $f(x) = x^n e^x$, it is handled at this part of codes

```
Symbolic integrate(const Symbolic &f,const Symbolic &x)
{
```

```
                ...
                ...
                eq = ((x^b)*exp(x)).match(f, (a,b)); // case for x^b
                     * exp(x)
                for(i=eq.begin(); i!=eq.end(); ++i)
                try {
                       Symbolic bp = rhs(*i, b);
                       if(bp.type() == typeid(Numeric)
                       && Number<void>(bp)->numerictype() == typeid(
                           int)
                       && Number<void>(bp)>Number<int>(0))
                       {
                              int n = CastPtr<const Number<int> >(bp
                                  )->n, sgn = 1;
                              Symbolic integral, nf = 1;
                              for(; n>=0; nf*=n, --n, sgn=-sgn)
                                  integral += sgn*nf*(x^n)*exp(x);
                              return integral;
                       }
                } catch(const SymbolicError &se) {}

        }
```

Code 4: *src/integrate.cpp to handle $x^n e^x$*

For the computation of mean and variance of exponential distribution with SymIntegration we will get this raw result, where the e^{-infty} is still as it is, and not returning to 0, we will work on it in the future.

```
#include <iostream>
#include "symintegrationc++.h"

using namespace std;

int main(void)
{
       Symbolic x("x"), l("l"), Inf("Inf"), y, Ex,
           Varx, u, dv, v, du;

       y = l*exp(-l*x);
       u = x;
       dv = l*exp(-l*x);
       du = df(u,x);
       v = integrate(dv,x);

       cout << "f(x) = " << y << endl;

       for(int i=1;i<=1;i++)
       {
              y = integrate(y,x);
```

```
                cout << i << "-st integral of f(x) = "
                     << y << endl;
            }
            cout << "int_{0}^{Inf} f(x) = " << y[x==Inf]
                - y[x==0] <<endl;

            Ex = x*l*exp(-l*x);

            cout << "\nx f(x) = " << Ex << endl;
            cout << "\nu = " << u << endl;
            cout << "dv = " << dv << endl;
            cout << "du = " << du << endl;
            cout << "v = " << v << endl;

            Ex = integrate(Ex,x);
            cout << "\nE(x) = int x f(x) = " << Ex <<
                endl;
            cout << "int_{0}^{Inf} x f(x) = " << Ex[x==
                Inf] - Ex[x==0] <<endl;

            Varx = x*x*l*exp(-l*x);
            Varx = integrate(Varx,x);

            cout << "\nint x^{2} f(x) = " << Varx << endl
                ;
            cout << "int_{0}^{Inf} x^{2} f(x) = " << Varx
                [x==INFINITY, l==1] - Varx[x==0] <<endl;
            cout << "\nVar(x) = int x^{2} f(x) - mu^{2} =
                 " << Varx[x==INFINITY, l==1] - Varx[x
                ==0] - (1/l^(2)) << endl;

            return 0;
        }
```

Code 5: *Integration by Parts*

IV. The Method of Substitution

- Some Examples

[SI*] Suppose we want to compute the integral

$$\int x \sin x^2 \, dx$$

Solution:
Here the appropriate substitution is

$$u = x^2$$
$$du = 2x \, dx$$

Thus

$$\begin{aligned}\int x\ \sin\ x^2\ dx &= \int \frac{1}{2}\ \sin\ x^2\ (2x\ dx) \\ &= \int \frac{1}{2} \sin\ u\ du \\ &= -\frac{1}{2} \cos\ u + C \\ &= -\frac{1}{2} \cos\ x^2 + C\end{aligned}$$

[SI*] In SymIntegration the method of substitution can be handled in **src/integrate.cpp**, we can create a pattern of equation to be solved, just like the integration by parts method.

V. Fraction Integral

- Fraction Level 1

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{1}{ax+b}$$

a and b are constants.

The integral will be

$$\int \frac{1}{ax+b}\,dx = \frac{\ln(ax+b)}{a} + C \tag{2.3}$$

For the integrate function alone in SymIntegration we do not include the constant C that is obtained after integration.

* Note that the C of the indefinite integration cancels out, as it always will, in the definite integration. That is why in the statement of the Second Fundamental Theorem of Calculus we could use the phrase **any derivative**. In particular, we may always choose $C = 0$ in applying the Second Fundamental Theorem of Calculus [6].

[SI*] The C++ code to handle this fraction level 1 integral problem to make sure the integral will return the correct result is located in **src/functions.cpp** under the function **Symbolic Power::integrate(const Symbolic &s) const**, we still use the function **integrate** for this type of integral.

```
Symbolic Power::integrate(const Symbolic &s) const
{
const Symbolic &a = parameters.front();
const Symbolic &b = parameters.back();
...
...
}
```

```
Symbolic Power::integrate(const Symbolic &s) const
{
...
...

if(b == -1 && parameters.front().coeff(s,2) == 0)
    return ln(parameters.front()) * (1/ parameters.
    front().df(s));
...
...
}
```

Code 6: *fraction level 1 integral*

The **if** statement above is a condition: if the input of a function meets the criteria that the power is -1 / a fraction ($f(x)^{-1} = \frac{1}{f(x)}$) then we will return it as **ln(parameters.front()) * (1/ parameters.front().df(s))**. The symbolic s is used to replace the variable in the integration problem.

I will introduce to the reader here about **parameters.front()** and **parameters.back()**, suppose we have a function

$$f(x) = (ax^2 + bx + c)^d \tag{2.4}$$

with a, b, c as constants.

* The **parameters.front()** of the function from equation (2.4) is $ax^2 + bx + c$.
  parameters.front().coeff(s,2) is a.
  parameters.front().coeff(s,1) is b.
  parameters.front().coeff(s,0) is c.

* The **parameters.back()** of the function from equation (2.4) is d.

With this knowledge we can create a lot of **if, if.. else..** conditions to make the integration computation complete and better.

- Fraction Level 2

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{1}{ax^2 + bx + c}$$

a, b and c are constants.

The integral will be

$$\begin{aligned}\int \frac{1}{ax^2 + bx + c}\, dx = &-\sqrt{\frac{-1}{4ac - b^2}} \ln\left(x + \frac{-4ac\sqrt{\frac{-1}{4ac-b^2}} + b^2\sqrt{\frac{-1}{4ac-b^2}} + b}{2a}\right) \\ &+ \sqrt{\frac{-1}{4ac - b^2}} \ln\left(x + \frac{4ac\sqrt{\frac{-1}{4ac-b^2}} - b^2\sqrt{\frac{-1}{4ac-b^2}} + b}{2a}\right) + C\end{aligned} \tag{2.5}$$

The codes for the SymIntegration to handle this integral is located in **src/functions.cpp**

```
...
...

Symbolic Power::integrate(const Symbolic &s) const
{
      ...
      ...
      if(b == -1 && parameters.front().coeff(s,2) != 0)
      {
```

```
                    double a1 = parameters.front().coeff(s,2);
                    double b1 = parameters.front().coeff(s,1);
                    double c1 = parameters.front().coeff(s,0);
                    double D_inv = sqrt(-1/(4*a1*c1-b1*b1));

                    return - D_inv * ln(s + (-4*a1*c1*D_inv + b1
                        *b1*D_inv + b1)/(2*a1)) + D_inv * ln(s +
                        (4*a1*c1*D_inv - b1*b1*D_inv + b1)/(2*a1
                        )) ;
                }
                ...
                ...
            }
            ...
            ...
```

Code 7: *fraction level 2 integral type 1*

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{a_1 x}{ax^2 + bx + c}$$

a_1, a, b, and c are constants.

The integral will be

$$\begin{aligned}\int \frac{a_1 x}{ax^2+bx+c}\,dx = a_1 \Bigg[& \left(-\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)} + \frac{1}{2a}\right) * \\ & \ln\left(x + \frac{-4ac\left(-\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)} + \frac{1}{2a}\right) + b^2\left(-\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)} + \frac{1}{2a}\right) + 2c}{b}\right) \\ & + \left(\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)} + \frac{1}{2a}\right) * \\ & \ln\left(x + \frac{-4ac\left(\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)} + \frac{1}{2a}\right) + b^2\left(\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)} + \frac{1}{2a}\right) + 2c}{b}\right) \Bigg] + C\end{aligned} \tag{2.6}$$

The codes for the SymIntegration to handle this integral is located in **src/integrate.cpp**, instead of **integrate** we create a new function **fraction integrate** to handle this type of integral because this type of integral cannot be read nicely with **Symbolic Power::integrate(const Symbolic &s) const** that is in **src/functions.cpp**.

By July 8th, 2025, the function **fraction integrate** already has around 190 lines of code.

```
    ...
    ...
```

```
Symbolic fractionintegrate(const Symbolic &fnum, const Symbolic &fdenom,
    const Symbolic &x)
{
       list<Equations> eq;
       list<Equations>::iterator i;
       UniqueSymbol anum, bnum;
       Symbolic integral_sol;
       double anump, bnump, ap, bp, cp;
       ap = fdenom.coeff(x,2);
       bp = fdenom.coeff(x,1);
       cp = fdenom.coeff(x,0);

       ...
       ...

       if (fnum.coeff(x,2)==0 && fnum.coeff(x,1)!=0 && fnum.coeff(x,0)==0
            && fdenom.coeff(x,2)!=0 && fdenom.coeff(x,1)!=0 && fdenom.
           coeff(x,0)!=0 ) // for (a1x)/(ax^2+bx+c)
       {
              eq = ( (bnum*x)).match(fnum, (bnum,cnum));
              for(i=eq.begin(); i!=eq.end(); ++i)
              try {
                     Symbolic a1 = rhs(*i, bnum);
                     bnump= a1;
              } catch(const SymbolicError &se) {}

              double D = (bp*sqrt(bp*bp-4*ap*cp))/(2*ap*(4*ap*cp-(bp*bp
                  ))) ;
              integral_sol = bnump*( (-D + (1/(2*ap)))*ln(x+(-4*ap*cp
                  *(-D + 1/(2*ap)) +bp*bp*(-D+ 1/(2*ap)) + 2*cp)/(bp))
                  + (D + (1/(2*ap)))*ln(x+(-4*ap*cp*(D + 1/(2*ap)) +bp*
                  bp*(D+ 1/(2*ap)) + 2*cp)/(bp)) );
       }

       ...

       return integral_sol;
}
       ...
       ...
```

Code 8: *fraction level 2 integral default type*

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{a_1 x}{ax^2 + bx}$$

$a_1, a,$ and b are constants.

The integral will be

$$\int \frac{a_1 x}{ax^2 + bx}\, dx = \frac{a_1 \ln(ax + b)}{a} + C \tag{2.7}$$

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{a_1 x}{ax^2 + c}$$

a_1, a and c are constants.

The integral will be

$$\int \frac{a_1 x}{ax^2 + c}\, dx = \frac{a_1 \ln(ax^2 + c)}{2a} + C \tag{2.8}$$

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{a_1 x}{bx + c}$$

a_1, b, and c are constants.

The integral will be

$$\int \frac{a_1 x}{bx + c}\, dx = a_1 \left(\frac{x}{b} - \frac{c \ln(bx + c)}{b^2} \right) + C \tag{2.9}$$

- Fraction Level 3

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{a_1 x + b_1}{ax^2 + bx + c}$$

a_1, b_1, a, b and c are constants.

The integral will be

$$\begin{aligned}\int \frac{a_1 x + b_1}{ax^2 + bx + c}\, dx = &\left(\frac{a_1}{2a} - \frac{(2ab_1 - a_1 b)\sqrt{b^2 - 4ac}}{2a(4ac - b^2)} \right) * \\ &\ln \left(x + \frac{4ac \left(\frac{a_1}{2a} - \frac{(2ab_1 - a_1 b)\sqrt{b^2 - 4ac}}{2a(4ac - b^2)} \right) - 2a_1 c - b^2 \left(\frac{a_1}{2a} - \frac{(2ab_1 - a_1 b)\sqrt{b^2 - 4ac}}{2a(4ac - b^2)} \right) + bb_1}{2ab_1 - a_1 b} \right) \\ &+ \left(\frac{a_1}{2a} + \frac{(2ab_1 - a_1 b)\sqrt{b^2 - 4ac}}{2a(4ac - b^2)} \right) * \\ &\ln \left(x + \frac{4ac \left(\frac{a_1}{2a} + \frac{(2ab_1 - a_1 b)\sqrt{b^2 - 4ac}}{2a(4ac - b^2)} \right) - 2a_1 c - b^2 \left(\frac{a_1}{2a} + \frac{(2ab_1 - a_1 b)\sqrt{b^2 - 4ac}}{2a(4ac - b^2)} \right) + bb_1}{2ab_1 - a_1 b} \right) + C\end{aligned} \tag{2.10}$$

The codes for the SymIntegration to handle this integral is located in **src/integrate.cpp**, instead of **integrate** we create a new function **fraction integrate** to handle this type of integral.

```
...

Symbolic fractionintegrate(const Symbolic &fnum, const Symbolic &fdenom,
    const Symbolic &x)
{
       list<Equations> eq;
       list<Equations>::iterator i;
       UniqueSymbol anum, bnum;
       Symbolic integral_sol;
       double anump, bnump, ap, bp, cp;
       ap = fdenom.coeff(x,2);
       bp = fdenom.coeff(x,1);
       cp = fdenom.coeff(x,0);

       ...

       if (fnum.coeff(x,2)==0 && fnum.coeff(x,1)!=0 && fnum.coeff(x,0)!=0
            && fdenom.coeff(x,2)!=0 && fdenom.coeff(x,1)!=0 && fdenom.
           coeff(x,0)!=0 ) // for (a1x+b1)/(ax^2+bx+c)
       {
              eq = ( (bnum*x+cnum)).match(fnum, (bnum,cnum));
              for(i=eq.begin(); i!=eq.end(); ++i)
              try {
                     Symbolic a1 = rhs(*i, bnum), b1 = rhs(*i, cnum);
                     bnump= a1;
                     cnump = b1;
              } catch(const SymbolicError &se) {}

              double D = (2*ap*cnump - bnump*bp)*sqrt(-4*ap*cp+(bp*bp))
                  /(2*ap*(4*ap*cp-(bp*bp)));
              double denom1 = 2*ap*cnump - bnump*bp;

              integral_sol = ((bnump/(2*ap)) - D)*ln(x + (4*ap*cp*((
                  bnump/(2*ap))-D) - 2*bnump*cp -(bp*bp)*(bnump/(2*ap)
                   - D) + bp*cnump)/(denom1)) + ((bnump/(2*ap)) + D)*ln(
                  x + (4*ap*cp*((bnump/(2*ap))+D) - 2*bnump*cp -(bp*bp)
                  *(bnump/(2*ap) + D) + bp*cnump)/(denom1)) ;
       }

       ...

       return integral_sol;
}

...
```

Code 9: *fraction level 3 integral default type showstringspaces*

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{a_1 x + b_1}{ax^2 + bx}$$

a_1, b_1, a, and b are constants.

The integral will be

$$\int \frac{a_1 x + b_1}{ax^2 + bx}\, dx = \frac{b_1 \ln(x)}{b} - \frac{(ab_1 - a_1 b) \ln\left(x + \frac{bb_1 + \frac{b(ab_1 - a_1 b)}{a}}{2ab_1 - a_1 b}\right)}{ab} + C \tag{2.11}$$

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{a_1 x + b_1}{ax^2 + c}$$

a_1, b_1, a, and c are constants.

The integral will be

$$\begin{aligned} \int \frac{a_1 x + b_1}{ax^2 + c}\, dx = &\left(\frac{a_1}{2a} - \frac{b_1\sqrt{-a^3 c}}{2a^2 c}\right) * \\ &\ln\left(x + \frac{2ac\left(\frac{a_1}{2a} - \frac{b_1\sqrt{-a^3 c}}{2a^2 c}\right) - a_1 c}{ab_1}\right) \\ &+ \left(\frac{a_1}{2a} + \frac{b_1\sqrt{-a^3 c}}{2a^2 c}\right) * \\ &\ln\left(x + \frac{2ac\left(\frac{a_1}{2a} + \frac{b_1\sqrt{-a^3 c}}{2a^2 c}\right) - a_1 c}{ab_1}\right) + C \end{aligned} \tag{2.12}$$

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{a_1 x + b_1}{bx + c}$$

a_1, b_1, b, and c are constants.

The integral will be

$$\int \frac{a_1 x + b_1}{bx + c}\, dx = \frac{a_1 x}{b} - \frac{(a_1 c - bb_1) \ln(bx = c)}{b^2} + C \tag{2.13}$$

- Fraction Level 4

[SI*] Suppose we have this function that we want to integrate toward x.

$$f(x) = \frac{a_1x^2 + b_1x + c_1}{ax^2 + bx + c}$$

a_1, b_1, c_1, a, b and c are constants.

The integral will be

$$\begin{aligned}\int \frac{a_1x^2 + b_1x + c_1}{ax^2 + bx + c}\, dx = &\left(\frac{ab_1 - a_1b}{2a^2} - \frac{(2a^2c_1 - 2aa_1c - abb_1 + a_1b^2)\sqrt{b^2 - 4ac}}{2a^2(4ac - b^2)}\right) * \\ &\ln\Bigg[x + \frac{4a^2c\left(\frac{ab_1 - a_1b}{2a^2} - \frac{(2a^2c_1 - 2aa_1c - abb_1 + a_1b^2)\sqrt{b^2 - 4ac}}{2a^2(4ac - b^2)}\right)}{2a^2c_1 - 2aa_1c - abb_1 + a_1b^2} \\ &- \frac{ab^2\left(\frac{ab_1 - a_1b}{2a^2} - \frac{(2a^2c_1 - 2aa_1c - abb_1 + a_1b^2)\sqrt{b^2 - 4ac}}{2a^2(4ac - b^2)}\right) + abc_1 - 2ab_1c + a_1bc}{2a^2c_1 - 2aa_1c - abb_1 + a_1b^2}\Bigg] \\ &+ \left(\frac{ab_1 - a_1b}{2a^2} + \frac{(2a^2c_1 - 2aa_1c - abb_1 + a_1b^2)\sqrt{b^2 - 4ac}}{2a^2(4ac - b^2)}\right) * \\ &\ln\Bigg[x + \frac{4a^2c\left(\frac{ab_1 - a_1b}{2a^2} + \frac{(2a^2c_1 - 2aa_1c - abb_1 + a_1b^2)\sqrt{b^2 - 4ac}}{2a^2(4ac - b^2)}\right)}{2a^2c_1 - 2aa_1c - abb_1 + a_1b^2} \\ &- \frac{ab^2\left(\frac{ab_1 - a_1b}{2a^2} + \frac{(2a^2c_1 - 2aa_1c - abb_1 + a_1b^2)\sqrt{b^2 - 4ac}}{2a^2(4ac - b^2)}\right) + abc_1 - 2ab_1c + a_1bc}{2a^2c_1 - 2aa_1c - abb_1 + a_1b^2}\Bigg] \\ &+ \frac{a_1x}{a} + C\end{aligned} \tag{2.14}$$

The codes for the SymIntegration to handle this integral is located in **src/integrate.cpp**, we use a new function **fraction integrate** to handle this type of integral.

```
...

Symbolic fractionintegrate(const Symbolic &fnum, const Symbolic &fdenom,
    const Symbolic &x)
{
       list<Equations> eq;
       list<Equations>::iterator i;
       UniqueSymbol anum, bnum, cnum;
       Symbolic integral_sol;
       double anump, bnump, cnump, ap, bp, cp;
       ap = fdenom.coeff(x,2);
       bp = fdenom.coeff(x,1);
       cp = fdenom.coeff(x,0);

       if (fnum.coeff(x,2)!=0 && fnum.coeff(x,1)!=0 && fnum.coeff(x,0)!=0
            && fdenom.coeff(x,2)!=0 && fdenom.coeff(x,1)!=0 && fdenom.
           coeff(x,0)!=0 ) // for (a1x^2 + b1x + c1)/(ax^2+bx+c)
```

```
      {
            eq = ( (anum*x*x+bnum*x+cnum)).match(fnum, (anum,bnum,cnum)
                );
            for(i=eq.begin(); i!=eq.end(); ++i)
            try {
                Symbolic a1 = rhs(*i, anum), b1 = rhs(*i, bnum), c1
                    = rhs(*i, cnum);
                anump= a1;
                bnump= b1;
                cnump = c1;
            } catch(const SymbolicError &se) {}

            double D = sqrt(bp*bp-(4*ap*cp))*(2*ap*ap*cnump-2*ap*
                anump*cp-ap*bp*bnump+anump*bp*bp)/(2*ap*ap*(4*ap*cp-(
                bp*bp))) ;

            integral_sol = ((ap*bnump-anump*bp)/(2*ap*ap) - D)*ln(x
                +(4*ap*ap*cp*((ap*bnump-anump*bp)/(2*ap*ap) - D) -
                ap*bp*bp*((ap*bnump-anump*bp)/(2*ap*ap) - D) + ap*bp*
                cnump-2*ap*bnump*cp+anump*bp*cp) / (2*ap*ap*cnump-2*
                ap*anump*cp-ap*bp*bnump+anump*bp*bp)) + ((ap*bnump-
                anump*bp)/(2*ap*ap) + D)*ln(x+(4*ap*ap*cp*((ap*bnump-
                anump*bp)/(2*ap*ap) + D) - ap*bp*bp*((ap*bnump-anump*
                bp)/(2*ap*ap) + D) + ap*bp*cnump-2*ap*bnump*cp+anump*
                bp*cp) / (2*ap*ap*cnump-2*ap*anump*cp-ap*bp*bnump+
                anump*bp*bp)) + (anump*x)/ap;
      }

...
...
```

Code 10: *fraction level 4 integral default type*

[SI*] The key to solve integral of fraction integral is to use the method of partial fraction decomposition, which allows us to decompose rational functions into sums of simpler, more easily integrated rational functions, for example if we have

$$\int \frac{P_n(x)}{P_m(x)}$$

with $P_n(x)$ is a polynomial with x as the independent variable and highest order of n, $P_m(x)$ is a polynomial with x as the independent variable and highest order of m. The partial fraction decomposition can be applied to a rational function

$$\frac{P_n(x)}{P_m(x)}$$

only if $n < m$.

For the case where $n \geq m$, it can be tricked by first we must perform long division to rewrite the quotient into a rational function that can fit the bill to be performed the

partial fraction decomposition

$$\frac{P_n(x)}{P_m(x)} = B(x) + \frac{P_c(x)}{P_d(x)}$$

where $c < d$.

From the previous part we can see that a solid formula to compute the fraction integral can be very long enough, with $m = 2$, it comes from deriving and generalizing this method of partial fraction decomposition by changing the constants into coefficients like a_1, b_1, c_1, a, b, c.

- List of Solvable Fraction Integral
  We will list all the fractions that can be solved with **fractionintegrate** function in SymInte-

gration here, they are:

$$\int \frac{1}{1+x^2}dx = \operatorname{atan}(x)$$

$$\int \frac{ax}{1+x^2}dx = \frac{a\ln(x^2+1)}{2}$$

$$\int \frac{1}{ax^2+bx+c}\,dx = -\sqrt{\frac{-1}{4ac-b^2}}\ln\left(x+\frac{-4ac\sqrt{\frac{-1}{4ac-b^2}}+b^2\sqrt{\frac{-1}{4ac-b^2}}+b}{2a}\right) + \sqrt{\frac{-1}{4ac-b^2}}\ln\left(x+\frac{4ac\sqrt{\frac{-1}{4ac-b^2}}-b^2\sqrt{\frac{-1}{4ac-b^2}}+b}{2a}\right)+C$$

$$\int \frac{a_1x}{ax^2+bx+c}\,dx = a_1\Big[\left(-\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)}+\frac{1}{2a}\right)* \ln\left(x+\frac{-4ac\left(-\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)}+\frac{1}{2a}\right)+b^2\left(-\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)}+\frac{1}{2a}\right)+2c}{b}\right) + \left(\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)}+\frac{1}{2a}\right)* \ln\left(x+\frac{-4ac\left(\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)}+\frac{1}{2a}\right)+b^2\left(\frac{b\sqrt{b^2-4ac}}{2a(4ac-b^2)}+\frac{1}{2a}\right)+2c}{b}\right)\Big]+C$$

$$\int \frac{a_1x+b_1}{ax^2+bx+c}\,dx = \left(\frac{a_1}{2a}-\frac{(2ab_1-a_1b)\sqrt{b^2-4ac}}{2a(4ac-b^2)}\right)* \ln\left(x+\frac{4ac\left(\frac{a_1}{2a}-\frac{(2ab_1-a_1b)\sqrt{b^2-4ac}}{2a(4ac-b^2)}\right)-2a_1c-b^2\left(\frac{a_1}{2a}-\frac{(2ab_1-a_1b)\sqrt{b^2-4ac}}{2a(4ac-b^2)}\right)+bb_1}{2ab_1-a_1b}\right) + \left(\frac{a_1}{2a}+\frac{(2ab_1-a_1b)\sqrt{b^2-4ac}}{2a(4ac-b^2)}\right)* \ln\left(x+\frac{4ac\left(\frac{a_1}{2a}+\frac{(2ab_1-a_1b)\sqrt{b^2-4ac}}{2a(4ac-b^2)}\right)-2a_1c-b^2\left(\frac{a_1}{2a}+\frac{(2ab_1-a_1b)\sqrt{b^2-4ac}}{2a(4ac-b^2)}\right)+bb_1}{2ab_1-a_1b}\right)+C$$

$$\int \frac{a_1x^2+b_1x+c_1}{ax^2+bx+c}\,dx = \left(\frac{ab_1-a_1b}{2a^2} - \frac{(2a^2c_1-2aa_1c-abb_1+a_1b^2)\sqrt{b^2-4ac}}{2a^2(4ac-b^2)}\right) *$$
$$\ln\left[x + \frac{4a^2c\left(\frac{ab_1-a_1b}{2a^2} - \frac{(2a^2c_1-2aa_1c-abb_1+a_1b^2)\sqrt{b^2-4ac}}{2a^2(4ac-b^2)}\right)}{2a^2c_1-2aa_1c-abb_1+a_1b^2}\right.$$
$$\left. - \frac{ab^2\left(\frac{ab_1-a_1b}{2a^2} - \frac{(2a^2c_1-2aa_1c-abb_1+a_1b^2)\sqrt{b^2-4ac}}{2a^2(4ac-b^2)}\right) + abc_1 - 2ab_1c + a_1bc}{2a^2c_1-2aa_1c-abb_1+a_1b^2}\right]$$
$$+ \left(\frac{ab_1-a_1b}{2a^2} + \frac{(2a^2c_1-2aa_1c-abb_1+a_1b^2)\sqrt{b^2-4ac}}{2a^2(4ac-b^2)}\right) *$$
$$\ln\left[x + \frac{4a^2c\left(\frac{ab_1-a_1b}{2a^2} + \frac{(2a^2c_1-2aa_1c-abb_1+a_1b^2)\sqrt{b^2-4ac}}{2a^2(4ac-b^2)}\right)}{2a^2c_1-2aa_1c-abb_1+a_1b^2}\right.$$
$$\left. - \frac{ab^2\left(\frac{ab_1-a_1b}{2a^2} + \frac{(2a^2c_1-2aa_1c-abb_1+a_1b^2)\sqrt{b^2-4ac}}{2a^2(4ac-b^2)}\right) + abc_1 - 2ab_1c + a_1bc}{2a^2c_1-2aa_1c-abb_1+a_1b^2}\right]$$
$$+ \frac{a_1x}{a} + C$$

If the denominator has determinant of 0 / $b^2 = 4ac$ then

$$\int \frac{b_1x+c_1}{ax^2+bx+c}\,dx = \frac{-ac_1+b_1c}{a^3x+a^2c} + \frac{b_1\ln(ax+c)}{a^2}$$

VI. Trigonometry Integral

- Trigonometry Level 1

[SI*] We will want to compute integral of trigonometry at the basic level like these:

$$\int \sin(x)\ dx = -\cos(x) + C$$
$$\int \sin(3x+5)\ dx = -\frac{1}{3}\cos(3x+5) + C$$
$$\int \cos(x)\ dx = \sin(x) + C$$
$$\int \cos(3x+5)\ dx = \frac{1}{3}\sin(3x+5) + C$$
$$\int \tan(x)\ dx = -\ln(\cos(x)) + C$$
$$\int \tan(3x+5)\ dx = \frac{1}{6}\ln(\tan^2(3x+5)+1) + C$$
$$\int \cot(x)\ dx = \ln(\sin(x)) + C$$
$$\int \cot(3x+5)\ dx = -\frac{1}{6}\ln(\tan^2(3x+5)+1) + \frac{1}{3}\ln(\tan(3x+5)) + C$$
$$\int \sec(x)\ dx = -\frac{\ln(\cos(x))}{\sin(x)} + C$$
$$\int \sec(3x+5)\ dx = -\frac{1}{3}\frac{\ln(\cos(3x+5))}{\sin(3x+5)}$$
$$\int \csc(x)\ dx = \frac{\ln(\sin(x))}{\cos(x)} + C$$
$$\int \csc(3x+5)\ dx = \frac{1}{3}\frac{\ln(\sin(3x+5))}{\cos(3x+5)}$$

or in general form

$$\int \sin(ax+b)\ dx = -\frac{1}{a}\cos(ax+b) + C$$
$$\int \cos(ax+b)\ dx = \frac{1}{a}\sin(ax+b) + C$$
$$\int \tan(ax+b)\ dx = \frac{1}{2a}\ln(\tan^2(ax+b)+1) + C$$
$$\int \cot(ax+b)\ dx = -\frac{1}{2a}\ln(\tan^2(ax+b)+1) + \frac{1}{a}\ln(\tan(ax+b)) + C$$
$$\int \sec(ax+b)\ dx = -\frac{1}{a}\frac{\ln(\cos(ax+b))}{\sin(ax+b)}$$
$$\int \csc(ax+b)\ dx = \frac{1}{a}\frac{\ln(\sin(ax+b))}{\cos(ax+b)}$$

At first, the raw codes from SymbolicC++ 3.35 cannot perform that computation, in the **src/functions.cpp** there are implementations for sin, cos, tan but they are imperfect and if the case is not covered it will has an output of **return Integral(*this,s)**, which mean the integral

cannot be computed.

So we correct it and put the formula so SymIntegration can compute the integral problems above, which we call them as Level 1 trigonometry problems.

```
...
...

Symbolic Tan::integrate(const Symbolic &s) const
{
       const Symbolic &x = parameters.front();
       if(x == s) return -ln(cos(x)) * (1 / parameters.front().
           df(s));
       if(df(s) == 0) return *this * s;
       return ln( tan(parameters.front()) * tan(parameters.
           front()) + 1) * ( 1 / (2*parameters.front().df(s)) )
            ;
}

...
```

Code 11: *Implementation of integral for tangent function*

README.md | functions.cpp | main.cpp

```
170
171 Symbolic Acos::integrate(const Symbolic &s) const
172 {
173  const Symbolic &x = parameters.front();
174  if(x == s) return x*acos(x) - sqrt(1-x*x) ;
175  if(df(s) == 0) return *this * s;
176  return ( - sqrt(1-(parameters.front())*(parameters.front()) ) /  parameters.front().df(s) ) + ( parameters.front()*acos(parameters.front()) ,
177 }
178
179
180 ////////////////////////////////
181 // Implementation of Tan      //
182 ////////////////////////////////
183
184 Tan::Tan(const Tan &s) : Symbol(s) {}
185
186 Tan::Tan(const Symbolic &s) : Symbol(Symbol("tan")[s]) {}
187
188 Simplified Tan::simplify() const
189 {
190  const Symbolic &s = parameters.front().simplify();
191  if(s == 0) return Number<int>(0);
192  if(s.type() == typeid(Product))
193  {
194   CastPtr<const Product> p(s);
195   if(p->factors.front() == -1) return -Tan(-s);
196  }
197  if(s.type() == typeid(Numeric) &&
198     Number<void>(s).numerictype() == typeid(double))
199   return Number<double>(tan(CastPtr<const Number<double> >(s)->n));
200  return *this;
201 }
202
203 Symbolic Tan::df(const Symbolic &s) const
204 { return tan(parameters.front()) * tan(parameters.front()) * parameters.front().df(s) + parameters.front().df(s) ; }
205
206 Symbolic Tan::integrate(const Symbolic &s) const
207 {
208  const Symbolic &x = parameters.front();
209  if(x == s) return -ln(cos(x)) * (1 / parameters.front().df(s));
210  if(df(s) == 0) return *this * s;
211  return ln( tan(parameters.front()) * tan(parameters.front()) + 1) * ( 1 / (2*parameters.front().df(s)) ) ;
212 }
213
```

Figure 2.1: *The implementation of tangent function to compute its derivative and integral in* ***src/functions.cpp****.*

- Trigonometry Level 2

  [SI*] The trigonometry integrals that we consider as level 2 are

$$\int \sin\ mx\ \cos\ nx\ dx$$
$$\int \cos\ mx\ \cos\ nx\ dx$$
$$\int \sin\ mx\ \sin\ nx\ dx$$

  Integrals of this type occurs in many physics and engineering applications. By hands we can use the product identitites to handle these integrals.

- Trigonometry Level 3

[SI*] The trigonometry integrals that we consider as level 3 are

$$\int \sin^n x\ dx$$
$$\int \cos^n x\ dx$$
$$\int \tan^n x\ dx$$
$$\int \sec^n x\ dx$$
$$\int \csc^n x\ dx$$
$$\int \cot^n x\ dx$$

We happen to check on Wolfram Alpha for these indefinite integrals. The Wolfram Alpha gives result of these integrals in hypergeometric function term, for example:

$$\int \sin^n\ x\ dx = -\cos(x)\sin^{n+1}(x)\sin^{-n-1}(x)\ {}_2F_1(\frac{1}{2}, \frac{1-n}{2}; \frac{3}{2}; \cos^2(x))$$

For the record, hypergeometric function ${}_2F_1(\frac{1}{2}, \frac{1-n}{2}; \frac{3}{2}; \cos^2(x))$ can be computed with SymIntegration, Boost, or even self made C++ codes, but it will produce approximated integral computation, and we tried to use this formulas with hypergeometric function, it compute very slow.

So we refer to the reduction formula to compute these trigonometry integral level 3.

$$\begin{aligned}
\int \sin^n x\ dx &= -\frac{1}{n}\sin^{n-1}(x)\ \cos(x) + \frac{n-1}{n}\int \sin^{n-2}(x)\ dx \\
\int \cos^n x\ dx &= \frac{1}{n}\cos^{n-1}(x)\ \sin(x) + \frac{n-1}{n}\int \cos^{n-2}(x)\ dx \\
\int \tan^n x\ dx &= \frac{\tan^{n-1}(x)}{n-1} - \int \tan^{n-2}(x)\ dx \\
\int \sec^n x\ dx &= \frac{\sec^{n-2}(x)\ \tan(x)}{n-1} + \frac{n-2}{n-1}\int \sec^{n-2}(x)\ dx \\
\int \csc^n x\ dx &= \frac{-\csc^{n-2}(x)\ \cot(x)}{n-1} + \frac{n-2}{n-1}\int \csc^{n-2}(x)\ dx \\
\int \cot^n x\ dx &= -\frac{1}{n-1}\cot^{n-1}(x) - \int \cot^{n-2}(x)\ dx
\end{aligned} \tag{2.15}$$

With this reduction formula we can use recursive method, with **for** loop technique in C++ after we find the pattern, we can code it easily. The computation time with reduction formula and this recursive method is faster than using Hypergeometric function.

[SI*] We will examine each integral and learn about their patterns, one by one. Starting with $\int \sin^n x\ dx$

i. $\int \sin^n x\ dx$

These are the formula of $\int \sin^n x\ dx$ with $n = 2$ to $n = 8$.

$$\int \sin^2 x\ dx = \frac{x}{2} - \frac{\sin\ x\ \cos\ x}{2}$$
$$\int \sin^3 x\ dx = \frac{\cos^3 x}{3} - \cos\ x$$
$$\int \sin^4 x\ dx = \frac{3x}{8} - \frac{\sin^3\ x\ \cos\ x}{4} - \frac{3\sin\ x\ \cos\ x}{8}$$
$$\int \sin^5 x\ dx = -\frac{\cos^5 x}{5} + \frac{2\cos^3 x}{3} - \cos\ x$$
$$\int \sin^6 x\ dx = \frac{5x}{16} - \frac{\sin^5\ x\ \cos\ x}{6} - \frac{5\sin^3\ x\ \cos\ x}{24} - \frac{5\sin\ x\ \cos\ x}{16}$$
$$\int \sin^7 x\ dx = \frac{\cos^7 x}{7} - \frac{3\cos^5 x}{5} + \frac{3\cos^3 x}{3} - \cos\ x$$
$$\int \sin^8 x\ dx = \frac{35x}{128} - \frac{\sin^7\ x\ \cos\ x}{8} - \frac{7\sin^5\ x\ \cos\ x}{48} - \frac{35\sin^3\ x\ \cos\ x}{192} - \frac{35\sin\ x\ \cos\ x}{128}$$

From the first 8 terms we will get this intuition that the pattern occur for even power and odd power, so it will divide into two cases, in C++ we can use the statement (**n % 2 == 0**) in the **if** statement to handle this.

The manual way to solve this can be traced back to the reduction formula, for example if you want to compute $\int \sin^8 x\ dx$, then you start in a decreasing **for** loop with $i = n, i >= 2, i = i - 2$.

Case 1: For $\int \sin^n x\ dx$ with even n
The numerator coefficient and denominator coefficient are making patterns that can be decoded.

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
const Symbolic &a = parameters.front();
const Symbolic &b = parameters.back();
int bpower = parameters.back().coeff(s,0);
if(a.type() == typeid(Sin))
{
       if(b.df(s) == 0 && b!=0 && bpower % 2 == 0) // even power
           case
       {
              Symbolic c = 1;
              Symbolic d = parameters.back().coeff(s,0);
              Symbolic integral;
              for(int i = bpower ; i >= 2 ; i = i-2)
              {
                     integral -= c*cos(s)*((sin(s))^(i-1))/(d) ;
```

```
                        d = d*(i-2);
                        c = c*(i-1);
                        if (i==4)
                        {
                                integral += ( (c*s)/(d) );
                        }
                }
        return integral;
        }
}
...
```

Code 12: *level 3 integral for sine even case*

As you can see the logic for the code above: we multiply the numerator with c that is the odd factorial ($1 \times 3 \times 5 \times \cdots \times n-1$), and multiply the denominator with the even factorial ($2 \times 4 \times 6 \times \cdots \times n$).

The looping starts from the **bpower**, the variable name to represents n, the power for the integral of $\sin^n x$, then descending to 2, and decreasing by 2.

Notice that the numerator and denominator coefficient for x is the same as $\sin\ x\ \cos\ x$, so we only need to compute the coefficient for $\sin\ x\ \cos\ x$ and then copy it as the coefficient for x.

We start to compute the coefficient with the highest power, for example we want to

compute $\int \sin^8 x \, dx$, then we start with $i = 8$ and we will have:

$$
\begin{aligned}
i &= 8 \\
c &= 1 \\
d &= 8 \\
\text{integral} &= -\frac{c * \cos(s) * \sin^{i-1}(s)}{d} \\
d &= d * (i-2) \\
c &= c * (i-1) \\
i &= i-2 \\
&* * * * * * * * * * * * \\
i &= 6 \\
c &= 7 \\
d &= 48 \\
\text{integral} &= -\frac{c * \cos(s) * \sin^{i-1}(s)}{d} \\
d &= d * (i-2) \\
c &= c * (i-1) \\
i &= i-2 \\
&* * * * * * * * * * * * \\
i &= 4 \\
c &= 35 \\
d &= 192 \\
\text{integral} &= -\frac{c * \cos(s) * \sin^{i-1}(s)}{d} \\
d &= d * (i-2) \\
c &= c * (i-1) \\
i &= i-2 \\
&* * * * * * * * * * * * \\
i &= 2 \\
c &= 105 \\
d &= 384 \\
\text{integral} &= -\frac{c * \cos(s) * \sin^{i-1}(s)}{d} \\
d &= d * (i-2) \\
c &= c * (i-1) \\
i &= i-2
\end{aligned}
$$

We stop when $i = 2$ and we will obtain $\frac{105}{384} = \frac{35}{128}$ as the coefficient for $\sin\ x\ \cos\ x$.

It only takes a lot of papers and pen to be able to decode this patterns. Then some patience and persistence with focus to create the C++ codes.

Case 2: For $\int \sin^n x\ dx$ with odd n
The numerator coefficient and denominator coefficient are making patterns that can be decoded. This time it is easier than the even case, but a little bit tricky with alternating sign and need to use combination formula.

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
...
if(a.type() == typeid(Sin))
{
if(b.df(s) == 0 && b!=0 && bpower % 2 != 0) // odd power case
{
       Symbolic sgn = -1;
       Symbolic integral;
       int j =1;
       int m = bpower-(0.5*(bpower+1));
       for(int i = 1 ; i <= bpower ; i = i+2)
       {
              integral += sgn*combinations(m,j-1)*((cos(s))^(i))
                  /(i);
              sgn = -sgn;
              j = j+1;
       }
       return integral;
}
}
```

Code 13: *level 3 integral for sine odd case*

When I was trying to find the pattern, it is nice to write from the lowest odd power to a certain odd power and see the pattern for the numerator and denominator as the power rises.

$$\int \sin^3 x\ dx = \frac{\cos^3 x}{3} - \cos\ x$$
$$\int \sin^5 x\ dx = -\frac{\cos^5 x}{5} + \frac{2\cos^3 x}{3} - \cos\ x$$
$$\int \sin^7 x\ dx = \frac{\cos^7 x}{7} - \frac{3\cos^5 x}{5} + \frac{3\cos^3 x}{3} - \cos\ x$$
$$\int \sin^9 x\ dx = -\frac{\cos^9 x}{9} + \frac{4\cos^7 x}{7} - \frac{6\cos^5 x}{5} + \frac{4\cos^3 x}{3} - \cos\ x$$

Right from the bat, the denominator has an easy pattern of odd number decreasing from the n to 1, the denominator has the same coefficient as the power for the $\cos(x)$. So it is clear already.

The problem now, is the numerator, we have this kind of pattern:

$$\begin{array}{ccccccccc} & & & 1 & & 1 & & & \\ & & 1 & & 2 & & 1 & & \\ & 1 & & 3 & & 3 & & 1 & \\ 1 & & 4 & & 6 & & 4 & & 1 \end{array}$$

It is the infamous Pascal' triangle. Which can be derived from the combination formula:

$$\begin{aligned} {}_0C_0 &= 1 \\ {}_1C_0 &= 1 \\ {}_1C_1 &= 1 \\ {}_2C_0 &= 1 \\ {}_2C_1 &= 2 \\ {}_2C_2 &= 1 \\ {}_3C_0 &= 1 \\ {}_3C_1 &= 3 \\ {}_3C_2 &= 3 \\ {}_3C_3 &= 1 \\ {}_4C_0 &= 1 \\ {}_4C_1 &= 4 \\ {}_4C_2 &= 6 \\ {}_4C_3 &= 4 \\ {}_4C_4 &= 1 \end{aligned}$$

This explains the usage of **combinations** function in the C++ code.

ii. $\int \cos^n x \, dx$

These are the formula of $\int \cos^n x \, dx$ with $n = 2$ to $n = 8$.

$$\begin{aligned} \int \cos^2 x \, dx &= \frac{x}{2} + \frac{\sin\ x\ \cos\ x}{2} \\ \int \cos^3 x \, dx &= -\frac{\sin^3 x}{3} + \sin\ x \\ \int \cos^4 x \, dx &= \frac{3x}{8} + \frac{\sin\ x\ \cos^3\ x}{4} + \frac{3\sin\ x\ \cos\ x}{8} \\ \int \cos^5 x \, dx &= \frac{\sin^5 x}{5} - \frac{2\sin^3 x}{3} + \sin\ x \\ \int \cos^6 x \, dx &= \frac{5x}{16} + \frac{\sin\ x\ \cos^5\ x}{6} + \frac{5\sin\ x\ \cos^3\ x}{24} + \frac{5\sin\ x\ \cos\ x}{16} \\ \int \cos^7 x \, dx &= -\frac{\sin^7 x}{7} + \frac{3\sin^5 x}{5} - \frac{3\sin^3 x}{3} + \sin\ x \\ \int \cos^8 x \, dx &= \frac{35x}{128} + \frac{\sin\ x\ \cos^7\ x}{8} + \frac{7\sin\ x\ \cos^5\ x}{48} + \frac{35\sin\ x\ \cos^3\ x}{192} + \frac{35\sin\ x\ \cos\ x}{128} \end{aligned}$$

Right after we observe the equations above, it is really familiar, the pattern for the numerator and denominator will be like the sine counterpart, thus we only need to exchange the sign, what is $-$ become $+$ and the other way around, and also exchanging sine into cosine and cosine into sine.

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
const Symbolic &a = parameters.front();
const Symbolic &b = parameters.back();
int bpower = parameters.back().coeff(s,0);
if(a.type() == typeid(Cos))
{
       if(b.df(s) == 0 && b!=0 && bpower % 2 == 0) // even power case
       {
              Symbolic c = 1;
              Symbolic d = parameters.back().coeff(s,0);
              Symbolic integral;
              for(int i = bpower ; i >= 2 ; i = i-2)
              {
                     integral += c*sin(s)*((cos(s))^(i-1))/(d) ;
                     d = d*(i-2);
                     c = c*(i-1);
                     if (i==4)
                     {
                            integral += ( (c*s)/(d) );
                     }
              }
              return integral;
       }
}
...
```

Code 14: *level 3 integral for cosine even case*

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
       const Symbolic &a = parameters.front();
       const Symbolic &b = parameters.back();
       int bpower = parameters.back().coeff(s,0);
       if(a.type() == typeid(Cos))
       {
              if(b.df(s) == 0 && b!=0 && bpower % 2 != 0) // odd power
                   case
              {
              Symbolic sgn = 1;
```

```
                Symbolic integral;
                int j =1;
                int m = bpower—(0.5*(bpower+1));
                for(int i = 1 ; i <= bpower ; i = i+2)
                {
                integral += sgn*combinations(m,j—1)*((sin(s))^(i))/(i);
                sgn = —sgn;
                j = j+1;
                }
                return integral;
                }
        }
        ...
```

Code 15: *level 3 integral for cosine odd case*

iii. $\int \sec^n x\, dx$

These are the formula of $\int \sec^n x\, dx$ with $n = 2$ to $n = 8$.

$$\int \sec^2 x\, dx = \frac{\sin\ x}{\cos\ x}$$
$$\int \sec^3 x\, dx = -\frac{\log(\sin\ (x) - 1)}{4} + \frac{\log(\sin\ (x) + 1)}{4} - \frac{\sin\ x}{2\sin^2 x - 2}$$
$$\int \sec^4 x\, dx = \frac{2\sin\ x}{3\cos\ x} + \frac{\sin\ x}{3\cos^3\ x}$$
$$\int \sec^5 x\, dx = -\frac{3\sin^3\ x - 5\sin\ x}{8\sin^4\ x - 16\sin^2\ x + 8} - \frac{3\log(\sin\ (x) - 1)}{16} + \frac{3\log(\sin\ (x) + 1)}{16}$$
$$\int \sec^6 x\, dx = \frac{8\sin\ x}{15\cos\ x} + \frac{4\sin\ x}{15\cos^3\ x} + \frac{\sin\ x}{5\cos^5\ x}$$
$$\int \sec^7 x\, dx = \frac{-15\sin^5\ x + 40\sin^3\ x - 33\sin\ x}{48\sin^6\ x - 144\sin^4\ x + 144\sin^2\ x - 48} - \frac{5\log(\sin\ (x) - 1)}{32} + \frac{5\log(\sin\ (x) + 1)}{32}$$
$$\int \sec^8 x\, dx = \frac{16\sin\ x}{35\cos\ x} + \frac{8\sin\ x}{35\cos^3\ x} + \frac{6\sin\ x}{35\cos^5\ x} + \frac{\sin\ x}{7\cos^7\ x}$$

The same case occurs again here, the pattern is repeated every $n + 2$, so there will be the odd power pattern and the even power pattern.

Case 1: For $\int \sec^n x\, dx$ with even n
The numerator coefficient and denominator coefficient are making patterns that can be decoded.

$$\int \sec^2 x\, dx = \frac{\sin\ x}{\cos\ x}$$
$$\int \sec^4 x\, dx = \frac{2\sin\ x}{3\cos\ x} + \frac{\sin\ x}{3\cos^3\ x}$$
$$\int \sec^6 x\, dx = \frac{8\sin\ x}{15\cos\ x} + \frac{4\sin\ x}{15\cos^3\ x} + \frac{\sin\ x}{5\cos^5\ x}$$
$$\int \sec^8 x\, dx = \frac{16\sin\ x}{35\cos\ x} + \frac{8\sin\ x}{35\cos^3\ x} + \frac{6\sin\ x}{35\cos^5\ x} + \frac{\sin\ x}{7\cos^7\ x}$$
$$\int \sec^{10} x\, dx = \frac{128\sin\ x}{315\cos\ x} + \frac{64\sin\ x}{315\cos^3\ x} + \frac{16\sin\ x}{105\cos^5\ x} + \frac{8\sin\ x}{63\cos^7\ x} + \frac{\sin\ x}{9\cos^9\ x}$$

The patterns that can be seen from those equations above are

- You may have misinterpreted it first, but the denominator are all the same, for example:

  $$\int \sec^6 x\, dx = \frac{8\sin\ x}{15\cos\ x} + \frac{4\sin\ x}{15\cos^3\ x} + \frac{\sin\ x}{5\cos^5\ x}$$

  is the same as

  $$\int \sec^6 x\, dx = \frac{8\sin\ x}{15\cos\ x} + \frac{4\sin\ x}{15\cos^3\ x} + \frac{3\sin\ x}{15\cos^5\ x}$$

  Learning from that, the pattern for the numerator can be decoded.

Knowing that the denominator is the same, we only need to find out, how we get the denominator coefficient of 1 for $n = 2$, or of 3 for $n = 4$, or of 15 for $n = 6$.

Turns out, the first intuition is quite correct, it is the factorial odd, we will declare the denominator as variable d_0:

$$d_0 = (n-1) * (n-3) * (n-5) * \cdots * (1)$$

If it is $\int \sec^8 x \, dx$, then the denominator will be

$$d_0 = 7 * 5 * 3 * 1 = 105$$

In the equation for $\int \sec^8 x \, dx$, the denominator is 35, instead of 105, this happens because the numerator is divisible with the denominator till they have no greatest common divisor anymore but 1, it is caused by the computer algebra system that always shows the most simplified version, so you can adjust the equation again to find the raw number for the denominator and the numerator. Making the denominator into the same number is not enough, so we need them in raw number version.

- After we make the denominator to become equal we will have this:

$$\int \sec^2 x \, dx = \frac{\sin\ x}{\cos\ x}$$
$$\int \sec^4 x \, dx = \frac{2\sin\ x}{3\cos\ x} + \frac{\sin\ x}{3\cos^3\ x}$$
$$\int \sec^6 x \, dx = \frac{8\sin\ x}{15\cos\ x} + \frac{4\sin\ x}{15\cos^3\ x} + \frac{3\sin\ x}{15\cos^5\ x}$$
$$\int \sec^8 x \, dx = \frac{16\sin\ x}{35\cos\ x} + \frac{8\sin\ x}{35\cos^3\ x} + \frac{6\sin\ x}{35\cos^5\ x} + \frac{5\sin\ x}{35\cos^7\ x}$$
$$\int \sec^{10} x \, dx = \frac{128\sin\ x}{315\cos\ x} + \frac{64\sin\ x}{315\cos^3\ x} + \frac{48\sin\ x}{315\cos^5\ x} + \frac{40\sin\ x}{315\cos^7\ x} + \frac{35\sin\ x}{315\cos^9\ x}$$

so the pattern of the numerator can be seen like this:

$$\begin{array}{ccccccccc} & & & & 1 & & & & \\ & & & 1 & & 2 & & & \\ & & 3 & & 4 & & 8 & & \\ & 5 & & 6 & & 8 & & 16 & \\ 35 & & 40 & & 48 & & 64 & & 128 \end{array}$$

Notice that we have to change the denominator into its raw version (for $\int \sec^8 x \, dx$ the denominator should be 105 instead of 35, and for $\int \sec^{10} x \, dx$ the denominator should be 945 instead of 315), this is the logic: the pattern can be decoded at its' raw numbers.

Thus

$$\begin{array}{ccccccccc} & & & & 1 & & & & \\ & & & 1 & & 2 & & & \\ & & 3 & & 4 & & 8 & & \\ & 15 & & 18 & & 24 & & 48 & \\ 105 & & 120 & & 144 & & 192 & & 384 \end{array}$$

The first entry for the next iteration is an odd multiplication of the current first entry from before with an increasing odd number. The first iteration will multiplied by 1, the

second will be multiplied by 3, the next iteration will be multiplied by 5, and so on. This odd multiplication will be called as k, While the second entry to the last entry will be multiplied by even multiplication that will be called as l and it is increasing by 2 at every iteration.

$$
\begin{aligned}
&\text{for } i = 1 \rightarrow \mathbf{v[0]} = 1 * k_{i=1} = 1 \\
&\text{for } i = 1 \rightarrow \mathbf{v[1]} = 1 * l_{i=1} = 2 \\
&\text{for } i = 2 \rightarrow \mathbf{v[0]} = \mathbf{v[0]}_{i=1} * k_{i=2} = 1 * 3 = 3 \\
&\text{for } i = 2 \rightarrow \mathbf{v[1]} = \mathbf{v[0]}_{i=1} * l_{i=2} = 1 * 4 = 4 \\
&\text{for } i = 2 \rightarrow \mathbf{v[2]} = \mathbf{v[1]}_{i=1} * l_{i=2} = 2 * 4 = 8 \\
&\text{for } i = 3 \rightarrow \mathbf{v[0]} = \mathbf{v[0]}_{i=2} * k_{i=3} = 3 * 5 = 15 \\
&\text{for } i = 3 \rightarrow \mathbf{v[1]} = \mathbf{v[0]}_{i=2} * l_{i=3} = 3 * 6 = 18 \\
&\text{for } i = 3 \rightarrow \mathbf{v[2]} = \mathbf{v[1]}_{i=2} * l_{i=3} = 4 * 6 = 24 \\
&\text{for } i = 3 \rightarrow \mathbf{v[3]} = \mathbf{v[2]}_{i=2} * l_{i=3} = 8 * 6 = 48
\end{aligned}
$$

Now we already have an idea about the pattern. For the C++ code, this time instead of using vector we will use array that is initialized with a very big size: **int v[999]**. Array is often used to replace vector, since vector has an invalid pointer problem / out of bounds memory access if you are not careful, array can also be used to become a matrix.

In order to determine the coefficient of the numerator with sine function we will use ascending **for** loop, to be exact we will use **for**$(i = 1; i < n - 1; i = i + 2)$.

We will declare variables / constants to help us compute, which are $d_0 = 1, k = 1, l = 2, c = 1$. Their value will be changing in every iteration (c will be used to store the first entry ($\mathbf{v}[0]$) of the previous array, $k = k + 2$ and $l = l + 2$).

We save the first entry of the previous array because it is going to be used to determine the first and second entry of the array at the current iteration.

$$
\begin{aligned}
k &= 1 \\
l &= 2 \\
d_0 &= 1 \\
c &= 1
\end{aligned}
$$

In every iteration we will clear the entries in the array.

So the first **for** loop will be like this:

$$
\begin{gathered}
i = 1 \\
d_0 = 3 \\
l = 4 \\
k = 3 \\
v[0] = 1 \\
v[1] = 2 \\
* * * * * * * * * * * \\
i = 3 \\
d_0 = 15 \\
l = 6 \\
k = 5 \\
v[0] = 3 \\
v[1] = 4 \\
v[2] = 8 \\
* * * * * * * * * * * \\
i = 5 \\
d_0 = 105 \\
l = 8 \\
k = 7 \\
v[0] = 15 \\
v[1] = 18 \\
v[2] = 24 \\
v[3] = 48 \\
* * * * * * * * * * * \\
i = 7 \\
d_0 = 945 \\
l = 10 \\
k = 9 \\
v[0] = 105 \\
v[1] = 120 \\
v[2] = 144 \\
v[3] = 192 \\
v[4] = 384
\end{gathered}
$$

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
       ...
```

```
if(a.type() == typeid(Sec))
{
...

if(b.df(s) == 0 && b!=0 && bpower % 2 == 0) // even power case
{
int v[999];
v[0] = 1;
Symbolic integral;
Symbolic d0 = 1;
int k = 1, l = 2, c=1;
for(int i = 1 ; i < bpower ; i = i+2) // to compute the denominator
{
       d0 *= i;
}
for(int i = 1 ; i < bpower - 1; i = i+2)
{
       c = v[0];
       int arrsec[999]; // make the size of the array as big as
           possible

       for(int j = 0 ; j < i-1 ; j = j+1)
       {
              arrsec[j] = v[j];
       }
       int d;
       for(int j = 1 ; j < i ; j = j+1)
       {
              d = arrsec[j-1];
              v[j] = d*l;
       }

       v[0] = c*k;
       v[1] = c*l;

       k= k + 2;
       l = l + 2;
}

int j_d = 1 ;
for(int i = 1 ; i < (0.5*bpower)+1; i = i+1)
{
       integral += ( v[i-1] * sin(s) ) / ( d0*(cos(s)^(bpower-j_d)) )
            ;
       j_d = j_d+2;
}
return integral;
}
```

```
        }
}
```

Code 16: *level 3 integral for secant even case*

Case 2: For $\int \sec^n x\, dx$ with odd n
Compared to the even case, the odd case for $\int \sec^n x\, dx$ is harder in terms of determining the pattern and then to code it with C++, we even have to use two vectors, the first one to store the coefficient for the numerator, and the second vector is used to store the "middle coefficient", in order to determine the correct coefficient for the numerator.

$$\int \sec^3 x\, dx = -\frac{\log(\sin(x)-1)}{4} + \frac{\log(\sin(x)+1)}{4} - \frac{\sin x}{2\sin^2 x - 2}$$

$$\int \sec^5 x\, dx = -\frac{3\sin^3 x - 5\sin x}{8\sin^4 x - 16\sin^2 x + 8} - \frac{3\log(\sin(x)-1)}{16} + \frac{3\log(\sin(x)+1)}{16}$$

$$\int \sec^7 x\, dx = \frac{-15\sin^5 x + 40\sin^3 x - 33\sin x}{48\sin^6 x - 144\sin^4 x + 144\sin^2 x - 48} - \frac{5\log(\sin(x)-1)}{32} + \frac{5\log(\sin(x)+1)}{32}$$

$$\int \sec^9 x\, dx = -\frac{105\sin^7 x - 385\sin^5 x + 511\sin^3 x - 279\sin x}{384\sin^8 x - 1536\sin^6 x + 2304\sin^4 x - 1536\sin^2 x + 384} - \frac{35\log(\sin(x)-1)}{256} + \frac{35\log(\sin(x)+1)}{256}$$

$$\int \sec^{11} x\, dx = \frac{-315\sin^9 x + 1470\sin^7 x - 2688\sin^5 x + 2370\sin^3 x - 965\sin x}{1280\sin^{10} x - 6400\sin^8 x + 12800\sin^6 x - 12800\sin^4 x + 6400\sin^2 x - 1280} - \frac{63\log(\sin(x)-1)}{512} + \frac{63\log(\sin(x)+1)}{512}$$

If we want to compute it with C++, then we can use the **for** as usual to help us. Then it is either ascending **for** loop or descending **for** loop. As time goes by and flight hours, you will know which one to use.

We can immediately notice some very obvious patterns:

- We have 3 different terms here, 2 are the log terms and the last is the term with numerator that has the sum of alternating sign of sine with odd power and denominator that has the sum of alternating sign of sine with even power.
- The terms with log has denominator of with multiplication of $(2 * i)$ that still store the previous value so we can determine it with $d_1 = d_1 * (2 * i)$. After we finish with the **for** loop, we will multiply again for the last time to obtain d_1 that will be used

$$d_1 = d_1 * (n-1)$$

- Now, we shall see the coefficient for the sum of alternating sign of sine with odd power:

$$1$$

$$3 \quad 5$$

$$15 \quad 40 \quad 33$$

$$105 \quad 385 \quad 511 \quad 279$$

315 1470 2688 2370 965

The first row represents $\int \sec^3 x \, dx$ and the last row represents $\int \sec^{11} x \, dx$. The sign is alternating actually, but it is omitted for better view, so the reader won't be confused. Alternating sign can be attached later on.

In this case, to determine the coefficient of the numerator with odd sine function we will use ascending **for** loop, to be exact we will use **for**$(i = 1; i < \frac{n-1}{2}; i = i + 1)$.

We also need to have extra variables to help compute the coefficient correctly, we give them initial value before all the **for** loop:

$$\begin{aligned} k &= 1 \\ l &= 2 \\ d_0 &= 2 \\ d_1 &= 2 \\ \text{first\_coeff} &= 3 \\ j &= 1 \\ m &= n - \left(\frac{n+1}{2}\right) \\ sgn &= -1 \end{aligned}$$

We are going to use vector **v** to represent the coefficient for the numerator of sine with odd power, the size of the vector is $\frac{n-1}{2}$.

At first, when $i = 1$, we know that $\mathbf{v}[0] = 1$, so how to determine $\mathbf{v}[0]$ and $\mathbf{v}[1]$ for $i = 2$ and continuing the loop to compute for n at higher number?

Go back and see the reduction formula at Equation (2.6) for $\int \sec^n x \, dx$, we will notice that there is $n-2$ and $n-1$ terms at every recursive step / the loop. So we will have to see the multiplication possibilities for the factorial odd: $(n-2) * (n-4) * \cdots * 1$ and the factorial even: $(n-1) * (n-3) * (n-5) * \cdots * 2$. We are focusing on the numerator now, so it is the factorial odd that we will use (the one related to $n-2$ factorial down with difference of 2).

If you take a look again at the triangle above, the trend is increasing then decreasing at the end, it is the typical of Pascal' triangle but we do not use combination here. Here we try step by step from $n = 3$ or $i = 1$:

$$\begin{aligned} n &= 3 \\ i &= 1 \\ \mathbf{v}[0] &= 1 \end{aligned}$$

that will make

$$\int \sec^3 x \, dx = -\frac{\log(\sin(x) - 1)}{4} + \frac{\log(\sin(x) + 1)}{4} - \frac{\mathbf{v}[0] \sin x}{2\sin^2 x - 2}$$

moving on to $n = 5$ and $i = 2$

$$\begin{aligned} n &= 5 \\ i &= 2 \\ \mathbf{v}[0] &= 3 \\ \mathbf{v}[1] &= 5 \end{aligned}$$

$$\int \sec^5 x \, dx = -\frac{\mathbf{v}[0] \sin^3 x - \mathbf{v}[1] \sin x}{8 \sin^4 x - 16 \sin^2 x + 8} - \frac{3 \log(\sin(x) - 1)}{16} + \frac{3 \log(\sin(x) + 1)}{16}$$

Then to $n = 7$ and $i = 3$

$$\begin{aligned} n &= 7 \\ i &= 3 \\ \mathbf{v}[0] &= 15 \\ \mathbf{v}[1] &= 40 \\ \mathbf{v}[2] &= 33 \end{aligned}$$

$$\int \sec^7 x \, dx = \frac{\mathbf{v}[0] \sin^5 x + \mathbf{v}[1] \sin^3 x - \mathbf{v}[2] \sin x}{48 \sin^6 x - 144 \sin^4 x + 144 \sin^2 x - 48} - \frac{5 \log(\sin(x) - 1)}{32} + \frac{5 \log(\sin(x) + 1)}{32}$$

So to be able to create the C++ code to compute this coefficients, when we have the initial condition of $\mathbf{v}[0] = 1$ at $i = 1$, how do we get into $\mathbf{v}[0] = 3, \mathbf{v}[1] = 5$ at $i = 2$ and $\mathbf{v}[0] = 15, \mathbf{v}[1] = 40, \mathbf{v}[2] = 33$ at $i = 3$ and so on correctly?

Watch this, we first define $last\_coeff = \mathbf{v}[j-1]$ with $j = 1, 2, 3, \ldots$ as the last index of

the vector from the previous i and $first\_coeff = 3$, so

$$
\begin{aligned}
n &= 3 \\
i &= 1 \\
\mathbf{v}[0] &= 1 \\
last\_coeff &= \mathbf{v}[0] = 1 \\
k &= 2 \\
&********** \\
n &= 5 \\
i &= 2 \\
\mathbf{v}[0] &= \mathbf{v}[0]_{i=1} * (n-2) = 3 \\
\mathbf{v}[1] &= [last\_coeff * (n-2)] + k = [1*3] + 2 = 5 \\
\mathbf{mc}[0] &= \mathbf{v}[0] + \mathbf{v}[1] = 8 \\
last\_coeff &= \mathbf{v}[1] = 5 \\
k &= 8 \\
&********** \\
n &= 7 \\
i &= 3 \\
\mathbf{v}[0] &= \mathbf{v}[0]_{i=2} * (n-2) = 3*5 = 15 \\
\mathbf{v}[1] &= \mathbf{mc}[0] * (n-2) = 8*5 = 40 \\
\mathbf{v}[2] &= [last\_coeff * (n-2)] + k = [5*5] + 8 = 33 \\
\mathbf{mc}[0] &= \mathbf{v}[0] + \mathbf{v}[1] = 55 \\
\mathbf{mc}[1] &= \mathbf{v}[1] + \mathbf{v}[2] = 73 \\
last\_coeff &= \mathbf{v}[2] = 33 \\
k &= 48 \\
&********** \\
n &= 9 \\
i &= 4 \\
\mathbf{v}[0] &= \mathbf{v}[0]_{i=3} * (n-2) = 15*7 = 105 \\
\mathbf{v}[1] &= \mathbf{mc}[0] * (n-2) = 55*7 = 385 \\
\mathbf{v}[2] &= \mathbf{mc}[1] * (n-2) = 73*7 = 511 \\
\mathbf{v}[3] &= [last\_coeff * (n-2)] + k = [33*7] + 48 = 279 \\
\mathbf{mc}[0] &= \mathbf{v}[0] + \mathbf{v}[1] = 490 \\
\mathbf{mc}[1] &= \mathbf{v}[1] + \mathbf{v}[2] = 896 \\
\mathbf{mc}[2] &= \mathbf{v}[2] + \mathbf{v}[3] = 790 \\
last\_coeff &= \mathbf{v}[2] = 279 \\
k &= 384
\end{aligned}
$$

The $last\_coeff$ and $first\_coeff$ will be used in the C++ codes to represent $n-2$, it is a little bit of a handy trick.

We finally know the pattern and formula to obtain the coefficients for the numerator. It is not an easy one as I spent days to realize how to obtain this.

- The denominator that has the sum of alternating sign of sine with even power has a distinctive pattern that can be seen when you take the greatest common diviser out:

$$8\sin^4 x - 16\sin^2 x + 8 = 8(\sin^4 x - 2\sin^2 x + 1)$$
$$48\sin^6 x - 144\sin^4 x + 144\sin^2 x - 48 = 48(\sin^6 x - 3\sin^4 x + 3\sin^2 x - 1)$$
$$384\sin^8 x - \cdots + \cdots + 384 = 384(\sin^8 x - 4\sin^6 x + 6\sin^4 x - 4\sin^2 x + 1)$$

We are focusing on the denominator now, so it is the factorial even that we will use, with the initial value of $d_0 = 2$ we will have:

$$\text{for } n = 5 \rightarrow \; d_0 = d_0 * (n-1) = d_0 * (2 + 2 * i) = 8$$
$$\text{for } n = 7 \rightarrow \; d_0 = d_0 * (n-1) = d_0 * (2 + 2 * i) = 48$$
$$\text{for } n = 9 \rightarrow \; d_0 = d_0 * (n-1) = d_0 * (2 + 2 * i) = 384$$

From this you can see clearly what to write for the C++ code.

So the first **for** loop will be like this:

$$
\begin{aligned}
i &= 1 \\
d_1 &= 4 \\
d_0 &= 8 \\
mc[0] &= 1 \\
v[0] &= 3 \\
v[1] &= 5 \\
&********** \\
i &= 2 \\
d_1 &= 16 \\
d_0 &= 48 \\
mc[0] &= 8 \\
v[0] &= 15 \\
v[1] &= 40 \\
v[2] &= 33 \\
&********** \\
i &= 3 \\
d_1 &= 96 \\
d_0 &= 384 \\
mc[0] &= 55 \\
mc[1] &= 73 \\
v[0] &= 105 \\
v[1] &= 385 \\
v[2] &= 511 \\
v[3] &= 279 \\
&********** \\
i &= 4 \\
d_1 &= 768 \\
d_0 &= 3840 \\
mc[0] &= 490 \\
mc[1] &= 896 \\
mc[2] &= 790 \\
v[0] &= 945 \\
v[1] &= 4410 \\
v[2] &= 8064 \\
v[3] &= 7110 \\
v[4] &= 2895
\end{aligned}
$$

The next two **for** loops won't be too hard to comprehend, we include the variable *sgn* to alternate the sign. Overall, to be able to detect the pattern out of this took quite some time,

and makes the author realize that integration to the power of n for basic trigonometry is related to sum / series, and also combination and Pascal' triangle. With that knowledge and logic, thus today Computer Algebra System, like SymIntegration or SymPy able to compute $\int \sec^n x \, dx$ for any integer number of n. It shows how beautiful mathematics is.

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
...
       if(a.type() == typeid(Sec))
       {
       if(b.df(s) == 0 && b!=0 && bpower % 2 != 0) // odd power case
       {
              int k = 1, l=2;
              vector<int> v={1};
              vector<int> mc={1}; // to store the middle coefficient
              Symbolic sgn = -1;
              Symbolic integral_numerator, integral_denominator;
              Symbolic d0 = 2, d1 = 2;
              int j =1;
              int m = bpower-(0.5*(bpower+1));
              int last_coeff;
              int first_coeff = 3;

              // For the coefficient at the numerator of sine with odd power
              for(int i = 1 ; i < (bpower-1)/2 ; i = i+1)
              {
                     if (i >= 2)
                     {
                            for(int ic = 1 ; ic < i ; ic = ic+1)
                            {
                                   mc[ic-1] = v[ic-1] + v[ic];
                            }
                     }
                     k = k*l;
                     last_coeff = v[j-1];
                     v[0] = v[0]*(first_coeff+2*(i-1));
                     v.assign({v[0]});

                     for(int ic = 1 ; ic < i ; ic = ic+1)
                     {
                            v.push_back(mc[ic-1]*(first_coeff+2*(i-1)));
                     }
                     d1 = d1*(2*i);
                     d0 = d0*(2+2*i);
                     v.push_back(last_coeff*(first_coeff+2*(i-1))+k);
                     l = l+2;
                     j = j+1;
```

```
            }
            int j_num = 0 ;
            for(int i = bpower-2 ; i >= 1 ; i = i-2)
            {
                  integral_numerator += sgn*v[j_num]*((sin(s))^(i));
                  sgn = -sgn;
                  j_num = j_num+1;
            }
            sgn = 1;
            j = 1;
            for(int i = bpower-1 ; i >= 0 ; i = i-2)
            {
                  integral_denominator += sgn*d0*combinations(m,j-1)*((
                      sin(s))^(i));
                  sgn = -sgn;
                  j = j+1;
            }
            d1 = d1*(bpower-1);
            return (-v[0]*ln(sin(s)-1))/(d1) + (v[0]*ln(sin(s)+1))/(d1) +
                (integral_numerator)/(integral_denominator);
      }
      }
```

Code 17: *level 3 integral for secant odd case*

iv. $\int \csc^n x \, dx$

These are the formula of $\int \csc^n x \, dx$ with $n = 2$ to $n = 8$.

$$\int \csc^2 x \, dx = -\frac{\cos x}{\sin x}$$

$$\int \csc^3 x \, dx = \frac{\log(\cos(x) - 1)}{4} - \frac{\log(\cos(x) + 1)}{4} + \frac{\cos x}{2\cos^2 x - 2}$$

$$\int \csc^4 x \, dx = -\frac{2\cos x}{3\sin x} - \frac{\cos x}{3\sin^3 x}$$

$$\int \csc^5 x \, dx = \frac{3\cos^3 x - 5\cos x}{8\cos^4 x - 16\cos^2 x + 8} + \frac{3\log(\cos(x) - 1)}{16} - \frac{3\log(\cos(x) + 1)}{16}$$

$$\int \csc^6 x \, dx = -\frac{8\cos x}{15\sin x} - \frac{4\cos x}{15\sin^3 x} - \frac{\cos x}{5\sin^5 x}$$

$$\int \csc^7 x \, dx = -\frac{-15\cos^5 x + 40\cos^3 x - 33\cos x}{48\cos^6 x - 144\cos^4 x + 144\cos^2 x - 48} + \frac{5\log(\cos(x) - 1)}{32} - \frac{5\log(\cos(x) + 1)}{32}$$

$$\int \csc^8 x \, dx = -\frac{16\cos x}{35\sin x} - \frac{8\cos x}{35\sin^3 x} - \frac{6\cos x}{35\sin^5 x} - \frac{\cos x}{7\sin^7 x}$$

The pattern for the numerator and denominator will be like the secant counterpart, thus we only need to exchange the sign, what is $-$ become $+$ and the other way around, and also exchanging sine into cosine and cosine into sine.

```
...
```

```
Symbolic Power::integrate(const Symbolic &s) const
{
       ...
       if(a.type() == typeid(Csc))
       {
              ...

              if(b.df(s) == 0 && b!=0 && bpower % 2 == 0) // even power case
              {
                     int v[999];
                     v[0] = 1;
                     Symbolic integral;
                     Symbolic d0 = 1;
                     int k = 1, l = 2, c=1;
                     for(int i = 1 ; i < bpower ; i = i+2) // to compute the
                         denominator
                     {
                            d0 *= i;
                     }
                     for(int i = 1 ; i < bpower — 1; i = i+2)
                     {
                            c = v[0];
                            int arrsec[999]; // make the size of the array as
                                 big as possible

                            for(int j = 0 ; j < i—1 ; j = j+1)
                            {
                                   arrsec[j] = v[j];
                            }
                            int d;
                            for(int j = 1 ; j < i ; j = j+1)
                            {
                                   d = arrsec[j—1];
                                   v[j] = d*l;
                            }

                            v[0] = c*k;
                            v[1] = c*l;

                            k= k + 2;
                            l = l + 2;
                     }

                     int j_d = 1 ;
                     for(int i = 1 ; i < (0.5*bpower)+1; i = i+1)
                     {
                            integral —= ( v[i—1] * cos(s) ) / ( d0*(sin(s)^(
```

```
                                bpower-j_d)) ) ;
                        j_d = j_d+2;
                }
                return integral;
        }
    }
}
```

Code 18: *level 3 integral for cosecant even case*

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
    ...
    if(a.type() == typeid(Csc))
    {
    ...

    if(b.df(s) == 0 && b!=0 && bpower % 2 != 0) // odd power case
    {
        int k = 1, l=2;
        vector<int> v={1};
        vector<int> mc={1}; // to store the middle coefficient
        Symbolic sgn = -1;
        Symbolic integral_numerator, integral_denominator;
        Symbolic d0 = 2, d1 = 2;
        int j =1;
        int m = bpower-(0.5*(bpower+1));
        int last_coeff;
        int first_coeff = 3;

        // For the coefficient at the numerator of cosine with odd
            power
        for(int i = 1 ; i < (bpower-1)/2 ; i = i+1)
        {
            if (i >= 2)
            {
                for(int ic = 1 ; ic < i ; ic = ic+1)
                {
                    mc[ic-1] = v[ic-1] + v[ic];
                }
            }
            k = k*l;
            last_coeff = v[j-1];
            v[0] = v[0]*(first_coeff+2*(i-1));
            v.assign({v[0]});

            for(int ic = 1 ; ic < i ; ic = ic+1)
```

```
                    {
                           v.push_back(mc[ic-1]*(first_coeff+2*(i-1)));
                    }
                    d1 = d1*(2*i);
                    d0 = d0*(2+2*i);
                    v.push_back(last_coeff*(first_coeff+2*(i-1))+k);
                    l = l+2;
                    j = j+1;
             }
             int j_num = 0 ;
             for(int i = bpower-2 ; i >= 1 ; i = i-2)
             {
                    integral_numerator += sgn*v[j_num]*((cos(s))^(i));
                    sgn = -sgn;
                    j_num = j_num+1;
             }
             sgn = 1;
             j = 1;
             for(int i = bpower-1 ; i >= 0 ; i = i-2)
             {
                    integral_denominator += sgn*d0*combinations(m,j-1)*((
                        cos(s))^(i));
                    sgn = -sgn;
                    j = j+1;
             }
             d1 = d1*(bpower-1);
             return (v[0]*ln(cos(s)-1))/(d1) - (v[0]*ln(cos(s)+1))/(d1) -
                 (integral_numerator)/(integral_denominator);
      }
      }
}
```

Code 19: *level 3 integral for cosecant odd case*

v. $\int \cot^n x \, dx$

These are the formula of $\int \cot^n x \, dx$ with $n = 2$ to $n = 8$.

$$\int \cot^2 x \, dx = -x - \frac{\cos\ x}{\sin\ x}$$
$$\int \cot^3 x \, dx = -\log(\sin\ (x)) - \frac{1}{2\sin^2\ x}$$
$$\int \cot^4 x \, dx = x + \frac{\cos\ x}{\sin\ x} - \frac{\cos^3\ x}{3\sin^3\ x}$$
$$\int \cot^5 x \, dx = \log(\sin\ (x)) + \frac{4\sin^2\ x - 1}{4\sin^4\ x}$$
$$\int \cot^6 x \, dx = -x - \frac{\cos\ x}{\sin\ x} + \frac{\cos^3\ x}{3\sin^3\ x} - \frac{\cos^5\ x}{5\sin^5\ x}$$
$$\int \cot^7 x \, dx = -\log(\sin\ (x)) - \frac{18\sin^4\ x - 9\sin^2\ x + 2}{12\sin^6\ x}$$
$$\int \cot^8 x \, dx = x + \frac{\cos\ x}{\sin\ x} - \frac{\cos^3\ x}{3\sin^3\ x} + \frac{\cos^5\ x}{5\sin^5\ x} - \frac{\cos^7\ x}{7\sin^7\ x}$$

Case 1: For $\int \cot^n x \, dx$ with even n
We will cluster only the even power

$$\int \cot^2 x \, dx = -x - \frac{\cos\ x}{\sin\ x}$$
$$\int \cot^4 x \, dx = x + \frac{\cos\ x}{\sin\ x} - \frac{\cos^3\ x}{3\sin^3\ x}$$
$$\int \cot^6 x \, dx = -x - \frac{\cos\ x}{\sin\ x} + \frac{\cos^3\ x}{3\sin^3\ x} - \frac{\cos^5\ x}{5\sin^5\ x}$$
$$\int \cot^8 x \, dx = x + \frac{\cos\ x}{\sin\ x} - \frac{\cos^3\ x}{3\sin^3\ x} + \frac{\cos^5\ x}{5\sin^5\ x} - \frac{\cos^7\ x}{7\sin^7\ x}$$

The patterns that can be seen from those equations above are

- The variable x is only alternating sign at every iteration.
- The numerator' coefficient with cosine function of odd power is only 1.
- The denominator coefficient is the same as the power of the sine function in the denominator.
- The integral computed at the previous iteration alternates its sign at the next iteration minus the new function with power of $n - 1$.

The C++ code is an easy one here, the simplest out of all level 3 trigonometry integral, it alternates the whole sign at every iteration. We will use an increasing **for** loop with $i = 2, i <= n, i = i + 2$.

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
      ...
      if(a.type() == typeid(Cot))
      {
```

```
        ...
        if(b.df(s) == 0 && b!=0 && bpower % 2 == 0) // even power case
        {
                Symbolic integral;
                Symbolic integral_front;
                Symbolic sgn = -1;
                for(int i = 2 ; i <= (bpower) ; i = i+2)
                {
                        integral = (-1)*integral;
                        integral += - (cos(s)^(i-1)) / ((i-1)*(sin(s)^(i-1)));
                }

                for(int i = 1 ; i <= (bpower)/2 ; i = i+1)
                {
                        integral_front = sgn;
                        sgn = -sgn;
                }
                return integral_front*s + integral;
        }
}
```

Code 20: *level 3 integral for cotangent even case*

Case 2: For $\int \cot^n x\, dx$ with odd n
We will cluster only the odd power

$$\int \cot^3 x\, dx = -\log(\sin(x)) - \frac{1}{2\sin^2 x}$$
$$\int \cot^5 x\, dx = \log(\sin(x)) + \frac{4\sin^2 x - 1}{4\sin^4 x}$$
$$\int \cot^7 x\, dx = -\log(\sin(x)) - \frac{18\sin^4 x - 9\sin^2 x + 2}{12\sin^6 x}$$
$$\int \cot^9 x\, dx = \log(\sin(x)) + \frac{48\sin^4 x - 36\sin^4 x + 16\sin^2 x - 3}{24\sin^8 x}$$
$$\int \cot^{11} x\, dx = -\log(\sin(x)) - \frac{300\sin^8 x - 300\sin^6 x + 200\sin^4 x - 75\sin^2 x + 12}{120\sin^{10} x}$$

The patterns that can be seen from those equations above are

- The $\log(\sin(x))$ function is only alternating the sign at every iteration.
- The denominator for the sine function with power of $n-1$ is making a pattern like this (d_0 is the variable to represent the denominator)

$$d_0 = d_0 * \left(\frac{n-1}{2}\right)$$

We start with $d_0 = 2$ then

$$d_0 = 2 * \left(\frac{5-1}{2}\right) = 4$$
$$d_0 = 4 * \left(\frac{7-1}{2}\right) = 12$$

$$d_0 = 12 * \left(\frac{9-1}{2}\right) = 48$$
$$d_0 = 48 * \left(\frac{11-1}{2}\right) = 240$$

Knowing that the denominator has been simplified then we know that we have to adjust the numerator so they become the raw numbers like this:

$$\int \cot^3 x \, dx = -\log(\sin(x)) - \frac{1}{2\sin^2 x}$$
$$\int \cot^5 x \, dx = \log(\sin(x)) + \frac{4\sin^2 x - 1}{4\sin^4 x}$$
$$\int \cot^7 x \, dx = -\log(\sin(x)) - \frac{18\sin^4 x - 9\sin^2 x + 2}{12\sin^6 x}$$
$$\int \cot^9 x \, dx = \log(\sin(x)) + \frac{96\sin^4 x - 72\sin^4 x + 32\sin^2 x - 6}{48\sin^8 x}$$
$$\int \cot^{11} x \, dx = -\log(\sin(x)) - \frac{600\sin^8 x - 600\sin^6 x + 400\sin^4 x - 150\sin^2 x + 24}{120\sin^{10} x}$$

It is very important to do this in order to know the pattern for the numerator.

- To be able to know the pattern for the numerator with sine function of even power we can easily put the numerator coefficients like this for a better look:

$$\begin{array}{ccccccccccc} & & & & & 1 & & & & & \\ & & & & 1 & & 4 & & & & \\ & & & 2 & & 9 & & 18 & & & \\ & & 6 & & 32 & & 72 & & 96 & & \\ & 24 & & 150 & & 400 & & 600 & & 600 & \end{array}$$

The rule of thumb is we have to vectorize the coefficients, making it into a vector of size $\frac{n-1}{2}$ at every iteration. Back to the reduction formula, if we want to compute $\int \cot^{11} x \, dx$ we will start from $\int \cot^1 x \, dx$ then $\int \cot^3 x \, dx$, this is the recursive method.

When we create the vector at certain iteration, we will realize that to know the entries of this vector, we will need the values from certain variables that are having pattern, e.g. increasing at every iteration or decreasing at every iteration or multiplying with increasing coefficient. Now we take a look again. At the first iteration we have a vector $\mathbf{v}$ with only has one entry / size of 1 which is

$$\mathbf{v}[0] = 1$$

then at the next iteration we have a vector of size 2 with entries

$$\mathbf{v}[0] = 1$$
$$\mathbf{v}[1] = 4$$

then at the next iteration we have a vector of size 3 with entries

$$\mathbf{v}[0] = 2$$
$$\mathbf{v}[1] = 9$$
$$\mathbf{v}[2] = 18$$

We will use 4 vectors this time, since this one is quite tricky compared to the secant case. The vectors are: $\mathbf{v}, \mathbf{v}_{temp}, \mathbf{mc}, \mathbf{mc}_{temp}$.

We also need to use variables to help compute the coefficients with their initial value before the **for** loop :

$$d_0 = 2$$

$$k = 1$$

$$l = 2$$

We will use an increasing **for** loop with $i = 1, i <= \frac{n-1}{2}, i = i + 1$ because the size of the vector is getting bigger at every iteration, the size of the vector is the same as the iteration number.

You can and probably should write it on paper so you can learn more. To be simple, after several days of trial and error, the formula is shown to be like this:

$$\begin{aligned}
\mathbf{v}[0]_i &= \mathbf{v}[0]_{i-1} * (i-1) * \mathbf{mc}[0]_i \\
\mathbf{v}[j]_i &= [\mathbf{v}[j-1]_{i-1} * (2 * i - j)] + [\mathbf{v}[0]_i * (\mathbf{mc}[j]_i)] \\
\mathbf{v}\left[\frac{n-1}{2}\right]_i &= \left[\mathbf{v}\left[\frac{n-1}{2} - 1\right]_{i-1} * (2 * i - j)\right] + (\mathbf{v}[0]_i * *\mathbf{mc}\left[\frac{n-1}{2}\right]_i)
\end{aligned}$$

with i is the iteration $i = 1, 2, \ldots, \frac{n-1}{2}$, and $j = 1, 2, \ldots, \frac{n-1}{2}$. The vector **mc**, the middle coefficient is the Pascal' triangle

$$\begin{array}{ccccccccccccc}
 & & & & & & 1 & & & & & & \\
 & & & & & 1 & & 1 & & & & & \\
 & & & & 1 & & 2 & & 1 & & & & \\
 & & & 1 & & 3 & & 3 & & 1 & & & \\
 & & 1 & & 4 & & 6 & & 4 & & 1 & & \\
 & 1 & & 5 & & 10 & & 10 & & 5 & & 1 & \\
1 & & 6 & & 15 & & 20 & & 15 & & 6 & & 1
\end{array}$$

the top row represents $i = 1$ for $\int \cot^3 x \, dx$, and this vector **mc** helps computing the numerator coefficient.

The computation / coding of C++ for the numerator coefficients will be a tough one, it takes an increasing **for** loop with $i = 1, i <= \frac{n-1}{2}, i = i + 1$ and we split it into several cases.

So if we want to compute $\int \cot^{11} x \, dx$ we will gain the numerator coefficients at the iteration $i = 5$.

So the first **for** loop will be like this:

$$
\begin{aligned}
i &= 1 \\
d_0 &= 2 \\
mc[0] &= 1 \\
v[0] &= 1 \\
&**********\\
i &= 2 \\
d_0 &= 4 \\
mc[0] &= 1 \\
mc[1] &= 1 \\
v[0] &= 1 \\
v[1] &= 4 \\
&**********\\
i &= 3 \\
d_0 &= 12 \\
mc[0] &= 1 \\
mc[1] &= 2 \\
mc[1] &= 1 \\
v[0] &= 2 \\
v[1] &= 9 \\
v[2] &= 18 \\
&**********\\
i &= 4 \\
d_0 &= 48 \\
mc[0] &= 1 \\
mc[1] &= 3 \\
mc[2] &= 3 \\
mc[3] &= 1 \\
v[0] &= 6 \\
v[1] &= 32 \\
v[2] &= 72 \\
v[3] &= 96 \\
&**********
\end{aligned}
$$

$$
\begin{aligned}
i &= 5 \\
d_0 &= 240 \\
mc[0] &= 1 \\
mc[1] &= 4 \\
mc[2] &= 6 \\
mc[3] &= 4 \\
mc[4] &= 1 \\
v[0] &= 24 \\
v[1] &= 150 \\
v[2] &= 400 \\
v[3] &= 600 \\
v[4] &= 600
\end{aligned}
$$

We omitted the **mc**$[0]$ and **mc**$[i-1]$ in the C++ codes since it will only returns 1 so it won't really change much.

The rest of the **for** loops are only used to return the correct integral.

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
	...
	if(a.type() == typeid(Cot))
	{
	...
	if(b.df(s) == 0 && b!=0 && bpower % 2 != 0) // odd power case
	{
		Symbolic integral;
		Symbolic integral_front;
		Symbolic sgn = -1;
		vector<int> v={1};
		vector<int> v_temp={1};
		vector<int> mc={1}; // to store the middle coefficient
		vector<int> mc_temp={1}; // to store the temporary / new middle
		     coefficient
		int d0 = 2;
		int k = 1, l=2;
		// For the coefficient at the numerator
		for(int i = 1 ; i <= (bpower-1)/2 ; i = i+1)
		{
			if (i == 1)
			{
				v[0] = 1;
				v.assign({v[0]});
			}
```

```
                    if (i ==2)
                    {
                           mc[0]=1;
                           mc.assign({mc[0]});
                           v_temp[0] = v[0]*(i-1);
                           v_temp.assign({v_temp[0]});

                           for(int j = 1 ; j < i ; j = j+1)
                           {
                                  v_temp.push_back(v[j-1]*((2*i)-j) +
                                      v_temp[0]*mc[j-1] );
                           }
                           v[0] = v_temp[0];
                           v.assign({v[0]});
                           for(int j = 1 ; j < i ; j = j+1)
                           {
                                  v.push_back(v_temp[j]);
                           }
                    }
                    if (i == 3)
                    {
                           mc[0] = mc[0] + 1; //
                           mc.assign({mc[0]});

                           v_temp[0] = v[0]*(i-1);
                           v_temp.assign({v_temp[0]});
                           for(int j = 1 ; j < i-1 ; j = j+1)
                           {
                                  v_temp.push_back(v[j-1]*((2*i)-j) +
                                      v_temp[0]*mc[j-1] );
                           }
                           v_temp.push_back((v[i-2]*(i+1) ) + (v_temp[0]) );
                           // Assign the temporary vector to vector v for
                               future use
                           v[0] = v_temp[0];
                           v.assign({v[0]});
                           for(int j = 1 ; j < i ; j = j+1)
                           {
                                  v.push_back(v_temp[j]);
                           }
                    }
                    if (i == 4)
                    {
                           mc[0] = mc[0] + 1;
                           mc.assign({mc[0]});
                           mc.push_back(mc[0]);

                           v_temp[0] = v[0]*(i-1);
```

```
                        v_temp.assign({v_temp[0]});
                        for(int j = 1 ; j < i-1 ; j = j+1)
                        {
                                v_temp.push_back(v[j-1]*((2*i)-j) +
                                    v_temp[0]*mc[j-1] );
                        }
                        v_temp.push_back((v[i-2]*(i+1) ) + (v_temp[0]) );
                        // Assign the temporary vector to vector v for
                            future use
                        v[0] = v_temp[0];
                        v.assign({v[0]});
                        for(int j = 1 ; j < i ; j = j+1)
                        {
                                v.push_back(v_temp[j]);
                        }
                }
                if (i >= 5)
                {
                        mc_temp[0] = mc[0]+1;
                        mc_temp.assign({mc_temp[0]});
                        for(int ic = 1 ; ic < l ; ic = ic+1)
                        {
                                mc_temp[ic] = mc[ic-1] + mc[ic];
                        }
                        mc[l] = (mc_temp[0]);
                        mc[l+1] = (mc_temp[0]);

                        mc[0]=mc_temp[0];
                        mc.assign({mc[0]});

                        for(int ic = 1 ; ic < l ; ic = ic+1)
                        {
                                mc.push_back(mc_temp[ic]);
                        }
                        mc.push_back(mc_temp[0]);
                        mc.push_back(0);

                        v_temp[0] = v[0]*(i-1);
                        v_temp.assign({v_temp[0]});
                        for(int j = 1 ; j < i-1 ; j = j+1)
                        {
                                v_temp.push_back(v[j-1]*((2*i)-j) +
                                    v_temp[0]*mc[j-1] );
                        }
                        v_temp.push_back((v[i-2]*(i+1) ) + (v_temp[0]) );
                        // Assign the temporary vector to vector v for
                            future use
                        v[0] = v_temp[0];
```

```
                        v.assign({v[0]});
                        for(int j = 1 ; j < i ; j = j+1)
                        {
                                v.push_back(v_temp[j]);
                        }

                        l = l+1;
                }

                d0 = d0*k;
                k = k+1;
        }
        for(int i = 1 ; i <= (bpower-1)/2 ; i = i+1)
        {
                integral += sgn*v[i-1]*(sin(s)^(2*(i-1)));
                sgn=-sgn;

        }

        for(int i = 1 ; i <= (bpower-1)/2 ; i = i+1)
        {
                integral_front = sgn;
                sgn = -sgn;
        }
        return integral_front*ln(sin(s)) + (integral)/(d0*(sin(s)^(
            bpower-1))) ;
    }
}
```

Code 21: *level 3 integral for cotangent odd case*

vi. $\int \tan^n x \, dx$

These are the formula of $\int \tan^n x \, dx$ with $n = 2$ to $n = 8$.

$$\int \tan^2 x \, dx = -x + \frac{\sin\ x}{\cos\ x}$$
$$\int \tan^3 x \, dx = \log(\cos\ x) + \frac{1}{2\cos^2\ x}$$
$$\int \tan^4 x \, dx = x - \frac{\sin\ x}{\cos\ x} + \frac{\sin^3\ x}{3\cos^3\ x}$$
$$\int \tan^5 x \, dx = -\log(\cos\ x) - \frac{4\cos^2\ x - 1}{4\cos^4\ x}$$
$$\int \tan^6 x \, dx = -x + \frac{\sin\ x}{\cos\ x} - \frac{\sin^3\ x}{3\cos^3\ x} + \frac{\sin^5\ x}{5\cos^5\ x}$$
$$\int \tan^7 x \, dx = \log(\cos\ x) + \frac{18\cos^4\ x - 9\cos^2\ x + 2}{12\cos^6\ x}$$
$$\int \tan^8 x \, dx = x - \frac{\sin\ x}{\cos\ x} + \frac{\sin^3\ x}{3\cos^3\ x} - \frac{\sin^5\ x}{5\cos^5\ x} + \frac{\sin^7\ x}{7\cos^7\ x}$$

The pattern for the numerator and denominator will be like the cotangent counterpart, thus we only need to exchange the sign, what is $-$ become $+$ and the other way around, and also exchanging sine into cosine and cosine into sine.

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
	...
	if(a.type() == typeid(Cot))
	{
		...
	if(b.df(s) == 0 && b!=0 && bpower % 2 == 0) // even power case
	{
		Symbolic integral;
		Symbolic integral_front;
		Symbolic sgn = -1;
		for(int i = 2 ; i <= (bpower) ; i = i+2)
		{
			integral = (-1)*integral;
			integral += (sin(s)^(i-1)) / ((i-1)*(cos(s)^(i-1)));
		}

		for(int i = 1 ; i <= (bpower)/2 ; i = i+1)
		{
			integral_front = sgn;
			sgn = -sgn;
		}
		return integral_front*s + integral;
	}
```

```
        }
}
```

Code 22: *level 3 integral for tangent even case*

```
...

Symbolic Power::integrate(const Symbolic &s) const
{
        ...
        if(a.type() == typeid(Cot))
        {
        ...
        if(b.df(s) == 0 && b!=0 && bpower % 2 != 0) // odd power case
        {
                Symbolic integral;
                Symbolic integral_front;
                Symbolic sgn = 1;
                vector<int> v={1};
                vector<int> v_temp={1};
                vector<int> mc={1}; // to store the middle coefficient
                vector<int> mc_temp={1}; // to store the temporary / new middle
                     coefficient
                int d0 = 2;
                int k = 1, l=2;
                // For the coefficient at the numerator
                for(int i = 1 ; i <= (bpower-1)/2 ; i = i+1)
                {
                        if (i == 1)
                        {
                                v[0] = 1;
                                v.assign({v[0]});
                        }

                        if (i ==2)
                        {
                                mc[0]=1;
                                mc.assign({mc[0]});
                                v_temp[0] = v[0]*(i-1);
                                v_temp.assign({v_temp[0]});

                                for(int j = 1 ; j < i ; j = j+1)
                                {
                                        v_temp.push_back(v[j-1]*((2*i)-j) +
                                            v_temp[0]*mc[j-1] );
                                }
                                v[0] = v_temp[0];
                                v.assign({v[0]});
                                for(int j = 1 ; j < i ; j = j+1)
```

```
                {
                        v.push_back(v_temp[j]);
                }
        }
        if (i == 3)
        {
                mc[0] = mc[0] + 1; //
                mc.assign({mc[0]});

                v_temp[0] = v[0]*(i-1);
                v_temp.assign({v_temp[0]});
                for(int j = 1 ; j < i-1 ; j = j+1)
                {
                        v_temp.push_back(v[j-1]*((2*i)-j) +
                            v_temp[0]*mc[j-1] );
                }
                v_temp.push_back((v[i-2]*(i+1) ) + (v_temp[0]) );
                // Assign the temporary vector to vector v for
                    future use
                v[0] = v_temp[0];
                v.assign({v[0]});
                for(int j = 1 ; j < i ; j = j+1)
                {
                        v.push_back(v_temp[j]);
                }
        }
        if (i == 4)
        {
                mc[0] = mc[0] + 1;
                mc.assign({mc[0]});
                mc.push_back(mc[0]);

                v_temp[0] = v[0]*(i-1);
                v_temp.assign({v_temp[0]});
                for(int j = 1 ; j < i-1 ; j = j+1)
                {
                        v_temp.push_back(v[j-1]*((2*i)-j) +
                            v_temp[0]*mc[j-1] );
                }
                v_temp.push_back((v[i-2]*(i+1) ) + (v_temp[0]) );
                // Assign the temporary vector to vector v for
                    future use
                v[0] = v_temp[0];
                v.assign({v[0]});
                for(int j = 1 ; j < i ; j = j+1)
                {
                        v.push_back(v_temp[j]);
                }
```

```
                }
                if (i >= 5)
                {
                        mc_temp[0] = mc[0]+1;
                        mc_temp.assign({mc_temp[0]});
                        for(int ic = 1 ; ic < l ; ic = ic+1)
                        {
                                mc_temp[ic] = mc[ic-1] + mc[ic];
                        }
                        mc[l] = (mc_temp[0]);
                        mc[l+1] = (mc_temp[0]);

                        mc[0]=mc_temp[0];
                        mc.assign({mc[0]});

                        for(int ic = 1 ; ic < l ; ic = ic+1)
                        {
                                mc.push_back(mc_temp[ic]);
                        }
                        mc.push_back(mc_temp[0]);
                        mc.push_back(0);

                        v_temp[0] = v[0]*(i-1);
                        v_temp.assign({v_temp[0]});
                        for(int j = 1 ; j < i-1 ; j = j+1)
                        {
                                v_temp.push_back(v[j-1]*((2*i)-j) +
                                    v_temp[0]*mc[j-1] );
                        }
                        v_temp.push_back((v[i-2]*(i+1) ) + (v_temp[0]) );
                        // Assign the temporary vector to vector v for
                            future use
                        v[0] = v_temp[0];
                        v.assign({v[0]});
                        for(int j = 1 ; j < i ; j = j+1)
                        {
                                v.push_back(v_temp[j]);
                        }

                        l = l+1;
                }

                d0 = d0*k;
                k = k+1;
        }
        for(int i = 1 ; i <= (bpower-1)/2 ; i = i+1)
        {
                integral += sgn*v[i-1]*(cos(s)^(2*(i-1)));
```

```
                    sgn=-sgn;

                }

                for(int i = 1 ; i <= (bpower-1)/2 ; i = i+1)
                {
                    integral_front = sgn;
                    sgn = -sgn;
                }
                return integral_front*ln(cos(s)) + (integral)/(d0*(cos(s)^(
                    bpower-1))) ;
            }
            }
}
```

Code 23: *level 3 integral for tangent odd case*

- Trigonometry Level 4

[SI*] The trigonometry integrals that we consider as level 4 is

$$\int \sin^n x \ \cos^m \ x \, dx$$
$$\int \tan^n x \ \sec^m \ x \, dx$$
$$\int \cot^n x \ \csc^m \ x \, dx$$

the reduction formula to compute these trigonometry integral level 4 are

$$\int \sin^n x \ \cos^m \ x \, dx = \frac{\sin^{n+1} x \ \cos^{m-1} x}{m+n} + \frac{m-1}{m+n} \int \sin^n x \ \cos^{m-2} x \, dx \tag{2.16}$$

vii. $\int \sin^n x \ \cos^m \ x \, dx$

The differentiation to obtain the reduction formula is coming from integration by parts and a smart algebra technique

$$\begin{aligned}
I_{m,n} &= \int \sin^n x \ \cos^m \ x \, dx \\
&= \int \cos^{m-1} x \ \sin^n x \ \cos \ x dx \\
&= \cos^{m-1} x \frac{\sin^{n+1} x}{n+1} - \frac{m-1}{n+1} \int \sin^{n+2} x \ \cos^{m-2} x \, dx \\
&= \cos^{m-1} x \frac{\sin^{n+1} x}{n+1} - \frac{m-1}{n+1} \int \sin^n x \ \cos^{m-2} x (1-\cos^2 x) \, dx \\
&= \frac{\sin^{n+1} x \ \cos^{m-1} x}{m+n} + \frac{m-1}{m+n} \int \sin^n x \ \cos^{m-2} x (1-\cos^2 x) \, dx \\
&= \frac{\sin^{n+1} x \ \cos^{m-1} x}{m+n} + \frac{m-1}{m+n} \int \sin^n x \ \cos^{m-2} x \, dx
\end{aligned}$$

These are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 1$ and $m = 1$ to $m = 8$.

$$\int \sin^1 x \ \cos^1 \ x \, dx = \frac{\sin^2 \ x}{2}$$
$$\int \sin^1 x \ \cos^2 \ x \, dx = \frac{-\cos^3 \ x}{3}$$
$$\int \sin^1 x \ \cos^3 \ x \, dx = \frac{-\cos^4 \ x}{4}$$
$$\int \sin^1 x \ \cos^4 \ x \, dx = \frac{-\cos^5 \ x}{5}$$
$$\int \sin^1 x \ \cos^5 \ x \, dx = \frac{-\cos^6 \ x}{6}$$
$$\int \sin^1 x \ \cos^6 \ x \, dx = \frac{-\cos^7 \ x}{7}$$
$$\int \sin^1 x \ \cos^7 \ x \, dx = \frac{-\cos^8 \ x}{8}$$
$$\int \sin^1 x \ \cos^8 \ x \, dx = \frac{-\cos^9 \ x}{9}$$

These are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 2$ and $m = 1$ to $m = 8$.

$$\int \sin^2 x \ \cos^1 \ x \, dx = \frac{\sin^3 \ x}{3}$$
$$\int \sin^2 x \ \cos^2 \ x \, dx = \frac{x}{8} - \frac{\sin(2x)\cos(2x)}{16}$$
$$\int \sin^2 x \ \cos^3 \ x \, dx = -\frac{\sin^5 \ x}{5} + \frac{\sin^3 \ x}{3}$$
$$\int \sin^2 x \ \cos^4 \ x \, dx = \frac{x}{16} - \frac{\sin(x)\cos^5(x)}{6} + \frac{\sin(x)\cos^3(x)}{24} + \frac{\sin(x)\cos(x)}{16}$$
$$\int \sin^2 x \ \cos^5 \ x \, dx = \frac{\sin^7 \ x}{7} - \frac{2\sin^5 \ x}{5} + \frac{\sin^3 \ x}{3}$$
$$\int \sin^2 x \ \cos^6 \ x \, dx = \frac{5x}{128} - \frac{\sin(x)\cos^7(x)}{8} + \frac{\sin(x)\cos^5(x)}{48} + \frac{5\sin(x)\cos^3(x)}{192} + \frac{5\sin(x)\cos(x)}{128}$$
$$\int \sin^2 x \ \cos^7 \ x \, dx = -\frac{\sin^9 \ x}{9} + \frac{3\sin^7 \ x}{7} - \frac{3\sin^5 \ x}{5} + \frac{\sin^3 \ x}{3}$$
$$\int \sin^2 x \ \cos^8 \ x \, dx = \frac{7x}{256} - \frac{\sin(x)\cos^9(x)}{10} + \frac{\sin(x)\cos^7(x)}{80} + \frac{7\sin(x)\cos^5(x)}{480} + \frac{7\sin(x)\cos^3(x)}{384} + \frac{7\sin(x)\cos(x)}{256}$$

These are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 3$ and $m = 1$ to $m = 8$.

$$\int \sin^3 x \ \cos^1 \ x \, dx = \frac{\sin^4 \ x}{4}$$
$$\int \sin^3 x \ \cos^2 \ x \, dx = \frac{\cos^5 \ x}{5} - \frac{\cos^3 \ x}{3}$$
$$\int \sin^3 x \ \cos^3 \ x \, dx = -\frac{\sin^6 \ x}{6} + \frac{\sin^4 \ x}{4}$$
$$\int \sin^3 x \ \cos^4 \ x \, dx = \frac{\cos^7 \ x}{7} - \frac{\cos^5 \ x}{5}$$
$$\int \sin^3 x \ \cos^5 \ x \, dx = \frac{\cos^8 \ x}{8} - \frac{\cos^6 \ x}{6}$$
$$\int \sin^3 x \ \cos^6 \ x \, dx = \frac{\cos^9 \ x}{9} - \frac{\cos^7 \ x}{7}$$
$$\int \sin^3 x \ \cos^7 \ x \, dx = \frac{\cos^{10} \ x}{10} - \frac{\cos^8 \ x}{8}$$
$$\int \sin^3 x \ \cos^8 \ x \, dx = \frac{\cos^{11} \ x}{11} - \frac{\cos^9 \ x}{9}$$

These are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 4$ and $m = 1$ to $m = 8$.

$$\int \sin^4 x \ \cos^1 \ x \, dx = \frac{\sin^5 \ x}{5}$$
$$\int \sin^4 x \ \cos^2 \ x \, dx = \frac{x}{16} + \frac{\sin^5(x)\cos(x)}{6} - \frac{\sin^3(x)\cos(x)}{24} - \frac{\sin(x)\cos(x)}{16}$$
$$\int \sin^4 x \ \cos^3 \ x \, dx = -\frac{\sin^7 \ x}{7} + \frac{\sin^5 \ x}{5}$$
$$\int \sin^4 x \ \cos^4 \ x \, dx = \frac{3x}{128} - \frac{\sin^3(2x)\cos(2x)}{128} - \frac{3\sin(2x)\cos(2x)}{256}$$
$$\int \sin^4 x \ \cos^5 \ x \, dx = \frac{\sin^9 \ x}{9} - \frac{2\sin^7 \ x}{7} + \frac{\sin^5 \ x}{5}$$
$$\int \sin^4 x \ \cos^6 \ x \, dx = \frac{3x}{256} + \frac{\sin(x)\cos^9(x)}{10} - \frac{11\sin(x)\cos^7(x)}{80} + \frac{\sin(x)\cos^5(x)}{160} + \frac{\sin(x)\cos^3(x)}{128} + \frac{3\sin(x)\cos(x)}{256}$$
$$\int \sin^4 x \ \cos^7 \ x \, dx = -\frac{\sin^{11} \ x}{11} + \frac{\sin^9 \ x}{3} - \frac{3\sin^7 \ x}{7} + \frac{\sin^5 \ x}{5}$$
$$\int \sin^4 x \ \cos^8 \ x \, dx = \frac{7x}{1024} + \frac{\sin(x)\cos^{11}(x)}{12} - \frac{13\sin(x)\cos^9(x)}{120} + \frac{\sin(x)\cos^7(x)}{320} + \frac{7\sin(x)\cos^5(x)}{1920} + \frac{7\sin(x)\cos^3(x)}{1536} + \frac{7\sin(x)\cos(x)}{1024}$$

These are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 5$ and $m = 1$ to $m = 8$.

$$\int \sin^5 x \ \cos^1 \ x \, dx = \frac{\sin^6 \ x}{6}$$
$$\int \sin^5 x \ \cos^2 \ x \, dx = -\frac{\cos^7 \ x}{7} + \frac{2\cos^5 \ x}{5} - \frac{\cos^3 \ x}{3}$$
$$\int \sin^5 x \ \cos^3 \ x \, dx = -\frac{\sin^8 \ x}{8} + \frac{\sin^6 \ x}{6}$$
$$\int \sin^5 x \ \cos^4 \ x \, dx = -\frac{\cos^9 \ x}{9} + \frac{2\cos^7 \ x}{7} - \frac{\cos^5 \ x}{5}$$
$$\int \sin^5 x \ \cos^5 \ x \, dx = \frac{\sin^{10} \ x}{10} - \frac{\sin^8 \ x}{4} + \frac{\sin^6 \ x}{6}$$
$$\int \sin^5 x \ \cos^6 \ x \, dx = -\frac{\cos^{11} \ x}{11} + \frac{2\cos^9 \ x}{9} - \frac{\cos^7 \ x}{7}$$
$$\int \sin^5 x \ \cos^7 \ x \, dx = -\frac{\cos^{12} \ x}{12} + \frac{\cos^{10} \ x}{5} - \frac{\cos^8 \ x}{8}$$
$$\int \sin^5 x \ \cos^8 \ x \, dx = -\frac{\cos^{13} \ x}{13} + \frac{2\cos^{11} \ x}{11} - \frac{\cos^9 \ x}{9}$$

These are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 6$ and $m = 1$ to $m = 8$.

$$\int \sin^6 x \ \cos^1 \ x \, dx = \frac{\sin^7 \ x}{7}$$
$$\int \sin^6 x \ \cos^2 \ x \, dx = \frac{5x}{128} + \frac{\sin^7(x)\cos(x)}{8} - \frac{\sin^5(x)\cos(x)}{48} - \frac{5\sin^3(x)\cos(x)}{192} - \frac{5\sin(x)\cos(x)}{128}$$
$$\int \sin^6 x \ \cos^3 \ x \, dx = -\frac{\sin^9 \ x}{9} + \frac{\sin^7 \ x}{7}$$
$$\int \sin^6 x \ \cos^4 \ x \, dx = \frac{3x}{256} - \frac{\sin^9(x)\cos(x)}{10} + \frac{11\sin^7(x)\cos(x)}{80} - \frac{\sin^5(x)\cos(x)}{160} - \frac{\sin^3(x)\cos(x)}{128} - \frac{3\sin(x)\cos(x)}{256}$$
$$\int \sin^6 x \ \cos^5 \ x \, dx = \frac{\sin^{11} \ x}{11} - \frac{2\sin^9 \ x}{9} + \frac{\sin^7 \ x}{7}$$
$$\int \sin^6 x \ \cos^6 \ x \, dx = \frac{5x}{1024} - \frac{\sin^5(2x)\cos(2x)}{768} - \frac{5\sin^3(2x)\cos(2x)}{3072} - \frac{5\sin(2x)\cos(2x)}{2048}$$
$$\int \sin^6 x \ \cos^7 \ x \, dx = -\frac{\sin^{13} \ x}{13} + \frac{3\sin^{11} \ x}{11} - \frac{\sin^9 \ x}{3} + \frac{\sin^7 \ x}{7}$$
$$\int \sin^6 x \ \cos^8 \ x \, dx = \frac{5x}{2048} - \frac{\sin(x)\cos^{13}(x)}{14} + \frac{29\sin(x)\cos^{11}(x)}{168} - \frac{37\sin(x)\cos^9(x)}{336} + \frac{\sin(x)\cos^7(x)}{896} + \frac{\sin(x)\cos^5(x)}{768} + \frac{5\sin(x)\cos^3(x)}{3072} + \frac{5\sin(x)\cos(x)}{2048}$$

These are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 7$ and $m = 1$ to $m = 8$.

$$\int \sin^7 x \ \cos^1 \ x \, dx = \frac{\sin^8 \ x}{8}$$
$$\int \sin^7 x \ \cos^2 \ x \, dx = \frac{\cos^9 \ x}{9} - \frac{3\cos^7 \ x}{7} + \frac{3\cos^5 \ x}{5} - \frac{\cos^3 \ x}{3}$$
$$\int \sin^7 x \ \cos^3 \ x \, dx = -\frac{\sin^{10} \ x}{10} + \frac{\sin^8 \ x}{8}$$
$$\int \sin^7 x \ \cos^4 \ x \, dx = \frac{\cos^{11} \ x}{11} - \frac{\cos^9 \ x}{3} + \frac{3\cos^7 \ x}{7} - \frac{\cos^5 \ x}{5}$$
$$\int \sin^7 x \ \cos^5 \ x \, dx = \frac{\sin^{12} \ x}{12} - \frac{\sin^{10} \ x}{5} + \frac{\sin^8 \ x}{8}$$
$$\int \sin^7 x \ \cos^6 \ x \, dx = \frac{\cos^{13} \ x}{13} - \frac{3\cos^{11} \ x}{11} + \frac{\cos^9 \ x}{3} - \frac{\cos^7 \ x}{7}$$
$$\int \sin^7 x \ \cos^7 \ x \, dx = -\frac{\sin^{14} \ x}{14} + \frac{\sin^{12} \ x}{4} - \frac{3\sin^{10} \ x}{10} + \frac{\sin^8 \ x}{8}$$
$$\int \sin^7 x \ \cos^8 \ x \, dx = \frac{\cos^{15} \ x}{15} - \frac{3\cos^{13} \ x}{13} + \frac{3\cos^{11} \ x}{11} - \frac{\cos^9 \ x}{9}$$

These are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 8$ and $m = 1$ to $m = 8$.

$$\int \sin^8 x \ \cos^1 \ x \, dx = \frac{\sin^9 \ x}{9}$$
$$\int \sin^8 x \ \cos^2 \ x \, dx = \frac{7x}{256} + \frac{\sin^9(x)\cos(x)}{10} - \frac{\sin^7(x)\cos(x)}{80} - \frac{7\sin^5(x)\cos(x)}{480} - \frac{7\sin^3(x)\cos(x)}{384} - \frac{7\sin(x)\cos(x)}{256}$$
$$\int \sin^8 x \ \cos^3 \ x \, dx = -\frac{\sin^{11} \ x}{11} + \frac{\sin^9 \ x}{9}$$
$$\int \sin^8 x \ \cos^4 \ x \, dx = \frac{7x}{1024} - \frac{\sin^{11}(x)\cos(x)}{12} + \frac{13\sin^9(x)\cos(x)}{120} - \frac{\sin^7(x)\cos(x)}{320} - \frac{7\sin^5(x)\cos(x)}{1920} - \frac{7\sin^3(x)\cos(x)}{1536} - \frac{7\sin(x)\cos(x)}{1024}$$
$$\int \sin^8 x \ \cos^5 \ x \, dx = \frac{\sin^{13} \ x}{13} - \frac{2\sin^{11} \ x}{11} + \frac{\sin^9 \ x}{9}$$
$$\int \sin^8 x \ \cos^6 \ x \, dx = \frac{5x}{2048} + \frac{\sin^{13}(x)\cos(x)}{14} - \frac{29\sin^{11}(x)\cos(x)}{168} + \frac{37\sin^9(x)\cos(x)}{336} - \frac{\sin^7(x)\cos(x)}{896} - \frac{\sin^5(x)\cos(x)}{768} - \frac{5\sin^3(x)\cos(x)}{3072} - \frac{5\sin(x)\cos(x)}{2048}$$
$$\int \sin^8 x \ \cos^7 \ x \, dx = -\frac{\sin^{15} \ x}{15} + \frac{3\sin^{13} \ x}{13} - \frac{3\sin^{11} \ x}{11} + \frac{\sin^9 \ x}{9}$$
$$\int \sin^8 x \ \cos^8 \ x \, dx = \frac{35x}{32768} - \frac{\sin^7(2x)\cos(2x)}{4096} - \frac{7\sin^5(2x)\cos(2x)}{24576} - \frac{35\sin^3(2x)\cos(2x)}{98304} - \frac{35\sin(2x)\cos(2x)}{65536}$$

Case 1: For $\int \sin^n x \ \cos^m \ x \, dx$ with even power for both n and m

This case occurs when both m and n are even number. We divide this case into 3 smaller

cases:

1. When $n > m$
2. When $n < m$
3. When $n = m$

Case 1-a: For $\int \sin^n x \ \cos^m \ x \, dx$ with $n > m$

We will start with a big example, these are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 14$ and $m = 2$ to $m = 12$ with increment of 2.

$$\int \sin^{14} x \ \cos^2 \ x \, dx = \frac{429x}{32768} + \frac{\sin^{15}(x)\cos(x)}{16} - \frac{\sin^{13}(x)\cos(x)}{224} - \frac{13\sin^{11}(x)\cos(x)}{2688} - \frac{143\sin^9(x)\cos(x)}{26880} - \frac{429\sin^7(x)\cos(x)}{71680} - \frac{143\sin^5(x)\cos(x)}{20480} - \frac{143\sin^3(x)\cos(x)}{16384} - \frac{429\sin(x)\cos(x)}{32768}$$

$$\int \sin^{14} x \ \cos^4 \ x \, dx = \frac{143x}{65536} - \frac{\sin^{17}(x)\cos(x)}{18} + \frac{19\sin^{15}(x)\cos(x)}{288} - \frac{\sin^{13}(x)\cos(x)}{1344} - \frac{13\sin^{11}(x)\cos(x)}{16128} - \frac{143\sin^9(x)\cos(x)}{161280} - \frac{143\sin^7(x)\cos(x)}{143360} - \frac{143\sin^5(x)\cos(x)}{122880} - \frac{143\sin^3(x)\cos(x)}{98304} - \frac{143\sin(x)\cos(x)}{65536}$$

$$\int \sin^{14} x \ \cos^6 \ x \, dx = \frac{143x}{262144} + \frac{\sin^{19}(x)\cos(x)}{20} - \frac{41\sin^{17}(x)\cos(x)}{360} + \frac{383\sin^{15}(x)\cos(x)}{5760} - \frac{\sin^{13}(x)\cos(x)}{5376} - \frac{13\sin^{11}(x)\cos(x)}{64512} - \frac{143\sin^9(x)\cos(x)}{645120} - \frac{143\sin^7(x)\cos(x)}{573440} - \frac{143\sin^5(x)\cos(x)}{491520} - \frac{143\sin^3(x)\cos(x)}{393216} - \frac{143\sin(x)\cos(x)}{262144}$$

$$\int \sin^{14} x \ \cos^8 \ x \, dx = \frac{91x}{524288} - \frac{\sin^{21}(x)\cos(x)}{22} + \frac{67\sin^{19}(x)\cos(x)}{440} - \frac{1367\sin^{17}(x)\cos(x)}{7920} + \frac{8441\sin^{15}(x)\cos(x)}{126720} - \frac{\sin^{13}(x)\cos(x)}{16896} - \frac{13\sin^{11}(x)\cos(x)}{202752} - \frac{13\sin^9(x)\cos(x)}{184320} - \frac{13\sin^7(x)\cos(x)}{163840} - \frac{91\sin^5(x)\cos(x)}{983040} - \frac{91\sin^3(x)\cos(x)}{786432} - \frac{91\sin(x)\cos(x)}{524288}$$

$$\int \sin^{14} x \ \cos^{10} \ x \, dx = \frac{273x}{4194304} + \frac{\sin^{23}(x)\cos(x)}{24} - \frac{97\sin^{21}(x)\cos(x)}{528} + \frac{1081\sin^{19}(x)\cos(x)}{3520} - \frac{1629\sin^{17}(x)\cos(x)}{7040} + \frac{7507\sin^{15}(x)\cos(x)}{112640} - \frac{\sin^{13}(x)\cos(x)}{45056} - \frac{13\sin^{11}(x)\cos(x)}{540672} - \frac{13\sin^9(x)\cos(x)}{491520} - \frac{39\sin^7(x)\cos(x)}{1310720} - \frac{91\sin^5(x)\cos(x)}{2621440} - \frac{91\sin^3(x)\cos(x)}{2097152} - \frac{273\sin(x)\cos(x)}{4194304}$$

$$\int \sin^{14} x \ \cos^{12} \ x \, dx = \frac{231x}{8388608} - \frac{\sin^{25}(x)\cos(x)}{26} + \frac{131\sin^{23}(x)\cos(x)}{624} - \frac{577\sin^{21}(x)\cos(x)}{1248} + \frac{4281\sin^{19}(x)\cos(x)}{8320} - \frac{4829\sin^{17}(x)\cos(x)}{16640} + \frac{17747\sin^{15}(x)\cos(x)}{266240} - \frac{\sin^{13}(x)\cos(x)}{106496} - \frac{\sin^{11}(x)\cos(x)}{98304} - \frac{11\sin^{9}(x)\cos(x)}{983040} - \frac{33\sin^{7}(x)\cos(x)}{2621440} - \frac{77\sin^{5}(x)\cos(x)}{5242880} - \frac{77\sin^{3}(x)\cos(x)}{4194304} - \frac{231\sin(x)\cos(x)}{8388608}$$

The patterns that can be seen from those equations above are

- The integral $\int \sin^{14} x \, dx$ represents some term for $\int \sin^{14} x \ \cos^2 \ x \, dx$ with denominator that is multiplied by $m + n = 14 + 2 = 16$.
- The higher the power of m the longer the new term that is having power of $\sin^{n+1}$ and above.
- As the iteration increases the denominator coefficient is the multiplication of $m + n$ with m starting from 2 to 12 with increment of 2.

This kind of integral is quite challenging, if we want to compute this integral with $n = 14$ and $m = 12$ we will start from $m = 0$, then $m = 2$. So let see when $m = 0$, we have computed and know the pattern for this kind of integral:

$$\int \sin^{14} x \ \cos^{0} \ x \, dx = \frac{429x}{2048} - \frac{\sin^{13}(x)\cos(x)}{14} - \frac{13\sin^{11}(x)\cos(x)}{168} - \frac{143\sin^{9}(x)\cos(x)}{1680} - \frac{429\sin^{7}(x)\cos(x)}{4480} - \frac{143\sin^{5}(x)\cos(x)}{1280} - \frac{143\sin^{3}(x)\cos(x)}{1024} - \frac{429\sin(x)\cos(x)}{2048}$$

now when $m = 2$, we will obtain this:

$$\int \sin^{14} x \ \cos^{2} \ x \, dx = \frac{429x}{32768} + \frac{\sin^{15}(x)\cos(x)}{16} - \frac{\sin^{13}(x)\cos(x)}{224} - \frac{13\sin^{11}(x)\cos(x)}{2688} - \frac{143\sin^{9}(x)\cos(x)}{26880} - \frac{429\sin^{7}(x)\cos(x)}{71680} - \frac{143\sin^{5}(x)\cos(x)}{20480} - \frac{143\sin^{3}(x)\cos(x)}{16384} - \frac{429\sin(x)\cos(x)}{32768}$$

From the reduction formula we know that this term:

$$\frac{c_{n/2}x}{d_{n/2}} + \sum_{i=1}^{n/2} - \frac{c_i \sin^{2i-1} x \ \cos \ x}{d_i}$$

will be multiplied by $\frac{m-1}{m+n}$ at every iteration from $m = 2$ till $m = 12$. So the problem is solved for this term.

For example, to obtain the denominator for function of x of 32768 from multiplying the denominator of function x from $\int \sin^{14} x \, dx$ with $d_0 = m + n = 16$, it is $2048 * 16 = 32768$.

The remaining term will have different pattern for the numerator, from now on we will omit the term of $\frac{c_{n/2}x}{d_{n/2}} + \sum_{i=1}^{n/2} -\frac{c_i \sin^{2i-1} x \ \cos \ x}{d_i}$, now take a look again:

$$\int \sin^{14} x \ \cos^2 \ x \, dx = \frac{\sin^{15}(x)\cos(x)}{16} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^4 \ x \, dx = -\frac{\sin^{17}(x)\cos(x)}{18} + \frac{19\sin^{15}(x)\cos(x)}{288} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^6 \ x \, dx = \frac{\sin^{19}(x)\cos(x)}{20} - \frac{41\sin^{17}(x)\cos(x)}{360} + \frac{383\sin^{15}(x)\cos(x)}{5760} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^8 \ x \, dx = -\frac{\sin^{21}(x)\cos(x)}{22} + \frac{67\sin^{19}(x)\cos(x)}{440} - \frac{1367\sin^{17}(x)\cos(x)}{7920} + \frac{8441\sin^{15}(x)\cos(x)}{126720} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^{10} \ x \, dx = \frac{\sin^{23}(x)\cos(x)}{24} - \frac{97\sin^{21}(x)\cos(x)}{528} + \frac{1081\sin^{19}(x)\cos(x)}{3520} - \frac{1629\sin^{17}(x)\cos(x)}{7040} + \frac{7507\sin^{15}(x)\cos(x)}{112640} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^{12} \ x \, dx = -\frac{\sin^{25}(x)\cos(x)}{26} + \frac{131\sin^{23}(x)\cos(x)}{624} - \frac{577\sin^{21}(x)\cos(x)}{1248} + \frac{4281\sin^{19}(x)\cos(x)}{8320} - \frac{4829\sin^{17}(x)\cos(x)}{16640} + \frac{17747\sin^{15}(x)\cos(x)}{266240} + \cdots - \ldots$$

We will learn how to decode this pattern, first we will make the denominator into its' raw number, so the numerator will change into its' raw number too, here it goes

$$\int \sin^{14} x \ \cos^2 \ x \, dx = \frac{\sin^{15}(x)\cos(x)}{16} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^4 \ x \, dx = -\frac{\sin^{17}(x)\cos(x)}{18} + \frac{19\sin^{15}(x)\cos(x)}{288} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^6 \ x \, dx = \frac{\sin^{19}(x)\cos(x)}{20} - \frac{41\sin^{17}(x)\cos(x)}{360} + \frac{383\sin^{15}(x)\cos(x)}{5760} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^8 \ x \, dx = -\frac{\sin^{21}(x)\cos(x)}{22} + \frac{67\sin^{19}(x)\cos(x)}{440} - \frac{1367\sin^{17}(x)\cos(x)}{7920} + \frac{8441\sin^{15}(x)\cos(x)}{126720} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^{10} \ x \, dx = \frac{\sin^{23}(x)\cos(x)}{24} - \frac{97\sin^{21}(x)\cos(x)}{528} + \frac{3243\sin^{19}(x)\cos(x)}{10560} - \frac{43983\sin^{17}(x)\cos(x)}{190080} + \frac{202689\sin^{15}(x)\cos(x)}{3041280} + \cdots - \ldots$$

$$\int \sin^{14} x \ \cos^{12} \ x \, dx = -\frac{\sin^{25}(x)\cos(x)}{26} + \frac{131\sin^{23}(x)\cos(x)}{624} - \frac{6347\sin^{21}(x)\cos(x)}{13728} + \frac{141273\sin^{19}(x)\cos(x)}{274560} - \frac{1434213\sin^{17}(x)\cos(x)}{4942080} + \frac{5270859\sin^{15}(x)\cos(x)}{79073280} + \cdots - \ldots$$

Then for a better look we will write it into Pascal' triangle form, for the numerator in the raw version

$$\begin{array}{ccccccccccc} & & & & & 1 & & & & & \\ & & & & 1 & & 19 & & & & \\ & & & 1 & & 41 & & 383 & & & \\ & & 1 & & 67 & & 1367 & & 8441 & & \\ & 1 & & 97 & & 3243 & & 43983 & & 202689 & \\ 1 & & 131 & & 6347 & & 141273 & & 1434213 & & 5270859 \end{array}$$

The C++ code will use an increasing **for** loop with $i = 1, i <= \frac{m}{2}, i = i + 1$.

For the computation, we are creating a variable for the denominator for function $\sin^{m+n-1} x \cos \ x$ as d_0, with $d_0 = m + n$. This variable will plays an important role to determine the value for the numerator of this kind of integral.

We will also use vector $\mathbf{v}$ to store the coefficient with

$$\begin{aligned} \mathbf{v}[0] &= 1 \\ \mathbf{v}[1]_i &= (d_0 * (i-1)) + 1, \quad i = 2, 3, \ldots, \frac{m}{2} - 1 \\ \mathbf{v}\left[\frac{m}{2} - 1\right]_i &= (d_0 * \mathbf{v}\left[\frac{m}{2} - 2\right]_{i-1}) + \textbf{factorialoddup}(i-1) \end{aligned}$$

$\mathbf{v}\left[\frac{m}{2} - 1\right]_i$ is the last entry for the vector at iteration i, while $\mathbf{v}\left[\frac{m}{2} - 2\right]_{i-1}$ is the last entry for the vector at the previous iteration $i - 1$.

We make a definition here of $\textbf{factorialoddup}(i-1)$:

$$\textbf{factorialoddup}(i-1) = 1 * 3 * 5 * \ldots$$

it is a multiplication of odd number from 1 upward till we get $i - 1$ terms.

$$\begin{aligned} \textbf{factorialoddup}(2) &= 1 * 3 \\ \textbf{factorialoddup}(3) &= 1 * 3 * 5 \\ \textbf{factorialoddup}(4) &= 1 * 3 * 5 * 7 \end{aligned}$$

For $i = 4$ we will make a **for** loop again to find the entry of the vector $\mathbf{v}[j]_i$ for $j = 2, 3, \ldots, \frac{m}{2} - 2$.

$$\begin{aligned} \mathbf{v}[j]_i &= (d_0 * \mathbf{v}[j-1]_{i-1}) + (k * \mathbf{v}[j-1]_{i-1} + \mathbf{v}[j]_{i-1}) \\ k &= k - 2 \end{aligned}$$

So the **for** loop will be like this:

$$
\begin{aligned}
i &= 1 \\
d_0 &= 16 \\
v[0] &= 1 \\
&* * * * * * * * * * * \\
i &= 2 \\
d_0 &= 18 \\
v[0] &= 1 \\
v[1] &= (d_0 * (2-1)) + 1 = 19 \\
&* * * * * * * * * * * \\
i &= 3 \\
d_0 &= 20 \\
v[0] &= 1 \\
v[1] &= (d_0 * (3-1)) + 1 = 41 \\
v[2] &= (d_0 * v[1]_{i=2}) + (1 * 3) = 383 \\
&* * * * * * * * * * * \\
i &= 4 \\
d_0 &= 22 \\
k &= 2 \\
v[0] &= 1 \\
v[1] &= (d_0 * (4-1)) + 1 = 67 \\
v[2] &= (d_0 * \mathbf{v}[1]_{i=3}) + (k * \mathbf{v}[1]_{i=3} + \mathbf{v}[2]_{i=3}) = 1367 \\
v[3] &= (d_0 * v[2]_{i=3}) + (1 * 3 * 5) = 8441
\end{aligned}
$$

Case 1-b: For $\int \sin^n x \ \cos^m \ x \, dx$ **with** $n < m$

Let's take a look at several integral related to this case

$$\int \sin^2 x \ \cos^4 \ x \, dx = \frac{x}{16} - \frac{\sin(x)\cos^5(x)}{6} + \frac{\sin(x)\cos^3(x)}{24} + \frac{\sin(x)\cos(x)}{16}$$

$$\int \sin^2 x \ \cos^6 \ x \, dx = \frac{5x}{128} - \frac{\sin(x)\cos^7(x)}{8} + \frac{\sin(x)\cos^5(x)}{48} + \frac{5\sin(x)\cos^3(x)}{192} + \frac{5\sin(x)\cos(x)}{128}$$

$$\int \sin^4 x \ \cos^6 \ x \, dx = \frac{3x}{256} + \frac{\sin(x)\cos^9(x)}{10} - \frac{11\sin(x)\cos^7(x)}{80} + \frac{\sin(x)\cos^5(x)}{160} + \frac{\sin(x)\cos^3(x)}{128} + \frac{3\sin(x)\cos(x)}{256}$$

$$\int \sin^2 x \ \cos^8 \ x \, dx = \frac{7x}{256} - \frac{\sin(x)\cos^9(x)}{10} + \frac{\sin(x)\cos^7(x)}{80} + \frac{7\sin(x)\cos^5(x)}{480} + \frac{7\sin(x)\cos^3(x)}{384} + \frac{7\sin(x)\cos(x)}{256}$$

$$\int \sin^4 x \ \cos^8 \ x \, dx = \frac{7x}{1024} + \frac{\sin(x)\cos^{11}(x)}{12} - \frac{13\sin(x)\cos^9(x)}{120} + \frac{\sin(x)\cos^7(x)}{320} + \frac{7\sin(x)\cos^5(x)}{1920} + \frac{7\sin(x)\cos^3(x)}{1536} + \frac{7\sin(x)\cos(x)}{1024}$$

$$\int \sin^6 x \ \cos^8 \ x \, dx = \frac{5x}{2048} - \frac{\sin(x)\cos^{13}(x)}{14} + \frac{29\sin(x)\cos^{11}(x)}{168} - \frac{37\sin(x)\cos^9(x)}{336} + \frac{\sin(x)\cos^7(x)}{896} + \frac{\sin(x)\cos^5(x)}{768} + \frac{5\sin(x)\cos^3(x)}{3072} + \frac{5\sin(x)\cos(x)}{2048}$$

This case does not need to be scrutinize further, since the numerator and denominator for all terms are exactly the same as those for the case 1-a where $n > m$, we only need to change the sine function into cosine and cosine function into sine in the result, since this case has the cosine function in the term that has the power. The plus and minus will also be interchange for the term with sine and cosine function, the term with x stays positive always.

Case 1-c: For $\int \sin^n x \ \cos^m \ x \, dx$ with $n = m$

These are the formula of $\int \sin^n x \ \cos^m \ x \, dx$ with $n = 1$ to $n = 8$.

$$\int \sin^1 x \ \cos^1 \ x \, dx = \frac{\sin^2 \ x}{2}$$
$$\int \sin^2 x \ \cos^2 \ x \, dx = \frac{x}{8} - \frac{\sin(2x)\cos(2x)}{16}$$
$$\int \sin^3 x \ \cos^3 \ x \, dx = -\frac{\sin^6 \ x}{6} + \frac{\sin^4 \ x}{4}$$
$$\int \sin^4 x \ \cos^4 \ x \, dx = \frac{3x}{128} - \frac{\sin^3(2x)\cos(2x)}{128} - \frac{3\sin(2x)\cos(2x)}{256}$$
$$\int \sin^5 x \ \cos^5 \ x \, dx = \frac{\sin^{10} \ x}{10} - \frac{\sin^8 \ x}{4} + \frac{\sin^6 \ x}{6}$$
$$\int \sin^6 x \ \cos^6 \ x \, dx = \frac{5x}{1024} - \frac{\sin^5(2x)\cos(2x)}{768} - \frac{5\sin^3(2x)\cos(2x)}{3072} - \frac{5\sin(2x)\cos(2x)}{2048}$$
$$\int \sin^7 x \ \cos^7 \ x \, dx = -\frac{\sin^{14} \ x}{14} + \frac{\sin^{12} \ x}{4} - \frac{3\sin^{10} \ x}{10} + \frac{\sin^8 \ x}{8}$$
$$\int \sin^8 x \ \cos^8 \ x \, dx = \frac{35x}{32768} - \frac{\sin^7(2x)\cos(2x)}{4096} - \frac{7\sin^5(2x)\cos(2x)}{24576} - \frac{35\sin^3(2x)\cos(2x)}{98304} - \frac{35\sin(2x)\cos(2x)}{65536}$$

We can tell that the case here will be divided into two, which are even case and odd case.

If n is even

$$\int \sin^2 x \ \cos^2 \ x \, dx = \frac{x}{8} - \frac{\sin(2x)\cos(2x)}{16}$$
$$\int \sin^4 x \ \cos^4 \ x \, dx = \frac{3x}{128} - \frac{\sin^3(2x)\cos(2x)}{128} - \frac{3\sin(2x)\cos(2x)}{256}$$
$$\int \sin^6 x \ \cos^6 \ x \, dx = \frac{5x}{1024} - \frac{\sin^5(2x)\cos(2x)}{768} - \frac{5\sin^3(2x)\cos(2x)}{3072} - \frac{5\sin(2x)\cos(2x)}{2048}$$
$$\int \sin^8 x \ \cos^8 \ x \, dx = \frac{35x}{32768} - \frac{\sin^7(2x)\cos(2x)}{4096} - \frac{7\sin^5(2x)\cos(2x)}{24576} - \frac{35\sin^3(2x)\cos(2x)}{98304} - \frac{35\sin(2x)\cos(2x)}{65536}$$

The patterns that can be seen from those equations above are

- The denominator with function of $\sin(2x)^{n-1}\cos(2x)$ can be determined with:

  $$d_0 = 2^n * (2n)$$

  This denominator will be used for the first integral result which is

  $$\frac{\sin^{n-1}(2x)\cos(2x)}{d_0}$$

  the next iteration will have the pattern like this:

  $$\frac{(n-1)\sin^{n-3}(2x)\cos(2x)}{d_0 * (n-2)}$$

The next iteration will have pattern like this:

$$\frac{(n-1)(n-3)\sin^{n-5}(2x)\cos(2x)}{d_0 * (n-2) * (n-4)}$$

Till we obtain the last iteration integral result with function $\sin(2x)\cos(2x)$ at the numerator we will stop. The easy thing here is the signs are all minus.

- The function x will have numerator and denominator of:

$$\frac{2 * (n-1) * (n-3) * \cdots * 1}{d_0 * (n-2) * (n-4) * \cdots * 1} * x$$

If n is odd

$$\begin{aligned}
\int \sin^1 x \ \cos^1 \ x \, dx &= \frac{\sin^2 \ x}{2} \\
\int \sin^3 x \ \cos^3 \ x \, dx &= -\frac{\sin^6 \ x}{6} + \frac{\sin^4 \ x}{4} \\
\int \sin^5 x \ \cos^5 \ x \, dx &= \frac{\sin^{10} \ x}{10} - \frac{\sin^8 \ x}{4} + \frac{\sin^6 \ x}{6} \\
\int \sin^7 x \ \cos^7 \ x \, dx &= -\frac{\sin^{14} \ x}{14} + \frac{\sin^{12} \ x}{4} - \frac{3\sin^{10} \ x}{10} + \frac{\sin^8 \ x}{8}
\end{aligned}$$

The patterns that can be seen from those equations above are

- The integral will have $\frac{n+1}{2}$ term/s, and have function of $\sin^{n+1} x$ at the numerator, at the next iteration we will have the function of $\sin^{(n+1)+2} x$ at the numerator with alternating sign, till we reach the $\sin^{2n} x$ at the numerator.
- The power determine the Pascal' triangle level / combinations formula that will be used to determine the numerator coefficients.
  For $n = 1$ we will use the first level of Pascal' triangle which is ${}_0C_0$.
  For $n = 3$ we will use the second level of Pascal' triangle which is ${}_1C_0$ and ${}_1C_1$.
  For $n = 5$ we will use the third level of Pascal' triangle which is ${}_2C_0, {}_2C_1$ and ${}_2C_2$.
  For $n = 7$ we will use the fourth level of Pascal' triangle which is ${}_3C_0, {}_3C_1, {}_3C_2$ and ${}_3C_3$.
- The denominator coefficient can be easily determined, it is the same as the power of the function in the numerator

Case 2: For $\int \sin^n x \ \cos^m \ x \, dx$ with at least one odd power in either n or m
This case occurs if either n or m is an odd number. We will recall the integral for $n = 1$,

$n = 2$, $n = 3$, $n = 5$,and $n = 6$

$$\int \sin^1 x \ \cos^2 \ x \, dx = \frac{-\cos^3 \ x}{3}$$
$$\int \sin^1 x \ \cos^3 \ x \, dx = \frac{-\cos^4 \ x}{4}$$
$$\int \sin^1 x \ \cos^4 \ x \, dx = \frac{-\cos^5 \ x}{5}$$
$$\int \sin^1 x \ \cos^5 \ x \, dx = \frac{-\cos^6 \ x}{6}$$
$$\int \sin^1 x \ \cos^6 \ x \, dx = \frac{-\cos^7 \ x}{7}$$
$$\int \sin^1 x \ \cos^7 \ x \, dx = \frac{-\cos^8 \ x}{8}$$
$$\int \sin^1 x \ \cos^8 \ x \, dx = \frac{-\cos^9 \ x}{9}$$

$$\int \sin^2 x \ \cos^3 \ x \, dx = -\frac{\sin^5 \ x}{5} + \frac{\sin^3 \ x}{3}$$
$$\int \sin^2 x \ \cos^5 \ x \, dx = \frac{\sin^7 \ x}{7} - \frac{2\sin^5 \ x}{5} + \frac{\sin^3 \ x}{3}$$
$$\int \sin^2 x \ \cos^7 \ x \, dx = -\frac{\sin^9 \ x}{9} + \frac{3\sin^7 \ x}{7} - \frac{3\sin^5 \ x}{5} + \frac{\sin^3 \ x}{3}$$

$$\int \sin^3 x \ \cos^1 \ x \, dx = \frac{\sin^4 \ x}{4}$$
$$\int \sin^3 x \ \cos^2 \ x \, dx = \frac{\cos^5 \ x}{5} - \frac{\cos^3 \ x}{3}$$
$$\int \sin^3 x \ \cos^4 \ x \, dx = \frac{\cos^7 \ x}{7} - \frac{\cos^5 \ x}{5}$$
$$\int \sin^3 x \ \cos^5 \ x \, dx = \frac{\cos^8 \ x}{8} - \frac{\cos^6 \ x}{6}$$
$$\int \sin^3 x \ \cos^6 \ x \, dx = \frac{\cos^9 \ x}{9} - \frac{\cos^7 \ x}{7}$$
$$\int \sin^3 x \ \cos^7 \ x \, dx = \frac{\cos^{10} \ x}{10} - \frac{\cos^8 \ x}{8}$$
$$\int \sin^3 x \ \cos^8 \ x \, dx = \frac{\cos^{11} \ x}{11} - \frac{\cos^9 \ x}{9}$$

$$\int \sin^5 x \ \cos^1 \ x \, dx = \frac{\sin^6 \ x}{6}$$
$$\int \sin^5 x \ \cos^2 \ x \, dx = -\frac{\cos^7 \ x}{7} + \frac{2\cos^5 \ x}{5} - \frac{\cos^3 \ x}{3}$$
$$\int \sin^5 x \ \cos^3 \ x \, dx = -\frac{\sin^8 \ x}{8} + \frac{\sin^6 \ x}{6}$$
$$\int \sin^5 x \ \cos^4 \ x \, dx = -\frac{\cos^9 \ x}{9} + \frac{2\cos^7 \ x}{7} - \frac{\cos^5 \ x}{5}$$
$$\int \sin^5 x \ \cos^6 \ x \, dx = -\frac{\cos^{11} \ x}{11} + \frac{2\cos^9 \ x}{9} - \frac{\cos^7 \ x}{7}$$
$$\int \sin^5 x \ \cos^7 \ x \, dx = -\frac{\cos^{12} \ x}{12} + \frac{\cos^{10} \ x}{5} - \frac{\cos^8 \ x}{8}$$
$$\int \sin^5 x \ \cos^8 \ x \, dx = -\frac{\cos^{13} \ x}{13} + \frac{2\cos^{11} \ x}{11} - \frac{\cos^9 \ x}{9}$$

$$\int \sin^6 x \ \cos^1 \ x \, dx = \frac{\sin^7 \ x}{7}$$
$$\int \sin^6 x \ \cos^3 \ x \, dx = -\frac{\sin^9 \ x}{9} + \frac{\sin^7 \ x}{7}$$
$$\int \sin^6 x \ \cos^5 \ x \, dx = \frac{\sin^{11} \ x}{11} - \frac{2\sin^9 \ x}{9} + \frac{\sin^7 \ x}{7}$$
$$\int \sin^6 x \ \cos^7 \ x \, dx = -\frac{\sin^{13} \ x}{13} + \frac{3\sin^{11} \ x}{11} - \frac{\sin^9 \ x}{3} + \frac{\sin^7 \ x}{7}$$

There will be two big cases here which are:

1. When $m > n$

2. When $m \leq n$

If $m > n$

$$\int \sin^1 x \ \cos^2 \ x \, dx = \frac{-\cos^3 \ x}{3}$$
$$\int \sin^1 x \ \cos^3 \ x \, dx = \frac{-\cos^4 \ x}{4}$$
$$\int \sin^1 x \ \cos^4 \ x \, dx = \frac{-\cos^5 \ x}{5}$$
$$\int \sin^1 x \ \cos^5 \ x \, dx = \frac{-\cos^6 \ x}{6}$$
$$\int \sin^1 x \ \cos^6 \ x \, dx = \frac{-\cos^7 \ x}{7}$$
$$\int \sin^1 x \ \cos^7 \ x \, dx = \frac{-\cos^8 \ x}{8}$$
$$\int \sin^1 x \ \cos^8 \ x \, dx = \frac{-\cos^9 \ x}{9}$$
$$\int \sin^2 x \ \cos^3 \ x \, dx = -\frac{\sin^5 \ x}{5} + \frac{\sin^3 \ x}{3}$$
$$\int \sin^2 x \ \cos^5 \ x \, dx = \frac{\sin^7 \ x}{7} - \frac{2\sin^5 \ x}{5} + \frac{\sin^3 \ x}{3}$$
$$\int \sin^2 x \ \cos^7 \ x \, dx = -\frac{\sin^9 \ x}{9} + \frac{3\sin^7 \ x}{7} - \frac{3\sin^5 \ x}{5} + \frac{\sin^3 \ x}{3}$$
$$\int \sin^3 x \ \cos^4 \ x \, dx = \frac{\cos^7 \ x}{7} - \frac{\cos^5 \ x}{5}$$
$$\int \sin^3 x \ \cos^5 \ x \, dx = \frac{\cos^8 \ x}{8} - \frac{\cos^6 \ x}{6}$$
$$\int \sin^5 x \ \cos^6 \ x \, dx = -\frac{\cos^{11} \ x}{11} + \frac{2\cos^9 \ x}{9} - \frac{\cos^7 \ x}{7}$$
$$\int \sin^5 x \ \cos^7 \ x \, dx = -\frac{\cos^{12} \ x}{12} + \frac{\cos^{10} \ x}{5} - \frac{\cos^8 \ x}{8}$$
$$\int \sin^5 x \ \cos^8 \ x \, dx = -\frac{\cos^{13} \ x}{13} + \frac{2\cos^{11} \ x}{11} - \frac{\cos^9 \ x}{9}$$
$$\int \sin^6 x \ \cos^7 \ x \, dx = -\frac{\sin^{13} \ x}{13} + \frac{3\sin^{11} \ x}{11} - \frac{\sin^9 \ x}{3} + \frac{\sin^7 \ x}{7}$$

The patterns that can be seen from those equations above are

- The integral will have the result with function of cosine, if n is an odd number.

  The integral will have the result of sine, if n is an even number.
- If n is an even number, then the pattern will be like this for the integral:

$$\int \sin^n x \ \cos^m \ x \, dx = \text{sgn}_0 * {}_{\frac{m-1}{2}}C_0 \frac{\sin^{n+1} \ x}{n+1} + \text{sgn}_1 * {}_{\frac{m-1}{2}}C_1 \frac{\sin^{n+3} \ x}{n+3} + \ldots$$
$$+ \text{sgn}_{\frac{m-1}{2}} * {}_{\frac{m-1}{2}}C_{\frac{m-1}{2}} \frac{\sin^{n+m} \ x}{n+m}$$

We start with $\text{sgn}_0 = 1$ and it is alternating at every iteration.

- If n is an odd number, then the pattern will be like this for the integral:

$$\int \sin^n x \ \cos^m \ x \, dx = \text{sgn}_0 * {}_{\frac{n-1}{2}}C_0 \frac{\cos^{m+1} \ x}{m+1} + \text{sgn}_1 * {}_{\frac{n-1}{2}}C_1 \frac{\cos^{m+3} \ x}{m+3} + \dots$$
$$\text{sgn}_{\frac{n-1}{2}} * + {}_{\frac{n-1}{2}}C_{\frac{n-1}{2}} \frac{\cos^{n+m} \ x}{n+m}$$

We start with $\text{sgn}_0 = -1$ and it is alternating at every iteration.

If $m \leq n$

$$\int \sin^3 x \ \cos^1 \ x \, dx = \frac{\sin^4 \ x}{4}$$
$$\int \sin^3 x \ \cos^2 \ x \, dx = \frac{\cos^5 \ x}{5} - \frac{\cos^3 \ x}{3}$$
$$\int \sin^5 x \ \cos^1 \ x \, dx = \frac{\sin^6 \ x}{6}$$
$$\int \sin^5 x \ \cos^2 \ x \, dx = -\frac{\cos^7 \ x}{7} + \frac{2\cos^5 \ x}{5} - \frac{\cos^3 \ x}{3}$$
$$\int \sin^5 x \ \cos^3 \ x \, dx = -\frac{\sin^8 \ x}{8} + \frac{\sin^6 \ x}{6}$$
$$\int \sin^5 x \ \cos^4 \ x \, dx = -\frac{\cos^9 \ x}{9} + \frac{2\cos^7 \ x}{7} - \frac{\cos^5 \ x}{5}$$
$$\int \sin^6 x \ \cos^1 \ x \, dx = \frac{\sin^7 \ x}{7}$$
$$\int \sin^6 x \ \cos^3 \ x \, dx = -\frac{\sin^9 \ x}{9} + \frac{\sin^7 \ x}{7}$$
$$\int \sin^6 x \ \cos^5 \ x \, dx = \frac{\sin^{11} \ x}{11} - \frac{2\sin^9 \ x}{9} + \frac{\sin^7 \ x}{7}$$

The patterns that can be seen from those equations above are

- If m is an even number, then the integral will be the function of cosine.

  If m is an odd number, then the integral will be the function of sine.
- If m is an even number, then the pattern will be like this for the integral:

$$\int \sin^n x \ \cos^m \ x \, dx = \text{sgn}_0 * {}_{\frac{n-1}{2}}C_0 \frac{\cos^{m+1} \ x}{m+1} + \text{sgn}_1 * {}_{\frac{n-1}{2}}C_1 \frac{\cos^{m+3} \ x}{m+3} + \dots$$
$$+ \text{sgn}_{\frac{n-1}{2}} * {}_{\frac{n-1}{2}}C_{\frac{m}{2}} \frac{\cos^{n+m} \ x}{n+m}$$

We start with $\text{sgn}_0 = -1$ and it is alternating at every iteration.

- If m is an odd number, then the pattern will be like this for the integral:

$$\int \sin^n x \ \cos^m \ x \, dx = \text{sgn}_0 * {}_{\frac{m-1}{2}}C_0 \frac{\sin^{n+1} \ x}{n+1} + \text{sgn}_1 * {}_{\frac{m-1}{2}}C_1 \frac{\sin^{n+3} \ x}{n+3} + \dots$$
$$+ \text{sgn}_{\frac{m-1}{2}} * {}_{\frac{m-1}{2}}C_{\frac{m-1}{2}} \frac{\sin^{n+m} \ x}{n+m}$$

We start with $\text{sgn}_0 = -1$ and it is alternating at every iteration.

viii. $\int \tan^n x \ \sec^m \ x \, dx$

These are the formula of $\int \tan^n x \ \sec^m \ x \, dx$ with $n = 1$ and $m = 1$ to $m = 8$.

$$\int \tan^1 x \ \sec^1 \ x \, dx = \frac{1}{\cos(x)}$$
$$\int \tan^1 x \ \sec^2 \ x \, dx = \frac{1}{2\cos^2(x)}$$
$$\int \tan^1 x \ \sec^3 \ x \, dx = \frac{1}{3\cos^3(x)}$$
$$\int \tan^1 x \ \sec^4 \ x \, dx = \frac{1}{4\cos^4(x)}$$
$$\int \tan^1 x \ \sec^5 \ x \, dx = \frac{1}{5\cos^5(x)}$$
$$\int \tan^1 x \ \sec^6 \ x \, dx = \frac{1}{6\cos^6(x)}$$
$$\int \tan^1 x \ \sec^7 \ x \, dx = \frac{1}{7\cos^7(x)}$$
$$\int \tan^1 x \ \sec^8 \ x \, dx = \frac{1}{8\cos^8(x)}$$

These are the formula of $\int \tan^n x \ \sec^m \ x \, dx$ with $n = 2$ and $m = 1$ to $m = 8$.

$$\int \tan^2 x \ \sec^1 \ x \, dx = \frac{\log(\sin(x)-1)}{4} - \frac{\log(\sin(x)+1)}{4} - \frac{\sin(x)}{2\sin^2(x)-2}$$
$$\int \tan^2 x \ \sec^2 \ x \, dx = -\frac{\sin(x)}{3\cos(x)} + \frac{\sin(x)}{3\cos^3(x)}$$
$$\int \tan^2 x \ \sec^3 \ x \, dx = \frac{\log(\sin(x)-1)}{16} - \frac{\log(\sin(x)+1)}{16} - \frac{-\sin^3(x)-\sin(x)}{8\sin^4(x)-16\sin^2(x)+8}$$
$$\int \tan^2 x \ \sec^4 \ x \, dx = -\frac{2\sin(x)}{15\cos(x)} - \frac{\sin(x)}{15\cos^3(x)} + \frac{\sin(x)}{5\cos^5(x)}$$
$$\int \tan^2 x \ \sec^5 \ x \, dx = \frac{\log(\sin(x)-1)}{32} - \frac{\log(\sin(x)+1)}{32} + \frac{3\sin^5(x)-8\sin^3(x)-3\sin(x)}{48\sin^6(x)-144\sin^4(x)+144\sin^2(x)-48}$$
$$\int \tan^2 x \ \sec^6 \ x \, dx = -\frac{8\sin(x)}{105\cos(x)} - \frac{4\sin(x)}{105\cos^3(x)} - \frac{\sin(x)}{35\cos^5(x)} + \frac{\sin(x)}{7\cos^7(x)}$$
$$\int \tan^2 x \ \sec^7 \ x \, dx = \frac{5\log(\sin(x)-1)}{256} - \frac{5\log(\sin(x)+1)}{256} - \frac{-15\sin^7(x)+55\sin^5(x)-73\sin^3(x)-15\sin(x)}{384\sin^8(x)-1536\sin^6(x)+2304\sin^4(x)-1536\sin^2(x)+384}$$
$$\int \tan^2 x \ \sec^8 \ x \, dx = -\frac{16\sin(x)}{315\cos(x)} - \frac{8\sin(x)}{315\cos^3(x)} - \frac{2\sin(x)}{105\cos^5(x)} - \frac{\sin(x)}{63\cos^7(x)} + \frac{\sin(x)}{9\cos^9(x)}$$

These are the formula of $\int \tan^n x \ \sec^m \ x \, dx$ with $n = 3$ and $m = 1$ to $m = 8$.

$$\int \tan^3 x \ \sec^1 \ x \, dx = \frac{1 - 3\cos^2(x)}{3\cos^3(x)}$$
$$\int \tan^3 x \ \sec^2 \ x \, dx = \frac{1 - 2\cos^2(x)}{4\cos^4(x)}$$
$$\int \tan^3 x \ \sec^3 \ x \, dx = \frac{3 - 5\cos^2(x)}{15\cos^5(x)}$$
$$\int \tan^3 x \ \sec^4 \ x \, dx = \frac{2 - 3\cos^2(x)}{12\cos^6(x)}$$
$$\int \tan^3 x \ \sec^5 \ x \, dx = \frac{5 - 7\cos^2(x)}{35\cos^7(x)}$$
$$\int \tan^3 x \ \sec^6 \ x \, dx = \frac{3 - 4\cos^2(x)}{24\cos^8(x)}$$
$$\int \tan^3 x \ \sec^7 \ x \, dx = \frac{7 - 9\cos^2(x)}{63\cos^9(x)}$$
$$\int \tan^3 x \ \sec^8 \ x \, dx = \frac{4 - 5\cos^2(x)}{40\cos^{10}(x)}$$

These are the formula of $\int \tan^n x \ \sec^m \ x \, dx$ with $n = 4$ and $m = 1$ to $m = 8$.

$$\int \tan^4 x \ \sec^1 \ x \, dx = -\frac{3\log(\sin(x) - 1)}{16} + \frac{3\log(\sin(x) + 1)}{16} - \frac{-5\sin^3(x) + 3\sin(x)}{8\sin^4(x) - 16\sin^2(x) + 8}$$
$$\int \tan^4 x \ \sec^2 \ x \, dx = \frac{\sin(x)}{5\cos(x)} - \frac{2\sin(x)}{5\cos^3(x)} + \frac{\sin(x)}{5\cos^5(x)}$$
$$\int \tan^4 x \ \sec^3 \ x \, dx = -\frac{\log(\sin(x) - 1)}{32} + \frac{\log(\sin(x) + 1)}{32} - \frac{-3\sin^5(x) - 8\sin^3(x) + 3\sin(x)}{48\sin^6(x) - 144\sin^4(x) + 144\sin^2(x) - 48}$$
$$\int \tan^4 x \ \sec^4 \ x \, dx = \frac{2\sin(x)}{35\cos(x)} + \frac{\sin(x)}{35\cos^3(x)} - \frac{8\sin(x)}{35\cos^5(x)} + \frac{\sin(x)}{7\cos^7(x)}$$
$$\int \tan^4 x \ \sec^5 \ x \, dx = -\frac{3\log(\sin(x) - 1)}{256} + \frac{3\log(\sin(x) + 1)}{256} - \frac{3\sin^7(x) - 11\sin^5(x) - 11\sin^3(x) + 3\sin(x)}{128\sin^8(x) - 512\sin^6(x) + 768\sin^4(x) - 512\sin^2(x) + 128}$$
$$\int \tan^4 x \ \sec^6 \ x \, dx = \frac{8\sin(x)}{315\cos(x)} + \frac{4\sin(x)}{315\cos^3(x)} + \frac{\sin(x)}{105\cos^5(x)} - \frac{10\sin(x)}{63\cos^7(x)} + \frac{\sin(x)}{9\cos^9(x)}$$
$$\int \tan^4 x \ \sec^7 \ x \, dx = -\frac{3\log(\sin(x) - 1)}{512} + \frac{3\log(\sin(x) + 1)}{512} + \frac{-15\sin^9(x) + 70\sin^7(x) - 128\sin^5(x) - 70\sin^3(x) + 15\sin(x)}{1280\sin^{10}(x) - 6400\sin^8(x) + 12800\sin^6(x) - 12800\sin^4(x) + 6400\sin^2(x) - 1280}$$
$$\int \tan^4 x \ \sec^8 \ x \, dx = \frac{16\sin(x)}{1155\cos(x)} + \frac{8\sin(x)}{1155\cos^3(x)} + \frac{2\sin(x)}{385\cos^5(x)} + \frac{\sin(x)}{231\cos^7(x)} - \frac{4\sin(x)}{33\cos^9(x)} + \frac{\sin(x)}{11\cos^{11}(x)}$$

These are the formula of $\int \tan^n x \ \sec^m \ x \, dx$ with $n = 5$ and $m = 1$ to $m = 8$.

$$\int \tan^5 x \ \sec^1 \ x \, dx = \frac{-(-15\cos^4(x) + 10\cos^2(x) - 3)}{15\cos^5(x)}$$
$$\int \tan^5 x \ \sec^2 \ x \, dx = \frac{-(-3\cos^4(x) + 3\cos^2(x) - 1)}{6\cos^6(x)}$$
$$\int \tan^5 x \ \sec^3 \ x \, dx = \frac{-(-35\cos^4(x) + 42\cos^2(x) - 15)}{105\cos^7(x)}$$
$$\int \tan^5 x \ \sec^4 \ x \, dx = \frac{-(-6\cos^4(x) + 8\cos^2(x) - 3)}{24\cos^8(x)}$$
$$\int \tan^5 x \ \sec^5 \ x \, dx = \frac{-(-63\cos^4(x) + 90\cos^2(x) - 35)}{315\cos^9(x)}$$
$$\int \tan^5 x \ \sec^6 \ x \, dx = \frac{-(-10\cos^4(x) + 15\cos^2(x) - 6)}{60\cos^{10}(x)}$$
$$\int \tan^5 x \ \sec^7 \ x \, dx = \frac{-(-99\cos^4(x) + 154\cos^2(x) - 63)}{693\cos^{11}(x)}$$
$$\int \tan^5 x \ \sec^8 \ x \, dx = \frac{-(-15\cos^4(x) + 24\cos^2(x) - 10)}{120\cos^{12}(x)}$$

These are the formula of $\int \tan^n x \ \sec^m \ x \, dx$ with $n = 6$ and $m = 1$ to $m = 8$.

$$\int \tan^6 x \ \sec^1 \ x \, dx = \frac{5\log(\sin(x) - 1)}{32} - \frac{5\log(\sin(x) + 1)}{32} + \frac{-33\sin^5(x) + 40\sin^3(x) - 15\sin(x)}{48\sin^6(x) - 144\sin^4(x) + 144\sin^2(x) - 48}$$
$$\int \tan^6 x \ \sec^2 \ x \, dx = -\frac{\sin(x)}{7\cos(x)} + \frac{3\sin(x)}{7\cos^3(x)} - \frac{3\sin(x)}{7\cos^5(x)} + \frac{\sin(x)}{7\cos^7(x)}$$
$$\int \tan^6 x \ \sec^3 \ x \, dx = \frac{5\log(\sin(x) - 1)}{256} - \frac{5\log(\sin(x) + 1)}{256} - \frac{-15\sin^7(x) - 73\sin^5(x) + 55\sin^3(x) - 15\sin(x)}{384\sin^8(x) - 1536\sin^6(x) + 2304\sin^4(x) - 1536\sin^2(x) + 384}$$
$$\int \tan^6 x \ \sec^4 \ x \, dx = -\frac{2\sin(x)}{63\cos(x)} - \frac{\sin(x)}{63\cos^3(x)} + \frac{5\sin(x)}{21\cos^5(x)} - \frac{19\sin(x)}{63\cos^7(x)} + \frac{\sin(x)}{9\cos^9(x)}$$
$$\int \tan^6 x \ \sec^5 \ x \, dx = \frac{3\log(\sin(x) - 1)}{512} - \frac{3\log(\sin(x) + 1)}{512} + \frac{15\sin^9(x) - 70\sin^7(x) - 128\sin^5(x) + 70\sin^3(x) - 15\sin(x)}{1280\sin^{10}(x) - 6400\sin^8(x) + 12800\sin^6(x) - 12800\sin^4(x) + 6400\sin^2(x) - 1280}$$
$$\int \tan^6 x \ \sec^6 \ x \, dx = -\frac{8\sin(x)}{693\cos(x)} - \frac{4\sin(x)}{693\cos^3(x)} - \frac{\sin(x)}{231\cos^5(x)} + \frac{113\sin(x)}{693\cos^7(x)} - \frac{23\sin(x)}{99\cos^9(x)} + \frac{\sin(x)}{11\cos^{11}(x)}$$
$$\int \tan^6 x \ \sec^7 \ x \, dx = \frac{5\log(\sin(x) - 1)}{2048} - \frac{5\log(\sin(x) + 1)}{2048}$$

$$- \frac{-15\sin^{11}(x) + 85\sin^9(x) - 198\sin^7(x) - 198\sin^5(x) + 85\sin^3(x) - 15\sin(x)}{3072\sin^{12}(x) - 18432\sin^{10}(x) + 46080\sin^8(x) - 61440\sin^6(x) + 46080\sin^4(x) - 18432\sin^2(x) + 3072}$$

$$\int \tan^6 x \ \sec^8 \ x \, dx = -\frac{16\sin(x)}{3003\cos(x)} - \frac{8\sin(x)}{3003\cos^3(x)} - \frac{2\sin(x)}{1001\cos^5(x)} - \frac{5\sin(x)}{3003\cos^7(x)} + \frac{53\sin(x)}{429\cos^9(x)} - \frac{27\sin(x)}{143\cos^{11}(x)} + \frac{\sin(x)}{13\cos^{13}(x)}$$

These are the formula of $\int \tan^n x \ \sec^m \ x \, dx$ with $n = 7$ and $m = 1$ to $m = 8$.

$$\int \tan^7 x \ \sec^1 \ x \, dx = \frac{-35\cos^6(x) + 35\cos^4(x) - 21\cos^2(x) + 5}{35\cos^7(x)}$$
$$\int \tan^7 x \ \sec^2 \ x \, dx = \frac{-4\cos^6(x) + 6\cos^4(x) - 4\cos^2(x) + 1}{8\cos^8(x)}$$
$$\int \tan^7 x \ \sec^3 \ x \, dx = \frac{-105\cos^6(x) + 189\cos^4(x) - 135\cos^2(x) + 35}{315\cos^9(x)}$$
$$\int \tan^7 x \ \sec^4 \ x \, dx = \frac{-10\cos^6(x) + 20\cos^4(x) - 15\cos^2(x) + 4}{40\cos^{10}(x)}$$
$$\int \tan^7 x \ \sec^5 \ x \, dx = \frac{-231\cos^6(x) + 495\cos^4(x) - 385\cos^2(x) + 105}{1155\cos^{11}(x)}$$
$$\int \tan^7 x \ \sec^6 \ x \, dx = \frac{-20\cos^6(x) + 45\cos^4(x) - 36\cos^2(x) + 10}{120\cos^{12}(x)}$$
$$\int \tan^7 x \ \sec^7 \ x \, dx = \frac{-429\cos^6(x) + 1001\cos^4(x) - 819\cos^2(x) + 231}{3003\cos^{13}(x)}$$
$$\int \tan^7 x \ \sec^8 \ x \, dx = \frac{-35\cos^6(x) + 84\cos^4(x) - 70\cos^2(x) + 20}{280\cos^{14}(x)}$$

Case 1: For $\int \tan^n x \ \sec^m \ x \, dx$ with n even
We divide this case into 2 smaller cases:

1. When m even

2. When m odd

Case 1-a: For $\int \tan^n x \ \sec^m \ x \, dx$ with n even and m even

$$\int \tan^2 x \ \sec^2 \ x \, dx = -\frac{\sin(x)}{3\cos(x)} + \frac{\sin(x)}{3\cos^3(x)}$$

$$\int \tan^2 x \ \sec^4 \ x \, dx = -\frac{2\sin(x)}{15\cos(x)} - \frac{\sin(x)}{15\cos^3(x)} + \frac{\sin(x)}{5\cos^5(x)}$$

$$\int \tan^2 x \ \sec^6 \ x \, dx = -\frac{8\sin(x)}{105\cos(x)} - \frac{4\sin(x)}{105\cos^3(x)} - \frac{\sin(x)}{35\cos^5(x)} + \frac{\sin(x)}{7\cos^7(x)}$$

$$\int \tan^2 x \ \sec^8 \ x \, dx = -\frac{16\sin(x)}{315\cos(x)} - \frac{8\sin(x)}{315\cos^3(x)} - \frac{2\sin(x)}{105\cos^5(x)} - \frac{\sin(x)}{63\cos^7(x)} + \frac{\sin(x)}{9\cos^9(x)}$$

$$\int \tan^4 x \ \sec^2 \ x \, dx = \frac{\sin(x)}{5\cos(x)} - \frac{2\sin(x)}{5\cos^3(x)} + \frac{\sin(x)}{5\cos^5(x)}$$

$$\int \tan^4 x \ \sec^4 \ x \, dx = \frac{2\sin(x)}{35\cos(x)} + \frac{\sin(x)}{35\cos^3(x)} - \frac{8\sin(x)}{35\cos^5(x)} + \frac{\sin(x)}{7\cos^7(x)}$$

$$\int \tan^4 x \ \sec^6 \ x \, dx = \frac{8\sin(x)}{315\cos(x)} + \frac{4\sin(x)}{315\cos^3(x)} + \frac{\sin(x)}{105\cos^5(x)} - \frac{10\sin(x)}{63\cos^7(x)} + \frac{\sin(x)}{9\cos^9(x)}$$

$$\int \tan^4 x \ \sec^8 \ x \, dx = \frac{16\sin(x)}{1155\cos(x)} + \frac{8\sin(x)}{1155\cos^3(x)} + \frac{2\sin(x)}{385\cos^5(x)} + \frac{\sin(x)}{231\cos^7(x)} - \frac{4\sin(x)}{33\cos^9(x)} + \frac{\sin(x)}{11\cos^{11}(x)}$$

$$\int \tan^6 x \ \sec^2 \ x \, dx = -\frac{\sin(x)}{7\cos(x)} + \frac{3\sin(x)}{7\cos^3(x)} - \frac{3\sin(x)}{7\cos^5(x)} + \frac{\sin(x)}{7\cos^7(x)}$$

$$\int \tan^6 x \ \sec^4 \ x \, dx = -\frac{2\sin(x)}{63\cos(x)} - \frac{\sin(x)}{63\cos^3(x)} + \frac{5\sin(x)}{21\cos^5(x)} - \frac{19\sin(x)}{63\cos^7(x)} + \frac{\sin(x)}{9\cos^9(x)}$$

$$\int \tan^6 x \ \sec^6 \ x \, dx = -\frac{8\sin(x)}{693\cos(x)} - \frac{4\sin(x)}{693\cos^3(x)} - \frac{\sin(x)}{231\cos^5(x)} + \frac{113\sin(x)}{693\cos^7(x)} - \frac{23\sin(x)}{99\cos^9(x)} + \frac{\sin(x)}{11\cos^{11}(x)}$$

$$\int \tan^6 x \ \sec^8 \ x \, dx = -\frac{16\sin(x)}{3003\cos(x)} - \frac{8\sin(x)}{3003\cos^3(x)} - \frac{2\sin(x)}{1001\cos^5(x)} - \frac{5\sin(x)}{3003\cos^7(x)} + \frac{53\sin(x)}{429\cos^9(x)} - \frac{27\sin(x)}{143\cos^{11}(x)} + \frac{\sin(x)}{13\cos^{13}(x)}$$

The patterns that can be seen from those equations above are

- The series are consisting of a function with $\sin(x)$ as the numerator and $\cos(x)$ as the denominator, the length of the series and the maximum power for the denominator depending on m and n.
- If you take a look at the $\int \sec^m \ dx$ for m even, you will notice that this case will use the result of $\int \sec^m \ dx$ for m even depending on n and using the combination to make the Pascal' triangle.

For example, let $n = 4$ and $m = 6$

$$\begin{aligned}\int \tan^4 x \ \sec^6 \ x \, dx &= \int (\sec^2 x - 1)^2 \ \sec^6 \ x \, dx \\ &= \int (\sec^4 x - 2\sec^2 x + 1) \sec^6 \ x \, dx \\ &= \int \sec^{10} x - 2\sec^8 x + \sec^6 \ x \, dx \\ &= \frac{8\sin(x)}{315\cos(x)} + \frac{4\sin(x)}{315\cos^3(x)} + \frac{\sin(x)}{105\cos^5(x)} - \frac{10\sin(x)}{63\cos^7(x)} + \frac{\sin(x)}{9\cos^9(x)}\end{aligned}$$

In the end we will refer to the formula for $\int \sec^m \ dx$ for m even that starts the power of the secant and end at the total power of the secant and tangent, with.

```
#include <iostream>
#include "symintegrationc++.h"
#include <vector>

using namespace std;

// Factorial and combinations
Symbolic factorial(int n) {
	if (n <= 1)
	{
		return 1;
	}
	Symbolic result = 1;
	for (int i = 2; i <= n; ++i)
	{
		result *= i;
	}
	return result;
}

Symbolic combinations(int n, int r) {
	if (r < 0 || r > n)
	{
		return 0; // Invalid input
	}
	return factorial(n) / (factorial(r) * factorial(n - r));
}

Symbolic integralsecanteven(int n) {
	Symbolic x("x");
	int v[999];
	v[0] = 1;
	Symbolic integral;
	Symbolic d0 = 1;
	int k = 1, l = 2, c=1;
	for(int i = 1 ; i < n ; i = i+2) // to compute the denominator
```

```
        {
                d0 *= i;
        }
        for(int i = 1 ; i < n — 1; i = i+2)
        {
                c = v[0];
                int arrsec[999]; // make the size of the array as big as
                    possible

                for(int j = 0 ; j < i—1 ; j = j+1)
                {
                        arrsec[j] = v[j];
                }
                int d;
                for(int j = 1 ; j < i ; j = j+1)
                {
                        d = arrsec[j—1];
                        v[j] = d*l;
                }

                v[0] = c*k;
                v[1] = c*l;

                k= k + 2;
                l = l + 2;
        }

        int j_d = 1 ;
        for(int i = 1 ; i < (0.5*n)+1; i = i+1)
        {
                integral += ( v[i—1] * sin(x) ) / ( d0*(cos(x)^(n—j_d)) )
                    ;
                j_d = j_d+2;
        }
        return integral;
}

int main(void)
{
        Symbolic x("x");
        int bpowersec = 4;
        int bpowertan = 6;
        int bpower = bpowersec+bpowertan;
        Symbolic result;
        Symbolic sgn = 1;
        int j = 1;
        int m = bpowertan/2;
        for(int i = bpower ; i >= bpowersec ; i = i—2)
```

```
        {
                result += sgn*combinations(m,j-1)*integralsecanteven(i);
                sgn = -sgn;
                j = j+1;
        }
        cout << "integral = "<< result << endl;
        //cout << "integral with subs = " << result[x==1] << endl;
        return 0;
}
```

Code 24: *level 4 integral for tangent even power and secant even power case*

Case 1-b: For $\int \tan^n x \ \sec^m \ x \, dx$ with n even and m odd

$$\int \tan^2 x \ \sec^1 \ x \, dx = \frac{\log(\sin(x)-1)}{4} - \frac{\log(\sin(x)+1)}{4} - \frac{\sin(x)}{2\sin^2(x)-2}$$

$$\int \tan^2 x \ \sec^3 \ x \, dx = \frac{\log(\sin(x)-1)}{16} - \frac{\log(\sin(x)+1)}{16} - \frac{-\sin^3(x)-\sin(x)}{8\sin^4(x)-16\sin^2(x)+8}$$

$$\int \tan^2 x \ \sec^5 \ x \, dx = \frac{\log(\sin(x)-1)}{32} - \frac{\log(\sin(x)+1)}{32} + \frac{3\sin^5(x)-8\sin^3(x)-3\sin(x)}{48\sin^6(x)-144\sin^4(x)+144\sin^2(x)-48}$$

$$\int \tan^2 x \ \sec^7 \ x \, dx = \frac{5\log(\sin(x)-1)}{256} - \frac{5\log(\sin(x)+1)}{256} - \frac{-15\sin^7(x)+55\sin^5(x)-73\sin^3(x)-15\sin(x)}{384\sin^8(x)-1536\sin^6(x)+2304\sin^4(x)-1536\sin^2(x)+384}$$

$$\int \tan^4 x \ \sec^1 \ x \, dx = -\frac{3\log(\sin(x)-1)}{16} + \frac{3\log(\sin(x)+1)}{16} - \frac{-5\sin^3(x)+3\sin(x)}{8\sin^4(x)-16\sin^2(x)+8}$$

$$\int \tan^4 x \ \sec^3 \ x \, dx = -\frac{\log(\sin(x)-1)}{32} + \frac{\log(\sin(x)+1)}{32} - \frac{-3\sin^5(x)-8\sin^3(x)+3\sin(x)}{48\sin^6(x)-144\sin^4(x)+144\sin^2(x)-48}$$

$$\int \tan^4 x \ \sec^5 \ x \, dx = -\frac{3\log(\sin(x)-1)}{256} + \frac{3\log(\sin(x)+1)}{256} - \frac{3\sin^7(x)-11\sin^5(x)-11\sin^3(x)+3\sin(x)}{128\sin^8(x)-512\sin^6(x)+768\sin^4(x)-512\sin^2(x)+128}$$

$$\int \tan^4 x \ \sec^7 \ x \, dx = -\frac{3\log(\sin(x)-1)}{512} + \frac{3\log(\sin(x)+1)}{512} + \frac{-15\sin^9(x)+70\sin^7(x)-128\sin^5(x)-70\sin^3(x)+15\sin(x)}{1280\sin^{10}(x)-6400\sin^8(x)+12800\sin^6(x)-12800\sin^4(x)+6400\sin^2(x)-1280}$$

$$\int \tan^6 x \ \sec^1 \ x \, dx = \frac{5\log(\sin(x)-1)}{32} - \frac{5\log(\sin(x)+1)}{32} + \frac{-33\sin^5(x)+40\sin^3(x)-15\sin(x)}{48\sin^6(x)-144\sin^4(x)+144\sin^2(x)-48}$$

$$\int \tan^6 x \ \sec^3 \ x \, dx = \frac{5\log(\sin(x)-1)}{256} - \frac{5\log(\sin(x)+1)}{256} - \frac{-15\sin^7(x)-73\sin^5(x)+55\sin^3(x)-15\sin(x)}{384\sin^8(x)-1536\sin^6(x)+2304\sin^4(x)-1536\sin^2(x)+384}$$

$$\int \tan^6 x \ \sec^5 \ x \, dx = \frac{3\log(\sin(x)-1)}{512} - \frac{3\log(\sin(x)+1)}{512} + \frac{15\sin^9(x)-70\sin^7(x)-128\sin^5(x)+70\sin^3(x)-15\sin(x)}{1280\sin^{10}(x)-6400\sin^8(x)+12800\sin^6(x)-12800\sin^4(x)+6400\sin^2(x)-1280}$$

This case is considered as the hardest of all cases in $\int \tan^n x \ \sec^m \ x \, dx$. If you try to solve it by hand, most calculus book will recommend to use method of substitution to replace $\tan^n x$ into $\sec x$ function so we can then compute the integral of secant with odd power, which already been done previously in level 3 integral section.

The secant with odd power make use of middle coefficient, the sum of two neighbor entry in the vector that will become the constant / coefficient for the numerator with sine function that has odd power.

Here, we will do the same, with an adding of some tricks.

The patterns that can be seen from those equations above are

- The terms are divided into 3, the same like integral of secant with odd power which are:
  1. The numerator with $\sin(x)$ function that has odd power
  2. The denominator with $\sin(x)$ function that has even power
  3. The log terms
- For all $\int \tan^n x \ \sec^m \ x \ dx$, when $m = 1$ the result is the same / equivalent with $\int \sec^{n+1} \ x \ dx$.

  To solve for any m that can be any number but 1, we will need to make another **for** loop to adjust the coefficient.

  So the starting is we will use $\int \sec^{n+1} \ x \ dx$ that we already know the pattern.
- The denominator coefficients have the exact same pattern with the integral of secant with odd power, but this time we will make the denominator coefficient to run another loop again till we reach m.

To learn the pattern we will try with small n and m, we will make use of

$$\tan^2 x = \sec^2 x - 1$$

to get everything in terms of sec x.

For $n = 2$ and $m = 3$

$$\begin{aligned}
\int \tan^2 x \ \sec^3 \ x \ dx &= (\sec^2 x - 1)\sec^3 x \ dx \\
&= \int \sec^5 x - \sec^3 \ x \ dx \\
&= \left[-\frac{3\log(\sin(x)-1)}{16} + \frac{3\log(\sin(x)+1)}{16} - \frac{3\sin^3 x - 5\sin \ x}{8\sin^4 x - 16\sin^2 x + 8} \right] \\
&- \left[-\frac{\log(\sin(x)-1)}{4} + \frac{\log(\sin(x)+1)}{4} - \frac{\sin \ x}{2\sin^2 x - 2} \right] \\
&= -\frac{\log(\sin(x)-1)}{16} + \frac{\log(\sin(x)+1)}{16} \\
&- \frac{3\sin^3 x - 5\sin \ x - (4\sin^2 x - 4)(\sin \ x)}{8\sin^4 x - 16\sin^2 x + 8} \\
&= -\frac{\log(\sin(x)-1)}{16} + \frac{\log(\sin(x)+1)}{16} - \frac{-\sin^3 x - \sin \ x}{8\sin^4 x - 16\sin^2 x + 8}
\end{aligned}$$

For $n = 4$ and $m = 3$

$$\begin{aligned}
\int \tan^4 x \ \sec^3 \ x \ dx &= (\sec^2 x - 1)^2 \sec^3 x \ dx \\
&= \int \sec^3 x(\sec^4 \ x - 2\sec^2 x + 1) \ dx \\
&= \int \sec^3 x - 2\sec^5 x + \sec^7 \ x \ dx \\
&= -\frac{\log(\sin(x)-1)}{32} + \frac{\log(\sin(x)+1)}{32} + \frac{-3\sin^5 x - 8\sin^3 x + 3\sin \ x}{48\sin^6 x - 144\sin^4 x + 144\sin^2 x - 48}
\end{aligned}$$

If we take a closer look and observe at this part $\int \sec^3 x - 2\sec^5 x + \sec^7 \ x \ dx$, we are seeing a binomial coefficient / pascal triangle with alternating sign, thus we can use this pattern to compute $\int \tan^n x \ \sec^m \ x \ dx$ with n even and m odd for any number of n and m.

```
#include <iostream>
#include "symintegrationc++.h"
#include <vector>

using namespace std;

// Factorial and combinations
Symbolic factorial(int n) {
	if (n <= 1)
	{
		return 1;
	}
	Symbolic result = 1;
	for (int i = 2; i <= n; ++i)
	{
		result *= i;
	}
	return result;
}

Symbolic combinations(int n, int r) {
	if (r < 0 || r > n)
	{
		return 0; // Invalid input
	}
	return factorial(n) / (factorial(r) * factorial(n - r));
}

Symbolic numeratorintegralsecantodd(int n) {
	Symbolic x("x");
	int k = 1, l=2;
	vector<int> v={1};
	vector<int> mc={1}; // to store the middle coefficient
	Symbolic sgn = -1;
	Symbolic integral_numerator;
	Symbolic d0 = 2, d1 = 2;
	int j =1;
	int m = n-(0.5*(n+1));
	int last_coeff;
	int first_coeff = 3;

	// For the coefficient at the numerator of sine with odd power
	for(int i = 1 ; i < (n-1)/2 ; i = i+1)
	{
```

```
            if (i >= 2)
            {
                  for(int ic = 1 ; ic < i ; ic = ic+1)
                  {
                        mc[ic—1] = v[ic—1] + v[ic];
                  }
            }

            k = k*l;
            last_coeff = v[j—1];
            v[0] = v[0]*(first_coeff+2*(i—1));
            v.assign({v[0]});

            for(int ic = 1 ; ic < i ; ic = ic+1)
            {
                  v.push_back(mc[ic—1]*(first_coeff+2*(i—1)));
            }
            d1 = d1*(2*i);
            d0 = d0*(2+2*i);
            v.push_back(last_coeff*(first_coeff+2*(i—1))+k);
            l = l+2;
            j = j+1;
      }
      int j_num = 0 ;
      for(int i = n—2 ; i >= 1 ; i = i—2)
      {
            integral_numerator += sgn*v[j_num]*((sin(x))^(i));
            sgn = —sgn;
            j_num = j_num+1;
      }
      return integral_numerator;
}

Symbolic denominatorintegralsecantodd(int n) {
      Symbolic x("x");
      int k = 1, l=2;
      vector<int> v={1};
      vector<int> mc={1}; // to store the middle coefficient
      Symbolic sgn = 1;
      Symbolic integral_denominator;
      Symbolic d0 = 2, d1 = 2;
      int j =1;
      int m = n—(0.5*(n+1));
      int last_coeff;
      int first_coeff = 3;

      // For the coefficient at the numerator of sine with odd power
      for(int i = 1 ; i < (n—1)/2 ; i = i+1)
```

```
	{
		if (i >= 2)
		{
			for(int ic = 1 ; ic < i ; ic = ic+1)
			{
				mc[ic-1] = v[ic-1] + v[ic];
			}
		}

		k = k*l;
		last_coeff = v[j-1];
		v[0] = v[0]*(first_coeff+2*(i-1));
		v.assign({v[0]});

		for(int ic = 1 ; ic < i ; ic = ic+1)
		{
			v.push_back(mc[ic-1]*(first_coeff+2*(i-1)));
		}
		d1 = d1*(2*i);
		d0 = d0*(2+2*i);
		v.push_back(last_coeff*(first_coeff+2*(i-1))+k);
		l = l+2;
		j = j+1;
	}
	j = 1;
	for(int i = n-1 ; i >= 0 ; i = i-2)
	{
		integral_denominator += sgn*d0*combinations(m,j-1)*((sin(x
		    ))^(i));
		sgn = -sgn;
		j = j+1;
	}
	return integral_denominator;
}

Symbolic logtermsintegralsecantodd(int n) {
	Symbolic x("x");
	int k = 1, l=2;
	vector<int> v={1};
	vector<int> mc={1}; // to store the middle coefficient
	Symbolic sgn = -1;
	Symbolic integral_denominator;
	Symbolic d0 = 2, d1 = 2;
	int j =1;
	int m = n-(0.5*(n+1));
	int last_coeff;
	int first_coeff = 3;
```

```
	// For the coefficient at the numerator of sine with odd power
	for(int i = 1 ; i < (n-1)/2 ; i = i+1)
	{
		if (i >= 2)
		{
			for(int ic = 1 ; ic < i ; ic = ic+1)
			{
				mc[ic-1] = v[ic-1] + v[ic];
			}
		}

		k = k*l;
		last_coeff = v[j-1];
		v[0] = v[0]*(first_coeff+2*(i-1));
		v.assign({v[0]});


		for(int ic = 1 ; ic < i ; ic = ic+1)
		{
			v.push_back(mc[ic-1]*(first_coeff+2*(i-1)));
		}
		d1 = d1*(2*i);
		d0 = d0*(2+2*i);
		v.push_back(last_coeff*(first_coeff+2*(i-1))+k);
		l = l+2;
		j = j+1;
	}
	d1 = d1*(n-1);
	return (-v[0]*ln(sin(x)-1))/(d1) + (v[0]*ln(sin(x)+1))/(d1) ;
}

int main(void)
{
	Symbolic x("x");
	int bpowersec = 3;
	int bpowertan = 6;
	int bpower = bpowersec+bpowertan;
	Symbolic sgn = 1;
	Symbolic result;
	int j = 1;
	int m = bpowertan/2;
	for(int i = bpower ; i >= bpowersec ; i = i-2)
	{
		result += sgn*combinations(m,j-1)*( (
		    numeratorintegralsecantodd(i)/
		    denominatorintegralsecantodd(i)) +
		    logtermsintegralsecantodd(i));
		sgn = -sgn;
```

```
            j = j+1;
        }
        //cout << "integral at numerator= "<< numeratorintegralsecantodd
            (3)<< endl;
        //cout << "integral at denominator= "<<
            denominatorintegralsecantodd(3)<< endl;
        //cout << "log terms= "<< logtermsintegralsecantodd(3)<< endl;
        cout << "for secant power of " << bpowersec << " and tangent power
             of " << bpowertan << endl;
        cout << "integral = "<< result << endl;
        //cout << "integral with subs = " << result[x==1] << endl;
        return 0;
}
```

Code 25: *level 4 integral for tangent even power and secant odd power case*

It is recommended that you use the:
cout « "integral at numerator= "« numeratorintegralsecantodd(3)« endl;
cout « "integral at denominator= "« denominatorintegralsecantodd(3)« endl;
cout « "log terms= "« logtermsintegralsecantodd(3)« endl;
(uncomment them in the C++ code)

and comment the :
cout « "for secant power of " « bpowersec « " and tangent power of " « bpowertan « endl;
cout « "integral = "« result « endl;

Because it is faster to show the result when we only show the log terms function, numerator only / denominator only of the fraction with sine function than to show it in the full fraction function, it is still slow and we wonder why, because SymPy is able to show the result very fast, knowing that C++ is basically / by default faster than SymPy we might suspect because it is the fraction that we are using combining with symbolic computation, if we only show the vector / array, the result can come up very fast, faster than SymPy.

We will figure it out and make it really fast one day. Any reader interested is always welcome to give a nice input for this.

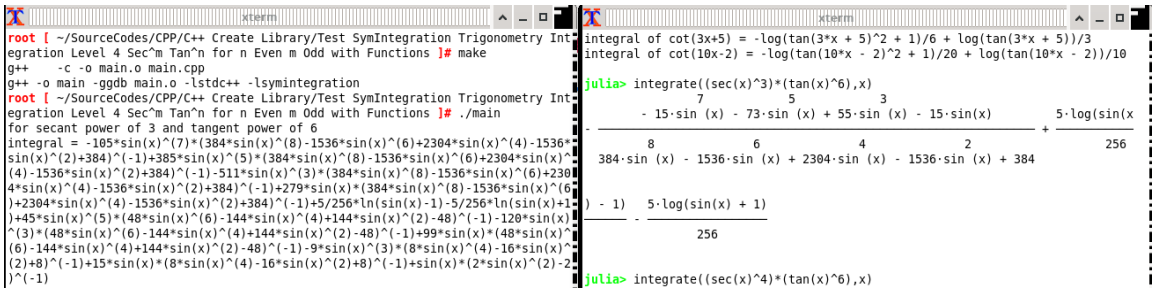

Figure 2.2: *The computation of* $\int \tan^6 x \ \sec^3 \ x \, dx$ *with SymPy and SymIntegration (**SymIntegration/Examples/Test SymIntegration Trigonometry Integration Level 4 Sec^m Tan^n for n Even m Odd with Functions/main.cpp**).*

Case 2: For $\int \tan^n x \ \sec^m \ x \, dx$ with n odd

We divide this case into 2 smaller cases:

1. When m even
2. When m odd

Case 2-a: For $\int \tan^n x \ \sec^m \ x \, dx$ with n odd and m even

$$\int \tan^1 x \ \sec^2 \ x \, dx = \frac{1}{2\cos^2(x)}$$

$$\int \tan^1 x \ \sec^4 \ x \, dx = \frac{1}{4\cos^4(x)}$$

$$\int \tan^1 x \ \sec^6 \ x \, dx = \frac{1}{6\cos^6(x)}$$

$$\int \tan^1 x \ \sec^8 \ x \, dx = \frac{1}{8\cos^8(x)}$$

$$\int \tan^3 x \ \sec^2 \ x \, dx = \frac{1-2\cos^2(x)}{4\cos^4(x)}$$

$$\int \tan^3 x \ \sec^4 \ x \, dx = \frac{2-3\cos^2(x)}{12\cos^6(x)}$$

$$\int \tan^3 x \ \sec^6 \ x \, dx = \frac{3-4\cos^2(x)}{24\cos^8(x)}$$

$$\int \tan^3 x \ \sec^8 \ x \, dx = \frac{4-5\cos^2(x)}{40\cos^{10}(x)}$$

$$\int \tan^5 x \ \sec^2 \ x \, dx = \frac{-(-3\cos^4(x)+3\cos^2(x)-1)}{6\cos^6(x)}$$

$$\int \tan^5 x \ \sec^4 \ x \, dx = \frac{-(-6\cos^4(x)+8\cos^2(x)-3)}{24\cos^8(x)}$$

$$\int \tan^5 x \ \sec^6 \ x \, dx = \frac{-(-10\cos^4(x)+15\cos^2(x)-6)}{60\cos^{10}(x)}$$

$$\int \tan^5 x \ \sec^8 \ x \, dx = \frac{-(-15\cos^4(x)+24\cos^2(x)-10)}{120\cos^{12}(x)}$$

The patterns that can be seen from those equations above are

- They are all function that only consist of function of cosine and constants.
- The power of cosine in the denominator is $m + n - 1$.
- You can solve it easily to determine the pattern by substituting

$$\sec^2 x = \tan^2 x + 1$$

then we will have the terms in $\tan(x)$ with odd power only, which we already have learned the pattern from the previous section.

For example, let $m = 4, n = 3$

$$\begin{aligned}\int \tan^3 x \ \sec^4 \ x \, dx &= \int \tan^3 x(\tan^2 x + 1)^2 dx \\ &= \int \tan^3 x(\tan^4 x + 2\tan^2 x + 1)dx \\ &= \int \tan^7 x + 2\tan^5 x + \tan^3 x dx\end{aligned}$$

It is using combination / Pascal' triangle again, this time the sign is the same, all positive.

```
#include <iostream>
#include "symintegrationc++.h"
#include <vector>

using namespace std;

// Factorial and combinations
Symbolic factorial(int n) {
	if (n <= 1)
	{
		return 1;
	}
	Symbolic result = 1;
	for (int i = 2; i <= n; ++i)
	{
		result *= i;
	}
	return result;
}

Symbolic combinations(int n, int r) {
	if (r < 0 || r > n)
	{
		return 0; // Invalid input
	}
	return factorial(n) / (factorial(r) * factorial(n - r));
}

Symbolic integraltangentodd(int n) {
```

```
Symbolic x("x");
Symbolic integral;
Symbolic integral_front;
Symbolic sgn = 1;
vector<int> v={1};
vector<int> v_temp={1};
vector<int> mc={1}; // to store the middle coefficient
vector<int> mc_temp={1}; // to store the temporary / new middle
    coefficient
int d0 = 2;
int k = 1, l=2;
// For the coefficient at the numerator
for(int i = 1 ; i <= (n-1)/2 ; i = i+1)
{
       if (i == 1)
       {
              v[0] = 1;
              v.assign({v[0]});
       }

       if (i ==2)
       {
              mc[0]=1;
              mc.assign({mc[0]});
              v_temp[0] = v[0]*(i-1);
              v_temp.assign({v_temp[0]});

              for(int j = 1 ; j < i ; j = j+1)
              {
                     v_temp.push_back(v[j-1]*((2*i)-j) + v_temp
                         [0]*mc[j-1] );
              }
              v[0] = v_temp[0];
              v.assign({v[0]});
              for(int j = 1 ; j < i ; j = j+1)
              {
                     v.push_back(v_temp[j]);
              }
       }
       if (i == 3)
       {
              mc[0] = mc[0] + 1; //
              mc.assign({mc[0]});

              v_temp[0] = v[0]*(i-1);
              v_temp.assign({v_temp[0]});
              for(int j = 1 ; j < i-1 ; j = j+1)
              {
```

```
                        v_temp.push_back(v[j-1]*((2*i)-j) + v_temp
                            [0]*mc[j-1] );
                }
                v_temp.push_back((v[i-2]*(i+1) ) + (v_temp[0]) );
                // Assign the temporary vector to vector v for
                    future use
                v[0] = v_temp[0];
                v.assign({v[0]});
                for(int j = 1 ; j < i ; j = j+1)
                {
                        v.push_back(v_temp[j]);
                }
        }
        if (i == 4)
        {
                mc[0] = mc[0] + 1;
                mc.assign({mc[0]});
                mc.push_back(mc[0]);

                v_temp[0] = v[0]*(i-1);
                v_temp.assign({v_temp[0]});
                for(int j = 1 ; j < i-1 ; j = j+1)
                {
                        v_temp.push_back(v[j-1]*((2*i)-j) + v_temp
                            [0]*mc[j-1] );
                }
                v_temp.push_back((v[i-2]*(i+1) ) + (v_temp[0]) );
                // Assign the temporary vector to vector v for
                    future use
                v[0] = v_temp[0];
                v.assign({v[0]});
                for(int j = 1 ; j < i ; j = j+1)
                {
                        v.push_back(v_temp[j]);
                }
        }
        if (i >= 5)
        {
                mc_temp[0] = mc[0]+1;
                mc_temp.assign({mc_temp[0]});
                for(int ic = 1 ; ic < l ; ic = ic+1)
                {
                        mc_temp[ic] = mc[ic-1] + mc[ic];
                }
                mc[l] = (mc_temp[0]);
                mc[l+1] = (mc_temp[0]);

                mc[0]=mc_temp[0];
```

```
                    mc.assign({mc[0]});

                    for(int ic = 1 ; ic < l ; ic = ic+1)
                    {
                           mc.push_back(mc_temp[ic]);
                    }
                    mc.push_back(mc_temp[0]);
                    mc.push_back(0);

                    v_temp[0] = v[0]*(i-1);
                    v_temp.assign({v_temp[0]});
                    for(int j = 1 ; j < i-1 ; j = j+1)
                    {
                           v_temp.push_back(v[j-1]*((2*i)-j) + v_temp
                               [0]*mc[j-1] );
                    }
                    v_temp.push_back((v[i-2]*(i+1) ) + (v_temp[0]) );
                    // Assign the temporary vector to vector v for
                        future use
                    v[0] = v_temp[0];
                    v.assign({v[0]});
                    for(int j = 1 ; j < i ; j = j+1)
                    {
                           v.push_back(v_temp[j]);
                    }

                    l = l+1;
             }

             d0 = d0*k;
             k = k+1;
      }
      for(int i = 1 ; i <= (n-1)/2 ; i = i+1)
      {
             integral += sgn*v[i-1]*(cos(x)^(2*(i-1)));
             sgn=-sgn;

      }
      sgn = 1;
      for(int i = 1 ; i <= (n-1)/2 ; i = i+1)
      {
             integral_front = sgn;
             sgn = -sgn;
      }
      integral = integral_front*ln(cos(x)) + (integral)/(d0*(cos(x)^(n
          -1))) ;
      return integral;
}
```

```
int main(void)
{
       Symbolic x("x");
       int bpowersec = 4;
       int bpowertan = 5;
       int bpower = bpowersec+bpowertan;
       Symbolic result;
       int j = 1;
       int m = bpowersec/2;
       for(int i = bpowertan ; i <= bpower ; i = i+2)
       {
              result += combinations(m,j-1)*integraltangentodd(i);
              j = j+1;
       }
       cout << "for secant power of " << bpowersec << " and tangent power
            of " << bpowertan << endl;
       cout << "integral = "<< result<< endl;
       return 0;
}
```

Code 26: *level 4 integral for tangent odd power and secant even power case*

Case 2-b: For $\int \tan^n x \ \sec^m \ x \, dx$ with n odd and m odd

$$\int \tan^1 x \ \sec^1 \ x \, dx = \frac{1}{\cos(x)}$$
$$\int \tan^1 x \ \sec^3 \ x \, dx = \frac{1}{3\cos^3(x)}$$
$$\int \tan^1 x \ \sec^5 \ x \, dx = \frac{1}{5\cos^5(x)}$$
$$\int \tan^1 x \ \sec^7 \ x \, dx = \frac{1}{7\cos^7(x)}$$

$$\int \tan^3 x \ \sec^1 \ x \, dx = \frac{1-3\cos^2(x)}{3\cos^3(x)}$$
$$\int \tan^3 x \ \sec^3 \ x \, dx = \frac{3-5\cos^2(x)}{15\cos^5(x)}$$
$$\int \tan^3 x \ \sec^5 \ x \, dx = \frac{5-7\cos^2(x)}{35\cos^7(x)}$$
$$\int \tan^3 x \ \sec^7 \ x \, dx = \frac{7-9\cos^2(x)}{63\cos^9(x)}$$

$$\int \tan^5 x \ \sec^1 \ x \, dx = \frac{-(-15\cos^4(x) + 10\cos^2(x) - 3)}{15\cos^5(x)}$$
$$\int \tan^5 x \ \sec^3 \ x \, dx = \frac{-(-35\cos^4(x) + 42\cos^2(x) - 15)}{105\cos^7(x)}$$
$$\int \tan^5 x \ \sec^5 \ x \, dx = \frac{-(-63\cos^4(x) + 90\cos^2(x) - 35)}{315\cos^9(x)}$$
$$\int \tan^5 x \ \sec^7 \ x \, dx = \frac{-(-99\cos^4(x) + 154\cos^2(x) - 63)}{693\cos^{11}(x)}$$

The patterns that can be seen from those equations above are

- They are all function that only consist of function of cosine and constants.
- The power of cosine in the denominator is $m + n - 1$.
- You can solve it easily to determine the pattern by substituting

$$\tan^2 x = \sec^2 x - 1$$

and then use method of substitution by assigning

$$u = \sec \ x$$
$$du = \sec \ x \tan \ x \, dx$$

then we will have the terms in $\sec(x)$ with even power only, which we already have learned the pattern from the previous section.

For example, let $m = 3, n = 5$

$$\begin{aligned}
\int \tan^5 x \ \sec^3 \ x \, dx &= \int \tan^4 x \, sec^2 x (\sec \ x \tan \ x \, dx) \\
&= \int [(\sec^2 x - 1)^2 sec^2 x \,] (\sec \ x \tan \ x \, dx) \\
&= \int [(\sec^4 x - 2\sec^2 x + 1) sec^2 x] \ (\sec \ x \tan \ x \, dx) \\
&= \int [\sec^6 x - 2\sec^4 x + \sec^2 x] \ (\sec \ x \tan \ x \, dx) \\
&= \int u^6 - 2u^4 + u^2 \, du \\
&= \frac{1}{7}u^7 - \frac{2}{5}u^5 + \frac{1}{3}u^3 \\
&= \frac{1}{7}\sec^7 x - \frac{2}{5}\sec^5 x + \frac{1}{3}\sec^3 x \\
&= \frac{1}{7}\frac{1}{\cos^7 x} - \frac{2}{5}\frac{1}{\cos^5 x} + \frac{1}{3}\frac{1}{\cos^3 x}
\end{aligned}$$

It is using combination / Pascal' triangle combining with a method of substitution, the sign is alternating this time. The thing is the method of substitution is a really helpful here, we do not even need to create a function, only integrating ordinary polynomial and then substituting back, we can substitute $u = \sec(x)$ or $u = \frac{1}{\cos(x)}$.

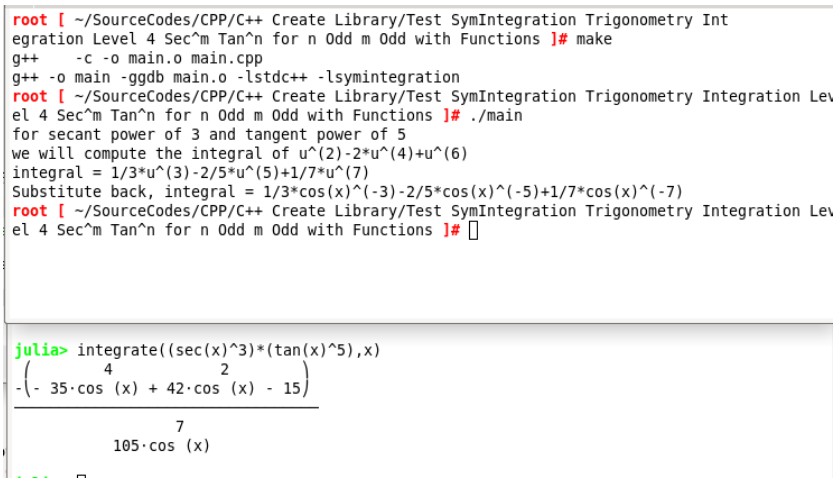

Figure 2.3: *The computation of* $\int \tan^5 x \ \sec^3 \ x \, dx$ *with SymPy and SymIntegration (**SymIntegration/Examples/Test SymIntegration Trigonometry Integration Level 4 Sec^m Tan^n for n Odd m Odd/main.cpp**).*

```
#include <iostream>
#include "symintegrationc++.h"
#include <vector>

using namespace std;

// Factorial and combinations
Symbolic factorial(int n) {
	if (n <= 1)
	{
		return 1;
	}
	Symbolic result = 1;
	for (int i = 2; i <= n; ++i)
	{
		result *= i;
	}
	return result;
}

Symbolic combinations(int n, int r) {
	if (r < 0 || r > n)
	{
		return 0; // Invalid input
	}
	return factorial(n) / (factorial(r) * factorial(n - r));
}

int main(void)
{
```

```
        Symbolic u("u"), x("x");
        int bpowersec = 3;
        int bpowertan = 5;
        int bpower = bpowersec+bpowertan;
        Symbolic result;
        int j = 1;
        int m = (bpowertan-1)/2;
        Symbolic sgn = 1;
        for(int i = bpowersec-1 ; i <= bpower-2 ; i = i+2)
        {
                result += sgn*combinations(m,j-1)*(u^i);
                j = j+1;
                sgn = -sgn;
        }
        cout << "for secant power of " << bpowersec << " and tangent power
             of " << bpowertan << endl;
        cout << "we will compute the integral of "<< result << endl;
        Symbolic f = integrate(result,u);
        cout << "integral = "<< f << endl;
        cout << "Substitute back, integral = "<< f[u==(1/cos(x))] << endl;
        return 0;
}
```

Code 27: *level 4 integral for tangent odd power and secant odd power case*

ix. $\int \cot^n x \ \csc^m \ x \, dx$

The formula of $\int \cot^n x \ \csc^m \ x \, dx$ has similar pattern with $\int \tan^n x \ \sec^m \ x \, dx$, we only change from $\sin(x)$ to $\cos(x)$ and from $\cos(x)$ to $\sin(x)$, the sign will be changed too, from minus to plus, and from plus to minus. Once we have decoded the pattern for $\int \tan^n x \ \sec^m \ x \, dx$, we only need to modify a bit for this integral type.

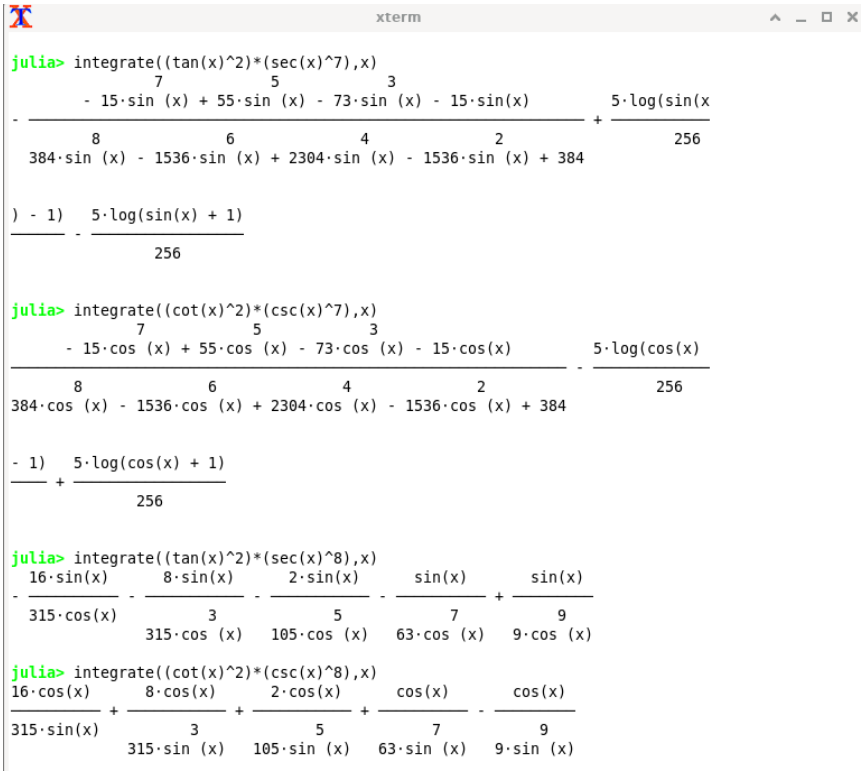

Figure 2.4: *The integral of* $\int \cot^n x \ \csc^m \ x \, dx$ *that has similar pattern with of* $\int \tan^n x \ \sec^m \ x \, dx$.

VII. Trigonometry and Transcendentals Formula

[SI*] We will list all the trignometry and trancendentals (exponential and logarithm) formulas that can be used to help you solve integral problems

[SI*] Trigonometry

Reciprocal identities

$$\begin{aligned}\tan\ x &= \frac{\sin\ x}{\cos\ x}\\ \cot\ x &= \frac{\cos\ x}{\sin\ x}\\ \csc\ x &= \frac{1}{\sin\ x}\\ \sec\ x &= \frac{1}{\cos\ x}\end{aligned}$$

Pythagorean identities

$$\begin{aligned}\sin^2 x + \cos^2 x &= 1\\ \tan^2 x &= \sec^2 x - 1\\ \cot^2 x &= \csc^2 x - 1\end{aligned}$$

Sum and difference identities

$$\begin{aligned}\sin(x+y) &= \sin\ x\ \cos\ y + \cos x\ \sin\ y\\ \sin(x-y) &= \sin\ x\ \cos\ y - \cos x\ \sin\ y\\ \cos(x+y) &= \cos\ x\ \cos\ y - \sin x\ \sin\ y\\ \cos(x-y) &= \cos\ x\ \cos\ y + \sin x\ \sin\ y\\ \tan(x+y) &= \frac{\tan\ x + \tan\ y}{1 - \tan\ x\ \tan\ y}\\ \tan(x-y) &= \frac{\tan\ x - \tan\ y}{1 + \tan\ x\ \tan\ y}\end{aligned}$$

Double-angle identities

$$\begin{aligned}\sin\ 2x &= 2\sin\ x\ \cos x = \frac{2\tan\ x}{1+\tan^2 x}\\ \cos\ 2x &= \cos^2 x - \sin^2 x\\ \cos\ 2x &= 2\cos^2 x - 1\\ \cos\ 2x &= 1 - 2\sin^2 x\\ \cos\ 2x &= \frac{1-\tan^2 x}{1+\tan^2 x}\\ \tan\ 2x &= \frac{2\tan\ x}{1-\tan^2 x}\\ \cot\ 2x &= \frac{\cot^2 x - 1}{2\cot\ x}\end{aligned}$$

Half-angle identities

$$\sin\ \frac{x}{2} = \pm\sqrt{\frac{1-\cos\ x}{2}}$$
$$\cos\ \frac{x}{2} = \pm\sqrt{\frac{1+\cos\ x}{2}}$$
$$\tan\ \frac{x}{2} = \pm\sqrt{\frac{1-\cos\ x}{1+\cos\ x}} = \frac{\sin\ x}{1+\cos\ x} = \frac{1-\cos\ x}{\sin\ x}$$

Integral

$$\int \sin\ x\, dx = -\cos\ x$$
$$\int \cos\ x\, dx = \sin\ x$$
$$\int \tan\ x\, dx = -\log(\cos(x))$$
$$\int \sec\ x\, dx = -\frac{\log(\sin(x)-1)}{2} + \frac{\log(\sin(x)+1)}{2}$$
$$\int \csc\ x\, dx = \frac{\log(\cos(x)-1)}{2} - \frac{\log(\cos(x)+1)}{2}$$
$$\int \cot\ x\, dx = \log(\sin(x))$$

Inverse Trigonometric integral

$$\int \frac{1}{1+x^2}dx = \operatorname{atan}(x)$$
$$\int \frac{1}{a+bx^2}dx = -\frac{\sqrt{\frac{-1}{ab}}\ln\left(-a\sqrt{\frac{-1}{ab}}+x\right)}{2} + \frac{\sqrt{\frac{-1}{ab}}\ln\left(a\sqrt{\frac{-1}{ab}}+x\right)}{2}$$

Inverse hyperbolic trigonometry integral

$$\int \frac{1}{\sqrt{1+x^2}}dx = \sinh^{-1}(x) + C$$
$$\int \frac{1}{\sqrt{x^2-1}}dx = \cosh^{-1}(x) + C$$
$$\int \frac{1}{1-x^2}dx = \tanh^{-1}(x) + C, \quad |x| < 1$$
$$\int \frac{1}{1-x^2}dx = \coth^{-1}(x) + C, \quad |x| > 1$$
$$\int \frac{-1}{x\sqrt{1-x^2}}dx = \operatorname{sech}^{-1}(x) + C$$
$$\int \frac{-1}{|x|\sqrt{1+x^2}}dx = \operatorname{csch}^{-1}(x) + C$$

Note that the derivatives of $\tanh^{-1}(x)$ and $\coth^{-1}(x)$ are the same. Thus, when we compute $\int \frac{1}{1-x^2}dx$, we need to select the proper antiderivative based on the domain of the functions and the values of x.

$$\int \frac{1}{1-x^2}dx = \begin{cases} \tanh^{-1}(x) + C, & |x| < 1 \\ \coth^{-1}(x) + C, & |x| > 1 \end{cases}$$

Derivative

$$\begin{aligned}
\frac{d}{dx}\sin\ x &= \cos\ x \\
\frac{d}{dx}\cos\ x &= -\sin\ x \\
\frac{d}{dx}\tan\ x &= \tan^2 x + 1 \\
\frac{d}{dx}\sec\ x &= \sec(x)\tan(x) \\
\frac{d}{dx}\csc\ x &= -\cot(x)\csc(x) \\
\frac{d}{dx}\cot\ x &= -\cot^2 x - 1
\end{aligned}$$

[SI*] Logarithm

Logarithm along with exponential play a very important role in solving differential equations, so we will need to take a look at these formulas often to help us when solving differential equations problem

$$\begin{aligned}
\log_a b &= c \\
a^c &= b \\
\log b &= c \\
e^c &= b \\
\log_a a &= 1 \\
\log(a*b) &= \log(a) + \log(b) \\
\log\left(\frac{a}{b}\right) &= \log(a) - \log(b) \\
\log(a^b) &= b\ \log(a) \\
a^{\log_a b} &= b \\
\log_b a &= \frac{\log_c a}{\log_c} \\
\log_b c &= \log_b a \log_a c \\
log_{b^n} a^m &= \frac{m}{n}\log_b a \\
-\log_b a &= \log_b\left(\frac{1}{a}\right) \\
-\log_b a &= \log_{\frac{1}{b}} a
\end{aligned}$$

with $e = 2.718281828459045$ is the Euler' number. When there is no base written in *log* function, we will use e as the base number. The base number cannot eb equal to 1 nad it has to be positive. For all $\log(a)$, a is a real number and has to be positive.

VIII. Vector Calculus

- Div, grad, and curl

[SI*] Vector calculus talks about vector-valued functions, that is, functions whose input is a real number and whose output is a vector.

In vector calculus we will use the operator ∇, read: del, or nabla, it is an operator used in mathematics as a vector differential operator. When applied to a function defined on a one-dimensional domain, it denotes the standard derivative of the function as defined in calculus.

When applied to a field (a function defined on a multi-dimensional domain), it may denote any one of three operations depending on the way it is applied: the gradient or steepest slope of a scalar field; the divergence of a vector field; or the curl (rotation) of a vector field.

Del is a very convenient mathematical notation for those three operations (gradient, divergent, and curl) that makes many equations easier to write and remember. The del symbol can be formally defined as a vector operator whose components are the corresponding partial derivative operators. As a vector operator, it can act on scalar and vector fields in three different ways, giving rise to three different differential operations:

1. It acts on a scalar field ($f(x,y,z)$) by a formal scalar multiplication-to produce a vector field (∇f) called the gradient.
2. It acts on a vector field ($\boldsymbol{F}(x,y,z)$) by a formal dot product-to produce a scalar field ($\nabla \cdot \boldsymbol{F}$) called the divergence.
3. It acts on a vector field ($\boldsymbol{F}(x,y,z)$) by a formal cross product-to produce a vector field ($\nabla \times \boldsymbol{F}$) called the curl.

These three uses are summarized as:
Gradient

$$\text{grad } f = \nabla f \tag{2.17}$$

Physical interpretation:
Points in the direction of the greatest rate of increase of the scalar field, and its magnitude is that rate of increase.
Example:
Consider a scalar field representing temperature $T(x,y,z)$ in a garden. The gradient ∇T would give a vector at each point indicating the direction in which the temperature increases most rapidly, and its magnitude would be the rate of that increase.

Divergent

$$\text{div } \boldsymbol{v} = \nabla \cdot \boldsymbol{v} \tag{2.18}$$

Physical interpretation:
It measures the "outward flux per unit volume" of a vector field at a point. It quantifies how much a vector field is "spreading out" or "compressing" at that point.
Example:
For a vector field representing fluid flow $\boldsymbol{v}(x,y,z)$, the divergence $\nabla \cdot \boldsymbol{v}$ indicates

whether the fluid is expanding (positive divergence, like a source) or contracting (negative divergence, like a sink). If $\nabla \cdot \boldsymbol{v} = 0$, the fluid is incompressible.

Curl

$$\text{curl } \boldsymbol{v} = \nabla \times \boldsymbol{v} \tag{2.19}$$

Physical interpretation:
It measurres the "rotation" or "circulation" of a vector field at a point. The direction of the curl vector indicates the axis of rotation, and its magnitude indicates the strength of the rotation.
Example:
For a vector field representing the velocity of a fluid $\boldsymbol{v}(x, y, z)$, the curl $\nabla \times \boldsymbol{v}$ indicates the local angular velocity of the fluid. If $\nabla \times \boldsymbol{v} = \mathbf{0}$, the fluid is irrotational.

The divergence of a curl is zero

$$\nabla \cdot (\nabla \times \boldsymbol{F}) = 0 \tag{2.20}$$

This means that the divergence of any curl field is always zero, in linear algebra it is a dot product of an orthogonal vector.

Curl of the gradient is zero

$$\nabla \times (\nabla f) = \mathbf{0} \tag{2.21}$$

This means that the curl of any gradient field (which is a conservative vector field) is always the zero vector.

The div, grad, curl operators are crucial in various fields, including fluid dynamics, electromagnetism (Maxwell's equations), and heat transfer.

[SI*] In three-dimensional Cartesian coordinate system $\mathbb{R}^3$ with coordinates (x, y, z) and standard basis or unit vectors of axes $\{\boldsymbol{e}_x, \boldsymbol{e}_y, \boldsymbol{e}_z\}$, del is written as

$$\nabla = \boldsymbol{e}_x \frac{\partial}{\partial x} + \boldsymbol{e}_y \frac{\partial}{\partial y} + \boldsymbol{e}_z \frac{\partial}{\partial z} = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right) \tag{2.22}$$

As a vector operator, del naturally acts on scalar fields via scalar multiplication, and naturally acts on vector fields via dot products and cross products.

Then using the above definition of ∇, we may write

$$\nabla f = \left(\boldsymbol{e}_x \frac{\partial}{\partial x} \right) f + \left(\boldsymbol{e}_y \frac{\partial}{\partial y} \right) f + \left(\boldsymbol{e}_z \frac{\partial}{\partial z} \right) f = \frac{\partial f}{\partial x} \boldsymbol{e}_x + \frac{\partial f}{\partial y} \boldsymbol{e}_y + \frac{\partial f}{\partial z} \boldsymbol{e}_z \tag{2.23}$$

and

$$\nabla \cdot \boldsymbol{F} = \left(\boldsymbol{e}_x \frac{\partial}{\partial x} \cdot \boldsymbol{F} \right) + \left(\boldsymbol{e}_y \frac{\partial}{\partial y} \cdot \boldsymbol{F} \right) + \left(\boldsymbol{e}_z \frac{\partial}{\partial z} \cdot \boldsymbol{F} \right) = \frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z} \tag{2.24}$$

and

$$\begin{aligned}\nabla \times \boldsymbol{F} &= \left(\boldsymbol{e}_x \frac{\partial}{\partial x} \times \boldsymbol{F}\right) + \left(\boldsymbol{e}_y \frac{\partial}{\partial y} \times \boldsymbol{F}\right) + \left(\boldsymbol{e}_z \frac{\partial}{\partial z} \times \boldsymbol{F}\right) \\ &= \frac{\partial}{\partial x}(0, -F_z, F_y) + \frac{\partial}{\partial y}(F_z, 0, -F_x) + \frac{\partial}{\partial z}(-F_y, F_x, 0) \\ &= \left(\frac{\partial F_z}{\partial y} - \frac{\partial F_y}{\partial z}\right)\boldsymbol{e}_x + \left(\frac{\partial F_x}{\partial z} - \frac{\partial F_z}{\partial x}\right)\boldsymbol{e}_y + \left(\frac{\partial F_y}{\partial x} - \frac{\partial F_x}{\partial y}\right)\boldsymbol{e}_z\end{aligned} \tag{2.25}$$

Del can also be expressed in other coordinate systems,e.g. in cylindrical and spherical coordinates.

[SI*] **Gradient**
the vector derivative of a scalar field f is called the gradient, and it can be represented as

$$\text{grad } f = \nabla f = \frac{\partial f}{\partial x}\hat{\boldsymbol{x}} + \frac{\partial f}{\partial y}\hat{\boldsymbol{y}} + \frac{\partial f}{\partial z}\hat{\boldsymbol{z}}$$

It always points in the direction of greatest increase of f, and it has a magnitude equal to the maximum rate of increase at the point-just like a standard derivative.

[SI*] **Divergence**
The divergence of a vector field $\boldsymbol{v}(x, y, z) = v_x\hat{\boldsymbol{x}} + v_y\hat{\boldsymbol{y}} + v_z\hat{\boldsymbol{z}}$ is a scalar field that can be represented as:

$$\text{div } \boldsymbol{v} = \nabla \cdot \boldsymbol{v} = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}$$

The divergence is roughly a measure of a vector field's increase in the direction it points; but more accurately, it is a measure of that field's tendency to converge toward or diverge from a point.

[SI*] **Curl**
The curl of a vector field $\boldsymbol{v}(x, y, z) = v_x\hat{\boldsymbol{x}} + v_y\hat{\boldsymbol{y}} + v_z\hat{\boldsymbol{z}}$ is a vector function that can be represented as:

$$\text{curl } \boldsymbol{v} = \nabla \times \boldsymbol{v} = \left(\frac{\partial v_z}{\partial y} - \frac{\partial v_y}{\partial z}\right)\hat{\boldsymbol{x}} + \left(\frac{\partial v_x}{\partial z} - \frac{\partial v_z}{\partial x}\right)\hat{\boldsymbol{y}} + \left(\frac{\partial v_y}{\partial x} - \frac{\partial v_x}{\partial y}\right)\hat{\boldsymbol{z}}$$

The curl at a point is proportional to the on-axis torque that a tiny pinwheel would be subjected to if it were centered at that point.

[SI*] In SymIntegration the div, grad , and curl can be handled in **src/divgradcurl.cpp**.

```
#include "symintegral/symintegrationc++.h"

#ifdef SYMBOLIC_DEFINE
#ifndef SYMINTEGRATION_CPLUSPLUS_DIVGRADCURL_DEFINE
#define SYMINTEGRATION_CPLUSPLUS_DIVGRADCURL_DEFINE

Symbolic div(const Symbolic &F, const Symbolic &x, const Symbolic &y,
    const Symbolic &z)
{
       Symbolic sol, i("i"), j("j"), k("k");

       Symbolic F1 = F.coeff(i,1);
```

```
       Symbolic F2 = F.coeff(j,1);
       Symbolic F3 = F.coeff(k,1);

       if(F1 != 0 || F2 != 0 || F3 != 0)
       {
              sol = df(F1,x) + df(F2,y) + df(F3,z);
       }
       else if(F1_hat != 0 || F2_hat != 0 || F3_hat != 0)
       {
              sol = df(F1_hat,x) + df(F2_hat,y) + df(F3_hat,z);
       }
       return sol;
}

Symbolic grad(const Symbolic &F, const Symbolic &x, const Symbolic &y,
    const Symbolic &z)
{
       Symbolic sol;

       if(F != 0)
       {
              sol = (df(F,x), df(F,y), df(F,z));
              sol = sol.transpose();
       }
       return sol;
}

...

Symbolic curl(const Symbolic &F, const Symbolic &x, const Symbolic &y,
    const Symbolic &z)
{
       Symbolic sol, i("i"), j("j"), k("k");

       Symbolic F1 = F.coeff(i,1);
       Symbolic F2 = F.coeff(j,1);
       Symbolic F3 = F.coeff(k,1);


       if(F1 != 0 || F2 != 0 || F3 != 0)
       {
              sol = (df(F3,y) - df(F2,z), -(df(F3,x) - df(F1,z)), df(F2
                  ,x) - df(F1,y));
              sol = sol.transpose();
       }
       else if(F1_hat != 0 || F2_hat != 0 || F3_hat != 0)
       {
              sol = (df(F3_hat,y) - df(F2_hat,z), -(df(F3_hat,x) - df(
```

```
                F1_hat,z)), df(F2_hat,x) - df(F1_hat,y));
            sol = sol.transpose();
        }
        return sol;
}
```

Code 28: *src/divgradcurl.cpp*

the source code can handle the input of a function along with the independent variables. For example, if we have a scalar field:

$$f(x,y,z) = -\frac{1}{2}xy + 2y + z^2$$

and we can compute the gradient with :
grad(f,x,y,z)

If we have a vector field:

$$\boldsymbol{F}(x,y,z) = y\hat{i} - x\hat{j} + yz\hat{k}$$

we can compute the divergence and curl with these functions:
div(F,x,y,z)
curl(F,x,y,z)

In SymIntegration the source code to compute the div, grad, curl for the example above will be like this:

```
#include <iostream>
#include "symintegrationc++.h"
#include <bits/stdc++.h>
#include <cmath>

using namespace std;
using namespace SymbolicConstant;

int main(void)
{
        Symbolic x("x"), y("y"), z("z"), f, f2, i("i"), j("j"), k("k");

        f = -(0.5*x)*y+2*y + z*z;
        f2 = y*i -x*j +y*z*k;
        Symbolic f2_hat = y*i -x*j +y*z*k;
        cout << "F(x,y,z) = " << f << endl;
        cout << "F2 (vector field) = " << f2_hat << endl;

        cout << "\ndiv(F2) = " << div(f2_hat,x,y,z) <<endl;
        cout << "\ngrad(F) = " << grad(f,x,y,z) <<endl;
        cout << "\ncurl(F2) = " << curl(f2_hat,x,y,z) <<endl;

        return 0;
}
```

Code 29: *Examples/test SymIntegration DivGradCurl/main.cpp*

```
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DivGradCurl ]# m
ake
g++    -c -o main.o main.cpp
g++ -o main -ggdb main.o -lstdc++ -lsymintegration
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DivGradCurl ]# ./main
F(x,y,z) = -0.5*x*y+2*y+z^(2)
F2 (vector field) = y*î-x*ĵ+y*z*k̂

div(F2) = y

grad(F) =
[ -0.5*y ]
[-0.5*x+2]
[   2*z  ]


curl(F2) =
[ z]
[ 0]
[-2]

root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DivGradCurl ]#
```

Figure 2.5: *The computation of div grad curl. (**SymIntegration/Examples/Test SymIntegration DivGradCurl/-main.cpp**).*

Chapter 3

SymIntegration to Solve Ordinary Differential Equations for Engineering Problems

"After Odin's defeat atop Yggdrasil and the revelation of Lezard's true nature, Freya quickly teleports there in a frantic state, telling Odin that Lezard's presence was the distortion she felt at Dipan" - ***Freya (Valkyrie Profile 2: Silmeria)***

"Everyone's favorite fighting fairy godmother is still kicking, taking out Ether Strike hits for her godfather Odin. The heretofore unmatched fury of her scowl derives from her lack of lines in the main story" - ***Freya (Valkyrie Profile: Covenant of the Plume)***

Dıfferential equations are of interest to engineers and other nonmathematicians, such as physicists. It is because of the possibility of using them to investigate a wide variety of problems in the physical, biological, and social sciences.

For example consider this Newton's law formula:

$$F = ma \tag{3.1}$$

If you read the Physics book dor undergraduate degree you will learn that it is not simply just $F = ma$, but it has a derivative as well

$$F = m\frac{dv}{dt} \tag{3.2}$$

Then add with drag force the equation will become

$$m\frac{dv}{dt} = mg - \gamma v \tag{3.3}$$

The Eq. (3.3) is what we call a differential equation that model an object falling in the atmosphere near sea level. From this equation we are able to determine the velocity at a given time, $v(t)$, from known g, m, γ. You can read more on this book [1], it is the first example of the book.

We will assume that you already read some books or learn from anywhere about differential equations, so we will ony focus on the SymIntegration function and capability that can help you

to compute the solution of a differential equation.

Starting on June 23rd, 2025, we have created a new function **dsolve** in SymIntegration, it is still very fresh and in infant stage and just able to solve a very simple first order linear ordinary differential equations.

I. Solving First Order Ordinary Differential Equations

- Method of Integrating Factors

[SI*] Solve the initial value problem

$$2y' + ty = 2 \tag{3.4}$$

with the initial value condition

$$y(0) = 1 \tag{3.5}$$

Solution:
We will convert the differential equation Eq. (3.4) into the standard form to become

$$y' + \left(\frac{t}{2}\right) y = 1 \tag{3.6}$$

Thus $p(t) = t/2$, and the integrating factor is

$$\mu(t) = e^{\int p(t)\ dt} = e^{\frac{t^2}{4}}$$

Then multiply Eq. (3.6) by $\mu(t)$ so that

$$e^{\frac{t^2}{4}} y' + \left(\frac{t}{2}\right) e^{\frac{t^2}{4}} y = e^{\frac{t^2}{4}} \tag{3.7}$$

Then integrate both sides of equation we will obtain

$$e^{\frac{t^2}{4}} y = \int e^{\frac{t^2}{4}}\ dt + c \tag{3.8}$$

thus

$$y = e^{-\frac{t^2}{4}} \int_0^t e^{\frac{t^2}{4}}\ ds + ce^{-\frac{t^2}{4}} \tag{3.9}$$

You can compute the value of the constant c by inputting the initial condition of $y(0) = 1$.

Now, in SymIntegration we will need the user to input the equation in term of $\frac{dy}{dx} = f(x, y)$, we will input the $f(x, y)$ and the coefficient of $\frac{dy}{dx}$ has to be 1.

```
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve ]# make
g++    -c -o main.o main.cpp
g++ -o main -ggdb main.o -lstdc++ -lsymintegration
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve ]# ./main
f(x) = -0.5*x*y+1
DSolve for f(x,y).

y(x) = 1/2*π^(1/2)*(erf(x*(-0.25)^(1/2)))*(-0.25)^(-1/2)*e^(-0.25*x^(2))+C*e^(-0.25*x^(2))
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve ]#
```

Figure 3.1: *The newly built function **dsolve** in SymIntegration is able to compute the solution $y(x)$ for the ordinary differential equation of $2y' + ty = 2$ (**SymIntegration/Examples/Test SymIntegration DSolve/main.cpp**).*

[SI*] For the coding in SymIntegration, we add a new C++ file:
src/dsolve.cpp
This is the very first beginning of the dsolve by June 23rd, 2025:

```
#include "symintegral/symintegrationc++.h"

#ifdef SYMBOLIC_DEFINE
#ifndef SYMINTEGRATION_CPLUSPLUS_DSOLVE_DEFINE
#define SYMINTEGRATION_CPLUSPLUS_DSOLVE_DEFINE

Symbolic dsolve(const Symbolic &fx, const Symbolic &y, const
    Symbolic &x)
{
       Symbolic dsol, mu, C("C");

       if(fx != 0)
       {
              list<Equations> eq;
              list<Equations>::iterator i;
              UniqueSymbol a, b, c, d;
              eq = (a*x*y + d).match(fx, (a,d));
              for(i=eq.begin(); i!=eq.end(); ++i)
              {
                     try {
                            Symbolic ap = rhs(*i, a), dp = rhs(*i,
                                 d);
                            mu = exp(integrate(-ap*x,x));
                            dsol = (integrate(mu*dp,x))/(mu) + (C)
                                /(mu);
                     } catch(const SymbolicError &se) {}
              }
       }
       return dsol;
}

#endif
#endif
```

Code 30: *src/dsolve.cpp*

and the include file as well **include/symintegral/dsolve.h**

```
#ifndef SYMINTEGRATION_CPLUSPLUS_DSOLVE

#ifdef SYMBOLIC_FORWARD
#ifndef SYMINTEGRATION_CPLUSPLUS_DSOLVE_FORWARD
#define SYMINTEGRATION_CPLUSPLUS_DSOLVE_FORWARD

#endif
#endif

#ifdef SYMBOLIC_DECLARE
```

```
#define SYMINTEGRATION_CPLUSPLUS_DSOLVE
#ifndef SYMINTEGRATION_CPLUSPLUS_DSOLVE_DECLARE
#define SYMINTEGRATION_CPLUSPLUS_DSOLVE_DECLARE

Symbolic dsolve(const Symbolic &, const Symbolic &, const Symbolic
    &);

#endif
#endif


#endif
```

Code 31: *include/symintegral/dsolve.h*

Then we also need to add a new function **erf**, as the integral of e^{x^2} is not an easy one to be computed and defined, it is like this

$$\int e^{ax^2}\,dx = \frac{\sqrt{\pi}\text{erf}(x\sqrt{-a})}{2\sqrt{-a}}, \quad a \neq 0 \tag{3.10}$$

To handle that, the need to return the $\int e^{ax^2}\,dx$ correctly in SymIntegration we modify these files:

- src/symintegrationc++.cpp

```
...
...

Symbolic erf(const Symbolic &s)
{ return Erf(s); }

...
...
```

Code 32: *src/symintegrationc++.cpp*

- src/functions.cpp

```
...
...


////////////////////////////////////
// Implementation of Erf //
////////////////////////////////////

Erf::Erf(const Erf &s) : Symbol(s) {}

Erf::Erf(const Symbolic &s) : Symbol(Symbol("erf")[s]) {}

Simplified Erf::simplify() const
```

```
{
       const Symbolic &s = parameters.front().simplify();
       if(s == 0) return Number<int>(0);
       return *this;
}

Symbolic Erf::df(const Symbolic &s) const
{ return Derivative(*this,s); }

Symbolic Erf::integrate(const Symbolic &s) const
{
       const Symbolic &x = parameters.front();
       if(x == s) return Integral(*this,s);
       if(df(s) == 0) return Integral(*this,s);
       return Integral(*this,s);
}

...
...
```

Code 33: *src/functions.cpp*

- include/symintegral/functions.h

```
...
...

class Erf;

...
...

class Erf: public Symbol
{
       public: Erf(const Erf&);
       Erf(const Symbolic&);

       Simplified simplify() const;
       Symbolic df(const Symbolic&) const;
       Symbolic integrate(const Symbolic&) const;

       Cloning *clone() const { return Cloning::clone(*this);
           }
};

...
...
```

Code 34: *include/symintegral/functions.h*

- include/symintegral/symintegrationc++.h

```
...
...

Symbolic erf(const Symbolic &);

...
...
```

Code 35: *include/symintegral/symintegrationc++.h*

- Separable Equations

[SI*] We can integrate a differential equation function that is classified as separable equations if they have this form:

$$\frac{dy}{dx} = \frac{f(x)}{g(y)} \tag{3.11}$$

with $f(x)$ is a function of x only and $g(y)$ is a function of y only. Thus we can write the differential equation in the form of

$$g(y)\,dy - f(x)\,dx = 0 \tag{3.12}$$

Such an equation is said to be separable. A separable equation can be solved by integrating the functions f and g.

[SI*] **Example:**
Show that the equation

$$\frac{dy}{dx} = \frac{x^2}{1-y^2}$$

is separable, and then find an equation for its integral curves.
Solution:

$$\begin{aligned} \frac{dy}{dx} &= \frac{x^2}{1-y^2} \\ 1-y^2\,dy &= x^2\,dx \\ y - \frac{y^3}{3} &= \frac{x^3}{3} + c \\ -x^3 + 3y - y^3 &= c \end{aligned}$$

where c is an arbitrary constant and $-x^3 + 3y - y^3 = c$ is the equation for the integral curves in the form of implicit function.

[SI*] **Homogeneous Ratio Equations**
If the right side of the equation

$$\frac{dy}{dx} = f(x,y)$$

can be expressed as a function of the ratio y/x only, then the equation is said to be a homogeneous ratio equation. Such equations can always be transformed into separable equations by a change of the dependent variable.

[SI*] **Example:**
Find the solution of

$$\frac{dy}{dx} = \frac{y-4x}{x-y}$$

Solution:
First we will make it into the form of y/x

$$\frac{dy}{dx} = \frac{y-4x}{x-y} = \frac{(y/x)-4}{1-(y/x)}$$

Make the change of variables

$$v = \frac{y}{x}$$
$$\frac{dv}{dx} = \frac{1}{x}\frac{dy}{dx} - \frac{y}{x^2}$$

$$y = xv$$
$$\frac{dy}{dx} = x\frac{dv}{dx} + \frac{y}{x} = x\frac{dv}{dx} + v$$

As a result, the ODE becomes

$$\begin{aligned}
\frac{dy}{dx} &= \frac{(y/x)-4}{1-(y/x)} \\
x\frac{dv}{dx} + v &= \frac{v-4}{1-v} \\
x\frac{dv}{dx} &= \frac{v-4}{1-v} - v \\
&= \frac{v-4}{1-v} - \frac{v(1-v)}{1-v} \\
&= \frac{v^2-4}{1-v} \\
\frac{1-v}{v^2-4}dv &= \frac{dx}{x} \\
\int \frac{1-v}{(v+2)(v-2)}dv &= \int \frac{dx}{x} \\
\int \left(-\frac{3}{4(v+2)} - \frac{1}{4(v-2)}\right)dv &= \int \frac{dx}{x} \\
-\frac{3}{4}\ln|v+2| - \frac{1}{4}\ln|v-2| &= \ln|x| + C \\
-\frac{3}{4}\ln|\frac{y}{x}+2| - \frac{1}{4}\ln|\frac{y}{x}-2| &= \ln|x| + C
\end{aligned}$$

We can write the solution as an implicit function like this

$$f(x,y) = -\frac{3}{4}\ln|\frac{y}{x}+2| - \frac{1}{4}\ln|\frac{y}{x}-2| - \ln|x| - C$$

Remember that C is a constant that depends on the initial value.

The C++ code to solve the homogeneous ratio equations is using a function named **dsolveseparable** and we need to input it like this
dsolveseparable(g(x,y),f(x,y),y,x)
with

$$\frac{dy}{dx} = \frac{f(x,y)}{g(x,y)}$$

```
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve ]# make
g++    -c -o main.o main.cpp
g++ -o main -ggdb main.o -lstdc++ -lsymintegration
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve ]# ./main


DSolve for dy/dx = (y-4x) / (x-y)

f(x,y) = -0.25*ln(y*x^(-1)-2)-0.75*ln(y*x^(-1)+2)-ln(x)-C
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve ]#
```

Figure 3.2: *The newly built function **dsolveseparable** in SymIntegration is able to compute the solution $f(x,y)$ for homogeneous ratio equations (**SymIntegration/Examples/Test SymIntegration DSolve/main.cpp**).*

Now, to draw a direction field and some integral curves, recall that the right hand side of

$$\frac{dy}{dx} = \frac{y-4x}{x-y}$$

depends only on the ratio $\frac{y}{x}$. This means that the integral curves have the same slope at all points on any given straight line through the origin, although the slope changes from one line to another. Therefore the direction field and the integral curves are symmetric with respect to the origin.

- List of Solvable Linear ODEs with **dsolve**
  We will list all the first order linear ordinary differential equations that can be solved with **dsolve** function in SymIntegration here, they are:

$$\begin{aligned} ay' + ty &= b \\ aty' + by &= ct^2 \\ ay' + by &= ce^{dt} \\ ay' + by &= ct + d \\ ay' + by &= c \\ ay' + by &= 0 \end{aligned}$$

  with a, b, c, d are integers, e is reserved for exponential symbol, the solution will be $y(t)$. Remember when inputting to SymIntegration, you have to input the function $f(t,y)$ with this criteria

$$y' = f(t,y) \tag{3.13}$$

  input the right hand side type of function in term of variables t and y only, and the coefficient of y' is 1.

```
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve ]# make
g++    -c -o main.o main.cpp
g++ -o main -ggdb main.o -lstdc++ -lsymintegration
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve ]# ./main
f(t,y) = -0.5*t*y+1

DSolve for 2y' + ty = 2

y(t) = 1/2*π^(1/2)*(erf(t*(-0.25)^(1/2)))*(-0.25)^(-1/2)*e^(-0.25*t^(2))+C*e^(-0.25*t^(2))

DSolve for ty' + 2y = 4t^2

y(t) = t^(2)+C*t^(-2)

DSolve for y' + 0.5y = 0.5*exp(t/3)

y(t) = 0.6*e^(0.333333*t)+C*e^(-0.5*t)

DSolve for y' - 2y = 4-t

y(t) = 1/2*t+C*e^(2*t)-7/4

DSolve for Q' = r/4 - rQ/100

Q(t) = C*e^(-1/100*t*r)+25

DSolve for S' = rS

S(t) = C*e^(t*r)
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve ]#
```

Figure 3.3: *The function **dsolve** in SymIntegration able to compute the solution $y(t)$ for several examples of linear first order ordinary differential equations (**SymIntegration/Examples/Test SymIntegration DSolve/-main.cpp**).*

- List of Solvable Linear ODEs with **dsolveseparable**
  We will list all the first order linear ordinary differential equations that can be solved with **dsolveseparable** function in SymIntegration here, they are:

$$\frac{dy}{dx} = \frac{f(x)}{g(y)}$$
$$\frac{dy}{dx} = \frac{f(x,y)}{g(x,y)}$$

  We create another function besides **dsolve** to make it easier to separate certain type of differential equation, with this **dsolveseparable** we need to input the function in the numerator and denominator. It is able to compute the simple separable equations and also the homogeneous ratio equations (we use this term instead of only 'homogeneous equations' that is written from this book [1] to avoid clashing of name at the next section).

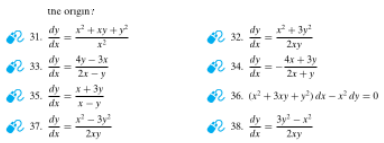

Figure 3.4: *The problems from book [1] with regards to the homogeneous ratio equation that can be solved with SymIntegration' **dsolveseparable**.*

```
]# make
g++    -c -o main.o main.cpp
g++ -o main -ggdb main.o -lstdc++ -lsymintegration
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolveseparable
]# ./main

DSolveseparable for dy/dx = (x^2) / (1-y^2)

f(x,y) = -1/3*y^(3)+y-1/3*x^(3)-C

DSolveseparable for dy/dx = (y-4x) / (x-y)

f(x,y) = -0.25*ln(y*x^(-1)-2)-0.75*ln(y*x^(-1)+2)-ln(x)-C

DSolveseparable for dy/dx = (x^2+xy+y^2) / (x^2)

f(x,y) = atan(y*x^(-1))-ln(x)-C

DSolveseparable for dy/dx = (4y-3x) / (2x-y)

f(x,y) = 0.25*ln(y*x^(-1)-1)-1.25*ln(y*x^(-1)+3)-ln(x)-C

DSolveseparable for dy/dx = (4x+3y) / (2x+y)

f(x,y) = -0.333333*ln(y*x^(-1)+1)-0.666667*ln(y*x^(-1)+4)-ln(x)-C

DSolveseparable for dy/dx = (x+3y) / (x-y)

f(x,y) = -ln(y*x^(-1)+1)-2*(y*x^(-1)+1)^(-1)-ln(x)-C

DSolveseparable for dy/dx = (x^2+3xy+y^2) / (x^2)

f(x,y) = -(y*x^(-1)+1)^(-1)-ln(x)-C

DSolveseparable for dy/dx = (x^2 - 3y^2) / (2xy)

f(x,y) = -0.2*ln(-5*y^(2)*x^(-2)+1)-ln(x)-C

DSolveseparable for dy/dx = (3y^2 - x^2 ) / (2xy)

f(x,y) = ln(y^(2)*x^(-2)-1)-ln(x)-C
```

Figure 3.5: *The solutions with SymIntegration' **dsolveseparable** of the problems from book [1] at the end of chapter 2.2.*

- Modeling with First Order Equations

[SI*] Mathematical modeling and experiment or observation are both critically important and have somewhat complementary roles in scientific investigations. Mathematical models are validated by comparison of their predictions with experimental results.

[SI*] Three identifiable steps that are always present in the process of mathematical modeling:

1. Construction of the Model.
2. Analysis of the Model.
3. Comparison with Experiment or Observation.

[SI*] **The Water Tank Mixing**
At time $t = 0$ a tank contains Q_0 lb of salt dissolved in 100 gal of water. Assume that water containing $\frac{1}{4}$ lb of salt/gal is entering the tank at a rate of r gal/min and that the well-stirred mixture is draining from the tank at the same rate. Set up the initial value problem that describes this flow process. Find the amount of salt $Q(t)$ in the tank at any time, and also find the limiting amount Q_L that is present after a very long time. If $r = 3$ and $Q_0 = 2Q_L$, find the time T after which the salt level is within 2% of Q_L. Also find the flow rate that is required if the value of T is not to exceed 45 min.

Solution:
We assume that salt is neither created nor destroyed in the tank. Therefore variations in

the amount of salt are due solely to the flows in and out of the tank. More precisely, the rate of change of salt in the tank, $\frac{dQ}{dt}$, is equal to the rate at which salt is flowing in minus the rate at which it is flowing out. In symbols,

$$\frac{dQ}{dt} = \text{rate in} - \text{rate out} \tag{3.14}$$

The rate at which salt enters the tank is the concentration $\frac{1}{4}$ lb/gal times the flow rate r gal/min, or $(r/4)$ lb/min. To find the rate at which salt leaves the tank, we need to multiply the concentration of salt in the tank by the rate of outflow, r gal.min. Since the rates of flow in and out are equal, the volume of water in the tank remains constant at 100 gal, and since the mixture is "well-stirred," the concentration throughout the tank is the same, namely, $\frac{Q(t)}{100}$ lb/gal.

Therefore the rate at which salt leaves the tank is $\frac{rQ(t)}{100}$ lb/min. Thus the differential equation governing this process is

$$\frac{dQ}{dt} = \frac{r}{4} - \frac{rQ}{100} \tag{3.15}$$

The initial condition is

$$Q(0) = Q_0 \tag{3.16}$$

Upon thinking about the problem physically, we might anticipate that eventually the mixture originally in the tank will be essentially replaced by the mixture flowing in, whose concentration is $\frac{1}{4}$ lb/gal. Consequently, we might expect that ultimately the amount of salt in the tank would be very close to 25 lb. We can also find the limiting amount $Q_L = 25$ by setting $\frac{dQ}{dt}$ equal to zero in Eq. (

$$\frac{dQ}{dt} = \text{rate in} - \text{rate out} \tag{3.17}$$

The rate at which salt enters the tank is the concentration $\frac{1}{4}$ lb/gal times the flow rate r gal/min, or $(r/4)$ lb/min. To find the rate at which salt leaves the tank, we need to multiply the concentration of salt in the tank by the rate of outflow, r gal.min. Since the rates of flow in and out are equal, the volume of water in the tank remains constant at 100 gal, and since the mixture is "well-stirred," the concentration throughout the tank is the same, namely, $\frac{Q(t)}{100}$ lb/gal.

Therefore the rate at which salt leaves the tank is $\frac{rQ(t)}{100}$ lb/min. Thus the differential equation governing this process is

$$\frac{dQ}{dt} = \frac{r}{4} - \frac{rQ}{100} \tag{3.18}$$

The initial condition is

$$Q(0) = Q_0 \tag{3.19}$$

Upon thinking about the problem physically, we might anticipate that eventually the mixture originally in the tank will be essentially replaced by the mixture flowing in, whose concentration is $\frac{1}{4}$ lb/gal. Consequently, we might expect that ultimately the amount of salt in the tank would be very close to 25 lb. We can also find the limiting

amount $Q_L = 25$ by setting $\frac{dQ}{dt}$ equal to zero in Eq. (3.15) and solving the resulting algebraic equation for Q.

To solve the initial value problem at Eqs (3.15), (3.16) analytically, note that Eq. (3.15) is both linear and separable. Rewriting it in the standard form for a linear equation, we have

$$\frac{dQ}{dt} + \frac{rQ}{100} = \frac{r}{4} \tag{3.20}$$

Thus the integrating factor is $e^{rt/100}$ and the general solution is

$$Q(t) = 25 + ce^{-\frac{rt}{100}} \tag{3.21}$$

where c is an arbitrary constant. To satisfy the initial condition (3.16), we must choose $c = Q_0 - 25$. Therefore the solution of the initial value problem at Eqs (3.15), (3.16) is

$$Q(t) = 25 + (Q_0 - 25)e^{-\frac{rt}{100}} \tag{3.22}$$

or

$$Q(t) = 25(1 - e^{-\frac{rt}{100}}) + Q_0 e^{-\frac{rt}{100}} \tag{3.23}$$

You can see that

$$Q(t) \to 25$$

as $t \to \infty$, so the limiting value Q_L is 25, confirming our physical intuition.

Further, $Q(t)$ approaches the limit more rapidly as r increases. In interpreting the solution (3.23), note that the second term on the right side is the portion of the original salt that remains at time t, while the first term gives the amount of salt in the tank due to the action of the flow processes.

Now suppose that $r = 3$ and $Q_0 = 2Q_L = 50$, then Eq. (3.22) becomes

$$Q(t) = 25 + (50 - 25)e^{-0.03t} \tag{3.24}$$

Since 2% of 25 is 0.5, we wish to find the time T at which $Q(t)$ has the value 25.5. Substituting $t = T$ and $Q(t) = 25.5$ in Eq. (3.24) and solving for T, we obtain

$$\begin{aligned} Q(t) &= 25 + (Q_0 - 25)e^{-\frac{rt}{100}} \\ 25.5 &= 25 + (50 - 25)e^{-\frac{3t}{100}} \\ e^{0.03t} &= \frac{25}{0.5} \\ T &= \frac{\ln|50|}{0.03} \\ &= 130.4 \end{aligned}$$

T is in minutes. To determine r so that $T = 45$, return to Eq. (3.22), set $t = 45, Q_0 =$

$50, Q(t) = 25.5$, and solve for r. The result is

$$\begin{aligned} Q(t) &= 25 + (Q_0 - 25)e^{-\frac{rt}{100}} \\ 25.5 &= 25 + (50 - 25)e^{-\frac{45r}{100}} \\ 0.5 &= 25e^{-\frac{45r}{100}} \\ e^{\frac{45r}{100}} &= \frac{25}{0.5} \\ \frac{45r}{100} &= \ln|50| \\ r &= \frac{100}{45}\ln|50| \\ &\approx 8.69 \end{aligned}$$

r is in gal/min.

We would like to look into the metrics and measurement a bit further, since this is really important. We have the units of gal/min which can be converted into

$$1 \text{ gal/min} = 0.00006309 \ m^3/s = 0.0037854 \ m^3/\text{min} \tag{3.25}$$

Now, we try to do some observations and experimentations with the help of computer simulation, namely Hamzstlab Physics. We will fill an empty tank with a water, that is made by water particle, sphere like, but since it is a 2D computer simulation, we make a circle instead.

And as a representation for the water tank, that is having a cylinder shape, we take the cross section of a cylinder that is like a rectangle.

We observe with a circle of radius $r = 0.2$ and a mass of 0.15 as the water particle that is coming into the empty tank, it takes $t = 45 \ s$ to fill the water tank to be full.

The volume of the water tank itself based on the computer simulation is

$$\begin{aligned} V &= \pi r^2 h \\ &= \pi(5)^2(10) \\ &= 785.39819 \end{aligned}$$

The metrics for V is m^3. So we will have the rate of the entering water as

$$\frac{785.39819 \ m^3}{45 \ s} = 17.453293 \ m^3/s \tag{3.26}$$

Knowing that 1 gal/min $= 0.00006309 \ m^3/s$, we will have the rate of the incoming water as

$$rate_{r=0.2, m=0.15} = \frac{17.453293 \ m^3/s}{0.00006309 \ m^3/s} * (1 \text{ gal/min}) = 276,641.19512 \text{ gal/min} \tag{3.27}$$

We are doing a 6 simulations with Hamzstlab Physics and we gather this data:

sim	$\mathbf{r}(m)$	N	$\mathbf{v}_x(m/s)$	$\mathbf{t}(s)$
1	0.2	23437.5	19.5	45
2	0.2	23437.5	25	44.7
3	0.3	6944.44	19.5	20
4	0.3	6944.44	25	20.3
5	0.4	2929.6875	19.5	11.8
6	0.4	2929.6875	25	11.5

Table 3.1: *The variable t denotes time of completion in seconds till the tank is full, v_x is the velocity of the water particle in m/s toward the x axis, while N is the number of water particle that will fit in a tank of volume $V = 785.4\ m^3$, derived from $\frac{V}{\frac{4}{3}\pi r^3}$.*

From table 3.1 we can then use multivariable regression so we can interpolate and approximate the time of completion, t, for any speed of the water that is filling the tank, for any radius of the water particle and the volume of the water tank. Thus, we will treat t as the independent variable, and the rest is the dependent variables.

Multivariable regression models aim to find the best-fitting equation that describes the relationship between the independent variables and the dependent variable. This equation has the form of

$$y = b_0 + b_1x_1 + b_2x_2 + \cdots + b_nx_n + \epsilon \tag{3.28}$$

where y is the dependent variable, b_0 is the intercept, $b_1, b_2, \ldots, b_n$ are the coefficients for each independent variable $(x_1, x_2, \ldots, x_n)$, and ϵ is the error term.

By analyzing the coefficients, we can understand how each factor relates to time of completion.

In this case we will use the independent variables of: N and v_x, we only need 2, since N is derived from the volume of the water tank divided by the volume of each particle so we don't need to use r anymore, it will become an extra cost and not necessary. Thus

$$x_1 = N$$
$$x_2 = v_x$$

Implementing multivariable regression in C++ typically involves handling matrix operations for calculations of coefficients. This can be achieved by implementing the necessary linear algebra functions from scratch or by utilizing existing C++ libraries designed for numerical linear algebra computation.

You can write your own C++ code from scratch to perform matrix multiplication, transposition, and inversion. But this provides a deep understanding of the underlying mathematics and can be complex, very time consuming and prone to errors.

Thankfully, in 2025 we have a high-performance C++ library that can handle float matrix and float vector operations with great precision, including transpose, inverse and multiplication, we will use **Armadillo / Arma** to help us computing the coefficients b_1 and b_2.

Another alternative besides **Armadillo** is **Eigen**.

Now, suppose we have this matrix that we construct from the simulation data above (table 3.1):

$$X = \begin{bmatrix} 1 & 23437.5 & 19.5 \\ 1 & 23437.5 & 25 \\ 1 & 6944.44 & 19.5 \\ 1 & 6944.44 & 25 \\ 1 & 2929.6875 & 19.5 \\ 1 & 2929.6875 & 25 \end{bmatrix}, \quad y = \begin{bmatrix} 45 \\ 44.7 \\ 20 \\ 20.3 \\ 11.8 \\ 11.5 \end{bmatrix}$$

By any means, our measurements for matrix X and vector y / the simulation data could contain human errors as well.

So we will have this matrix equation for this multivariable regression problem:

$$y = X * B + E \tag{3.29}$$

y is the vector of dependent variable values, X is the matrix of independent variables values (including a column of ones for the intercept term), B is the vector of regression coefficients to be estimated, and E is the vector of error terms.

We will use the most common method to estimate the coefficients B, which is the Ordinary Least Squares (OLS), which minimizes the sum of squared residuals. The formula for OLS coefficient estimation is:

$$B = (X^T * X)^{-1} * X^T * y \tag{3.30}$$

From **Armadillo** we will obtain

$$B = \begin{bmatrix} 8.3696 \\ 0.0016 \\ -0.0182 \end{bmatrix}$$

Thus

$$y = b_0 + b_1 x_1 + b_2 x_2 = 8.3696 + 0.0016N - 0.0182 v_x$$

the equation above tells us about the relation / proportion between y and the independent variables

$$y \propto N$$
$$y \propto \frac{1}{v_x}$$

```
root [ ~/SourceCodes/CPP/C++ Create Library/Armadillo-Test ]# make
g++    -c -o main.o main.cpp
g++ -o main -ggdb main.o -lstdc++ -larmadillo
root [ ~/SourceCodes/CPP/C++ Create Library/Armadillo-Test ]# ./main
Matrix X:
   1.0000e+00   2.3438e+04   1.9500e+01
   1.0000e+00   2.3438e+04   2.5000e+01
   1.0000e+00   6.9444e+03   1.9500e+01
   1.0000e+00   6.9444e+03   2.5000e+01
   1.0000e+00   2.9297e+03   1.9500e+01
   1.0000e+00   2.9297e+03   2.5000e+01
Vector y:
   45.0000
   44.7000
   20.0000
   20.3000
   11.8000
   11.5000

Matrix X^T:
   1.0000e+00   1.0000e+00   1.0000e+00   1.0000e+00   1.0000e+00   1.0000e+00
   2.3438e+04   2.3438e+04   6.9444e+03   6.9444e+03   2.9297e+03   2.9297e+03
   1.9500e+01   2.5000e+01   1.9500e+01   2.5000e+01   1.9500e+01   2.5000e+01

Matrix B:
   8.3696
   0.0016
  -0.0182

Xnew:
   1.0000e+00   6.9444e+03   2.5000e+01

Xnew * B:
   18.9128
```

Figure 3.6: *The matrix B is computed with* ***Armadillo*** *and the predicted t for* $N = 6944.44$ *and* $v_x = 25$ *is* 18.9128, *while the t obtained from the simulation observation is* 20.3, *the error is* 6.83% (***DFSimulatorC/Source Codes/C++/C+++ Gnuplot SymbolicC++/ch23-Numerical Linear Algebra/Multivariable Regression with Armadillo/main.cpp***).

With this multivariable regression we are able to predict the time of completion to fill a water tank with any given volume of the water tank and any given speed of the particle for the filling, or perhaps in other case, to clear up an organ from dangerous substance; how many dosage of drugs should one person drinks with certain mass and age as the independent variables to be healed from certain disease? How many chlorine needed to clear up a pool with certain volume and water quality?

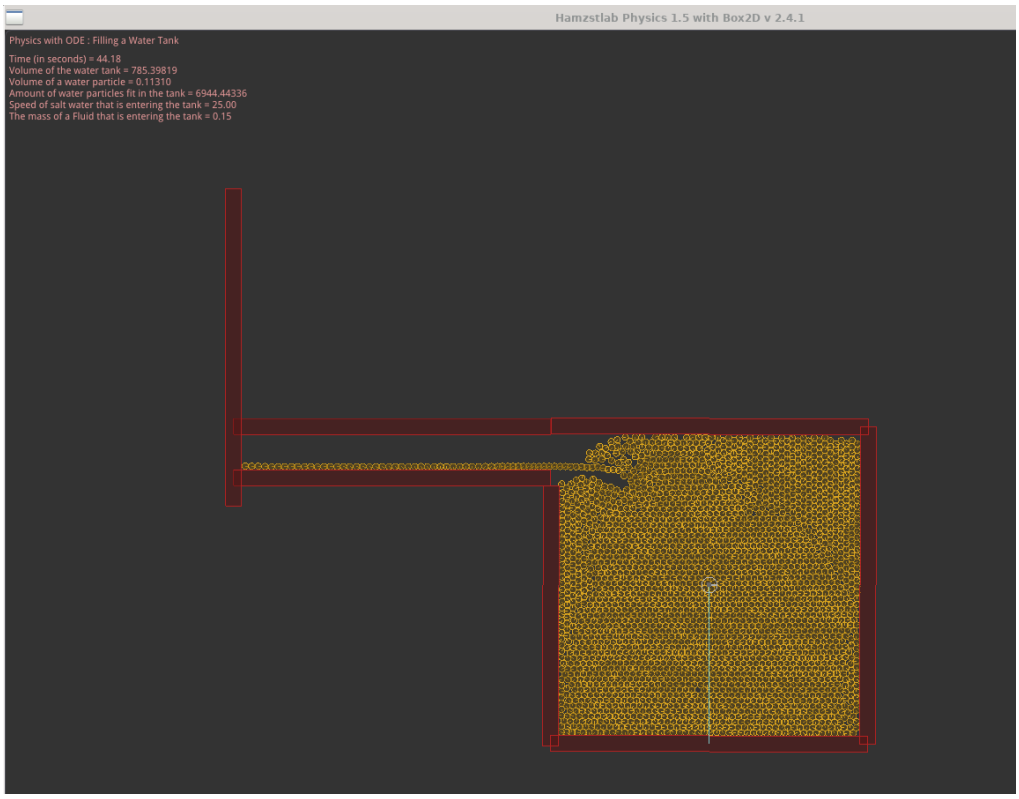

Figure 3.7: *The simulation for a water tank filling with a water particle that has the radius of* $r = 0.4$ *and* $v_x = 25$ *the* t *obtained from the simulation observation is* 11.5 s (***Hamzstlab-Physics2D/tests/ode_watertankfilling.cpp***).

We would like to introduce a book we had made in 2023-2025 that has covered Elementary Linear Algebra for undergraduate level knowledge along with its' C++ code that can be tested by the reader here [4]. We are summarizing the theorems, formulas, definitions, solving the problems from chapter 1 to the last chapter of Chris-Rorres book [8], the addition is that we make the C++ codes ourselves with the help of using already established C++ libraries like **Armadillo, giNaC, Eigen, SymbolicC++**. It is very fortunate that we have learned Linear Algebra first, since it is proven now that it becomes very helpful when we learn other topics such as physics, mechanical engineering, differential equations, and so on. That all the C++ codes that we have made and put in that book is very helpful and can be reused again.

The book can be downloaded and all the source codes are available at:
https://github.com/glanzkaiser/DFSimulatorC

Since this example of water tank mixing is hypothetical, the validity of the model is not in question. If the flow rates are as stated, and if the concentration of salt in the tank is uniform, then the differential equation (3.17) is an accurate description of the flow process.

Model of this kind are often used in problems involving a pollutant in a lake, or a drug

in an organ of the body, for example, rather than a tank of salt water. In such cases the flow rates may not be easy to determine or may vary with time.

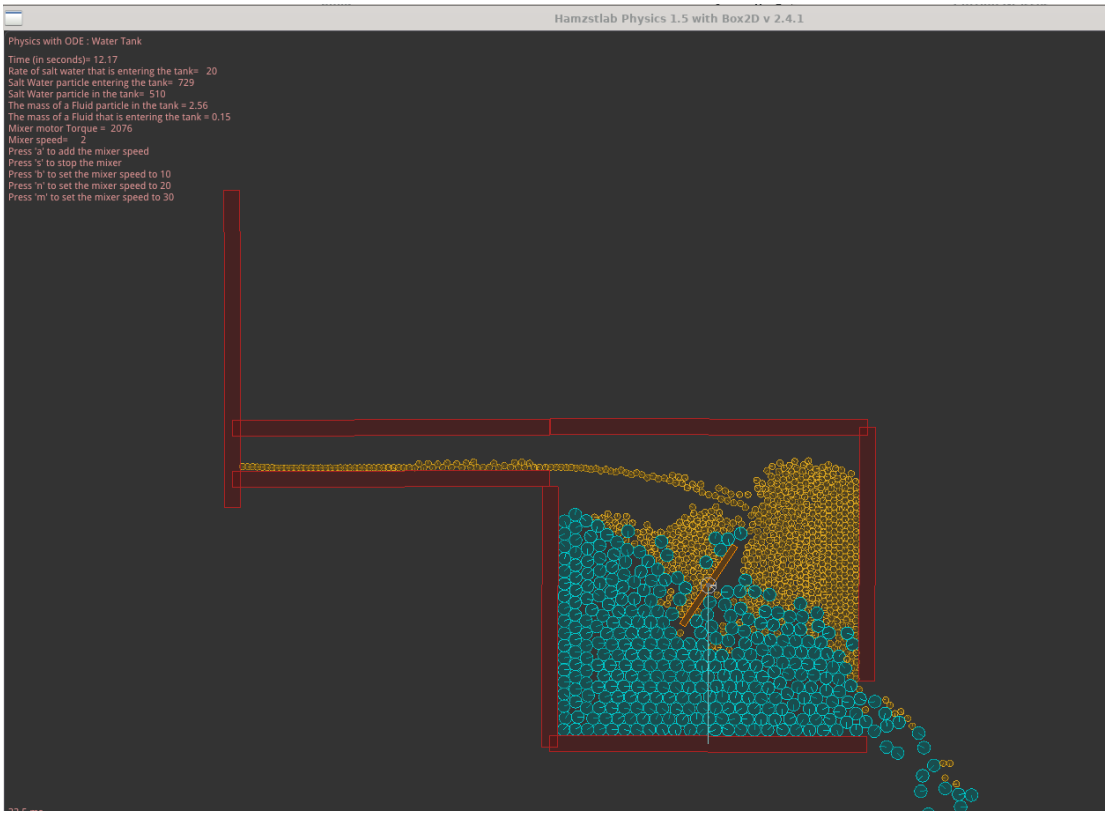

Figure 3.8: *The 2D simulation of a salt water mixing in a tank made with Hamzstlab Physics 2D that is using Box2D version 2.4.2 as the backend library (**Hamzstlab-Physics2D/tests/ode_watertankmixing.cpp**).*

We would like to introduce again one of our creation: **Hamzstlab Physics 2D**, a 2D classical physics simulator, that you can download and try / can be accessed at **https://github.com/glanzkaiser/Hamzstlab-Physics2D**

We make a lot of classical mechanics Physics simulations in 2D with Hamzstlab Physics 2D that is using the backend of Box2D version 2.4.2, at this opportunity we try to make a 2D simulation for this example so people / reader can see the illustration of this water tank mixing with salt water that is coming with certain rate / speed into the tank.

If SymIntegration is able to compute the differential equation solution, and combining it with Hamzstlab Physics we can make a better simulation that is near to real-life physics.

[SI*] **Home Loan / Mortgage**

A home buyer can afford to spend no more than USD 500 / month on mortgage payments. Suppose that the interest rate is 6%, that interest is compounded continuously, and that payments are also made continuously.

(a) Determine the maximum amount that this buyer can afford to borrow on a 20-year

mortgage.

(b) Determine the total interest paid during the term of the mortgage.

Solution:

(a) The amount of money $S(t)$ that the buyer has to pay changes in time due to two factors, the compound interest and its' continuous payments. The rate of growth for compounding is rS, and the rate of decay due to the continuous payments is k.

$$\frac{dS}{dt} = rS - k$$

$$\frac{dS}{dt} - rS = -k$$

This is a linear first-order inhomogeneous ODE, so it can be solved by multiplying both sides by an integrating factor $\mu(t)$

$$\mu(t) = e^{\int^t (-r)\ ds} = e^{-rt}$$

proceed with the multiplication and solve it for $S(t)$

$$\begin{aligned} e^{-rt}\frac{dS}{dt} - re^{-rt}S &= -ke^{-rt} \\ \frac{d}{dt}(e^{-rt}S) &= -ke^{-rt} \\ e^{-rt}S &= \frac{k}{r}e^{-rt} + C \\ S(t) &= \frac{k}{r} + Ce^{rt} \end{aligned}$$

Apply the initial condition $S(0) = S_0$ to determine C

$$\begin{aligned} S(0) &= \frac{k}{r} + Ce^{rt} \\ S_0 &= \frac{k}{r} + Ce^{rt} \\ C &= S_0 - \frac{k}{r} \end{aligned}$$

Therefore, the amount of money the buyer has to pay after t years is

$$\begin{aligned} S(t) &= \frac{k}{r} + \left(S_0 - \frac{k}{r}\right)e^{rt} \\ &= \frac{k}{r}(1 - e^{rt}) + S_0 e^{rt} \end{aligned}$$

For a year the home buyer will need to pay $k = 500 \times 12 = USD\ 6,000$ year and $r = 6\% = 0.06$.

$$S(t) = \frac{6000}{0.06}(1 - e^{0.06t}) + S_0 e^{0.06t}$$

For a 20-year mortgage, $S(t) = 0$ at $t = 20$.

$$0 = \frac{6000}{0.06}(1 - e^{0.06t}) + S_0 e^{0.06t}$$
$$S_0 = 100000 - 100000e^{-1.2}$$
$$S_0 \approx USD\ 69,880.58$$

The value of S_0 is how much the buyer can afford to borrow initially.

(b) For the 20-year mortgage, the home buyer pays a total of

$$USD\ 6,000 \times 20 = USD\ 120,000$$

so the total interest paid is

$$USD\ 120,000 - USD\ 69,880.58 = USD\ 50,119.42$$

```
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve and IVP f
or Mortgage case ]# make
g++    -c -o main.o main.cpp
g++ -o main -ggdb main.o -lstdc++ -lsymintegration
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve and IVP f
or Mortgage case ]# ./main

DSolve for S' = rS - k

S(t) = k*r^(-1)+C*e^(t*r)
S(t=0) = k*r^(-1)+C
For ivp S(0) = S0,
C = -k*r^(-1)+S0

Test ivp = k*r^(-1)-k*r^(-1)*e^(t*r)+S0*e^(t*r)
For ivp k = 6000, r = 6%, t = 20, S(t) = 0,
S(0) = -100000*e^(-1.2)+100000
```

Figure 3.9: *The function **dsolve** and **ivp** in SymIntegration is able to compute the solution $S(t)$ for the ordinary differential equation of $\frac{dS}{dt} = Sr - k$ with the initial value problem $S(0) = S_0$ (**SymIntegration/Examples/Test SymIntegration DSolve and IVP for Mortgage case/main.cpp**).*

As of July 20th, 2025 we just add a new function **ivp**, and it is managed in **/usr/include/symintegral/dsolve.h** and **/src/dsolve.cpp**

```
...
...

Symbolic ivp(const Symbolic &fx, const Symbolic &x, const Symbolic &c)
{
	Symbolic ivpsol, f0, C("C"), S0("S0");

	if(fx != 0)
	{
		list<Equations> eq;
		list<Equations>::iterator i;
		UniqueSymbol a, b;
		// Will work for this model: dS/dt = rS—k
		f0 = fx[x==0];
		C = solve(f0—S0,C).front().rhs ;
		ivpsol = fx[c==C];
```

```
        }
        return ivpsol;
}

...
```

Code 36: *src/dsolve.cpp*

[SI*] **Radiocarbon Dating Example:**
An important tool in archeological research is radiocarbon dating, developed by the American chemist Willard F. Libby. This is a means of determining the age of certain wood and plant remains, hence of animal or human bones or artifacts found buried at the same levels.

Radiocarbon dating is based on the fact that some wood or plant remains contain residual amounts of carbon-14, a radioactive isotope of carbon. This isotope is accumulated during the lifetime of the plant and begins to decay at its death. Since the half-life of carbon-14 is long (approximately 5730 years), measurable amounts of carbon-14 is still present, then by appropriate laboratory measurements the proportion of the original amount of carbon-14 that remains can be accurately determined. In other words, if $Q(t)$ is the amount of carbon-14 at time t and Q_0 is the original amount, then the ratio $\frac{Q(t)}{Q_0}$ can be determined, as long as this quantity is not too small. Present measurement techniques permit the use of this method for time periods of 50,000 years or more.

(a) Assuming that Q satisfies the differential equation $Q' = -rQ$, determine the decay constant r for carbon-14.

(b) Find an expression for $Q(t)$ at any time t, if $Q(0) = Q_0$

(c) Suppose that certain remains are discovered in which the current residual amounts of carbon-14 is 20% of the original amount. Determine the age of the remains.

Solution:

(a) The rate that the mass of carbon-14 decreases is assumed to be proportional to how much is present at any given time

$$\frac{dQ}{dt} \propto -Q$$

Change this proportionality to an equation by introducing a constant r on the right side.

$$Q' = -rQ$$

Divide both sides by Q.

$$\frac{Q'}{Q} = -r$$

The left side can be written as the derivative of a logarithm by the chain rule.

$$\begin{aligned} \frac{d}{dt} \ln\ Q &= -r \\ \ln\ Q &= -rt + C \\ Q(t) &= e^{-rt+C} \\ &= ce^{-rt} \end{aligned}$$

(b) Suppose that there is a certain amount of mass Q_0 initially. Then the initial condition is $Q(0) = Q_0$. We will use it to determine c.

$$\begin{aligned} Q(0) &= ce^{-r(0)} \\ c &= Q_0 \end{aligned}$$

Consequently, the mass decays exponentially. Substituting back we will have the solution to the initial value problem as

$$Q(t) = Q_0 e^{-rt}$$

(c) From the formula that we obtain at (b) we know that r can be determined with knowledge of the half-life. For carbon-14, specifically, we know that half the initial mass is lost after 5730 years.

$$\frac{Q_0}{2} = Q_0 e^{-r(5730)}$$

Solve this equation for r

$$\begin{aligned} e^{-5730r} &= \frac{1}{2} \\ \ln|e^{-5730r}| &= \ln\left|\frac{1}{2}\right| \\ r &= -\frac{\ln\left|\frac{1}{2}\right|}{5730} \\ &= \frac{\ln|2|}{5730} \\ r &\approx 0.0001210 \end{aligned}$$

r is the rate of the decreasing mass per year. Therefore, the mass of a sample of carbon-14 is

$$Q(t) = Q_0 e^{-\frac{\ln|2|}{5730}t}$$

where t is in years. If some remains have 20% of the original amount of carbon-14, then $Q(t) = 0.2Q_0$. Solve the resulting equation for t to determine how old the remains are.

$$\begin{aligned} 0.2Q_0 &= Q_0 e^{-\frac{\ln|2|}{5730}t} \\ e^{-\frac{\ln|2|}{5730}t} &= 0.2 \\ \ln|e^{-\frac{\ln|2|}{5730}t}| &= \ln|0.2| \\ -\frac{\ln|2|}{5730}t &= \ln\frac{1}{5} \\ -\frac{\ln|2|}{5730}t &= -\ln|5| \\ t &= \frac{\ln|5|}{\ln|2|}5730 \\ t &\approx 13,305 \end{aligned}$$

so the age of the remains are around 13,305 years old.

```
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve and IVP for Radiocarbon Dating ]# make
g++    -c -o main.o main.cpp
g++ -o main -ggdb main.o -lstdc++ -lsymintegration
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve and IVP for Radiocarbon Dating ]# ./ma
in

DSolve for Q' = -rQ

Q(t) = C*e^(-t*r)
Q(t=0) = C
For ivp Q(0) = Q0,
C = Q0

Solution for the ivp,
Q(t) = Q0*e^(-t*r)

Half-life problem for carbon-14, the ivp becomes:
Q0*e^(-5730*r)-0.5*Q0 = 0

Determining the rate by solving the ivp solution
r = 0.000120968

Determining the age of the remains with 20% of carbon-14
t = 13304.6
```

Figure 3.10: *The computation of the rate and the age of remains with SymIntegration' **dsolve** and **ivp** functions. (**SymIntegration/Examples/Test SymIntegration DSolve and IVP for Radiocarbon Dating/-main.cpp**).*

On this example, we have added another **else if** conditional so SymIntegration' **solve** function will be able to solve this equation:

$$ce^{-rx} = a$$

$$ce^{-rx} = a$$

$$x = -\frac{\ln \frac{a}{c}}{r} = \frac{\ln \frac{c}{a}}{r}$$

```
...
double division(Symbolic x, Symbolic y)
{
       return x/y;
}

Equations solve(const Symbolic &e, const Symbolic &x)
{
Equations soln;

...
...

else if(e.coeff(1,0) == e) // for case c * exp(—tx) = a or ln|tx| = a
{
       if(df(e,x).coeff(x,0) != 0 && df(df(e,x),x).coeff(x,0) != df(e,x).
           coeff(x,0) && —df(df(e,x),x).coeff(x,0) != df(e,x).coeff(x,0)
           ) // for case c * exp(—tx) = a ,
       {
              Symbolic t = df(df(e,x),x).coeff(x,0) / df(e,x).coeff(x,0);
                   // will work for c == 1 or c != 1
              Symbolic a = —e.coeff(SymbolicConstant::e,0) ;
              Symbolic c = df(e,x).coeff(x,0)/t;
              soln = (soln, x == ln(division(a,c))/t);
```

```
        }
        else if(df(e,x).coeff(x,0) != 0 && -df(df(e,x),x).coeff(x,0) ==
            df(e,x).coeff(x,0) ) // for case c * exp(-tx) = a , c !=1, t
            ==1
        {
               Symbolic a = - e.coeff(SymbolicConstant::e,0) ;
               Symbolic c = - df(e,x).coeff(x,0) ;
               soln = (soln, x == ln(division(c,a)));
        }
        ...
}
return soln;
}
...
```

Code 37: *src/solve.cpp*

[SI*] **Pollutants in a Lake:**
Consider a lake of constant volume V containing at time t an amount $Q(t)$ of pollutant, evenly distributed throughout the lake with a concentration $c(t)$, where $c(t) = \frac{Q(t)}{V}$. Assume that water containing a concentration k of pollutant enters the lake at a rate r, and that water leaves the lake at the same rate. Suppose that pollutants are also added directly to the lake at a constant rate P.

Note that given assumptions neglect a number of factors that may, in some cases, be important-for example, the water added or lost by precipitation, absorption, and evaporation; the stratifying effect of temperature differences in a deep lake; the tendency of irregularities in the coastline to produce sheltered bays; and the fact that pollutants are not deposited evenly throughout the lake but (usually) at isolated points around its periphery. The results below must be interpreted in the light of the neglect of such factors as these.

(a) If at time $t = 0$ the concentration of pollutant is c_0, find an expression for the concentation $c(t)$ at any time. What is the limiting concentration as $t \to \infty$?

(b) If the addition of pollutants to the lake is terminated ($k = 0$ and $P = 0$ for $t > 0$), determine the time interval T that must elapse before the concentration of pollutants is reduced to 50% of its original value; to 10% of its original value.

(c) Table 5.1 contains data for several of the Great Lakes. Using these data, determine from part (b) the time T necessary to reduce the contamination of each of these lakes to 10% of the original value.

Solution:

(a) The conservation law that governs the mass of pollutant in the lake is as follows.

rate of mass accumulation = rate of mass in - rate of mass out

Lake	V($km^3 \times 10^3$)	r($km^3/year$)
Superior	12.2	65.2
Michigan	4.9	158
Erie	0.46	175
Ontario	1.6	209

Table 3.2: *Volume and Flow Data for the Great Lakes*

the rate of mass in is the sum of rk and P, and the rate of mass out is $rc(t)$.

$$\begin{aligned}\frac{dQ}{dt} &= rk + P - rc(t)\\ &= rk + P - \frac{r}{V}Q(t)\\ \frac{dQ}{dt} + \frac{r}{V}Q &= rk + P\end{aligned}$$

This is a first-order linear inhomogeneous ODE, so it can be solved by multiplying both sies by an integrating factor $\mu(t)$.

$$\mu(t) = e^{\int \frac{r}{V}\, ds} = e^{\frac{rt}{V}}$$

$$\begin{aligned}e^{\frac{rt}{V}}\frac{dQ}{dt} + \frac{r}{V}e^{\frac{rt}{V}}Q &= rke^{\frac{rt}{V}} + Pe^{\frac{rt}{V}}\\ \frac{d}{dt}(e^{\frac{rt}{V}}Q) &= rke^{\frac{rt}{V}} + Pe^{\frac{rt}{V}}\\ e^{\frac{rt}{V}}Q &= \int^t (rke^{\frac{rs}{V}} + Pe^{\frac{rs}{V}})\, ds + C\\ &= rk\left(\frac{V}{r}\right)e^{\frac{rt}{V}} + P\left(\frac{V}{r}\right)e^{\frac{rt}{V}} + C\\ e^{\frac{rt}{V}}Q &= kVe^{\frac{rt}{V}} + \frac{PV}{r}e^{\frac{rt}{V}} + C\\ Q(t) &= kV + \frac{PV}{r}e^{\frac{rt}{V}} + Ce^{\frac{-rt}{V}}\end{aligned}$$

Divide both sides by V to obtain the concentration.

$$c(t) = \frac{Q(t)}{V} = k + \frac{P}{r} + \frac{C}{V}e^{\frac{-rt}{V}}$$

Apply the initial condition $c(0) = c_0$ to determine C.

$$\begin{aligned}c(0) &= k + \frac{P}{r} + \frac{C}{V}\\ c_0 &= k + \frac{P}{r} + \frac{C}{V}\\ \frac{C}{V} &= c_0 - k - \frac{P}{r}\end{aligned}$$

Therefore, the concentration is

$$c(t) = \frac{Q(t)}{V} = k + \frac{P}{r} + \left(c_0 - k - \frac{P}{r}\right) e^{\frac{-rt}{V}}$$

Because of the decaying exponential function, the limit of $c(t)$ as $t \to \infty$ is

$$\lim_{t\to\infty} c(t) = k + \frac{P}{r}$$

(b) If the pollution stops, then the rate of mass in becomes zero in the conservation law.

rate of mass accumulation = - rate of mass out

The rate of mass out is still $rc(t)$.

$$\begin{aligned}
\frac{dQ}{dt} &= -rc(t) \\
&= -\frac{r}{V}Q(t) \\
\frac{\frac{dQ}{dt}}{Q} &= -\frac{r}{V} \\
\frac{dQ}{Q} &= -\frac{r}{V}\, dt \\
\ln Q &= -\frac{rt}{V} + C_1 \\
Q &= e^{-\frac{rt}{V}+C_1} \\
Q(t) &= C_2 e^{-\frac{rt}{V}}
\end{aligned}$$

Since Q is the mass of pollutant, divide both sides by V to get the concentration.

$$c(t) = \frac{Q(t)}{V} = \frac{A_1}{V} e^{-\frac{rt}{V}}$$

Use the initial condition $c(0) = c_0$ to determine C_2

$$c(0) = \frac{C_2}{V} = c_0$$

Therefore

$$c(t) = c_0 e^{-\frac{rt}{V}}$$

Set $c(t) = 0.5c_0$ and solve for $t = T$ to find how long it will take for the concentration to reach half its initial value.

$$\begin{aligned}
0.5c_0 &= c_0 e^{-\frac{rT}{V}} \\
e^{-\frac{rT}{V}} &= 0.5 \\
\ln e^{-\frac{rT}{V}} &= \ln 0.5 \\
-\frac{rT}{V} &= \ln \frac{1}{2} \\
T &= \frac{V}{r} \ln 2
\end{aligned}$$

On the other hand, set $c(t) = 0.1c_0$ and solve for $t = T$ to find how long it will take for the concentration to reach one-tenth its initial value.

$$\begin{aligned} 0.1c_0 &= c_0 e^{-\frac{rt}{V}} \\ e^{-\frac{rt}{V}} &= 0.1 \\ \ln e^{-\frac{rt}{V}} &= \ln 0.1 \\ -\frac{rT}{V} &= \ln \frac{1}{10} \\ T &= \frac{V}{r} \ln 10 \end{aligned}$$

(c) Plug in the numbers fo V and r into the formula at part (b).
Lake Superior:

$$T = \frac{V}{r} \ln 10 = \frac{12.2 \times 10^3}{65.2} \ln 10 \approx 431 \; \textit{years}$$

Lake Michigan:

$$T = \frac{V}{r} \ln 10 = \frac{4.9 \times 10^3}{158} \ln 10 \approx 71.4 \; \textit{years}$$

Lake Erie:

$$T = \frac{V}{r} \ln 10 = \frac{0.46 \times 10^3}{175} \ln 10 \approx 6.05 \; \textit{years}$$

Lake Ontario:

$$T = \frac{V}{r} \ln 10 = \frac{1.6 \times 10^3}{209} \ln 10 \approx 17.6 \; \textit{years}$$

```
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve and IVP for Pollutant in Great Lakes ]# make
g++    -c -o main.o main.cpp
g++ -o main -ggdb main.o -lstdc++ -lsymintegration
root [ ~/SourceCodes/CPP/C++ Create Library/Test SymIntegration DSolve and IVP for Pollutant in Great Lakes ]# ./main

DSolve for Q' = rk +  P - rc(t) ,        c(t) = Q(t)/V

Q(t) = k*V+P*V*r^(-1)+C*e^(-V^(-1)*t*r)
c(t) = k+P*r^(-1)+C*e^(-V^(-1)*t*r)*V^(-1)
c(0) = k+P*r^(-1)+C*V^(-1)
c0 - c(0) = c0-k-P*r^(-1)-C*V^(-1)
For ivp c(0) = c0,
C/V = c0-k-P*r^(-1)

Therefore, the concentration is,
c(t) = k+P*r^(-1)+c0*e^(-V^(-1)*t*r)-k*e^(-V^(-1)*t*r)-P*r^(-1)*e^(-V^(-1)*t*r)

The limit of c(t) as t -> ∞
lim_{t -> ∞} c(t) = k+P*r^(-1)+c0*e^(-inf*V^(-1)*r)-k*e^(-inf*V^(-1)*r)-P*r^(-1)*e^(-inf*V^(-1)*r)

If the pollution stops, then
Q(t) = C*e^(-V^(-1)*t*r)

The concentration when the pollution stops is
c(t) = C*e^(-V^(-1)*t*r)*V^(-1)

For ivp c(0) = c0,
c0-C*V^(-1) = 0

Therefore the ivp solution will be,
c(t) = c0*e^(-V^(-1)*t*r)

Set c(t) = 0.5 c0 and solve for t=T,
T = 0.693147*V*r^(-1)

Set c(t) = 0.1 c0 and solve for t=T,
T = 2.30259*V*r^(-1)

Time to reduce the contamination in each lakes to 10 percent of the original value

Lake Superior:  T = 430.852

Lake Michigan:  T = 71.4093

Lake Erie:      T = 6.05251

Lake Ontario:   T = 17.6274
```

Figure 3.11: *The computation of initial value problem solution for the pollutant concentration in a lake, $c(t)$, and the time to clean up a lake to make the pollutant concentration less than* 10% *with SymIntegration'* ***dsolve*** *function. (**SymIntegration/Examples/Test SymIntegration DSolve and IVP for Pollutant in Great Lakes/main.cpp**).*

II. Higher Order Linear Equations

- General Theory of nth Order Linear Equations

[SI*] The first
[SI*]

- The Method of Variation of Parameters

[SI*] The first
[SI*]

III. Boundary Value Problems and Sturm-Liouville Theory

- The Occurrence of Two-Point Boundary Value Problems

  [SI*] The first
  [SI*]

- Nonhomogeneous Boundary Value Problems

  [SI*] The first
  [SI*]

Bibliography

[1] Boyce, William E., DiPrima, Richard C. (2010) Elementary Differential Equations and Boundary Value Problems 9th Edition, John Wiley & Sons, Hoboken, New Jersey, United States.

[2] Colley, Susan Jane (2012) Vector Calculus 4th Edition, Pearson, Boston, MA, United States.

[3] Dorf, Richard C., Bishop, Robert H. (2010) Modern Control Systems 12th Edition, Pearson, New Jersey, United States.

[4] Freya, Glanzsche, DS (2025) DianFreya Math Physics Simulator, Berlin-Sentinel Academy of Science, AliceGard.

[5] Lasthrim, Glanzsche, DS, Freya (2025) Lasthrim Projection 1st Edition, Berlin-Sentinel Academy of Science, AliceGard.

[6] Purcell, Rigdon, Varberg (2007) Calculus 9th Edition, Pearson, Upper Saddle River, New Jersey, United States.

[7] Kenneth H. Rosen (2006) Discrete Mathematics and its Applications 6th Edition, McGraw-Hill, Boston, United States.

[8] Anton, Rorres (2006) Elementary Linear Algebra with Supplemental Applications 10th Edition, Wiley, Boston, United States.

[9] Anton, Howard, Rorres, Chris, Elementary Linear Algebra with Supplemental Applications 12th edition, Wiley, Hoboken, New Jersey, United States.