User

See the code below given by you in the previous session:

#include <boost/math/quadrature/gauss\_kronrod.hpp>
#include <cmath>
#include <iostream>

double f(double x) {
return std::log(x);
}

double F(double x) {
return x\*std::log(x) - x; // Indefinite integral of log(x)
}

int main() {
using namespace boost::math::quadrature;

double a = 1; // lower limit of integration

double b = 2; // upper limit of integration

gauss\_kronrod<double, 15> integrator;

double Q = integrator.integrate(f, a, b);

std::cout << "The indefinite integral of log(x) dx is " << "x\*log(x) - x + C" << std::endl;

std::cout << "The definite integral of log(x) dx from " << a << " to " << b << " is " << Q << std::endl;

return 0;

}

Particularly this part:
std::cout << "The indefinite integral of log(x) dx is " << "x\*log(x) - x + C" << std::endl;

You have not arrived at solution of indenifite integral using C++ Boost library but manually worked on it and just given the output - I need solution using any c++ library (not necessarily boost if that one does not provide indefinite integration derivations)

Bing

I understand your requirement. Unfortunately, there isn't a C++ library that directly provides indefinite integration. The reason is that indefinite integration results in a family of functions, represented by the integral plus an arbitrary constant (C). This constant cannot be determined without additional information, such as an initial condition.

However, if you're working with polynomials, you can write a simple function to calculate the indefinite integral[**2**](https://stackoverflow.com/questions/2982167/integration-math-in-c). Here's an example:

#include <vector>

std::vector<double> indefiniteIntegral(const std::vector<double>& coeffs) {

std::vector<double> result(coeffs.size() + 1);

for (size\_t i = 0; i < coeffs.size(); ++i) {

result[i+1] = coeffs[i] / (i + 1);

}

result[0] = 0; // The constant of integration, C, is typically set to 0

return result;

}

In this function, coeffs is a vector of coefficients of the polynomial you want to integrate. The function returns a new vector of coefficients representing the indefinite integral of the input polynomial.

Please note that this function only works for polynomials. For other types of functions, indefinite integration might not be possible to express in terms of elementary functions. In such cases, numerical methods or special functions might be needed, and the result can only be represented as a definite integral over a specific interval.

Source(s)

1. [math - C++ polynomials: indefinite integrals - Stack Overflow](https://stackoverflow.com/questions/22619803/c-polynomials-indefinite-integrals)

2. [Integration (math) in C++ - Stack Overflow](https://stackoverflow.com/questions/2982167/integration-math-in-c)

3. [GitHub - CD3/libIntegrate: A C++ library for integrating discretized ...](https://github.com/CD3/libIntegrate)

4. [C math library with integration? - Stack Overflow](https://stackoverflow.com/questions/1564543/c-math-library-with-integration)

5. [How to calculate indefinite integral programmatically](https://stackoverflow.com/questions/3239336/how-to-calculate-indefinite-integral-programmatically)

User

But I do not get how to use this. What is the input. Suppose I tell you to modify or apply the above code to find integration of log(x) or sin(x), can you do that - If so, provide me the code.

Bing

I understand your confusion. The function I provided earlier works for polynomials, but log(x) and sin(x) are not polynomials. They are transcendental functions, and their antiderivatives cannot be expressed in terms of elementary functions.

However, the antiderivatives of log(x) and sin(x) are well-known:

* The antiderivative of log(x) is x\*log(x) - x + C.
* The antiderivative of sin(x) is -cos(x) + C.

Here's how you can define these functions in C++:

#include <cmath>

double antiderivative\_log(double x) {

return x\*std::log(x) - x;

}

double antiderivative\_sin(double x) {

return -std::cos(x);

}

In these functions, x is the input, and C is the constant of integration. You can add C to the result if you know its value.

Please note that these functions give you the antiderivative at a specific point x. They do not perform indefinite integration in the sense of finding a function that represents the antiderivative for all x. That's a more complex problem that typically requires symbolic computation, which is beyond the scope of standard C++ libraries. For symbolic computation, you might want to look into libraries like SymPy, but note that it's a Python library. There's also the possibility of using software like Mathematica or Maple, but these are not libraries and are not free. They are full-featured software packages for symbolic computation.

I hope this helps! Let me know if you have any other questions. 😊

User

Again, in the above code, you have indirectly provided solution and function both in the code. If Python can do this, why not C++ which is even a lower level computer programming? By the way, is it possible even in Python - indifinite integral? If not then how online indefinite integration derivation sites work. Surely, there must be options which they use.

Bing

You're correct that Python has libraries like SymPy[**2**](https://how.okpedia.org/en/python/how-to-calculate-an-indefinite-integral-in-python) that can perform symbolic computation, including indefinite integration[**2**](https://how.okpedia.org/en/python/how-to-calculate-an-indefinite-integral-in-python)[**3**](https://www.symbolab.com/solver/indefinite-integral-calculator). This is possible because Python supports dynamic typing and has extensive support for symbolic computation, which is not typically found in statically-typed languages like C++.

In Python, you can use the integrate function from the SymPy library to compute the indefinite integral of a function[**2**](https://how.okpedia.org/en/python/how-to-calculate-an-indefinite-integral-in-python)[**3**](https://www.symbolab.com/solver/indefinite-integral-calculator). Here's an example:

import sympy as sp

x = sp.symbols('x')

f = sp.log(x) # The function to integrate

antiderivative = sp.integrate(f, x)

print(antiderivative)

This code will output x\*log(x) - x, which is the antiderivative of log(x).

As for how online indefinite integration derivation sites work, they typically use symbolic computation libraries similar to SymPy[**5**](https://docs.scipy.org/doc/scipy/tutorial/integrate.html)[**6**](https://www.geeksforgeeks.org/how-to-find-definite-integral-using-python/). These libraries can manipulate mathematical expressions symbolically, allowing them to compute antiderivatives exactly[**5**](https://docs.scipy.org/doc/scipy/tutorial/integrate.html)[**6**](https://www.geeksforgeeks.org/how-to-find-definite-integral-using-python/).

However, it's important to note that not all functions have an antiderivative that can be expressed in terms of elementary functions. In such cases, these tools might return the original integral as the "antiderivative", or they might use special functions to express the antiderivative[**6**](https://www.geeksforgeeks.org/how-to-find-definite-integral-using-python/).

I hope this answers your question! Let me know if you have any other questions. 😊

Source(s)

1. [How to Calculate Definite and Indefinite Integrals in Python](https://www.freecodecamp.org/news/calculate-definite-indefinite-integrals-in-python/)

2. [[Python] How to Calculate Indefinite Integral - Okpedia](https://how.okpedia.org/en/python/how-to-calculate-an-indefinite-integral-in-python)

3. [Indefinite Integral Calculator - Symbolab](https://www.symbolab.com/solver/indefinite-integral-calculator)

4. [Integral Calculator • With Steps!](https://www.integral-calculator.com/)

5. [Integration (scipy.integrate) — SciPy v1.11.4 Manual](https://docs.scipy.org/doc/scipy/tutorial/integrate.html)

6. [How to find Definite Integral using Python - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-find-definite-integral-using-python/)

7. [Indefinite Integrals - Definition, Properties, Formulas & Examples - BYJU'S](https://byjus.com/maths/indefinte-integrals/)

8. [Indefinite Integrals - Definition, Formulas, Properties, Examples](https://www.geeksforgeeks.org/indefinite-integrals/)

9. [Indefinite Integral - Definition, Calculate, Formulas - Cuemath](https://www.cuemath.com/calculus/indefinite-integral/)

User

What is the backend code of sympy derived from? Must be C if not C++. If so whether you can provide me C code to work out and output integration?

Bing

SymPy is a Python library for symbolic mathematics[**4**](https://stackoverflow.com/questions/1564543/c-math-library-with-integration). It's written entirely in Python and does not have a C or C++ backend[**1**](https://github.com/sympy)[**2**](https://docs.sympy.org/latest/modules/plotting.html). SymPy uses Python's capabilities for dynamic typing and symbolic computation to perform tasks like indefinite integration[**2**](https://docs.sympy.org/latest/modules/plotting.html).

As for your question about a C or C++ library for indefinite integration, the GNU Scientific Library (GSL) provides functions for numerical integration[**6**](https://www.symbolab.com/solver/indefinite-integral-calculator). However, these functions compute definite integrals over a specific interval, not indefinite integrals[**6**](https://www.symbolab.com/solver/indefinite-integral-calculator).

Indefinite integration, which involves finding a function that represents the antiderivative of another function, typically requires symbolic computation[**7**](https://docs.sympy.org/latest/modules/codegen.html). Symbolic computation is not a standard feature of C or C++ and is not provided by the GSL or other common C/C++ math libraries[**6**](https://www.symbolab.com/solver/indefinite-integral-calculator).

Online tools for indefinite integration, like Symbolab[**8**](https://github.com/sympy/sympy/blob/master/sympy/core/backend.py), typically use symbolic computation libraries similar to SymPy[**5**](https://www.geeksforgeeks.org/mathematics-indefinite-integrals/)[**6**](https://www.symbolab.com/solver/indefinite-integral-calculator). These libraries can manipulate mathematical expressions symbolically, allowing them to compute antiderivatives exactly[**5**](https://www.geeksforgeeks.org/mathematics-indefinite-integrals/)[**6**](https://www.symbolab.com/solver/indefinite-integral-calculator).

In conclusion, while C and C++ are powerful languages for many tasks, they do not natively support symbolic computation. Libraries like SymPy in Python fill this gap by providing symbolic computation capabilities, including indefinite integration[**2**](https://docs.sympy.org/latest/modules/plotting.html)[**3**](https://stackoverflow.com/questions/43869393/how-do-i-use-the-backend-attribute-of-a-sympy-plot). If you need to perform indefinite integration in C or C++, you might need to use a library with symbolic computation capabilities or implement your own symbolic computation functions. However, such libraries are not common, and implementing symbolic computation from scratch is a complex task.

I hope this answers your question! Let me know if you have any other questions. 😊.

Source(s)

1. [SymPy · GitHub](https://github.com/sympy)

2. [Plotting - SymPy 1.12 documentation](https://docs.sympy.org/latest/modules/plotting.html)

3. [How do I use the `\_backend` attribute of a Sympy `plot`](https://stackoverflow.com/questions/43869393/how-do-i-use-the-backend-attribute-of-a-sympy-plot)

4. [C math library with integration? - Stack Overflow](https://stackoverflow.com/questions/1564543/c-math-library-with-integration)

5. [Mathematics | Indefinite Integrals - GeeksforGeeks](https://www.geeksforgeeks.org/mathematics-indefinite-integrals/)

6. [Indefinite Integral Calculator - Symbolab](https://www.symbolab.com/solver/indefinite-integral-calculator)

7. [Code Generation - SymPy 1.12 documentation](https://docs.sympy.org/latest/modules/codegen.html)

8. [sympy/backend.py at master · sympy/sympy · GitHub](https://github.com/sympy/sympy/blob/master/sympy/core/backend.py)

9. <http://root.cern.ch/root/html/ROOT_>

10. [en.wikipedia.org](https://en.wikipedia.org/wiki/SymPy)

User

What is the native language for sympy?

Bing

SymPy is written entirely in Python. It's a Python library for symbolic mathematics. It does not have a C or C++ backend. SymPy uses Python's capabilities for dynamic typing and symbolic computation to perform tasks like indefinite integration. I hope this answers your question! Let me know if you have any other questions. 😊

User

IS there no symbolic computation library in C or C++ - Can you search and tell me?

Bing

There are a few libraries in C++ that can perform symbolic computation, but they may not be as comprehensive or user-friendly as SymPy in Python. Here are some of them:

1. **GiNaC**: An open framework for symbolic computation within the C++ programming language[**2**](https://stackoverflow.com/questions/11325514/symbolic-computation-in-c). However, it's noted in a StackOverflow discussion that GiNaC might not be able to integrate all types of functions[**1**](https://www.ginac.de/tutorial/).
2. **SymbolicC++**: Mentioned in the same StackOverflow discussion, it can integrate functions like sine, cosine, or exp (x) and ln (x), but it is not very powerful[**1**](https://www.ginac.de/tutorial/).
3. **Ev3**: A library for symbolic computation in C++ using n-ary Trees[**4**](https://www.lix.polytechnique.fr/~liberti/Ev3.pdf)[**5**](https://research.ijcaonline.org/volume46/number3/pxc3879194.pdf). It's designed for performing symbolic computation, particularly the calculation of symbolic derivatives and various expression simplification[**4**](https://www.lix.polytechnique.fr/~liberti/Ev3.pdf)[**5**](https://research.ijcaonline.org/volume46/number3/pxc3879194.pdf).
4. **SymEngine and Piranha**: These are other C++ libraries mentioned in the StackOverflow discussion that do symbolic computation[**1**](https://www.ginac.de/tutorial/). However, SymEngine does not yet support integration, but Piranha does[**1**](https://www.ginac.de/tutorial/).

Please note that while these libraries can perform some symbolic computations, they might not be as extensive or easy to use as SymPy in Python. Also, the capabilities of these libraries can vary, and they might not support all the features you need, such as indefinite integration of all types of functions. It's best to check the documentation of these libraries for detailed information about their capabilities and limitations. I hope this helps! Let me know if you have any other questions. 😊

Source(s)

1. [GiNaC, an open framework for symbolic computation within the C++ ...](https://www.ginac.de/tutorial/)

2. [symbolic computation in C++ - Stack Overflow](https://stackoverflow.com/questions/11325514/symbolic-computation-in-c)

3. [Ev3: A Library for Symbolic Computation in C++ using n-ary Trees](https://www.academia.edu/67948767/Ev3_A_Library_for_Symbolic_Computation_in_C_using_n_ary_Trees)

4. [Ev3: A Library for Symbolic Computation in C++ using n-ary Trees](https://www.lix.polytechnique.fr/~liberti/Ev3.pdf)

5. [Applications of Symbolic computation in C++ Programming Language](https://research.ijcaonline.org/volume46/number3/pxc3879194.pdf)

6. <https://github.com/symengine/symengine%29>

7. <https://github.com/bluescarni/piranha%29>