Symbolic Differentiation of Complex Expressions

May 1, 2019

Akash Gaonkar
akash.gaonkar@colorado.edu
Undergraduate Student, University of Colorado at Boulder

Valliappan Chidambaram
vach7169@colorado.edu
Undergraduate Student, University of Colorado at Boulder

Abstract

When working with large or complicated expressions, it can often be challenging to take the derivative by hand. We can compute a numerical approximation of the derivative at a certain point, but this loses the flexibility that an expression provides. This issue is solved in the real case by taking symbolic derivatives, which leave variables unsubstituted and produce a new expression representing the derivative. While similar techniques can be applied to take the derivative of complex functions, it is far more difficult because the derivative only exists where the expression is analytic. We consider a certain class of elementary functions and compositions of these functions. For any such expression, we give an algorithm that attempts to determine where the function is analytic and produces its symbolic derivative. We implement this algorithm and evaluate it over several examples, and compare it to existing methods of complex differentiation.

1 Introduction

A complex function is analytic on some region R if it has a Taylor series that converges to it pointwise on that region. Functions are differentiable on a region R if and only if they are analytic on that region. This means that to take the derivative of a function, you must first know if and where it is analytic. We don't allow expressions that contain functions that are analytic nowhere, like $\overline{z}$, $Re(z)$, and $Im(z)$, so we only need to find singular points of functions that are mostly analytic. Finding these singular points enables various useful things. For example, if you could classify singular points as isolated, branch, and cluster points, and you could calculate the residues at the isolated singular points, then it would be possible to use the Cauchy-Residue theorem and indented contours to calculate the integral of any closed loop in the complex plane on the allowed functions.

Finding the singular points of a symbolic function is difficult because it is difficult to calculate a Taylor series and test its convergence on a symbolic function. Even if that were possible, it would be difficult to find the singular points, the set of points on which a pointwise convergent Taylor series couldn't be found. It is possible to find the singularities of a function in different ways, using our knowledge of the singularities of simpler expressions, like our method described in Section 3. Other computer algebra systems, such as Maple and WolframAlpha use similar methods to calculate singular points in the functions they are given. Maple and WolframAlpha can both find all isolated singular points and branch points of arbitrary functions, although WolframAlpha reports singularities in terms of the inputs to functions (for example, the singularities for $sin(1/z)$ are reported in the $1/z$-plane) and very often fails. Our method allows the user to find all isolated singular points, branch points, and cluster points of functions that we can simplify correctly. When we are unable to solve for the roots of certain equations, we report the answer similar to WolframAlpha. Unlike Maple and WolframAlpha, we are unable to classify our singular points into poles and essential singular points, and we are unable to perform additional computations such as calculating residues at singular points. We are also unable to determine if infinity is a singular point, or the type of singular point at infinity.

2 Expressions

$$
\begin{aligned}
\langle Expr \rangle &::= z \mid \mathbb{C} \mid \sin(\langle Expr \rangle) \mid \cos(\langle Expr \rangle) \mid \exp(\langle Expr \rangle) \\
&\quad \mid \log(\langle Expr \rangle) \mid \langle Sum \rangle \mid \langle Term \rangle \\
\langle Sum \rangle &::= \langle Expr \rangle \times \mathbb{C} \mid \langle Expr \rangle \times \mathbb{C} + \langle Sum \rangle \\
\langle Term \rangle &::= \langle Expr \rangle^{\mathbb{C}} \mid \langle Expr \rangle^{\mathbb{C}} \times \langle Term \rangle
\end{aligned}
$$

Figure 1. The grammar for the allowed expressions

Our system allows the expressions defined by **Expr** in Figure 1. More specifically, we allow the complex variable z, complex numbers, sums, products, powers, sin, cos, log, exp, and compositions of these functions. Each of these expressions are analytic except for a countable number of singular points. The goal of our algorithm is to produce a list of the isolated singular points, branch points, and cluster points. For example, the function $1/log(z-1)$ has an isolated singular point at $z = 2$ and a branch point at $z = 1$. This is because expressions have isolated singular points where their denominators are 0, and $log(z-1) = 0$ when $z = 2$. Expressions also have branch points where the input to a log or non-integer power is zero or infinity, and $z - 1 = 0$ when $z = 1$, so that is a branch point. An example with a cluster point would be $1/cos(1/z)$. Its singularities can be found as follows:

$$
cos\left(\frac{1}{z}\right) = 0 \implies \frac{1}{z} = \frac{\pi}{2} + n\pi \implies z = \frac{1}{\frac{\pi}{2} + n\pi},\ n \in \mathbb{Z}.
$$

Because $lim_{n\to\infty} z = 0$, it means that there are an infinite number of singular points about $z = 0$, so it's a cluster point. It is also possible for there to be infinitely many cluster points. For example, the singular points of $1/sin(1/sin(z))$ can be found as follows:

$$sin\left(\frac{1}{sin(z)}\right) = 0 \implies \frac{1}{sin(z)} = n\pi \implies sin(z) = \frac{1}{n\pi}$$
$$\implies z = sin^{-1}\left(\frac{1}{n\pi}\right),\ n \in \mathbb{Z}.$$

Because $lim_{n\to\infty} z = sin^{-1}(0) = m\pi,\ m \in \mathbb{Z}$, there are an infinite number of cluster points. We have shown how to find the singularities of specific expressions, we will now show how to find the singularities for arbitrary expressions.

3 Analyticity of Compositions

- Singularities -> Rootfinding
- Branch Points -> Rootfinding + Singularities
- Cluster Points -> Evaluation + Singularities + Branch Points

4 Symbolic Root Finding

Our expressions are represented using something known as an abstract syntax tree (AST). ASTs are useful for representing things that can be written as grammars, such as the expressions we allow.

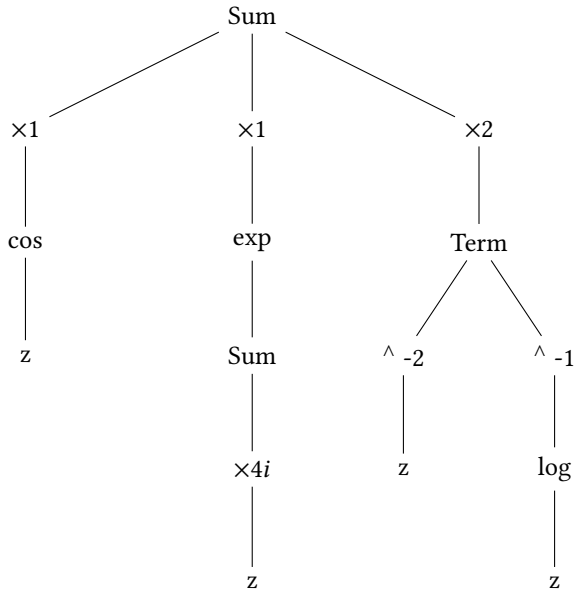

Figure 2. Example AST for $cos(z) + e^{4iz} + 2/(z^2 log(z))$. Note that Sum takes the sum of all of its children, and that Term takes the product of all of its children. Also note that multiplication by a constant takes place in Sums and not Terms.

As you can see in Figure 2, ASTs can show expressions in a surprisingly intuitive way. To find roots on an AST, the first thing we do is simplify it. We perform simplification on ASTs using rules. These rules say that if a subtree of the AST has some property, then part of it can be replaced with a simplified AST. Some of the rules we use are constant evaluation, removal of things with a coefficient of zero in a Sum, and removal of things with a power of one in a Term. We can't use all of the rules from the real numbers. For example, we allow the simplification $e^{log(z)} \to z$, but not $log(e^z) \to z$, because log is a multivalued function (we could simplify $log(e^z) \to z + 2n\pi i$, but this rule was never implemented). We don't have any simplification rules that implement factoring, so we can't successfully simplify the expression $z/(z^2 - z) \to 1/(z-1)$, so our algorithms produce incorrect results on such expressions.

We implemented a routine that tried to find where two expressions were equal. To find roots, we ran this routine to find where an expression equaled zero. Our solving routine worked by trying to solve the outermost expression on the right hand side, and calling itself recursively on the input to the outermost expression. For example, when trying to solve $sin(1/z) = 0$, we find that $sin(z) = 0$ when $z = n\pi$, and then we try to solve $1/z = n\pi$. This doesn't work when trying to solve Sums of more than one expression, or when trying to solve Terms for anything other than zero. We were unable to come up with an algorithm for doing this, so our algorithm simply reports the equations it was unable to solve. WolframAlpha and Maple have similar problems, but they are able to solve more equations with sums and products, and often report numerical solutions when they are unable to solve.

5 Implementation

6 Future Work