Multilinear Polynomial Calculator Proposal

Caiwu Chen, Khaliun Gerel

0. Team Member

This project team will be composed of Caiwu Chen and Khaliun Gerel. Since we are a two-student team, we will both take on roles including manager, language and compiler designer, and tester. We plan to meet every Friday afternoon from 2:00 PM to 5:00 PM, with each meeting lasting one hour. The duration and timing of the meetings are subject to change based on project needs and availability.

1. Introduction

The goal of this project is to design a compiler which is able to solve multilinear polynomial equations given the value of variates. For basic cases, quadratic equations or cubic functions for example, we will apply formulas. However, general multivariate polynomial equations usually don't have general equations. In that case, given the value of certain variates, we will investigate Newton's method and gradient descent to find an approximate value under tolerance. If the equations have multiple solutions, it will return all possible variates.

2. Motivation

Considering the importance of multivariate polynomial i.e. multilinear polynomial in Mathematical and Theoretical Analysis, and Optimization and Machine Learning, we are designing a multivariate polynomial calculator to help researchers and students in studying the features of polynomial. Meanwhile, as the designers of that language, we aim to investigate the application of Newton's method or gradient descent in finding the solutions. This language will be implemented by Python and its related packages.

3. Creativity/Novelty

The novelty of the Multilinear Polynomial Calculator lies in its focus on efficiently solving nonlinear multivariate polynomial equations using well-established numerical methods like Newton's method. While the concept itself isn't entirely new or overly complex, the design of a specialized calculator tailored for such equations can be valuable for specific use cases, such as scientific computations or symbolic algebra systems. Its simplicity and practicality make it suitable for applications in education and research, providing a focused tool for students and professionals dealing with complex polynomial equations across multiple variables.

4. Language and Compiler small description

The language will be designed to parse and evaluate multilinear polynomial equations, identifying input variables and computing results based on provided values. It supports basic arithmetic operators like addition, subtraction, multiplication, and exponentiation, allowing users to input complex equations with multiple variables. Parentheses are used to enforce operator precedence, ensuring correct evaluation of nested expressions. By parsing input equations and applying methods like Newton's method for solving nonlinear equations, the language can efficiently compute solutions for variable assignments. This makes it suitable for handling complex polynomial expressions while offering simple syntax and error handling for invalid inputs.

5. **Example Simple Program Written in your Language**

```
GIVEN EQUATION:  f(x,y): 2 * (x^2) + 3 * (y^3)=1, x=1
SAY RESULT : 0.7

Get f(x,y)'
Guess y

While y > tolerance: {
        this_y = f(x,y)
        prime_y = f(x,y)' (this_y)
        y_new = y - (this_y / prime_y)
        update y
}

Return y
```