Simplification and / or Derivation of Mathematical Expressions in RPN

Introduction

We currently have a package (C++) to parse, validate and tokenize a mathematical infix expression with numbers, free variables, operators, build-in constants, mathematical functions and user defined functions. The tokenized infix expression is then converted into a Reverse Polish Notation (RPN) expression based on Shunting-yard algorithm (1961). With RPN expression we can then conveniently evaluate the expression with the required parameters, which correspond to the free variables. The current implementation can recognize constant sub-expressions and replace them with their actual value. Furthermore, we replace sub-expressions multiplied with zero by a zero.

(a+2)*(b+3) → a, 2, +, b, 3, +, *

Problem 1 – Simplification

The current implementation is limited to numbers and lacks the general symbolic simplification of sub-expression, e.g. x-x → 0 or cos(x)-cos(-x) → 0.

Mentor: Thierry Matthey

Problem 2 – Symbolic Derivation

Furthermore, at moment we are missing a symbolic derivation of the RPN expression and use numerical derivation, e.g., x^2 → 2x.

Mentor: Thierry Matthey

Goal

The goal of the project is to find and evaluate potential algorithms to simplify and derivation of RPN expression. Then picking the adequate algorithms and implement them Test-Driven-Development (TDD) them into the current code base in C++. TTD helps not only improve code quality, but we can also measure also the performance or accuracy of the implantation.

This project involves some important challenge in the algorithmic part as much as in the programming part.

Advantages

- TDD
- Using Visual Studio, ReShaprer and C++
- Contributing new and needed functionality to the dynamic multiphase flow simulator OLGA, which is de facto-standard in the oil & gas industry
- If the solution is innovative enough, a patent will be filled.