High Performance Computer Architectures Practical Course - Exercise 4 -

Tutorium 1

David Jordan (6260776)
Florian Rüffer (7454628)
Michael Samjatin (7485765)

May 14, 2023

1 Matrix

2 Quadratic Equation

3 Newton

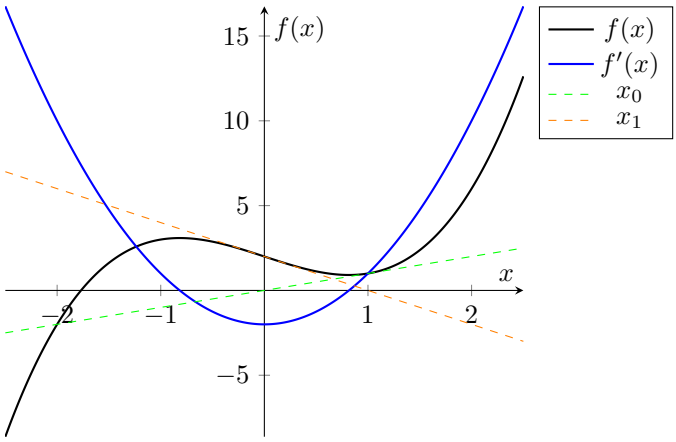
In this task we have to vectorize the Newton method. As mentioned in the task description the Newton Method is an interative process to approximate the x-axis intersection.

```
1   float_v FindRootVector(const float_v& p1, const
        float_v& p2)
2   {
3     float_v x = 1.f, x_new = 0.f;
4     float_m mask(true);
5     for( ; !mask.isEmpty(); ) {
6         // for(int i = 0; i < 1000; ++i){
7       x = x_new;
8       x_new(mask) = x - F(x,p1,p2) / Fd(x,p1,p2);
9       mask = abs((x_new - x)/x_new) > P;
10    }
11    return x_new;
12  }
```

File 1: Newton.cpp

The main idea of vectorizing the Newton method, is the calculation of multiple roots at once. This comes along with an issue, because if multiple roots are calculate, then also some of those will reach the desired precision earlier than others. To solve this problem we will use masks. The underlying principle of masks is a vector with each value mapped to a boolean. Each boolean value corresponds to the value of another vector. Using this methodology we can assign 'false' to a corresponding number, which already has our desired precision and therefore exclude it from further unneccessary calculations.

The first iteration will start with a mask of only 'true' values, as no value has yet reached the desired precision. This means every value will be updated accordingly with $x - F(x, p1, p2)/Fd(x, p1, p2)$. The next iteration will only update those values, which need further refinement ($\rightarrow$ mask value equals 'true') with $mask = abs((x_new - x)/x_new) > P$.



4 Random Access

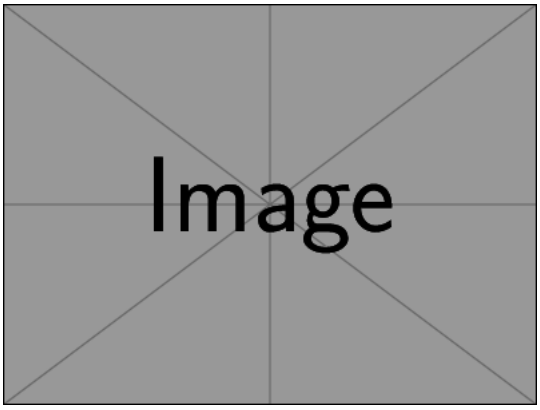


Figure 1: Output