Complex numbers and templating

Victor Eijkhout, Susan Lindsey

Fall 2022
last formatted: April 13, 2023

Complex numbers

Complex Newton

Templated functions

You can templatize your Newton function and derivative:

```cpp
template<typename T>
T f(T x) { return x*x - 2; };
template<typename T>
T fprime(T x) { return 2 * x; };
```

and then write

```cpp
double x{1.};
while ( true ) {
  auto fx = f<double>(x);
  cout << "f( " << x << " ) = " << fx << '\n';
  if (std::abs(fx)<1.e-10 ) break;
  x = x - fx/fprime<double>(x);
}
```

Exercise 1

Update your Newton program with templates. If you have it
working for `double`, try using *complex*<`double`>. What goes wrong?

First of all, you can not operate immediately between a complex
number and an integer. Use static_cast; section **??**.

Exercise 2

Use your complex Newton method to compute $\sqrt{2}$. Does it work?

Exercise 3

Can you templatize your Newton code that used lambda expressions? Your function header would now be:

```
template<typename T>
T newton_root
   ( function< T(T) > f,
     function< T(T) > fprime,
     T init) {
```

You would for instance compute $\sqrt{2}$ as:

```
cout << "sqrt -2 = " <<
    newton_root<complex<double>>
        ( [] (complex<double> x) {
              return x*x + static_cast<complex<double>>(2); },
          [] (complex<double> x) {
            return x * static_cast<complex<double>>(2); },
          complex<double>{.1,.1}
          )
    << '\n';
```