CS 4424
Newton iteration
eschost@uwo.ca
What this is about

Newton iteration is a way to compute approximate solutions to various problems.

It is classically defined in analysis: to compute a root of $P(x)$, we use the iteration

$$x_0 = \text{random}, \quad x_{i+1} = x_i - \frac{P(x_i)}{P'(x_i)}.$$

Here, we will use it for power series computations.
Approximations

There are strong analogies between real numbers

\[ a = 0.93493847630496 \cdots = \sum_{i \geq 1} a_i \frac{1}{10^i} \]

and power series

\[ S = \sum_{i \geq 0} s_i x^i. \]

- both are infinite expansions;
- computationally, we are interested in computing truncations at finite precision;
- similar techniques apply.

Remark: series are easier to handle than real numbers, because there is no carry.
Newton iteration

The example we saw can be vastly generalized and applied to series computations.

In a nutshell:

- applies to compute exponential, inverses, logarithms, square roots, ... of series,
- more generally, solutions of polynomial or differential equations.

Main feature: efficiency!

- typical behaviour: the number of correct terms doubles each step;
- combined with polynomial multiplication $\implies$ quasi-optimal.
Multiplication
Reminder: polynomial multiplication

Let $M(d)$ denote the cost of polynomial multiplication in degree $d$:

- $M(d) \in O(d^2)$ for a naive algorithm
- $M(d) \in O(d^{1.6})$ for Karatsuba algorithm
- $M(d) \in O(d \log d)$ using Fast Fourier Transform (if the field has roots of 1)
- $M(d) \in O(d \log d \log \log d)$ using Fast Fourier Transform in general.

Technically, we ask $M(d + d') \geq M(d) + M(d')$. 
A few rules for estimating complexity

We know that

\[ 1 + 2 + 4 + \cdots + 2^{n-1} = 2^n - 1 \leq 2^n. \]

We have similar estimates for polynomial multiplication:

\[ M(1) + M(2) + M(4) + \cdots + M(2^{n-1}) \leq M(2^n). \]

Proof.

\[ M(a) + M(b) \leq M(a + b) \implies 2M(a) \leq M(2a) \]
\[ \implies 2^k M(a) \leq M(2^k a) \]
\[ \implies 2^k M(2^n / 2^k) \leq M(2^n) \]
\[ \implies M(2^n / 2^k) \leq 2^{-k} M(2^n) \]

Corollary. If \( T(2n) \leq T(n) + CM(n) \), then \( T(n) \in O(M(n)) \).
Inversion
Iteration for the inverse

Given a series
\[ f = f_0 + f_1 x + f_2 x^2 + \cdots, \quad f_0 \neq 0 \]
we want to compute the coefficients of
\[ g = g_0 + g_1 x + g_2 x^2 + \cdots \]
such that \( fg = 1 \).

Basic idea:

- compute one term after the other, by identification.
- **slow**: \( O(n^2) \) operations for \( n \) terms.
- at least, this proves that \( g \) exists and is unique.
Newton’s iteration

Idea: $g$ is a root of $P(g) = 0$, with

$$P(g) = \frac{1}{g} - f.$$

With this $P$, the Newton iteration becomes

$$h_0 = 1/f_0, \quad h_{(i+1)} = 2h_i - h_i^2 f \text{ rem } x^{2^{i+1}}.$$

So we consider the operation $h \mapsto N(h) = 2h - h^2 f$.

- Suppose $h = g \text{ rem } x^k$.
- Multiplying by $f$, we get $hf \text{ rem } x^k = 1$. This means $hf = 1 + x^k R$.
- Then, $N(h)f = 2hf - h^2 f^2 = 1 - (hf - 1)^2 = 1 - x^{2k} R^2$.
- So $N(h) = g \text{ rem } x^{2k}$. 
Cost analysis

The previous argument shows that

\[ h(i) = g \text{ rem } x^{2^i}. \]

Cost of a single step:

- To get \( h(i+1) \) from \( h(i) \), we compute \( 2h(i) - h_i^2f \mod x^{2^{i+1}} \).
- This costs \( O(M(2^{i+1})) \).

Cumulated cost:

- To get \( h(i) \) from \( h(0) = 1 \), the cost is

\[ O(M(2) + M(4) + \cdots + M(2^i)) = O(M(2^i)). \]

- In other words: to get \( 1/f \mod x^n \), the cost is \( O(M(n)) \).
Algebraic series
Roots of polynomial

Def.

- A series

\[ f = \sum_{i \geq 0} f_i x^i \]

is algebraic if there exists a polynomial

\[ P(x, z) \] such that \[ P(x, f) = 0. \]

Examples.

- rational series \( f(x) = n(x)/d(x) \)

- \( \sqrt{1 + x} = 1 + \frac{1}{2} x - \frac{1}{8} x^2 + \frac{1}{16} x^3 - \frac{5}{128} x^4 + \frac{7}{256} x^5 + \cdots \)

- \( \exp(x) = 1 + x + \frac{1}{2} x^2 + \frac{1}{6} x^3 + \frac{1}{24} x^4 + \frac{1}{120} x^5 + \cdots \) is not algebraic.
Examples from combinatorics

A lot of sequences arising from enumeration problems satisfy nice properties, like having an algebraic generating series.

Example: Catalan numbers.

Let $C_n$ be the number of binary trees with $n$ nodes.

$C_0 = 1, \ C_1 = 1, \ C_2 = 2, \ C_3 = 5, \ldots$
Recurrence relation

To build a tree with $n$ nodes, you

- set the root (so you have $n - 1$ nodes left)
- put $p$ nodes on the left
- and $n - 1 - p$ nodes on the right.

This gives

$$C_n = \sum_{p=0}^{n-1} C_p C_{n-1-p}$$

for $n \geq 1$. 
The generating series

Let \( f = \sum_{i \geq 0} C_i x^i \). Then,

\[
\sum_{p=0}^{n-1} C_p C_{n-1-p}
\]

is the coefficient of \( x^{n-1} \) in \( f^2 \).

Multiplying the recurrence relation by \( x^n \) and summing for \( n \geq 1 \), we get

\[
f - 1 = x f^2.
\]

So \( f \) is algebraic, with

\[
P(x, z) = x z^2 - z + 1
\]
Extracting the coefficients

In this case, we have an explicit formula

\[ f = \frac{1 - \sqrt{1 - 4x}}{2x}, \]

from which one can deduce

\[ C_n = \frac{1}{n+1} \binom{2n}{n}. \]

In general, though, there are no closed formula.
Let $f(x) = \sqrt{1 + x}$. The power series expansion of $f$ gives successive approximations at $x = 0$. 

\[ 
\begin{align*} 
\pm 2 & \quad \pm 1 \\
\pm 1 & \quad 1 & \quad 2 & \quad 3 & \quad 4 \\
\end{align*} 
\]
Let \( f(x) = \sqrt{1 + x} \). The power series expansion of \( f \) gives successive approximations at \( x = 0 \).
Let \( f(x) = \sqrt{1+x} \). The power series expansion of \( f \) gives successive approximations at \( x = 0 \).

\[
f \text{ rem } x^2 = 1 + \frac{1}{2}x
\]
Link to geometry

Let \( f(x) = \sqrt{1 + x} \). The power series expansion of \( f \) gives successive approximations at \( x = 0 \).

\[
f \text{ rem } x^3 = 1 + \frac{1}{2}x - \frac{1}{8}x^2
\]
Computing the expansion

We will compute the expansion of $f$ such that

$$P(x, f) = 0$$

subject to the following:

- the constant term $f_0$ of $f$ is known
  - we need it to start the process

- the partial derivative $\frac{\partial P}{\partial z}(0, f_0)$ is not zero.
  - at the starting point, the tangent to the curve exists, and is not vertical
The slow algorithm

Suppose that we know the first terms

\[ f_{\text{init}} = f_0 + \cdots + f_{n-1}x^{n-1}, \]

such that

\[ P(x, f_{\text{init}}) \text{ rem } x^n = 0. \]

Basic step

- we look for a single extra term \( f_nx^n \), to get

\[ f_{\text{next}} = f_0 + \cdots + f_{n-1}x^{n-1} + f_nx^n, \]

such that

\[ P(x, f_{\text{next}}) \text{ rem } x^{n+1} = 0. \]

- we get it by identification.
Getting the next term

For any \( k \), we have

\[
(f_0 + \cdots + f_{n-1}x^{n-1} + f_nx^n)^k = \text{known stuff in } f_0, \ldots, f_{n-1} + k f_0^{k-1} f_n x^n + \text{higher order terms.}
\]

If we write

\[
P(x, z) = p_d(x)z^d + \cdots + p_1(x)z + p_0(x),
\]

then the coefficient of \( x^n \) in \( P(x, f_{\text{next}}) \) is

\[
\text{known stuff } + (dp_d(0)f_0^{d-1} + \cdots + p_1(0)f_0)f_n = \text{known stuff } + \frac{\partial P}{\partial z}(0, f_0)f_n.
\]

So we can solve for \( f_n \).
Newton iteration

Computing the $n$th term requires at least $n$ operations, whence a cumulated cost of at least $n^2$ for $f_1, \ldots, f_n$ (disregarding the dependency in $d$).

Newton iteration:

$$f_{(0)} = f_0 \quad f_{(i+1)} = f_{(i)} - \frac{P(x, f_{(i)})}{\partial P/\partial z(x, f_{(i)})} \text{ rem } x^{2i+1}.$$ 

Prop.

- this correctly computes the expansion of $f$;
- the cost is $O(dM(n))$ for order $n$. 
Warm-up

The slow construction showed that given the initial condition $f_0$, $f$ exists and is unique.

Prop.

- For a polynomial $g$ of degree $< k$, equivalence between

  $$P(x, g) \text{ rem } x^k = 0 \text{ and } g = f \text{ rem } x^k$$

Proof.

- If $g = f \text{ rem } x^k$ then $P(x, g) \text{ rem } x^k = P(x, f) \text{ rem } x^k = 0$.
- Converse by induction, using the explicit construction.
Why it works

Taylor formula

• For any \( h, g \), we have

\[
P(x, h + g) = P(x, h) + \frac{\partial P}{\partial z}(x, h)g + g^2 R.
\]

Application

• \( f_{(i)} = f \text{ rem } x^{2^i} \implies P(x, f_{(i)}) \text{ rem } x^{2^i} = 0 \).

• take

\[
h = f_{(i)} \quad \text{and} \quad g = -\frac{P(x, f_{(i)})}{\partial P/\partial z(x, f_{(i)})} \text{ rem } x^{2^{i+1}}.
\]

• so \( h + g = f_{(i+1)} \)

• remark

\[
g \text{ rem } x^{2^i} = 0 \quad \text{so} \quad g^2 \text{ rem } x^{2^{i+1}} = 0.
\]

• so \( P(x, f_{(i+1)}) \text{ rem } x^{2^{i+1}} = 0 \implies f_{(i+1)} = f \text{ rem } x^{2^{i+1}}.\)