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 multiplications:

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

**Proof.** \( M(a) + M(b) \leq M(a + b) \implies 2M(a) \leq M(2a) \)
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 multiplications:

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

**Proof.** \( M(a) + M(b) \leq M(a + b) \implies 2^k M(a) \leq M(2^k a) \)
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 multiplications:

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

Proof. \( M(a) + M(b) \leq M(a + b) \implies 2^k M(2^n/2^k) \leq M(2^n) \)
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 multiplications:

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

Proof. $M(a) + M(b) \leq M(a + b) \implies M(2^n / 2^k) \leq 2^{-k}M(2^n)$
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 multiplications:

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

**Proof.** \( \text{M}(a) + \text{M}(b) \leq \text{M}(a + b) \implies \text{M}(2^n / 2^k) \leq 2^{-k} \text{M}(2^n) \)

**Corollary.** If \( T(2n) \leq T(n) + CM(n) \), then \( T(n) \in O(M(n)) \).

Similarly, if \( T(2n) \leq 2T(n) + CM(n) \), then \( T(n) \in O(M(n) \log(n)) \)
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.
Enter Newton

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

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

With this $P$, the Newton iteration becomes

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

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

- Suppose $h = g \mod x^k$.
- Multiplying by $f$, we get $hf = 1 \mod x^k$. 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 $N(h) = g \mod x^{2k}$. 
Cost analysis

The previous argument shows that

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

Cost of a single step:

- To get \( h(i+1) \) from \( h(i) \), we compute \( 2h(i) - h^2(i) f \mod x^{2^{i+1}} \).
- 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 \( f(x) = n(x)/d(x) \)

• \( \sqrt{1 + x} = 1 + \frac{1}{2} x - \frac{1}{8} x^2 + \frac{1}{16} x^3 - \frac{5}{128} x^4 + \frac{7}{256} x^5 - \frac{21}{1024} x^6 + \frac{33}{2048} x^7 + \cdots \)

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

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

\[
f = 1 + \frac{1}{2}x \mod x^2
\]
Let $f(x) = \sqrt{1+x}$. The power series expansion of $f$ gives successive approximations at $x = 0$.

\[ f = 1 + \frac{1}{2} x - \frac{1}{8} x^2 \mod x^3 \]
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

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

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

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

• we get it by identification.
Getting the next term

For any $k$, we have

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

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

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

So we can solve for $f_n$. 
Enter Newton

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:**

\[
    f(0) = f_0 \quad f(i+1) = f(i) - \frac{P(x, f(i))}{\frac{\partial P}{\partial z}(x, f(i))} \mod x^{2i+1}.
\]

**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.

• Equivalence between

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

Proof.

• If $g = f \mod x^k$ then $P(x, g) = P(x, f) \mod x^k = 0 \mod x^k$.
• Converse by induction, using the explicit construction.
Why it works

Taylor formula

- For any $h, g$, we have

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

Application

- Suppose $f_{(i)} = f \mod x^{2i} \implies P(x, f_{(i)}) = 0 \mod x^{2i}$.
- Take $h = f_{(i)}$ and $g = -\frac{P(x, f_{(i)})}{\frac{\partial P}{\partial z}(x, f_{(i)})} \mod x^{2^{i+1}}$.
- So $h + g = f_{(i+1)}$
- Remark

\[ g = 0 \mod x^{2^i} \quad \text{so} \quad g^2 = 0 \mod x^{2^{i+1}}. \]
- So $P(x, f_{(i+1)}) = 0 \mod x^{2^{i+1}} \implies f_{(i+1)} = f \mod x^{2^{i+1}}$. 
Final word

Extension

- We have considered all along that $P(x, z)$ is a polynomial in $x$ and $z$.
- Actually, everything works similarly if $P$ is polynomial in $z$, with series coefficients in $x$. 