CS 830
Newton iteration
Éric Schost
eschost@uwo.ca
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 $\Rightarrow$ 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 $M(d + d') \geq M(d) + M(d')$. 
Suppose you want to evaluate $F(X) \in \mathbb{C}[X]$ at all $N$-roots of 1

$$1, \exp\frac{2i\pi}{N}, \exp\frac{4i\pi}{N}, \ldots, \exp\frac{2(N-1)i\pi}{N},$$

with $\deg(F) < N$. 
FFT in a nutshell

Suppose you want to evaluate $F(X) \in \mathbb{C}[X]$ at all $N$-roots of 1

$$1, \exp \frac{2i\pi}{N}, \exp \frac{4i\pi}{N}, \ldots, \exp \frac{2(N-1)i\pi}{N},$$

with $\deg(F') < N$.

Write $F = F_{\text{even}}(X^2) + X F_{\text{odd}}(X^2)$. Then

$$F(\exp \frac{2ik\pi}{N}) = F_{\text{even}}(\exp \frac{2ik\pi}{N'}) + \frac{2ik\pi}{N} F_{\text{odd}}(\exp \frac{2ik\pi}{N'}),$$

with $N' = N/2$. 
FFT in a nutshell

Suppose you want to evaluate $F(X) \in \mathbb{C}[X]$ at all $N$-roots of 1

$$1, \exp \frac{2i\pi}{N}, \exp \frac{4i\pi}{N}, \ldots, \exp \frac{2(N-1)i\pi}{N},$$

with $\deg(F) < N$.

Write $F = F_{\text{even}}(X^2) + XF_{\text{odd}}(X^2)$. Then

$$F(\exp \frac{2ik\pi}{N}) = F_{\text{even}}(\exp \frac{2ik\pi}{N'}) + \frac{2ik\pi}{N}F_{\text{odd}}(\exp \frac{2ik\pi}{N'}),$$

with $N' = N/2$.

So it suffices to

- $F_{\text{even}}$ at all $N'$-roots of 1;
- $F_{\text{odd}}$ at all $N'$-roots of 1;
- combine the values.
FFT in a nutshell

Suppose you want to evaluate $F(X) \in \mathbb{C}[X]$ at all $N$-roots of 1

$$1, \exp \frac{2i\pi}{N}, \exp \frac{4i\pi}{N}, \ldots, \exp \frac{2(N-1)i\pi}{N},$$

with $\deg(F) < N$.

Write $F = F_{\text{even}}(X^2) + XF_{\text{odd}}(X^2)$. Then

$$F(\exp \frac{2ik\pi}{N}) = F_{\text{even}}(\exp \frac{2ik\pi}{N'}) + \frac{2ik\pi}{N} F_{\text{odd}}(\exp \frac{2ik\pi}{N'})$$

with $N' = N/2$.

So it suffices to

- $F_{\text{even}}$ at all $N'$-roots of 1;
- $F_{\text{odd}}$ at all $N'$-roots of 1;
- combine the values.

The complexity satisfies $T(N) \leq 2T(N/2) + CN$ so $T(N) \in O(N \log N)$. 
**FFT in a nutshell**

**Proposition**  The inverse FFT can be performed for the same cost as the direct FFT.

**Corollary**  One can multiply $F(X), G(X) \in \mathbb{C}[X]$, both of them having degree $< N$, in $O(N \log N)$ operations

- Evaluate $F$ and $G$ at $2N$-th roots of 1
- Multiply the values
- Do inverse-FFT to interpolate the product $FG$.

**Extension**  to any field having “roots of unity”.

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

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

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

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

**Similarly,** if \( T(2n) \leq 2T(n) + C M(n) \), 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 \( 2h(i) - h(i)^2 f \mod x^{2^{i+1}} \).
- 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 \( O(M(n)) \).
Application: Euclidean division

We want to compute the quotient and remainder

\[ A = BQ + R, \]

with \( \deg(A) = 2n \) and \( \deg(B) = n \).

Prop.

- The cost is \( O(M(n)) \).

How?

- We reduce this to power series inversion.
- We get \( Q \) first.
- Once \( Q \) is known, we get \( R = A - BQ \).
Rewrite the equality as
\[
\frac{A}{B} = Q + \frac{R}{B}.
\]

**Idea:** let \( x \to \infty \).

- Remember \( \deg(R) < \deg(B) \).
- So \( R/B \to 0 \).
- So the expansion of \( A/B \) at \( \infty \) gives \( Q \).

Formally, replace \( x \) by \( 1/y \), giving
\[
\frac{A(1/y)}{B(1/y)} = Q(1/y) + \frac{R(1/y)}{B(1/y)}.
\]
More details

From $A = a_0 + a_1 x + \cdots + a_{2n} x^{2n}$, we get

$$A = a_0 + \frac{a_1}{y} + \cdots + \frac{a_{2n}}{y^{2n}}.$$

Doing the same with the others, we get

$$\frac{a_0 + \frac{a_1}{y} + \cdots + \frac{a_{2n}}{y^{2n}}}{b_0 + \frac{b_1}{y} + \cdots + \frac{b_n}{y^n}} = \frac{q_0 + \frac{q_1}{y} + \cdots + \frac{q_n}{y^n}}{b_0 + \frac{b_1}{y} + \cdots + \frac{b_n}{y^n}} + \frac{r_0 + \frac{r_1}{y} + \cdots + \frac{r_{n-1}}{y^{n-1}}}{b_0 + \frac{b_1}{y} + \cdots + \frac{b_n}{y^n}}.$$
Multiply by \(y^n\).

1. On the left, we get
\[
\frac{a_{2n} + a_{2n-1}y + \cdots + a_0y^{2n}}{b_n + b_{n-1}y + \cdots + b_0y^n}
\]

2. On the right, we get
\[
q_n + q_{n-1}y + \cdots + q_0y^n \quad + \quad y^{n+1} \frac{r_{n-1} + r_{n-2}y + \cdots + r_0y^{n-1}}{b_n + b_{n-1}y + \cdots + b_0y^n}.
\]

So
\[
q_n + q_{n-1}y + \cdots + q_0y^n = \frac{a_{2n} + a_{2n-1}y + \cdots + a_0y^{2n}}{b_n + b_{n-1}y + \cdots + b_0y^n} \mod y^{n+1}.
\]

So we can get it in \(O(M(n))\).
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

\[
P(x, z) = xz^2 - z + 1
\]
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 \( x = 0 \).

\[
f = 1 \text{ mod } x
\]
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 \( f_n x^n \), to get

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

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

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

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 $P(x, f_{\text{new}})$ 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 $f_n$. 
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:

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

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

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

Application

• suppose $f(i) = f \mod x^{2^i} \iff P(x, f(i)) = 0 \mod x^{2^i}$.

• take

$$h = f(i) \quad \text{and} \quad g = -\frac{P(x, f(i))}{\partial P/\partial z(x, f(i))} \mod x^{2^{i+1}}.$$

• remark

$$g = 0 \mod x^{2^i} \quad \text{so} \quad g = 0 \mod x^{2^{i+1}}.$$

• so $P(x, f(i+1)) = 0 \mod x^{2^{i+1}} \iff f(i+1) = f \mod x^{2^{i+1}}$. 
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 $x$. 
Exponential and logarithm
The exponential of a series

Def.

- Let

\[ f = f_1 x + f_2 x^2 + \cdots \]

remark that there is no constant coefficient.

- Its exponential is

\[
\exp(f) = \sum_{i \geq 0} \frac{1}{i!} f^i = 1 + f + \frac{1}{2} f^2 + \frac{1}{6} f^3 + \cdots .
\]

Remark.

- \( f^i = f_1^i x^i + \cdots \) so \( \exp(f) \) is well-defined.

- Naive: \( O(n^2) \) for \( n \) terms.
The logarithm of a series

Def.

- Let

\[ f = f_1 x + f_2 x^2 + \cdots \]

remark that there is no constant coefficient.

- Its logarithm is

\[
\log(1 + f) = \sum_{i \geq 1} \frac{(-1)^{i+1}}{i} f^i = f - \frac{1}{2} f^2 + \frac{1}{3} f^3 - \cdots .
\]

Remark.

- \( f^i = f_1^i x^i + \cdots \) so \( \log(1 + f) \) is well-defined.

- Naive: \( O(n^2) \) for \( n \) terms.
Computing the logarithm

Differentiation:

\[ \log(1 + f)' = (f - \frac{1}{2}f^2 + \frac{1}{3}f^3 + \cdots)' \]
\[ = f' - ff' + ff'^2 + \cdots \]
\[ = \frac{f'}{1+f} \]

Algorithm

- We know how to compute \( n \) terms of \( f'/(1 + f) \).
  cost: \( O(M(n)) \)
- By integrating, we get \( n \) terms of \( \log(1 + f) \).
  cost: \( n \)
Computing the exponential

Composition

\[ \log(1 + (\exp(f) - 1)) = f. \]

**Proof:** extract the coefficients and check.

**Newton iteration:** to compute \( \exp(f) - 1 \), we solve

\[ P(g) = 0 \quad \text{with} \quad P(g) = \log(1 + g) - f. \]

The iteration is

\[ g(0) = 0, \quad g(i+1) = g(i) - (1 + g(i))(\log(1 + g(i)) - f) \mod x^{2^{i+1}}. \]

- **Cost:** \( O(M(n)) \)
- **Proof:** Taylor expansion for the log:

\[ \log(1 + g + h) = \log(1 + h) + \frac{g}{1 + h} + g^2 R \]
Power sums

Let

\[ F = x^n + f_1 x^{n-1} + \cdots + f_{n-1} x + f_n = (x - x_1) \cdots (x - x_n). \]

Power sums

\[ S_0 = n, \quad S_1 = \sum x_i, \quad S_2 = \sum x_i^2, \ldots \]

Main points

- one can convert between power sums and coefficients easily (without knowing the roots);
- this is useful.
Generating series

Let

\[ G = 1 + f_1 x + \cdots + f_n x^n = (1 - x_1 x) \cdots (1 - x_n x). \]

Prop. 1

\[ G' = -(S_1 + S_2 x + S_3 x^2 + \cdots) G. \]

Corollary 1

- Given \( F \) (and thus \( G \)), one can deduce \( S_0, \ldots, S_n \) in \( O(M(n)) \).

Prop. 2

\[ G = \exp \left( - \int (S_1 + S_2 x + S_3 x^2 + \cdots) \right) \]

Corollary 2

- Given \( S_0, \ldots, S_n \), one can deduce \( F \) (and thus \( G \)) in \( O(M(n)) \).
Proof

1. logarithmic derivative:

\[ \frac{G'}{G} = \sum_{j=1}^{n} - \frac{x_j}{1 - x_j x}. \]

2. geometric series:

\[ - \frac{x_j}{1 - x_j x} = - \sum_{i \geq 0} x_j^{i+1} x^i \]

3. summation:

\[ \sum_{j=1}^{n} \sum_{i \geq 0} x_j^{i+1} x^i = \sum_{i \geq 0} \sum_{j=1}^{n} x_j^{i+1} x^i = - \sum_{i \geq 0} S_{i+1} x^i. \]
You want to compute the characteristic polynomial $\chi$ of a matrix $A$.

Recall:

$$\chi(x) = \prod(x - x_i), \quad x_i \text{ eigenvalues of } A.$$ 

**Leverrier**

- The trace of $A^j$ is $\sum_i x_i^j$.
- So compute the traces and deduce $\chi$.
- Not so useful for general matrices.
- Good for special cases (where you can get the traces easily) or for parallel algorithms.
Linear differential equations
D-finite series

Def.

• A series \( f(x) \) is D-finite if there exists a linear differential equation with polynomial coefficients such that

\[
q_d(x)f^{(d)} + q_{d-1}(x)f^{(d-1)} + \cdots + q_0(x)f = 0.
\]

• Equivalently, we can take rational functions as coefficients.

Examples.

• polynomials,
• rational functions,
• algebraic series (e.g., \( \sqrt{1 + x^2} \))
• exp, sin, cos,
• a lot more
Solving the equation

Problem

• Suppose that $q_d(0) \neq 0$.

• Then given initial conditions $f_0, \ldots, f_{d-1}$, there exists a unique

$$f = f_0 + f_1 x + \cdots + f_{d-1} x^{d-1} + \cdots$$

such that

$$q_d(x)f^{(d)} + q_{d-1}(x)f^{(d-1)} + \cdots + q_0(x)f = 0.$$

Remark

• Here, we can as well suppose that the $q_i(x)$ are series in $x$. 
Derivation formulas

For

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

we get

\[ f' = \sum_{i \geq 1} if_i x^{i-1} \quad \text{and} \quad f'' = \sum_{i \geq 2} i(i - 1)f_i x^{i-2}, \quad \ldots \]

Multiplying by a monomial shifts the coefficients:

\[ x^\ell f' = \sum_{i \geq 1} if_i x^{i+\ell-1} = \sum_{i \geq \ell} (i - \ell + 1)f_{i-\ell+1}x^i, \quad \ldots \]

\[ x^\ell f'' = \sum_{i \geq 2} i(i - 1)f_i x^{i+\ell-2} = \sum_{i \geq \ell} (i - \ell + 1)(i - \ell + 2)f_{i-\ell+2}x^i, \quad \ldots \]
Coefficient extraction

1. The coefficient of $x^i$ in $x^\ell f^{(k)}$ is

\[(i - \ell + 1)(i - \ell + 2) \cdots (i - \ell + k)f_{i-\ell+k}\]

2. The coefficient of $x^i$ in $(\sum_{\ell \geq 0} a_\ell x^\ell) \times f^{(k)}$ is

\[\sum_{\ell=0}^{i+k} a_\ell (i - \ell + 1)(i - \ell + 2) \cdots (i - \ell + k)f_{i-\ell+k},\]

so it depends on $f_0, \ldots, f_{i+k}$. 
Coefficient extraction

3. Suppose you know $f_0, \ldots, f_{i+d-1}$. Then the coefficient of $x^i$ in

$$q_d(x)f^{(d)} + q_{d-1}(x)f^{(d-1)} + \cdots + q_0(x)f$$

is

known stuff + $q_d(0)(i+1)\cdots(i+d)f_{i+d}$.

4. If we want to set this coefficient to 0, this defines $f_{i+d}$ uniquely.

5. The cost is at least $n$ for one step, and at least $n^2$ cumulated.
Newton iteration

For equations of order 1:

\[q_1(x)f' + q_0(x)f = 0, \quad f(0) = f_0 \quad \iff \quad f = f_0 \exp \int -\frac{q_0}{q_1}.
\]

Consequences

- only requires inversion and exponentiation;
- so cost for \(n\) terms is \(O(M(n))\).

Higher order equations: course project.

- involves matrix computations;
- cost for \(n\) terms is \(O(d^\omega M(n))\).
Extra

The same ideas can be used to solve non-homogeneous equations:

\[ q_1(x)f' + q_0(x)f = g, \quad f(0) = f_0 \iff f = f_0 + \int \frac{Jg}{q_1}, \]

with

\[ J = \exp \int \frac{q_0}{q_1}. \]

Consequences

- only requires inversion and exponentiation;
- so cost for \( n \) terms is \( O(M(n)) \).

Remarks

- the algorithm works as well if the coefficients are series;
- extension to higher order.
Non-linear differential equations
(on an example)
Elliptic curves

An elliptic curve is defined by an equation

$$E : y^2 = x^3 + Ax + B,$$

with the condition that $x^3 + Ax + B$ has no multiple root.

- a lot of maths around them (Fermat);
- since the 1980’s, used in cryptology as well.

Today’s question

- find a map $F$ between two curves $E$ and $E’$;
- knowing that $F(x, y) = (\varphi(x), y\varphi’(x))$. 
A first differential equation

Plugging the coordinates of $F$ in the equation defining $E'$ yields

$$(y\varphi(x)')^2 = \varphi(x)^3 + A'\varphi(x) + B'$$

that is

$$(x^3 + Ax + B)\varphi(x)'^2 = \varphi(x)^3 + A'\varphi(x) + B'.$$

However, we don’t know the initial condition at 0 but at $\infty$:

$$\varphi(x) = x + \sum_i \frac{h_i}{x^i}.$$ 

We change the unknown function, defining

$$S(x) = \frac{1}{\sqrt{\varphi\left(\frac{1}{x^2}\right)}} \quad \text{or} \quad \varphi(x) = \frac{1}{S\left(\frac{1}{\sqrt{x}}\right)^2}.$$
The equation we solve

After simplifications, we get

\[(Bx^6 + Ax^4 + 1)S'^2 - (B'S^6 + A'S^4 + 1) = 0, \quad S(0) = 0.\]

So we want to apply Newton iteration to the operator

\[P(S) = (Bx^6 + Ax^4 + 1)S'^2 - (B'S^6 + A'S^4 + 1).\]

To find out its “derivative”, use Taylor expansion

\[P(S + T) = P(S) + 2(Bx^6 + Ax^4 + 1)S'T' - (6B'S^5 + 4A'S^3)T + T'^2R_0 + T^2R_1.\]

This amounts to linearize the equation.
Solving the linearized equation

Suppose that

• you know $S = s_0 + \cdots + s_{n-1}x^{n-1}$,

• you look for $T = s_n x^n + \cdots$.

Then $T' = ns_n x^{n-1} + \cdots$, so

$$T'^2 = 0 \mod x^{2n-2} \quad \text{and} \quad T^2 = 0 \mod x^{2n-2}.$$ 

Hence, we can solve the linearized equation

$$2(Bx^6 + Ax^4 + 1)S'T' - (6B'S^5 + 4A'S^3)T = -P(S) \mod x^{2n-2}.$$ 

Conclusion.

• This gives $T \mod x^{2n-1}$ in $O(M(n))$

• So the cumulated cost is $O(M(n))$. 