CS 830
D-finiteness and P-recursiveness
Éric Schost
eschost@uwo.ca
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
P-recursive sequences

Def.

• A sequence $u_n$ is P-recursive if it satisfies a recurrence with polynomial coefficients

$$p_d(n) u_{n+d} + p_{d-1}(n) u_{n+d-1} + \cdots + p_0(n) u_n = 0$$

Examples.

• constant sequences,
• recurrences with constant coefficients,
• factorial,
• hypergeometric $u_{n+1}/u_n = a(n)/b(n)$. 
Equivalence

Theorem.

- The series

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

is D-finite if and only if the sequence \((f_i)\) is P-recursive.

Examples.

- recurrence with constant coefficients \(\iff\) rational series.

- \(f_i = 1/i! \iff\) exponential.

Proof for the exponential

Suppose that $f$ is a solution of

$$f' = f.$$ 

We know that $f$ is the exponential.

With

$$f = \sum_{i \geq 0} f_i x^i,$$

we get

$$f' = \sum_{i \geq 0} (i + 1) f_{i+1} x^i.$$ 

So

$$(i + 1) f_{i+1} = f_i.$$
Proof in general

In general, with

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

we get

\[ f' = \sum_{i \geq 0} (i + 1) f_{i+1} x^i \quad \text{and} \quad f'' = \sum_{i \geq 0} (i + 1)(i + 2) f_{i+2} x^i, \ldots \]

Multiplying by a monomial shifts the coefficients:

\[ x^\ell f' = \sum_{i \geq 0} (i + 1) f_{i+1} x^{i+\ell} = \sum_{i \geq \ell} (i - \ell + 1) f_{i-\ell+1} x^i, \ldots \]

So extracting coefficients gives a recurrence on the \( f_i \).
Converse on an example

Consider the factorial

\[ f_i = i!, \quad \text{so that} \quad f_{i+1} = (i + 1)f_i. \]

Let \( f = \sum_{i \geq 0} f_i x^i \).

Multiply by \( x^{i+1} \) and sum over all \( i \geq 0 \).

\[ \sum_{i \geq 0} f_{i+1} x^{i+1} = f - 1 \quad \text{and} \quad \sum_{i \geq 0} (i + 1) x^{i+1} = x(x f' + f). \]

So

\[ x^2 f' + (x - 1) f = -1 \quad \text{or} \quad x^2 f'' + (3x - 1)f' - f = 0. \]
Questions to deal with

1. Computing **one term** in a P-recursive sequence
   - baby steps / giant steps
   - binary splitting

2. Computing **several terms**
   - convert to the differential equation and use Newton
   - unroll the recurrence

3. **Prove** and **discover** identities
   - **easy**: hypergeometric sequences
   - **hard**: Gröbner bases
Computing one term
Baby steps / giant steps

This is the method to use when coefficient size does not matter.

Prop.

- Consider $u_n$ defined by a recurrence of order $d$ with coefficients of degree $p$.
- Then the $n$th term can be computed in $O(M(\sqrt{n}) \log n)$, where the big-Oh depends on $d$ and $p$.

Example

- The sequence $u_{n+1} = (n + 1)u_n$, computed modulo an integer $N$.
- This leads to the best deterministic, proved algorithm for factoring integers.
Preliminaries

Evaluation and interpolation

- Given a polynomial $P$ of degree $m - 1$, and $m$ evaluation points $a_0, \ldots, a_{m-1}$ one can compute

$$P(a_0), \ldots, P(a_{m-1})$$

in $O(M(m) \log(m))$ operations.

- Conversely, given the values, one can recover $P$ in the same cost.

Main ideas

- **Divide-and-conquer:** replace the original problem by the evaluation of a polynomial $P_0$ at the first half of the points and a polynomial $P_1$ at the second half.

- **Cost:** $C(n) \leq 2C(n/2) + O(M(n))$. 
The example of the factorial

Consider the sequence $u_{n+1} = (n + 1)u_n$, $u_0 = 1$.

To compute $u_n$, let $m = \sqrt{n}$ and introduce

$$P = (x + 1) \cdots (x + m).$$

Then $u_n$ is given by

$$u_n = P(0) P(m) P(2m) \cdots P((m - 1)m).$$

Algorithm

- Compute $P$ (divide-and-conquer) $O(M(m) \log m)$
- Evaluate it at $0, m, \ldots, (m - 1)m$ $O(M(m) \log m)$
- Multiply the values $O(m)$
Application to factoring integers

Suppose you want to factor $p \in \mathbb{N}$ into primes.

- It’s enough to find all prime factors $< \sqrt{p}$.
- Testing one number mod $p$ costs $O((\log p)^{O(1)})$.
- So naive cost $O(\sqrt{p}(\log p)^{O(1)})$

Better: let $n = \sqrt{p}$ and $m = \sqrt{n}$, and compute the slices

$a_0 = 1 \cdot \cdots \cdot m \mod p$, $a_1 = (m+1) \cdot \cdots \cdot (2m) \mod p$, $\ldots a_{m-1} = (m^2-m+1) \cdot \cdots \cdot m^2 \mod p$,

- cost in $\sqrt[4]{p}$.
- if gcd($a_i, p$) = 1, no divisor in the slice $i$.
- as soon as you found gcd($a_i, p$) $\neq 1$, test all elements in $a_i$.
- repeat.
Binary splitting

When the coefficient size matters,

• the previous analysis is not adapted;
• quasi-optimal algorithms exist.

Example: factorial in $\mathbb{N}$.

• We still write $M(n)$ for the cost of multiplying integers of size $n$.
• The factorial $n!$ has about $n \log n$ digits.

Prop.

• Using binary splitting, one can compute $n!$ in $O(M(n \log n) \log n)$ bit operations.
Splitting

Let $P(a, b) = a(a + 1) \cdots b$, so that we want $P(1, n)$.

Binary splitting:

$$P(a, b) = P(a, m)P(m, b) \quad \text{with} \quad m = \lfloor (a + b)/2 \rfloor.$$ 

Cost:

$$C(a, b) = C(a, m) + C(m, b) + M(\log P(m, b)) \leq 2C(m, b) + M(\log P(m, b)).$$

The splitting scheme

- $C(1, n) \leq 2C(n/2, n) + M(\log P(n/2, n))$
- $C(n/2, n) \leq 2C(3n/4, n) + M(\log P(3n/4, n))$
- $C(3n/4, n) \leq 2C(7n/8, n) + M(\log P(7n/8, n))$
Let \( P(a, b) = a(a + 1) \cdots b \), so that we want \( P(1, n) \).

Binary splitting:

\[
P(a, b) = P(a, m)P(m, b) \quad \text{with} \quad m = \lfloor (a + b)/2 \rfloor.
\]

Cost:

\[
C(a, b) = C(a, m) + C(m, b) + M(\log P(m, b)) \leq 2C(m, b) + M(\log P(m, b)).
\]

The splitting scheme

- \( C(1, n) \leq 2C(n/2, n) + M(\log P(n/2, n)) \)
- \( 2C(n/2, n) \leq 4C(3n/4, n) + 2M(\log P(3n/4, n)) \)
- \( 4C(3n/4, n) \leq 8C(7n/8, n) + 4M(\log P(7n/8, n)) \)
Solving the recurrence

These equalities give

\[ C(1, n) \leq 2^k C(n - \frac{n}{2^k}, n) + \sum_{j=1}^{k} 2^{j-1} M(\log P(n - \frac{n}{2^j}, n)). \]

Simplifications

- remember that
  \[ P(n - \frac{n}{2^j}, n) = (n - \frac{n}{2^j}) \cdots n \leq n^{n/2^j} \]
  so its log is \( \leq \frac{n}{2^j} \log n \).
- so its contribution is \( \leq M(n \log n) \).
Solving the recurrence

Putting everything together gives

\[ C(1, n) \leq 2^k C(n - \frac{n}{2^k}, n) + kM(n \log n). \]

We stop the recursion for \( k = \log n \), which gives

\[ C(1, n) \in O(M(n \log n) \log n). \]
Second example: computing $e = \exp(1)$

The sequence

$$e_n = \sum_{k=1}^{n} \frac{1}{k!}$$

converges to $e$, and $0 \leq e - e_n \leq \frac{1}{nn!}$.

Consequence

- To compute $m$ digits of $e$, compute $e_n$, with

$$n = \frac{m}{\log m}$$
The recursion

The sequence $f_n = 1/n!$ satisfies the recurrence

$$(n + 1)f_{n+1} = f_n.$$ 

Because $e_{n+1} - e_n = f_{n+1}$, we get

$$(n + 1)(e_{n+1} - e_n) = (n + 1)f_{n+1} = f_n = e_n - e_{n-1},$$

which becomes

$$\begin{bmatrix} e_{n+1} \\ e_n \end{bmatrix} = \frac{1}{n + 1} \begin{bmatrix} n + 2 & -1 \\ n + 1 & 0 \end{bmatrix} \begin{bmatrix} e_n \\ e_{n-1} \end{bmatrix} = M(n) \begin{bmatrix} e_n \\ e_{n-1} \end{bmatrix}.$$ 

So to compute $e_n$, we actually compute

$$\frac{1}{n!} M(n) \cdots M(1).$$

Same thing as the factorial!