Nth term of a P-recursive sequence (Part II)
Solutions of linear differential equations

Marc Mezzarobba

November 9, 2023
Solutions to last week’s exercises
Exercise 1

Let $f(x) = (1 + x + x^2)^N \in \mathbb{Z}[x]$. Give an algorithm that computes the parity of all coefficients of $f$ in $O(M(N))$ bit operations.
Exercise 1

Let \( f(x) = (1 + x + x^2)^N \in \mathbb{Z}[x] \). Give an algorithm that computes the parity of all coefficients of \( f \) in \( O(M(N)) \) bit operations.

Solution. Binary powering in \( (\mathbb{Z}/2\mathbb{Z})[x] \). Cost:

\[
\leq M(2^k) + M(2^k) + \cdots + M(2) \quad \text{where} \quad k = \lceil \log_2 N \rceil
\]

\[
= O(M(N)).
\]

(Binary powering in \( \mathbb{Z}[x] \) would take \( \Omega(N^2) \) bit operations.)
Exercise 2

Let $f \in K[x]$ be a polynomial of degree less than $d$.

1. Show that the sequence $(f(n))_{n \in \mathbb{N}}$ satisfies a linear recurrence with constant coefficients, of characteristic polynomial $(t - 1)^d$. 
Exercise 2

Let \( f \in \mathbb{K}[x] \) be a polynomial of degree less than \( d \).

1. Show that the sequence \( (f(n))_{n \in \mathbb{N}} \) satisfies a linear recurrence with constant coefficients, of characteristic polynomial \((t - 1)^d\).

**Solution.** The polynomial \( \Delta f = f(n + 1) - f(n) \) has degree \( d - 1 \).

By induction, \( \Delta^d f = 0 \).
Exercise 2

2. Assuming $n \geq d^2$, give an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n-1$ in $O(n M(d) / d)$ operations in $\mathbb{K}$. Generalize to evaluation on an arbitrary arithmetic progression $a, a+\delta, a+2\delta, \ldots, a+(n-1)\delta$. 
Exercise 2

2. Assuming $n \geq d^2$, give an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n-1$ in $O(n \, M(d) / d)$ operations in $\mathbb{K}$. Generalize to evaluation on an arbitrary arithmetic progression $a, a + \delta, a + 2 \delta, \ldots, a + (n-1) \delta$.

Solution. By the previous question,

$$\sum_{n=0}^{\infty} f(n) \, x^n = \frac{p(x)}{(1 - t)^d}$$

for some $p \in \mathbb{K}[x]_{<d}$.

Algorithm.

3. Compute $f(0), f(1), \ldots, f(2 \, d - 1)$

4. Compute $p$ as above

5. Expand the rational series to precision $n$
Exercise 2

2. Assuming $n \geq d^2$, give an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n - 1$ in $O(n \cdot M(d) / d)$ operations in $\mathbb{K}$. Generalize to evaluation on an arbitrary arithmetic progression $a, a + \delta, a + 2\delta, \ldots, a + (n - 1)\delta$.

Solution. By the previous question,

$$
\sum_{n=0}^{\infty} f(n) x^n = \frac{p(x)}{(1 - t)^d} \quad \text{for some } p \in \mathbb{K}[x]_{<d}.
$$

Algorithm.

3. Compute $f(0), f(1), \ldots, f(2d - 1)$  
   $\quad \text{O}(M(d) \log(d))$

4. Compute $p$ as above

5. Expand the rational series to precision $n$
Exercise 2

2. Assuming $n \geq d^2$, give an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n-1$ in $O(n \frac{M(d)}{d})$ operations in $\mathbb{K}$. Generalize to evaluation on an arbitrary arithmetic progression $a, a + \delta, a + 2\delta, \ldots, a + (n-1)\delta$.

Solution. By the previous question,

$$\sum_{n=0}^{\infty} f(n) x^n = \frac{p(x)}{(1 - t)^d}$$

for some $p \in \mathbb{K}[x]_{<d}$.

Algorithm.

3. Compute $f(0), f(1), \ldots, f(2d - 1)$ \hspace{2cm} $O(M(d) \log(d))$

4. Compute $p$ as above \hspace{2cm} $O(M(d))$

5. Expand the rational series to precision $n$
Exercise 2

2. Assuming $n \geq d^2$, give an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n - 1$ in $O(n \cdot M(d)/d)$ operations in $\mathbb{K}$. Generalize to evaluation on an arbitrary arithmetic progression $a, a + \delta, a + 2 \delta, \ldots, a + (n - 1) \delta$.

Solution. By the previous question,

$$\sum_{n=0}^{\infty} f(n) x^n = \frac{p(x)}{(1 - t)^d} \quad \text{for some } p \in \mathbb{K}[x]_{<d}. $$

Algorithm.

3. Compute $f(0), f(1), \ldots, f(2d - 1)$ \hspace{1cm} $O(M(d) \log(d))$
4. Compute $p$ as above \hspace{1cm} $O(M(d))$
5. Expand the rational series to precision $n$ \hspace{1cm} $O(n \cdot M(d)/d)$
Exercise 2

2. Assuming $n \geq d^2$, give an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n - 1$ in $O(nM(d)/d)$ operations in $\mathbb{K}$. Generalize to evaluation on an arbitrary arithmetic progression $a, a + \delta, a + 2\delta, \ldots, a + (n - 1)\delta$.

Solution. By the previous question,

$$
\sum_{n=0}^{\infty} f(n)x^n = \frac{p(x)}{(1 - t)^d} \quad \text{for some } p \in \mathbb{K}[x]_{<d}.
$$

Algorithm.

1. Compute $g(x) = f(\delta x) \in \mathbb{K}[x]$
2. Compute $h(x) = g(a + x) \in \mathbb{K}[x]$
3. Compute $h(0), h(1), \ldots, h(2d - 1)$ \hspace{2em} $O(M(d) \log(d))$
4. Compute $p$ as above \hspace{1em} $O(M(d))$
5. Expand the rational series to precision $n$ \hspace{1em} $O(nM(d)/d)$
2. Assuming $n \geq d^2$, give an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n - 1$ in $O(n M(d)/d)$ operations in $\mathbb{F}$. Generalize to evaluation on an arbitrary arithmetic progression $a, a + \delta, a + 2 \delta, \ldots, a + (n - 1) \delta$.

**Solution.** By the previous question,

$$
\sum_{n=0}^{\infty} f(n) x^n = \frac{p(x)}{(1 - t)^d}
$$
for some $p \in \mathbb{F}[x]_{<d}$.

**Algorithm.**

1. Compute $g(x) = f(\delta x) \in \mathbb{F}[x] \quad \text{O(d)}$
2. Compute $h(x) = g(a + x) \in \mathbb{F}[x] \quad \text{O(M(d) log(d))}$
3. Compute $h(0), h(1), \ldots, h(2d - 1) \quad \text{O(M(d))}$
4. Compute $p$ as above \quad \text{O(M(d))}
5. Expand the rational series to precision $n \quad \text{O(n M(d)/d)}$
Exercise 2

2. Assuming $n \geq d^2$, give an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n-1$ in $O(n M(d) / d)$ operations in $\mathbb{K}$. Generalize to evaluation on an arbitrary arithmetic progression $a, a+\delta, a+2\delta, \ldots, a+(n-1)\delta$.

Solution. By the previous question,

$$
\sum_{n=0}^{\infty} f(n) x^n = \frac{p(x)}{(1-t)^d}
$$

for some $p \in \mathbb{K}[x]_{<d}$.

Algorithm.

1. Compute $g(x) = f(\delta x) \in \mathbb{K}[x]$ \hspace{1cm} $O(d)$
2. Compute $h(x) = g(a+x) \in \mathbb{K}[x]$ \hspace{1cm} $O(d^2)$, even $O(M(d) \log(d))$
3. Compute $h(0), h(1), \ldots, h(2d-1)$ \hspace{1cm} $O(M(d) \log(d))$
4. Compute $p$ as above \hspace{1cm} $O(M(d))$
5. Expand the rational series to precision $n$ \hspace{1cm} $O(n M(d) / d)$
Exercise 2

3. Sketch an algorithm to evaluate \( f \) on the \( n \) points \( 0, \ldots, n - 1 \) in

\[ O(d \, n) \] additions or subtractions \( + \) \( \tilde{O}(d) \) multiplications
Exercise 2

3. Sketch an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n - 1$ in $O(d \, n)$ additions or subtractions $+$ $\tilde{O}(d)$ multiplications

Solution. In $\tilde{O}(d)$ operations, we can compute $f(0), f(1), \ldots, f(d - 1)$. Using the relation $\Delta^k f(d + 1) = \Delta^k f(d) + \Delta^{k-1} f(d)$ and the fact that $\Delta^{d-1} f$ is constant, we can then recover all values in $O(d \, n)$ additions and subtractions:

$$
\begin{array}{cccccccc}
  f(0) & f(1) & f(2) & \cdots & f(d - 2) & f(d - 1) \\
\end{array}
$$
Exercise 2

3. Sketch an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n - 1$ in

$$O(d \ n) \text{ additions or subtractions} + \tilde{O}(d) \text{ multiplications}$$

Solution. In $\tilde{O}(d)$ operations, we can compute $f(0), f(1), \ldots, f(d - 1)$. Using the relation $\Delta^k f(d + 1) = \Delta^k f(d) + \Delta^{k-1} f(d)$ and the fact that $\Delta^{d-1} f$ is constant, we can then recover all values in $O(d \ n)$ additions and subtractions:

$$
\begin{align*}
&f(0) \quad f(1) \quad f(2) \quad \ldots \quad f(d - 2) \quad f(d - 1) \\
&\Delta f(0) \quad \Delta f(1) \quad \Delta f(2) \quad \ldots \quad \Delta f(d - 2)
\end{align*}
$$
Exercise 2

3. Sketch an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n-1$ in

$$O(d \ n) \text{ additions or subtractions} + \tilde{O}(d) \text{ multiplications}$$

**Solution.** In $\tilde{O}(d)$ operations, we can compute $f(0), f(1), \ldots, f(d-1)$. Using the relation $\Delta^k f(d+1) = \Delta^k f(d) + \Delta^{k-1} f(d)$ and the fact that $\Delta^{d-1} f$ is constant, we can then recover all values in $O(d \ n)$ additions and subtractions:

\[
\begin{array}{cccccccc}
  f(0) & f(1) & f(2) & \cdots & f(d-2) & f(d-1) \\
  \Delta f(0) & \Delta f(1) & \Delta f(2) & \cdots & \Delta f(d-2) \\
  \Delta^2 f(0) & \Delta^2 f(1) & \Delta^2 f(2) \\
\end{array}
\]
Exercise 2

3. Sketch an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n - 1$ in $O(d \cdot n)$ additions or subtractions, $+ \tilde{O}(d)$ multiplications.

Solution. In $\tilde{O}(d)$ operations, we can compute $f(0), f(1), \ldots, f(d - 1)$. Using the relation $\Delta^k f(d + 1) = \Delta^k f(d) + \Delta^{k-1} f(d)$ and the fact that $\Delta^{d-1} f$ is constant, we can then recover all values in $O(d \cdot n)$ additions and subtractions:

\[
\begin{array}{ccccccc}
  f(0) & f(1) & f(2) & \ldots & f(d - 2) & f(d - 1) \\
  \Delta f(0) & \Delta f(1) & \Delta f(2) & \ldots & \Delta f(d - 2) \\
  \Delta^2 f(0) & \Delta^2 f(1) & \Delta^2 f(2) & \ldots & \Delta^2 f(d - 2) \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  \Delta^{d-2} f(0) & \Delta^{d-2} f(1) &
\end{array}
\]
Exercise 2

3. Sketch an algorithm to evaluate \( f \) on the \( n \) points \( 0, \ldots, n - 1 \) in

\[ O(d \, n) \text{ additions or subtractions} + \tilde{O}(d) \text{ multiplications} \]

Solution. In \( \tilde{O}(d) \) operations, we can compute \( f(0), f(1), \ldots, f(d - 1) \).

Using the relation \( \Delta^k f(d + 1) = \Delta^k f(d) + \Delta^{k-1} f(d) \) and the fact that \( \Delta^{d-1} f \) is constant, we can then recover all values in \( O(d \, n) \) additions and subtractions:

\[
\begin{array}{cccccccc}
  f(0) & f(1) & f(2) & \cdots & f(d - 2) & f(d - 1) \\
\Delta f(0) & \Delta f(1) & \Delta f(2) & \cdots & \Delta f(d - 2) \\
\Delta^2 f(0) & \Delta^2 f(1) & \Delta^2 f(2) & \cdots & \cdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\Delta^{d-2} f(0) & \Delta^{d-2} f(1) \\
\Delta^{d-1} f(0)
\end{array}
\]
Exercise 2

3. Sketch an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n - 1$ in

$$O(d \, n) \text{ additions or subtractions} + \tilde{O}(d) \text{ multiplications}$$

Solution. In $\tilde{O}(d)$ operations, we can compute $f(0), f(1), \ldots, f(d - 1)$.

Using the relation $\Delta^k f(d + 1) = \Delta^k f(d) + \Delta^{k-1} f(d)$ and the fact that $\Delta^{d-1} f$ is constant, we can then recover all values in $O(d \, n)$ additions and subtractions:

\[
\begin{array}{cccccccc}
  f(0) & f(1) & f(2) & \cdots & f(d-2) & f(d-1) \\
  \Delta f(0) & \Delta f(1) & \Delta f(2) & \cdots & \Delta f(d-2) \\
  \Delta^2 f(0) & \Delta^2 f(1) & \Delta^2 f(2) & & \\
  \vdots & & & & \\
  \Delta^{d-2} f(0) & \Delta^{d-2} f(1) & & & \\
  \Delta^{d-1} f(0) &= \Delta^{d-1} f(1) &= \Delta^{d-1} f(2) &= \cdots &= \Delta^{d-2} f(d-1) &= \Delta^{d-1} f(d-1) &\cdots
\end{array}
\]
Exercise 2

3. Sketch an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n-1$ in

$$O(d \, n) \text{ additions or subtractions} + \tilde{O}(d) \text{ multiplications}$$

Solution. In $\tilde{O}(d)$ operations, we can compute $f(0), f(1), \ldots, f(d-1)$.

Using the relation $\Delta^k f(d+1) = \Delta^k f(d) + \Delta^{k-1} f(d)$ and the fact that $\Delta^{d-1} f$ is constant, we can then recover all values in $O(d \, n)$ additions and subtractions:

\[
\begin{array}{cccccc}
  f(0) & f(1) & f(2) & \cdots & f(d-2) & f(d-1) \\
  \Delta f(0) & \Delta f(1) & \Delta f(2) & \cdots & \Delta f(d-2) \\
  \Delta^2 f(0) & \Delta^2 f(1) & \Delta^2 f(2) & \cdots & \Delta^2 f(d-2) \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
  \Delta^{d-2} f(0) & \Delta^{d-2} f(1) & \Delta^{d-2} f(2) \\
  \Delta^{d-1} f(0) & = & \Delta^{d-1} f(1) & = & \Delta^{d-1} f(2) & = \cdots = \Delta^{d-2} f(d-1) = \Delta^{d-1} f(d-1) & \cdots \\
\end{array}
\]
Exercise 2

3. Sketch an algorithm to evaluate \( f \) on the \( n \) points 0, \ldots, \( n - 1 \) in

\[
O(dn) \text{ additions or subtractions} \quad + \quad \tilde{O}(d) \text{ multiplications}
\]

**Solution.** In \( \tilde{O}(d) \) operations, we can compute \( f(0), f(1), \ldots, f(d - 1) \).

Using the relation \( \Delta^k f(d + 1) = \Delta^k f(d) + \Delta^{k-1} f(d) \) and the fact that \( \Delta^{d-1} f \) is constant, we can then recover all values in \( O(dn) \) additions and subtractions:

\[
\begin{array}{cccccc}
  f(0) & f(1) & f(2) & \cdots & f(d - 2) & f(d - 1) \\
  \Delta f(0) & \Delta f(1) & \Delta f(2) & \cdots & \Delta f(d - 2) \\
  \Delta^2 f(0) & \Delta^2 f(1) & \Delta^2 f(2) & \cdots & \Delta^2 f(d - 2) \\
  \vdots & \vdots & \vdots & \ddots & \vdots \\
  \Delta^{d-2} f(0) & \Delta^{d-2} f(1) & \Delta^{d-2} f(2) \\
  \Delta^{d-1} f(0) & = & \Delta^{d-1} f(1) & = & \Delta^{d-1} f(2) & = & \cdots & = & \Delta^{d-2} f(d - 1) & = & \Delta^{d-1} f(d - 1) & \cdots
\end{array}
\]
Exercise 2

3. Sketch an algorithm to evaluate $f$ on the $n$ points $0, \ldots, n-1$ in

$O(d \cdot n)$ additions or subtractions + $\tilde{O}(d)$ multiplications

Solution. In $\tilde{O}(d)$ operations, we can compute $f(0), f(1), \ldots, f(d-1)$.

Using the relation $\Delta^k f(d+1) = \Delta^k f(d) + \Delta^{k-1} f(d)$ and the fact that $\Delta^{d-1} f$ is constant, we can then recover all values in $O(d \cdot n)$ additions and subtractions:

\[
\begin{array}{cccccc}
& f(0) & f(1) & f(2) & \cdots & f(d-2) & f(d-1) \\
\Delta f(0) & & & & \cdots & & \\
\Delta^2 f(0) & & & & \cdots & & \\
\vdots & & & & \cdots & & \\
\Delta^{d-2} f(0) & & & & \cdots & & \\
\Delta^{d-1} f(0) & = & \Delta^{d-1} f(1) & = & \Delta^{d-1} f(2) & = & \cdots = \Delta^{d-2} f(d-1) = \Delta^{d-1} f(d-1) \cdots
\end{array}
\]
Exercise 2

3. Sketch an algorithm to evaluate \( f \) on the \( n \) points \( 0, \ldots, n - 1 \) in

\[
O(d \, n) \text{ additions or subtractions} + \tilde{O}(d) \text{ multiplications}
\]

**Solution.** In \( \tilde{O}(d) \) operations, we can compute \( f(0), f(1), \ldots, f(d - 1) \).

Using the relation \( \Delta^k f(d + 1) = \Delta^k f(d) + \Delta^{k-1} f(d) \) and the fact that \( \Delta^{d-1} f \) is constant, we can then recover all values in \( O(d \, n) \) additions and subtractions:

\[
\begin{align*}
f(0) & \quad f(1) & \quad f(2) & \quad \cdots & \quad f(d - 2) & \quad f(d - 1) & \quad f(d) \\
\Delta f(0) & \quad \Delta f(1) & \quad \Delta f(2) & \quad \cdots & \quad \Delta f(d - 2) & \quad \Delta f(d - 1) \\
\Delta^2 f(0) & \quad \Delta^2 f(1) & \quad \Delta^2 f(2) & \quad \cdots & \quad \Delta^2 f(d - 2) \\
\vdots & \quad \vdots & \quad \vdots & \quad \ddots & \quad \vdots \\
\Delta^{d-2} f(0) & \quad \Delta^{d-2} f(1) & \quad \Delta^{d-2} f(2) \\
\Delta^{d-1} f(0) & = & \Delta^{d-1} f(1) & = & \Delta^{d-1} f(2) & = & \cdots & = & \Delta^{d-2} f(d - 1) & = & \Delta^{d-1} f(d - 1) & \cdots
\end{align*}
\]
Nth term of a P-recursive sequence (Part II)
Summary of Part I

For a C-finite sequence (order $s$, characteristic polynomial $\chi$):

- $N$th term in $O(M(s) \log(N))$ ops (binary powering modulo $\chi$ or repeated Gräffe transforms)
  (over $\mathbb{Z}$, in $O(M_\mathbb{Z}(N))$ bit ops)

- Extension: $2s$ terms in $O(M(s))$ ops (recover rational function, divide)

- First $N$ terms in $O\left(N \frac{M(s)}{s}\right)$ ops (iterate previous point)

For a P-recursive sequence (order $s$, degree $d$):

- $N$th term in $O\left(\sqrt{N}d \log(Nd) s^\theta\right)$ ops (baby steps-giant steps: $\sqrt{N}d$ products of $\sqrt{N}d$ matrices, multipoint evaluation)

Note: no known way of getting rid of $s^\theta$ while keeping the $\tilde{O}(N^{1/2})$ cost.
1 Baby steps, giant steps (Part II)
Let $\Sigma_n = \sum_{k=0}^{n-1} u_k \xi^k$ for some fixed $\xi \in \mathbb{R}$.

If $(u_n)_{n \in \mathbb{N}}$ satisfies a rec. with poly. coeffs, then $(\Sigma_n)$ too. (why?)

Better:

$$
\begin{pmatrix}
\frac{u_n+1 \xi^{n+1}}{\Sigma_{n+1}} \\
\vdots \\
\frac{u_{n+s} \xi^{n+1}}{\Sigma_{n+1}}
\end{pmatrix} = \frac{1}{b_s(n)} \begin{pmatrix}
B(n) \xi & 0 & \cdots & 0 \\
0 & \ddots & \vdots & \vdots \\
0 & \cdots & B(n) \xi & 0 \\
b_s(n) & 0 & \cdots & 0
\end{pmatrix} \begin{pmatrix}
\frac{u_n \xi^n}{\Sigma_n} \\
\vdots \\
\frac{u_{n+s-1} \xi^n}{\Sigma_n}
\end{pmatrix}
$$

Working with $p$-bit approximations and ignoring rounding errors:

$$
\Sigma_N \text{ to } p\text{-bit precision } \text{ in } \mathcal{O}(M(\sqrt{N}) \log(N) M_\mathbb{Z}(p)) \text{ ops}
$$

Target accuracy $2^{-t}$ typically requires $N, p = \mathcal{O}(t)$

$\leadsto$ evaluation of D-finite series to precision $t$ in $\tilde{\mathcal{O}}(t^{3/2})$ ops
Rectangular splitting for polynomials

[Paterson & Stockmeyer 1973]

Goal: evaluate \( p(\xi) = a_{d-1} \xi^{d-1} + \cdots + a_0 \) with “small” \( a_i \) at a “large” (p-bit) \( \xi \in \mathbb{R} \)

\[
p(x) = \left( a_0 + a_1 x + \cdots + a_{\ell-1} x^{\ell-1} \right) x^0 \quad (d = m \ell)
+ \left( a_\ell + a_{\ell+1} x + \cdots + a_{2\ell-1} x^{\ell-1} \right) x^\ell
\vdots
+ \left( a_{(m-1)\ell} + a_{(m-1)\ell+1} x + \cdots + a_{m\ell-1} x^{\ell-1} \right) x^{(m-1)\ell}
\]

Same idea for evaluating \( p \in \mathbb{K}[x] \) on a polynomial / matrix / …
Rectangular splitting for polynomials

[Paterson & Stockmeyer 1973]

Goal: evaluate $p(\xi) = a_{d-1} \xi^{d-1} + \cdots + a_0$ with “small” $a_i$ at a “large” ($p$-bit) $\xi \in \mathbb{R}$

$$p(x) = \left( a_0 + a_1 x + \cdots + a_{\ell-1} x^{\ell-1} \right) x^0 \quad (d = m \ell)$$
$$+ \left( a_\ell + a_{\ell+1} x + \cdots + a_{2\ell-1} x^{\ell-1} \right) x^\ell$$
$$\vdots$$
$$+ \left( a_{(m-1)\ell} + a_{(m-1)\ell+1} x + \cdots + a_{m\ell-1} x^{\ell-1} \right) x^{(m-1)\ell}$$

Algorithm.

1. (Baby steps) Compute $\xi^2, \ldots, \xi^\ell$ \quad $O(\ell)$ costly ops
2. Evaluate the inner polynomials \quad $O(\ell m)$ cheap ops
3. (Giant steps) Compute $\xi^{2\ell}, \ldots, \xi^{(m-1)\ell}$ \quad $O(m)$ costly ops
4. Evaluate the outer polynomial \quad $O(m)$ costly ops

Same idea for evaluating $p \in \mathbb{K}[x]$ on a polynomial / matrix / …
Rectangular splitting for hypergeometric series

\[ f(x) = a_0 + a_0 a_1 x + a_0 a_1 a_2 x^2 + \cdots \quad a_n = \frac{p(n)}{q(n)} \]

\[ + a_0 \cdots a_{\ell-1} \left( a_\ell (1 + a_{\ell+1} (x + a_{\ell+2} (x^2 + \cdots + a_{2\ell-1} x^{\ell-1})))) x^\ell \right. \]

\[ + a_\ell \cdots a_{2\ell-1} \left( a_0 (1 + a_1 (x + a_2 (x^2 + \cdots + a_{\ell-1} x^{\ell-1})))) x^0 \right. \]

\[ + a_{(m-1)\ell} \cdots a_{m\ell-1} \left( a_{(m-1)\ell} (1 + \cdots (\cdots (\cdots + a_{m\ell-1} x^{\ell-1})))) x^{(m-1)\ell} \right) \]
2 Binary splitting
Computing $N!$ in quasi-linear time
Computing $N!$ in quasi-linear time

**Algorithm.** Use a product tree. That is, split the product as

$$N! = 1 \cdot 2 \cdots m \cdot (m + 1) \cdots N,$$

$$m = \lfloor N/2 \rfloor;$$

and recurse.

\[P(0,m) \quad \text{and} \quad P(m,N)\]
Computing $N!$ in quasi-linear time

**Algorithm.** Use a product tree. That is, split the product as

$$N! = 1 \cdot 2 \cdots m \cdot (m+1) \cdots N, \quad m = \lfloor N/2 \rfloor;$$

and recurse.

Using \text{size}(P(\ell, h)) \leq 1 + (h - \ell) \log_2 N, the cost $C(\ell, h)$ of computing $P(\ell, h)$ satisfies

$$C(\ell, h) \leq C(\ell, m) + C(m, h) + M_Z(1 + (m+1) \log_2 N) \quad m = \lfloor (\ell + h)/2 \rfloor.$$
Computing $N!$ in quasi-linear time

**Algorithm.** Use a product tree. That is, split the product as

$$N! = \underbrace{1 \cdot 2 \cdots m}_P(0,m) \cdot \underbrace{(m+1) \cdots N}_P(m,N), \quad m = \lfloor N/2 \rfloor;$$

and recurse.

Using $\text{size}(P(\ell, h)) \leq 1 + (h - \ell) \log_2 N$, the cost $C(\ell, h)$ of computing $P(\ell, h)$ satisfies

$$C(\ell, h) \leq C(\ell, m) + C(m, h) + M_Z(1 + (m + 1) \log_2 N) \quad m = \lfloor (\ell + h)/2 \rfloor.$$ 

The total cost of the multiplications at any given recursion depth is

$$\leq \sum_i M_Z(1 + (m_i + 1) \log_2 N) \quad \text{where} \quad \sum_i m_i \leq \frac{N}{2}.$$
Computing $N!$ in quasi-linear time

**Algorithm.** Use a product tree. That is, split the product as

$$N! = 1 \cdot 2 \cdots m \cdot (m + 1) \cdots N, \quad m = \lfloor N/2 \rfloor;$$

and recurse.

Using $\text{size}(P(\ell, h)) \leq 1 + (h - \ell) \log_2 N$, the cost $C(\ell, h)$ of computing $P(\ell, h)$ satisfies

$$C(\ell, h) \leq C(\ell, m) + C(m, h) + M_Z(1 + (m + 1) \log_2 N) \quad m = \lfloor (\ell + h)/2 \rfloor.$$ 

The total cost of the multiplications at any given recursion depth is

$$\leq \sum_i M_Z(1 + (m_i + 1) \log_2 N) \quad \text{where} \quad \sum_i m_i \leq \frac{N}{2}$$

$$\leq M_Z\left(\frac{N}{2} \log_2 N + O(N)\right).$$
Computing $N!$ in quasi-linear time

**Algorithm.** Use a product tree. That is, split the product as

\[ N! = \underbrace{1 \cdot 2 \cdots m \cdot (m + 1) \cdots N}, \quad m = \lfloor N/2 \rfloor; \]

and recurse.

Using $\text{size}(P(\ell, h)) \leq 1 + (h - \ell) \log_2 N$, the cost $C(\ell, h)$ of computing $P(\ell, h)$ satisfies

\[ C(\ell, h) \leq C(\ell, m) + C(m, h) + M\mathbb{Z}(1 + (m + 1) \log_2 N) \quad m = \lfloor (\ell + h)/2 \rfloor. \]

The total cost of the multiplications at any given recursion depth is

\[ \leq \sum_i M\mathbb{Z}(1 + (m_i + 1) \log_2 N) \quad \text{where} \quad \sum_i m_i \leq \frac{N}{2} \]

\[ \leq M\mathbb{Z}\left( \frac{N}{2} \log_2 N + O(N) \right). \]

Total $O(M\mathbb{Z}(N \log N) \log N)$. 

Nth term of a P-recursive sequence

[Chudnovsky & Chudnovsky 1987]

\[ b_s(n) u_{n+s} + \cdots + b_1(n) u_{n+1} + b_0(n) u_n = 0, \quad b_i \in \mathbb{Z}[n] \]

Same idea as before:
write \( U_n = (u_n, \ldots, u_{n+s-1}) \) and \( U_N = \frac{1}{b_s(N-1) \cdots b_s(1) b_s(0)} B(N-1) \cdots B(1) B(0) U_0 \)

**Algorithm.**

1. Compute \( B(N-1) \cdots B(1) B(0) \) by binary splitting
2. Compute \( b_s(N-1) \cdots b_s(1) b_s(0) \) by binary splitting
3. Divide

**Theorem.** One can compute the \( N \)th term of a sequence \( (u_n) \in \mathbb{Q}^N \) given by a nonsingular recurrence with coefficients in \( \mathbb{Z}[n] \) in \( O(n \log n) \) bit operations.

Exercise: what is the benefit of pulling out the denominators?
Nth term of a P-recursive sequence

\[ b_s(n) u_{n+s} + \cdots + b_1(n) u_{n+1} + b_0(n) u_n = 0, \quad b_i \in \mathbb{Z}[n] \]

Same idea as before:
write \( U_n = (u_n, \ldots, u_{n+s-1}) \) and \( U_N = \frac{1}{b_s(N-1) \cdots b_s(1) b_s(0)} B(N-1) \cdots B(1) B(0) U_0 \)

\textbf{Algorithm.} (costs for fixed \( b_0 : s \), hides \( s^\theta \) and dependency on \( d \))

1. Compute \( B(N-1) \cdots B(1) B(0) \) \textbf{by binary splitting} \( \text{O}(M(n \log n) \log(n)) \)
2. Compute \( b_s(N-1) \cdots b_s(1) b_s(0) \) \textbf{by binary splitting} \( \text{O}(M(n \log n) \log(n)) \)
3. Divide

\textbf{Theorem.} One can compute the \( N \)th term of a sequence \( (u_n) \in \mathbb{Q}^N \) given by a nonsingular recurrence with coefficients in \( \mathbb{Z}[n] \) in \( \text{O}(M(n \log n) \log(n)) \) bit operations.

Exercise: what is the benefit of pulling out the denominators?
Nth term of a P-recursive sequence

[Chudnovsky & Chudnovsky 1987]

\[ b_s(n) u_{n+s} + \cdots + b_1(n) u_{n+1} + b_0(n) u_n = 0, \quad b_i \in \mathbb{Z}[n] \]

Same idea as before:
write \( u_n = (u_n, \ldots, u_{n+s-1}) \) and
\[
U_N = \frac{1}{b_s(N-1) \cdots b_s(1) b_s(0)} B(N-1) \cdots B(1) B(0) U_0
\]

**Algorithm.**
(costs for fixed \( b_0: s \), hides \( s^\Theta \) and dependency on \( d \))

1. Compute \( B(N-1) \cdots B(1) B(0) \) **by binary splitting** \( O(M(n \log n \log(n))) \)
2. Compute \( b_s(N-1) \cdots b_s(1) b_s(0) \) **by binary splitting** \( O(M(n \log n \log(n))) \)
3. Divide (gcd!)

**Theorem.** One can compute the \( N \)th term of a sequence \( (u_n) \in \mathbb{Q}^N \) given by a nonsingular recurrence with coefficients in \( \mathbb{Z}[n] \) in \( O(M(n \log^2 n)) \) bit operations.

Exercise: what is the benefit of pulling out the denominators?
Problem. Compute the coefficient of $x^{2N}$ in

$$(1 + x)^{2N} (1 + x + x^2)^N.$$
Problem. Compute the coefficient of $x^{2N}$ in

$$(1 + x)^{2N} (1 + x + x^2)^N.$$ 

Let $f(x) = (1 + x)^{2N} (1 + x + x^2)^N$. One has

$$\frac{f'(x)}{f(x)} = 2N \frac{1}{1 + x} + N \frac{2x + 1}{1 + x + x^2}.$$
Partial sums of D-finite series

Again: $\sum_{n=0}^{n-1} u_k \xi^k$ satisfies a recurrence (but $\xi$ enters into the recurrence!)

**Corollary.** One can evaluate the $N$th partial sum of a fixed D-finite series at a fixed point $\xi \in \mathbb{Q}$ in $O(M(N \log^2 N))$ bit operations.

Typical case: $N = O(t)$

$t = \text{target bit accuracy}$

If $N$ and the bit size of $\xi$ are both $\Theta(t)$, the cost becomes quadratic in $t$!
Partial sums of D-finite series: analysis

Recall that we can use the recurrence

\[
\begin{pmatrix}
u_{n+1} \xi^{n+1} \\
\vdots \\
u_{n+s} \xi^{n+1} \\
\sum_{n+1}
\end{pmatrix} = \frac{1}{b_s(n)} \begin{pmatrix}
\begin{pmatrix}
B(n) \xi \\
b_s(n) 0 \ldots 0 b_s(n)
\end{pmatrix} 0 \\
\end{pmatrix} \begin{pmatrix}
u_n \xi^n \\
\vdots \\
u_{n+s-1} \xi^n \\
\sum_n
\end{pmatrix}
\]

If $\xi$ is of bit size $\leq h$, then (for a fixed differential equation):

- the matrices, taken at $n \leq N$, have bit size $O(h + \log N)$,
- the cost of computing the product tree for $N$ terms becomes

\[
O \left( M \left( N \left( h + \log N \right) \right) \log N \right).
\]
Application: High-precision evaluation of classical constants

- $e = \exp(1)$ with error $\leq 2^{-t}$ in $O(M(t \log t))$ bit operations
Application: High-precision evaluation of classical constants

- \( e = \exp(1) \) with error \( \leq 2^{-t} \) in \( O(M(t \log t)) \) bit operations

\[
e = \sum_{k=0}^{n-1} \frac{1}{k!} + \frac{1}{n!} \sum_{k=0}^{\infty} \frac{1}{(n+1) \cdots (n+k)}, \quad \frac{e}{n!} \leq 2^{-t} \text{ for } n = \frac{t + o(t)}{\log_2 t}
\]
Application: High-precision evaluation of classical constants

- \( e = \exp(1) \) with error \( \leq 2^{-t} \) in \( O(M(t \log t)) \) bit operations

\[
e = \sum_{k=0}^{n-1} \frac{1}{k!} + \frac{1}{n!} \sum_{k=0}^{\infty} \frac{1}{(n+1) \cdots (n+k)}, \quad \frac{e}{n!} \leq 2^{-t} \text{ for } n = \frac{t + o(t)}{\log_2 t}
\]

Cost of the binary splitting method: \( O\left(M\left(\frac{t}{\log_2 t} \log \left(\frac{t}{\log_2 t}^2\right)\right)\right) = O(M(t \log t)) \).
Application: High-precision evaluation of classical constants

- $e = \exp(1)$ with error $\leq 2^{-t}$ in $O(M(t \log t))$ bit operations

\[
e = \sum_{k=0}^{n-1} \frac{1}{k!} + \frac{1}{n!} \sum_{k=0}^{\infty} \frac{1}{(n+1) \cdots (n+k)}, \quad \frac{e}{n!} \leq 2^{-t} \text{ for } n = \frac{t + o(t)}{\log_2 t}
\]

Cost of the binary splitting method: $O\left(M\left(\frac{t}{\log_2 t} \log \left(\frac{t}{\log_2 t}\right)^2\right)\right) = O(M(t \log t))$.

- $\ln(2)$ in $O(M(t \log(t)^2))$ bit operations:
Application: High-precision evaluation of classical constants

- $e = \exp(1)$ with error $\leq 2^{-t}$ in $O(M(t \log t))$ bit operations

$$e = \sum_{k=0}^{n-1} \frac{1}{k!} + \frac{1}{n!} \sum_{k=0}^{\infty} \frac{1}{(n+1) \cdots (n+k)}, \quad \frac{e}{n!} \leq 2^{-t} \text{ for } n = \frac{t + o(t)}{\log_2 t}$$

Cost of the binary splitting method: $O\left(M\left(\frac{t}{\log_2 t} \log \left(\frac{t}{\log_2 t}\right)^2\right)\right) = O(M(t \log t))$.

- $\ln(2)$ in $O(M(t \log(t)^2))$ bit operations: $\ln(2) = -\ln(1 + \xi)$ with $\xi = -\frac{1}{2}$
Application: High-precision evaluation of classical constants

- $e = \exp(1)$ with error $\leq 2^{-t}$ in $O(M(t \log t))$ bit operations

$$
e = \sum_{k=0}^{n-1} \frac{1}{k!} + \frac{1}{n!} \sum_{k=0}^{\infty} \frac{1}{(n+1) \cdots (n+k)}, \quad \left| \sum_{k=0}^{n-1} \frac{1}{k!} \right| \leq \frac{e}{n!}
$$

Cost of the binary splitting method: $O\left(M\left(\frac{t}{\log_2 t} \log \left(\frac{t}{\log_2 t}\right)^2\right)\right) = O(M(t \log t))$.

- $\ln(2)$ in $O(M(t \log(t)^2))$ bit operations: $\ln(2) = -\ln(1 + \xi)$ with $\xi = -\frac{1}{2}$

Radius of convergence = 1 $\Rightarrow$ general term = $O(2^{-k})$ $\Rightarrow$ need $O(t)$ terms.
Application: High-precision evaluation of classical constants

• $e = \exp(1)$ with error $\leq 2^{-t}$ in $O(M(t \log t))$ bit operations

$$e = \sum_{k=0}^{n-1} \frac{1}{k!} + \frac{1}{n!} \sum_{k=0}^{\infty} \frac{1}{(n+1) \cdots (n+k)}, \quad \frac{e}{n!} \leq 2^{-t} \text{ for } n = \frac{t + o(t)}{\log_2 t}$$

Cost of the binary splitting method: $O\left( M\left( \frac{t}{\log_2 t} \log \left( \frac{t}{\log_2 t} \right)^2 \right) \right) = O(M(t \log t))$.

• $\ln(2)$ in $O(M(t \log(t)^2))$ bit operations: $\ln(2) = -\ln(1 + \xi)$ with $\xi = -\frac{1}{2}$

Radius of convergence = 1 $\Rightarrow$ general term $= O(2^{-k})$ $\Rightarrow$ need $O(t)$ terms.

• $\frac{1}{\pi} = \frac{12}{c^{3/2}} \sum_{n=0}^{\infty} (-1)^n \frac{(6n)!}{(3n)! n!^3} \frac{(a n + b)}{c^{3n}}, \quad \begin{cases} a = 545140134 \\ b = 13591409 \\ c = 640320 \end{cases}$

1 hypergeometric series, 1 square root, 1 division

Used in record computations — although another algo. yields $t$ digits of $\pi$ in only $O(M(t \log t))$ bit ops

[Chudnovsky² 1987, Salamin 1976, Brent 1978]
Fast high-precision evaluation of the exponential function

[1976 Brent]

Goal: for a real number $\frac{1}{2} \leq \xi < 1$, compute $\exp(\xi)$ with error $\leq 2^{-t}$ in $\tilde{O}(t)$ bit ops.

We assume that a sufficiently accurate approximation of $\xi$, is given ($t + O(1)$ bits suffice)

Remark: can reduce to $\xi \in [1/2, 1)$ using $\exp(2x) = \exp(x)^2$. 
Fast high-precision evaluation of the exponential function

Goal: for a real number $\frac{1}{2} \leq \xi < 1$, compute $\exp(\xi)$ with error $\leq 2^{-t}$ in $\tilde{O}(t)$ bit ops.

We assume that a sufficiently accurate approximation of $\xi$ is given ($t + O(1)$ bits suffice)

Write $\xi = 0.\overline{\xi_1\xi_2\xi_3\xi_4\xi_5\xi_6\xi_7\xi_9\xi_{10}\xi_{11}\xi_{12}\xi_{13}\xi_{14}\xi_{15}\xi_{16}\xi_{17} \ldots}$

$$= m_0 + m_1 + m_2 + \cdots + m_{K-1}$$

where $\left\{ \begin{array}{l} m_k \leq 2^{-2^k+1} \\ m_k \text{ fits on } 2^k \text{ bits} \end{array} \right.$

Then $\exp(\xi) = \exp(m_0) \exp(m_1) \cdots \exp(m_{K-1})$ and $K = O(\log t)$

Remark: can reduce to $\xi \in [1/2, 1)$ using $\exp(2x) = \exp(x)^2$. 
Fast high-precision evaluation of the exponential function

[Brent 1976]

Goal: for a real number \( \frac{1}{2} \leq \xi < 1 \), compute \( \exp(\xi) \) with error \( \leq 2^{-t} \) in \( \tilde{O}(t) \) bit ops.

We assume that a sufficiently accurate approximation of \( \xi \), is given \( (t + O(1) \) bits suffice)

Write

\[ \xi = 0.\xi_1\xi_2\xi_3\xi_4\xi_5\xi_6\xi_7\xi_8\xi_9\xi_{10}\xi_{11}\xi_{12}\xi_{13}\xi_{14}\xi_{15}\xi_{16}\xi_{17} \ldots \]

\[ = m_0 + m_1 + m_2 + \cdots + m_{K-1} \]

where \( \begin{cases} m_k \leq 2^{-2^k+1} \\ m_k \text{ fits on } 2^k \text{ bits} \end{cases} \)

Then \( \exp(\xi) = \exp(m_0) \exp(m_1) \cdots \exp(m_{K-1}) \) and \( K = O(\log t) \)

**Algorithm.** Evaluate each \( m_k \) by binary splitting, then multiply.

The final multiplications cost \( O(M(t) \log t) \) in total.

Remark: can reduce to \( \xi \in [1/2, 1) \) using \( \exp(2x) = \exp(x)^2 \).
Fast high-precision exponential: analysis

\[ \xi = m_0 + m_1 + m_2 + \cdots + m_{K-1} \]

Computation of a single \( \exp(m_k) \):

where
\[
\begin{align*}
& m_k \leq 2^{-2^k + 1} \\
& m_k \text{ fits on } 2^k \text{ bits}
\end{align*}
\]
Fast high-precision exponential: analysis

\[ \xi = m_0 + m_1 + m_2 + \cdots + m_{K-1} \]

where \[ \begin{cases} m_k \leq 2^{-2^k + 1} \\ m_k \text{ fits on } 2^k \text{ bits} \end{cases} \]

Computation of a single \( \exp(m_k) \):

- Because \( m_k \leq 2^{-2^k + 1} \), only \( N = O(2^{-k} t) \) terms of the series are needed
Fast high-precision exponential: analysis

\[ \xi = m_0 + m_1 + m_2 + \cdots + m_{K-1} \]

where \( m_k \leq 2^{-2^k+1} \)

\( m_k \) fits on \( 2^k \) bits

Computation of a single \( \exp(m_k) \):

- Because \( m_k \leq 2^{-2^k+1} \), only \( N = O(2^{-k} t) \) terms of the series are needed
- Cost of binary splitting:

\[
O\left( M \left( N \left( h + \log N \right) \right) \log N \right)
\]

size of each row

size of each leaf

depth
Fast high-precision exponential: analysis

\[ \xi = m_0 + m_1 + m_2 + \cdots + m_{2^k-1} \]

where \( \left\{ \begin{array}{l}
    m_k < 2^{-2^k+1} \\
    m_k \text{ fits on } 2^k \text{ bits}
\end{array} \right. \)

Computation of a single \( \exp(m_k) \):

- Because \( m_k < 2^{-2^k+1} \), only \( N = O(2^{-k} t) \) terms of the series are needed
- Cost of binary splitting:

\[
O\left( M\left( N \left( h + \log N \right) \right) \log N \right) = O\left( M\left( 2^{-k} t \left( 2^k + \log t \right) \right) \log t \right)
\]
Fast high-precision exponential: analysis

\[ \xi = m_0 + m_1 + m_2 + \cdots + m_{K-1} \]

where \( \begin{cases} m_k \leq 2^{-2^k+1} \\ m_k \text{ fits on } 2^k \text{ bits} \end{cases} \)

Computation of a single \( \exp(m_k) \):

- Because \( m_k \leq 2^{-2^k+1} \), only \( N = O(2^{-k} t) \) terms of the series are needed
- Cost of binary splitting:

\[
O\left( M\left( \frac{N}{\text{size of each row}} \left( \frac{\text{size of each leaf} \cdot \log N}{\text{depth}} \right) \right) \right) = O\left( M\left( 2^{-k} t \left( \frac{2^k + \log t}{\text{size of each leaf} \cdot \text{depth}} \right) \right) \log t \right)
\]

\[
= O(M(t \log t + 2^{-k} t \log^2 t))
\]
Fast high-precision exponential: analysis

\[ \xi = m_0 + m_1 + m_2 + \cdots + m_{K-1} \]

where \( m_k \leq 2^{-2^k+1} \) \( m_k \) fits on \( 2^k \) bits

Computation of a single \( \exp(m_k) \):

- Because \( m_k \leq 2^{-2^k+1} \), only \( N = O(2^{-k} t) \) terms of the series are needed
- Cost of binary splitting:

\[
O\left( M\left( N \frac{h + \log N}{\text{size of each leaf}} \right) \log N \right) = O\left( M\left( 2^{-k} t \frac{2^k + \log t}{\text{size of each leaf}} \right) \log t \right) = O(M(t \log t + 2^{-k} t \log^2 t))
\]

Total: \( \sum_{k=0}^{K-1} C M(t \log t + 2^{-k} t \log^2 t) \)
Fast high-precision exponential: analysis

\[ \xi = m_0 + m_1 + m_2 + \cdots + m_{K-1} \]

where \( m_k \leq 2^{-2^k + 1} \)

\( m_k \) fits on \( 2^k \) bits

Computation of a single \( \exp(m_k) \):

- Because \( m_k \leq 2^{-2^k + 1} \), only \( N = O(2^{-k} t) \) terms of the series are needed
- Cost of binary splitting:

\[
O \left( M \left( \frac{N}{2^k} \left( h + \log N \right) \right) \log N \right) = O \left( M \left( 2^{-k} t \left( 2^k + \log t \right) \right) \log t \right) \\
= O(M \left( t \log t + 2^{-k} t \log^2 t \right))
\]

Total:

\[
\sum_{k=0}^{K-1} C M \left( t \log t + 2^{-k} t \log^2 t \right) \leq C \cdot M \left( \sum_{k=0}^{K-1} \left( t \log t + 2^{-k} t \log^2 t \right) \right)
\]
Fast high-precision exponential: analysis

\[ \xi = m_0 + m_1 + m_2 + \cdots + m_{K-1} \]

where \( m_k \leq 2^{-2^k+1} \)
\( m_k \) fits on \( 2^k \) bits

Computation of a single \( \text{exp}(m_k) \):

- Because \( m_k \leq 2^{-2^k+1} \), only \( N = O(2^{-k} t) \) terms of the series are needed
- Cost of binary splitting:

\[
O\left( M\left( \frac{N (h + \log N)}{\text{size of each row}} \right) \log N \right) = O\left( M\left( \frac{2^{-k} t (2^k + \log t)}{\text{size of each row}} \right) \log t \right) \\
= O(M(t \log t + 2^{-k} t \log^2 t))
\]

Total: \[ \sum_{k=0}^{K-1} C M(t \log t + 2^{-k} t \log^2 t) \leq C \cdot M \left( \sum_{k=0}^{K-1} (t \log t + 2^{-k} t \log^2 t) \right) = O(M(t \log(t)^2)) \]
We describe an algorithm for arbitrary-precision computation of the elementary functions (exp, log, sin, atan, etc.) which, after a cheap pre-computation, gives roughly a factor-two speedup over previous state-of-the-art algorithms at precision from a few thousand bits up to millions of bits. Following an idea of Schönhage, we perform argument reduction using Diophantine combinations of logarithms of primes; our contribution is to use a large set of primes instead of a single pair, aided by a fast algorithm to solve the associated integer relation problem. We also list new, optimized Machin-like formulas for the necessary logarithm and arctangent precomputations.
Fast high-precision evaluation of D-finite functions (sketch)  
[Chudnovsky & Chudnovsky 1987, van der Hoeven 1999]

Fix a differential operator $L$; assume that $0$ is an ordinary point.

Consider a basis $y_1, \ldots, y_r$ of analytic solutions.
Fast high-precision evaluation of D-finite functions (sketch)
[Chudnovsky & Chudnovsky 1987, van der Hoeven 1999]

Fix a differential operator $L$; assume that $0$ is an ordinary point.

Consider a basis $y_1, \ldots, y_r$ of analytic solutions.

- Suppose that the series expansion of $y_k$ converges on $\{ |\xi| < \rho \}$.

  Binary splitting $\leadsto y(\xi)$ for $|\xi| \leq \frac{1}{2}\rho$ of bit size $O(1)$ in $\tilde{O}(t)$ bit ops.
Fix a differential operator $L$; assume that $0$ is an ordinary point.

Consider a basis $y_1, \ldots, y_r$ of analytic solutions.

- Suppose that the series expansion of $y_k$ converges on $\{ |\xi| < \rho \}$.

  Binary splitting $y(\xi)$ for $|\xi| \leq \frac{1}{2} \rho$ of bit size $O(1)$ in $\tilde{O}(t)$ bit ops.

- The necessary error bounds can be computed automatically.
Fix a differential operator $L$; assume that $0$ is an ordinary point.

Consider a basis $y_1, \ldots, y_r$ of analytic solutions.

- Suppose that the series expansion of $y_k$ converges on $\{ |\xi| < \rho \}$.

  Binary splitting $\sim y(\xi)$ for $|\xi| \leq \frac{1}{2} \rho$ of bit size $O(1)$ in $\tilde{O}(t)$ bit ops.

- The necessary error bounds can be computed automatically.

- Derivatives have the same radius of convergence, are still D-finite.

  $\sim (y(\xi), y'(\xi), \ldots, y^{(r-1)}(\xi))$ in $\tilde{O}(t)$ ops
Fix a differential operator $L$; assume that 0 is an ordinary point.

Consider a basis $y_1, \ldots, y_r$ of analytic solutions.

- Suppose that the series expansion of $y_k$ converges on $\{ |\xi| < \rho \}$.
  
  Binary splitting $\leadsto y(\xi)$ for $|\xi| \leq \frac{1}{2} \rho$ of bit size $O(1)$ in $\tilde{O}(t)$ bit ops.

- The necessary error bounds can be computed automatically.

- Derivatives have the same radius of convergence, are still D-finite.
  
  $\leadsto (y(\xi), y'(\xi), \ldots, y^{(r-1)}(\xi))$ in $\tilde{O}(t)$ ops

- We can do that for $y_1, \ldots, y_r \leadsto \begin{pmatrix} y_1(\xi) & \cdots & y_r(\xi) \\ \vdots & \ddots & \vdots \\ y_1^{(r-1)}(\xi) & \cdots & y_r^{(r-1)}(\xi) \end{pmatrix}$ in $\tilde{O}(t)$ ops
Fast high-precision evaluation of D-finite functions (sketch)

[Chudnovsky & Chudnovsky 1987, van der Hoeven 1999]

- By multiplying matrices
  \[
  \begin{pmatrix}
  y_1(\xi) & \cdots & y_r(\xi) \\
  \vdots & & \vdots \\
  y_1^{(r-1)}(\xi) & \cdots & y_r^{(r-1)}(\xi)
  \end{pmatrix},
  \]
  we can evaluate the (analytic continuation of) the solutions outside the disk \(|\xi| < \rho|.

(Can be stated rigorously with more care.)
Fast high-precision evaluation of D-finite functions (sketch)

[Chudnovsky & Chudnovsky 1987, van der Hoeven 1999]

By multiplying matrices

\[
\begin{pmatrix}
    y_1(\xi) & \cdots & y_r(\xi) \\
    \vdots & \ddots & \vdots \\
    y_1^{(r-1)}(\xi) & \cdots & y_r^{(r-1)}(\xi)
\end{pmatrix},
\]

we can evaluate the (analytic continuation of) the solutions outside the disk \(|\xi| < \rho|.

By multiplying the same matrices for steps corresponding to a decomposition

\[
\xi = 0.\xi_1\xi_2\xi_3\xi_4\xi_5\xi_6\xi_7\xi_8\xi_9\xi_{10}\xi_{11}\xi_{12}\xi_{13}\xi_{14}\xi_{15}\xi_{16}\xi_{17}\ldots
\]

we can evaluate the solutions at complex points of bit size \(t\) in \(\tilde{O}(t)\) ops.
Fast high-precision evaluation of D-finite functions (sketch)

[Chudnovsky & Chudnovsky 1987, van der Hoeven 1999]

- By multiplying matrices
  \[
  \begin{pmatrix}
  y_1(\xi) & \cdots & y_r(\xi) \\
  \vdots & \ddots & \vdots \\
  y_1^{(r-1)}(\xi) & \cdots & y_r^{(r-1)}(\xi)
  \end{pmatrix},
  \]
  we can evaluate the (analytic continuation of) the solutions outside the disk \(|\xi| < \rho|.

- By multiplying the same matrices for steps corresponding to a decomposition
  \[
  \xi = 0, \xi_1, \xi_2, \xi_3, \xi_4, \xi_5, \xi_6, \xi_7, \xi_8, \xi_9, \xi_{10}, \xi_{11}, \xi_{12}, \xi_{13}, \xi_{14}, \xi_{15}, \xi_{16}, \xi_{17}, \ldots
  \]
  we can evaluate the solutions at complex points of bit size \(t\) in \(\tilde{O}(t)\) ops.

Pseudo-theorem: “one can evaluate a \textbf{fixed} D-finite function at a \textbf{fixed} point \(\xi \in \mathbb{C}\)
with an error \(\leq 2^{-t}\) in \(\tilde{O}(t)\) bit operations”.

(Can be stated rigorously with more care.)
Solutions of linear differential equations
Introduction

In this lecture, $\mathbb{K}$ is an effective field of characteristic zero.

**Problem.** Given a differential equation

$$a_r(x) y^{(r)} + \cdots + a_1(x) y'(x) + a_0(x) y(x) = 0, \quad a_i \in \mathbb{K}[x],$$

compute a basis of...

a) formal series solutions $y \in \mathbb{K}[[x]]$,

b) polynomial solutions $y \in \mathbb{K}[x]$,

c) rational solutions $y \in \mathbb{K}(x)$.  

1 Series solutions
Singular points

\[ a_r y^{(r)} + \cdots + a_1 y' + a_0 y = 0 \]

**Definition.** A point \( \xi \in K \) is called

- an **ordinary point** of \( L \) if \( a_r(\xi) \neq 0 \),
- a **singular point** of \( L \) otherwise.

Recall: if \( K = \mathbb{C} \) and \( \xi \) is an ordinary point, the space of analytic solutions at \( \xi \) has \( \dim r \) and a solution \( y \) is characterized by \( y(\xi), \ldots, y^{(r-1)}(\xi) \).

Corollary: if \( K = \mathbb{C} \) and \( 0 \) is an ordinary point, \( r \) linearly independent solutions in \( \mathbb{C}[[x]] \).

- Generalize to arbitrary \( K \) of char zero
- Look at what happens at singular points (how to compute series solutions?)
**Definition.** We denote

$$\mathbb{K}((x)) = \bigcup_{n_0 \in \mathbb{Z}} \left\{ \sum_{n \geq n_0} u_n x^n \mid u_n \in \mathbb{K} \right\}.$$ 

The elements of $\mathbb{K}((x))$ are called **formal Laurent series**.

- **Warning:** $\mathbb{C}((x)) \neq$ Laurent series from analysis (even when convergent).
  In complex analysis, Laurent series are double-sided: $\sum_{n \in \mathbb{Z}} u_n x^n$.
  But formal double-sided series do not form a ring!

- $\mathbb{K}((x))$ is the field of fractions of $\mathbb{K}[[x]]$.

- Rational functions in $\mathbb{K}(x)$ can be expanded in formal Laurent series at any point of $\mathbb{K}$. 
Laurent series solutions and recurrences

For \( y \in K((x)) \), define \((y_n)_{n \in \mathbb{Z}}\) by \( y(x) = \sum_{n \in \mathbb{Z}} y_n x^n \). (Thus \( y_n = 0 \) eventually as \( n \to -\infty \).)

Let \( \theta : K((x)) \to K((x)) \) and \( S : K^\mathbb{Z} \to K^\mathbb{Z} \)
\[ y \mapsto x y'(x) \quad \text{and} \quad (y_n) \mapsto (y_{n+1}) \]

**Proposition.** The series \( y \in K((x)) \) is solution to
\[ \tilde{a}_r(x) \theta^r + \cdots + \tilde{a}_1(x) \theta + \tilde{a}_0(x) \]
if and only if the sequence \((y_n)_{n \in \mathbb{Z}}\) is solution to
\[ R = \tilde{a}_r(S^{-1}) n^r + \cdots + \tilde{a}_1(S^{-1}) n + \tilde{a}_0(S^{-1}). \]

**Proof.** Substitute and compare coefficients.

Write \( S^\delta R = q_0(n) - q_1(n) S^{-1} - \cdots - q_s(n) S^{-s} \) for \( \delta \) such that \( q_0 \neq 0 \).
Solutions of singular recurrences

\[ \forall n \in \mathbb{Z}, \quad q_0(n) y_n - q_1(n) y_{n-1} - \cdots - q_s(n) y_{n-s} = 0 \]

Recall: for a nonsingular recurrence of order \( s \), solutions are determined by \( s \) initial values, can be computed efficiently.
Solutions of singular recurrences

\[ \forall n \in \mathbb{Z}, \quad q_0(n) y_n - q_1(n) y_{n-1} - \cdots - q_s(n) y_{n-s} = 0 \]

Recall: for a nonsingular recurrence of order \( s \), solutions are determined by \( s \) initial values, can be computed efficiently.

At a singular index (\( = \) root of \( q_0 \)):

\[
\begin{align*}
\vdots \\
q_0(n - 2) y_{n-2} &= q_1(n - 2) y_{n-3} + \cdots + q_s(n - 2) y_{n-2-s} \\
q_0(n - 1) y_{n-1} &= q_1(n - 1) y_{n-2} + \cdots + q_s(n - 1) y_{n-1-s} \\
0 y_n &= q_1(n) y_{n-1} + \cdots + q_s(n) y_{n-s} \\
q_0(n + 1) y_{n+1} &= q_1(n + 1) y_n + \cdots + q_s(n + 1) y_{n+1-s} \\
q_0(n + 2) y_{n+2} &= q_1(n + 2) y_{n+1} + \cdots + q_s(n + 2) y_{n+2-s} \\
\vdots
\end{align*}
\]

For a solution \((y_n)_{n \in \mathbb{Z}}\) with \( y_N \neq 0 \) to exist, \( N \) must be a root of \( q_0 \).

A partial solution \((y_n)_{n < N}\) with \( N > \max \{\text{roots of } q_0 \text{ in } \mathbb{Z}\}\) extends to a unique solution \((y_n)_{n \in \mathbb{Z}}\).
Solutions of singular recurrences

\[ \forall n \in \mathbb{Z}, \quad q_0(n) y_n - q_1(n) y_{n-1} - \cdots - q_s(n) y_{n-s} = 0 \]

Recall: for a nonsingular recurrence of order \( s \), solutions are determined by \( s \) initial values, can be computed efficiently.

At a singular index (= root of \( q_0 \)):

\[
\begin{align*}
q_0(n - 2) y_{n-2} &= q_1(n - 2) y_{n-3} + \cdots + q_s(n - 2) y_{n-2-s} \\
q_0(n - 1) y_{n-1} &= q_1(n - 1) y_{n-2} + \cdots + q_s(n - 1) y_{n-1-s} \\
0 y_n &= q_1(n) y_{n-1} + \cdots + q_s(n) y_{n-s} \\
q_0(n + 1) y_{n+1} &= q_1(n + 1) y_n + \cdots + q_s(n + 1) y_{n+1-s} \\
q_0(n + 2) y_{n+2} &= q_1(n + 2) y_{n+1} + \cdots + q_s(n + 2) y_{n+2-s} \\
\vdots
\end{align*}
\]

\rightarrow \text{free choice of} \ y_n

\rightarrow \text{extra linear constraint on} \ (y_{n-s}, \ldots, y_{n-1})
Solutions of singular recurrences

\[ \forall n \in \mathbb{Z}, \quad q_0(n) y_n - q_1(n) y_{n-1} - \cdots - q_s(n) y_{n-s} = 0 \]

Recall: for a nonsingular recurrence of order \( s \), solutions are determined by \( s \) initial values, can be computed efficiently.

At a **singular index** (= root of \( q_0 \):

\[
\begin{align*}
q_0(n-2) y_{n-2} &= q_1(n-2) y_{n-3} + \cdots + q_s(n-2) y_{n-2-s} \\
q_0(n-1) y_{n-1} &= q_1(n-1) y_{n-2} + \cdots + q_s(n-1) y_{n-1-s} \\
0 y_n &= q_1(n) y_{n-1} + \cdots + q_s(n) y_{n-s} \\
q_0(n+1) y_{n+1} &= q_1(n+1) y_{n} + \cdots + q_s(n+1) y_{n+1-s} \\
q_0(n+2) y_{n+2} &= q_1(n+2) y_{n+1} + \cdots + q_s(n+2) y_{n+2-s} \\
\vdots
\end{align*}
\]

→ free choice of \( y_n \)

→ extra linear constraint on \( (y_{n-s}, \ldots, y_{n-1}) \)

- For a solution \( (y_n)_{n \in \mathbb{Z}} = (\ldots, 0, 0, 0, y_N, y_{N+1}, y_{N+2}, \ldots) \) with \( y_N \neq 0 \) to exist, \( N \) must be a root of \( q_0 \).

- A partial solution \( (y_n)_{n \leq N} \) with \( N \geq \max \{ \text{roots of } q_0 \text{ in } \mathbb{Z} \} \) extends to a unique solution \( (y_n)_{n \in \mathbb{Z}} \).
**Lemma.** The index of the first nonzero term in a solution \((y_n)_{n \in \mathbb{Z}}\) to \(q_0(n) y_n = q_1(n) y_{n-1} + \cdots + q_s(n) y_{n-s}\) that is ultimately zero as \(n \to -\infty\) is a root of \(q_0\).

**Exercise.** Find the dimension of the space of solutions of

\[(n - 1) (n - 2) u_n = u_{n-1} + (n - 2) u_{n-2}\]

that are ultimately zero as \(n \to -\infty\).
An exercise

Lemma. The index of the first nonzero term in a solution \((y_n)_{n \in \mathbb{Z}}\) to \(q_0(n) y_n = q_1(n) y_{n-1} + \cdots + q_s(n) y_{n-s}\) that is ultimately zero as \(n \to -\infty\) is a root of \(q_0\).

Exercise. Find the dimension of the space of solutions of

\[
(n-1)(n-2) u_n = u_{n-1} + (n-2) u_{n-2}
\]

that are ultimately zero as \(n \to -\infty\).

\[
\begin{align*}
(n=0) & \quad 2u_0 = u_{-1} - 2u_{-2} \\
(n=1) & \quad 0 = u_0 - u_{-1} \\
(n=2) & \quad 0 = u_1 + 0u_0 \\
(n=3) & \quad 2u_3 = u_2 + u_1 \\
& \quad \vdots
\end{align*}
\]
An exercise

**Lemma.** The index of the first nonzero term in a solution \((y_n)_{n \in \mathbb{Z}}\) to \(q_0(n) y_n = q_1(n) y_{n-1} + \cdots + q_s(n) y_{n-s}\) that is ultimately zero as \(n \to -\infty\) is a root of \(q_0\).

**Exercise.** Find the dimension of the space of solutions of

\[(n-1)(n-2)u_n = u_{n-1} + (n-2)u_{n-2}\]

that are ultimately zero as \(n \to -\infty\).

**Solution.** The first nonzero term of such a solution must be \(u_1\) or \(u_2\).

Evaluating the equation at \(n = 2\) yields \(u_1 = 0\).

In contrast, starting from any value of \(u_2\), one can define a solution with support \(\subseteq \{2, 3, \ldots\}\).

The dimension is 1 — less than the number of integer roots of \(q_0\)!
The indicial polynomial

\[ L = \tilde{a}_r(x) \theta^r + \cdots + \tilde{a}_0(x) \]

\[ R = \tilde{a}_r(S^{-1}) n^r + \cdots + \tilde{a}_0(S^{-1}) 
= S^{-\delta} \left( q_0(n) - \cdots - q_s(n) S^{-s} \right) \]

**Proposition.** For any solution \( y \in K((x)) \) of \( L(y) = 0 \), the valuation of \( y \) is a root of \( q_0 \).

**Corollary.** The space of formal Laurent series solutions of \( L \) has dimension \( \leq r \).

All this also holds for solutions in \( K[[x]] \).
The indicial polynomial

\[ L = \tilde{a}_r(x) \theta^r + \cdots + \tilde{a}_0(x) \quad \rightarrow \quad R = \tilde{a}_r(S^{-1}) n^r + \cdots + \tilde{a}_0(S^{-1}) \]
\[ = S^{-\delta} (q_0(n) - \cdots - q_s(n) S^{-s}) \]

**Proposition.** For any solution \( y \in \mathbb{K}((x)) \) of \( L(y) = 0 \), the valuation of \( y \) is a root of \( q_0 \).

**Corollary.** The space of formal Laurent series solutions of \( L \) has dimension \( \leq r \).

**Proof.** Echelonize a basis and note that \( \deg q_0 \leq r \).

All this also holds for solutions in \( \mathbb{K}[[x]] \).
The indicial polynomial

\[ L = \tilde{a}_r(x) \theta^r + \cdots + \tilde{a}_0(x) \quad \rightarrow \quad R = \tilde{a}_r(S^{-1}) n^r + \cdots + \tilde{a}_0(S^{-1}) = S^{-\delta} (q_0(n) - \cdots - q_s(n) S^{-s}) \]

**Proposition.** For any solution \( y \in \mathbb{K}((x)) \) of \( L(y) = 0 \), the valuation of \( y \) is a root of \( q_0 \).

**Corollary.** The space of formal Laurent series solutions of \( L \) has dimension \( \leq r \).

**Proof.** Echelonize a basis and note that \( \deg q_0 \leq r \).

**Remark.** \( \deg q_0 \) can be \( < r \), and the dimension can be \( < \#\{\text{roots of } q_0 \text{ in } \mathbb{Z}\} \).

All this also holds for solutions in \( \mathbb{K}[[x]] \).
The indicial polynomial

\[ L = \tilde{\alpha}_r(x) \theta^r + \cdots + \tilde{\alpha}_0(x) \quad \rightarrow \quad R = \tilde{\alpha}_r(S^{-1}) n^r + \cdots + \tilde{\alpha}_0(S^{-1}) \]

\[ = S^{-\delta} (q_0(n) - \cdots - q_s(n) S^{-s}) \]

**Proposition.** For any solution \( y \in K((x)) \) of \( L(y) = 0 \), the valuation of \( y \) is a root of \( q_0 \).

**Corollary.** The space of formal Laurent series solutions of \( L \) has dimension \( \leq r \).

**Proof.** Echelonize a basis and note that \( \deg q_0 \leq r \).

**Remark.** \( \deg q_0 \) can be \( < r \), and the dimension can be \( < \# \{ \text{roots of } q_0 \text{ in } \mathbb{Z} \} \).

All this also holds for solutions in \( K[[x]] \).

**Definition.** The polynomial \( q_0 \) is called the **indicial polynomial** of \( L \) at 0.

The polynomial obtained in the same way after \( x \leftarrow \xi + x \) is called the indicial polynomial at \( \xi \).
Formal series solutions of differential equations

\[ L = \tilde{a}_r(x) \theta^r + \cdots + \tilde{a}_0(x) \quad (\text{deg} \tilde{a}_i < d) \]

\[ \rightarrow \quad R = \tilde{a}_r(S^{-1}) n^r + \cdots + \tilde{a}_0(S^{-1}) \]
\[ = S^{-\delta} \left( q_0(n) - \cdots - q_s(n) S^{-s} \right) \]

Goal: find a basis of the solution space of \( L \) in \( \mathbb{K}((x)) \).
Formal series solutions of differential equations

\[ L = \tilde{a}_r(x) \, \theta^r + \cdots + \tilde{a}_0(x) \quad \rightarrow \quad R = \tilde{a}_r(S^{-1}) \, n^r + \cdots + \tilde{a}_0(S^{-1}) = S^{-\delta} \left( q_0(n) - \cdots - q_s(n) \, S^{-s} \right) \]

Goal: find a basis of the solution space of \( L \) in \( \mathbb{K}((x)) \).

**Idea.** Let \( \lambda, \mu \) be the smallest/largest root of \( q_0 \) in \( \mathbb{Z} \).
Make an ansatz \( y(x) = y_\lambda x^\lambda + \cdots + y_\mu x^\mu + O(x^{\mu+1}) \), plug into the equation:

For solutions in \( \mathbb{K}[[x]] \): same method, with \( \lambda, \mu \in \mathbb{N} \).
Formal series solutions of differential equations

\[ L = \tilde{a}_r(x) \theta^r + \cdots + \tilde{a}_0(x) \quad \rightarrow \quad R = \tilde{a}_r(S^{-1}) n^r + \cdots + \tilde{a}_0(S^{-1}) = S^{-\delta} (q_0(n) - \cdots - q_s(n) S^{-s}) \]

Goal: find a basis of the solution space of \( L \) in \( \mathbb{K}((x)) \).

**Idea.** Let \( \lambda, \mu \) be the smallest/largest root of \( q_0 \) in \( \mathbb{Z} \).

Make an ansatz \( y(x) = y_\lambda x^\lambda + \cdots + y_\mu x^\mu + O(x^{\mu+1}) \), plug into the equation:

\[ L(y)(x) = [\ldots] x^\lambda + [\ldots] x^{\lambda+1} + [\ldots] x^{\mu+d-1} + O(x^{\mu+d}) \]

linear expressions in \( y_\lambda, \ldots, y_\mu \)

partial solutions are guaranteed to extend so as to make this zero

For solutions in \( \mathbb{K}[[x]] \): same method, with \( \lambda, \mu \in \mathbb{N} \).
Formal series solutions of differential equations

\[ L = \tilde{a}_r(x) \theta^r + \cdots + \tilde{a}_0(x) \quad \text{→} \quad R = \tilde{a}_r(S^{-1}) n^r + \cdots + \tilde{a}_0(S^{-1}) = S^{-\delta} (q_0(n) - \cdots - q_s(n) S^{-s}) \]

Goal: find a basis of the solution space of \( L \) in \( K((x)) \).

**Idea.** Let \( \lambda, \mu \) be the smallest/largest root of \( q_0 \) in \( \mathbb{Z} \).

Make an ansatz \( y(x) = y_\lambda x^\lambda + \cdots + y_\mu x^\mu + O(x^{\mu+1}) \), plug into the equation:

\[
L(y)(x) = \left[\cdots\right] x^\lambda + \left[\cdots\right] x^{\lambda+1} + \left[\cdots\right] x^{\mu+d-1} + O(x^{\mu+d})
\]

linear expressions in \( y_\lambda, \ldots, y_\mu \)

partial solutions are guaranteed to extend so as to make this zero

For solutions in \( K[[x]] \): same method, with \( \lambda, \mu \in \mathbb{N} \).

**Limitation.** The dimension \( \mu - \lambda \) can be large! (exponential in the bit size of the input)

For example the solutions of \( x^2 y''(x) = 999999 x y'(x) \) are spanned by \( 1 \) and \( x^{10^6} \).
Formal series solutions: a better algorithm

**Algorithm (sketch).** *Input:* L, N  
*Output:* a basis of $\text{Sol}(L, \mathbb{K}[[x]])$ to prec. N

1. Convert L to a recurrence. Let $q_0$ be the indicial polynomial.
2. Let $\lambda_1 < \lambda_2 < \cdots < \lambda_m$ be the roots of $q_0$ in $\mathbb{N}$. \[ m \leq \text{diff. eq. order} \]
3. For $n = \lambda_1, \lambda_1 + 1, \ldots, \max(N, \lambda_m)$:
   - a. If $n = \lambda_k$ for some $k$:
     i. Set $u_n$ to a new indeterminate $C_k$.
     ii. Evaluate the recurrence at $n$, record the resulting relation on previous $C_k$.
   - b. Otherwise compute $u_n$ using the recurrence.
4. Solve the linear system on $C_1, \ldots, C_m$ consisting of the collected relations.

Even better: use fast algorithms (baby steps-giant steps, binary splitting) to “jump” from one singular index to the next.

Remark: the $C_k$ that remain free after solving play the role of generalized initial values.
The case of ordinary points

**Proposition.** If 0 is an ordinary point,

- the space of power series solutions has dimension exactly \( r \), and
- there exists a basis of solutions with valuations \( 0, 1, \ldots, r - 1 \).
The case of ordinary points

**Proposition.** If 0 is an ordinary point,

- the space of power series solutions has dimension exactly $r$, and
- there exists a basis of solutions with valuations $0, 1, \ldots, r - 1$.

**Proof sketch.** One can check that the recurrence takes the form

$$a_r(0) n (n - 1) \cdots (n - r + 1) u_n = [\text{poly}(n)] (n - 1) \cdots (n - r + 1) u_{n-1}$$

$$+ [\text{poly}(n)] (n - 2) \cdots (n - r + 1) u_{n-2}$$

$$+ \cdots$$

$$+ [\text{poly}(n)] (n - r + 1) u_{n-r+1}$$

$$+ \cdots$$
The case of ordinary points

**Proposition.** If $0$ is an ordinary point,

- the space of power series solutions has dimension exactly $r$, and
- there exists a basis of solutions with valuations $0, 1, \ldots, r - 1$.

**Proof sketch.** One can check that the recurrence takes the form

$$a_r(0) n (n - 1) \cdots (n - r + 1) u_n = [\text{poly}(n)] (n - 1) \cdots (n - r + 1) u_{n-1}$$

$$+ [\text{poly}(n)] (n - 2) \cdots (n - r + 1) u_{n-2}$$

$$+ \cdots$$

$$+ [\text{poly}(n)] (n - r + 1) u_{n-r+1}$$

$$+ \cdots$$

- Since $a_r(0) \neq 0$, the indicial polynomial is $a_r(0) n (n - 1) \cdots (n - r + 1)$. This shows that the only possible valuations are $0, \ldots, r - 1$. (Gen. ini. val. = usual ini. val.)

\[ \square \]
The case of ordinary points

**Proposition.** If 0 is an ordinary point,

- the space of power series solutions has dimension exactly $r$, and
- there exists a basis of solutions with valuations $0, 1, \ldots, r - 1$.

**Proof sketch.** One can check that the recurrence takes the form

$$a_r(0) n (n - 1) \cdots (n - r + 1) u_n = [\text{poly}(n)] (n - 1) \cdots (n - r + 1) u_{n-1} + [\text{poly}(n)] (n - 2) \cdots (n - r + 1) u_{n-2} + \cdots + [\text{poly}(n)] (n - r + 1) u_{n-r+1} + \cdots$$

- Since $a_r(0) \neq 0$, the indicial polynomial is $a_r(0) n (n - 1) \cdots (n - r + 1)$. This shows that the only possible valuations are $0, \ldots, r - 1$. (Gen. ini. val. = usual ini. val.)
- All partial solutions $(y_0, \ldots, y_k)$ for $k \leq r - 1$ extend thanks to the shape of the rhs. □
A paradox

We have seen that:

- the space of solutions in $\mathbb{K}[[x]]$ of a nonsingular ODE of order $r$ has dimension $r$;
- the space of solutions in $\mathbb{K}^\mathbb{N}$ of a nonsingular recurrence of order $s$ has dimension $s$;
- the recurrence associated to an ODE of order $r$ and degree $d$ has order $\leq d + r$, typically $\neq r$. 

(It dependson the integer rootsof the leading coecient, and hence on the degree of the recurrence.)
A paradox

We have seen that:

- the space of solutions in $\mathbb{K}[[x]]$ of a nonsingular ODE of order $r$ has dimension $r$;
- the space of solutions in $\mathbb{K}^\mathbb{N}$ of a nonsingular recurrence of order $s$ has dimension $s$;
- the recurrence associated to an ODE of order $r$ and degree $d$ has order $\leq d + r$, typically $\neq r$.

However, solutions in $\mathbb{K}[[x]]$ of the ODE correspond to solutions $(y_n)_{n \in \mathbb{Z}}$ with $y_n = 0$ for $n < 0$ of the recurrence.

The dimension of the latter solution space is unrelated to the order!

(It depends on the integer roots of the leading coefficient, and hence on the degree of the recurrence.)
Generalized series solutions

- When $q_0(\lambda) = 0$ for some $\lambda \notin \mathbb{Z}$, look for solutions $x^\lambda f(x)$ with $f(x) \in \mathbb{K}[[x]]$. (Everything works as before.)
Generalized series solutions

- When \( q_0(\lambda) = 0 \) for some \( \lambda \notin \mathbb{Z} \), look for solutions \( x^\lambda f(x) \) with \( f(x) \in \mathbb{K}[[x]] \). (Everything works as before.)

- To recover \( \deg q_0 \) linearly independent solutions in all cases, consider solutions \( x^\lambda \left( f_0(x) + f_1(x) \log x + \cdots + f_{r-1}(x) \log(x)^{r-1} \right) \). (~ same algos)
Generalized series solutions

- When $q_0(\lambda) = 0$ for some $\lambda \notin \mathbb{Z}$, look for solutions $x^\lambda f(x)$ with $f(x) \in K[[x]]$. (Everything works as before.)

- To recover $\deg q_0$ linearly independent solutions in all cases, consider solutions $x^\lambda (f_0(x) + f_1(x) \log x + \cdots + f_{r-1}(x) \log(x)^{r-1})$. (~same algos)

**Definition.** A singular point where the indicial polynomial has degree $r$ is called a **regular singular point**.
Generalized series solutions

- When $q_0(\lambda) = 0$ for some $\lambda \notin \mathbb{Z}$, look for solutions $x^\lambda f(x)$ with $f(x) \in \mathbb{K}[[x]]$. (Everything works as before.)

- To recover $\deg q_0$ linearly independent solutions in all cases, consider solutions $x^\lambda (f_0(x) + f_1(x) \log x + \cdots + f_{r-1}(x) \log(x)^{r-1})$. (~ same algos)

**Definition.** A singular point where the indicial polynomial has degree $r$ is called a **regular singular point**.

- When $\deg q_0 < r$, many new phenomena.

**Theorem.** Any linear differential equation of order $r$ with coefficients in $\mathbb{K}((x))$ admits $r$ linearly independent formal solutions of the form

$$\exp \left( a_1 x^{-1/p} + \cdots + a_\ell x^{-\ell/p} \right) x^\lambda \left( f_0(x^{1/p}) + \cdots + f_{r-1}(x^{1/p}) \log(x)^{r-1} \right)$$

where $p \in \mathbb{N}$, $a_i, \lambda \in \overline{\mathbb{K}}$, and $f_j \in \overline{\mathbb{K}}[[x]]$. 
2 Polynomial and rational solutions
Introduction

\[ a_r y^{(r)} + \cdots + a_1(x) y' + a_0 y = 0 \]
\[ \text{deg}(a_i) < d \]

Suppose we had a bound \( D \) on the degrees of polynomial solutions.

Make an ansatz: \( y(x) = c_0 + c_1 x + \cdots + c_{D-1} x^{D-1} \),

\[ a_r(x) y^{(r)} + \cdots + a_0(x) y(x) = [\ldots] + [\ldots] x + [\ldots] x^{d+D-2} \]

linear expressions in \( c_0, \ldots, c_{D-1} \)

\( \rightarrow D + d - 1 \) linear equations in \( D \) unknowns.
Introduction

\[ a_r y^{(r)} + \cdots + a_1(x) y' + a_0 y = 0 \]
\[ \text{deg}(a_i) < d \]

Suppose we had a bound \( D \) on the degrees of polynomial solutions.

Make an ansatz:
\[ y(x) = c_0 + c_1 x + \cdots + c_{D-1} x^{D-1}, \]
\[ a_r(x) y^{(r)} + \cdots + a_0(x) y(x) = [\ldots] + [\ldots] x + [\ldots] x^{d+D-2} \]

\[ \text{linear expressions in } c_0, \ldots, c_{D-1} \]

\[ \rightarrow D + d - 1 \text{ linear equations in } D \text{ unknowns.} \]

Questions:  
- Compute the degree bound
- Do better than \( D^\theta \)
Finite-support solutions of recurrences

A polynomial solution is just a power series solution that terminates a solution with finite support of the associated recurrence.

Consider the recurrence

\[ \forall n \in \mathbb{Z}, \quad b_s(n) y_{n+s} + \cdots + b_1(n) y_{n+1} + b_0(n) y_n = 0. \]

For a solution

\[ (y_n)_{n \in \mathbb{Z}} = (\ldots, y_{N-2}, y_{N-1}, y_N, 0, 0, 0, \ldots) \quad \text{with } y_N \neq 0 \]

to exist, \( N \) must be a root of \( b_0 \).
Finite-support solutions of recurrences

A polynomial solution is just a power series solution that terminates a solution with finite support of the associated recurrence.

Consider the recurrence
\[ \forall n \in \mathbb{Z}, \quad b_s(n) y_{n+s} + \cdots + b_1(n) y_{n+1} + b_0(n) y_n = 0. \]

For a solution
\[ (y_n)_{n \in \mathbb{Z}} = (\ldots, y_{N-2}, y_{N-1}, y_N, 0, 0, 0, \ldots) \quad \text{with } y_N \neq 0 \]
to exist, \( N \) must be a root of \( b_0 \).

If \( R = S^{-\delta} \left( q_0(n) - q_1(n) S^{-1} - \cdots \right) \) with \( q_0 \neq 0 \)
\[ = S^\gamma \left( b_0(n) + b_1(n) S + \cdots \right) \] with \( b_0 \neq 0 \),
for a solution \((y_n)_{n \in \mathbb{Z}} = (\ldots, 0, 0, y_\ell, y_{\ell+1} \ldots, y_{h-1}, y_h, 0, 0, 0, \ldots)\) with \( y_\ell, y_h \neq 0 \) to exist, one must have \( q_0(\ell) = b_0(h) = 0. \)
**Degree bounds**

\[
L = \tilde{a}_r(x) \theta^r + \cdots + \tilde{a}_0(x) \quad \rightarrow \quad R = \tilde{a}_r(S^{-1}) n^r + \cdots + \tilde{a}_0(S^{-1})
\]

\[
= S^\gamma (b_0(n) + \cdots - b_s(n) S^s) \quad \gamma \text{ such that } b_0 \neq 0
\]

**Definition.** The polynomial \( b_0 \) is called the **indicial polynomial at infinity** of \( L \).

One can check that it is the indicial polynomial at 0 of the equation obtained by \( x \leftarrow x^{-1} \).

**Proposition.** For any solution \( y \in \mathbb{K}[x] \) of \( L(y) = 0 \), the degree of \( y \) is a root of \( b_0 \).

Remark: degrees of solutions can be large (and polynomial solutions can be dense). Cf. previous example about coefficients of \( (1 + x)^{2N} (1 + x + x^2)^N \).
Algorithm. *Input:* $L \in \mathbb{K}[x]\langle \theta \rangle$, with 0 ordinary

*Output:* a basis of $\text{Sol}(L, \mathbb{K}[x])$

1. Compute the indicial polynomial at $\infty$ of $L$.
   Let $D$ be its largest root in $\mathbb{N}$ if any; return $\emptyset$ if there is none.

2. Compute a basis $y_1, \ldots, y_r$ of solutions in $\mathbb{K}[x]$, to order $D + \max(r, d)$.

3. Solve

$$
\begin{pmatrix}
  c_1 & \cdots & c_r
\end{pmatrix}
\begin{pmatrix}
  y_{D+1}^{[1]} & \cdots & y_{D+d}^{[1]} \\
  \vdots & & \vdots \\
  y_{D+1}^{[r]} & \cdots & y_{D+d}^{[r]}
\end{pmatrix} = 0.
$$

4. Return $\{ \sum_i c_i y_i^{[i]} \mid (c_1, \ldots, c_r) \in \text{a basis of solutions of this system} \}$. 
Algorithm. \textit{Input:} \( L \in \mathbb{K}[x][\theta] \), with \( 0 \) ordinary \hspace{1cm} \textit{Output:} a basis of \( \text{Sol}(L, \mathbb{K}[x]) \)

1. Compute the indicial polynomial at \( \infty \) of \( L \). 
   Let \( D \) be its largest root in \( \mathbb{N} \) if any; return \( \emptyset \) if there is none.

2. Compute a basis \( y_1, \ldots, y_r \) of solutions in \( \mathbb{K}[[x]] \), to order \( D + \max(r, d) \).

3. Solve

\[
\begin{pmatrix}
\cdots \\
y_{D+1}^{[1]} & \cdots & y_{D+d}^{[1]} \\
\vdots & & \vdots \\
y_{D+1}^{[r]} & \cdots & y_{D+d}^{[r]}
\end{pmatrix}
= 0.
\]

4. Return \( \{ \sum_i c_i y^{[i]} | (c_1, \ldots, c_r) \in \text{a basis of solutions of this system} \} \).

Proof. By the degree bound, any polynomial solution \( y \) has \( y_{D+1} = \cdots = y_{D+d} = 0 \).

Conversely, the recurrence associated to \( L \) has order \( d \) and (ordinary point) no singular indices \( \geq r \), so \( y_{D+1} = \cdots = y_{D+d} = 0 \) implies \( y_n = 0 \) for all \( n > D \).

If \( 0 \) is a singular point, shift \( x \) (or take into account solutions of valuation \( > D \)). \( \square \)
Algorithm. Input: \( L \in \mathbb{K}[x] \langle \theta \rangle \), with 0 ordinary \( \text{Output: a basis of } \text{Sol}(L, \mathbb{K}[x]) \)

1. Compute the indicial polynomial at \( \infty \) of \( L \).
   Let \( D \) be its largest root in \( \mathbb{N} \) if any; return \( \emptyset \) if there is none.

2. Compute a basis \( y_1, \ldots, y_r \) of solutions in \( \mathbb{K}[[x]] \), to order \( D + \max (r, d) \).

3. Solve

\[
\begin{pmatrix}
    c_1 & \cdots & c_r \\
    y_{D+1}^{[1]} & \cdots & y_{D+d}^{[1]} \\
    \vdots & & \vdots \\
    y_{D+1}^{[r]} & \cdots & y_{D+d}^{[r]}
\end{pmatrix}
= 0.
\]

\( r = \text{ord}(L), \ d = \text{deg}(L) \)

4. Return \( \left\{ \sum_i c_i y_i^{[i]} \mid (c_1, \ldots, c_r) \in \text{a basis of solutions of this system} \right\} \).

Quick existence check for polynomial solutions when \( \mathbb{K} = \mathbb{Q} \):
use the baby steps-giant steps method to compute \( y_i^{[j]} \mod p \) for some prime \( p \).

More generally, one can represent the polynomial solutions by rec. + ini. cond. and use fast algorithms for computing some terms only.
Rational solutions of differential equations

\[ a_r y^{(r)} + \cdots + a_1(x) y' + a_0 y = 0 \] \hspace{1cm} (D)

Rational solutions reduce to polynomial solutions given a \textbf{multiple of the denominator}.
Rational solutions of differential equations

\[ a_r y^{(r)} + \cdots + a_1(x) y' + a_0 y = 0 \]  \hspace{1cm} (D)

Rational solutions reduce to polynomial solutions given a multiple of the denominator.

**Observation 1.** Any pole \( \xi \in \overline{K} \) of \( y \) must be a singular point.

**Observation 2.** If \( y \) has a pole of multiplicity \( m \) at \( \xi \in \overline{K} \), its series expansion provides a solution in \( \overline{K}((x - \xi)) \) of valuation \( m \).
Rational solutions of differential equations

\[ a_r y^{(r)} + \cdots + a_1(x) y' + a_0 y = 0 \quad (D) \]

Rational solutions reduce to polynomial solutions given a **multiple** of the denominator.

**Observation 1.** Any pole \( \xi \in \overline{K} \) of \( y \) must be a singular point.

**Observation 2.** If \( y \) has a pole of multiplicity \( m \) at \( \xi \in \overline{K} \), its series expansion provides a solution in \( \overline{K}((x - \xi)) \) of valuation \( m \).

**Proposition.** For all \( \zeta \in \overline{K} \) such that \( a_r(\zeta) = 0 \), let \( m_\zeta \) be the smallest root in \( \mathbb{Z}_{<0} \) of the indicial polynomial of (D) at \( \zeta \) (if any, and \( m_\zeta = 0 \) otherwise).

Then the denominator of any rational solution of (D) is divisible by \( Q = \prod_\zeta (x - \zeta)^{m_\zeta} \).

Better variant: attach indicial polynomials to **factors** of \( a_r \) instead of roots to avoid working over \( \overline{K} \).
Rational solutions of differential equations

\[ a_r y^{(r)} + \cdots + a_1(x) y' + a_0 y = 0 \quad (D) \]

Rational solutions reduce to polynomial solutions given a multiple of the denominator.

**Observation 1.** Any pole \( \xi \in \overline{\mathbb{K}} \) of \( y \) must be a singular point.

**Observation 2.** If \( y \) has a pole of multiplicity \( m \) at \( \xi \in \overline{\mathbb{K}} \), its series expansion provides a solution in \( \mathbb{K}((x - \xi)) \) of valuation \( m \).

**Proposition.** For all \( \zeta \in \overline{\mathbb{K}} \) such that \( a_r(\zeta) = 0 \), let \( m_\zeta \) be the smallest root in \( \mathbb{Z}_{<0} \) of the indicial polynomial of \( (D) \) at \( \zeta \) (if any, and \( m_\zeta = 0 \) otherwise). Then the denominator of any rational solution of \( (D) \) is divisible by \( Q = \prod_\zeta (x - \zeta)^{m_\zeta} \).

Better variant: attach indicial polynomials to factors of \( a_r \) instead of roots to avoid working over \( \overline{\mathbb{K}} \).

**Algorithm.** Compute \( Q \) as above, change \( y \) to \( \frac{w}{Q} \), basis of solutions \( w \in \mathbb{K}[x] \).
Proposition. Any fixed linear combination of the entries of a solution $Y$ of
\[ Y'(x) = A(x) Y(x), \quad A \in \mathbb{K}(x)^{r \times r} \]
satisfies a scalar differential equation of order $r \leq r$ with coefficients in $\mathbb{K}(x)$.

Proof.
\[
\begin{align*}
Y &= I_r Y \\
Y' &= A Y
\end{align*}
\]
Proposition. Any fixed linear combination of the entries of a solution $Y$ of
\[
Y'(x) = A(x) Y(x), \quad A \in \mathbb{K}(x)^{r \times r}
\]
satisfies a scalar differential equation of order $r \leq r$ with coefficients in $\mathbb{K}(x)$.

Proof.

\[
\begin{align*}
Y &= I_r \quad Y \\
Y' &= A \quad Y \\
Y'' &= A' Y + A Y'
\end{align*}
\]
Proposition. Any fixed linear combination of the entries of a solution $Y$ of

$$Y'(x) = A(x) Y(x), \quad A \in \mathbb{K}(x)^{r \times r}$$

satisfies a scalar differential equation of order $r \leq r$ with coefficients in $\mathbb{K}(x)$.

Proof.

\[
\begin{align*}
Y &= I_r \ Y \\
Y' &= A \ Y \\
Y'' &= A' Y + A Y' \\
&= (A' + A^2) \ Y
\end{align*}
\]
Proposition. Any fixed linear combination of the entries of a solution $Y$ of

$$Y'(x) = A(x) Y(x), \quad A \in \mathbb{K}(x)^{r \times r}$$

satisfies a scalar differential equation of order $r \leq r$ with coefficients in $\mathbb{K}(x)$.

Proof.

\[
\begin{align*}
Y &= I_r Y \\
Y' &= A Y \\
Y'' &= A' Y + A Y' \\
&= (A' + A^2) Y \\
Y''' &= (\Box' + \Box A) Y
\end{align*}
\]
Proposition. Any fixed linear combination of the entries of a solution $Y$ of

$$Y'(x) = A(x)\, Y(x), \quad A \in \mathbb{K}(x)^{r \times r}$$

satisfies a scalar differential equation of order $r \leq r$ with coefficients in $\mathbb{K}(x)$.

Proof.

$$Y = I_r \, Y$$
$$Y' = A \, Y$$
$$Y'' = A' \, Y + A \, Y'$$
$$= (A' + A^2) \, Y$$
$$Y''' = (\Box' + \Box A) \, Y$$
$$\vdots$$
$$Y^{(r)} = (\ldots) \, Y$$
Proposition. Any fixed linear combination of the entries of a solution $Y$ of

$$Y'(x) = A(x) Y(x), \quad A \in \mathbb{K}(x)^{r \times r}$$

satisfies a scalar differential equation of order $r \leq r$ with coefficients in $\mathbb{K}(x)$.

Proof.

$I_r$

$A$

$A'Y + AY'$

$(A' + A^2)$

$(\square' + \square A)$

(...)

□
Proposition. Any fixed linear combination of the entries of a solution $Y$ of

$$Y'(x) = A(x) Y(x), \quad A \in \mathbb{K}(x)^{r \times r}$$

satisfies a scalar differential equation of order $r \leq r$ with coefficients in $\mathbb{K}(x)$.

Proof. Let $K \in \mathbb{K}^r$.

\[
\begin{align*}
K \cdot & \quad I_r \\
K \cdot & \quad A \\
K \cdot & \quad A' Y + A Y' \\
K \cdot & \quad (A' + A^2) \\
K \cdot & \quad (\Box' + \Box A) \\
K \cdot & \quad (\ldots)
\end{align*}
\]
From differential systems to scalar equations

**Proposition.** Any fixed linear combination of the entries of a solution $Y$ of

$$Y'(x) = A(x) Y(x), \quad A \in \mathbb{K}(x)^{r \times r}$$

satisfies a scalar differential equation of order $r \leq r$ with coefficients in $\mathbb{K}(x)$.

**Proof.** Let $K \in \mathbb{K}^r$.

\[
\begin{array}{c}
K \cdot & I_r \\
K \cdot & A \\
K \cdot & A' Y + A Y' \\
K \cdot & (A' + A^2) \\
K \cdot & (\Box' + \Box A) \\
K \cdot & (\ldots)
\end{array}
\begin{array}{c}
\{ r + 1 \text{ vectors in dimension } r \}
\end{array}
\]
Proposition. Any fixed linear combination of the entries of a solution $Y$ of
\[ Y'(x) = A(x) Y(x), \quad A \in K(x)^{r \times r} \]
satisfies a scalar differential equation of order $r \leq r$ with coefficients in $K(x)$.

Proof. Let $K \in K^r$.

\[
\begin{align*}
K \cdot & I_r \\
K \cdot & A \\
K \cdot & A'Y + AY' \\
K \cdot & (A' + A^2) \\
K \cdot & (\Box' + \Box A) \\
\end{align*}
\]

\[ r + 1 \text{ vectors in dimension } r \Rightarrow a_0 K Y + \cdots + a_r K Y^{(r)} = 0 \]
Solutions of differential systems

Assume for simplicity that there was no relation between $K I_r$, $KA$, $K (A' + A^2)$, … before the $(r + 1)$th element of the sequence.

After solving the scalar equation (for rational solutions, say):

\[
\begin{pmatrix}
- & K I_r & - \\
- & KA & - \\
- & K (A' + A^2) & - \\
\vdots & & \\
- & K (\ldots) & -
\end{pmatrix}
\begin{pmatrix}
Y \\
\vdots \\
K \cdot Y^{(r-1)}
\end{pmatrix}
= 
\begin{pmatrix}
K \cdot Y \\
K \cdot Y' \\
K \cdot Y'' \\
\vdots \\
K \cdot Y^{(r-1)}
\end{pmatrix}
\text{invertible}
\text{known}
\]
Finally . . .
Two exercises for next time

**Exercise 1.** Consider the differential equation

\[(x - 1) y''(x) + (-x + 3) y'(x) - y(x) = 0. \quad (E)\]

1. Let \( y(x) = \sum_{n=-N}^{\infty} y_n (x - 1)^n \) be a solution of (E) and set \( y_n = 0 \) for \( n < -N \). Show that the sequence \((y_n)_{n \in \mathbb{Z}}\) satisfies

\[\forall n \in \mathbb{Z}, \quad (n + 1) (n + 2) u_{n+1} = (n + 1) u_n.\]

2. Find all rational solutions of (E).

   (Hint: significant shortcuts are possible w.r.t. the full algorithm!)

**Exercise 2.** Give an algorithm to convert an \( n \)-bit number from base 2 to base 10 in \( O(M_{\mathbb{Z}}(n) \log n) \) bit operations, where \( M_{\mathbb{Z}}(n) \) is a bound on the cost of \( n \)-bit integer multiplication.
The rest of the course

This was the last lecture of the first period.

- Nov. 16 (next week): **exercise session**
  → come with your questions!
- Nov. 23: **first-period exam**

**Second period** (starting Dec. 7):

- polynomial matrices and Hermite-Padé approximation (Vincent Neiger)
- factorization of polynomials, lattice reduction (Pierre Lairez)
- introductions to more specialized topics (Alin Bostan, Pierre Lairez)