Evaluating parametric holonomic sequences using rectangular splitting

Fredrik Johansson

ISSAC 2014, Kobe, Japan
Goal

Compute the entry $c(n)$ in a sequence satisfying a linear recurrence equation

$$c(k + 1) = M(k) \cdot c(k)$$

- $c(k + 1)$ is a vector
- $M(k)$ is a square matrix with entries polynomials in $k$
- $c(k)$ is a vector
Goal

Compute the entry $c(n)$ in a sequence satisfying a linear recurrence equation

$$c(k + 1) = M(k) \cdot c(k)$$

- **vector**
- **square matrix**
- **vector**

entries polynomials in $k$

Examples

- $n!$, $\prod_{k=0}^{n-1} (x + k)$, $\sum_{k=1}^{n} \frac{1}{x + k}$,
- $\exp(x) \approx \sum_{k=0}^{n} \frac{x^k}{k!}$, $J_n(x) \approx \sum_{k=0}^{n} \frac{(-1)^k}{k!(k + n)!} \left(\frac{x}{2}\right)^{2k+n}$
How to do it
Naively
How to do it

Naively

\[ c(1) = M(0)c(0) \]
How to do it

Naively

\[ c(1) = M(0)c(0) \]
\[ c(2) = M(1)c(1) \]
How to do it

Naively

\[ c(1) = M(0)c(0) \]
\[ c(2) = M(1)c(1) \]
\[ c(3) = M(2)c(2) \]
\[ \vdots \]
How to do it

Naively

\[ c(1) = M(0)c(0) \]
\[ c(2) = M(1)c(1) \]
\[ c(3) = M(2)c(2) \]
\[ \vdots \]

Cleverly

Compute

\[ M(n - 1)M(n - 2) \cdots M(1)M(0) \]

Then multiply by the initial vector \( c(0) \)
Binary splitting

Example: \[ M(k) = (x + k), \quad n = 4 \]
Binary splitting

Example: $M(k) = (x + k)$, $n = 4$

$$(x + 3) (x + 2) (x + 1) (x + 0)$$
Binary splitting

Example: \( M(k) = (x + k), \quad n = 4 \)

\[
(x + 3) \quad (x + 2) \quad (x + 1) \quad (x + 0) \\
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\
(x^2 + 5x + 6) \quad (x^2 + x)
\]
Binary splitting

Example: $M(k) = (x + k)$, \hspace{1cm} n = 4

$(x + 3) (x + 2) \quad (x + 1) (x + 0)$

$(x^2 + 5x + 6) \quad (x^2 + x)$

$(x^4 + 6x^3 + 11x^2 + 6x)$
Binary splitting

Example: $M(k) = (x + k)$, $n = 4$

$$
(x + 3) \quad (x + 2) \quad (x + 1) \quad (x + 0) \\
\Rightarrow \\
(x^2 + 5x + 6) \quad (x^2 + x) \\
\Rightarrow \\
(x^4 + 6x^3 + 11x^2 + 6x)
$$

Useful if the cost grows with the entries

$$
R[x] \quad O(M(n) \log n) \quad = \quad O^{\sim}(n) \quad R\text{-operations} \\
\mathbb{Z} \quad O(M(n \log n) \log n) \quad = \quad O^{\sim}(n) \quad \text{bit operations}
$$
Fast multipoint evaluation

Example: $M(k) = (k + 1)$, $n = 9$
Fast multipoint evaluation

Example: \( M(k) = (k + 1) \), \( n = 9 \)

\[ P = (k + 3)(k + 2)(k + 1) = k^3 + 6k^2 + 11k + 6 \]

(binary splitting)
Fast multipoint evaluation

Example: \( M(k) = (k + 1), \quad n = 9 \)

\[
P = (k + 3)(k + 2)(k + 1) = k^3 + 6k^2 + 11k + 6
\]

binary splitting

\([P(6), P(3), P(0)] = [504, 120, 6]\)

fast multipoint evaluation
Fast multipoint evaluation

**Example:** \( M(k) = (k + 1), \quad n = 9 \)

\[ P = (k + 3)(k + 2)(k + 1) = k^3 + 6k^2 + 11k + 6 \]

binary splitting

\[ [P(6), P(3), P(0)] = [504, 120, 6] \]

fast multipoint evaluation

\[ P(6)P(3)P(0) = 362880 \]

repeated multiplication
Fast multipoint evaluation

Example: \( M(k) = (k + 1), \quad n = 9 \)

\[
P = (k + 3)(k + 2)(k + 1) = k^3 + 6k^2 + 11k + 6
\]

binary splitting

\[
[P(6), P(3), P(0)] = [504, 120, 6]
\]

fast multipoint evaluation

\[
P(6)P(3)P(0) = 362880
\]

repeated multiplication

Useful if arithmetic operations have fixed cost:
\[O(M(n^{1/2}) \log n) = O^\sim(n^{1/2})\] operations

(Bostan, Gaudry, Schost, 2007): \[O(M(n^{1/2}))\]
Parametric sequences

\[ M(k) = M(x, k), \text{ where entries of } M \text{ are in } R[x][k] \]

Evaluate at some given “expensive” value of the parameter \( x \)
Parametric sequences

\[ M(k) = M(x, k), \text{ where entries of } M \text{ are in } R[x][k] \]

Evaluate at some given “expensive” value of the parameter \( x \)

\[ c = 42 \in R \]
\[ x = 3.141592653589793238462643383279502884 \]
Parametric sequences

\[ M(k) = M(x, k), \text{ where entries of } M \text{ are in } R[x][k] \]

Evaluate at some given “expensive” value of the parameter \( x \)

\[ c = 42 \in R \]
\[ x = 3.141592653589793238462643383279502884 \]

Distinguish between operations

<table>
<thead>
<tr>
<th></th>
<th>GOOD</th>
<th>OK</th>
<th>BAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coefficient</td>
<td>( c + c, \ c \cdot c )</td>
<td>( x + x, \ c \cdot x )</td>
<td>( x \cdot x )</td>
</tr>
<tr>
<td>Scalar</td>
<td>( x + x, \ c \cdot x )</td>
<td>( x \cdot x )</td>
<td></td>
</tr>
<tr>
<td>Nonscalar</td>
<td>( x \cdot x )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Rectangular splitting

Evaluate the polynomial $\sum_{i=0}^{n} \square x^i$
Rectangular splitting

Evaluate the polynomial \( \sum_{i=0}^{n} \Box x^i \)

\[
\begin{align*}
( \Box + \Box x + \Box x^2 + \Box x^3 ) & \quad + \\
( \Box + \Box x + \Box x^2 + \Box x^3 ) x^4 & \quad + \\
( \Box + \Box x + \Box x^2 + \Box x^3 ) x^8 & \quad + \\
( \Box + \Box x + \Box x^2 + \Box x^3 ) x^{12} & \quad +
\end{align*}
\]
Rectangular splitting

Evaluate the polynomial \[ \sum_{i=0}^{n} \square x^i \]

\[
\begin{align*}
( \square & + \square x + \square x^2 + \square x^3 ) \quad + \\
( \square & + \square x + \square x^2 + \square x^3 ) \quad x^4 \quad + \\
( \square & + \square x + \square x^2 + \square x^3 ) \quad x^8 \quad + \\
( \square & + \square x + \square x^2 + \square x^3 ) \quad x^{12}
\end{align*}
\]
Rectangular splitting

Evaluate the polynomial \( \sum_{i=0}^{n} \Box x^i \)

\[
\begin{align*}
( & \Box + \Box x + \Box x^2 + \Box x^3 ) & + \\
( & \Box + \Box x + \Box x^2 + \Box x^3 ) & \cdot x^4 + \\
( & \Box + \Box x + \Box x^2 + \Box x^3 ) & \cdot x^8 + \\
( & \Box + \Box x + \Box x^2 + \Box x^3 ) & \cdot x^{12}
\end{align*}
\]

- \( O(n) \) scalar operations, and
- \( O(n^{1/2}) \) nonscalar multiplications

(Paterson-Stockmeyer, 1973)
Rectangular splitting for sequences

Expand $P = M(x, n - 1) \cdots M(x, 1)M(x, 0)$ using binary splitting

Evaluate each entry in $P$ using rectangular splitting
Rectangular splitting for sequences

Expand \( P = M(x, n - 1) \cdots M(x, 1)M(x, 0) \) using binary splitting

Evaluate each entry in \( P \) using rectangular splitting

- \( O(M(n) \log n) \) coefficient operations
- \( O(n) \) scalar operations
- \( O(n^{1/2}) \) nonscalar operations
This is actually bad

\[ M = (x + k), \quad n = 16 \]

\[ P = x^{16} + 120x^{15} + 6580x^{14} + 218400x^{13} + 4899622x^{12} + 78558480x^{11} + 928095740x^{10} + 8207628000x^9 + 54631129553x^8 + 272803210680x^7 + 1009672107080x^6 + 2706813345600x^5 + 5056995703824x^4 + 6165817614720x^3 + 4339163001600x^2 + 1307674368000x \]

Coefficients grow to \( O(n \log n) \) bits.
Scalar multiplications can become slower than nonscalar multiplications!
Improved rectangular splitting

Expand \( O(n^{1/2}) \) polynomials of degree \( O(n^{1/2}) \)
Improved rectangular splitting

Expand $O(n^{1/2})$ polynomials of degree $O(n^{1/2})$

$(M(x, 15) \ M(x, 14) \ M(x, 13) \ M(x, 12)) \cdot$
$(M(x, 11) \ M(x, 10) \ M(x, 9) \ M(x, 8)) \cdot$
$(M(x, 7) \ M(x, 6) \ M(x, 5) \ M(x, 4)) \cdot$
$(M(x, 3) \ M(x, 2) \ M(x, 1) \ M(x, 0))$
Improved rectangular splitting
Expand $O(n^{1/2})$ polynomials of degree $O(n^{1/2})$

\[(x^4 + 54x^3 + 1091x^2 + 9774x + 32760) \cdot (x^4 + 38x^3 + 539x^2 + 3382x + 7920) \cdot (x^4 + 22x^3 + 179x^2 + 638x + 840) \cdot (x^4 + 6x^3 + 11x^2 + 6x + 0)\]
Improved rectangular splitting

Expand $O(n^{1/2})$ polynomials of degree $O(n^{1/2})$

\[
\begin{align*}
(x^4 + 54x^3 + 1091x^2 + 9774x + 32760) & \cdot \\
(x^4 + 38x^3 + 539x^2 + 3382x + 7920) & \cdot \\
(x^4 + 22x^3 + 179x^2 + 638x + 840) & \cdot \\
(x^4 + 6x^3 + 11x^2 + 6x + 0) &
\end{align*}
\]
Improved rectangular splitting

Expand $O(n^{1/2})$ polynomials of degree $O(n^{1/2})$

\[(x^4 + 54x^3 + 1091x^2 + 9774x + 32760) \cdot (x^4 + 38x^3 + 539x^2 + 3382x + 7920) \cdot (x^4 + 22x^3 + 179x^2 + 638x + 840) \cdot (x^4 + 6x^3 + 11x^2 + 6x + 0)\]

- $O(M(n) \log n)$ coefficient operations
- $O(n)$ scalar operations
- $O(n^{1/2})$ nonscalar operations
- $O(n^{1/2} \log n)$-bit coefficients
Numerical evaluation

Asymptotic speedup when the parameter $x$ is a real number with $p \sim n$ bits: $500^{1/2} \approx 20$ (no further improvement when step length exceeds $\sim 500^{1/2}$)
Hypergeometric series summation

Smith (1989): rectangular splitting with content removal → $O(\log n)$ bit coefficients

$$\exp(x) \approx \left(1 + \frac{1}{1} \left(x + \frac{1}{2} \left(x^2 + \frac{1}{3} x^3\right)\right)\right)$$
$$+ \frac{x^4}{4!} \left(1 + \frac{1}{5} \left(x + \frac{1}{6} \left(x^2 + \frac{1}{7} x^3\right)\right)\right)$$
$$+ \frac{x^8}{8!} \left(1 + \frac{1}{9} \left(x + \frac{1}{10} \left(x^2 + \frac{1}{11} x^3\right)\right)\right)$$
Hypergeometric series summation

Smith (1989): rectangular splitting with content removal $\rightarrow O(\log n)$ bit coefficients

$$\exp(x) \approx \left(1 + \frac{1}{1} \left(x + \frac{1}{2} \left(x^2 + \frac{1}{3} x^3\right)\right)\right)$$

$$+ \frac{x^4}{4!} \left(1 + \frac{1}{5} \left(x + \frac{1}{6} \left(x^2 + \frac{1}{7} x^3\right)\right)\right)$$

$$+ \frac{x^8}{8!} \left(1 + \frac{1}{9} \left(x + \frac{1}{10} \left(x^2 + \frac{1}{11} x^3\right)\right)\right)$$

Our algorithm: gives larger coefficients, but works for more general sequences, and avoids divisions
Smith’s algorithm for rising factorials

Smith (2001): use
\[ \Delta = \prod_{i=0}^{3}(x + k + 4 + i) - \prod_{i=0}^{3}(x + k + i) \]

\[ \Delta = (840 + 632k + 168k^2 + 16k^3) \]
\[ + (632 + 336k + 48k^2)x \]
\[ + (168 + 48k)x^2 \]
\[ + 16x^3 \]
Smith’s algorithm for rising factorials

Smith (2001): use
\[ \Delta = \prod_{i=0}^{3}(x + k + 4 + i) - \prod_{i=0}^{3}(x + k + i) \]

\[ \Delta = (840 + 632k + 168k^2 + 16k^3) \]
\[ + (632 + 336k + 48k^2)x \]
\[ + (168 + 48k)x^2 \]
\[ + 16x^3 \]

Our algorithm: generalizes this to arbitrary step length (plus complexity analysis), arbitrary holonomic sequences; slight simplification
Rising factorial benchmark

Compute $\prod_{k=0}^{n-1}(x + k)$ where $x$ is a real number with $p$ bits of precision.
Rising factorial benchmark

Compute $\prod_{k=0}^{n-1}(x + k)$ where $x$ is a real number with $p$ bits of precision

- Naive algorithm
- Algorithm 1: fast multipoint evaluation
- Algorithm 2: poor rectangular splitting
- Algorithm 3: improved rectangular splitting
- Algorithm 4: difference version

Algorithm 3 and 4 use step length $\min(0.2p^{0.4}, n^{0.5})$

In the benchmark: $p = 4n$
Rising factorial benchmark results

- Relative time
  - naive recurrence
  - fast multipoint eval
  - rect. splitting (poor)
  - rect. splitting (improved)
  - rect. splitting (∆)

Graph shows the relative time for different methods as a function of $n$. The y-axis represents the relative time on a logarithmic scale, while the x-axis shows $n$ on a logarithmic scale from $10^1$ to $10^6$. The methods are compared across a range of $n$ values, with the relative time decreasing as $n$ increases, indicating improved efficiency for larger values of $n$.
Speedup for $\Gamma(x)$

\[
\Gamma(x) = \frac{\Gamma(x + n)}{(x(x + 1) \cdots (x + n - 1))}, \text{ Stirling series for large } x + n
\]
Asymptotically fast gamma function

\[ \Gamma(x) \approx \gamma(x, N) = x^{-1} N^x e^{-N} \frac{1}{1} F_1(1, 1 + x, N) \]
Asymptotically fast gamma function

$$\Gamma(x) \approx \gamma(x, N) = x^{-1} N^x e^{-N} \, _1F_1(1, 1 + x, N)$$

Partial sums for \(_1F_1(1, 1 + x, N)\) satisfy order-2 recurrence with

$$M(x, k) = \frac{1}{1 + k + x} \begin{pmatrix} 1 + k + x & 1 + k + x \\ 0 & N \end{pmatrix}$$
Asymptotically fast gamma function

\[ \Gamma(x) \approx \gamma(x, N) = x^{-1} N^x e^{-N} {}_1F_1(1, 1 + x, N) \]

Partial sums for \( {}_1F_1(1, 1 + x, N) \) satisfy order-2 recurrence with

\[ M(x, k) = \frac{1}{1 + k + x} \begin{pmatrix} 1 + k + x & 1 + k + x \\ 0 & N \end{pmatrix} \]

For \( p \)-bit precision: \( n \sim N \sim p, O^\sim(p^{1.5}) \) with fast multipoint evaluation
Comparison of gamma function algorithms

### Graph

![Graph comparing different gamma function algorithms](image)

- $1_F^1$ naive recurrence
- $1_F^1$ fast multipoint eval
- $1_F^1$ rectangular splitting
- Stirling (no cache)
- Stirling (cache)

### Precision $p$ (bits)

- $10^0$
- $10^1$
- $10^2$
- $10^3$
- $10^4$
- $10^5$
- $10^6$

### Relative time

- $10^0$
- $10^1$
- $10^2$
- $10^3$
- $10^4$
- $10^5$
- $10^6$
Summary

Efficient rectangular splitting for parametric sequences:
Summary

Efficient rectangular splitting for parametric sequences:

- Simple, works for a very general class of sequences
Summary

Efficient rectangular splitting for parametric sequences:

- Simple, works for a very general class of sequences
- Asymptotically slower than fast multipoint evaluation, but competitive in practice
Summary

Efficient rectangular splitting for parametric sequences:

- Simple, works for a very general class of sequences
- Asymptotically slower than fast multipoint evaluation, but competitive in practice
- Generalizes two different algorithms by Smith
Summary

Efficient rectangular splitting for parametric sequences:

- Simple, works for a very general class of sequences
- Asymptotically slower than fast multipoint evaluation, but competitive in practice
- Generalizes two different algorithms by Smith
- Fastest available high-precision implementation of the gamma function