# Summing Symbols in Mutual Recurrences

Berkeley Churchill

University of California, Santa Barbara
Santa Barbara, CA 93117

COCOON 2011
August 16, 2011

Joint work with Ed Lamagna, University of Rhode Island

# Problem Statement

How can computers determine summation identities for recursive sequences in a "nice" way?
For example, consider a mutual recurrence similar to this:

$$\begin{aligned} A_n &= 2A_{n-1} + B_{n-1} \\ B_n &= A_{n-1} + 2B_{n-2}. \end{aligned}$$

Is there a way for a computer to derive a general formula for $\sum_{i=0}^{n} A_i$?

- The algorithm should apply to any mutual recurrence.
- Want solution in terms of the original sequence itself
- E.g. for the Fibonacci sequence, $\sum_{i=0}^{n} F_i = F_{n+2} - 1$.
- Algorithm should handle inhomogeneous terms
- Previously solved for a single recurrence [Greene & Wilf, 2007], [Ravenscroft & Lamagna, 2008]

# Example: Fibonacci Sequence

Solution [Ravenscroft & Lamagna, 2008]: Sum the recurrence itself!

$$
\begin{array}{rcl}
F_n & = & F_{n-1} + F_{n-2} \\
F_{n-1} & = & F_{n-2} + F_{n-3} \\
 & \vdots & \\
+ \quad F_2 & = & F_1 + F_0 \\
\hline
\sum_{i=2}^{n} F_i & = & \sum_{i=1}^{n-1} F_i + \sum_{i=0}^{n-2} F_i
\end{array}
$$

Then adjust the limits:

$$\sum_{i=0}^{n} F_i - F_1 - F_2 = \sum_{i=0}^{n} F_i - F_1 - F_n + \sum_{i=0}^{n} F_i - F_n - F_{n-1}$$

Everything cancels out (usually...) !

$$\sum_{i=1}^{n} F_i = 2F_n + F_{n-1} - F_2 = (F_{n+1} + F_n) - 1 = F_{n+2} - 1$$

# Approach: Just Keep Summing!

What happens if we sum each recurrence in this example?

$$\begin{aligned} A_n &= 2A_{n-1} + B_{n-1} \\ B_n &= A_{n-1} + 2B_{n-2} \end{aligned}$$

If $S_n^A = \sum_{i=0}^{n} A_i$ and $S_n^B = \sum_{i=0}^{n} B_i$ then these recurrences sum to: (after simplifying)

$$\begin{aligned} S_A + S_B &= 2A_n + A_0 - A_1 + B_n + B_0 \\ S_A + S_B &= A_n + A_0 + 2B_n + 2B_{n-1} - B_0 - B_1 \end{aligned}$$

- A priori, this system cannot be solved for $S_A$ and $S_B$.
- Sometimes this technique will work. (formalized later)
- How do we continue?

## Definition (Mutual Recurrence)

Let $K$ be a field, and $m, d \in \mathbb{Z}^+$. A system of mutual linear recurrence relations with constant coefficients on $K$ of order $d$ in $m$ variables is a set of $m$ functions $A_1(n), \ldots, A_m(n)$ mapping $\mathbb{N}$ into $K$ satisfying

$$\begin{pmatrix} A_1(n) \\ A_2(n) \\ \vdots \\ A_m(n) \end{pmatrix} = M_1 \begin{pmatrix} A_1(n-1) \\ A_2(n-1) \\ \vdots \\ A_m(n-1) \end{pmatrix} + \cdots + M_d \begin{pmatrix} A_1(n-d) \\ A_2(n-d) \\ \vdots \\ A_m(n-d) \end{pmatrix} + \begin{pmatrix} g_1(n) \\ g_2(n) \\ \vdots \\ g_m(n) \end{pmatrix}$$

for some $M_1, \ldots, M_d \in M_m(K)$ and $g_1, \ldots, g_m$ mapping $\mathbb{N} \to K$.

# Example

This mutual recurrence

$$\begin{aligned} A_n &= 2A_{n-1} + B_{n-1} \\ B_n &= A_{n-1} + 2B_{n-2} \end{aligned}$$

can be written as

$$\begin{pmatrix} A_1(n) \\ A_2(n) \end{pmatrix} = \begin{pmatrix} 2 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} A_1(n-1) \\ A_2(n-1) \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} A_1(n-2) \\ A_2(n-2) \end{pmatrix}.$$

The order, $d$, is 2. The goal is to express both

$$\sum_{i=0}^{n} A_i \quad \text{and} \quad \sum_{i=0}^{n} B_i$$

as a linear combination of the functions $A_n, A_{n-1}, A_{n-2}, B_n, B_{n-1}, B_{n-2}$ over $K$ along with an inhomogeneous term depending on initial conditions.

# Homogeneous Case

Let $E$ denote the shift operator, that is $EA_n = A_{n+1}$. A mutual recurrence over $K$ can be written as a set of polynomials in $K[E]$.

$$\begin{array}{rcl} A_n & = & 2A_{n-1} + B_{n-1} \\ B_n & = & A_{n-1} + 2B_{n-2} \end{array} \quad \Leftrightarrow \quad \begin{array}{rcl} (E-2)A(n) - B(n) & = & 0, \\ (E^2-2)B(n) - EA(n) & = & 0. \end{array}$$

One can solve a system like this for $A_n$ and $B_n$ in $K(E)$. In this case for $B(n)$ we have

$$(E^3 - 2E^2 - 3E + 4)B_n = 0$$

so

$$B_n = 2B_{n-1} + 3B_{n-2} - 4B_{n-3}$$

Can solve this using well-known methods [Greene & Wilf, 2007], [Ravenscroft & Lamagna, 2008]

# Non-Homogeneous Case

For a given mutual recurrence $A_1, \ldots, A_m$ with order $d$, define

$$S(f(n)) = \sum_{i=d}^{n} f(i)$$

for any $f : \mathbb{N} \to K$. Also define

$$\begin{aligned} S_i(n) &= S_i^1(n) = S(A_i(n)) \\ S_i^j(n) &= S(S_i^{j-1}(n)). \end{aligned}$$

Let $V$ be the vector space over $K$ that has each defined $A_i(n-j)$ and $S_i^j(n)$ as basis elements.

$S$ is linear on $V$! It's easy to explicitly compute the action of $S$ on each member of the basis.

# Re-Writing the Recurrence

$$\begin{pmatrix} A_1(n) \\ A_2(n) \\ \vdots \\ A_m(n) \end{pmatrix} = M_1 \begin{pmatrix} A_1(n-1) \\ A_2(n-1) \\ \vdots \\ A_m(n-1) \end{pmatrix} + \cdots + M_d \begin{pmatrix} A_1(n-d) \\ A_2(n-d) \\ \vdots \\ A_m(n-d) \end{pmatrix} + \begin{pmatrix} g_1(n) \\ g_2(n) \\ \vdots \\ g_m(n) \end{pmatrix}$$

$$\begin{pmatrix} I & -M_1 & -M_2 & \cdots & -M_d \end{pmatrix} \left(\begin{array}{c} A_1(n) \\ A_2(n) \\ \vdots \\ A_m(n) \\ \hline \vdots \\ \hline A_1(n-d) \\ A_2(n-d) \\ \vdots \\ A_m(n-d) \end{array}\right) = \begin{pmatrix} g_1(n) \\ g_2(n) \\ \vdots \\ g_m(n) \end{pmatrix}.$$

# Re-Writing the Recurrence 2

$$\begin{pmatrix} 0 & \cdots & 0 & | & I & -M_1 & -M_2 & \cdots & -M_d \end{pmatrix} \begin{pmatrix} S_1^d(n) \\ S_2^d(n) \\ \vdots \\ S_m^d(n) \\ \hline \vdots \\ \hline S_1(n) \\ S_2(n) \\ \vdots \\ S_d(n) \\ \hline A_1(n) \\ A_2(n) \\ \vdots \\ A_m(n) \\ \hline \vdots \\ \hline A_1(n-d) \\ A_2(n-d) \\ \vdots \\ A_m(n-d) \end{pmatrix} = \begin{pmatrix} g_1(n) \\ g_2(n) \\ \vdots \\ g_m(n) \end{pmatrix}.$$

Shorthand: $\begin{pmatrix} 0 & \cdots & 0 & | & I & -M_1 & \cdots & -M_d & | & G \end{pmatrix}$

# Row-Operations

Shorthand: $\begin{pmatrix} 0 & \cdots & 0 & | & I & -M_1 & \cdots & -M_d & | & G \end{pmatrix}$

Given the augmented matrix, we can perform four row operations:

1. Swap two rows
2. Multiply a row by a scalar in $K$
3. Add one row to another
4. Apply the summation operator to a row

The goal is to make the new augmented matrix look like this:

$$\begin{pmatrix} 0 & \cdots & 0 & I & | & X_1 & \cdots & X_{m+1} & | & Y \end{pmatrix}$$

so that the $S_i(n)$ can be solved in terms of the $A_i(n-j)$.

There is a detailed example in the paper.

# The Algorithm

Take an augmented matrix,

$$U = \left( \begin{array}{c|cccc|c} 0 & I & -M_1 & -M_2 & \cdots & -M_d & G \end{array} \right).$$

For each $t \geq 0$ do the following starting with $U$:

1. Augment the matrix with $t+1$ blocks of $m \times m$ zero matrices on the left-hand side.
2. Duplicate each block row of the matrix $t$ times. Number the block rows from top to bottom starting at 1.
3. Apply $S$ to the $(t+2-i)^{th}$ block row $i$ times for $1 \leq i \leq t+1$.
4. If placing this matrix in row-reduced echelon form results in a matrix of the form

$$\left( \begin{array}{ccccc|cccc|c} 0 & 0 & \cdots & 0 & I & * & * & \cdots & * & * \end{array} \right)$$

then stop; back-substitute to solve for each $S_i(n)$ when $i \in \{1, \ldots, m\}$. Otherwise increment $t$ by one and continue.

# Results

- If the algorithm terminates, the solution is correct.
- If $A_1 + \cdots + A_d$ does not have an eigenvalue of 1, the algorithm terminates when $t = 1$. This is common.

**Theorem**

*If $m = 1$, that is, there is only a single recurrence, the algorithm terminates.*

**Theorem**

*If $d = 1$, that is, each $A_i(n)$ depends only on various $A_j(n-1)$, then the algorithm terminates.*

# Runtime Analysis

- In both the $m = 1$ and $d = 1$ case, an appropriate value of $t$ can be determined from counting the factors of $(x - 1)$ in the appropriate polynomial. This can be bounded by the polynomial's degree.
  - For $m = 1$, $t \leq d$.
  - For $d = 1$, $t \leq m$.
  - This suggests the existence of a more general polynomial that could be used to finish the proof for all cases.
- For the cases where termination is known, the algorithm runs in polynomial time, namely $O(m^3(t + d)^3) \subset O(m^6 d^3)$.
  - This bound is not tight.
  - The bottleneck is in the row-reduction algorithm. For simplicity, $O(n^3)$ was used as the running time for this applied to an $n \times n$ matrix.
- Easy to implement!

# References


Greene, Curtis, & Wilf, Herbert S. 2007.
Closed form summation of $C$-finite sequences.
*Transactions of the American Mathematical Society*, **359**, 1161–1189.

Ravenscroft, Robert A., & Lamagna, Edmund A. 2008.
Summation of linear recurrence sequences.
*Pages 125–132 of: Milestones in Computer Algebra*.

# Questions?

Berkeley Churchill <berkeley@berkeleychurchill.com>

You can find copies of these slides at
http://www.berkeleychurchill.com/research