

# Summing Systems of Recurrence Relations

## Project Proposal

Berkeley Churchill

NSF REU in Digital Forensics  
University of Rhode Island  
Kingston, RI 02881

June 24, 2010



## Problem Statement

How can computers calculate sums indefinitely? How can we give a computer a command like

$$\sum_{k=1}^{\infty} \frac{1}{k^2}$$

and get the result  $\pi^2/6$ ? How about:

$$\sum_{i=1}^n \frac{\left(\frac{1+\sqrt{5}}{2}\right)^i - \left(\frac{1-\sqrt{5}}{2}\right)^{-i}}{\sqrt{5}}$$

## Example: Fibonacci Sequence

Consider the Fibonacci Sequence:

$$F_n = F_{n-1} + F_{n-2}$$

where  $F_1 = F_2 = 1$  (one may define  $F_0 = 0$  if desired). How can express

$$S_n = \sum_{i=1}^n F_i = F_1 + F_2 + F_3 + \cdots + F_n$$

in terms of the values of the  $F_i$  themselves?

- ▶ Not so obvious
- ▶ We want a closed form solution
- ▶ Solution must be algorithmic and preferably simple

## Example: Fibonacci Sequence

Solution (Lamagna and Ravenscroft): Sum the recurrence itself!

$$\begin{aligned} F_n &= F_{n-1} + F_{n-2} \\ F_{n-1} &= F_{n-2} + F_{n-3} \\ &\vdots \\ F_4 &= F_3 + F_2 \\ + F_3 &= F_2 + F_1 \\ \hline \sum_{i=3}^n F_i &= \sum_{i=2}^{n-1} F_i + \sum_{i=1}^{n-2} F_i \end{aligned}$$

Then adjust the limits:

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

Everything cancels out (usually...) !

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

## Example: Fibonacci Sequence

This summing method has established that

$$\sum_{i=1}^n F_i = F_{n+2} - 1$$

Indeed,

$$\begin{aligned} F_1 &= F_2 &= 1 \\ F_3 &= 2 \\ F_4 &= 3 \\ F_5 &= 5 \\ F_6 &= 8 \\ F_7 &= 13 \end{aligned}$$

$$F_1 + F_2 + F_3 + F_4 + F_5 = 1 + 1 + 2 + 3 + 5 = 12 = F_7 - 1$$

## Goal: Summing Systems of Linear Recurrences

- ▶ How can this approach be generalized to systems of linear recurrences?
- ▶ For example, given

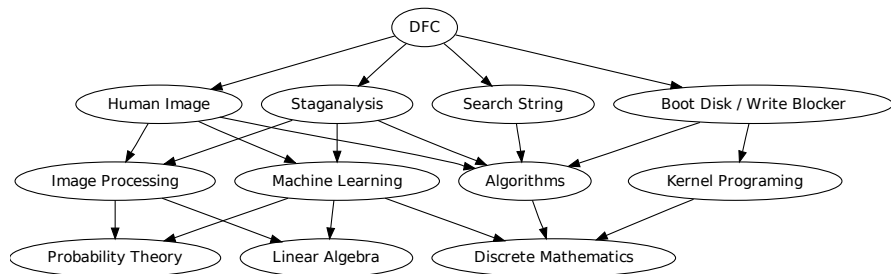
$$A_n = 2A_{n-1} + B_{n-2}$$

$$B_n = A_{n-1} + 2B_{n-2}$$

how can we compute  $\sum_{i=1}^n A_i$ ? Or  $\sum_{i=1}^n B_i$ ?

- ▶ Only closed-form solutions when possible
- ▶ Expressed in terms of the symbols we start with (the  $A_i$  and  $B_i$ )
- ▶ Non-homogeneous terms?
- ▶ Can we get a simple algorithm?
- ▶ Implement algorithm in Maple

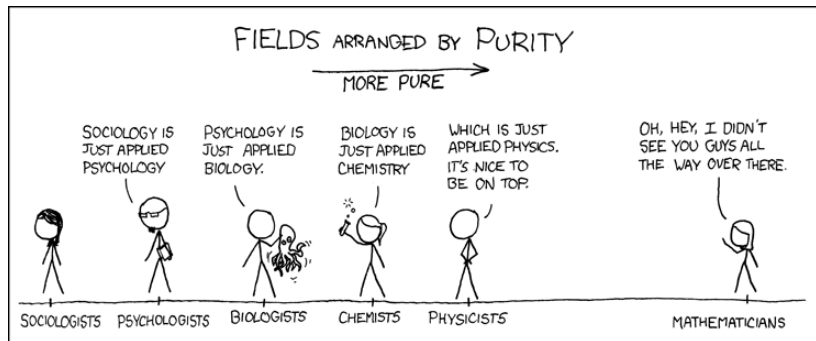
# Why We Care



- ▶ All of our cutting-edge projects rely heavily on mathematics
- ▶ Advancing discrete math and linear algebra helps
- ▶ Applications aren't obvious, but they're there!

# Significance

- ▶ Applications to difference equations and image processing
- ▶ Field theory has been applicable in the past (e.g. RAID-6)
- ▶ Maple is commercially supported and people use it



Source: <http://xkcd.com/435/>

# Approaches

- ▶ Summing Methods
- ▶ Linear Algebra
  - ▶ Conceptual linear algebra
  - ▶ How linear algebra applies to the problem
- ▶ Evaluating our work

$$\begin{bmatrix} \cos 90^\circ & \sin 90^\circ \\ -\sin 90^\circ & \cos 90^\circ \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

xkcd.com

## Approach: Just Keep Summing!

What happens if we sum each recurrence in this example?

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

If  $S_n^A = \sum_{i=1}^n A_i$  and  $S_n^B = \sum_{i=1}^n B_i$  then these recurrences sum to: (after simplifying)

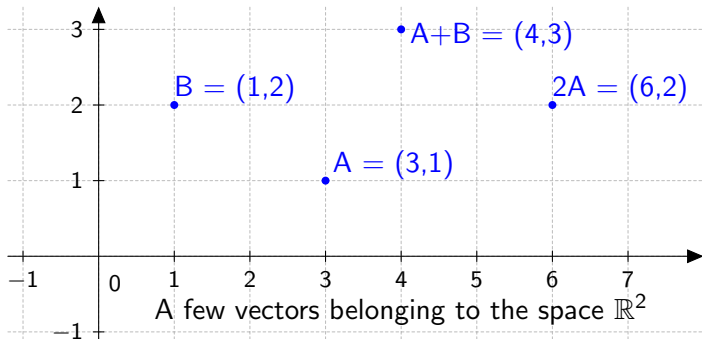
$$S_A + S_B = 2A_n + B_n + B_{n-1} + A_1 - A_2$$

$$S_A + S_B = A_n + 2B_n + 2B_{n-1} + A_1 - B_1 - B_2$$

- ▶ This system of equations does not have a unique solution, but the sum has a unique answer.
- ▶ How do we continue?
- ▶ Observation: the system can be solved if the original coefficient matrix does not have an eigenvalue of one.

## Approach: Linear Algebra

- ▶ A vector space is a set of objects (“vectors”) that can be added and subtracted from each other, and also multiplied by numbers.
- ▶ Typical examples include  $\mathbb{Q}$ ,  $\mathbb{R}$ ,  $\mathbb{C}$  along with higher dimensional spaces like  $\mathbb{R}^n$ .
- ▶ Key observation: because we can add linear recurrences together in a “nice” manner, they form a vector space



## Linear Algebra Continued

You can think of  $F_1, F_2, F_3, \dots$  as vectors:

- ▶ You can add them.
- ▶ You can multiply them.

A useful representation is as an ordered tuple:

$$F_1 = (1, 0, 0, \dots,)$$

$$F_2 = (0, 1, 0, \dots,)$$

$$\vdots$$

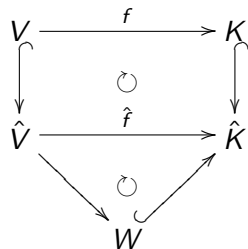
This way,  $2F_1 + 3F_2 - F_3 = (2, 3, -1, 0, \dots)$ .

- ▶ Transforms the problem statement into linear algebraic terms
- ▶ Allows us to rigorously prove an algorithm is correct
- ▶ Provides insights on how to proceed
- ▶ Makes an enormous body of knowledge applicable

# From a System of Linear Recurrences to a Vector Space

- ▶ Let  $V$  over  $K$  have basis  $\{\dots, A_0, A_1, A_2 \dots\}$  or more generally  $\{A_j^{(i)} : 1 \leq i \leq m, j \in \mathbb{Z}\}$  for a system.
- ▶ Consider  $f : V \rightarrow K$  a linear transformation that “evaluates” each of the  $A_j^{(i)}$ . E.g.  $f(F_0) = f(F_1) = 1$  for Fibonacci.

- ▶  $\hat{V}$  is space of functions  $g : \mathbb{N} \rightarrow V$
- ▶  $\hat{K}$  is space of functions  $g : \mathbb{N} \rightarrow K$
- ▶ Extend  $f$  to  $\hat{f} : \hat{V} \rightarrow \hat{K}$  naturally
- ▶ Let  $W = \ker(\hat{f})$



- ▶ E.g.  $g(n) = F_n - F_{n-1} - F_{n-2} \in W$  since  $F_n = F_{n-1} + F_{n-2}$ .
- ▶ The vector space  $W$  is the subspace of recurrence relations we care about!
- ▶ We can use linear algebra to make sure the algorithm is correct.

# Criteria for Evaluation

Our algorithm must:

1. work
2. be “provably” correct, in addition to working in practice
3. should run faster than algorithms using generating functions
4. solve some systems where no previous algorithm applied
5. express answer in a closed form using the symbols we start with

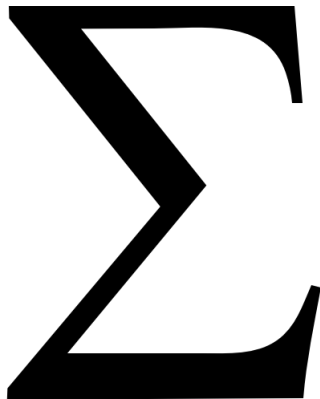
Our performance tests will include:

1. Complicated systems, with perhaps hundreds of recurrences
2. Recurrences of larger order (E.g.  $A_n = A_{n-32} + \dots$ )

***“Beware of bugs in the above code; I have only proved it correct, not tried it.”***

***- Donald Knuth***

## Summing it all up...



- ▶ Summing is an important task for computer algebra systems
- ▶ We don't have “nice” algorithms to sum systems of linear recurrences with respect to the original symbols
- ▶ Even in digital forensics, we care about advancing mathematics and computer science as a whole
- ▶ Linear algebra and other abstractions help dramatically
- ▶ Our algorithms will be implemented in Maple
- ▶ We will prove the correctness of our algorithm in addition to testing

# Questions?

Berkeley Churchill <berkeley@berkeleychurchill.com>

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