# Mind the Gap

Jan van Eijck

CKI-presentatiedag, 21 Juni, 2002

## Abstract

- Intelligent Tasks: Finding the Next Term of a Sequence
- Difference Analysis of Polynomial Sequences
- Charles Babbage's Difference Engine
- Finding the Form of the Sequence.
- Gaussian Elimination.
- Example Application: the Pie Cutting Sequence
- What has this to do with Intelligence?
- What has it all to do with Consciousness (if anything)?


```
module MGT

where
```

## Intelligence

Asking whether computers can think is like asking whether submarines can swim.

Edsger W. Dijkstra

Computers are intelligent to the extent that they are able to carry out intelligent tasks.

Idea behind Turing Test

## Intelligent Tasks: Finding the Next Term of a Sequence

Guess the next number:

$0$

$1$

$4$

$9$

$16$

$\ldots$

$n^2$

Guess the next number:

$0$

$2$

$6$

$12$

$20$

$\ldots$

$n^2 + n$

Guess the next number:

$1$

$2$

$9$

$28$

$65$

$\ldots$

$n^3 + 1$

Guess the next number:

$0$

$1$

$3$

$6$

$10$

$15$

$\ldots$

$\frac{1}{2}n^2 + \frac{1}{2}n = \frac{n(n+1)}{2}$

Guess the next number:

$3$

$7$

$17$

$39$

$79$

$\ldots$

$n^3 + 3n + 3$

## Analysing Sequences by Difference Analysis

Suppose $a_0, a_1, a_2, a_3, a_4, \ldots$ is an infinite sequence of natural numbers.

Then $f = \lambda n.a_n$ is a function in $\mathbb{N} \longrightarrow \mathbb{N}$.

The function $f$ is a *polynomial function of degree* $k$ if $f$ can be presented in the form

$$c_k n^k + c_{k-1} n^{k-1} + \cdots + c_1 n + c_0,$$

with $c_i \in \mathbb{Q}$ and $c_k \neq 0$.

Example: the sequence

$$[1, 4, 11, 22, 37, 56, 79, 106, 137, 172, 211, 254, \ldots]$$

is given by the polynomial function

$f = \lambda n.(2n^2 + n + 1)$.

This is a function of the second degree.

```
f = \ n -> 2*n^2 + n + 1
```

```
MTG> take 12 (map f [0..])
[1,4,11,22,37,56,79,106,137,172,211,254]
```

Consider the *difference sequence* given by the function

$$d(f) = \lambda n.a_{n+1} - a_n.$$

```
difs :: [Integer] -> [Integer]
difs [] = []
difs [n] = []
difs (n:m:ks) = m-n : difs (m:ks)
```

```
MGT> difs [1,4,11,22,37,56,79,106,137]
[3,7,11,15,19,23,27,31]
```

Fact: The difference function $d(f)$ of a polynomial function $f$ is itself a polynomial function.

If $f = \lambda n.(2n^2 + n + 1)$, then:

$$\begin{aligned} d(f) &= \lambda n.(2(n+1)^2 + (n+1) + 1 - (2n^2 + n + 1) \\ &= \lambda n.4n + 3. \end{aligned}$$

```
MGT> take 12 (map (\n -> 4*n + 3) [0..])
[3,7,11,15,19,23,27,31,35,39,43,47]
MGT> take 12 (difs (map f [0..]))
[3,7,11,15,19,23,27,31,35,39,43,47]
```

Fact: if $f$ is a polynomial function of degree $k$ then $d(f)$ is a polynomial function of degree $k-1$.

For suppose $f(n)$ is given by

$$c_k n^k + c_{k-1} n^{k-1} + \cdots + c_1 n + c_0.$$

Then $d(f)(n)$ is given by

$$c_k(n+1)^k + c_{k-1}(n+1)^{k-1} + \cdots + c_1(n+1) + c_0 - (c_k n^k + c_{k-1} n^{k-1} + \cdots + c_1 n + c_0).$$

$f(n+1) - f(n)$ has no term with $n^k$.

Suppose $f$ is a polynomial function of degree $k$.

Then $d^k(f)$ will be a polynomial function of degree $0$.

A polynomial function of degree $0$ is a constant function.

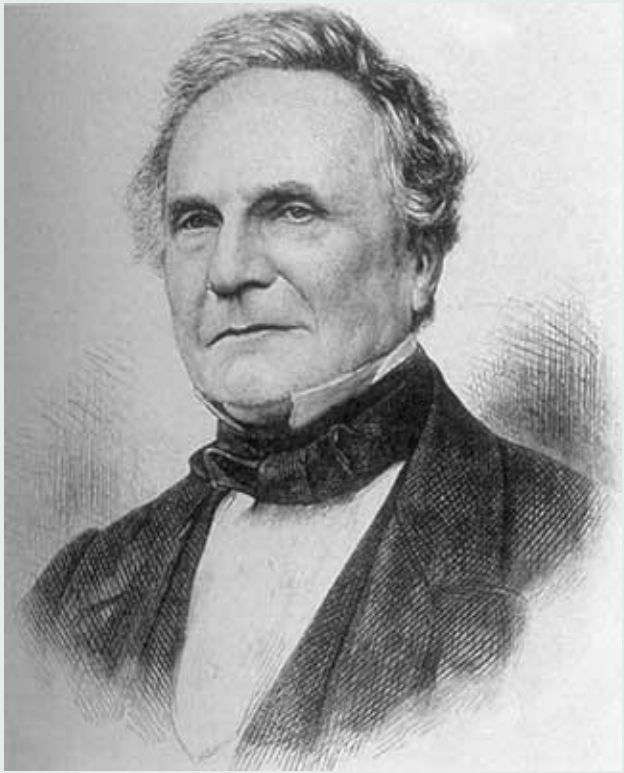
Example of computing difference sequences until we hit at a constant sequence:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **-12** | **-11** | **6** | **45** | **112** | **213** | **354** | **541** |
| 1 | 17 | 39 | 67 | 101 | 141 | 187 | |
| 16 | 22 | 28 | 34 | 40 | 46 | | |
| 6 | 6 | 6 | 6 | 6 | | | |

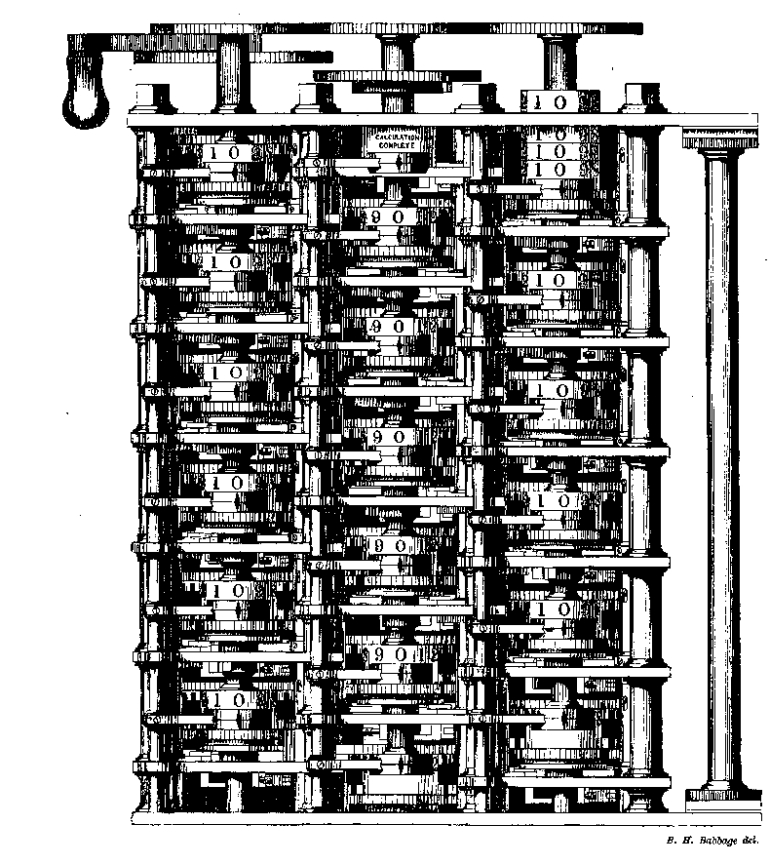
The sequence of third differences is constant, so the form of the original sequence is a polynomial of degree $3$.

To find the next number in the sequence, take the sum of the last elements of the rows.

## Charles Babbage (1791–1871)



# Charles Babbage's Difference Engine



If the input list has a polynomial form of degree $k$, then after $k$ steps of taking differences the list is reduced to a constant list:

```
MGT> difs [-12,-11,6,45,112,213,354,541]
[1,17,39,67,101,141,187]
MGT> difs [1,17,39,67,101,141,187]
[16,22,28,34,40,46]
MGT> difs [16,22,28,34,40,46]
[6,6,6,6,6]
```

```
difLists ::[[Integer]]->[[Integer]]
difLists [] = []
difLists l@(xs:xss) =
  if constant xs then l else
  difLists ((difs xs):l)
    where
    constant (n:m:ms) =
      all (==n) (m:ms)
    constant  _       =
      error "lack of data/not a pf"
```

This gives the lists of all the difference lists that were generated from the initial sequence, with the constant list upfront.

```
MGT> difLists [[-12,-11,6,45,112,213,354]]
[[6,6,6,6],
 [16,22,28,34,40],
 [1,17,39,67,101,141],
 [-12,-11,6,45,112,213,354]]
```

If a given list is generated by a polynomial, then the degree of the polynomial can be computed by difference analysis, as follows:

```
degree :: [Integer] -> Int
degree xs =
   length (difLists [xs]) - 1
```

The following function gets the list of last elements of the difference lists for a given sequence:

```
genDifs :: [Integer] -> [Integer]
genDifs xs =
   map last (difLists [xs])
```

A new list of last elements of difference lists is computed from the current one by keeping the constant element $d_1$, and replacing each $d_{i+1}$ by $d_i + d_{i+1}$.

```
nextD :: [Integer] -> [Integer]
nextD [] = error "no data"
nextD [n] = [n]
nextD (n:m:ks) = n : nextD (n+m:ks)
```

The next element of the original sequence is given by the last element of the new list of last elements of difference lists:

```
next :: [Integer] -> Integer
next = last . nextD . genDifs
```

```
MTG> next [-12,-11,6,45,112,213,354,541]
780
```

The difference engine is quite as intelligent as the members of any Society for the Highly Gifted:

```
MTG> next [1,5,14,30,55]
91

MTG> next [1,9,36,100,225,441])
784
```

## Gaussian Elimination

Difference analysis yields an algorithm for continuing any finite sequence with a polynomial form. Is it also possible to give an algorithm for finding the form?

The answer is ‘yes’, and the method is Gaussian elimination.

## Carl Friedrich Gauss (1777–1855)

## Example Application: Pie Cutting

## The Pie Cutting Sequence

The pie cutting sequence is:

$$1, 2, 4, 7, 11, \ldots$$

Difference analysis yields:

| | | | | |
|---|---|---|---|---|
| **1** | | **2** | | **4** | | **7** | | **11** |
| | 1 | | 2 | | 3 | | 4 | |
| | | 1 | | 1 | | 1 | | |

Thus, the polynomial is of the second degree, it has form $an^2 + bn + c$. We have to find $a, b, c$.

We know:

$$\begin{aligned} c &= 1 \\ a + b + c &= 2 \\ 4a + 2b + c &= 4 \end{aligned}$$

Elimination of $c$ gives:

$$\begin{aligned} a + b &= 1 \\ 4a + 2b &= 3 \end{aligned}$$

$$\begin{aligned} a + b &= 1 \\ 4a + 2b &= 3 \end{aligned}$$

$$\begin{aligned} 4a + 4b &= 4 \\ 4a + 2b &= 3 \\ \hline 2b &= 1 \end{aligned}$$

We get: $a = \frac{1}{2}$, $b = \frac{1}{2}$, and $c = 1$.

The form is: $\frac{1}{2}n^2 + \frac{1}{2}n + 1 = \frac{n(n+1)}{2} + 1$.

## Generalization

Third degree:

$$\begin{aligned} d &= a_0 \\ a+b+c+d &= a_1 \\ 8a+4b+2c+d &= a_2 \\ 27a+9b+3c+d &= a_3 \end{aligned}$$

Fourth degree:

$$\begin{aligned} e &= a_0 \\ a+b+c+d+e &= a_1 \\ 16a+8b+4c+2d+e &= a_2 \\ 81a+27b+9c+3d+e &= a_3 \\ 256a+64b+16c+4d+e &= a_4 \end{aligned}$$

Fifth degree:

$$
\begin{aligned}
f &= a_0 \\
a + b + c + d + e + f &= a_1 \\
32a + 16b + 8c + 4d + 2e + f &= a_2 \\
243a + 81b + 27c + 9d + 3e + f &= a_3 \\
1024a + 256b + 64c + 16d + 4e + f &= a_4 \\
3125a + 625b + 125c + 25d + 5e + f &= a_5
\end{aligned}
$$

## Matrix Manipulation

Solving sets of linear equations can be viewed as manipulation of matrices of coefficients.

The quadruple of linear equations in $a, b, c, d$ for a polynomial of the third degree gives the following matrix:

$$\begin{pmatrix} 0 & 0 & 0 & 1 & a_0 \\ 1 & 1 & 1 & 1 & a_1 \\ 8 & 4 & 2 & 1 & a_2 \\ 27 & 9 & 3 & 1 & a_3 \end{pmatrix}$$

To solve this, we transform it to an equivalent matrix in so-called *echelon form* or *left triangular form*, i.e., a matrix of the form:

$$\begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} & b_0 \\ 0 & a_{11} & a_{12} & a_{13} & b_1 \\ 0 & 0 & a_{22} & a_{23} & b_2 \\ 0 & 0 & 0 & a_{33} & b_3 \end{pmatrix}$$

From this form, compute the value of variables $d$, $c$, $b$, $a$ by *backward substitution*.

## Implementation

```
type Matrix = [Row]
type Row    = [Integer]

rows, cols :: Matrix -> Int
rows m = length m
cols m | m == []    = 0
       | otherwise = length (head m)
```

The function `genMatrix` produces the appropriate matrix for a list generated by a polynomial:

```
genMatrix :: [Integer] -> Matrix
genMatrix xs =
  zipWith (++) (genM d)
               [[x]| x <- xs ]
   where
   d      = degree xs
   genM n = [[(toInteger x^(n-m))
              | m <- [0..n]]
                | x <- [0..n] ]
```

```
MTG> genMatrix [-7,-2,15,50,109,198,323]
[[0,0,0,1,-7],
 [1,1,1,1,-2],
 [8,4,2,1,15],
 [27,9,3,1,50]]
```

Transformation to echelon form: use one row to eliminate the first coefficient from the other rows by means of *adjustment*:

```
adjustWith :: Row -> Row -> Row
adjustWith (m:ms) (n:ns) =
   zipWith (-) (map (n*) ms)
               (map (m*) ns)
```

This is used in:

```
echelon   :: Matrix -> Matrix
```

```
MTG> echelon [[0,0,0,1,-7],
              [1,1,1,1,-2],
              [8,4,2,1,15],
              [27,9,3,1,50]]
[[1,1,1,1,-2],
 [0,4,6,7,-31],
 [0,0,12,22,-142],
 [0,0,0,-48,336]]
```

The function for computing the values of the variables is backsubst.

```
MTG> backsubst [[1,1,1,1,-2],
                [0,4,6,7,-31],
                [0,0,12,22,-142],
                [0,0,0,-48,336]]
[1 % 1,3 % 1,1 % 1,-7 % 1]
```

To use all this to analyze a polynomial sequence:

1. find the degree of the polynomial by difference analysis,
2. generate the appropriate matrix,
3. put it in echelon form,
4. compute the values of the unknowns by backward substitution.

```
solveSeq :: [Integer] -> [Rational]
solveSeq =
  backsubst . echelon . genMatrix
```

The sequence for the pie cutting process yields:

```
MTG> solveSeq [1,2,4,7,11]
[1 % 2,1 % 2,1 % 1]
```

## What has this to do with Intelligence?

We have managed to implement the intelligent task of guessing next elements of polynomial sequences.

We have also implemented the intelligent task of analysing polynomial sequences to find the form of the generating polynomial function.

The computer is *much* better at these tasks than any human.

Does the task of building intelligent machines consist of programming more and more of these tricks (plus 'meta-tricks' for selecting the right candidate from the bag of tricks)?

"Yes": You are an AI behaviourist.

"No": You are an AI essentialist.

## What has this to do with Consciousness?

Nothing.

Consciousness is first-person-singular.

Consciousness can only be experienced.

Consciousness is the root of all experience.

Whatever is object of attention is *never* consciousness.

## Consciousness = See for Yourself

Some experiments suggested by
Douglass Harding, ‘The Headless Way’

**Douglas Harding (1909– )**

**Pointing here.**

Point to your friend's feet, then yours; to his legs, then yours; to his torso, then yours; to his head, then? What, on present evidence, is your finger now pointing at?

**Single eye.**

In your own experience at this moment, are you peering through two little holes in a kind of meat-ball? If so, what's it like in there - dark, stuffy, congested, small? Slowly put on a pair of spectacles and notice how those two little 'windows' become one vast 'window' - spotlessly clean and with nobody looking out of it.

**Putting on a no-face.**

Cut a head-sized hole in a card. Hold the card out at arm's length, noting the hole's boundaries. See how they vanish into your boundlessness as you bring the card forward and put it right on - to your face?

**Paper bag.**

Get an ordinary bag (preferably white) about 12" square, and cut the bottom off. Fit your face into one end while your friend fits his into the other. How many faces are given in the bag? Dropping memory and imagination, are you face-to-face or face-to-no-face?

**In the body?**

By stroking and pinching and pummelling, try to build up here on your shoulders the sort of thing you see over there on your friend's shoulders. Now try to get inside it, and describe its contents. Aren't you still out-of-doors, as much at large as ever? Look at your hand. Are you in it, or is it in you?

# Further Reading