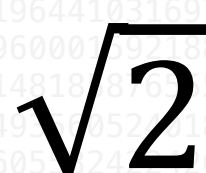
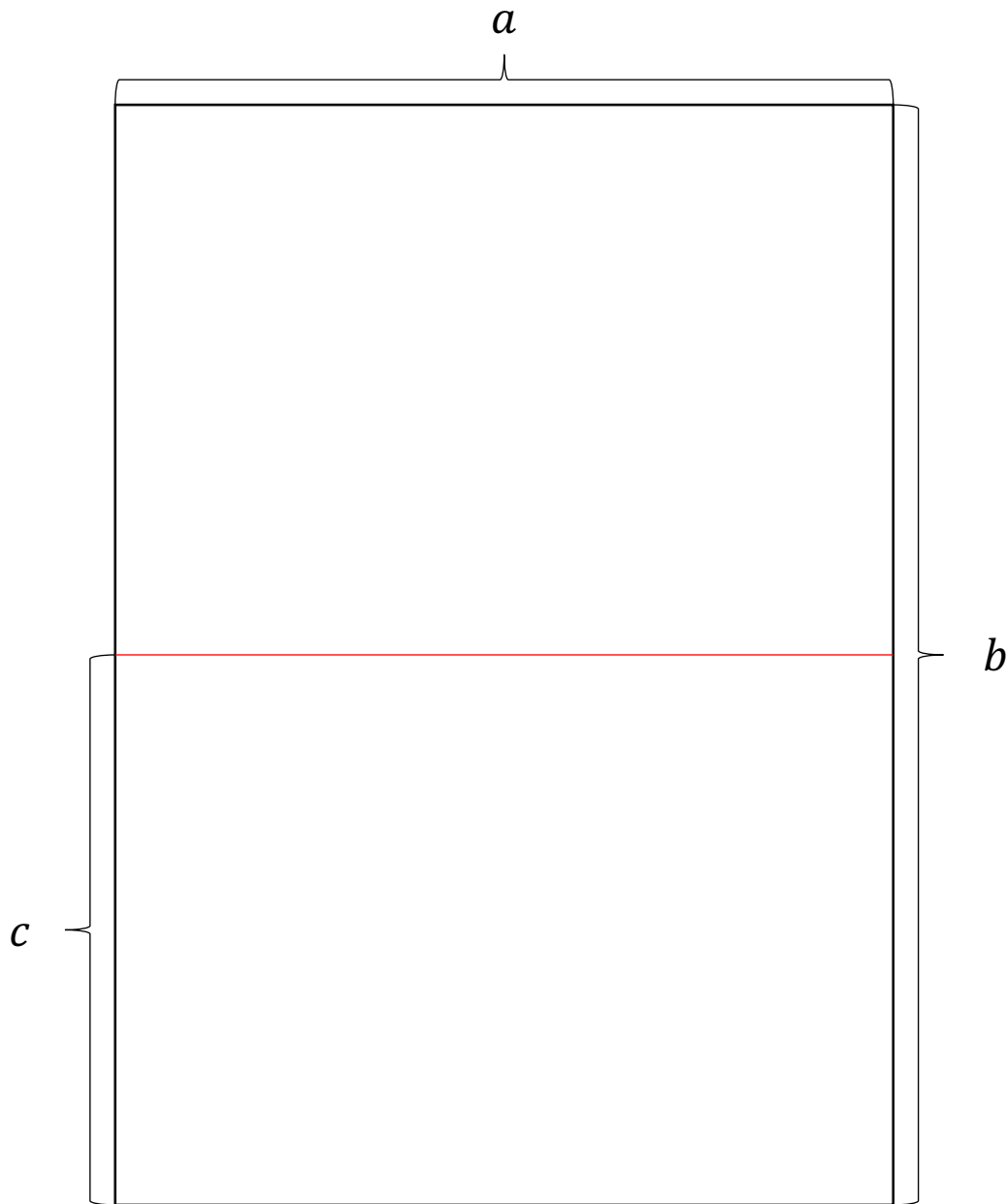


Noah Schlenker

Leon Baptist Kniffki

Christian Krinitsin





Wofür ist $\sqrt{2}$ nötig?

- Gesucht: Seitenverhältnis x
- $b = xa, a = xc, c = \frac{1}{2}b$
- $b = xxc = x^2c$
- $b = \frac{x^2}{2}b$
- $1 = \frac{x^2}{2}$
- $2 = x^2$
- $x = \sqrt{2}$
- Durch das Seitenverhältnis $1:\sqrt{2}$ bleibt das Format beim Halbieren erhalten

Wie bestimmt man $\sqrt{2}$?

$$\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}^n = \begin{pmatrix} x_{n-1} & x_n \\ x_n & x_{n+1} \end{pmatrix} \Rightarrow \lim_{n \rightarrow \infty} 1 + \frac{x_n}{x_{n+1}} = \sqrt{2}$$

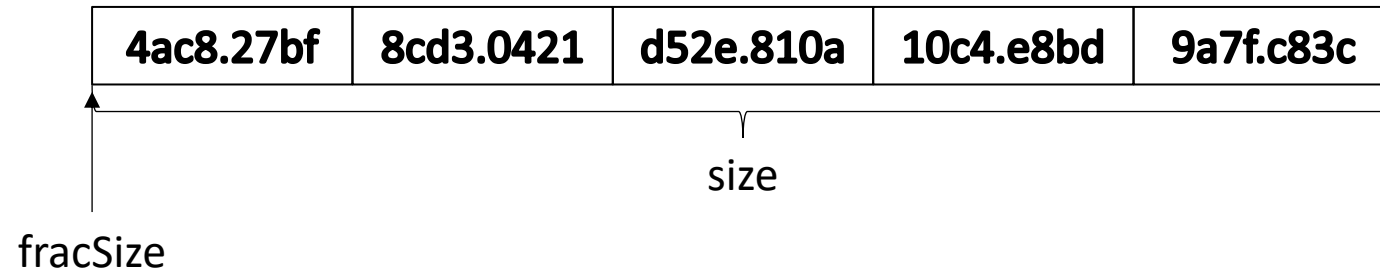
Wie bestimmt man $\sqrt{2}$?

$$\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}^n = \begin{pmatrix} x_{n-1} & x_n \\ x_n & x_{n+1} \end{pmatrix} \Rightarrow \lim_{n \rightarrow \infty} 1 + \frac{x_n}{x_{n+1}} = \sqrt{2}$$

- Probleme:
 - Beliebig große Zahlen
 - Exponentiationen dauern lange
 - Division beliebig genauer Zahlen

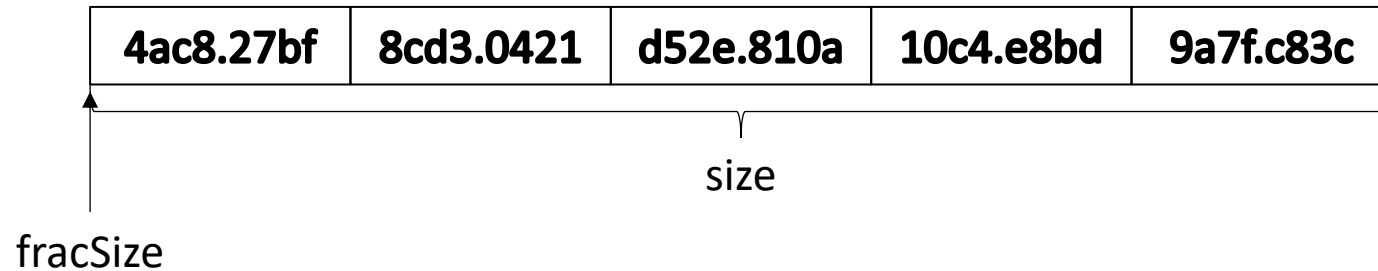
Zahlen mit beliebiger Genauigkeit

- Die Fixkommazahl



Zahlen mit beliebiger Genauigkeit

- Die Fixkommazahl

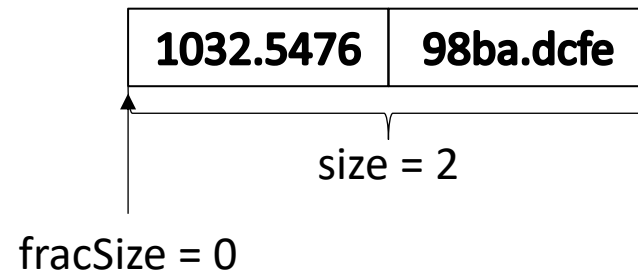


- Implementierung in C:
 - Little-Endian wie in x86-64

```
struct bignum {  
    uint32_t *digits;  
    size_t size;  
    size_t fracSize;  
};
```

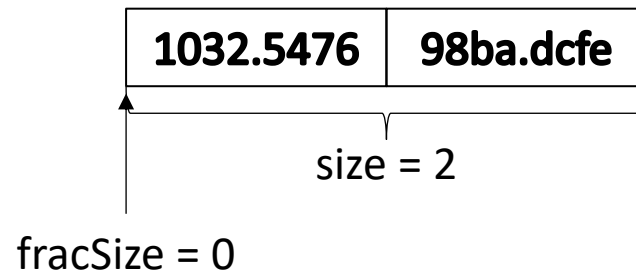
Zahlen mit beliebiger Genauigkeit

- 0xfedc.ba98.7654.3210

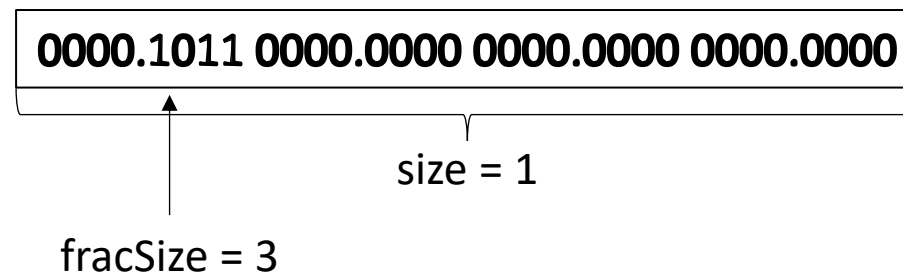


Zahlen mit beliebiger Genauigkeit

- 0xfedc.ba98.7654.3210



- 1,375 = 1,011



Grundlegende Arithmetik

Einfach

Addition

$$a + b = \sum_{i=0}^{size-1} 2^{32i} (a_i + b_i)$$

Multiplikation

$$a \cdot b = \sum_{i=0}^{size-1} \left(2^{32i} b_i \sum_{j=0}^{size-1} 2^{32j} a_j \right)$$

Grundlegende Arithmetik

Einfach

Mit SIMD

Addition

$$a + b = \sum_{i=0}^{size-1} 2^{32i} (a_i + b_i)$$

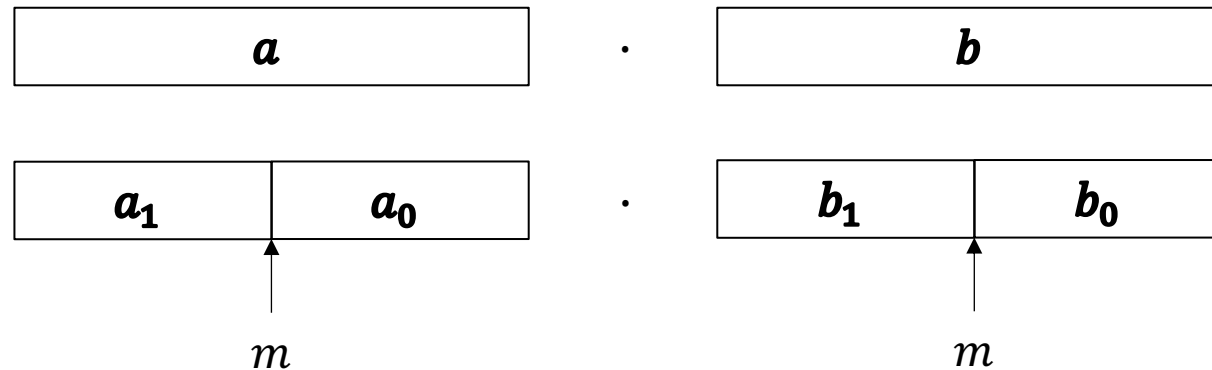
$$= \sum_{i=0}^{\lfloor \frac{size}{4} \rfloor - 1} 2^{128i} (a_i + b_i) + \sum_{i=size-size\%4}^{size-1} 2^{32i} (a_i + b_i)$$

Multiplikation

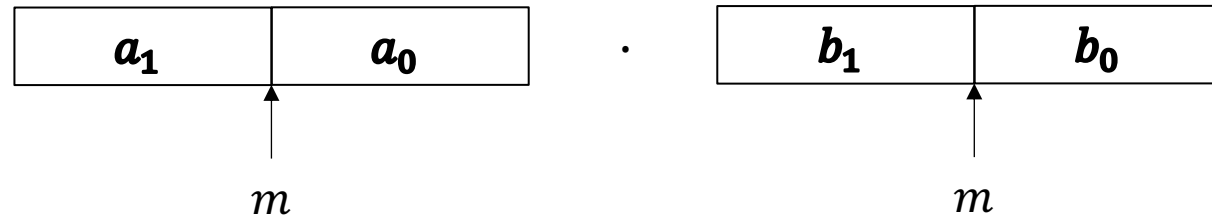
$$a \cdot b = \sum_{i=0}^{size-1} \left(2^{32i} b_i \sum_{j=0}^{size-1} 2^{32j} a_j \right) = \sum_{i=0}^{\lfloor \frac{size}{2} \rfloor - 1} \left(2^{64i} b_i \sum_{j=0}^{\lfloor \frac{size}{2} \rfloor - 1} 2^{64j} a_j + \sum_{j=size-size\%2}^{size-1} \dots \right) + \dots$$

Karazuba – Eine bessere Multiplikation

- Russische Bauernmultiplikation liegt in $O(n^2)$
- Eine bessere Methode:
 - Divide and conquer
 - Wähle m , sodass $a = a_0 + 2^m a_1$ und $b = b_0 + 2^m b_1$

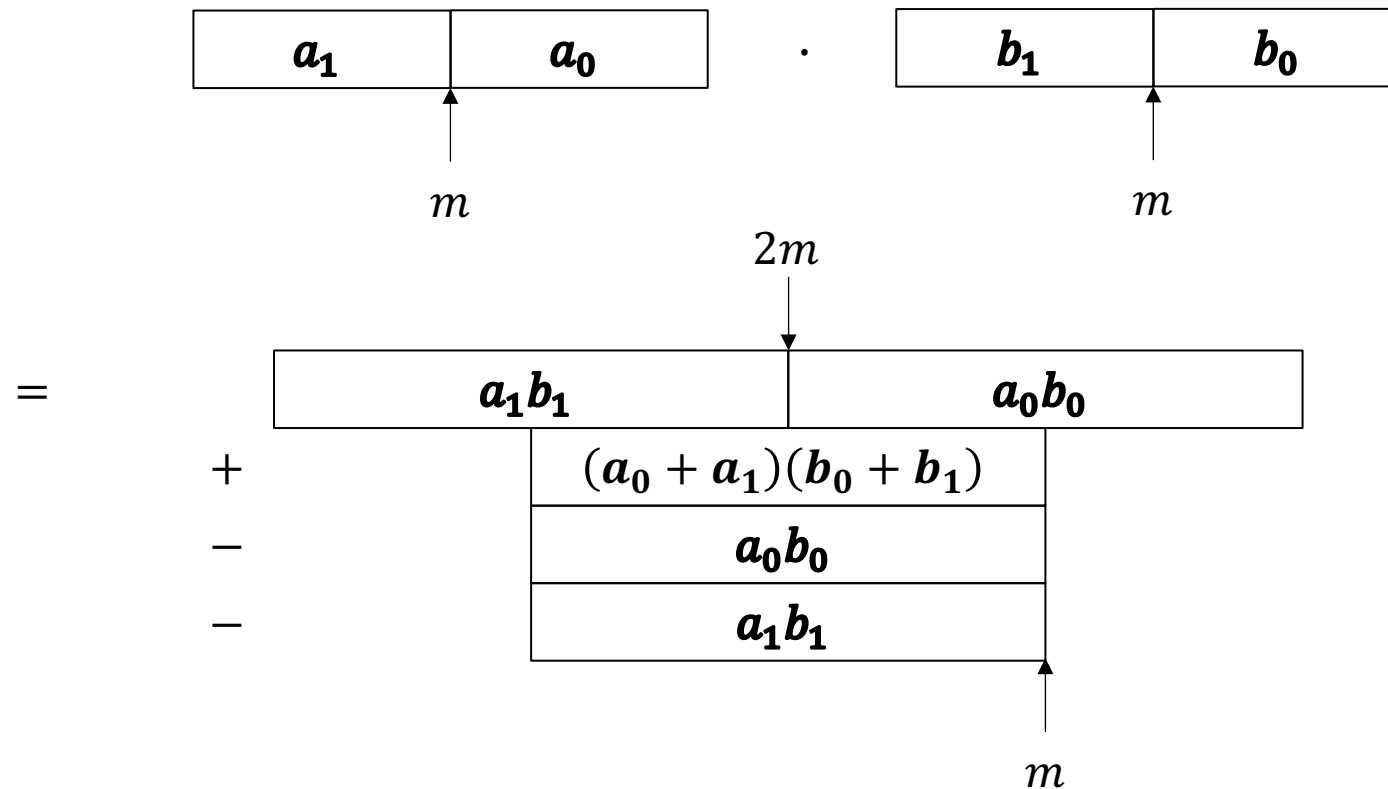


Karazuba – Eine bessere Multiplikation



$$\begin{aligned}(a_0 + 2^m a_1)(b_0 + 2^m b_1) &= a_0 b_0 + 2^m (a_0 b_1 + a_1 b_0) + 2^{2m} a_1 b_1 \\ &= a_0 \cdot b_0 + 2^m \left((a_0 + a_1) \cdot (b_0 + b_1) - a_0 b_0 - a_1 b_1 \right) + 2^{2m} a_1 \cdot b_1\end{aligned}$$

Karazuba – Eine bessere Multiplikation



Die Matrixmultiplikation

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} & a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} & a_{21} \cdot b_{12} + a_{22} \cdot b_{22} \end{pmatrix}$$

Acht teure Multiplikationen

Die Matrixmultiplikation

$$\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{pmatrix} = \begin{pmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} & a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} & a_{21} \cdot b_{12} + a_{22} \cdot b_{22} \end{pmatrix}$$

Acht teure Multiplikationen

$$\begin{pmatrix} a_1 & a_2 \\ a_2 & a_3 \end{pmatrix} \begin{pmatrix} a_1 & a_2 \\ a_2 & a_3 \end{pmatrix} = \begin{pmatrix} a_1 \cdot a_1 + a_2 \cdot a_2 & a_1 \cdot a_2 + a_2 \cdot a_3 \\ \mathbf{a_2 \cdot a_1} + \mathbf{a_3 \cdot a_2} & \mathbf{a_2 \cdot a_2} + a_3 \cdot a_3 \end{pmatrix}$$

Nur noch fünf Multiplikationen

Mit $\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}$

$$\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} a_1 & a_2 \\ a_2 & a_3 \end{pmatrix} = \begin{pmatrix} a_2 & a_3 \\ a_1 + 2a_2 & a_2 + 2a_3 \end{pmatrix}$$

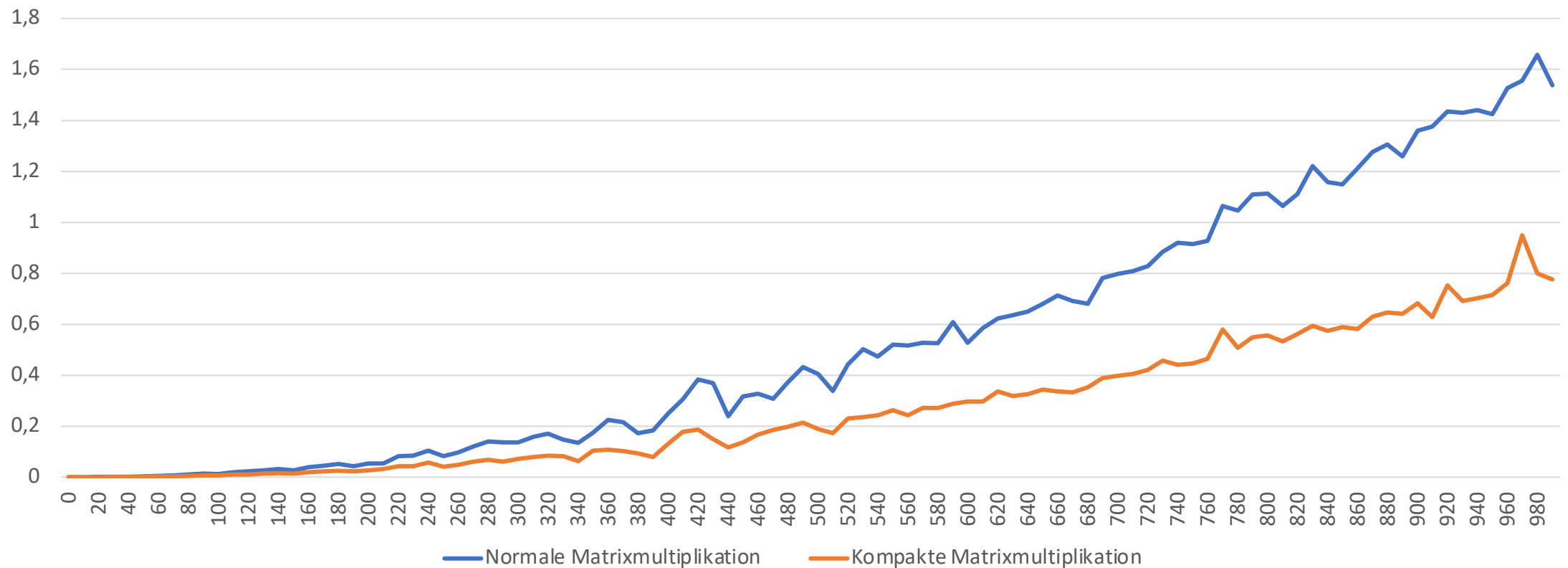
Durch Bitshifts und Addition realisierbar

$$\begin{aligned} x_0 &= 0, x_1 = 1 \\ x_n &= 2x_{n-1} + x_{n-2} \end{aligned}$$

Vier Multiplikationen

Die Matrixmultiplikation

Laufzeitentwicklungen im Vergleich



Die schnelle Exponentiation

- Iteratives Aufmultiplizieren benötigt $O(n)$ Matrixmultiplikationen
- Schneller: Wiederholtes quadrieren der Basis

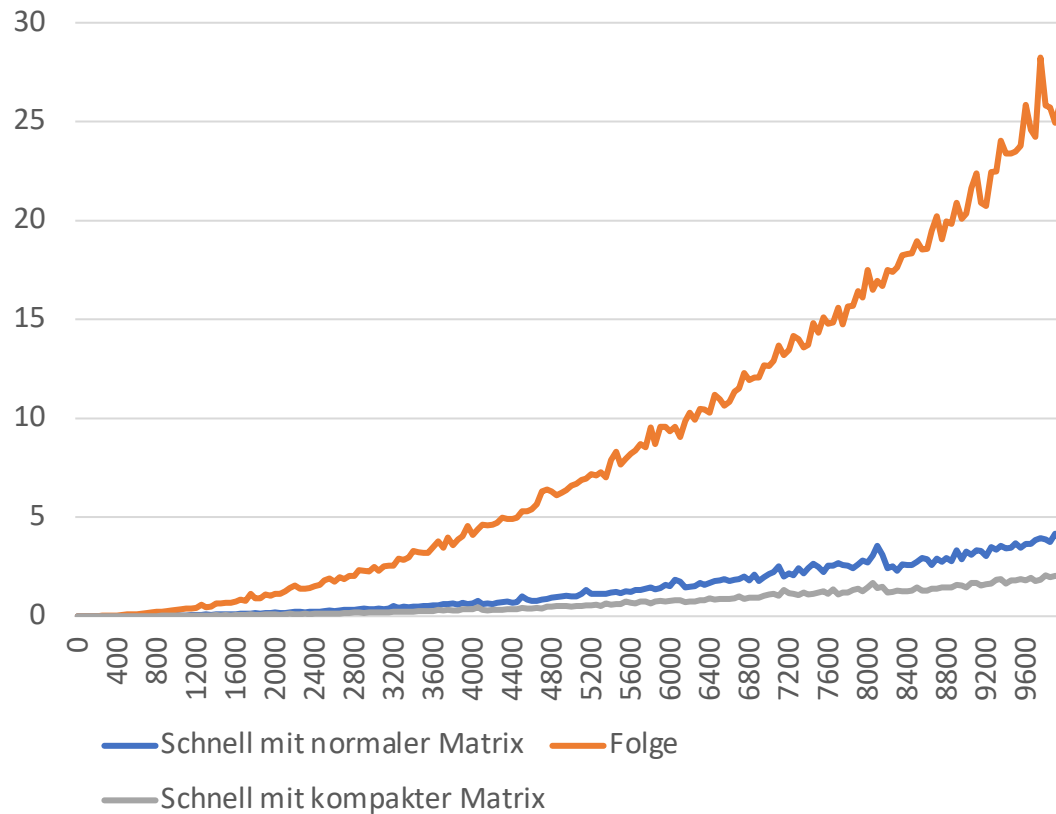
$$\begin{aligned} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}^6 &= \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \\ &= \left(\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \right) \left(\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \right) \end{aligned}$$

Die schnelle Exponentiation

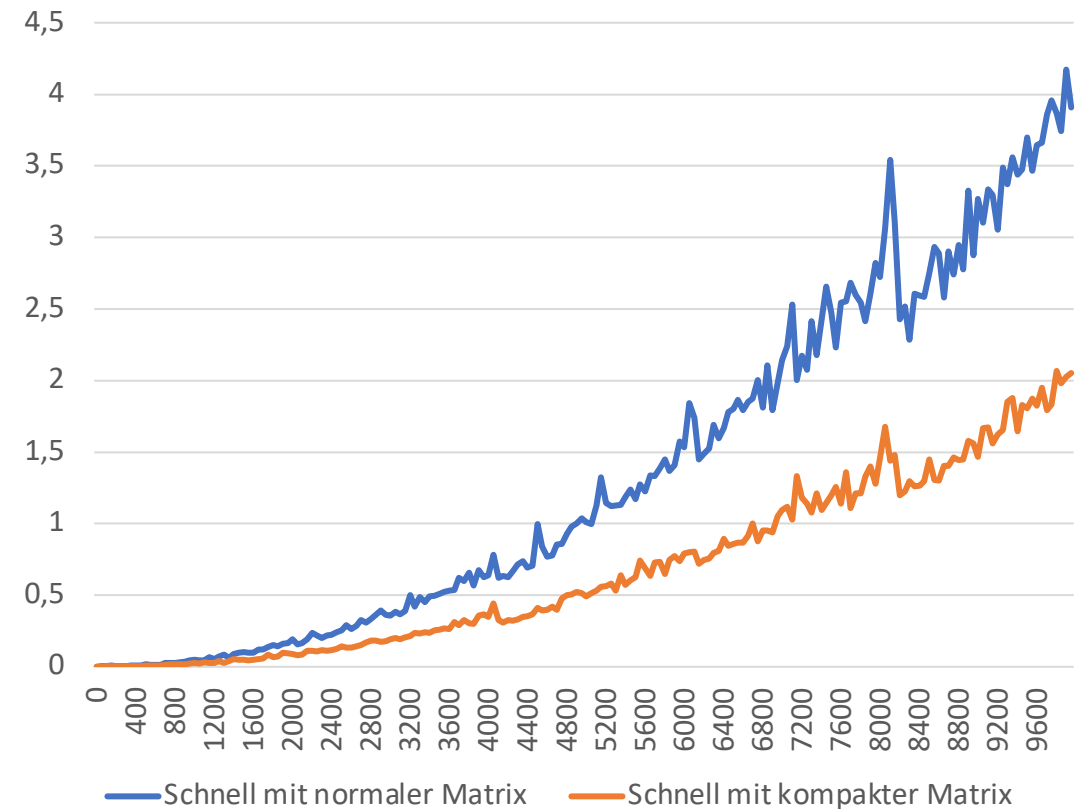
$$\begin{aligned}\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}^6 &= \left(\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \right) \left(\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix} \right) \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}^2 \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}^4 \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}^2 \left(\begin{pmatrix} 0 & 1 \\ 1 & 2 \end{pmatrix}^2 \right)^2\end{aligned}$$

Die schnelle Exponentiation

Laufzeitentwicklungen



Laufzeitentwicklungen



Division von Bignums

Berechnung von $\sqrt{2}$

Beliebige Genauigkeit

