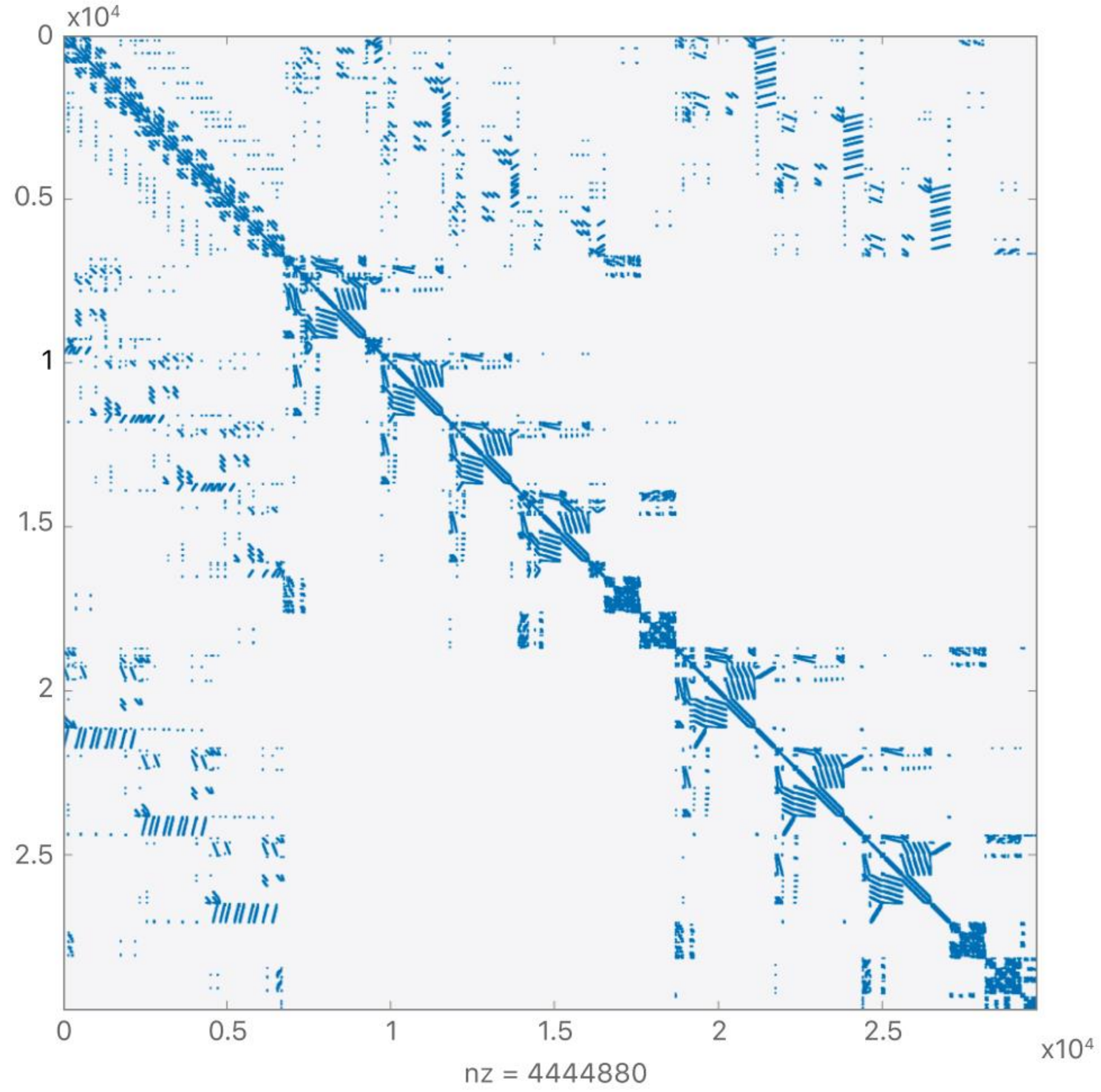
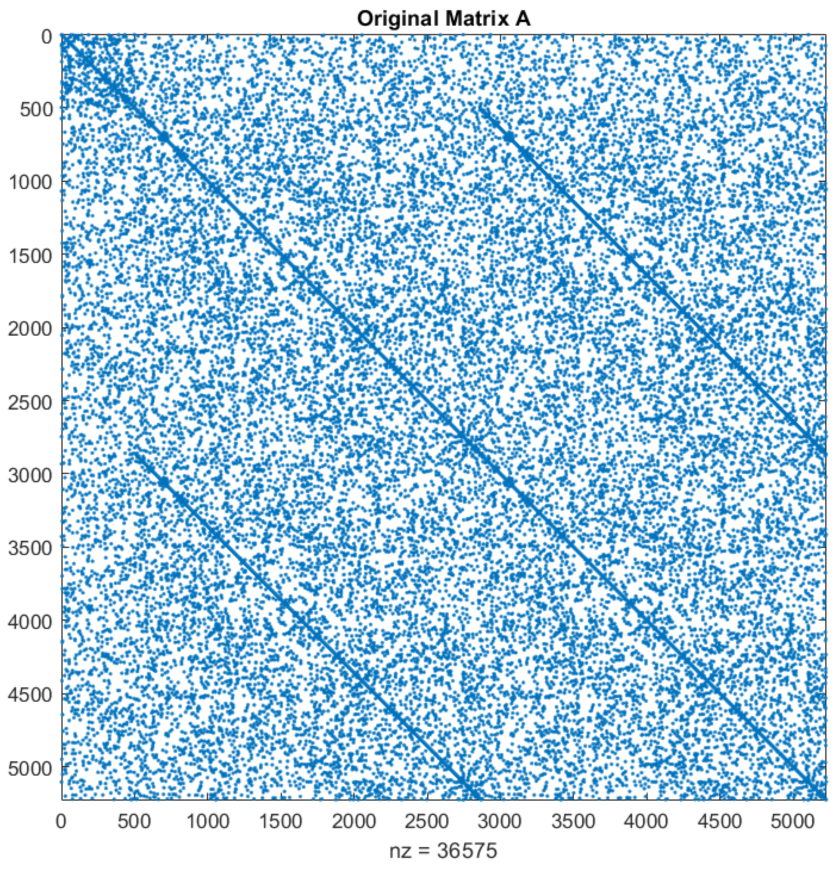
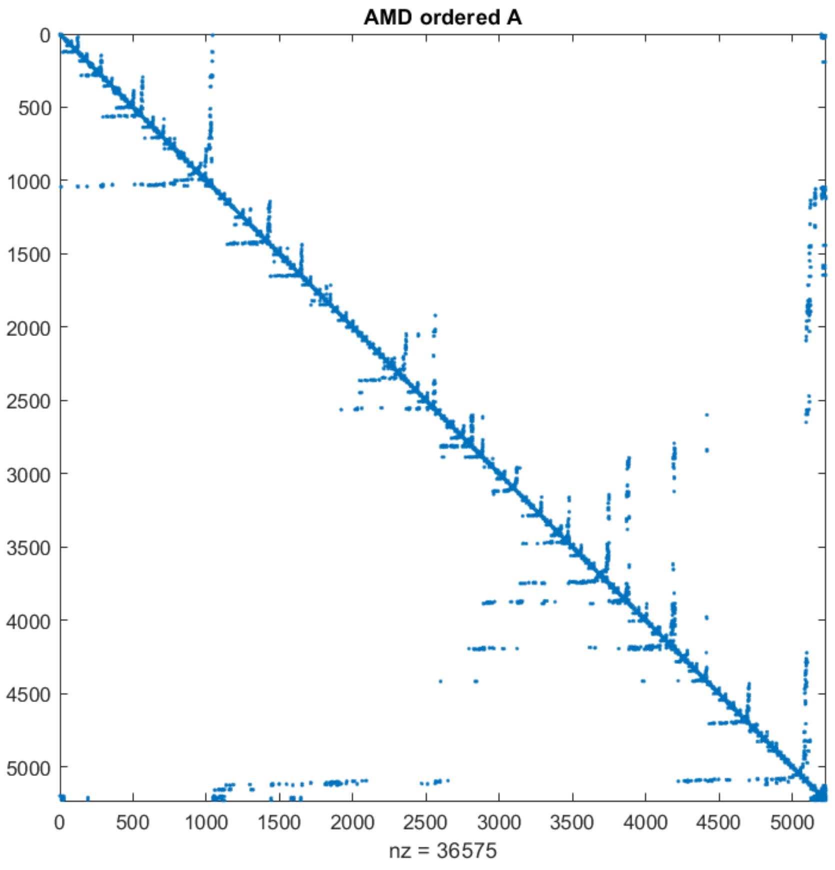
Multiplikation dünnbesetzter Matrizen: CSR (A310) | T016

Projektaufgabe – Aufgabenbereich Algorithmik

Erdemoglu, Eremenko, Kronast-Reichert

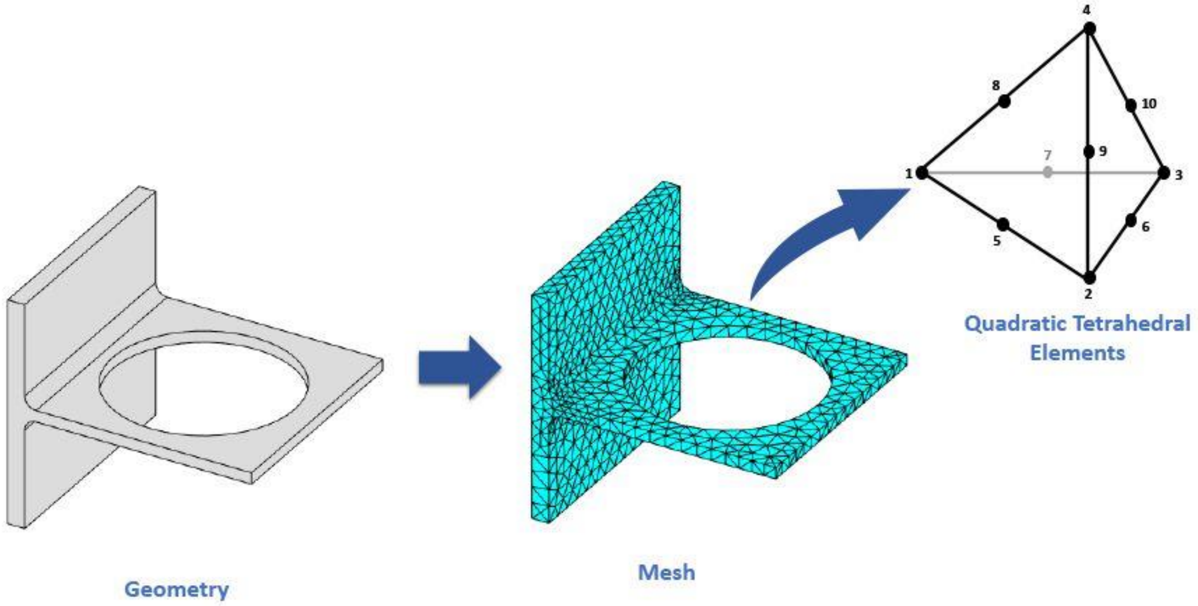


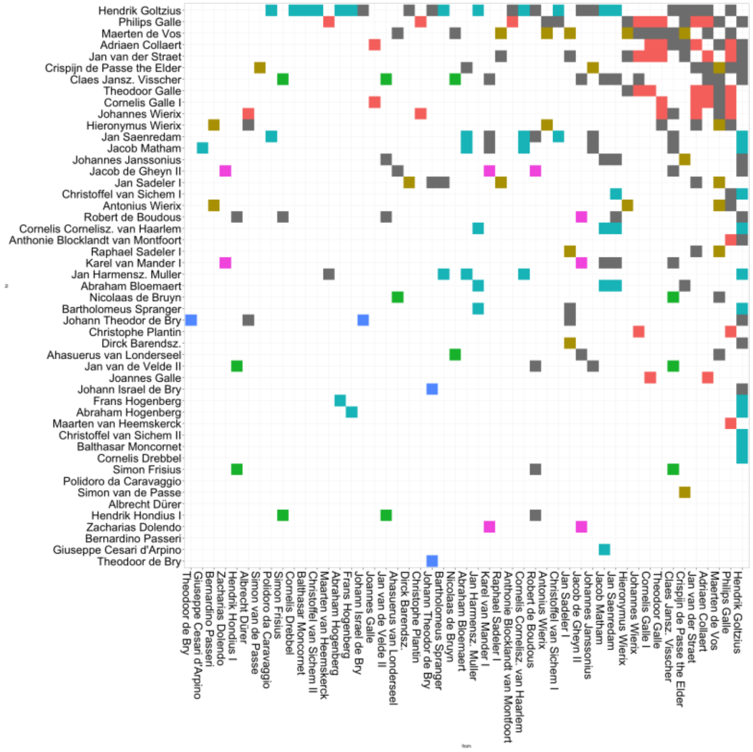

Original Matrix A
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
nz = 36575


AMD ordered A
0
500
1000
1500
2000
2500
3000
3500
4000
4500
5000
nz = 36575

Relevanz

- Netzwerkmodellierung → Adjazenzmatrizen
- Numerische Simulationen
- Machine Learning





Aufgabe und Problematik

- Effiziente Matrixmultiplikation in CSR

CSR

$$\begin{pmatrix} 5 & 0 & 0 & 0 \\ 0 & 6 & 0 & 0 \\ 0.5 & 7 & 0 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

```
LINE  |  CONTENT
   1  |  <noRows>,<noCols>
   2  |  <values>
   3  |  <col_indices>
   4  |  <row_ptr>
```

```
4,4
5,6,0.5,7,3
0,1,0,1,3
0,1,2,4,5
```

Aufgabe und Problematik

- Effiziente Matrixmultiplikation in CSR

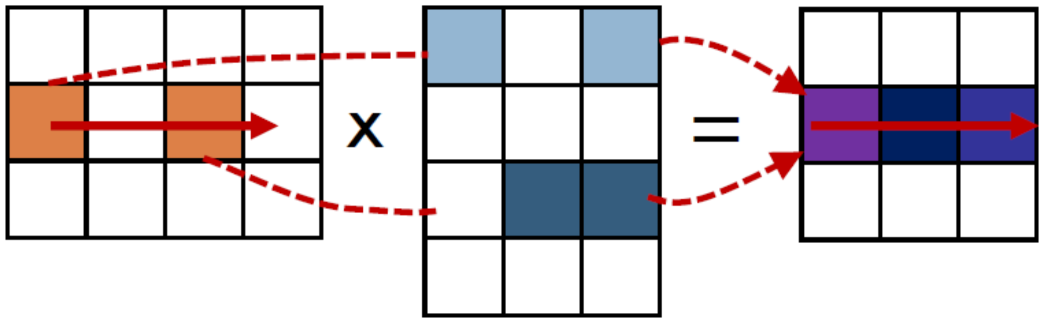
- Arbeit mit CSR nicht trivial
- Abschätzung der Matrixdichte
- Dynamische Auswahl des Algorithmus
- Berechnung großer Matrizen

Lösungsansatz

- Gustavson-Algorithmus
- Abschätzung des Speicherbedarfs
- Cacheoptimierung durch Buckets
- SIMD
- Multithreading

Gustavson
dataflow

```
for m in [0, M)
 for k in [0, K)
  for n in [0, N)
   C[m,n] += A[m,k] * B[k,n]
```



Number-non-zero prediction (nnz)

$$\begin{pmatrix}1 & 0 & 0\\1 & 0 & 0\\1 & 0 & 0\end{pmatrix} * \begin{pmatrix}1 & 1 & 1\\0 & 0 & 0\\0 & 0 & 0\end{pmatrix} = \begin{pmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{pmatrix}$$

$$\begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix} * \begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix} = \begin{pmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{pmatrix}$$

$$\begin{pmatrix}1 & 1 & 0\\1 & 1 & 0\\0 & 0 & 1\end{pmatrix} * \begin{pmatrix}1 & 1 & 0\\1 & 1 & 0\\0 & 0 & 1\end{pmatrix} = \begin{pmatrix}2 & 2 & 0\\2 & 2 & 0\\0 & 0 & 1\end{pmatrix}$$

[2] Abbildung 4.1

	Abschätzung	Tatsächlich	Abweichung
a)	$3*3+0*0+0*0=9$	9	0
b)	$0*3+3*0+0*0=0$	0	0
c)	$1*1+1*1+1*1=3$	3	0
d)	$2*2+2*2+1*1=9$	5	4
e)	$1*1+2*2+1*1=6$	5	1

[2] Abbildung 4.2

Multithreading

Konzept

$$
\begin{array}{l}
\text{T-1} \\
\text{T-2} \\
\text{T-3}
\end{array}
\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}
$$

values: [1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1]
cols: [0 , 1 , 2 , 0 , 1 , 2 , 0 , 1 , 2]
rowPtrs: [0 , 3 , 6 , 9]

Workload-Balancer

$$
\begin{array}{l}
\text{T-1} \\
\text{T-1} \\
\text{T-2} \\
\text{T-2} \\
\text{T-3} \\
\text{T-3}
\end{array}
\begin{pmatrix} \text{row-1} \\ \text{row-2} \\ \text{row-3} \\ \text{row-4} \\ \text{row-5} \\ \text{row-6} \end{pmatrix}
$$

SIMD

Folgendes Beispiel bezieht sich auf 128-Bit SSE Register.

$$\begin{pmatrix} 1 & 0 & 2 & 3 \\ 9 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 1 & 0 & 3 & 8 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 1 & 2 & 6 & 8 & 9 & 4 & 5 & 6 & 2 \\ 3 & 4 & 0 & 5 & 3 & 4 & 2 & 3 & 1 & 4 & 0 \\ 5 & 6 & 7 & 8 & 0 & 1 & 0 & 0 & 0 & 2 & 3 \\ 0 & 3 & 2 & 1 & 2 & 1 & 3 & 4 & 1 & 0 & 0 \end{pmatrix} = \begin{pmatrix} 11 & 21 & 21 & 21 & 12 & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots \\ 0 & 0 & 0 & 0 & 0 & \dots \end{pmatrix}$$

SIMD

SEQUENTIELL

Floats werden vereinfacht als 32-Bit Integer dargestellt.

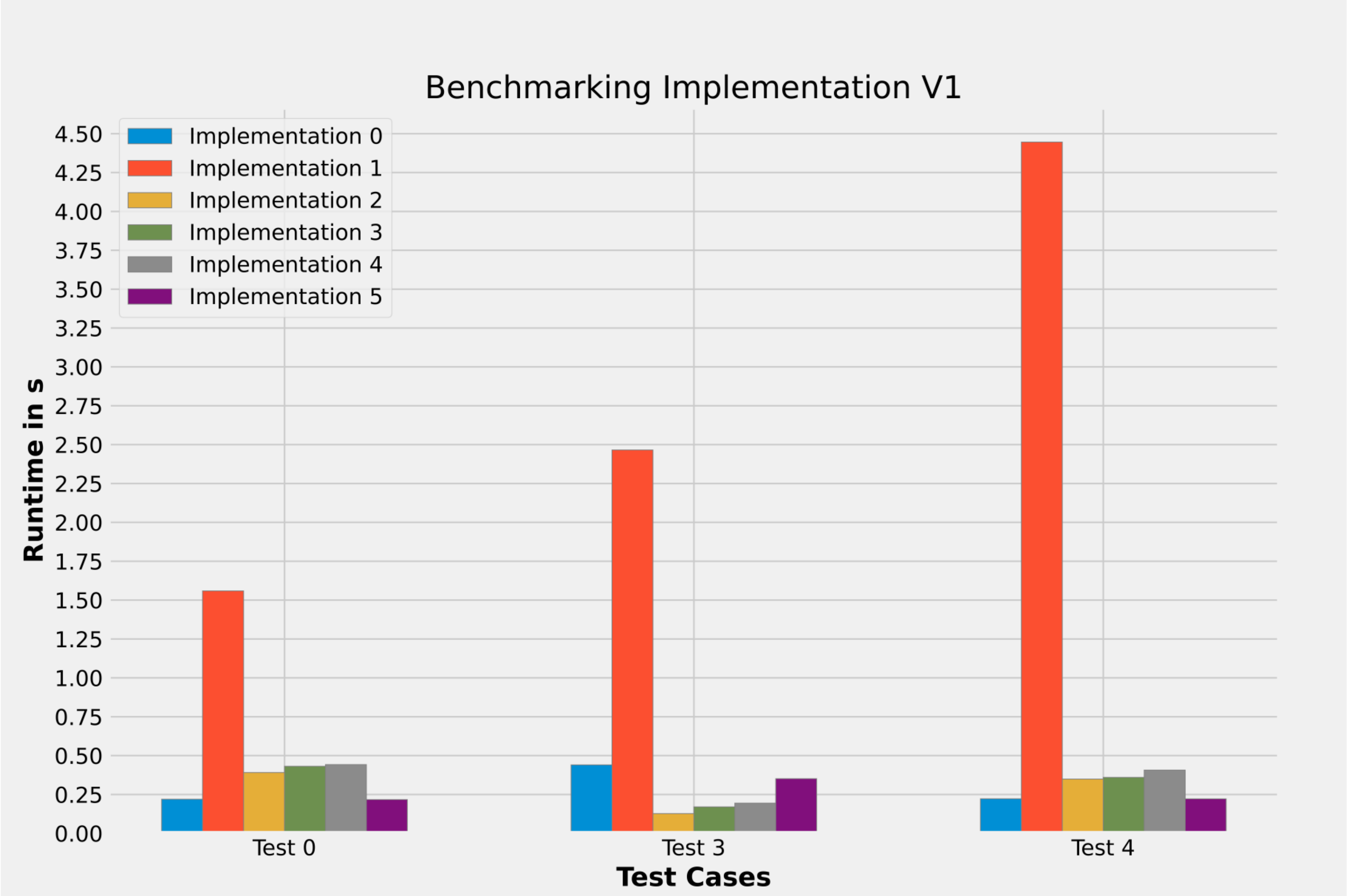
Implementierungen

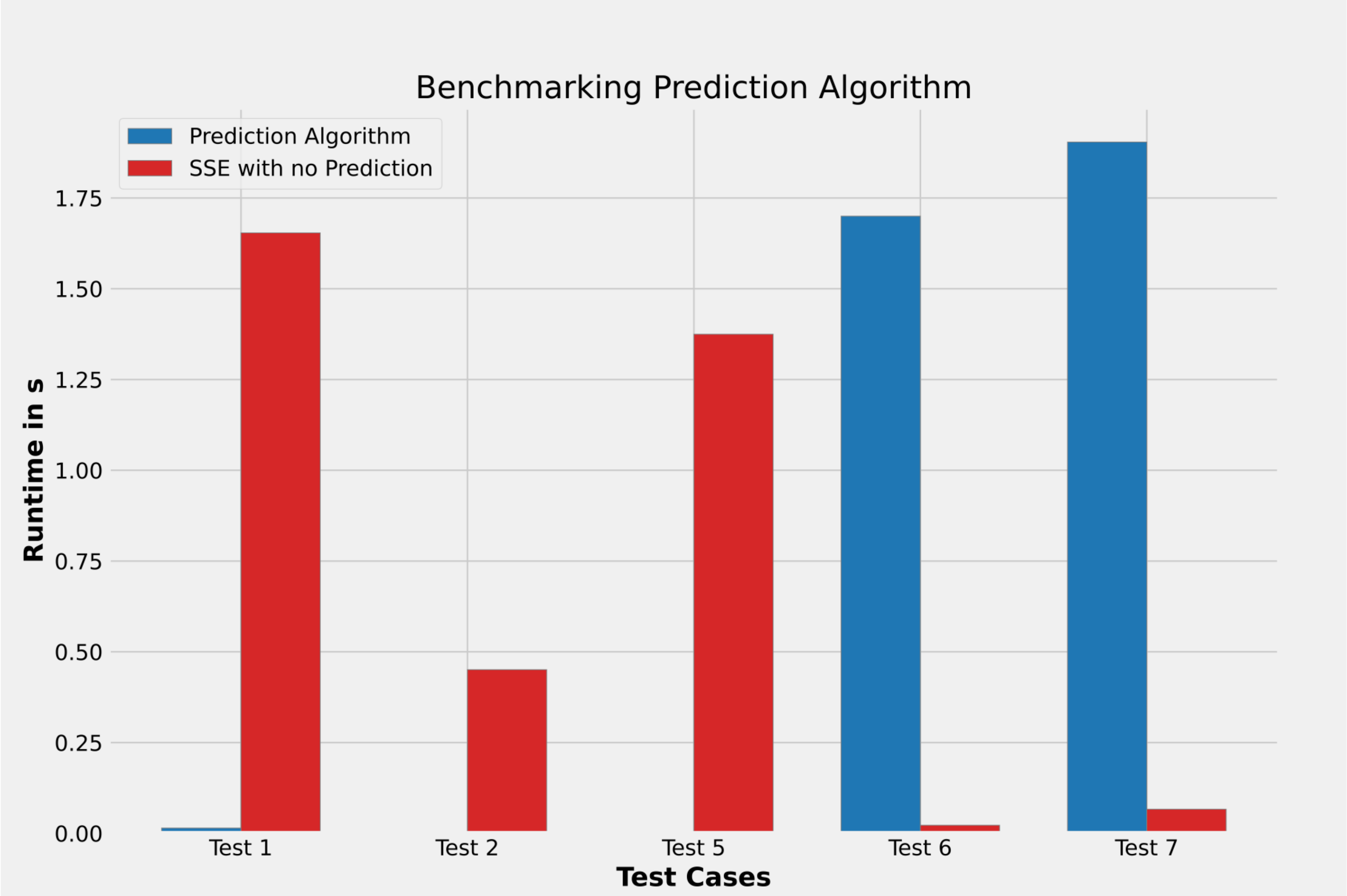
- **V0:** Gustavson-Algorithmus **Multithreading, Strategy Pattern**
- **V1:** Konvertierung (CSR ↔ dichtes Format), vollbesetzte SISD Multiplikation
- **V2:** Gustavson-Algorithmus **SISD**
- **V3:** Gustavson-Algorithmus **SIMD (SSE)**
- **V4:** Gustavson-Algorithmus **SIMD (AVX)**
- **V5:** Gustavson-Algorithmus **Größenabschätzung**

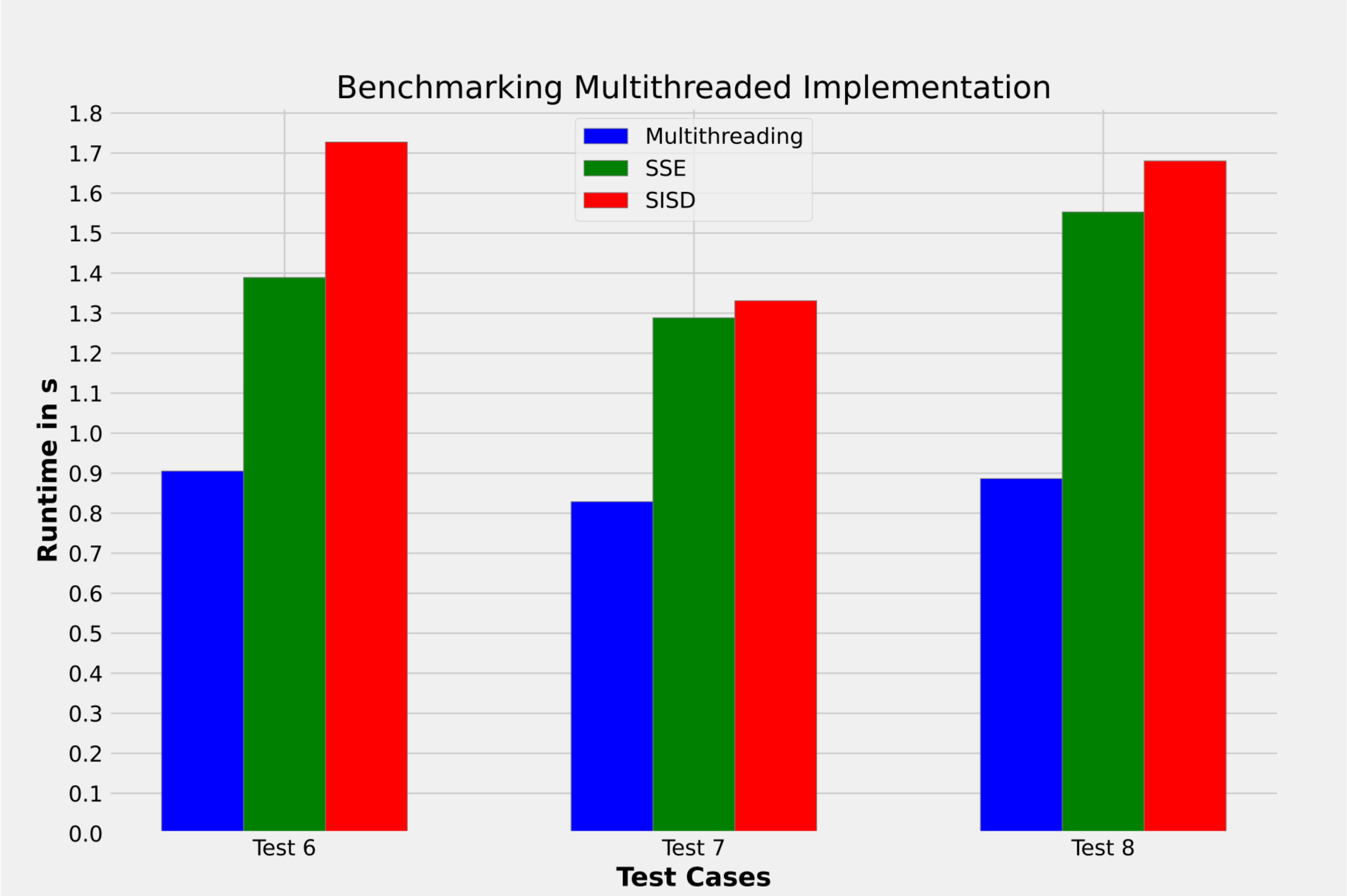
Benchmarking

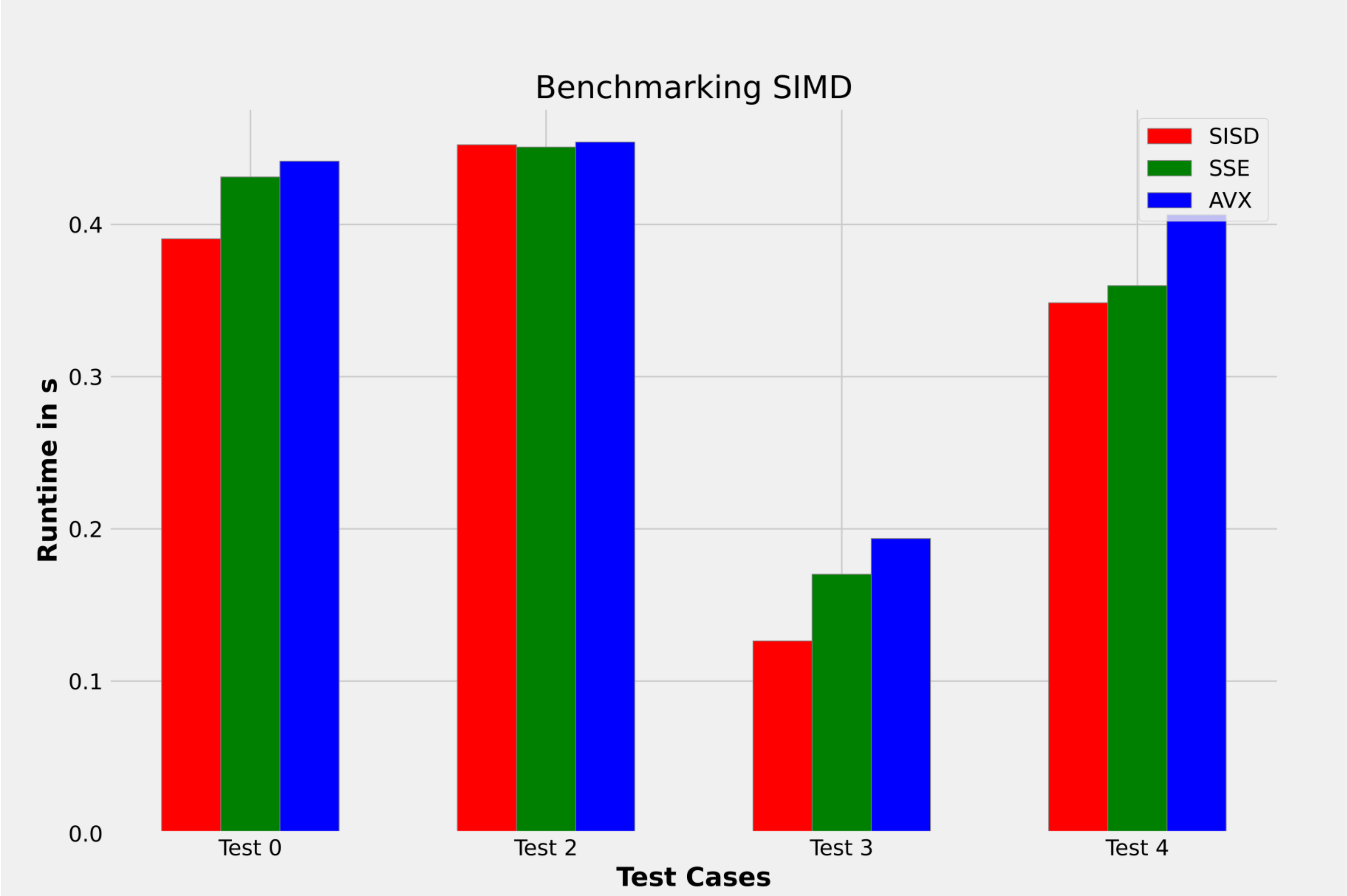
Benchmarking

- Implementierung V1
- Abschätzungsalgorithmus
- Multithreading
- SIMD vs. SISD

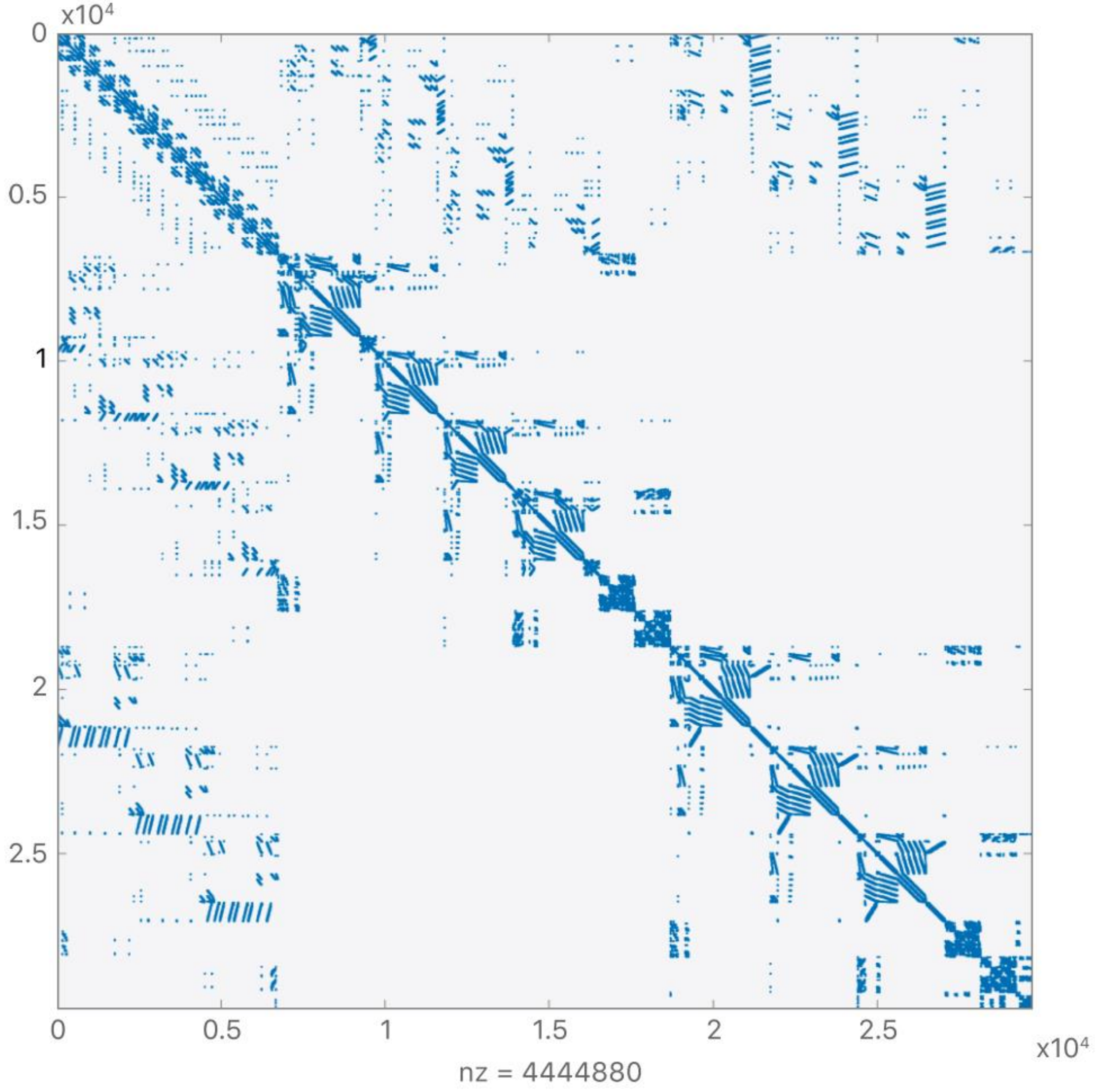

Benchmarking Implementation V1
Implementation 0
Implementation 1
Implementation 2
Implementation 3
Implementation 4
Implementation 5
Runtime in s
4.50
4.25
4.00
3.75
3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Test 0
Test 3
Test 4
Test Cases


Benchmarking Prediction Algorithm
Prediction Algorithm
SSE with no Prediction
Runtime in s
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Test 1
Test 2
Test 5
Test 6
Test 7
Test Cases


Benchmarking Multithreaded Implementation
Multithreading
SSE
SISD
Runtime in s
1.8
1.7
1.6
1.5
1.4
1.3
1.2
1.1
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Test 6
Test 7
Test 8
Test Cases


Benchmarking SIMD
SISD
SSE
AVX
0.4
0.3
0.2
0.1
0.0
Runtime in s
Test 0
Test 2
Test 3
Test 4
Test Cases

Auswertung



Quellen

- [1]
- M. D. Lincoln, „Adjacency matrix plots with R and ggplot2“, Matthew Lincoln, PhD. Zugegriffen: 21. Juli 2024. [Online]. Verfügbar unter: https://matthewlincoln.net/2014/12/20/adjacency-matrix-plots-with-r-and-ggplot2.html
- [2]
- T. Scharpff, „Analyse und Optimierung von Operationen auf dünn besetzten Matrizen“, 2012. [Online]. Verfügbar unter: https://www10.cs.fau.de/publications/theses/2012/Scharpff_SA_2012.pdf
- [3]
- G. Zhang, N. Attaluri, J. S. Emer, und D. Sanchez, „Gamma: leveraging Gustavson’s algorithm to accelerate sparse matrix multiplication“, in *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, Virtual USA: ACM, Apr. 2021, S. 687–701. doi: 10.1145/3445814.3446702.
- [4]
- N. Srivastava, H. Jin, J. Liu, D. Albonesi, und Z. Zhang, „MatRaptor: A Sparse-Sparse Matrix Multiplication Accelerator Based on Row-Wise Product“, in *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, Athens, Greece: IEEE, Okt. 2020, S. 766–780. doi: 10.1109/MICRO50266.2020.00068.
- [5]
- Luben Alexandrov, „Parallel Sparse Matrix-Matrix Multiplication (Master’s Thesis)“, Institute of Theoretical Informatics, Algorithmics Department of Informatics Karlsruhe Institute of Technology, 2014. [Online]. Verfügbar unter: https://publikationen.bibliothek.kit.edu/1000128898/100477434
- [6]
- Arpad Bürmen, „Sparse matrices, solving large systems of linear equations“, Circuit Analysis and Optimization. Zugegriffen: 21. Juli 2024. [Online]. Verfügbar unter: https://zrojec.fe.uni-lj.si/cao/sparse-matrices-solving-large-systems-of-linear-equations/
- [7]
- „What Is Finite Element Analysis?“, Mathworks. Zugegriffen: 21. Juli 2024. [Online]. Verfügbar unter: https://kr.mathworks.com/discovery/finite-element-analysis.html