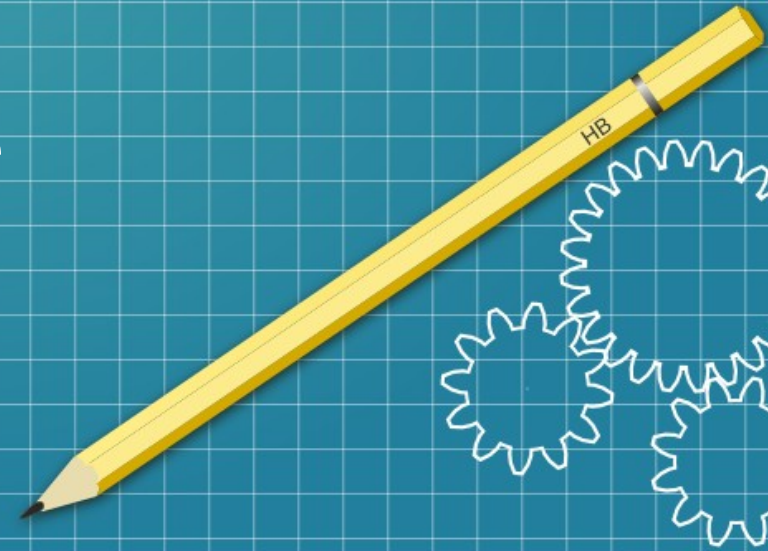


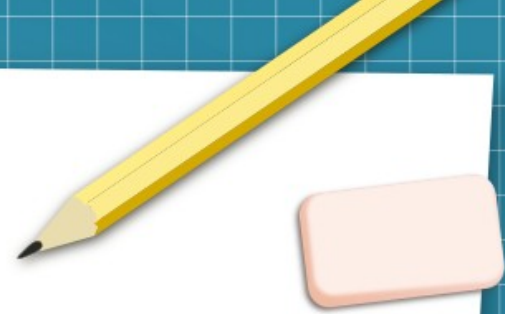
Tridiagonallöser

Ferdinand Vanmaele



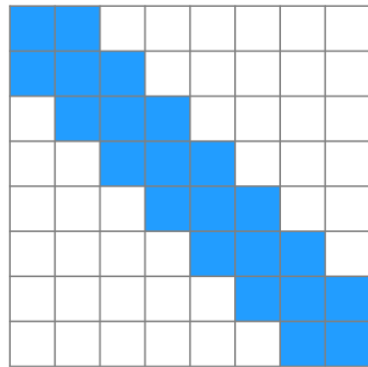
Einführung

- Paralleles Lösen von Tridiagonalsystemen
 - Naiv: Thomas-Algorithmus, $O(n)$ Operationen

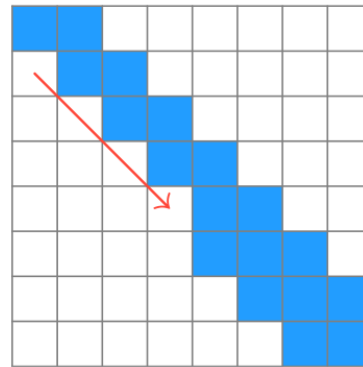


Einführung

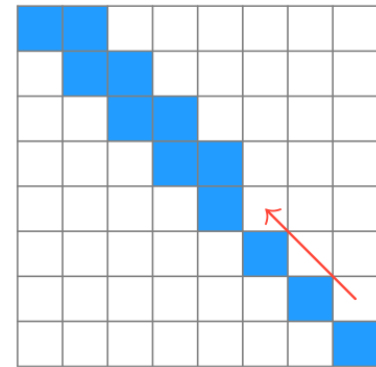
- Paralleles Lösen von Tridiagonalsystemen $Ax = b$
 - Naiv: Thomas-Algorithmus, $O(n)$ Operationen
 - Eliminiere untere Diagonale, dann Rückwärtseinsetzen
 - **Sequentiell!**



(a) Initial tridiagonal system



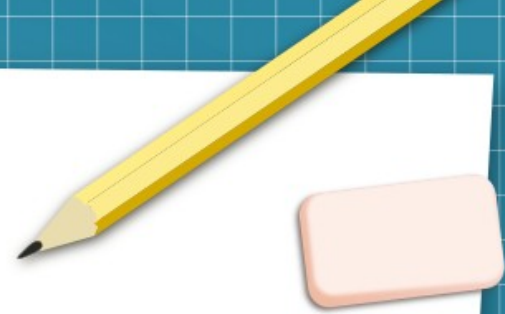
(b) Downwards oriented elimination phase



(c) Upwards oriented substitution phase

Einführung

- Paralleles Lösen von Tridiagonalsystemen $Ax = b$
 - Parallele Algorithmen?
 - Redundante Operationen

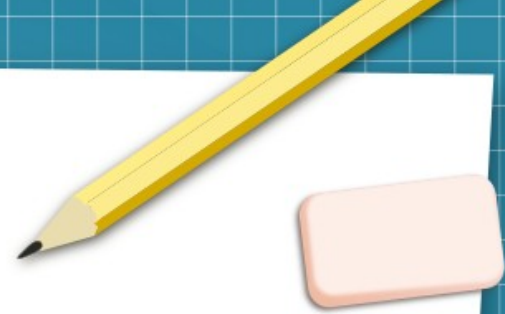


-
- A 7x7 grid with a blue diagonal line from the top-left to the bottom-right.

(c) Second reduction step.
The green entries form a system of two unknowns, which can be solved directly.

Einführung

- Paralleles Lösen von Tridiagonalsystemen $Ax = b$
 - Parallele Algorithmen?
 - Redundante Operationen
 - Cyclic Reduction
 - **Divide and Conquer**
 - Rekursive Aufteilung



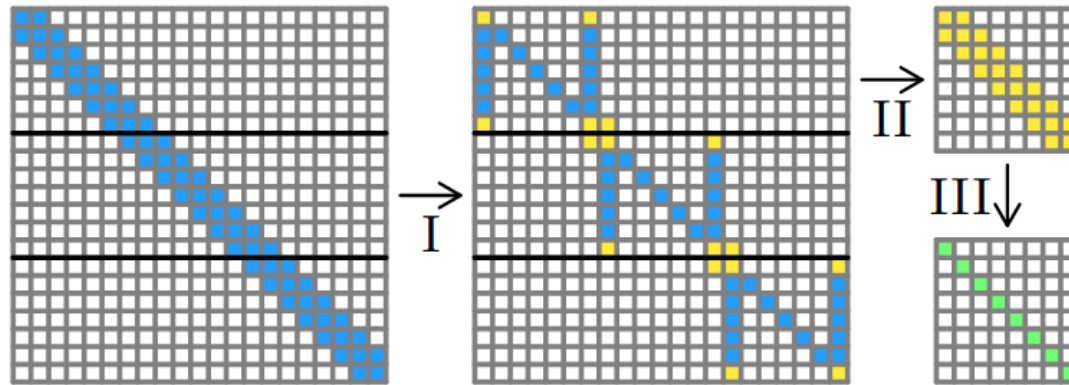
RPTS

- **Recursive Partitioned Tridiagonal Schur Complement [1]**
 - Divide and Conquer
 - Bilden von “Spikes”

[1] C. Klein, R. Strzodka. Tridiagonal GPU Solver with Scaled Partial Pivoting at Maximum Bandwidth, ICPP 2021

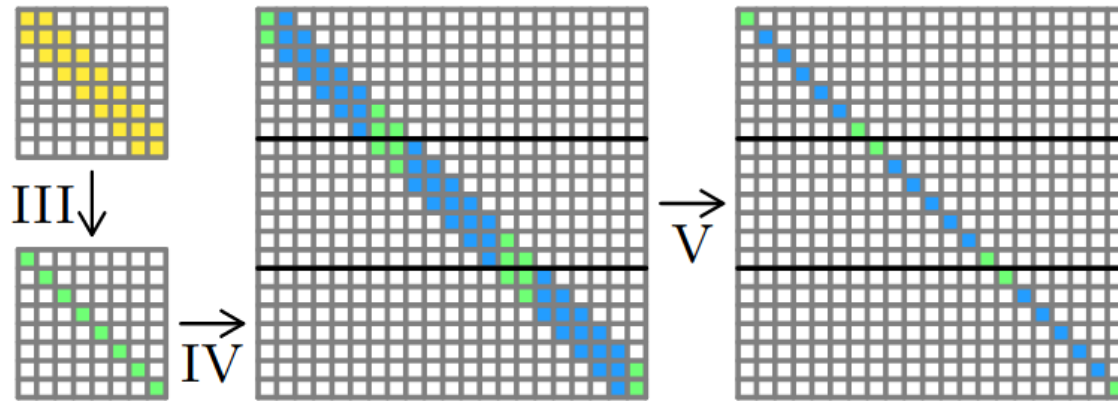
RPTS

- **Recursive Partitioned Tridiagonal Schur Complement** [1]
 - Reduction step

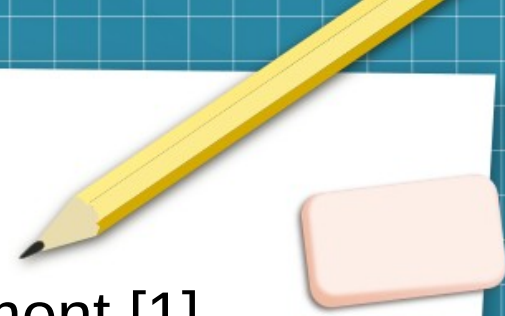


RPTS

- **Recursive Partitioned Tridiagonal Schur Complement** [1]
 - Substitution step



RPTS

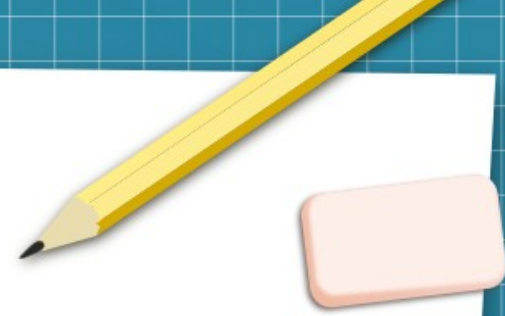


- **Recursive Partitioned Tridiagonal Schur Complement [1]**
 - Setze **rekursiv** fort, bis System klein genug
 - **Pivotisierung** für bessere Stabilität
 - Koeffizientmatrix A: **Partitions Grenzen?**

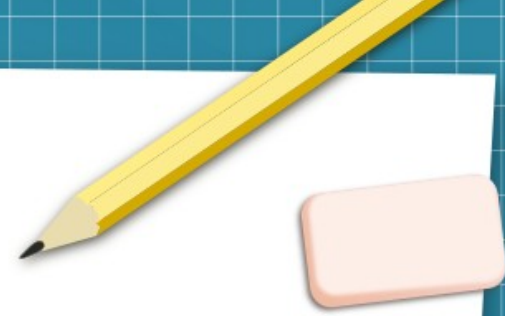
[1] C. Klein, R. Strzodka. Tridiagonal GPU Solver with Scaled Partial Pivoting at Maximum Bandwidth, ICPP 2021

RPTS – Partitionierung

- Feste Partitionierung
 - Billig, aber ggf. ungünstig gewählt

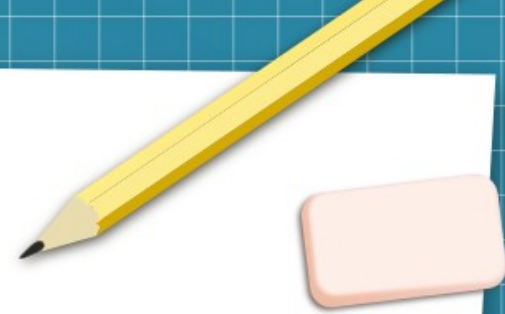


RPTS – Partitionierung



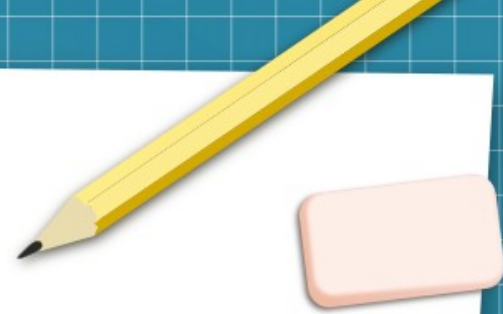
- Feste Partitionierung
 - Billig, aber ggf. ungünstig gewählt
- **Dynamisch erzeugt**
 - Verringern der *Kondition* des groben Systems
 - Kondition groß => kleiner Fehler in b erzeugt großer Fehler in Lsg. X
 - **Ziel:** $\text{Kond}(\text{GS})$ nicht größer als $\text{Kond}(A)$
 - Ggf. teuer
 - Verwende Grenzen mehrfach ($Ax = b_1, b_2, \dots$)

RPTS – Partitionierung

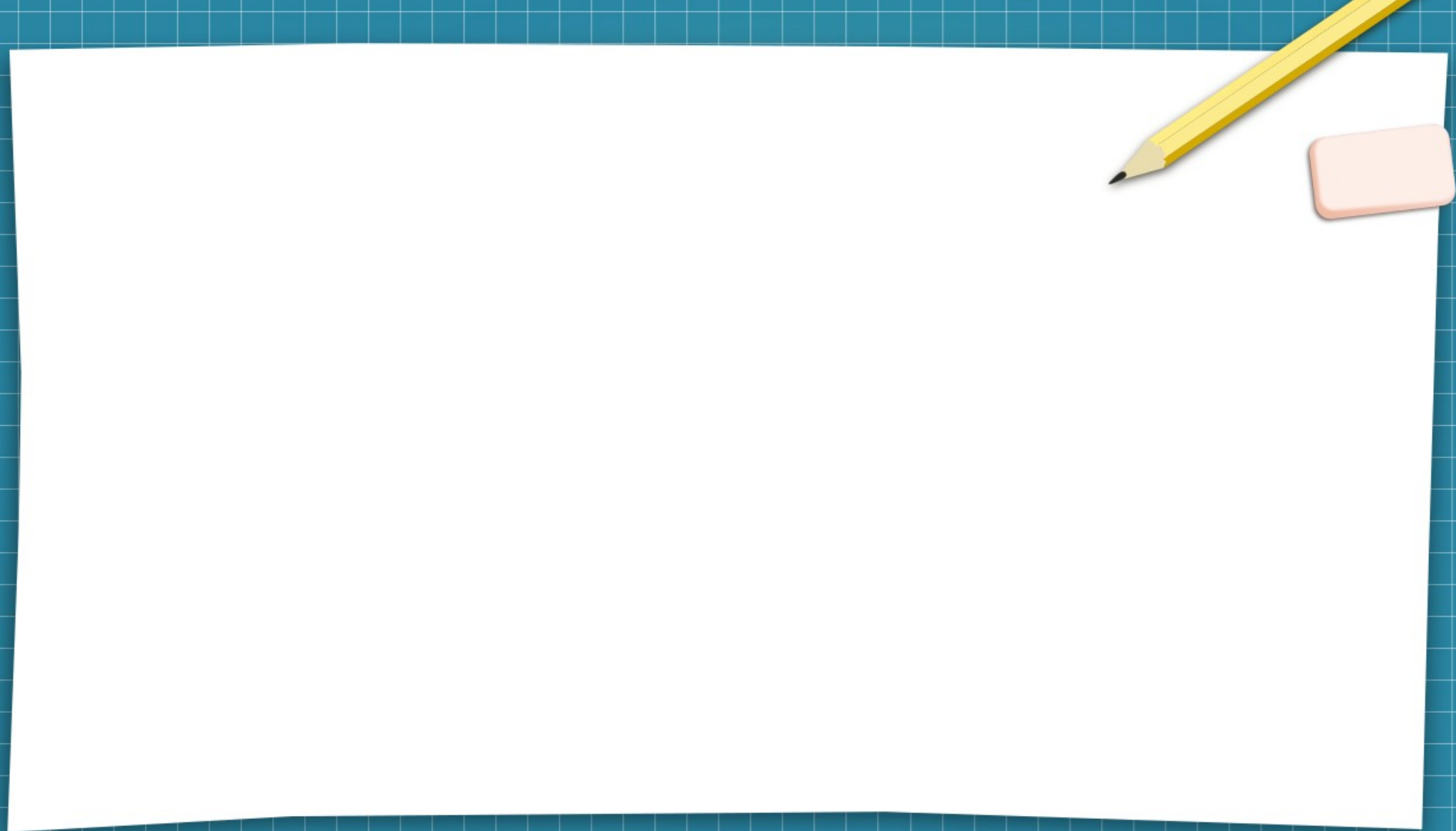


- Ansätze für dynamische Grenzen
 - Durchprobieren von festen Partitionen
 - Randomisiert erzeugte Grenzen
 - Wähle Grenzen, welche Kond(GS) minimieren
 - Thresholding
 - Spikes werden (partielle Pivot.) i.A. nicht klein genug :-(
 - Zeilenpaaren GS
 - Jeder Block erzeugt 2 Zeilen im groben System
 - Maximiere Determinante für 2x2 Systeme
 - Auswirkung auf das globale System?

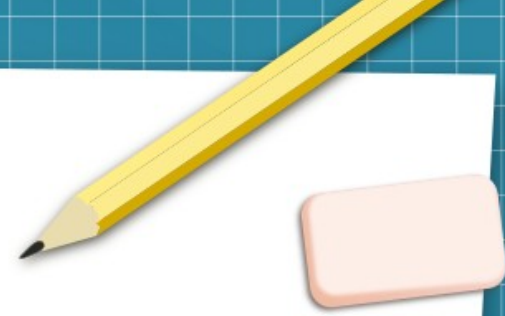
Performance



- **Auswertung** der verschiedenen Ansätze
 - $Ax = b$ mit $x \sim N(3, 1)$
 - Relativer Vorwärtsfehler $\|y - x\| / \|x\|$
 - $n = 512$, 1 Rekursionsschritt
 - 2500 Samples
 - Beispielsklassen
 - *randsvd*: Matrix mit randomisierten Singulärwerte
 - *unif*: Nebendiagonalen gleichverteilt, Diagonale “klein”
 - Kondition graduell hochgeschraubt



Schlussfolgerungen



- Fest gewählte Partitionen funktionieren recht gut
 - Ausprobieren verschiedener Größen
 - Im Mittel selbe Größenordnung als andere Ansätze
 - Billig
- Kondition des groben Systems als Nebenrechnung
 - Erkennen, wann das System nicht mehr zu retten ist
- Statistische Analysen für Erfolgsquoten

Danke für die Aufmerksamkeit!

