Parallele Numerik - Aufgabe 1

Rebecca Seelos, Alexander Lüngen, Joshua Link

8. Juli 2019

Aufgabe 1 - Teil a

Die Reihenfolge in welcher die IDs der Threads ausgegeben werden ist jedes Mal unterschiedlich und ist auch nicht vorherzusagen.

Aufgabe 1 - Teil b

-	Number of Threads	N	Time (s)	Speedup
sequential	1	10^7	0.239	-
	1	10^{10}	239.508	-
atomic	2	10^7	0.634	-
	4	10^{10}	~ 120	-

Aufgabe 1 - Teil b

-	Number of Threads	N	Time (s)	Speedup
sequential	1	10^7	0.239	-
	1	10^{10}	239.508	-
reduction	2	10^7	0.121	1.9
	2	10^{10}	118.204	2
	4	10^7	0.062	3.9
	4	10^{10}	59.183	4
	8	10^{10}	29.689	8.0
	16	10^{10}	29.674	8.1
	32	10^{10}	29.551	8.1

Aufgabe 1 - Teil c

Auflösung N	No. Threads	Time (s)	Speedup
1000	1	4.411	-
1000	16	0.567	7.780
2000	1	17.625	-
2000	16	2.248	7.840
4000	1	71.761	-
4000	16	8.980	7.991
5000	1	110.143	-
5000	16	14.020	7.856

Aufgabe 2 - Teil a

1. Race-Condition auf dem Array a über Index i
2. Threads existieren in gesamter paralleler Region
3. Hier ist die Variable x zunächst global definiert und wird somit implizit zwischen den Threads geshared.
4. f ist global definiert und wird durch jeden Thread private gesetzt.
5. Race-Condition auf die Variable sum.

Aufgabe 2 - Teil b

- Werden Matrizen in C zeilenweise im Speicher hinterlegt, ist es zur optimalen Ausnutzung von Caching-Effekten ideal, wenn auch zeilenweise über Einträge der Matrix iteriert wird.
- Klassischer IJK-Algorithmus: K iteriert über Spalten $\rightarrow$ Optimierung : IKJ-Algorithmus

Aufgabe 2 - Teil b

Mode	N	Seq	16 Threads	
			Zeit	Speedup
gcc - IJK	100	0.0190	0.0070	2.71
gcc - IJK	1000	8.8990	0.7240	12.29
gcc - IKJ	100	0.0166	0.0060	2.77
gcc - IKJ	1000	7.3744	1.6994	4.34
gcc - ... +Inv	100	0.0160	0.0056	2.86
gcc - ... +Inv	1000	6.0300	0.5086	11.86
gcc - ... +O2	100	0.0060	0.0050	1.2
gcc - ... +O2	1000	6.0412	0.5078	11.9

Aufgabe 2 - Teil c

- Berechnung in einigen Threads intensiver → dynamisches Scheduling führt zu optimaleren Verteilung der Arbeitslast
- Veränderung der chunk size hat keinen Einfluss

Mode	N, It., Chunk	Seq.	8 Thr.	S.Up
static	4000, 500, -	34.080	15.061	2.26
static	8000, 500, -	135.882	60.522	2.25
dynamic	4000, 500, 1	33.955	4.803	7.07
dynamic	8000, 500, 1	135.966	19.110	7.11

Aufgabe 3 - Teil a

- **Speedup**: $S(n) = T(1)/T(n)$.
  Der Zusammenhang zwischen serieller und paralleler Ausführungszeit eines Programmes.
- **Effizienz**: Die Effizienz $E(n) = S(n)/n$ gibt die relative Verbesserung der Verarbeitungsgeschwindigkeit an.
- **Auslastung**: $R(n)/(n * T(n))$. Gibt an, wie viele Operationen (Tasks) jeder Prozessor im Durchschnitt pro Zeiteinheit ausgeführt hat.
- **Mehraufwand**: $R(n) = P(n)/P(1)$. Beschreibt den bei einem Multiprozessorsystem erforderlichen Mehraufwand für die Organisation, Synchronisation und Kommunikation der Prozessoren.

Quelle: Vorlesungsfolien Rechnerstrukturen SS2018

Aufgabe 3 - Teil b

- Race-Conditions: Wenn zwei Threads unabhängig voneinander auf eine Ressource lesend oder auch schreibend zugreifen können
- Ungünstige Ausführungszeiten führen zu falschen Ergebnissen
- Um eine Race-Condition zu vermeiden, können die kritischen Abschnitte in der Art und weise gesichert werden

Aufgabe 3 - Teil c

-	GPUs	CPUs	FPGAs
Energieeff.	Gut	Schlecht	Gut
Anwenderfr.	▶ Braucht Einarbeitungszeit ▶ Es gibt Bibliotheken	▶ Am einfachsten zu programmieren ▶ Viele Bibliotheken ▶ Kurze Compilierzeit	▶ Aufwändig zu programmieren ▶ Lange Compilierzeit ▶ Wenig Bibliotheken

Aufgabe 4 - Teil a

Alle Programmdurchläufe für die Messungen hatten eine Fehlerschranke von 0.000001.

- **h**: Verfeinerung $h = \frac{1}{2^l}$
- **l**: $l \subset \mathbb{N}$

	l=4; h=1/16	l=5; h=1/32	l=6; h=1/64
	0.050s	1.429s	37.296s
	0.041s	1.426s	37.014s
	0.040s	1.392s	37.061s
	0.037s	1.428s	37.416s
	0.051s	1.424s	37.295s
Summe	= 0.044s	= 1.420s	= 37.216s

Aufgabe 4 - Teil b

- Innere Schleifendurchgänge direkt voneinander Datenabhängig
- Schleifen müssen in korrekter Reihenfolge abhängig von äußerer Schleife ablaufen
- Naive Parallelisierung der äußersten Schleife liefert falsches Ergebnis

Aufgabe 4 - Teil c

Vorgehensweise zur optimalen Parallelisierung:

- Innere Schleifen zur Summenberechnung parallel berechnen
- Matrix a und Vector u explizit shared
- "+"-Reduction auf Summenvariable firstsum bzw. secondsum

	Laufzeit (seriell)	Laufzeit (parallel)	SpeedUp	Efficiency
l=4; h=1/16	0.044s	0.242s	0.182	0.004
l=5; h=1/32	1.420s	1.154s	1.231	0.026
l=6; h=1/64	37.216s	26.884s	1.384	0.029

Aufgabe 5 - Teil a + b

- u: Eine Lösung $u(x, y)$ muss mind. zweifach differenzierbar sein
- f: $f(x, y)$ muss definiert sein auf Ω
- Γ: ist der Rand von Ω und es gilt: $\Gamma \subset \Omega$

Function:

$$f(x, y) = (N^2 + M^2) * 4 * \pi^2 * sin(2 * M * \pi * x) * sin(2 * N * \pi * y)$$

Aufgabe 5 - c - Visualisiert

- Bei der Lösungsmethodik handelt es sich um eine $h - FEM$

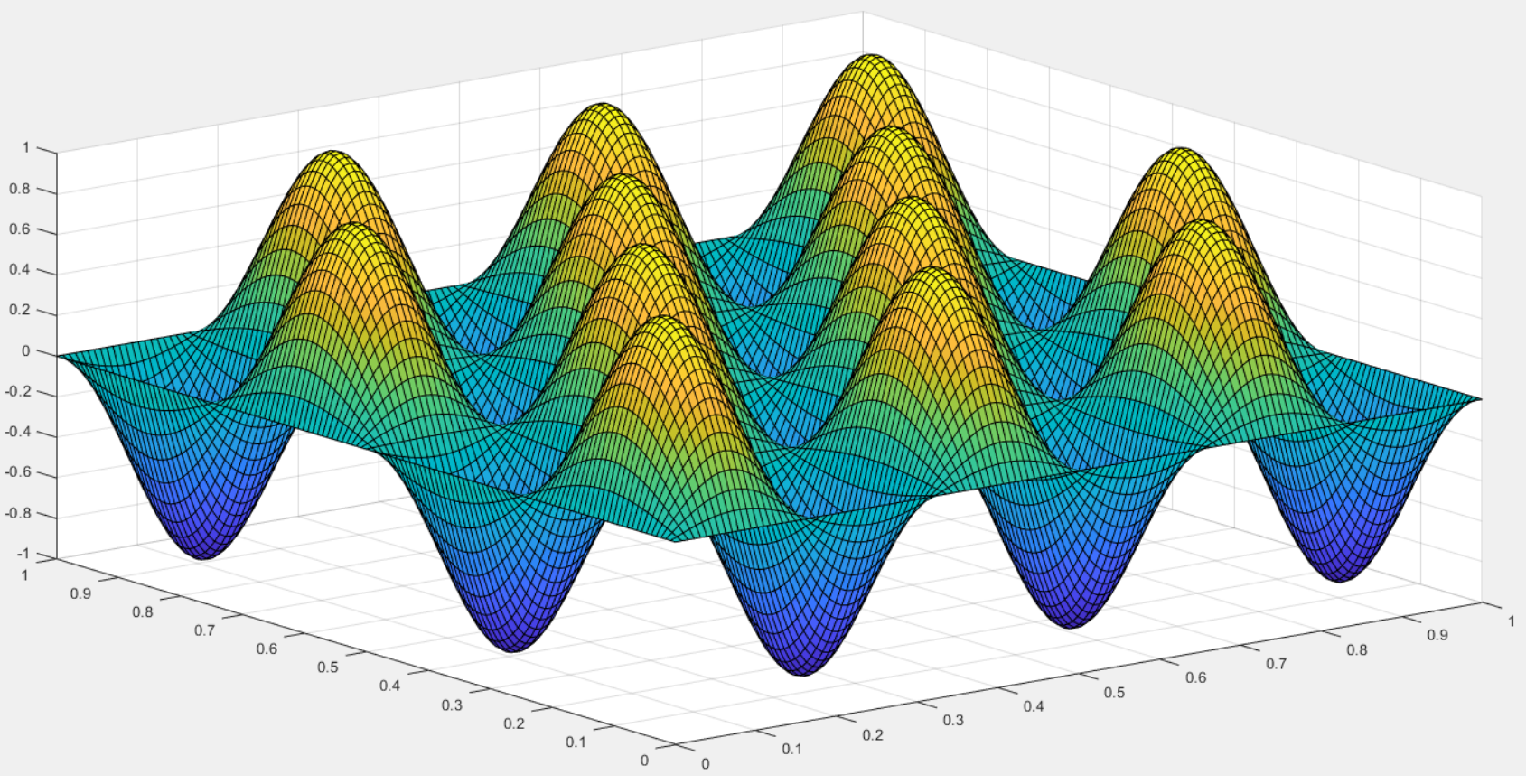

Abbildung: l=7; h=1/128; M=3; N=2

Aufgabe 6 - Visualisiert

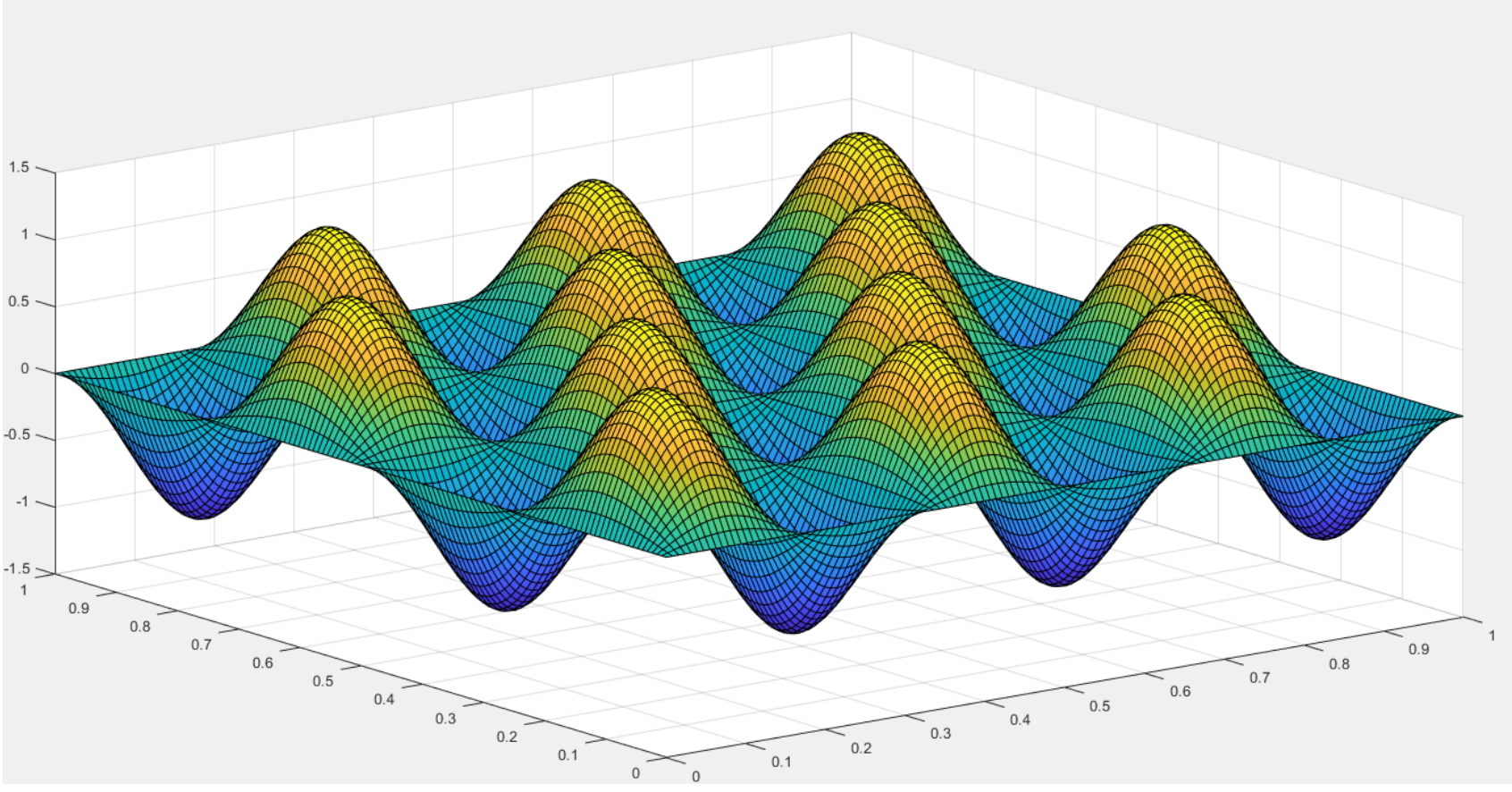

Abbildung: GMRES; I=7; h=1/128; M=3; N=2

Aufgabe 6 - Speedup

-	Aufgabe 5(s)	Aufgabe 6(s)	Speedup
l=5; h=1/32	1.424	0.025	56.96
l=6; h=1/64	13.276	0.135	98.488
l=7; h=1/128	105.447	1.300	81.113

Aufgabe 6 - Visualisiert

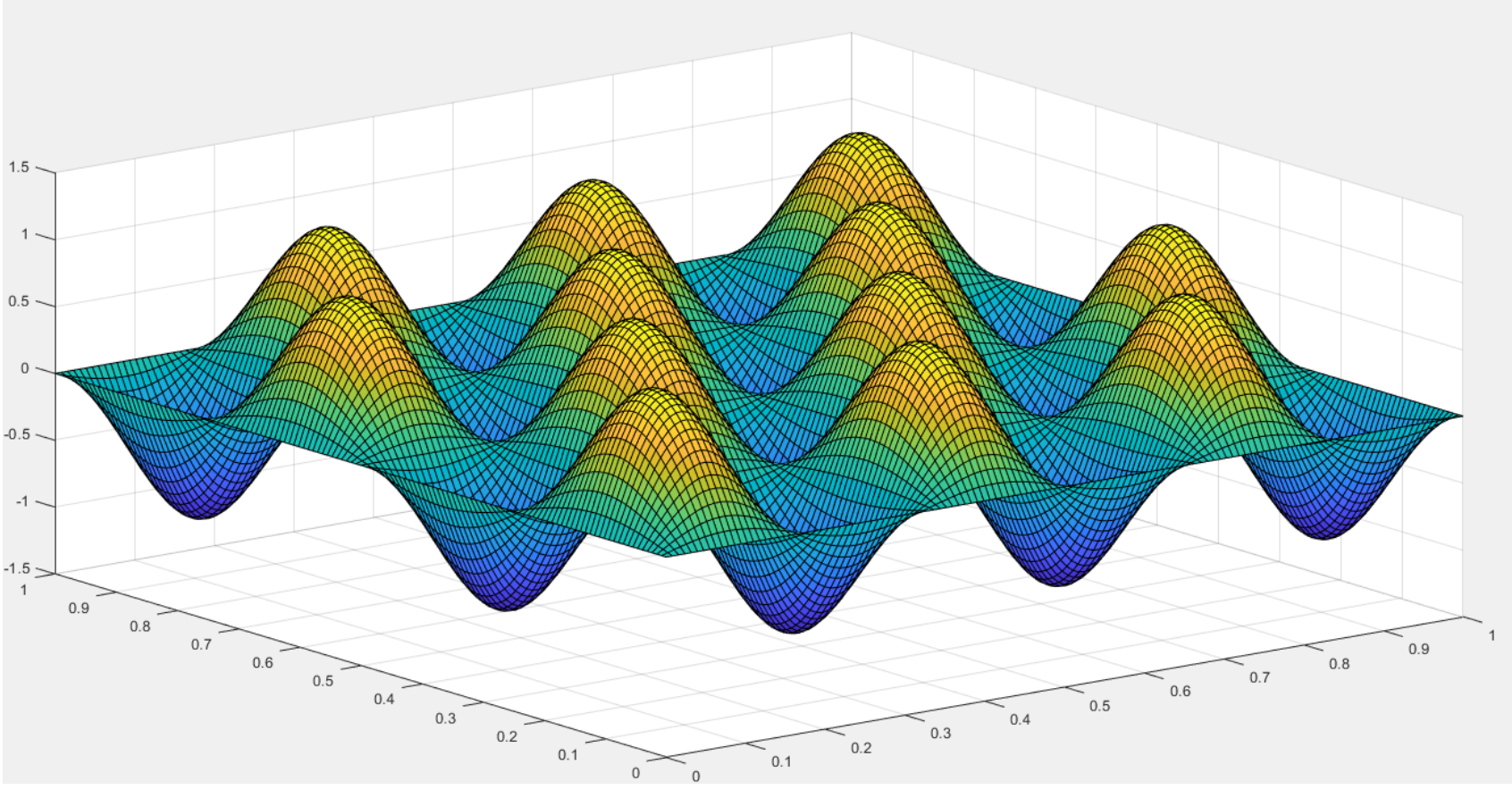

Abbildung: GMRES; I=7; h=1/128; M=3; N=2

Aufgabe 7