Serie 4

Aufgabe 4.1 (Fallunterscheidung)
Wir betrachten die Funktion $f : [0, \pi] \to \mathbb{R}$ mit

$$f(x) = \begin{cases} \alpha x^4 + \beta & \text{für } x \in [0, 1), \\ \sin(x) & \text{für } x \in [1, 2), \\ \dfrac{1}{x - \gamma} + \delta & \text{für } x \in [2, \pi]. \end{cases}$$

mit

$$\alpha = 0{,}1351, \quad \beta = 0{,}7064, \quad \gamma = 0{,}4498 \quad \text{und} \quad \delta = 0{,}2642.$$

Implementieren Sie die Funktion f mithilfe von AVX bzw. AVX-512 in der Datei `conditional.c`.

Aufgabe 4.2 (Natürlicher Logarithmus)
Auch die Funktion zur Berechnung des natürlichen Logarithmus einer positiven Zahl $x \in \mathbb{R}_+$ wird in vielen Anwendungen benötigt. Ihre Implementierung in der Standard-C-Bibliothek ist allerdings nicht sehr performant. Die Funktion lässt sich jedoch durch die Reihenentwicklung

$$\ln(x) = \sum_{k=0}^{\infty} \frac{2}{2k+1} \left(\frac{x-1}{x+1} \right)^{2k+1}$$

darstellen.
Gehen Sie daher analog zur Berechnung der Exponentialfunktion vor. Nutzen Sie so viele Terme, dass Sie für alle Zahlen einen relativen Fehler von ca. $10^{-15} \dots 10^{-16}$ erreichen.
Diese Reihe zeigt ein besseres Konvergenzverhalten, wenn x in der Nähe von 1 liegt.
Nutzen Sie daher zunächst die Logarithmengesetze, um x in einen passenden Wertebereich zu skalieren. Dies kann durch

$$\ln(x) = \ln(x 2^{-m} 2^m) = \ln(x 2^{-m}) + \ln(2^m) = \ln(x 2^{-m}) + m \ln(2)$$

mit einer geeigneten Wahl von $m \in \mathbb{N}$ erreicht werden.

Implementieren Sie auf diese Weise die skalare Berechnung des Logarithmus als Funktion `log_diy` sowie die vektorisierte Variante `log_avx512` in der Datei `log.c`. Als Befehlssatz sollten Sie entweder AVX2 oder AVX-512 nutzen.

Hinweis: Machen Sie sich klar, bevor Sie anfangen zu programmieren, wie nach dem IEEE-754-Standard die Bits für den Datentyp `double` zu interpretieren sind. Informieren Sie sich dazu insbesondere über den *Bias-Wert*.
Nutzen Sie dazu zum Beispiel die Wikipedia: https://de.wikipedia.org/wiki/IEEE_754

Für die skalare Variante dürfen Sie außerdem das `union u_int64_double` nutzen, mit dem Sie die 64 Bits einer `double`-Größe wahlweise als `double` oder als 64-Bit-Integer interpretieren können. Dies ist notwendig, da in C Bit-Operatoren nur auf Integer-Datentypen erlaubt sind. Für die Vektorisierung entspricht dies den `_mm512_cast*`-Befehlen.

Bonus:
Wie bereits erwähnt, konvergiert die obige Reihe besonders schnell für x in der Nähe von 1 bzw. für $z := (x-1)/(x+1)$ nahe 0. Können Sie eine weitere Skalierung des Eingabebereichs finden, die mit noch weniger Termen für die Auswertung der Reihe auskommt?

Website: https://lms.uni-kiel.de/auth/RepositoryEntry/5699797077/CourseNode/106568399409711
Abgabe: Bis Mo, 17.11.25, 16:00 Uhr über das OLAT.