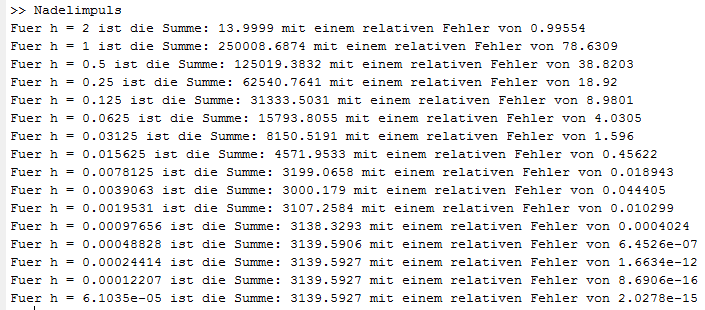
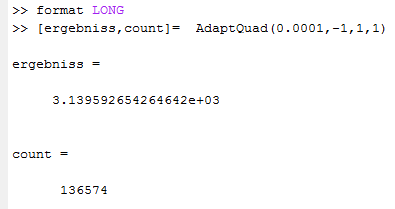
**Aufgabe 1)**

**b)**

Testergebnisse:





Die Anzahl an benötigten Funktionsaufrufen bei dem mit AdaptQuad implementierten Verfahren ist deutlich geringer als bei Ausführung des Scripts Nadelimpuls.

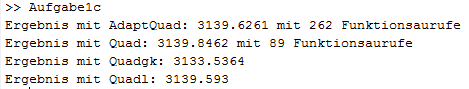
Dies lässt sich dadurch erklären, dass in Intervallen mit geringer Steigung des Funktionsgraphen das adaptive Verfahren Funktionsaufrufe „einsparen“ kann – hier ist keine so feine Auflösung (also eine numerische Integration über eine sehr kleine Intervallbreite) notwendig.

Man erreicht also mit einer deutlich verringerten Anzahl an Funktionsaufrufen mit der AdaptQuad-Funktion eine vergleichbare Genauigkeit zum Nadelimpuls-Skript, welches stur über die gesamte Intervallbreite mit äquidistanten Stützstellen bei konstanter Breite der Teilintervalle rechnet, unabhängig davon, ob eine derart feine Auflösung über das gesamte Intervall notwendig ist.

**c)**

Bei den Werten sind wir von einem relativen Fehler von 0,1 ausgegangen.

Testergebnis:



Dabei fiel auf, dass das genaueste Ergebnis von der Matlab Funktion quadl ausgegeben wurde, gefolgt von unserer eigenen Funktion AdaptQuad, danach kommt quad, und am schlechtesten hat quadgk abgeschnitten.

Die höhere Genauigkeit bei AdaptQuad haben wir uns durch eine viel höhere Anzahl an Funktionsaufrufen erkauft; quad hat nur 89 Aufrufe benötigt, während AdaptQuad ganze 262 benötigt hat, die Differenz zwischen den Ergebniswerten liegt aber noch nicht mal bei 0,2 .

**Aufgabe 2.**

**a.**

Matrix A:

// Zeilen 1 und 3 tauschen, Zeilen 2 und 4 tauschen

🡺

🡺

🡺

A \* x = b 🡺 x

Matrix B

**b.**

Wir würden in diesen Fall eine QR-Zerlegung machen, da deren Komplexität geringer ist als die Komplexität einer LU-Zerlegung und ihre numerische Stabilität zudem höher ist.

Die LU Zerlegung hat eine Komplexität von O(n³), während QR „nur“ eine Komplexität von O(n²) hat.

Bei großen n dauert die Berechnung mit LU Zerlegung deutlich länger als mit QR Zerlegung.

**Aufgabe 3.**

**a.**

**c.**

Es wird die gleiche Anzahl an Rechenoperationen benötigt wie bei einer Hessenberg Matrix, da alle Stellen, die bereits 0 sind nicht mehr berechnet werden müssen. Auch entspricht die Anzahl an Stellen, die mit 1 belegt sind, exakt der Anzahl an Stellen der Nebendiagonalen bei der Hessenberg Matrix (bei Verwendung der Givens-Rotation).

Als Lösungsverfahren würde sich die Householder Methode anbieten, da diese nur einen Funktionsaufruf benötigt, um das Gleichungssystem zu berechnen.