Zettel 2 / Aufgabe 6d

Notation: x_n sei die exakte Berechnung, g_n sei die Berechnung mittels Gleitkommazahlen.

Algorithmus 1:

Schritt 1: $F^1(p,q) = (p, \sqrt{p^2+q})$

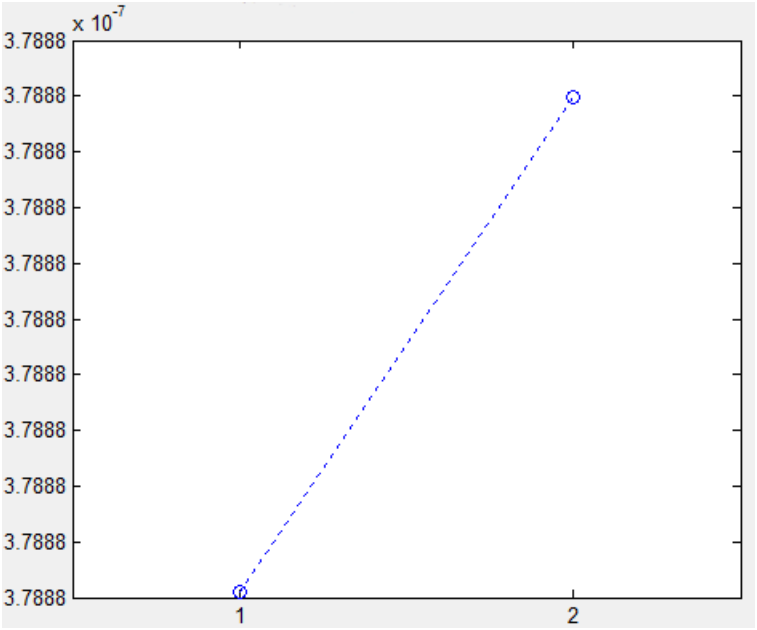
Schritt 2: $F^2(y_1, y_2) = y_2 - y_1$

Sei $q = 1$ und $p = -10$:

Wertetabelle:

Schritt	x_n	g_n
1	(-10, 10.0499)	(-10, 10.0499)
2	(20.0499, 0)	(20.0499, 0)

Plot:



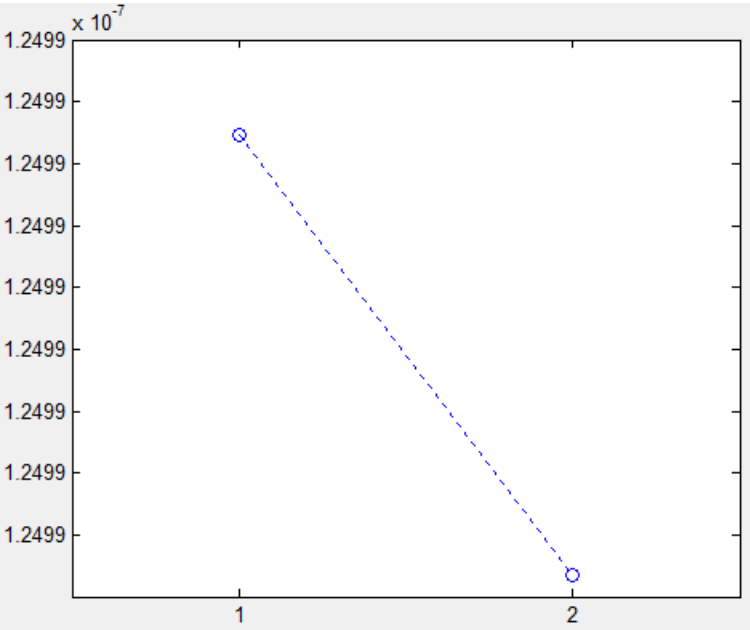
Auch wenn die Wertetabelle zwischen x_n und g_n keinen Unterschied bei den zwei Schritten anzeigt, kann man dennoch im Plot einen Unterschied erkennen. Scheinbar ist die Differenz zwischen x_n und g_n bei Schritt 1 etwas kleiner als bei Schritt 2. Ergo sind die Rundunteefehler bei Schritt 1 kleiner als bei Schritt 2.

Sei $q = 1$ und $p = -100$:

Wertetabelle:

Schritt	x_n	g_n
1	(-100, 100.005)	(-100, 100.005)
2	(200.005, 0)	(200.005, 0)

Plot:



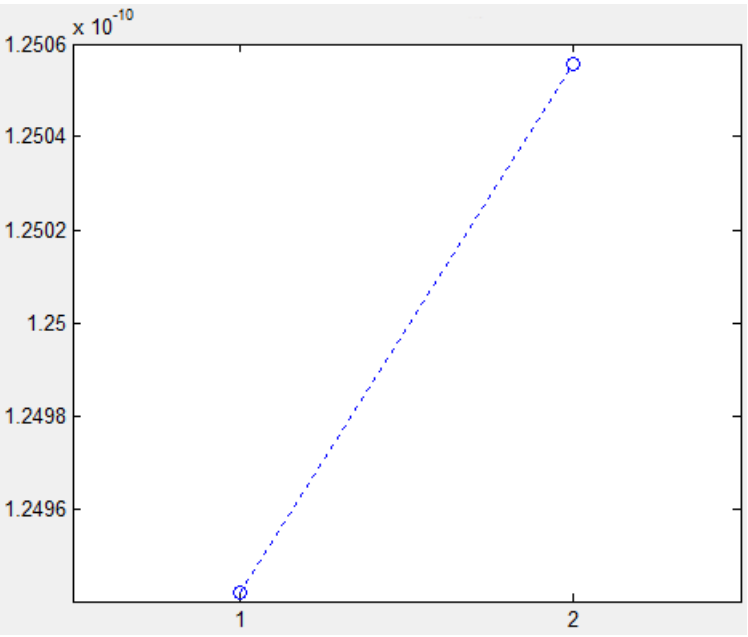
Für die Wertetabelle gilt dasselbe wie oben. Allerdings zeigt uns der Plot wieder einen Unterschied zwischen den zwei Schritten. Diesmal sind die Rundungsfehler des zweite Schritts kleiner als beim erste Schritt.

Sei $q = 1$ und $p = -1000$:

Wertetabelle:

Schritt	x_n	g_n
1	(-1000, 1000.0005)	(-1000, 1000.0005)
2	(200.0005, 0)	(2000.0005, 0)

Plot:



Wieder eine ungenaue Wertetabelle. Dafür zeigt uns der Plot, dass Schritt 1 einen kleineren Rundungsfehler hat als Schritt 2.

Bemerkung:

Das die Wertetabellen keine Unterschiede anzeigen, ist vollkommen klar. Denn rechnet man mit Matlab den Algorithmus mal nach, zeigt sich z.B. bei $q = -10$, dass die Differenz zwischen x_n und g_n etwa 0.000024389 beträgt. Für diesen kleinen Wert ist die Wertetabelle des GUIs nicht ausgelegt. Daher erfolgt eine Rundung und die beiden Berechnungen scheinen die selben Werte zu haben.

Algorithmus 2:

Schritt 1: $F^1(p,q) = (p, q, \sqrt{p^2+q})$

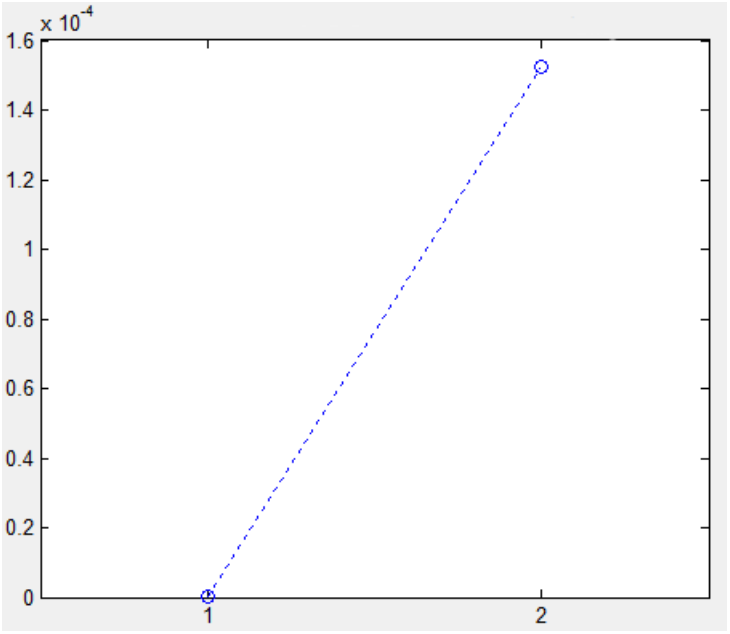
Schritt 2: $F^2(y_1, y_2, y_3) = \frac{y_2}{y_1 + y_3}$

Sei $q = 1$ und $p = -10$:

Wertetabelle:

Schritt	x_n	g_n
1	(-10, 1, 10.0499)	(-10, 1, 10.0499)
2	(20.0499, 0, 0)	(20.0497, 0, 0)

Plot:



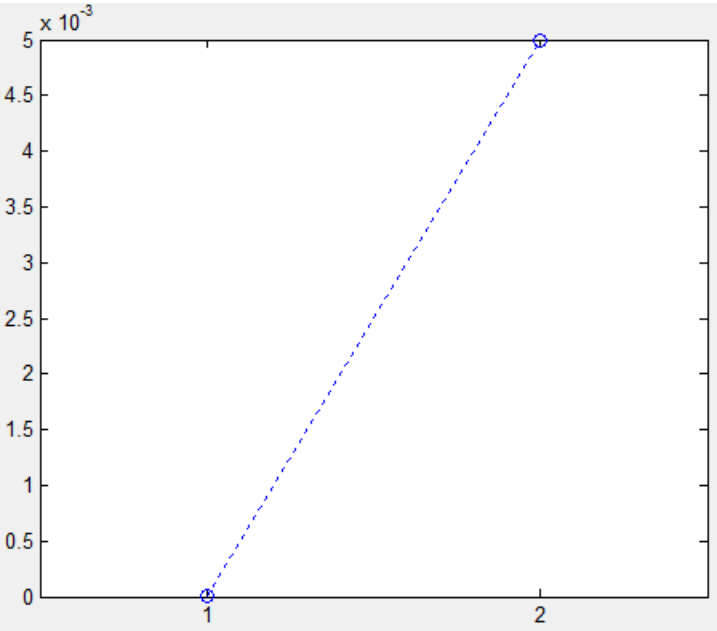
Wie in der Wertetabelle und im Plot zu sehen, gibt es bei Schritt 2 schon einen ordentlichen Rundungsfehler. Die Differenz zwischen x_n und g_n ist sogar erstmals groß genug, um in der Wertetabelle angezeigt werden zu können. Schritt 1 ist dagegen sehr exakt berechnet worden.

Sei $q = 1$ und $p = -100$:

Wertetabelle:

Schritt	x_n	g_n
1	(-100, 1, 100.005)	(-100, 1, 100.005)
2	(200.005, 0, 0)	(200, 0, 0)

Plot:



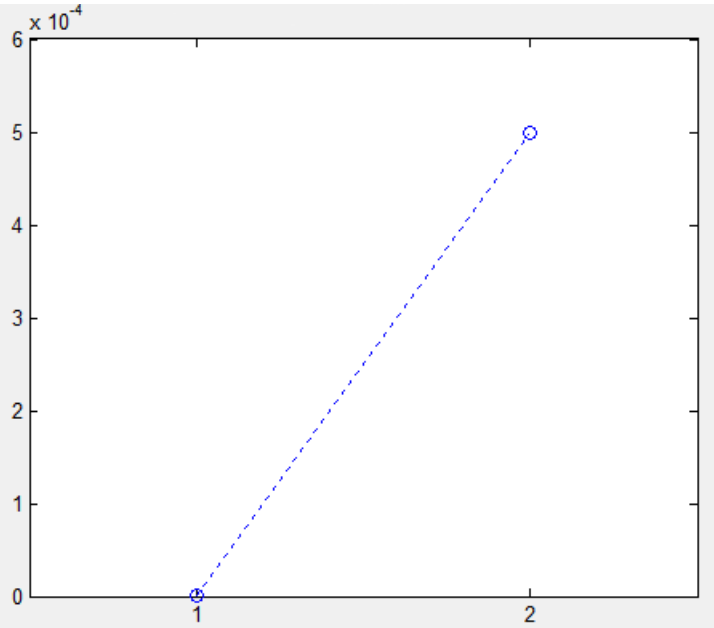
Während der erste Schritt weiterhin gut berechnet wird, hat sich die Differenz (und somit der Rundungsfehler) beim zweiten Schritt deutlich verstärkt.

Sei $q = 1$ und $p = -1000$:

Wertetabelle:

Schritt	x_n	g_n
1	(-1000, 1, 1000.0005)	(-1000, 1, 1000.0005)
2	(2000.0005, 0, 0)	(2000, 0, 0)

Plot:



Für Schritt 1 gilt wieder dasselbe wie bei den vorigen zwei. Allerdings hat sich die Differenz zwischen x_n und g_n im Gegensatz zu $q = -100$ wieder verkleinert. Der Rundungsfehler ist also wieder kleiner geworden.

Eine ähnliche Anomalie zeigt sich auch beim Algorithmus 1. Auch dort tanzt der Algorithmus für $q = -100$ etwas aus der Reihe. Möglicherweise wird $\sqrt{10001}$ ungenauer gerundet als die anderen beiden Wurzeln. Folglich werden schlechte Werte an den zweiten Schritt übergeben.

Schlussfolgerung:

Insgesamt zeigt sich, dass der erste Algorithmus besser konditioniert ist. Beim zweiten gibt es warscheinlich auf Grund des Division bei Schritt 2 stärkere Rundungsfehler.

Im Gegensatz zur Vorlesung sind p und q nicht beide größer Null. Daran liegt es warscheinlich, dass diesmal der erste Algorithmus besser ist.