Übung 02: Verfahrens- und Maschinenfehler

Tobias Blesgen und Leonardo Thome

19.05.2021

Im Folgenden wollen wir die 2-Punktformel und 3-Punktformel auf ihre Verfahrensfehler untersuchen und durch Verwendung verschiedener Datentypen den Maschinenfehler abschätzen.

Die 2-Punktformel beschreibt die Ableitung einer Funktion $f(x)$ an der Stelle x_i mit

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} + \mathcal{O}(h). \tag{1}$$

Die 3-Punktformel leistet die Berechnung hingegen mit einem kleineren Verfahrensfehler der Ordnung $\mathcal{O}(h^2)$.

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} + \mathcal{O}(h^2) \tag{2}$$

Verfahrens- und Maschinenfehler

Verfahrensfehler

Verfahrensfehler der 2-Punktformel

Nach der Taylorentwicklung einer Funktion ergibt sich

$$f(x_{i+1}) = f(x_i) + hf'(x_i) + \frac{h^2}{2}f''(x_i) + \mathcal{O}(h^3). \tag{3}$$

Stellt man die Gleichung nach der Ableitung um, ergibt sich die gesuchte Form für die 2-Punktformel mit Fehlerordnungen als

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{h}{2}f''(x_i) + \mathcal{O}(h^2). \tag{4}$$

Somit wird die höchste Fehlerordnung und damit der Verfahrensfehler der 2-Punktformel durch $\frac{h}{2}f''(x_i)$ beschrieben.

Verfahrensfehler der 3-Punktformel

Nach einem ähnlichen Ansatz wie zuvor betrachten wir in der 3-Punktformel die Taylorformel vor und nach x_i:

$$f(x_{i\pm1}) = f(x_i) \pm hf'(x_i) + \frac{h^2}{2}f''(x_i) \pm \frac{h^3}{6}f'''(x_i) + \mathcal{O}(h^4). \tag{5}$$

Subtrahieren wir die positive und negative Gleichung ergibt sich

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{h^2}{6}f'''(x_i) + \mathcal{O}(h^3). \tag{6}$$

Der Verfahrensfehler liegt also bei $\frac{h^2}{6}f'''(x_i)$.

Maschinenfehler

Da die darstellbaren Zahlen im Computer nur endlich viele Stellen haben, geschehen Rundungsfehler bei Berechnung von Zahlen, die mehr Stellen verwenden als der benutzte Zahlentyp. Diese Genauigkeit nennen wir Maschiengenauigkeit δ_M. Sie ist definiert über die größte reelle Zahl δ_M, für die der Rechner noch $1 + \delta_M = 1$ berechnet (Seite 12 [1]).

Maschinenfehler der 2-Punktformel

Mit

$$\Delta M(f(x+h) - f(x)) = \sqrt{f(x+h)^2\sigma^2 + f(x)^2\sigma^2} \tag{7}$$

$$\approx \sqrt{2\sigma^2}|f(x)| \tag{8}$$

$$= \sqrt{2}\sigma|f(x)| = \sqrt{\frac{2}{3}}\delta_M|f(x)| \tag{9}$$

ergibt sich für den Maschinenfehler F_M der 2-Punktformel:

$$F_M = \sqrt{\frac{2}{3}}\frac{\delta_M}{h}|f(x)| \tag{10}$$

Maschinenfehler der 3-Punktformel

Analog zur 2-Punktformel ergibt sich mit

$$\Delta M(f(x+h) - f(x-h)) = \sqrt{\frac{2}{3}}\delta_M|f(x)| \tag{11}$$

der Maschinenfehler F_M durch:

$$F_M = \sqrt{\frac{1}{6}}\frac{\delta_M}{h}|f(x)| \tag{12}$$

Gesamtfehler

Somit ergibt sich der Gesamtfehler der 2-Punktformel als:

$$\delta_{Ges} = \sqrt{\frac{2}{3}}\frac{\delta_M}{h}|f(x)| + \frac{h}{2}f''(x_i) \tag{13}$$

bzw. in unserem Fall:

$$\delta_{Ges} = (\sqrt{\frac{2}{3}}\frac{\delta_M}{h} + \frac{h}{2})e^x \tag{14}$$

Ähnlich ergibt sich für den Gesamtfehler der 3-Punktformel:

$$\delta_{Ges} = \sqrt{\frac{1}{6}}\frac{\delta_M}{h}|f(x)| + \frac{h^2}{6}f'''(x_i) \tag{15}$$

bzw. in unserem Fall:

$$\delta_{Ges} = (\sqrt{\frac{1}{6}}\frac{\delta_M}{h} + \frac{h^2}{6})e^x \tag{16}$$

Um dieses Verhalten zu bestätigen, werden die Gesamtfehler im späteren Verlauf an die aufgetragenen Fehlerverläufe gefittet. Dabei wird für `Float` (single precision) eine Maschinengenauigkeit von $\delta_{Mf} \approx 6*10^{-8}$ und für `Double` (double precision) eine von $\delta_{Md} \approx 1,1 * 10^{-16}$ angenommen [2].

Beispiel anhand der Differentation

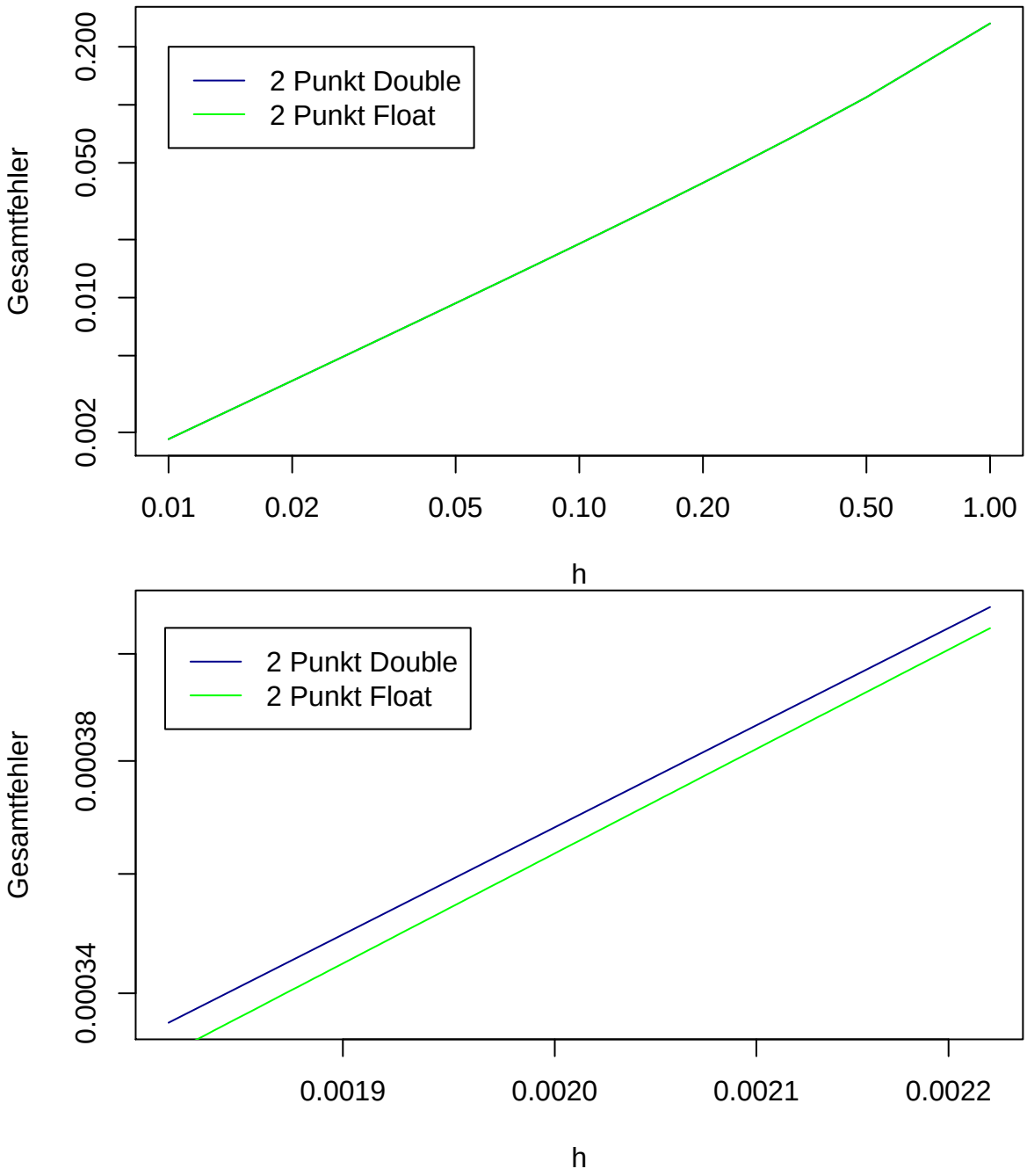
Der 2-Punkt-Ansatz

Am Beispiel der Differentation der Funktion $f(x) = e^x$ an der Stelle x = -1 sollen die Fehlertypen aufgezeigt werden. Wir betrachten zuerst die 2-Punktformel und den Unterschied zwischen einem Ansatz der `Double` und einem der `Floats` verwendet. Da der Code bis auf den Typwechsel identisch ist, liegt hier nur der `Double`-Code vor:

```
#include <Rcpp.h>
#include <math.h>

using namespace Rcpp;

//[[Rcpp::export]]
Rcpp::List diff2PunktDouble(const double x, const int r){
  // Array der ersten 100 Werte:
    Rcpp::NumericVector xValue(100);
    Rcpp::NumericVector yValue(100);
  // Quelltext
    for (int i = r; i<=99+r; i++){
      xValue[i-r] = 1./i;
      yValue[i-r] = (exp(x+1./i) - exp(x))/(1./i);
    }
  // Rückgabe für eine grafische Wiedergabe
    return List::create(Named("x") = xValue, Named("y") = yValue);
}
```



Während für h in der Größenordnung 1 bis 0,01 keine sichtbare Differenz zwischen den Gesamtfehlern auftritt, kann in der nächst kleineren Größenordnung eine Divergenz der beiden Fehlergrößen beobachtet werden.

Der 3-Punkt-Ansatz

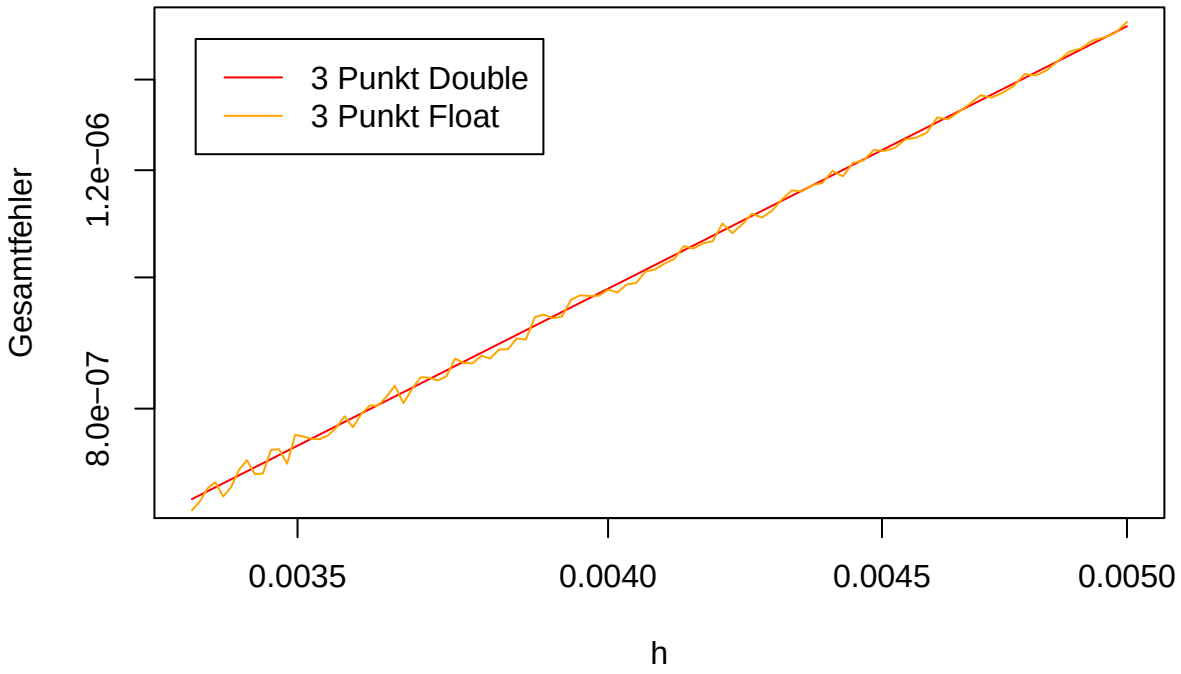
Um das 2-Punktverfahren auch mit einem anderen Verfahren vergleichen zu können, verwenden wir das 3-Punktverfahren. Erneut implementieren wir es einmal mit `Double` und einmal mit `Float`-Precision.

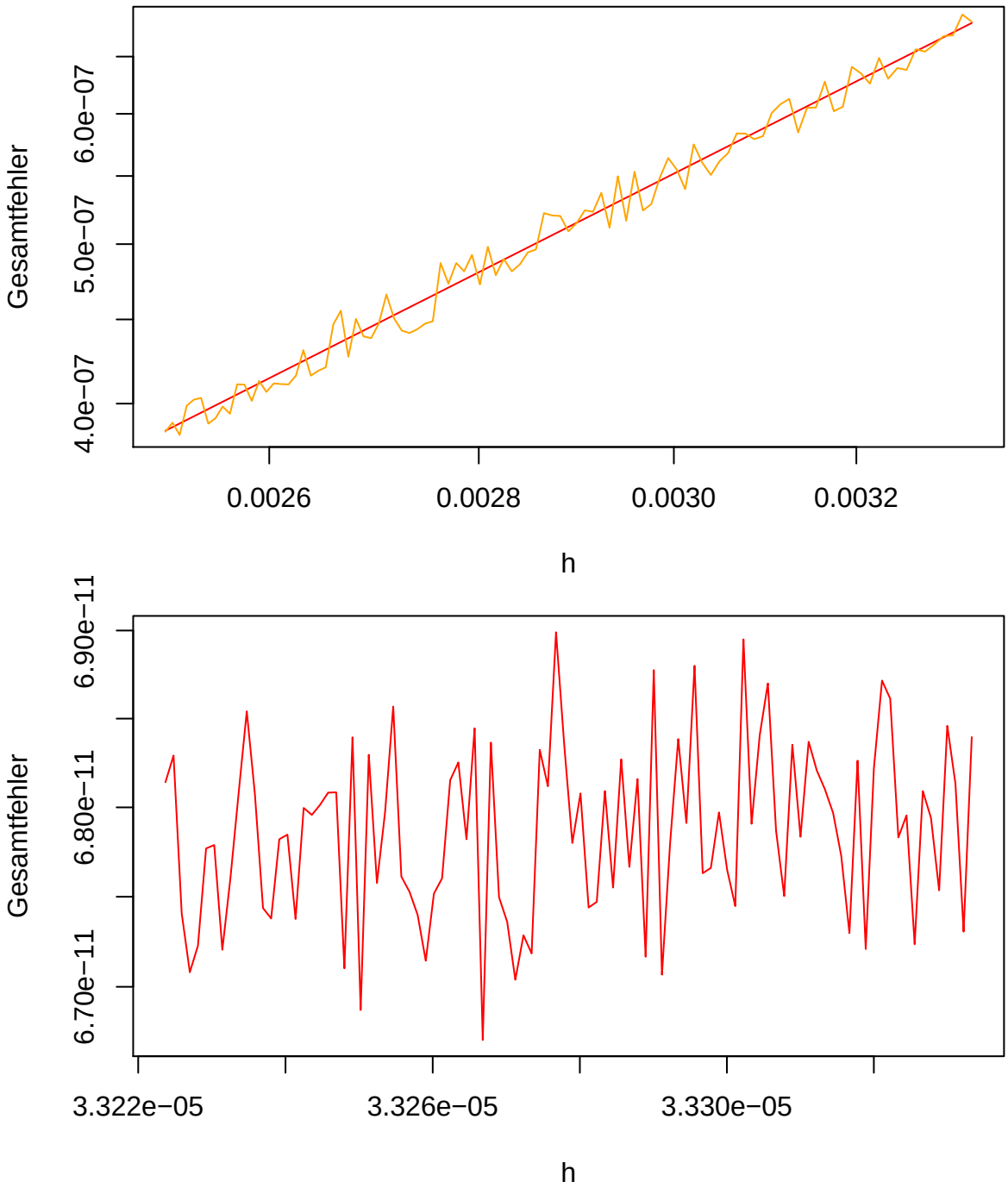
```
#include <Rcpp.h>
#include <math.h>
```

```
using namespace Rcpp;

//[[Rcpp::export]]
Rcpp::List diff3PunktDouble(const double x, const int r){
  // Array der ersten 100 Werte:
    Rcpp::NumericVector xValue(100);
    Rcpp::NumericVector yValue(100);
  // Quelltext
    for (int i = r; i<=99+r; i++){
      xValue[i-r] = 1./i;
      yValue[i-r] = (exp(x+1./i) - exp(x-1./i))/(2./i);
    }
  // Rückgabe für eine grafische Wiedergabe
    return List::create(Named("x") = xValue, Named("y") = yValue);
}
```

Trägt man nun die beiden Graphen grafisch auf, so findet man für h bei ungefähr 0,005 die ersten auffälligen Unterschiede:





Während der `Double`-Graph eine glatte Gerade beschreibt, springt der `Float`-Graph wahllos wirkend über und unter diese Gerade. Dieses Zittern wird mit sinkenden h immer extremer und verdeutlicht, dass die Unterschiede der Näherungswerte die Größenordnung des Maschinenfehlers erreichen. Um diese Zitterbewegung auch am `Double`-Graphen zu entdecken, muss das h in der Größenordnung von 10^{-5} betrachtet werden. Da das Zittern ein Resultat des Maschienefehlers ist können wir über die Stärke des Zitterns die Maschienengenauigkeit von `Float` und `Double` abschätzen. Für `Float` gilt mit dem Werten $F_M = 0.2 * 10^{-7}$ bei h $= 0.003$ (aus dem Graphen):

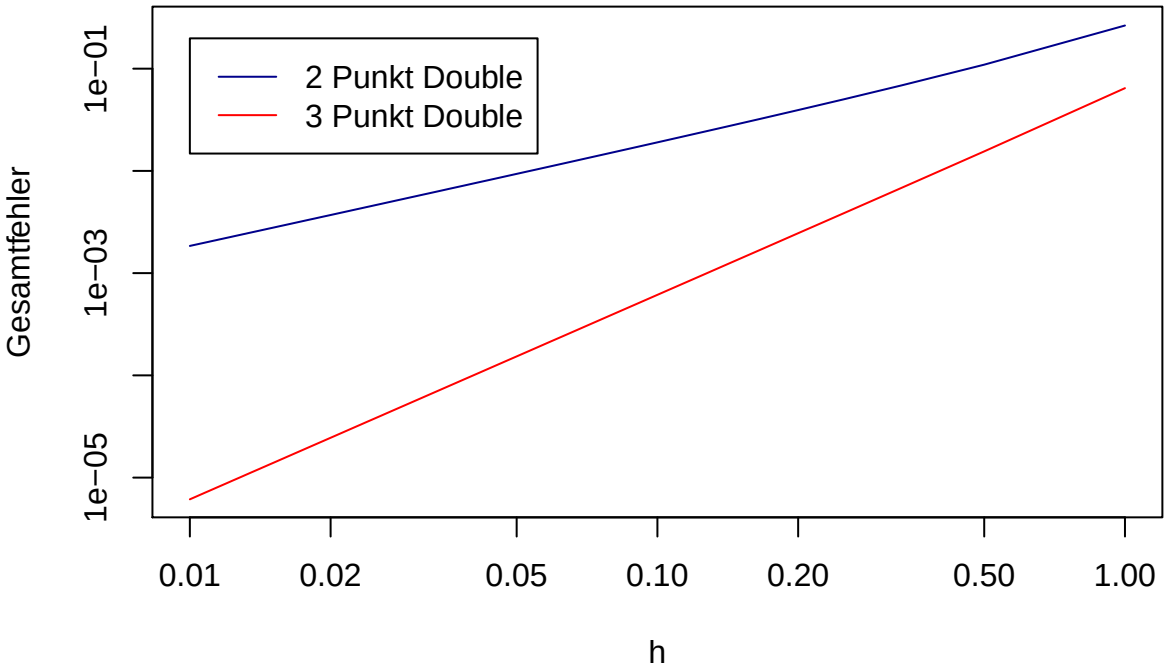
$$\delta_M = \sqrt{\frac{3}{2}} \frac{h * F_M}{|f(-1)|} = \sqrt{\frac{3}{2}} \frac{0.003 * 0.2 * 10^{-7}}{e^{-1}} \approx 2 * 10^{-10} \tag{17}$$

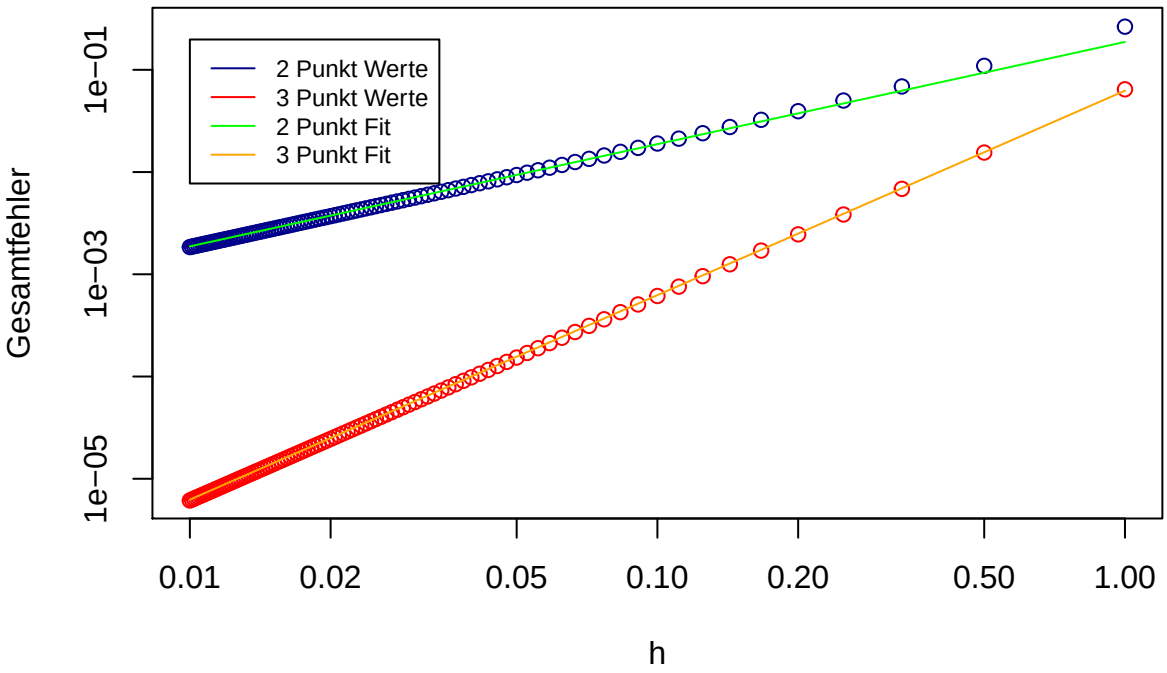
Und für `Double` mit den Werten $F_M = 0.2 * 10^{-11}$ bei h $= 3.332 * 10^{-5}$ (aus dem Graphen):

$$\delta_M = \sqrt{6}\frac{h * F_M}{|f(-1)|} = \sqrt{6}\frac{3.332 * 10^{-5} * 0.2 * 10^{-11}}{e^{-1}} \approx 4.4 * 10^{-16} \tag{18}$$

Vergleicht man diese Werte mit den erwarteten Genauigkeiten von $\delta_{Mf} \approx 6 * 10^{-8}$ für `Floats` und $\delta_{Md} \approx 1,1 * 10^{-16}$ für `Double`, sieht man, dass die Genauigkeit des `Doubles` mit $\delta_M \approx 4.4 * 10^{-16}$ zwar bei uns etwas kleiner ist, jedoch in Näherung gut zum Literaturwert passt. Beim `Float` sieht es jedoch anders aus. Wir erhalten mit $\delta_M \approx 2 * 10^{-10}$ eine Genauigkeit die etwa 2 Zehnerpotenzen zu genau ist. Dies kann an einem Rechenfehler liegen, den wir jedoch nicht gefunden haben. Eine weitere Möglichkeit könnte sein, dass der Rechner in Teilen der Rechnungen einen Typcast vom `Floats`zu `Double` macht und wieder zurück, wodurch unsere Genauigkeit verbessert wird.

Um dieses Verhalten zu bestätigen werden die Gesamtfehler im späteren Verflauf an die aufgetragenen Fehlerverläufe gefittet. Dabei wird für (single precision) eine Maschienengenauigkeit und für (double precision) eine von angenommen [2].





Wie wir sehen, passen die zuvor bestimmten Fehlerfunktionen mit den angepassten Parametern mit den Fehlerkurven der Berechung für 2-Punkt- und 3-Punktformel in Näherung überein. Wir bemerken aber auch, dass bei der 2-Punktformel bei größerem h der Fehler der Berechnung nach oben abweicht. Dies liegt an dem Verfahrensfehler durch die Taylerentwicklung, wo der nächste Fehlerterm der Ordnung h^2 wäre, was für kleine h irrelevant ist, jedoch nicht für z.B. $h = 1$. Ebenso fällt bei Betrachtung der Verfahrensfehler auf, dass der Verfahrensfehler jeweils nicht wie erwartet mit $\frac{h}{2}$ für die 2-Punkt- und $\frac{h^2}{6}$ für die 3-Punktformel ins Gewicht fällt. Sondern mit $\frac{h}{2*2,667}$ für die 2-Punkt- und $\frac{h^2}{6*2,667}$ für die 3-Punktformel. Wodurch unsere Berechnung unerwarteterweise um etwa einen Faktor 2,667 besser ist als erwartet. Für diese Verkleinerung des Fehlers können wir keine Erklärung finden.

Literatur

[1] C. Urbach, *Vorlesungsskript Computerphysik*,Seite 12,2021.

[2] Wikipedia, Artikel: https://de.wikipedia.org/wiki/Maschinengenauigkeit, 15.05.2021.