Übung 1 *Rundungsfehler Skalarprodukt*
Führen Sie eine Rundungsfehleranalyse des Skalarprodukts im $\mathbb{R}^n$

$$F_n(\vec{x},\vec{y}) := \vec{x}\cdot\vec{y} = \sum_{i=1}^{n} x_i y_i \qquad \vec{x},\vec{y}\in\mathbb{R}^n$$

durch, wobei wir annehmen, dass der Algorithmus hierzu durch

$$\begin{aligned} z_0 &:= 0 \\ z_i &:= z_{i-1} + x_i y_i \end{aligned}$$

gegeben ist und z_n die Lösung enthält. Leiten Sie hierzu zunächst eine rekursive Beziehung für $\Delta F_n\,(\,\Delta F_{n-1}\,, rd\,(F_{n-1})\;, x_n,\, y_n\,)$ her.

(4 Punkte)

Übung 2 *Problematische Auswertung*
Betrachten Sie die Funktion

$$f(x) = \frac{1-\cos x}{x}, \quad |x| \ll 1.$$

a) Für welche x ist die Auswertung von $f(x)$ gut bzw. schlecht konditioniert?

b) Warum ist der Algorithmus, diesen Ausdruck in der gegebenen Form für $|x| \ll 1$ zu berechnen, instabil?

c) Finden Sie für $|x| \ll 1$ einen stabilen Algorithmus zur Berechnung von $f(x)$. Dabei sei angenommen, dass $\cos x$ mit Maschinengenauigkeit berechnet wird. Hinweis: Die Darstellung von f kann mit Hilfe der Rechenregeln für trigonometrische Funktionen umgeformt werden ($\sin^2 x + \cos^2 x = 1$).

(4 Punkte)

Übung 3 *Kondition quadratischer Gleichungen*
Das analytische Lösen der quadratischen Gleichung

$$x^2 - 2px + 1 = 0$$

(z.B. mittels quadratischer Ergänzung) führt auf zwei Lösungen $x_1(p)$ und $x_2(p)$, die natürlich vom Parameter p abhängen. Berechnen Sie die relativen Konditionszahlen (Verstärkungsfaktoren) der Funktion

$$F(p) := \begin{pmatrix} x_1(p) \\ x_2(p) \end{pmatrix},$$

welche eine Lösungsvorschrift der quadratischen Gleichung darstellt. Für welche Werte von p ist die Lösung überhaupt in der Menge der reellen Zahlen definiert?

Betrachten Sie nun das alternativ parametrisierte Problem:

$$x^2 - \frac{t^2+1}{t}x + 1 = 0 \qquad\qquad t\in[1,\infty).$$

Berechnen Sie auch für diesen Fall die relativen Konditionszahlen der zugehörigen Lösungsvorschrift $\tilde{F}(t)$ und vergleichen Sie die relativen Konditionszahlen mit denen der ursprünglichen Formulierung. Für welche Werte von p bzw. t wird das jeweilige Problem schlecht konditioniert?

(5 Punkte)

Übung 4 *Potenzreihe für die Exponentialfunktion (Praktische Übung)*
Die Exponentialfunktion e^x lässt sich für $x \in \mathbb{R}$ als Potenzreihe auffassen, wobei der Konvergenzradius unendlich ist. Siehe den Beispiel 1.1 in der Vorlesung `http://cox.iwr.uni-heidelberg.de/teaching/numerik0_ss2013/Beispiel1.1.pdf`.

Die rekursive Formel zur Berechnung der Potenzreihe lautet

$$\begin{aligned} y_1 &:= x, & f_1 &:= 1 + y_1, \\ y_n &:= \frac{x}{n} y_{n-1} & f_n &:= f_{n-1} + y_n. \end{aligned} \tag{1}$$

Schreiben Sie nun ein Programm `potenzreihe`, welches die Exponentialfunktion für einen gegebenen Wert von x, n und einen Fließkommatyp berechnet. Die Benutzereingabe soll über die Kommandozeile

```
./potenzreihe <Zahl> <Iteration> <Datentyp>
```

möglich sein, d.h. der Benutzer des Programms `potenzreihe` gibt für `<Zahl>` eben eine beliebige Zahl x ein, für `<Iteration>` die maximale Anzahl der Iterationschritte und für `<Datentyp>` entweder `double` oder `float`.

a) Testen Sie das Programm mit $x = 5$ und $x = -10$ für 100 Iterationschritte und verschiedene Datentypen. Bilden Sie die Differenz zwischen dem exakten Wert von e^x und der Näherung f_n

$$e_n = |e^x - f_n|.$$

b) Insbesonders für die Werte $x \ll 0$ ist das Ergebnis um mehrere Größenordnungen daneben. Probieren Sie die Rekursionformel (1) so umzuschreiben, dass der Fehler kleiner wird. Testen Sie es mit $x = -20$ und `float`.
Hinweis: Man sollte die Auslöschung bei der Substraktion $x_1 - x_2$ mit $x_1 \approx x_2$ vermeiden.

c) Wenn für $x = -40$ und $n = 1000$, eine sinnvolle Berechnung auch für `long double` versagt. Implementieren Sie das Algorithmus mit der GNU multiprecision library (GMP) (`http://gmplib.org/`) und testen Sie es für diesen Fall.

Schreiben Sie die Werte e_n in einer Datei und stellen Sie alle Ergebnisse in graphischer Form, e_n über Iterationschritte n dar.

Hinweise:

- Der exakte Wert von e^x kann man mit `long double` berechnen:

  ```
  long double exakt = exp(x);
  ```

  wobei die Funktion `exp` ist in der Header-Datei `math.h` definiert

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

- Zur graphischen Darstellung eignet sich das auch im Pool installierte Programm gnuplot. Eine kurze Einführung findet sich auf der Homepage `http://conan.iwr.uni-heidelberg.de/teaching/phlr_ws2012/gnuplot.html`, gut ist außerdem die Einführung von D. Völker der FU Berlin:
  `http://userpage.fu-berlin.de/~voelker/gnuplotkurs/gnuplotkurs.html`

Hinweise zu GMP:

- Die Online-dokumentation befindet sich auf `http://gmplib.org/manual/`.
- Die Einbindung folgender Header-Datei ist für GMP notwendig:

  ```
  #include <gmpxx.h>
  ```

- GMP Programm kann mit

  ```
  g++ -o potenzreihe potenzreihe.cc -lgmpxx -lgmp
  ```

  im Computer Pool kompiliert werden. Benutzen Sie gerne die C++-Erweiterung von GMP.
- Setzen Sie die Geunaugkeit in GMP auf 128 bit mit dem Befehl

  ```
  mpf_set_default_prec(128);
  ```

  der am Anfang des Programms stehen sollte.
- Die `float` Zahlen sind in GMP (C++) als Klassen definiert. Beispiel mit dem Konstruktor:

  ```
  mpf_class a(1.0);
  ```

  Im Konstruktor kann nur ein `string` sein. Wenn man den Konstruktor mit einer `double` Variable aufrufen will, muss zuerst in `string` umgewandelt werden.

  ```
  double a = exp(x);
  mpf_class b(a); // geht nicht, Konstruktor braucht ein string
  ```

  Die Standardoperationen ($+,-,\times,/$) sind auch für `mpf_class` definiert.

(7 Punkte)