Übung 1 *Rechengesetze in Fließkommazahlen*
Welche der folgenden Behauptungen sind wahr:

a) Die Fließkommaoperation $\oplus$ und $\odot$ auf $\mathbb{F}$ sind kommutativ.

b) Für die Addition auf $\mathbb{F}$ gilt das Assoziativgesetz.

c) Zwischen Multiplikation und Addition auf $\mathbb{F}$ gilt das Distributivgesetz.

Widerlegen Sie die falschen Aussagen (geben Sie einfach jeweils ein Gegenbeispiel an), die richtige Aussagen müssen Sie nicht beweisen.

(3 Punkte)

Übung 2 *Problematische Auswertung*
Betrachten Sie die Funktion

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

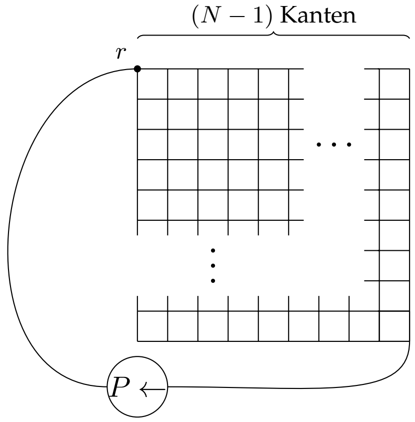
a) Für welche x ist die Auswertung von $f(x)$ schlecht konditioniert?

b) Zeigen Sie, dass der Algorithmus, welcher diesen Ausdruck in der gegebenen Form für $|x| \ll 1$ berechnet, instabil ist. Führen Sie hierzu eine Rundungsfehleranalyse durch. Dabei sei angenommen, dass $\cos(x)$ mit Maschinengenauigkeit berechnet wird.

c) Finden Sie für $|x| \ll 1$ einen stabilen Algorithmus zur Berechnung von $f(x)$. Zeigen Sie in Analogie zu b) die Stabilität Ihres Algroithmus. Hinweis: Die Darstellung von f kann mit Hilfe der Rechenregeln für trigonometrische Funktionen umgeformt werden ($\sin(x)^2 + \cos(x)^2 = 1$).

(1+2+3 Punkte)

Übung 3 *Strömung in Rohrleitungsnetzwerken*



Das abgebildete Röhrennetzwerk hat $n := N^2$ Knoten V und $m := 2N(N-1)$ Kanten E (die alle nach rechts bzw. unten gerichtet sind) zuzüglich der Verbindungen zur Pumpe P mit konstanter Flussrate q_P. Zur Bestimmung des Drucks in den einzelnen Knoten kann mit Hilfe der Kirchhoffschen Gesetze (Knoten und Maschenregel) ein lineares Gleichungssystem aufgestellt werden. Hierzu wird der Druck des Referenzknoten r auf Null gesetzt. Für alle anderen erhält man aus der Knotenregel

$$\sum_{e\in E_v^+} q_e - \sum_{e\in E_v^-} q_e = 0.$$

Hierbei enthalten E_v^+ bzw. E_v^- die Einfluss- bzw. Ausflusskanten am Knoten v und q_e bezeichnet den Fluss durch die Kante e. Endet die Kante an der Pumpe so ist dieser durch die Flussrate $q_e = q_P$ gegeben. Ansonsten gilt

$$q_e = L_e \Delta p_e.$$

Für gegebenes $e = (v, w) \in E$ ist dabei

$$\Delta p_e = \begin{cases} p_w - p_v & v \neq r \wedge w \neq r \\ -p_v & w = r \\ p_w & v = r. \end{cases}$$

In unserem Beispiel sei $L_e = 1$ für alle Kanten.

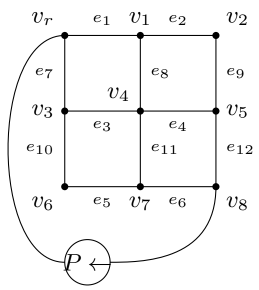
Beantworten Sie die folgenden Fragen für beliebiges $N \in \mathbb{N}$ (ohne Beweis):

a) Wie groß ist die Matrix des linearen Gleichungssystems?

b) Wie viele Werte ungleich 0 gibt es in jeder Zeile?

c) Hat das Gleichungssystem eine eindeutige Lösung?

d) Stellen Sie außerdem das Gleichungssystem für $N = 3$ auf. Nummerieren Sie dafür die Knoten und Kanten nach dem folgenden Schema:



(1+1+1+2 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. Die rekursive Formel zur Berechnung der Potenzreihe lautet

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

Schreiben Sie nun ein Programm `potenzreihe`, welches für gegebenes x und n die entsprechende Näherung der Exponentialfunktion berechnet. Der verwendete Datentyp soll dabei variabel sein. Die Benutzereingabe soll über die Kommandozeile

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

möglich sein, d.h. der Benutzer des Programms `potenzreihe` gibt für `<Zahl>` 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.

Hinweise:

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

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

  wobei die Funktion `exp` ist in der Header-Datei `cmath` definiert

  ```
  #include <cmath>
  ```

(4+2 Punkte)