Georg-August-Universität Göttingen
Institut für Theoretische Physik
Prof. Dr. M. Müller
Priv.-Doz. Dr. S.R. Manmana
SoSe 2018

Übungen zur Vorlesung
Computergestütztes Wissenschaftliches Rechnen

Blatt 1

Lernziele dieses Übungsblattes

- Wiederholung Grundlagen: Umgang mit dem Dateisystem, Editoren, Kompilieren eines C-Codes.
- Einführung in das Konzept "Makefile".
- Umgang/Eigenschaften von Gleitkommazahlen: numerische Genauigkeit, Rundungsfehler.

Aufgabe 1 *Einführung*

Für ein produktives Arbeiten mit Computern ist es nützlich, sich mit den wesentlichen Befehlen und Programmen auseinanderzusetzen, die für das Erstellen und Benutzen von Programmen nötig sind.

a) Machen Sie sich mit den wesentlichen Befehlen zum Erstellen, Löschen und Anzeigen von Verzeichnissen und Dateien vertraut: mkdir, rmdir, rm, ls, cat. Nutzen Sie dazu falls nötig die manual-Seiten (z.B. man ls), oder einschlägige Websites (wiki.ubuntuusers.de, ...).

b) Wählen Sie ihren bevorzugten Texteditor (emacs, vi, nano, notepad, gedit, ...) und erstellen Sie ein Hello-World Programm in C:

```
#include<stdio.h>

main(){
    printf("Hello World");
}
```

Hinweis: Es verbessert die Lesbarkeit Ihres Codes, wenn Sie sich an einem etablierten Schreib-Stil orientieren (Stil von Kernighan und Ritchie, Stroustrup oder Linux-Kernel, siehe z.B. `https://en.wikipedia.org/wiki/Indent_style`).

c) Kompilieren Sie das Programm. Zur Erinnerung: Aus dem C-Kurs werden Sie den Compiler gcc kennen. Führen Sie den Befehl

```
gcc -o HelloWorld HelloWorld.c
```

aus.

d) Für komplexere Projekte ist es oft von Vorteil, mit dem build-Werkzeug „make“ zu arbeiten. Beispielsweise wird es beim Standardprozedere zur Installation von Paketen auf Unix-Systemen aufgerufen:

```
./configure
make
make install
make clean
```

Das Verhalten von make wird von einem sogenannten Makefile gesteuert. Für unser Programm kann es folgendermaßen aussehen:

```
#-----------------------------------------------------------------------
SOURCE=HelloWorld.cpp
PROGRAM=HelloWorld
CC=gcc
CFLAGS=-Wall
#-----------------------------------------------------------------------
OBJECTS=$(SOURCE:.c=.o)
all: $(PROGRAM)

$(PROGRAM): $(OBJECTS)
    $(CC) $(CFLAGS) $(SOURCE) -o$@

%.o: %.c
    $(CC) $(CFLAGS) -c $< -o $@

clean:
    rm -f $(PROGRAM) *.o
```

Erstellen Sie eine solche Textdatei (mit den korrekten Einrückungen!) und nennen Sie diese „Makefile“. Tippen Sie nun „make“, und das HelloWorld-Programm wird kompiliert. Durch „make clean“ werden sowohl die in dem Prozess erstellte Datei HelloWorld.o als auch das ausführbare Programm HelloWorld gelöscht. Im Internet können Sie weitere Erläuterungen zu „make “finden.

Aufgabe 2 *Rundungsfehler*

In einem Computer werden Gleitkommazahlen typischerweise als 64 Bit (double) Werte dargestellt. Aus historischen Gründen und in speziellen Fällen werden auch 32 Bit (single, float in C/C++) genutzt. Dabei wird ein Bit für das Vorzeichen genutzt. Von den übrigen werden 52 (23) Bit für die Mantisse und 11 (8) Bit für den Exponenten benutzt. Für eine double-Variable ergibt sich entsprechend:

$$x = (-1)^s \left(1 + \sum_{i=1}^{52} b_{52-i} 2^{-i}\right) 2^{c-1023}.$$

Hierbei ist s das durch Bit 1 definierte signum, c der durch die Bits 2-12 definierte, ganzzahlige Exponent und b_i die logischen Werte der letzten Bits. Dies bedeutet, dass die Rechengenauigkeit für Gleitkommazahlen auf $53 \log_{10} 2 \approx 16$ beziehungsweise $24 \log_{10} 2 \approx 7$ Dezimalstellen beschränkt ist, und dass bei jeder Operation Rundungsfehler auftreten können.

a) Berechnen Sie für $n = 7$ und $n = 8$ den Ausdruck [n (1/3) - n/3]. Geben Sie die Zahl einmal als reinen Dezimalbruch mit 4 Nachkommastellen (z.B. 0.0010) und einmal in der Exponentialschreibweise (z.B. 1.423e-10) aus.

b) Was lernen Sie daraus für den Fall, dass Sie in einer numerischen Simulation zwei Gleitkommazahlen auf Gleichheit testen wollen?

Aufgabe 3 *Minimierung von Rundungsfehlern*

Um die Rechengenauigkeit Ihres Computers zu testen, berechnen Sie die Reihe

$$F = \sum_{i=0}^{200} i^3 \,.$$

a) Schreiben Sie ein Programm, das die Summation mittels einfacher Genauigkeit (single precision, Typ float) durchführt. Führen Sie die Summation einmal aufsteigend von 0 bis 200 und einmal absteigend von 200 bis 0 aus und geben Sie die Differenz auf dem Bildschirm aus. Wie lässt sich dieses Ergebnis erklären?

b) Wiederholen Sie diese Aufgabe mit doppelter Genauigkeit (double precision, Typ double).

Aufgabe 4 *Vorsicht Falle*

Die Sinus-Funktion kann mit Hilfe folgender Reihe approximiert werden:

$$\sin(x) = \sum_{k=0}^{\infty} \frac{(-1)^k x^{1+2k}}{(1+2k)!} \,.$$

a) Schreiben Sie ein Programm, das die obige Funktion im Intervall $x \in [0; 3]$ berechnet und das Ergebnis in eine Datei ausgibt. Überlegen Sie sich im Vorfeld, wie ein geeignetes Abbruchkriterium für die Reihe aussehen könnte.

b) Stellen Sie das Resultat graphisch dar und vergleichen Sie es mit der Ausgabe der in gnuplot eingebauten sin()-Funktion.

c) Berechnen Sie nun die Funktion im Intervall $x \in [40; 43]$. Was stellen Sie fest? Versuchen Sie eine Erklärung für Ihre Beobachtungen zu finden.

Selbsttest

- Was ist das grundlegende Vorgehen beim Programmieren? Welche Werkzeuge benötige ich, und wie kann ich feststellen, ob diese vorhanden sind?
- Wie stelle ich numerische Ergebnisse, die ich mit Hilfe eines selbst geschriebenen Programms erhalten habe, graphisch dar?
- Was sind Grundelemente eines C-Codes? Darf ich Variablen an beliebiger Stelle deklarieren?