Georg-August-Universität Göttingen
Institut für Theoretische Physik
Prof. Dr. M. Müller, Prof. Dr. S.R. Manmana
SoSe 2025

UNIVERSITÄT GÖTTINGEN

Übungen zur Vorlesung
Computergestütztes Wissenschaftliches Rechnen
Blatt 1

Lernziele dieses Übungsblattes

- Schreiben eines C-Programms und Kompilieren mit gcc
- Numerische Integration einer 1D-Funktion
- Programmierkonzepte: `for`-Schleifen, Funktionen
- Grafische Darstellung von Funktionswerten

Aufgabenmodus

Für die Aufgaben 1 und 2 finden Sie begleitende Tutorials am Ende des Dokuments. Abhängig von Ihren Vorkenntnissen können Sie die Aufgaben entweder eigenständig bearbeiten, oder dem dazugehörigen Tutorial folgen. Ziel der Tutorials ist es, Sie durch die grundlegenden Lernkonzepte zu führen und Ihre C-Kenntnisse aufzufrischen. Nach Abschluss eines Tutorials haben Sie ein funktionierendes Programm vorliegen und die entsprechende (Teil-)Aufgabe hinreichend bearbeitet.

Die Tutorials fallen zunächst sehr feinschrittig aus, werden aber im Laufe des Semesters zunehmend aufeinander aufbauen. Bereits erläuterte Konzepte müssen Sie dann ggf. in früheren Übungen nachlesen.

Die Tutorials sind lediglich als Lösungs-Vorschlag zu verstehen. Wir ermutigen Sie, auch Ihre eigenen Implementierungen auszuprobieren.

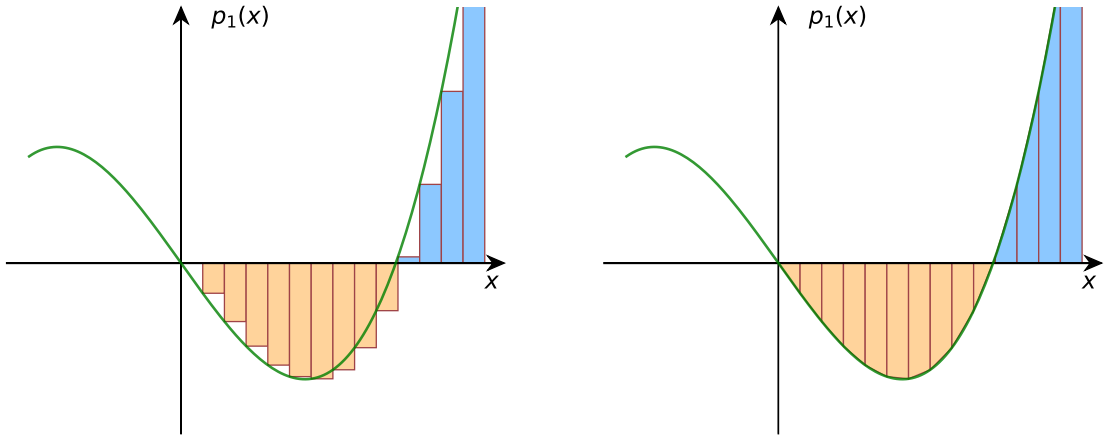
Übungsaufgaben

Aufgabe 1 *Einfache Integration eines Polynoms (Tutorial)*

Hinweis: Teilaufgabe 1.2 hat ein Tutorial, mit dem Sie beginnen können.

Ziel dieser Aufgabe ist es, durch numerische Approximation folgendes Integral auszuwerten:

$$P_1(x) = \int_0^x \left(y^3 - \frac{y}{2}\right) \mathrm{d}y \,.$$



Rechtecke (linksseitig) Trapeze

Mit $p_1(x)$ bezeichnen wir im Folgenden die Ableitung von $P_1(x)$, d.h. den Integranden

$$p_1(x) = x^3 - \frac{x}{2}.$$

1. Lösen Sie das Integral $P_1(x)$ analytisch. Die Lösung dient Ihnen im weiteren Verlauf als Referenz.
2. Schreiben Sie ein C-Programm, das das Integral $P_1(x)$ numerisch durch eine Stufenfunktion wie in der linken Abbildung approximiert. Teilen Sie hierfür das Integrationsintervall $[0, x]$ in N Teilintervalle ein und nähern Sie die Fläche unter der Funktion dann mit Rechtecken der Teilintervallbreite an. Durch Summieren aller Rechteckflächen erhalten Sie die gesuchte Lösung des Integrals.
3. Berechnen Sie mit Ihrem Programm für verschiedene x und N die Werte des Integrals. Vergleichen Sie Ihre Numerik mit der analytischen Lösung. Mit steigendem x erwarten Sie zunehmend ungenaue Werte, während Sie mit steigendem N genauere Lösungen beobachten sollten.

Aufgabe 2 *Modularisierung des Programmes (Tutorial)*

Hinweis: Diese Aufgabe hat ein Tutorial.

Ziel dieser Aufgabe ist es, Ihr Programm modularer zu gestalten und den Integrator vom Integranden $p_1(y)$ logisch zu trennen.

1. Schreiben Sie eine Funktion mit der Signatur

   ```
   double p1(double x);
   ```

   die den Wert von $p_1(x)$ berechnet und zurückgibt.

2. Kapseln Sie den Integrations-Code aus der ersten Aufgabe in einer Funktion mit der folgenden Signatur ein:

   ```
   double integrate(double left, double right,
       int N, double integrand(double));
   ```

   Die Intervallgrenzen, die Auflösung (d.h. die Anzahl der Teilintervalle N) und der Integrand werden jetzt als Parameter übergeben werden. Das Integral $P_1(2)$ kann dann z.B. wie folgt berechnet werden:

   ```
   integrate(0, 2, 100, p1);
   ```

3. Berechnen Sie $P_1(2)$ für $N = 10^1 \dots 10^7$. Tragen Sie die absolute Differenz zwischen der analytischen Lösung und der numerischen Approximation doppellogarithmisch gegen die Anzahl der Stützstellen N auf. Bestimmen Sie die Steigung der Geraden. Wie hängt diese mit der Fehlerordnung der numerischen Näherung zusammen?

4. Schreiben Sie eine zweite Integrator-Funktion, die anstatt der Rechteckfunktionen Trapeze zur Näherung der Teilintervallflächen benutzt. Plotten Sie wieder die Abweichung des numerischen Integrals doppellogarithmisch gegen die Anzahl der Stützstellen N. Ist ein Unterschied in der Fehlerordnung zur vorherigen Integrator-Funktion beobachtbar?

Aufgabe 3 *Berechnung der Fehlerfunktion*

Hinweis: Der Code dieser Aufgabe wird auf dem nächsten Arbeitsblatt wiederverwendet.

Die Fehlerfunktion

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-y^2}\, dy$$

ist ein Beispiel für eine Funktion, die nicht in geschlossener Form analytisch dargestellt werden kann. Numerische Integration bietet sich an, wenngleich auch noch schnellere numerische Approximationen verfügbar sind.

Ziel dieser Aufgabe ist die Berechnung und die grafische Darstellung der Fehlerfunktion.

1. Schreiben Sie zwei Funktionen

   ```
   double erf_midpoint(double x, double delta_x);
   double erf_simpson(double x, double delta_x);
   ```

   die den Wert der Fehlerfunktion zurückgeben und zwar entweder mit der Mittelpunktsformel (siehe VL) oder der Simpsonregel. Modifizieren Sie dafür die Integratoren aus der vorherigen Aufgabe und übergeben jeweils einen geeigneten Integranden. Der Parameter `delta_x` sei der Abstand der Stützstellen. Berechnen Sie mit ihm die Anzahl N der notwendigen Teilintervalle.

2. Berechnen Sie `erf_simpson(x, 1e-4)` an 100 gleichverteilten Positionen im Interval `x`$= [-2, 2]$ und speichern Sie die Werte in eine Datei. Benutzen Sie hierfür eine einfache `for`-Schleife um `erf_simpson` wiederholt mit verschiedenen Parameterwerten aufzurufen. Eine kurze Anleitung zum Schreiben von Dateien finden Sie am Ende dieses Dokuments.

3. Stellen Sie die berechnete Funktion grafisch dar. Eine kurze Anleitung zum Plotten mit matplotlib finden Sie am Ende des Dokumentes.

   Tipp: Sie finden die Exponential-Funktion `exp()` im Header `tgmath.h`, den Sie wie schon `stdio.h` und `stdlib.h` am Kopf Ihres Quellcodes einbinden. Beim Kompilieren sollten Sie dem Compiler die Option `-lm` mitgeben:

   ```
   gcc exercise_3.c -Wall -lm -o exercise_3
   ```

   Dies liegt daran, dass C-Compiler aus historischen Gründen die beiliegende Library mit mathematischen Funktionen nicht automatisch verlinken, da auf großen Rechneranlagen oft spezifische, auf den Computer zugeschnittene Implementierungen verwendet wurden.

4. Berechnen Sie jeweils

   $$I_1 = \int_{-1}^{1} dx\, \frac{1}{x}$$

   und

   $$I_2 = \int_{-1}^{1} dx\, \frac{1}{\sqrt{x}}$$

   mithilfe der Simpson- und Mittelpunktsformel als Funktion der Intervallbreite `delta_x`. Bestimmen Sie zunächst analytisch die Werte von I_1 und I_2 und nehmen Sie diese

Werte als Referenzwerte. Tragen Sie nun als Funktion von `delta_x` die Differenz der Ergebnisse zum Referenzwert doppellogarithmisch auf. Beschreiben Sie Ihre Beobachtung.

Hinweise:
i) Modifizieren Sie hierfür `erf_simpson` und `erf_midpoint` in geeigneter Weise, so dass Sie Funktionen `one_over_x_simpson` und `one_over_x_midpoint` bzw. `one_over_sqrtx_simpson` und `one_over_sqrtx_midpoint` erhalten.
ii) Man kann in `C` mit komplexen Zahlen arbeiten, indem man den Header `complex.h` einbindet.

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, grafisch dar?
- Was sind Grundelemente eines C-Codes? Darf ich Variablen an beliebiger Stelle deklarieren?
- Worauf muss ich achten, wenn ich numerisch das Integral einer Funktion bestimme?

Tutorials

Aufgabe 1: Einfache Integration eines Polynoms

1) Erstellen Sie eine neue Datei namens `integrate.c` und öffnen Sie diese in einem Texteditor Ihrer Wahl.

2) Stellen Sie grundlegende (I/O)-Funktionen bereit, indem Sie den passenden C-Standardheader in den Quellcode einbinden. Fügen Sie hierfür die Zeilen

```
#include <stdio.h>
#include <stdlib.h>
```

   am Anfang von `integrate.c` ein. Dadurch können Sie z.B. in Ihrem Programm nun mit der `printf`-Funktion Text in die Konsole schreiben oder Dateien auf der Festplatte bearbeiten.

3) Schreiben Sie darunter den Rahmen der `main`-Funktion:

```
int main(void)
{
    // Befehl 1 ...
    // Befehl 2 ...

    return EXIT_SUCCESS;
}
```

Die `main`-Funktion ist der Einstiegspunkt Ihres Programms. Nach Programmstart beginnt die CPU die Befehle innerhalb der geschweiften Klammer (im "body" von `main`) von oben nach unten abzuarbeiten. Schreiben Sie alle weiteren Befehle dieser Aufgabe in den body der `main`-Funktion.

4) Deklarieren Sie einige notwendige Variablen für die Integration:

```
double left  = 0.0;
double right = 2.0;
int N = 100;
double sum = 0;
```

Der Datentyp `double` bezeichnet eine Gleitkommazahl (als Näherung für reelle Zahlen) und hat auf den meisten Systemen typischerweise eine Genauigkeit von $64\,\mathrm{bit}$ (wobei der C-Standard hier aber keine Vorgaben macht). Mit Gleitkommazahlen und ihren Problemen beschäftigen wir uns auf einem der nächsten Übungsblätter. Der Datentyp `int` ist eine Ganzzahl mit Vorzeichen, während `unsigned` eine Nichtnegative Ganzzahl ist. Der C-Standard garantiert eine Genauigkeit von mindestens $16\,\mathrm{bit}$ (Zahlen von -32767 bis +32767 für `int`) kann aber prozessorabhängig größer ausfallen (meistens $32\,\mathrm{bit}$, also $4\,\mathrm{Byte}$). Informieren Sie sich über die verwendeten Datentypen auf Ihrem Rechnersystem, wenn Sie mit größeren Zahlen arbeiten wollen.

5) Berechnen Sie die Breite der Teilintervalle:

```
double delta_y = (right - left) / N;
```

Beachten Sie, dass die Division immer den "stärkeren" der beiden Datentypen zurückgibt. Hier wird ein `double` durch eine `int` geteilt, das Resultat ist daher `double`. Wenn Sie zwei `int` durcheinander teilen, ist das Resultat ebenfalls ein (ggf. abgerundeter) `int`.
Achtung: es ist eine typische Fehlerquelle, zu übersehen, dass bei der Division von `int` abgerundet wird und daher ggf. nicht der gewünschte Zahlenwert zurückgegeben wird. Achten Sie beim Programmieren daher auf den Typ Ihrer Variablen!

6) Schreiben Sie eine `for`-Schleife für die N Teilintervalle und definieren Sie eine Hilfsvariable `sum`, um die Summe über alle Teilintervalle auszuführen:

```
double sum = 0.0;
for(int k = 0; k < N; ++k) {

}
```

Der Code innerhalb der geschweiften Klammern der `for`-Schleife wird exakt N mal ausgeführt, wobei die Variable `k` den aktuelle Schleifendurchlauf nummeriert.

7) Berechnen Sie innerhalb der `for`-Schleife den linken Rand y des gegenwärtigen Teilintervalls. Bestimmen sie außerdem den Wert $f(y)$ des Integranden am linken Rand; dies sei Ihre Näherung für die Höhe des Rechteckes. Das Produkt von Teilintervallbreite und Rechteckhöhe ist die Fläche im Teilintervall. Diese Fläche summieren wir mit Hilfe von `sum`:

```
double y = left + k * delta_y;
double f = y*y*y - y/2;
```

```
double A = f * delta_y;
sum += A;
```

Scheuen Sie nicht davor zurück, lokale Hilfsvariablen (wie hier `y`, `f` und `A`) zu benutzen und Rechnungen in logische Teilschritte zu unterteilen. Der Code wird dadurch übersichtlicher, verständlicher und leichter zu warten. Das Programm wird dadurch nicht langsamer, denn der Compiler reduziert und optimiert die Codestruktur, so dass der Maschinencode identisch bleibt.

Die meisten Editoren bieten die Möglichkeit, eine vertikale Linie rechts z.B. in der 80ten Zeichenspalte anzuzeigen. Wenn Ihr Code breiter als diese Grenze wird, versuchen Sie ihn neu zu strukturieren. In den meisten Fällen gibt es eine elegantere und übersichtlichere Weise, um Ihr Ziel zu erreichen!

Berechnen Sie Potenzen mit ganzzahligem Exponenten nach Möglichkeit immer durch wiederholtes Multiplizieren. Die `tgmath`-Bibliothek stellt zwar die Funktion `pow` bereit, diese benutzt jedoch eine Reihenentwicklung zur Berechnung und ist um Größenordnungen langsamer als eine einfache Multiplikation. Benutzen Sie `pow` deshalb nur, wenn Sie nicht ganzzahlige Exponenten benötigen.

8) Nach der `for`-Schleife steht in `sum` der Wert des gesuchten Integrals. Sie können den Wert dann mit dem `printf`-Befehl in die Konsole schreiben:

```
printf("Integralwert: %f\n", sum);
```

Sie übergeben `printf` einen String, also eine Zeichenkette, die dann in die Konsole geschrieben wird. Enthält der String nur einfachen Text, passiert nichts weiter als die Ausgabe dieses Strings. Die Sequenz `%f` zeigt hier aber an, dass eine Gleitkomma-Variable (`float` oder `double`), in diesem Fall `sum`, substituiert werden soll. `\n` löst einen Zeilenumbruch aus.

9) Öffnen Sie ein Konsolenfenster und navigieren Sie zu ihrer Datei `integrate.c`. Kompilieren Sie den Quellcode mit dem Befehl

```
gcc integrate.c -Wall -o integrate
```

Der Compiler erzeugt dann aus Ihrem Quellcode lauffähigen Maschinencode und legt ihn in der Datei `integrate` ab. Wenn im Quellcode Syntaxfehler oder erkennbare logische Probleme vorhanden sind, wird der Compiler Sie darauf hinweisen. Bitte kompilieren Sie Ihre Programme immer mit der Option `-Wall` ("enable all warnings") und berücksichtigen Sie die Warnungen. Alle Programme dieses Kurses sollten bei Abgabe frei von Warnungen sein.

Führen Sie Ihr Programm mit

```
./integrate
```

aus. Der Wert Ihres berechneten Integrals wird in der Konsole angezeigt.

Bearbeiten Sie nun noch Aufgabe 1.1 und 1.3.

Aufgabe 2: Modularisieren

1) Erstellen Sie oberhalb der `main`-Funktion eine neue Funktion:

```
double p1(double y)
```

```
{
    return y*y*y - y/2;
}
```

Das erste `double` zeigt an, dass die Funktion als Rückgabewert Gleitkommazahlen liefert. `p1` bezeichnet den Namen, unter der Sie die Funktion in Ihrem Programm aufrufen können. `double y` bedeutet, dass die Funktion eine Gleitkommazahl als Parameter erhält, welche innerhalb der Funktion den Variablennamen `y` erhält.

2) Ersetzen Sie in Ihrer `for`-Schleife die Zeile

```
double f = y*y*y - y/2;
```

durch

```
double f = p1(y);
```

Die `for`-Schleife enthält jetzt nur noch Code, der die Integration selbst betrifft.

3) Überprüfen Sie die Funktion Ihres Programmes. Das Verhalten sollte sich durch die Einführung von `p1` nicht verändert haben.

4) Erstellen Sie oberhalb von `main` eine neue Funktion und verschieben Sie Ihre Integration dort hin. Fügen Sie außerdem einen `return`-Befehl ein, um das Ergebnis der Integration zurückzugeben:

```
double integrate(double left, double right,
        int N, double integrand(double))
{

    double sum = 0;
    double delta_y = (right - left) / N;

    for(int k = 0; k < N, ++k) {
        double y = left + k * delta_y;
        double f = p1(y);
        double A = f * delta_y;
        sum += A;
    }

    return sum;
}
```

5) Der letzte Parameter der `integrate`-Funktion lautet

```
double integrand(double)
```

Es wird hier keine Zahl eines bestimmten Datentyps übergeben, sondern eine ganze Funktion. Die übergebene Funktion wird innerhalb der `integrate`-Funktion `integrand` genannt und kann mit genau diesem Namen aufgerufen werden. Das erste `double` bedeutet, dass die Funktion eine Gleitkommazahl als Rückgabewert liefert. Das zweite `double` bedeutet, dass die Funktion eine Gleitkommazahl als Parameter entgegen nimmt. Dies entspricht also exakt der Signatur, die auch Ihre Funktion `p1` aufweist.

Ersetzen Sie nun die Zeile

```
double f = p1(y);
```

durch

```
double f = integrand(y);
```

Ihr Integrator ist jetzt vollständig von der Implementierung der Integranden entkoppelt. Das bedeutet, dass Sie die `integrate`-Funktion nicht mehr verändern müssen, um sie auf verschiedene Funktionen anzuwenden.

6) Wenden Sie in der `main`-Funktion die `integrate`-Funktion auf die Funktion `p1` an und geben Sie den Wert des Integrals in die Konsole aus:

```
int main(void)
{
    double left  = 0.;
    double right = 1.;
    int N = 100;

    double result = integrate(left, right, N, p1);
    printf("Integralwert: %f\n", result);
    return EXIT_SUCCESS;
}
```

Sie können nun den Rest der Aufgaben bearbeiten. Einen kurzen Überblick über das Plotten mit Python finden Sie am Ende dieses Dokumentes.

Tipps

Dateiausgabe in C

Um Dateien in C zu schreiben, binden Sie zunächst den Header der entsprechenden Library ein:

```
#include <stdio.h>
```

Anschließend erzeugen Sie einen Datei-Handle mit Hilfe des Datentyps `FILE` und verknüpfen ihn mit `fopen` mit einem Objekt des Dateisystems:

```
FILE* myFile = fopen("data.csv", "w");
```

Achten Sie auf den Asterisk (“`*`”) nach dem Datentyp um einen Pointer zu erstellen. Der zweite Parameter `"w"` bewirkt ein Überschreiben vorhandener Dateien. Mit `"a"` können Sie Daten an eine bestehende Datei anhängen, mit `"r"` öffnen Sie die Datei schreibgeschützt.

Mit `"fprintf"` können Sie Dateien wie die Konsole beschreiben:

```
fprintf(myFile, "X=%g,Y=%g,I=%d\n", x, y, i)
```

Wenn die Datei fertig bearbeitet wurde, schließen Sie sie mit

```
fclose(myFile);
```

(Dateien werden mit Programm-Ende automatisch geschlossen; wenn das Programm abstürzt können jedoch Daten verloren gehen. Schließen Sie Dateien daher, wenn Sie sie nicht mehr brauchen, oder flushen Sie den Buffer mit `fflush()`!) Detaillierte Informationen zur Dateiverwaltung mit `stdio.h` finden Sie unter `https://en.cppreference.com/w/c/io`.

Plotten mit matplotlib

matplotlib ist ein Pythonmodul zur Erstellung wissenschaftlicher Grafiken aus unterschiedlichsten Datensätzen.

Daten werden häufig als CSV-Dateien (comma separated variables) gespeichert. Dabei werden Werte horizontal durch Kommas separiert und vertikal durch Zeilenumbrüche. Die x-y-Koordinaten einer Parabel könnten z.B. folgendermaßen in einer Textdatei abgespeichert sein:

```
-2,4
-1,1
-0.5,0.25
0,0
0.5,0.25
1,1
2,4
```

Die erste Spalte stellt die x-Koordinate dar, die zweite Spalte die y-Koordinate. Wenn Sie eine solche Datei plotten möchten können Sie folgendermaßen vorgehen:

1) Öffnen Sie ein Konsolenfenster und navigieren Sie zu Ihrer CSV-Datei. Geben Sie das Kommando `python3` ein.[1]

2) Laden Sie matplotlib mit dem Befehl

   ```
   import matplotlib.pyplot as plt
   ```

   Dadurch ist matplotlib unter dem Kürzel `plt` verfügbar.

3) Laden Sie numpy mit dem Befehl

   ```
   import numpy as np
   ```

   numpy stellt effiziente und performante Matrizen und Arrays für Python bereit und ist die Basis unzähliger Numerik-Bibliotheken unter Python. Für mehr Informationen besuchen Sie `https://www.scipy.org/`

4) Laden Sie die Daten aus der CSV-Datei:

   ```
   data = np.loadtxt("data.csv", delimiter=",")
   ```

5) Im Objekt `data` befinden sich jetzt Ihre numerischen Werte. Wie in C können Sie auf einen Array-Eintrag mit eckigen Klammern zugreifen. Die Koordinaten werden aber durch ein Komma getrennt, also z.B. `data[6,2]`.

[1] Sie können die die Python-Befehle auch in einer Datei, z.B. `plot.py`, speichern, und diese dann mit `python3 plot.py` ausführen. Eine weitere Alternative ist die Verwendung von Jupyter Notebooks, um Python-Code und Plots gemischt in einem Browserfenster bearbeiten und anzeigen zu können.

In numpy können Sie auch auf ganze Spalten oder Zeilen auf einmal zugreifen, indem Sie mit einem Doppelpunkt indizieren. Der Befehl `data[3,:]` liefert z.B. die gesamte vierte Zeile als 1D-Array zurück.

Um die zweite Spalte ihrer CSV gegen die erste Spalte zu plotten, benutzen Sie folgenden Befehl:

```
plt.plot(data[:,0], data[:,1])
plt.show()
```

Sie können doppellogarithmische Plots mit

```
plt.loglog(data[:,0], data[:,1])
plt.show()
```

erstellen. Sie beenden Python mit dem Befehl `quit()` oder durch die Eingabe von CTRL-D.

Wenn Sie mehr über matplotlib lernen möchten, finden Sie unter `https://matplotlib.org` umfangreiche Dokumentation und viele Beispielgrafiken mit Code.

Loslegen mit git und GitLab

git ist ein Tool zur Versionsverwaltung von Dateien und ist enorm hilfreich bei der Kollaboration in Teams, sei es beim Programmierung oder für das Schreiben eines Papers. Die Sicherung von Snapshots (bzw. "commits" im Vokabular von git) gewährleistet es, dass man jederzeit zu einem früheren (funktionsfähigen) Zustand zurückkehren kann, um z.B. vorherige Ergebnisse reproduzieren zu können, oder um experimentelle Änderungen ggf. schnell verwerfen zu können. Die Möglichkeit, sich den Unterschied ("diff") zwischen Commits anzeigen zu lassen, erlaubt es, eventuelle neue Fehler viel schneller aufzuspüren.

Ein mit git verwaltetes Verzeichnis ("repository") kann über einen Server für andere verfügbar gemacht werden. Ein solcher Server ist GitLab, der zudem ein gut sortiertes Web-Interface für Ihre Repositories zur Verfügung stellt.

Im Folgenden erstellen wir uns ein mit dem GitLab-Server der GWDG verknüpftes Repository.

1) Loggen Sie sich bei `gitlab.gwdg.de` mit Ihrer Uni-Mail-Adresse und Ihrem Kennwort an.

2) Optional: Klicken Sie nach dem Login ganz oben recht auf Ihr Avatarbild und gehen Sie auf Einstellungen ("Preferences"). Wählen Sie dann in der Navigation links "SSH Keys" aus und fügen Sie Ihren öffentlichen SSH-Schlüssel hinzu (folgen Sie dazu einfach den Instruktionen). Das erspart Ihnen die ständige Eingabe Ihrer Login-Daten in der Konsole bei der Verwendung von git mit `gitlab.gwdg.de`.

3) Erstellen Sie nun Ihr neues CWR-Projekt, z.B. mit Hilfe des Plus-Buttons in der Navigation oben. Wählen Sie im nächsten Schritt "Create blank project" aus. Auf der nächsten Seite können Sie dem Projekt einen beliebigen Namen geben, z.B. "CWR". Drücken Sie dann auf "Create project".

4) Nun befinden Sie sich auf der Seite des neuen Projekts. Da es noch leer ist, hat GitLab ein paar Vorschläge, wie Sie mit Hilfe der Konsole ein git-Repository lokal

auf Ihrem Computer erstellen oder verknüpfen können.

5) (Dieser Schritt kann übersprungen werden, wenn Sie git auf Ihrem Computer schon mal verwendet haben.) Kopieren Sie zunächst die Befehle unter der Überschrift "Git global setup" in Ihre Konsole, damit git weiß, unter welchem Namen Ihre Commits angelegt werden sollen.

6) Führen Sie dann die Befehle unter der Überschrift "Create a new repository" in Ihrer Konsole aus. Machen Sie das am Besten Zeile für Zeile und versuchen Sie zu erraten, was die einzelnen Befehle machen könnten (bzw. fragen Sie Ihre/n Tutor/in!)

7) Nach dem letzten Befehl (`git push`) sehen Sie die neu angelegte Datei `README.md` auch auf der GitLab-Projektseite (ggf. muss die Seite neu geladen werden), denn `git push` schickt alle neuen Änderungen zum git-Server, in diesem Fall `gitlab.gwdg.de`.

8) Probieren Sie die Befehle `git status` und `git log` in Ihrer Konsole aus. Bearbeiten (und speichern) Sie mit einem Texteditor die Datei `README.md`. Was gibt `git status` jetzt aus? Was sehen Sie bei der Eingabe von `git diff`? Verwenden Sie dann

   ```
   git add README.md
   git commit -m "Edit README"
   ```

   um zunächst Ihre Änderungen für den nächsten commit vorzumerken (`git add`). und dann einen neuen Snapshot/commit zu erstellen. Sehen Sie sich die aktualisierte Versionshistorie mit `git log` an. Verwenden Sie nun `git push`, um den neuen Commit zum GitLab-Server zu schicken.

9) Die Spitze Ihrer aktuellen Zweigs in der Versionshistorie (oder "branch") sollte den voreingestellten Namen "main" (oder ggf. bei älteren Projekten "master") tragen. Dieser Name wird Ihnen z.B. mit `git status` oder `git log` angezeigt. Führen Sie

   ```
   git checkout main^
   ```

   (bzw. `git checkout master^`) aus, um in der Versionshistorie einen Schritt vor die Spitze von "main" (bzw. "master") zu springen. Prüfen Sie nach, dass Ihre letzte Änderung von `README.md` tatsächlich verschwunden ist. Verwenden Sie dann `git checkout main` (bzw. `git checkout master`), um wieder zum neuesten Commit von "main" (bzw. "master") zu springen.

   So können Sie zu früheren Zuständen zurückkehren. Sie können Ihre Versionshistorie sogar verzweigen lassen (und später wieder zusammenführen). Und vieles Nützliche mehr; und da sich die allermeisten git-Befehle rückgängig machen lassen (und Sie durch die Synchronisation mit GitLab automatisch ein weiteres Backup Ihrer Arbeit haben), brauchen Sie (fast) keine Angst zu haben, etwas kaputt zu machen, also seien Sie mutig!

10) Sie können Ihr Projekt-Repository über GitLab teilen, z.B. mit Ihrem/r Tutor/in. Gehen Sie dazu einfach auf Ihrer Projektseite links in der Navigation auf "Mem-

bers", und laden Sie die entsprechende Person ein, die dann mit `git clone` eine lokale Kopie des Repositories auf ihrem Computer erstellen kann. Mit Hilfe von `git push` und `git pull` kann man neue Änderungen auf den Server schicken oder von dort holen.

Weitere Infos und eine Einleitung auf Englisch (zusammengestellt von Kai Oliver Meister) können Sie z.B. hier finden.