# Praktikum Wissenschaftliches Rechnen

**Nichtlineare Gleichungssysteme**

**Sebastian Schöps, Timon Seibel**

TECHNISCHE UNIVERSITÄT DARMSTADT

**Sommersemester 2025**
**23. Juni 2025**
**Übungsblatt 5**

Ziel dieser Übung ist es, das Newton-Verfahren zur Lösung Nichtlinearer Gleichungssysteme in `Python` zu implementieren. Hierzu wird Ihnen auf `Moodle` das Grundgerüst des abzugebenden `Python`-Projekts zur Verfügung gestellt. Ihre Aufgabe ist es, den Code entsprechend der Aufgabenstellung fertigzustellen.
Sie dürfen weitere (Hilfs-)Methoden hinzufügen, um Ihren Code übersichtlicher zu gestalten. Aus der Sicht des Software Engineerings wird dies sogar empfohlen. Kommentieren Sie Ihren Code, sodass er für Dritte verständlich ist.

**Wichtig:** Die Übung ist **in Gruppen** zu bearbeiten und die vorgegebenen (noch zu vervollständigenden) Methoden sind zwingend zu verwenden. Insbesondere dürfen weder die Namen der vorgegebenen Methoden noch die Reihenfolge ihrer Argumente abgeändert werden. Ebenso dürfen Sie die Rückgabewerte nicht ändern. Andernfalls könnten die zugehörigen Tests fehlschlagen, was zu Punktverlust führt.
Weiterhin möchten wir Sie daran erinnern, dass Sie, mit Ausnahme von `numpy` und `matplotlib`, keine externen Bibliotheken benutzen dürfen, sofern dies die Aufgabenstellung nicht ausdrücklich erlaubt.

Zur Bearbeitung der Übung stellen wir Ihnen die folgenden Dateien zur Verfügung:

**newton.py** Grundgerüst der Implementierung des Newton-Verfahrens

**main.py** Von hieraus sollen die geforderten Plots **automatisiert** erstellt und gespeichert werden

**beobachtungen.txt** Hier sollen sie Ihre Beobachtungen schriftlich festhalten

**Abgabe:** Komprimieren Sie alle relevanten und Verzeichnisse in `gruppe_xy_uebung_5.zip` (oder `.tar.gz`), wobei `xy` durch Ihre Gruppennummer zu ersetzen ist, und reichen Sie die Archiv-Datei bis **spätestens 06.07.2025, 23:59 Uhr** auf `Moodle` ein. Spätere Abgaben können nicht berücksichtigt werden. Reichen Sie Ihre Abgabe daher frühzeitig ein und melden Sie sich bei Problemen mit dem Upload vor Ablauf der Deadline, damit wir gegebenenfalls eine Lösung finden können.

## Aufgabe 5.1: Newton-Verfahren (10 Punkte)

Ziel dieser Übung ist es, das Newton-Verfahren in 1D – sowohl das lokale als auch das globalisierte – zu implementieren und zur Lösung nichtlinearer Gleichungssysteme zu nutzen.

### 5.1a)

Zunächt sollen Sie das lokale Newton-Verfahren implementieren. Vervollständigen Sie hierfür die Methode `newton`. Diese erwartet als Input die Funktion `f: Callable`, deren Nullstelle zu finden ist, die dazugehörige Ableitungsfunktion `df: Callable` sowie den Startwert `x0: float`. Darüber hinaus kann optional die Toleranz `tol: float` mit Standardwert $10^{-8}$ und die Höchstzahl an Iterationen `maxiter: float` mit Standardwert übergeben werden.
Die Methode soll ein Array zurückgeben geben, welches $x^{(0)}$ sowie die anderen Iterierten $x^{(k)}$ eines jeden Iterationsschritts beinhalten soll. Hierbei sollte $x^{(k)}$ den Index $k$ im Array haben.

Die Methode soll nach maximal `maxiter` Iterationen beendet werden. Die Iteration soll frühzeitig enden, wenn die aktuelle Iterierte $x^{(k)}$ nah genug an der exakten Lösung ist, sprich, $|f(x^{(k)})| <$ `tol`. Achten Sie stets darauf, dass die aktuelle Iterierte keine Nullstelle der Ableitung ist und werfen Sie andernfalls einen `ValueError`.
Stellen Sie außerdem sicher, dass das zurückgegebene Array nicht länger ist als nötig. Wenn Ihre Implementierung also $N$ Iterationen benötigt um zu terminieren, dann soll das zurückgegebene Array die Länge $N + 1$ besitzen und die Einträge $x^{(0)}, x^{(1)}, \dots, x^{(N-1)}, x^{(N)}$ beinhalten.

### 5.1b)

Als nächstes sollen Sie das globalisierte Newton-Verfahren implementieren. Vervollständigen Sie hierzu die Methode `newton_global`, welche die selben Inputs wie `newton` sowie zusätzlich die optionalen Parameter `beta: float` mit Standardwert $0.5$ und `delta: float` mit Standardwert $10^{-3}$ übergeben bekommt. Der Rückgabewert ist wieder das Array aller Iterierten $x^{(k)}$. Stellen Sie auch hier wieder sicher, dass das zurückgegeben Array nicht länger als nötig ist. Die globalisierte Version soll außerdem die selben Fehlermeldungen wie die lokale Version werfen.
Der Unterschied zur lokalen Version ist, dass Sie in der globalen Version die Schrittweitensuche nach Armijo implementieren sollen (siehe (5.4) im Skript). Hierfür werden die neuen Inputparameter `beta` und `delta` benötigt[1].
Wenn es mit der Schrittweitensuche nach Armijo nicht gelingt eine passende Schrittweite zu finden, werfen Sie einen `RuntimeError`. Dies ist dann gegeben, wenn $\sigma_k$ zu klein wird. Verwenden Sie daher $\sigma_k < 10^{-12}$ als Abbruchkriterium.

## Aufgabe 5.2: Plots (10 Punkte)

In dieser Aufgabe sollen Sie das lokale und globalisierte Newtonverfahren zur Nullstellensuche verwenden und Ihre Ergebnisse graphisch darstellen. Denken Sie daran, jeder Plot benötigt Achsenbeschriftungen, eine Legende sowie einen Titel. Außerdem sollen alle geforderten Plots bei Ausführung von `main.py` automatisch und ohne äußeres Zutun erstellt und unter dem vorgegebenen Namen als `PDF` exportiert werden.

### 5.2a)

Betrachten Sie die Funktion $f(x) = \frac{1}{20}x^3 + x - 2 + \cos(\frac{6}{5}x)$, welche eine Nullstelle bei $\bar{x} \approx 2.314\,487\,789\,99$ hat. Sie sollen nun das Newtonverfahren auf $f$ anwenden. Verwenden Sie hierbei als Startwert $x^{(0)} = 3.5$ und übergeben Sie zusätzlich `maxiter=` $30$. Plotten Sie nun $f$ auf dem Intervall $[2, 4]$ und fügen Sie eine horizontale Linie bei $y = 0$ und eine vertikale Linie bei $\bar{x}$ hinzu. Zeichnen Sie die Iterierten $x^{(k)}$ mit zugehörigen Funktionswerten $f(x^{(k)})$ als Kreuze ein und schreiben Sie den Index der Iterierten explizit dazu. Sprich, wenn Sie den Startpunkt $(x_0, f(x_0))$ mit einem Kreuz markieren, dann soll neben diesem Punkt $0$ stehen. Exportieren Sie den Plot als `Local_Newton_Ex_1.pdf`.
Was können Sie beobachten? Halten Sie Ihre Beobachtungen in ein bis zwei Sätzen schriftlich in `beobachtungen.txt` fest.

### 5.2b)

Betrachten Sie nun die Funktion $f(x) = x^3 - 2x + 2$ mit Nullstelle $\bar{x} \approx -1.7693$. Berechnen Sie die Nullstelle sowohl mit dem lokalen als auch dem globalisierten Newtonverfahren. Wählen Sie hierfür $x_0 = 0$, $\beta = 0.33$ und $\delta = 10^{-3}$.
Plotten Sie $f$ auf $[-2, 2]$ und fügen Sie wieder eine horizontale Linie bei $y = 0$ und eine vertikale Linie bei $\bar{x}$ hinzu. Plotten Sie die Ergebnisse der beiden Newtonverfahren wie in voriger Teilaufgabe beschrieben, sprich, mit Kreuzen und Nummerierung. Verwenden Sie unterschiedliche Symbole für das lokale und das globale Newtonverfahren. Exportieren Sie den Plot als `Local_vs_Global.pdf`.
Erstellen Sie einen weiteren Plot und plotten Sie $|x^{(k)} - \bar{x}|$ über $k$ sowohl für das lokale als auch das globale Verfahren. Beide Varianten sollen im **selben** Plot dargestellt werden, welchen Sie als `Local_vs_Global_Error.pdf` exportieren lassen sollen.

[1] Im Skript wird $\sigma_k \in \{1, \frac{1}{2}, \frac{1}{4}, \dots\}$ betrachtet. Sie sollen hier die allgemeinere Version $\sigma_{k+1} = \beta \cdot \sigma_k$ mit $\sigma_0 = 1$ implementieren. Für den Standardwert `beta`$=0.5$ erhalten Sie jedoch genau den Fall aus dem Skript.

Was stellen Sie fest? Halten Sie Ihre Beobachtungen schriftlich in ein bis zwei Sätzen in `beobachtungen.txt` fest.
**Hinweis:** Ziel dieser Aufgabe ist es, festzustellen, dass es Fälle gibt, in denen zwar das globale Newtonverfahren konvergiert, nicht aber das lokale.

### 5.2c)

Als letzte Teilaufgabe sollen Sie einen Konvergenzplot für das lokale Newtonverfahren anfertigen. Betrachten Sie hierfür die Funktion $f(x) = x^n - 2,\ n \in \{2, 3, 4\}$ mit Nullstelle $\bar{x}_n = \sqrt[n]{2}$ und Startwert $x_0 = 0.5$. Plotten Sie $|x_n^{(k)} - \bar{x}_n|$ für $n \in \{2, 3, 4\}$ im selben Plot und wählen Sie eine semilogarithmische Darstellung mit logarithmischer $y$-Achse. Lassen Sie das Ergebnis als `Convergence_Plot.pdf` exportieren.
Was stellen Sie fest? Halten Sie Ihre Beobachtungen schriftlich in ein bis zwei Sätzen in `beobachtungen.txt` fest.

## Aufgabe 5.3: Tests

Damit Sie selbst einen Eindruck davon erhalten können wie automatisierte Tests in `Python` aussehen, stellen wir Ihnen die Datei `test.py` zur Verfügung, welche bereits Beispieltests beinhaltet. Diese überprüfen lediglich, dass Ihre Implementierung den korrekten Input entgegennimmt und der Output das richtige Format hat. Gerne dürfen Sie für sich selbst weitere Tests ergänzen, um Ihre Implementierung hiermit zu überprüfen. Das wird aber nicht bewertet.