Hinweis: Für die praktische Aufgabe müssen Sie `hdnum` auf den aktuellen Stand bringen, andernfalls wird die Newton-Methode nicht für komplexe Zahlen kompilieren. Dazu können Sie in Ihrem hdnum-Ordner die folgenden Kommandos ausführen:

```
git stash
git pull
git stash pop
```

Übung 1 *Lösen als Minimierungsproblem*

Wir betrachten in dieser Aufgabe die Lösung eines linearen Gleichungssystemes $Ax = b$ als Minimierungsaufgabe. Dazu nehmen wir an, dass $A \in \mathbb{R}^{n\times n}$ symmetrisch und positiv definit ist und betrachten die Abbildung

$$F : \mathbb{R}^n \to \mathbb{R}, \qquad F(x) = \frac{1}{2}\left(Ax, x\right)_2 - \left(b, x\right)_2 . \tag{1}$$

1. Zeigen Sie, dass gilt
$$\nabla F(x) = Ax - b.$$
2. Zeigen Sie nun die folgende Aussage: x^* ist Lösung von $Ax = b$ genau dann wenn x^* das eindeutige Minimum der Abbildung F ist.
3. Es seien nun $x, p \in \mathbb{R}^n$ mit $p \neq 0$ gegeben. Zeigen Sie, dass $g(\alpha) = F(x + \alpha p)$ sein Minimum bei
$$\alpha = \frac{(p, b - Ax)_2}{(p, Ap)_2}.$$
annimmt.

(5 Punkte)

Übung 2 *Wurzelberechnung*

Wir betrachten die Berechnung der Quadratwurzel von $\mathbb{R} \ni a > 0$, also die Lösung der Gleichung

$$f(x) = x^2 - a = 0$$

mittels Relaxation und Newton-Verfahren.

1. Relaxation ist die Fixpunktiteration mit der Funktion
$$g_R(x) = x - \sigma f(x).$$
Zeigen Sie die folgenden Aussagen:
   (a) Für die Wahl $\sigma = 1/a$ gilt $g([0, a]) = [1, a/4 + 1]$.
   (b) Für $a > 1$, $\sigma = 1/a$ und $0 < x, y < a$ ist g_R eine Kontraktion, d.h.
   $$|g(x) - g(y)| < |x - y|.$$
   (c) Die Konvergenzrate (der Kontraktionsfaktor) q geht für $x^k \to \sqrt{a}$ gegen $1 - 2\sqrt{a}/a$.

2. Betrachten Sie nun das Newton-Verfahren mit der Iterationsfunktion

$$g_N(x) = x - \frac{x^2 - a}{2x}.$$

Zeigen Sie: Es gibt ein $\epsilon > 0$ so dass g_N ein Kontraktion ist für $|x - \sqrt{a}|, |y - \sqrt{a}| < \epsilon$. Gehen Sie dazu wie folgt vor. Mit der Taylorentwicklung von g_N und einem beliebigen z zeigen Sie

$$g_N(x) - g_N(y) = g'_N(z)(x - y) + g''_N(\eta_x)(x - z)^2 + g''_N(\eta_y)(y - z)^2.$$

Nun setzen Sie $z = (x + y)/2$ und argumentieren von da aus.

3. Zeigen Sie: Das Newton-Verfahren konvergiert global für alle Startwerte $x^0 > 0$ und jedes $a > 0$. Gehen Sie dazu wie folgt vor: Zeigen Sie, wenn $x^k > \sqrt{a}$ dann gilt $\sqrt{a} < x^{k+1} < x^k$, also monotone Konvergenz gegen die Nullstelle von oben. Nun überlegen Sie was für $0 < x^k < \sqrt{a}$ passiert. Sie können geometrisch argumentieren. Interpretieren Sie dazu das Newton-Verfahren wie folgt: Für gegebenes x^k lege die Tangente an f im Punkt $(x^k, f(x^k))$, also

$$T_f(x^k) = f'(x^k)(x - x^k) + f(x^k).$$

Nun berechnen Sie x^{k+1} als Nullstelle der Tangente, also $T_f(x^{k+1}) = 0$. Überlegen Sie, dass dies genau das Newton-Verfahren ist.

(5 Punkte)

Übung 3 *Nichtlineares Lösen als Minimierungsproblem*

Sei $f : \mathbb{R}^n \to \mathbb{R}^n$. Nun betrachten wir die Lösung der nichtlinearen Gleichung $f(x) = 0$ als Minimierungsaufgabe in dem wir Minimalpunkte von

$$F(x) = \|f(x)\|_2$$

suchen. Folgendes Verfahren soll dabei zum Einsatz kommen: Für gegebenes x^k und $\mathbb{R}^n \ni p^k \neq 0$ setzen wir an

$$x^{k+1} = x^k + \alpha p^k$$

und bestimmen α so dass $F(x^k + \alpha p^k)$ *näherungsweise* minimiert wird. Dazu nutzen wir die Taylorentwicklung $f(x^k + \alpha p^k) \doteq f(x^k) + \alpha J_f(x^k)p^k$ und minimieren die Funktion

$$H(\alpha) = \|f(x^k) + \alpha J_f(x^k)p^k\|_2.$$

Zeigen Sie, dass H minimal wird für

$$\alpha_{opt} = -\frac{(f(x^k), J_f(x^k)p^k)_2}{(J_f(x^k)p^k, J_f(x^k)p^k)_2}$$

und geben Sie die Iteration

$$x^{k+1} = x^k + \alpha_{opt} p^k$$

jeweils für folgende Wahlen von p^k an:

1. Relaxation $p^k = f(x^k)$.
2. Gradientenverfahren: Bestätigen Sie $p^k = -\nabla F(x^k) = -2J_f(x^k)f(x^k)$.
3. Newton-Verfahren: $p^k = J_f^{-1}(x^k)f(x^k)$.

(5 Punkte)

Übung 4 *Nichtlineare Löser (Praktische Übung)*

In dieser Aufgabe werden wir uns mit nichtlinearen Problemen befassen. Der größte Teil des Programmcodes ist bereits vorgegeben, daher keine Angst vor dem langen Text :)

Nullstellengolf Wir beginnen mit dem Problem aus der Vorlesung: Finde x sodass $f(x) = 0$, wobei

$$f_1(x_1, x_2) = x_1^2 + x_2^2 - 4 = 0,$$
$$f_2(x_1, x_2) = \frac{x_1^2}{9} + x_2^2 - 1 = 0.$$

Die Nullstellen sind hier die Schnittpunkte eines Kreises und einer Ellipse. Nutzen Sie im Folgenden das Programmgerüst `prog_nonlinear_solvers_methods.cc`.

- Implementieren Sie die obenstehende Funktion samt ihrer Jacobimatrix in der Problemklasse.
- Finden Sie mit dem Newton-Verfahren durch Variieren der Startpunkte numerisch alle Nullstellen. Geben Sie diese Nullstellen x_i sowie $f(x_i)$ an. Als Qualitätsmerkmal sollte mindestens $||f(x_i)||_2 < 10^{-6}$ gelten. Für jede Nullstelle gibt es $1/4$ Punkte.
- Bringen Sie das Relaxationsverfahren für eine dieser Nullstellen zum Konvergieren. Geben Sie das Resultat, die gewählten Parameter, den Startpunkt und die Anzahl der benötigten Iterationen an.

Malen nach Nullstellen Wie Sie vielleicht schon im ersten Teil bemerkt haben, ist nicht unbedingt klar, zu welcher Nullstelle ein gewisser Startpunkt führt. Manche können unter Umständen sogar zu nicht konvergierenden Lösern führen. Besonders interessant ist das Verhalten bei komplexwertigen Polynomen. Wir betrachten

$$f(x) = x^3 - 2x + 2,$$

wobei $f : \mathbb{C} \to \mathbb{C}$. Nutzen Sie im Folgenden das Programmgerüst `prog_nonlinear_solvers_fractal.cc`. Da `hdnum` den benutzten Zahlentyp als Template offen lässt, können wir ihn ohne Weiteres auf `std::complex<double>` umstellen. Dieser Typ repräsentiert komplexe Zahlen, wobei in diesem Fall reelle und imaginäre Komponente als `double` dargestellt werden, Beispiele dazu finden Sie im Internet.

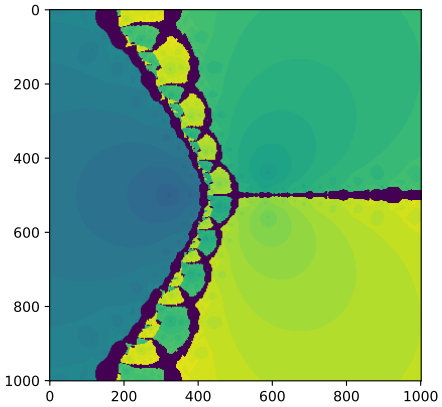

Abbildung 1: Ausgabe für $x^3 - 2x + 2$ auf dem Gebiet $[-5, 5] \times [-5i, 5i]$.

Wir wollen nun betrachten zu welchen Nullstellen diverse Startpunkte im Newton-Verfahren führen. Wenn die Startpunkte in einem Raster durchprobiert und je nach getroffener Nullstelle farbkodiert werden, führt das zu einem Fraktal wie in Abbildung 1.

- Gehen Sie das vorgegebene Programm durch, verstehen Sie dessen Funktion und testen Sie es. Die Ausgabe des Programms können Sie mithilfe der Kommandozeile in eine Datei namens 'out' schreiben lassen:

  `./prog_nonlinear_solvers_fractal > out`

  Rufen Sie nun das Python-Skript mit

  `python3 prog_nonlinear_solvers_fractal.py`

  im gleichen Ordner auf. Es liest die Ausgabe aus 'out' und generiert das zugehörige Bild. Verändern Sie das Raster der Startpunkte und zeigen Sie das resultierende Bild.

  (Falls nötig können Sie Python über `sudo apt-get install python3 python3-pip` auf Ihrem Linux-System installieren. Das Skript benötigt weiterhin `matplotlib` und `numpy`, zu installieren via `pip3 install matplotlib` und `pip3 install numpy`.)

- Modifizieren Sie den Code für ein Polynom mit vier komplexwertigen Nullstellen. Insbesondere müssen Sie dazu die Identifikation der Nullstellen für Ihr Polynom anpassen. Zeigen Sie die resultierende Grafik.

(1+1+1+1+1 Punkte)