Projekt: Schwärme – Simulation und Anwendung.

Abgabe bis: **30.06.2024**

In diesem Projekt werden wir uns mit Schwärmen und damit verwandten Algorithmen beschäftigen. Im ersten Teil des Projekts implementieren wir Methoden zur Approximation der Lösung einer gewöhnlichen Differentialgleichung, die die Bewegung einzelner Individuen in einem Schwarm beschreibt. Im zweiten Teil des Projektes betrachten wir eine auf Schwärme basierende Methode, um das globale Minimum einer Funktion zu finden.

Teil 1: Simulation eines Schwarms

Cucker-Smale-Modell

Im Artikel von Felipe Cucker und Steve Smale [a] ist die mathematische Beschreibung der Bewegung von einzelnen Individuen eines Schwarms zu finden. Dazu wird ein System von N Individuen oder Agenten betrachtet. Der i-te Agent wird durch seine Position $x_i : [T_{init}, T_{end}] \to \mathbb{R}^2$ und seine Geschwindigkeit $v_i : [T_{init}, T_{end}] \to \mathbb{R}^2$ eindeutig beschrieben. All diese Bewegungs- und Geschwindigkeitsdaten werden gespeichert in den Vektoren

$$\boldsymbol{x}(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \\ \vdots \\ x_N(t) \end{pmatrix} \in \mathbb{R}^{N\times 2} \quad \text{und} \quad \boldsymbol{v}(t) = \begin{pmatrix} v_1(t) \\ v_2(t) \\ \vdots \\ v_N(t) \end{pmatrix} \in \mathbb{R}^{N\times 2}.$$

Die Evolution der Position und Geschwindigkeit der einzelnen Agenten über den Zeitraum $[T_{init}, T_{end}]$ wird nun durch das System von Differentialgleichungen

$$\begin{aligned} \dot{\boldsymbol{x}}(t) &= \boldsymbol{v}(t), \\ \dot{\boldsymbol{v}}(t) &= -\boldsymbol{L}_x \boldsymbol{v}(t) = -(\boldsymbol{D}_x - \boldsymbol{A}_x)\boldsymbol{v}(t) \end{aligned} \tag{1}$$

beschrieben, wobei die Ableitungen $\dot{\boldsymbol{x}}(t) := \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}(t)$ und analog $\dot{\boldsymbol{v}}(t) := \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{v}(t)$ komponentenweise zu verstehen sind. Die Adjazenzmatrix $\boldsymbol{A}_x$ ist gegeben durch

$$\boldsymbol{A}_x = \left[a_{i,j}\right]_{i,j=1}^{N} \quad \text{mit} \quad a_{i,j} := \eta\left(\|x_i - x_j\|_2\right), \quad \text{wobei} \quad \eta(y) = \frac{K}{(\sigma^2 + y)^\beta}.$$

Der Eintrag $a_{i,j}$ beschreibt den Einfluss des Agenten j auf die Geschwindigkeit des Agenten i. Die Parameter $K, \sigma > 0$ und $\beta \geq 0$ und deren Einfluss auf die Simulation des Schwarms werden wir uns später anschauen. Schliesslich besitzt die Diagonalmatrix $\boldsymbol{D}_x$ die Einträge

$$\boldsymbol{D}_x = \mathrm{diag}(d_1, d_2, \ldots, d_N) \quad \text{mit} \quad d_i := \sum_{j=1}^{N} a_{i,j}.$$

[a] Felipe Cucker und Steve Smale. Emergent Behavior in Flocks. *IEEE Transactions on automatic control*, **52**:852–862 (2007).

1.1) Implementiere die Funktion `dxvdt = sysDiffGlgen(x,v,params)`.
Diese Funktion erhält als Eingabewerte die Matrizen der Positionen $x(t) \in \mathbb{R}^{N\times 2}$ und der Geschwindigkeiten $v(t) \in \mathbb{R}^{N\times 2}$ zum Zeitpunkt t und die drei Parameter K, σ und β im MATLAB-`struct`-Objekt `params`.

Mit dieser Funktion werden die Matrizen $\boldsymbol{A}_x$ und $\boldsymbol{D}_x$ erstellt und gibt die Terme zurück, die auf der rechten Seite der Gleichungen im Differentialgleichungssystem (1) stehen. Die Matrix `dxvdt` hat daher die Dimension $N \times 4$

Explizites Euler-Verfahren

Für das Euler-Verfahren wird die Zeitableitung formuliert gemäss

$$\lim_{\Delta t \to 0} \frac{\boldsymbol{x}(t+\Delta t) - \boldsymbol{x}(t)}{\Delta t} = \dot{\boldsymbol{x}}(t),$$
$$\lim_{\Delta t \to 0} \frac{\boldsymbol{v}(t+\Delta t) - \boldsymbol{v}(t)}{\Delta t} = \dot{\boldsymbol{v}}(t).$$

Die Gleichungen aus (1) werden auf der rechten Seite dieser Formel eingesetzt. Wenn nun die Zeitschrittweite $\Delta t > 0$ als konstante, kleine Zahl gewählt wird und alle bekannten Einträge auf die rechte Seite gebracht werden, entsteht das explizite Euler-Verfahren:

$$\begin{aligned} \boldsymbol{x}^{(i+1)} &= \boldsymbol{x}^{(i)} + \Delta t\, \boldsymbol{v}^{(i)}, \\ \boldsymbol{v}^{(i+1)} &= \boldsymbol{v}^{(i)} + \Delta t \left(-\boldsymbol{L}_x \boldsymbol{v}^{(i)} \right), \\ t^{(i+1)} &= t^{(i)} + \Delta t = (i-1)\Delta t \quad \text{und} \quad \boldsymbol{x}^{(i+1)} \approx \boldsymbol{x}(t^{(i+1)}),\ \boldsymbol{v}^{(i+1)} \approx \boldsymbol{v}(t^{(i+1)}). \end{aligned} \tag{2}$$

1.2) Implementiere die Funktion `[x,v] = explEulSchwarm(tspan,x0,v0,params)`.
Diese Funktion erhält als Eingabewerte den Startzeitpunkt T_{init}, den Endzeitpunkt T_{end} und die Zeitschrittweite Δt im Vektor `tspan` der Länge drei, die Matrizen der Positionen $x(T_{init}) \in \mathbb{R}^{N\times 2}$ und der Geschwindigkeiten $v(T_{init}) \in \mathbb{R}^{N\times 2}$ zum Zeitpunkt T_{init} und die Parameter K, σ und β im MATLAB-`struct`-Objekt `params`.
Diese Funktion soll das explizite Euler-Verfahren (2) vom Zeitpunkt T_{init} bis zum Zeitpunkt T_{end} mit Zeitschrittweite Δt ausführen. Die Positionen und Geschwindigkeiten von allen N Individuen sollen in `x` bzw. `v` abgespeichert werden.
Hinweis: Nutze die Funktion **sysDiffGlgen** aus Aufgabe 1.1).

Heun-Verfahren

Eine andere Methode, eine Lösung des Problems (1) zu approximieren, bieten die Runge-Kutta-Verfahren. Für das Runge-Kutta-Verfahren zweiter Ordnung, auch das Verfahren von Heun genannt, wird ein Schritt des expliziten Euler-Verfahrens berechnet und die Resultate dieser Rechnung nennen wir $\tilde{\boldsymbol{x}}^{(i+1)}$ und $\tilde{\boldsymbol{v}}^{(i+1)}$. Nachfolgend wird die Approximation von $\boldsymbol{x}^{(i+1)}$ und $\boldsymbol{v}^{(i+1)}$ durch das Heun-Verfahren berechnet mit den Formeln

$$\begin{aligned} \boldsymbol{x}^{(i+1)} &= \boldsymbol{x}^{(i)} + 0.5\,\Delta t \cdot \left(\boldsymbol{v}^{(i)} + \tilde{\boldsymbol{x}}^{(i+1)} \right), \\ \boldsymbol{v}^{(i+1)} &= \boldsymbol{v}^{(i)} + 0.5\,\Delta t \cdot \left(-\boldsymbol{L}_x \boldsymbol{v}^{(i)} + \tilde{\boldsymbol{v}}^{(i+1)} \right). \end{aligned} \tag{3}$$

1.3) Implementiere eine Funktion `[x,v] = heunSchwarm(tspan,x0,v0,params)`.
Diese Funktion erhält dieselben Eingabewerte wie die Funktion **explEulSchwarm** aus Aufgabe 1.2).
Die Funktion in dieser Aufgabe soll das Heun-Verfahren für die Gleichungen (3) vom Zeitpunkt T_{init} bis zum Zeitpunkt T_{end} mit Zeitschrittweite Δt ausführen. Auch hier sollen die Positionen und Geschwindigkeiten von allen N Individuen in `x` bzw. `v` abgespeichert werden.
Hinweis: Nutze die Funktionen **sysDiffGlgen** aus Aufgabe 1.1) und **explEulSchwarm** aus Aufgabe 1.2).

1.4) Erstelle ein MATLAB-Skript **A1_1**.
Darin soll ein Schwarm mit $N = 50$ Individuen mit zufälligen Positionen im Gebiet $\Omega = [-5, 5] \times [-5, 5]$ und zufälligen Geschwindigkeiten, die in jeder Komponente im Bereich $[-1, 1]$ sind, initiiert werden. Dazu kann der MATLAB-Befehl `rand` benutzt werden. Setze die Parameter auf $K = 1$, $\alpha = 1$ und $\beta = 1$. Die Zeitschrittweite soll in dieser Aufgabe auf $\Delta t = 5 \times 10^{-2}$ und der Zeitbereich auf $t \in [0, 10]$ gesetzt werden.
Nutze nun deine Implementierungen von explizitem Euler- und Heun-Verfahren, um einen Schwarm zu simulieren. Sind Unterschiede in den Simulationen mit den zwei Verfahren zu sehen?
Erstelle mit dem Befehl `subplot` ein 4×4 Gitter, in dem 16 Bilder zu verschiedenen Zeiten des Schwarms angezeigt werden sollen. In jedem einzelnen Bild soll jedes Individuum des Schwarms als einzelner Punkt zu sehen sein. Zusätzlich soll die zu jedem Individuum dazugehörende Geschwindigkeit als vom Punkt ausgehender Pfeil angezeigt werden. Dafür kann der Befehl `quiver` nützlich sein. Die 16 Zeitpunkte, von denen Bilder eingefügt werden, können frei gewählt werden. Füge zu jedem `subplot` einen Titel hinzu, in dem steht, welcher Zeitpunkt ausgesucht wurde.
Bemerkung: Mit der Befehlsfolge

```
currObj = VideoWriter("VideoName" ,"MPEG-4");
currObj.FrameRate = 20;
open(currObj)
figure(2)
for jj= % Iteration über das gesamte Zeitintervall
    clf
    % plot-Erstellung
    axis([-10 10 -10 10])
    frame = getframe(gcf);
    writeVideo(currObj,frame);
end
close(currObj)
```

kann ein Animationsvideo von der Simulation erstellt werden. Die Erstellung einer Video-Animation der Simulation ist **optional**.

1.5) Der Parameter K bestimmt die Interaktionsstärke zwischen Individuen. Der Parameter σ kann als Regularisierungsparameter verstanden werden. Der Parameter β bestimmt die Abklingrate der Beeinflussung über die Distanz von einzelnen auf andere Individuen des Schwarms. Im Cucker-Smale-Modell ist zu bemerken, dass sich nach einer gewissen Zeit die Geschwindigkeiten der Agenten in Richtung und Grösse synchronisiert haben. Diese Parameter beeinflussen diese Synchronisation.
Ändere im Skript **A1_1** jeweils einen der drei Parameter K, σ oder β, während die zwei anderen unverändert den Wert von Aufgabe 1.4) behalten. Welcher Einfluss ist bei jedem einzelnen der drei Parameter auf die Bewegung der Individuen im Schwarm festzustellen? Beschreibe im Bericht deine Beobachtungen zu jedem Parameter.

1.6) Erstelle ein MATLAB-Skript **A1_2**.
Darin soll analog zu Aufgabe 1.4) ein Schwarm mit $N = 20$ Schwarmindividuen mit zufälligen Positionen und Geschwindigkeiten initiiert werden. Wähle drei Werte für die Parameter K, α und β.
Nutze nun deine Implementierung des Heun-Verfahrens, um einen Referenzlauf mit $t \in [0, 1]$ und kleiner Zeitschrittweite $\Delta t = 10^{-6}$ zu erstellen. Nenne die erhaltenen Werte x_{ref} und v_{ref}.
Beachte: Da die Zeitschrittweite klein gewählt ist, kann diese Rechnung länger dauern als bisherige Rechnungen. Damit die Rechnung nicht allzu lange dauert, soll mit dem kleineren Zeitintervall $t \in [0, 1]$ gerechnet werden.

Nutze nun deine Implementierungen des expliziten Euler-Verfahrens und des Heun-Verfahrens, um Simulationsläufe mit $t \in [0, 1]$ und Zeitschrittweiten $\Delta t = 0.1, 0.05, 0.01, 0.005, 0.001$ zu erstellen. Nenne die Resultate dieser Rechnungen x_i^{EE} und v_i^{EE}, bzw. x_i^H und v_i^H für $i = 1, \dots, 5$.
Erstelle zwei Vektoren **`err_ExpEul`** und **`err_Heun`** der Länge fünf mit den Einträgen

$$\|x_{ref}(1) - x_i^{EE}(1)\|_2 + \|v_{ref}(1) - v_i^{EE}(1)\| \quad \text{bzw.} \quad \|x_{ref}(1) - x_i^{H}(1)\|_2 + \|v_{ref}(1) - v_i^{H}(1)\|.$$

Zeichne diese zwei Fehler in einen gemeinsamen Plot. Füge in diesen Plot eine Vergleichsgerade hinzu, die sich linear zu den Zeitschrittweiten Δt entwickelt. Füge in diesen Plot eine zweite Vergleichsgerade hinzu, die sich quadratisch zu den Zeitschrittweiten Δt entwickelt.
Füge diesen Plot in deinen Bericht ein. Schreibe zu dem Plot einen Kommentar, insbesondere dazu, wie sich die Fehler des expliziten Euler-Verfahrens und des Heun-Verfahrens im Vergleich zu den Vergleichsgeraden entwickeln.

Teil 2: Partikelschwarmoptimierung

Die Bewegungen von Individuen in einem Schwarm inspirierten die Entwicklung von Algorithmen, die bei der Findung von Minima eingesetzt werden. Wir betrachten also das Problem: Gegeben ist eine Funktion $f : \mathbb{R}^2 \to \mathbb{R}$ und auf einem Gebiet $\Omega \subset \mathbb{R}^2$ ist gesucht

$$\min_{x \in \Omega} f(x).$$

Zwei Funktionen, die wir in diesem Projektteil betrachten, um die Stärken und Schwächen dieses Algorithmus zu untersuchen, sind die Levy-Funktion und die Shubert-Funktion [a], beides Funktionen von $\mathbb{R}^2 \to \mathbb{R}$. Die Levy-Funktion ist gegeben durch die Formel

$$L(\mathbf{x}) = \sin^2(\pi w_1) + (w_1 - 1)^2 \left[1 + 10 \sin^2(\pi w_1 + 1)\right] + (w_2 - 1)^2 \left[1 + \sin^2(2\pi w_2)\right] \quad (4)$$

mit $w_i = 1 + \frac{x_i - 1}{4}$ für $i = 1, 2$. Die Shubert-Funktion wiederum ist beschrieben durch die Formel

$$S(\mathbf{x}) = \left(\sum_{i=1}^{5} i \cos\left((i+1)x_1 + i\right)\right)\left(\sum_{i=1}^{5} i \cos\left((i+1)x_2 + i\right)\right). \quad (5)$$

[a] Sonja Surjanovic und Derek Bingham. Virtual Library of Simulation Experiments: Test Functions and Datasets. Verfügbar unter https://www.sfu.ca/ ssurjano/optimization.html. Simon Fraser Universität, zuletzt zugegriffen im März 2024.

2.1) Erstelle ein MATLAB-Skript **`A2_1`**.
Plotte hier die Levy-Funktion (4) für $\boldsymbol{x} \in \Omega_L := [-10, 10]^2$ und die Shubert-Funktion (5) für $\boldsymbol{x} \in \Omega_S := [-5.1, 5.1]^2$ jeweils mit dem Befehl `surf` und dem Befehl `contour`. Füge die Plots in deinen Bericht ein. Welche lokalen und globalen Minima haben diese Funktionen und inwiefern unterscheiden sie sich diesbezüglich voneinander? Schreibe dazu einen kurzen Kommentar in deinen Bericht.

Partikelschwarmoptimierung

Die Partikelschwarmoptimierung ist ein stochastisches, iteratives Verfahren zur Bestimmung des globalen Minimums einer Funktion.
Gegeben ist eine Funktion $f : \mathbb{R}^2 \to \mathbb{R}$, ein Suchgebiet $\Omega \subset \mathbb{R}^2$ und eine Anzahl N an

Partikeln. Die N Partikel haben jeweils eine gegebene Initialposition $\boldsymbol{x}_1^{(1)}, \boldsymbol{x}_2^{(1)}, \dots, \boldsymbol{x}_N^{(1)} \in \Omega$ und Initialgeschwindigkeit $\boldsymbol{v}_1^{(1)}, \boldsymbol{v}_2^{(1)}, \dots, \boldsymbol{v}_N^{(1)} \in \mathbb{R}^2$. Weiter sind für den Algorithmus gegeben:

- die maximale Anzahl an Iterationen I_{max}, bevor die Optimierung abgebrochen wird,
- die *Trägheit der Bewegung* $\omega \in [0, 1]$,
- der *kognitive Gewichtungsfaktor* $c_k \in \mathbb{R}$ und
- der *soziale Gewichtungsfaktor* $c_s \in \mathbb{R}$.

Zu Beginn des Algorithmus ist die für jeden Partikel individuell bisher beste Position $\boldsymbol{b}_i^{(1)}$, an der der Funktionswert minimal ist, die initiale Position $\boldsymbol{x}_i^{(1)}$. Die globale beste Position $\boldsymbol{g}^{(1)} \in \mathbb{R}^2$, an der die Funktion den kleinsten Wert annimmt, wird abgespeichert. Es gilt damit: $f(\boldsymbol{g}^{(1)}) \leq f(\boldsymbol{b}_i^{(1)}) = f(\boldsymbol{x}_i^{(1)})$ für alle $i \in \{1, 2, \dots, N\}$.
Für alle Partikel wird nun die Geschwindigkeit aufdatiert mit der Formel

$$\boldsymbol{v}_i^{(2)} = \omega \boldsymbol{v}_i^{(1)} + c_k \cdot r \cdot \left(\boldsymbol{b}_i^{(1)} - \boldsymbol{x}_i^{(1)}\right) + c_s \cdot s \cdot \left(\boldsymbol{g}^{(1)} - \boldsymbol{x}_i^{(1)}\right) \quad \text{für alle } i \in \{1, 2, \dots, N\},$$

wobei $r, s \in [0, 1]$ zufällige Werte sind.
Die neuen Positionen werden anschliessend bestimmt als $\boldsymbol{x}_i^{(2)} = \boldsymbol{x}_i^{(1)} + \boldsymbol{v}_i^{(2)}$ für alle $i \in \{1, 2, \dots, N\}$. Sollte ein Partikel nun ausserhalb des Gebiets Ω sein, so wird dieser an den Rand des Gebiets gesetzt.
Mit den neuen Positionen $\boldsymbol{x}_i^{(2)}$ kann nun erneut die für jeden Partikel individuell bisher beste Position $\boldsymbol{b}_i^{(2)}$ und die bisher global beste Position $\boldsymbol{g}^{(2)}$ bestimmt werden. Damit gilt nun

$$f(\boldsymbol{g}^{(2)}) \leq f(\boldsymbol{g}^{(1)}) \quad \text{und} \quad f(\boldsymbol{g}^{(2)}) \leq f(\boldsymbol{b}_i^{(2)}) \leq f(\boldsymbol{x}_i^{(j)}) \quad \text{für alle } i \in \{1, 2, \dots, N\} \text{ und } j = 1, 2.$$

In $\boldsymbol{g}^{(I_{max})}$ ist nach der maximalen Anzahl an Iterationen I_{max} die beste Approximation der Position des globalen Minimums, die durch den Algorithmus bestimmt wurde, gespeichert.

2.2) Schreibe ein Funktion `[bestX,bestFx,xHist,bHist] = PSO(funcH,gebiet,params)`.
Diese Funktion erhält als Eingabewerte ein MATLAB-"function handle" `funcH` passend zur Funktion, für die die Position des globalen Minimums angenähert werden soll. Weiter sind in dem Vektor `gebiet` die Werte der Intervallgrenze des Gebiets gegeben. Im MATLAB-`struct` `params` sind die Werte N der Anzahl an Partikel, die maximale Anzahl I_{max} an Iterationsschritten und die drei Parameter ω, c_k und c_s gegeben.
In der Funktion soll die Partikelschwarmoptimierung so wie oben beschrieben durchgeführt werden. Der Rückgabewert `bestX` soll die angenäherte Position des globalen Minimas (also $\boldsymbol{g}^{(I_{max})}$) sein, `bestFx` der Wert der Funktion in `bestX`, `xHist` die Werte aller Positionen von der Initialisierung bis zum letzten Iterationsschritt und `bHist` die Werte aller bisher besten Positionen des globalen Minimas in jedem Schritt. Diese Rückgabewerte sollen für jeden Iterationsschritt ermöglichen, dass ein Plot mit allen Partikeln gezeichnet werden kann.

2.3) Erstelle ein MATLAB-Skript **`A2_2`**.
Benutze in diesem Skript die von dir implementierte Funktion **`PSO`**, um das globale Minimum der Levy-Funktion (4) und der Shubert-Funktion (5) anzunähern. Wähle dazu geeignete Werte für N und I_{max}. Wähle die Werte $\omega = 0.7$, $c_k = 2.0$ und $c_s = 2.0$ für die weiteren Parameter.
Benutze anschliessend die Befehle `subplot` und `contour`, um in einem (4×4)-Plot-Gitter den Stand der Minimasuche in 16 voneinander verschiedenen Iterationsschritten darzustellen. Zeichne dazu in jedem abgebildeten Iterationsschritt die N Partikel an ihren Positionen ein. Füge diese Abbildung deinem Bericht hinzu mit einem kurzen Kommentar über den Erfolg der Minimasuchen.

Bemerkung: Mit der Befehlsfolge in der Bemerkung in Aufgabe 1.4) kann auch zu dieser Aufgabe hier ein Animationsvideo von der Minimasuche erstellt werden. Die Erstellung einer Video-Animation der Minimasuche ist auch hier **optional**.

Zufall in MATLAB

Mit dem Befehl `rng` lässt sich der Zufallszahlen-Generator von MATLAB kontrollieren.
Wenn der Befehl `rand` mehrmals ausgeführt wird, dann wird jedes Mal eine andere Zahl aus $[0, 1]$ angezeigt.
Wenn der Variablen `z` eine Zahl ≥ 0 zugewiesen wird und der Befehl `rng(z)` zusammen mit einer mehrmaligen Wiederholung des Befehls `rand` ausgeführt wird, dann wird jedes Mal *die gleiche* Zahlenfolge aus $[0, 1]$ angezeigt. Diese Zahlenfolge ist eine andere für jede Ausführung dieser Befehle mit einem anderen Wert der Variablen `z`.

2.4) Erstelle ein MATLAB-Skript **`A2_3`**.
In dieser Aufgabe sollen die Wahl der Werte für den kognitiven Gewichtungsfaktor c_k und den sozialen Gewichtungsfaktors untersucht werden. Gesucht ist das Minimum der Shubert-Funktion (5). Dabei soll der Befehl `rng` ausgenutzt werden, um mehrmals dieselbe Minimasuche durchführen zu können, unabhängig von der Zufallsgeneration der Zahlen in der Partikelschwarmoptimierung.
Die Werte der Parameter c_k und c_s sollen wie in Aufgabe 2.3) gewählt werden, oder einer der beiden Faktoren soll auf 0 gesetzt werden, während der andere Werte den Wert aus Aufgabe 2.3) behält.
Wie verändert sich die Minimasuche, wenn der kognitive Gewichtungsfaktor c_k auf 0 gesetzt wird? Was passiert, wenn der soziale Gewichtungsfaktor c_s auf 0 gesetzt wird? Was unterscheidet diese Minimasuchen von denen mit den Werten aus Aufgabe 2.3)? Führe für jeden Fall mehrere Minimasuchen aus, in denen du den Befehl `rng` ausnutzt. Schreibe einen Kommentar zu den Beobachtungen und Ergebnissen in den Bericht. Illustriere deine Beobachtungen falls erforderlich mit Plots von dem, was du beschreibst.

Abgabe. Als Abgabe soll ein Bericht in Form eines einzigen PDF-Dokumentes erstellt werden. In dieser PDF-Datei sollen enthalten sein:

- Antworten auf die Fragen zu den Aufgaben 1.4), 1.5), 1.6), 2.1), 2.3) und 2.4),
- Plots aus den Aufgaben 1.4), 1.6), 2.1) und 2.3), und
- den zur Lösung der Aufgaben erstellten MATLAB-Code, d.h. die Funktionen **`sysDiffGlgen`**, **`explEulSchwarm`**, **`heunSchwarm`** und **`PSO`** sowie die Skripte **`A1_1`**, **`A1_2`**, **`A2_1`**, **`A2_2`** und **`A2_3`**.

Es wurde eine mit LaTeXverfasste Musterabgabe für den Bericht in den Zusatzmaterialien bereitgestellt. In der `tex`-Datei wird unter anderem vorgeführt, wie du deine Plots und deinen MATLAB-Code in eine mit LaTeX erstellte PDF-Datei einfügen kannst.