# Modellierung WS 22-23 Blatt 11

## Abgabe: 03.02.23 23:59 auf Mattermost oder per Mail an jadissel

## Aufgabe 1

Wir wollen ein paar Lineare ODEs lösen (link zur vorlesung)

Sei

$$\frac{d}{dt}x(t) = Ax(t) = \begin{bmatrix} -5 & 1 \\ 4 & -2 \end{bmatrix} x(t)$$

mit $x(0) = \begin{bmatrix} -1/2 \\ 3 \end{bmatrix}$

1. Was sind die Eigenwerte von $A$?
2. Was sind die Eigenvektoren von $A$?
3. Löse die Lineare ODE mit dem Ansatz aus der Vorlesung

## Aufgabe 2

Der Ansatz aus den Folien funktioniert auch für komplexe Eigenwerte.

Sei

$$\frac{d}{dt}x(t) = Ax(t) = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} x(t)$$

1. Was sind die Eigenwerte von $A$? (Vorsicht, komplexe Zahlen!)
2. Was sind die Eigenvektoren von $A$?
3. Löse die Lineare ODE mit dem Ansatz aus der Vorlesung für $x(0) = \begin{bmatrix} a \\ b \end{bmatrix}$

# Aufgabe 3

Wir wollen eine kleine Simulation schreiben. Ladet hierzu zunächst das Python file herunter (Es ist möglich dass ihr zum Ausführen zuerst tkinter installieren müsst.)

Das skript enthält die Klasse `PointSimulation`, welche wiederholt Punkt an bestimmten koordinaten zeichnet.

Als Beispiel könnt ihr euch die Funktion `hook_law_simulation` ansehen. Sie nimmt die simulation als input an und manipuliert die Koordinaten und Geschwindigkeiten um eine einfache periodische simulation zu erzeugen.

Das Hooksche gesetz als Differenzialgleichung sieht folgendermaßen aus:

$$\frac{d}{dt}x_i(t) = v_i(t)$$

$$\frac{d}{dt}v_i(t) = -k(x_i(t) - center)$$

Wobei $k$ und $center$ konstanten sind. Im code können wir dies diskret lösen indem wir für kleine $dt$ berechnen:

$$x_i(t + dt) = x_i(t) + dt \cdot v_i(t)$$

1. Implementiert die Funktion `gravity_simulation_sun` auf folgende Art:

Wir setzen in die Mitte des Bildes (bei $M = [0.5, 0.5]$) eine Sonne die auf die Punkte einwirkt. Die Dynamiken sind dann folgendermaßen für jeden Punkt $P_i$ gegeben:

$$\frac{d}{dt}x_i(t) = v_i(t)$$

$$\frac{d}{dt}v_i(t) = -k\frac{x_i - M}{||x_i - M||_2^3}$$

Achtet darauf dass der Nenner sehr klein werden kann. Um Numerische explosionen zu vermeiden können wir ein kleines epsilon dazu addieren.
Sucht euch schöne werte für `line_length`, `number_of_points` aus.

2. Implementiert die Funktion `gravity_simulation_many_body` auf folgende Art:

Statt eine Sonne im Mittelpunkt zu haben wollen wir gravitation zwischen jedem paar an Punkten berechnen. Dies bedeutet dass unsere Dynamik nun folgendermaßen aussieht:

$$\frac{d}{dt}x_i(t) = v_i(t)$$

$$\frac{d}{dt}v_i(t) = -k\sum_{j\neq i}\frac{x_i - x_j}{||x_i - x_j||_2^3}$$

Auch hier müsst ihr aufpassen dass der Nenner nicht explodiert.