2. Gyakorló feladatsor

Python mérnöki alkalmazásai (*Monori Bence*)

1 Bevezető feladatok

1.1 NumPy Tömbműveletek - ujadat

Definiáljunk egy `A=np.array([])` tömböt és töltsük fel tetszés szerint néhány elemmel. Írassuk ki a tömb maximumát! Ezt követően fűzzünk hozzá egy felhasználótól bekért elemet és nézzük meg ezután is a tömb maximumát!

1.2 Plotolás - Kéttárolós rendszer

Adott egy kéttárolós tag, melynek válaszfüggvénye az alábbi alakban írható fel:

$$y(t) = A_0 \left[1 - e^{-\zeta\omega_n t}\left(\cos(\omega_d t) + \frac{\zeta}{\sqrt{1-\zeta^2}}\sin(\omega_d t)\right)\right] \tag{1}$$

ahol:

- $y(t)$ a rendszer válaszfüggvénye
- A_0 a rendszer *állandósult értéke*
- ω_n a rendszer *csillapítatlan sajátkörfrekvenciája*
- ζ a rendszer csillapítása
- ω_d a rendszer csillapított sajátkörfrekvenciája és

$$\omega_d = \omega_n\sqrt{1-\zeta^2} \tag{2}$$

Ábrázoljuk ezt a függvényt a $t \in [0, 2.5]$ intervallumon, ha a rendszer $A_0 \equiv 1$, $\omega_n = 10 \left[\frac{\mathrm{rad}}{\mathrm{s}}\right]$ és $\zeta = 0.3$ paraméterekkel rendelkezik! A vízszintes tengelyfelirat legyen `"idő - t [s]"`, a függőleges pedig `"Kimenet - y [-]"`! A diagram címe pedig legyen `"Kéttárolós rendszer"`!

1.3 Függvények - Másodfokú egyenlet

Készítsünk egy `masodfoku(a,b,c)` függvényt, amely megoldja a

$$ax^2 + bx + c = 0 \tag{3}$$

alakban felírt másodfokú egyenletet!

Segítségül a másodfokú egyenlet megoldóképlete:

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{4}$$

2 Gyakorló feladatok

Útmutató: a szövegben `ezen betűtípussal szedett szavak` olyan változókat jelölnek, melyeket mindig a felhasználó (azaz Te) definiál(sz)! Ezért javaslom, hogy mindig ezen változók megadásával kezdjétek a feladatokat!

2.1 Feszültségek

Ábrázoljuk az

$$u : [-2, 8] \subset \mathbb{R} \to \mathbb{R}, u(t) = \frac{A_0}{\pi}\left(\sin(3t) - \frac{\sin(6t)}{2} + \frac{\sin(9t)}{3} - \frac{\sin(12t)}{4}\right) \tag{5}$$

függvényt, $A_0 = 3, 4, 5$ értékekre! Készítsük el a tengelyfeliratokat: feszültséget ábrázolunk idő függvényében! Nevezzük el a grafikont és kapcsoljuk be a gridet!

Mire hasonlít ez és hogyan lehetne "folytatni"? Készítsük el úgy is a grafikont, hogy egy kiválasztott A_0 értékre egyszerre látszódik $2, 4, 6, 8$ és 10 tag esetén is a függvény!

2.2 Nemlineáris egyenletrendszer grafikus megoldása

Tekintsük a következő nemlineáris egyenletrendszert:

$$x^2 + y^2 - z = -1 \tag{6}$$

$$3x - y + 2z = 5 \tag{7}$$

Ábrázoljuk (egy megfelelő átalakítás után) ezt a két felületet egy kellően nagy intervallumon! Olvassuk le grafikusan, hogy milyen pontok alkotják az egyenletrendszer megoldását! Milyen görbére illeszkednek ezek?

2.3 Átlagok

Készítsünk egy `atlag()` függvényt, amely bemenetén két szám (`a` és `b`), melyeket a felhasználótól kérünk be. Ezt követően számoljuk ki ezen két szám számtani, mértani, harmonikus és kvadratikus átlagait! Hasonlítsuk össze ezeket az értékeket és próbáljuk meg igazolni az átlagokra vonatkozó egyenlőtlenséget!

2.4 Mátrix-számológép

Írjunk egy `matrixCaclulator()` függvényt, amely bemenetén vár egy `A` és `B`, kvadratikus és azonos méretű mátrixot és a kimenetén kiíratja a két mátrix összegét, különbségét, szorzatát és az egyes mátrixok determinánsát!

3 Ajánlott feladatok

Az alábbi feladatok alapvetően túlnyúlnak az órai anyagon, és megoldásukhoz a dokumentációk és az Internet böngészése szükséges! A Teams fileok közé feltöltöttem az alábbi segédleteket, melyeket mindenképpen érdemes áttekinteni:

- `cheatsheets.pdf`: Rövid összefoglaló a MatPlotLibről
- `handout-beginner.pdf`: Kezdő leírás a plotolásról. Ennek jelentős részét áttekintettük az órán.
- `handout-intermediate.pdf`: Haladó összefoglaló a plotolással kapcsolatban.
- `handout-tips.pdf`: Tippek és trükkök.

3.1 Normális eloszlás

Feladatunk, hogy elkészítsük a *Figure 1* ábrán látható plotot, melyen a normális eloszlás látható. A Gauss-eloszlás általánosan az alábbi alakban adható meg:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{8}$$

ahol:

- x egy adott valószínűségi változó
- $f(x)$ a valószínűségi változó *sűrűségfüggvény*e
- $\sigma \in \mathbb{R}^+$ a valószínűségi változó *szórás*a
- $\mu \in \mathbb{R}$ a valószínűségi változó *átlag*a, illetve *várható érték*e

Feladat:

1. Ábrázoljuk a (8) által definiált görbét a $[-3\sigma, 3\sigma] \in \mathbb{R}$ intervallumon, ha ismert, hogy az x valószínűségi változó szórása egységnyi és várható értéke zérus! (Ez az úgynevezett standard normális eloszlás.)
   (a) A plot színe legyen `navy`!
   (b) A görbe alatti területet vonalkázzuk be szintén `navy` színűre, $\alpha = 0.5$ áttetszőséggel.
   (c) A vízszintes és függőleges tengelyfeliratok legyenek: `"Random variable x [-]"` és `"Probability density f[-]"`
   (d) Az ábrának adjunk címet: `"Normal distribution"`
   (e) A felső és jobb oldali tengelyeket kapcsoljuk ki! Az alsó és bal oldali tengelyt pedig helyezzük át úgy, hogy az origóból induljanak!
2. Készítsünk σ, 2σ és 3σ távolságokban egyre sötétedő, szimmetrikus sávokat, az ábrának megfelelően!
   (a) Tetszőleges σ és μ esetén is a legbelső sáv tartalmazza a görbe alatti terület 68%-át, a középső a 95%-át és végül a harmadik már a görbe alatti terület 99.7%-át! A sávok bal felső sarkaiba írjuk fel ezeket az értékeket!
   (b) Az értékek könnyen beleolvadnak a háttérbe, ezért készítsünk egy fehér kontúrt is a szöveg köré!
3. Készítsünk egy függvényt, amely tetszőleges σ és μ bemenetre elkészíti ezt az ábrát és lementi `"normal_distribution.png"` néven, `DPI = 200` beállítással.

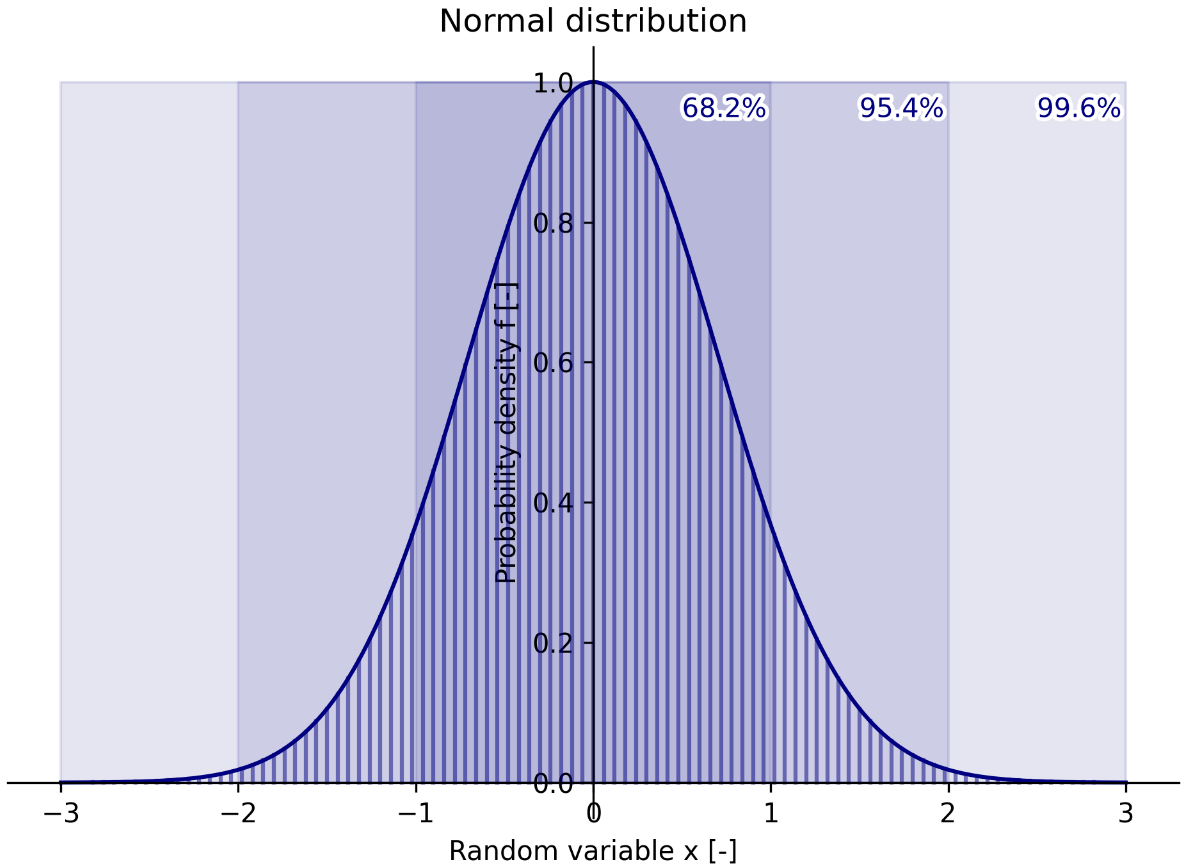

Figure 1: A normális eloszlás ábrája

3.2 Mandelbrot-halmaz

A *Mandelbrot-halmaz* egyike a legérdekesebb matematikai alakzatoknak, szerepe jelentős mind a komplex számok elméletében, mind pedig a fraktálgeometriák megértésében. Tekintsük az alábbi x_n sorozatot, ahol $c \in \mathbb{C}$:

$$x_1 := 0 \quad \rightarrow \quad x_{n+1} = (x_n)^2 + c \tag{9}$$

Különböző c értékekre különböző sorozatokat kapunk (próbáljuk is ki ezeket egy számológép segítségével!), így például:

- $c = 1$-re a sorozat: $0, 1, 2, 5, 26 \ldots$. Ebben az esetben azt látjuk (és teljes indukcióval könnyen be is bizonyíthatjuk), hogy a sorozat divergálni fog a végtelenbe!
- $c = -0.5 + 0.5i$-re a sorozat: $0, -0.5 + i0.5, -0.5 + i0, -0.25 + i0.5, -0.6875 + i0.25 \ldots$ Azt vehetjük észre, hogy tetszőlegesen sokadik tagra sem fog "elszállni" a sorozat és mindig egy megadott határon belül marad. (Azaz a sorozat korlátos.)

Tekintsük most összes olyan c komplex számot, melyre a sorozat nem a végtelenbe tart! Ezen számok összessége definiálja a Mandelbrot-halmazt:

$$M = \{c \in \mathbb{C} | x_n \nrightarrow \infty\} \tag{10}$$

A Mandelbrot-halmaz egy nagyon érdekes, úgynevezett fraktált alkot, melyek már régóta szolgál tudományos kutatások alapjául. Itt találhattok is egy linket, amelyben a végtelenül bonyolult, önhasonló struktúrája megjelenik a halmaznak: `https://www.youtube.com/watch?v=LhOSM6uCWxk`

Feladat:

1. Osszuk fel a $[-2, 1] \times [-1.5i, 1.5i] \in \mathbb{C}$ tartományt kellően sok részre!
2. A tartomány minden pontjában iteratívan[1] nézzük meg, hogy `maxIter` lépés végén hova tart a sorozat.
3. Ha az adott pont komplex abszolútértéke egy megadott `threshold` határ alatt van, akkor vegyük be ezt a pontot az `M` halmazba.
4. MatPlotLib segítségével ábrázoljuk ezt a halmazt!

\+ Vizsgáljuk meg, hogy különböző `maxIter` és `threshold` értékek esetén hogyan változik a Mandelbrot-halmaz!

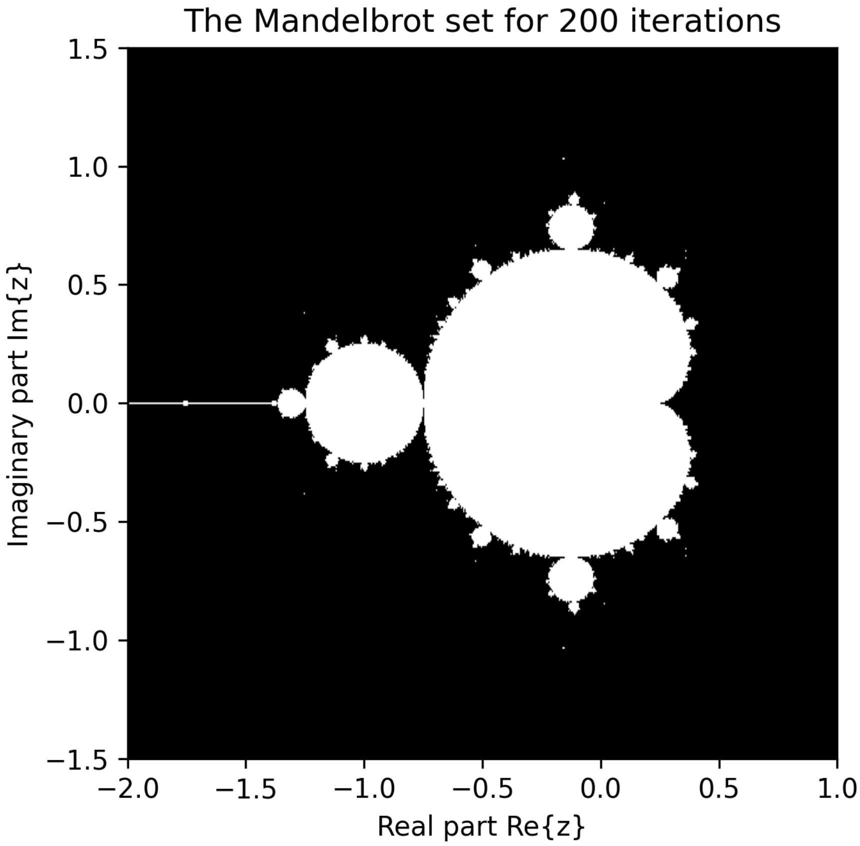

Figure 2: A Mandelbrot-halmaz ábrája

[1] Úgynevezett `for` ciklus segítségével

Megoldások

Az alábbiakban találhatóak megoldások a *bevezető* példákhoz. Kérlek szánjatok kellő időt és figyelmet a megoldások tanulmányozására, próbáljátok ki őket, futtassátok le, kísérletezzetek! Illetve hasonlítsátok össze a saját elgondolásaitokkal, gondoljátok végig melyik miért lehet célravezetőbb!

1.1 ujadat

```
import numpy as np      # Szükséges importok

# Definiáljuk a változóinkat
A = np.array([5, 12, 4.7, -3, 2.2])
print(f"Az A tömb legnagyobb eleme {np.max(A)}")

# Hozzáfűzés: input() castolása int-re!
A = np.append(A, int(input("Adj meg egy számot!")))
print(f"Hozzáfűzés után az A tömb legnagyobb eleme {np.max(A)}")
```

1.2 Kéttárolós rendszer

```
# Szükséges importok
mport numpy as np
import matplotlib.pyplot as plt

# A rendszer paraméterei
A_0 = 1          # [-]
omega_n = 10     # [rad/s]
zeta = 0.3       # [-]
omega_d = np.sqrt(1-zeta**2)*omega_n    # [rad/s]

# Idő felosztása
t = np.linspace(0, 2.5, 1000)

# Függvényértékek kiszámítása
y = A_0 * (1-np.e ** (-zeta*omega_n*t)*(np.cos(omega_d*t)+
            zeta/np.sqrt(1-zeta**2)*np.sin(omega_d*t)))

# Plotolás
fig, ax = plt.subplots()
ax.plot(t, y, color = 'b')
ax.set_xlabel("Idő - t [s]")
ax.set_ylabel("Kimenet - y(t) [-]")
ax.set_title("Kéttárolós rendszer")

plt.show()
```

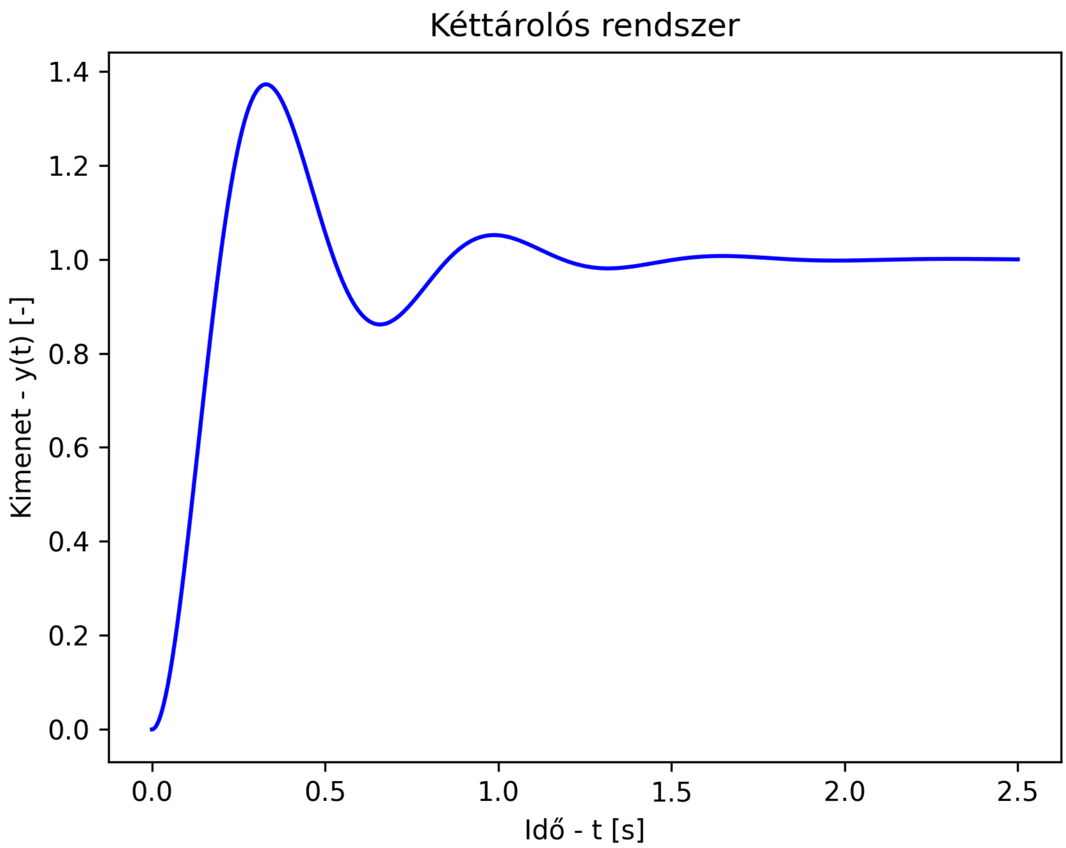

Figure 3: Kéttárolós rendszer

1.3 Másodfokú egyenlet

```
import numpy as np       # Szükséges import

# Definiáljuk a függvényt: INPUT - a,b,c
def masodfoku(a,b,c):
    x1 = (-b + np.sqrt(b**2-4*a*c))/2*a
    x2 = (-b - np.sqrt(b**2-4*a*c))/2*a
    return x1, x2        # OUTPUT - x1, x2

# Tesztelés: x^2-6x+5 = (x-5)(x-1)
print(masodfoku(1,-6, 5))
```