Podstawy Sterowania Optymalnego - Labolatorium 1

Wprowadzenie do języka Python

Prowadzący: mgr inż. Krzysztof Hałas
Wykonał: Ryszard Napierała

23 Październik 2021

1 Zadanie 2

1. **Wykorzystując Pythona oraz środowisko Spyder wyznaczyć wartość x.**

   (a) $x = 3^{12} - 5$

```
x = 3**12-5
print(x)
```

Output: 531436

   (b) $x = \begin{bmatrix} 2 & 5 \end{bmatrix} \begin{bmatrix} 1 & 4 \\ -1 & 3 \end{bmatrix} \begin{bmatrix} -1 \\ -3 \end{bmatrix}$

```
import numpy as np

x = np.array([[2, 0.5]])@\
    np.array([[1, 4],[-1, 3]])@\
    np.array([[-1],[-3]])
print(x)
```

Output: $[[-30.]]$

   (c) $x = rank(\begin{bmatrix} 1 & -2 & 0 \\ -2 & 4 & 0 \\ 2 & -1 & 7 \end{bmatrix})$

```
import numpy as np

x = np.array([
    [1, -2, 0],
```

```
    [-2, 4, 0],
    [2, -1, 7]])

x = np.linalg.matrix_rank(x)
print(x)
```

Output: 2

(d) $\begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix} x$

```
import numpy as np

b = np.array([[-1], [2]])
a = np.array([[1, 2], [-1, 0]])
x = np.linalg.solve(a, b)
print(x)
```

Output:
[[−2.]
[0.5]]

2. **Dana jest tablica [1, 1, -129, 1620]. Napisać skrypt tworzący zmienną przechowującą tę tablicę. Przyjmując zawartość tablicy jako współczynniki wielomianu (zaczynając od najwyższej potęgi) wyznaczyć wartość tego wielomianu w punktach $x_1 = -46, x_2 = 14$.**

```
import numpy as np

a = np.array([1, 1, -129, 171, 1620])
p = np.poly1d(a)
print('dla x=-46, y={0}'.format(p(-46)))
print('dla x=14, y={0}'.format(p(14)))
```

Output:
dla $x = -46$, $y = 4100910$
dla $x = 14$, $y = 19890$

2 Zadanie 3

1. **Bazując na poprzednim zadaniu rozbudować skrypt tak, by w sposób numeryczny wyznaczył największą i najmniejszą wartość wielomianu w przedziale $\begin{bmatrix} -46 & 14 \end{bmatrix}$.**

```python
import numpy as np

a = np.array([1, 1, -129, 171, 1620])
p = np.poly1d(a)
space = np.arange(-46, 15)
minmax = np.array([float('inf'), float('-inf')])
for x in space:
    y = p(x)
    minmax[0] = min(minmax[0], y)
    minmax[1] = max(minmax[1], y)
print(minmax)
```

Output:
[−4536. 4100910.]

2. **Uzależnić dokładność wyznaczania ekstremów od wartości dodatkowej zmiennej.**

```python
import numpy as np

precision = 0.01

a = np.array([1, 1, -129, 171, 1620])
p = np.poly1d(a)
space = np.arange(-46, 14+precision, step=precision)
minmax = np.array([float('inf'), float('-inf')])
for x in space:
    y = p(x)
    minmax[0] = min(minmax[0], y)
    minmax[1] = max(minmax[1], y)
print(minmax)
```

Output:
[−4561.29354219 4100910.]

3 Zadanie 4

1. **Zmodyfikować skrypt z poprzedniego zadania tak, by funkcjonalność wyznaczająca maksimmum i minimum wielomianu zawarta była w funkcji przyjmującej współczynniki wielomianu 4 stopnia, granice wyznaczania oraz wskaźnik dokładności jako argumenty. Wyznaczone wartości minimum i maksimum powinny być zwracane jako tablica dwuelementowa.**

```python
import numpy as np

def zad4_1(a, b, c, d, e, bounds, precision=1):
    p = np.poly1d(np.array([a, b, c, d, e]))
    space = np.arange(bounds[0], bounds[1]+precision, step=precision)
    minmax = np.array([float('inf'), float('-inf')])
    for x in space:
        y = p(x)
        minmax[0] = min(minmax[0], y)
        minmax[1] = max(minmax[1], y)
    return minmax

minmax = zad4_1(1, 1, -129, 171, 1620, (-46, 14), precision=0.01)
print(minmax)
```

Output:
[−4561.29354219 4100910.]

2. **Rozbudować skrypt, by funkcja przyjmowała jako argument tablicę współczynników wielomianu dowolnego stopnia.**

```python
import numpy as np

def zad4_2(*coefficients, bounds, precision=1):
    p = np.poly1d(np.array(coefficients))
    space = np.arange(
        bounds[0],
        bounds[1]+precision,
        step=precision
    )
    minmax = np.array([float('inf'), float('-inf')])
    for x in space:
        y = p(x)
        minmax[0] = min(minmax[0], y)
        minmax[1] = max(minmax[1], y)
    return minmax

minmax = zad4_2(1, 1, -129, 171, 1620, bounds=(-46, 14), precision=0.01)
print(minmax)
```

Output:
[−4561.29354219 4100910.]

4 Zadanie 5

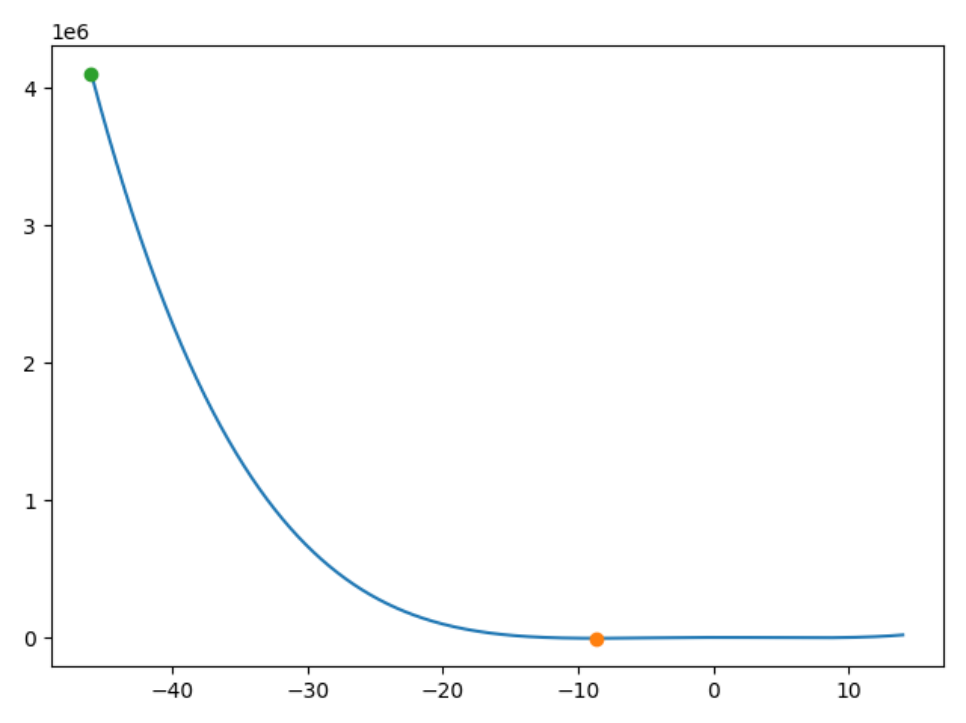
1. **Zmodyfikować skrypt z poprzedniego zadania tak, funkcja wyznaczająca ekstrema wielomianu wykreślała przebieg wielomianu w całym wymaganym zakresie.**

```python
import numpy as np
import matplotlib.pyplot as plt

def zad5_1(*coefficients, bounds, precision=1):
    p = np.poly1d(np.array(coefficients))
    space = np.arange(bounds[0], bounds[1]+precision, precision)
    y_min = float('inf')
    x_min = float('inf')
    y_max = float('-inf')
    x_max = float('-inf')
    y = np.array([])
    for x in space:
        temp = p(x)
        y = np.append(y, [temp])
        if temp < y_min:
            y_min = temp
            x_min = x
        if temp > y_max:
            y_max = temp
            x_max = x
    plt.plot(space, y)
    plt.plot([x_min], [y_min], 'o')
    plt.plot([x_max], [y_max], 'o')
    plt.tight_layout()
    plt.show()

zad5_1(1, 1, -129, 171, 1620, bounds=(-46, 14), precision=0.01)
```

Output:



2. **Dodać do generowanego wykresu legendę, opisy osi, tytuł, itp.**

```
import numpy as np
import matplotlib.pyplot as plt

def zad5_2(*coefficients, bounds, precision=1):
    p = np.poly1d(np.array(coefficients))
    space = np.arange(bounds[0], bounds[1]+precision, precision)
    y_min = float('inf')
    x_min = float('inf')
    y_max = float('-inf')
    x_max = float('-inf')
    y = np.array([])
    for x in space:
        temp = p(x)
        y = np.append(y, [temp])
        if temp < y_min:
            y_min = temp
            x_min = x
        if temp > y_max:
            y_max = temp
            x_max = x
    plt.plot(space, y)
    plt.plot([x_min], [y_min], 'o')
    plt.plot([x_max], [y_max], 'o')
    plt.annotate(
        f'x={x_min:.2f}\ny={y_min:.2f}',
        xy=(x_min, y_min),
```

```
        xytext=(x_min+2, y_min+100000)
    )
    plt.annotate(
        f'x={x_max:.2f}\ny={y_max:.2f}',
        xy=(x_max, y_max),
        xytext=(x_max+2, y_max-250000)
    )
    plt.legend(['polynomial', 'minimum', 'maximum'])
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title(''.join(
        ['{0}{1}x'.format('+' if c >= 0 else '',c)\
            for c in coefficients])
    )
    plt.tight_layout()
    plt.show()

zad5_2(1, 1, -129, 171, 1620, bounds=(-46, 14), precision=0.01)
```

Output: