PYTHON

ZVEZA ZA TEHNIČNO KULTURO SLOVENIJE

# Funkcije odvodi in ekstremi

Python urejevalnik se nahaja na strani:
`https://www.w3resource.com/python-exercises/python-basic-exercises.php#EDITOR`

## Funkcija podana v priponski obliki

Iz standardnega vhoda preberite števili $a$ in $b$ in definicijo funkcije zapisano v priponski notaciji. Spremenljivko označimo z ('x'). Nato načrtajte njen graf na intervalu $[a, b]$. Uporabljamo operacije `'+'`, `'-'`, `'*'`, `'/'`, `'**'`.

## Dualna števila

- Dualna števila se zapišejo kot vsota $a + b\varepsilon$, kjer sta $a$ in $b$ realni števili, medtem ko je $\varepsilon$ dualna enota, ki zadošča pogoju $\varepsilon^2 = 0$ in $\varepsilon \neq 0$.
- Dualna števila seštevamo po komponentah in jih množimo kot binome

$$(a + b\varepsilon)(c + d\varepsilon) = a\,c + (a\,d + b\,c)\varepsilon,$$

  kjer upoštevamo $\varepsilon^2 = 0$.
- Dualna števila lahko uporabimo za izračun odvoda analitične funkcije v dani točki.
- Vzemimo najpreprostejši primer, to je polinom

$$P(x) = p_0 + p_1\,x + p_2\,x^2 + \cdots + p_n\,x^n.$$

- Vrednosti polinoma na dualnih številih dobimo tako, da upoštevamo

$$(a + b\varepsilon)^n = a^n + n\,a^{n-1}\varepsilon + \ldots.$$

  Različna od nič sta le prva dva člena, velja:

$$P(x + y\varepsilon) = P(x) + y\,P'(x)\varepsilon.$$

- Vsako realno (analitično) funkcijo lahko razširimo na dualna števila. S pomočjo Taylorjeve vrste izpeljemo ustrezno formulo:

$$f(x + y\epsilon) = f(x) + y\,f'(x)\varepsilon, \quad \rightarrow \quad \varepsilon\,f'(x) = f(x + \varepsilon) - f(x).$$

### Realizacija s pomočjo kompleksnih števil

Postopek za iskanje vrednosti odvoda analitične funkcije s pomočjo kompleksnih števil v dani točki je sledeč:

1. Kompleksno število se zapiše kot vsota $a + b\mathrm{i}$, kjer sta $a$ in $b$ realni števili in $\mathrm{i}$ je imaginarna enota ($\mathrm{i}^2 = -1$).
2. Dualno enoto predstavimo z imaginarnim številom, kjer pomnožimo imaginarno enoto $\mathrm{i}$ z majhnim številom $\epsilon$, $\varepsilon = \mathrm{i}\epsilon$, če vzamemo, de je $\epsilon = 10^{-40}$ je njegov kvadrat $-\epsilon^2 \approx 0$.
3. V funkciji $f(x)$ nadomestite realno število $x$ s kompleksnim številom $z = x + \mathrm{i}\epsilon$.
4. Izračunamo vrednost $f(z)$.
5. Prva dva člena Taylorjeve vrste sta $f(x + \mathrm{i}\epsilon) = f(x) + f'(x)\,\mathrm{i}\epsilon + \ldots$, medtem ko ostali členi vsebujejo $\epsilon^k$, kjer je $k > 1$. Ti presegajo območje aritmetike s končno 64 bitno mantiso in se v končnem seštevku ne poznajo.
6. Na koncu delimo imaginarno komponento rezultata z $\epsilon$ in dobimo vrednost $f'(x)$.

Več o dualnih številih na naslovu:
`https://www.youtube.com/watch?v=ceaNqdHdqtg`

## Ekstremi in metoda zlatega reza

- Razmerje dolžin daljic $a$ in $b$ je v zlatem rezu, če velja:

$$\frac{a}{b} = \frac{b}{a-b}, \quad \text{če je} \quad (a = 1, b = x), \quad x = \frac{1}{x} - 1, \quad x^2 + x - 1 = 0.$$

- Pozitivna rešitev enačbe je:

$$\phi = \frac{\sqrt{5}-1}{2}, \quad \phi^2 = 1 - \phi, \quad \frac{1}{\phi} = 1 + \phi, \quad \Phi = \frac{1}{\phi} = \frac{1+\sqrt{5}}{2}.$$

- Metoda zlatega reza:
  1. Poiščimo najmanjšo vrednost funkcije $f(x)$ na intervalu $[a, b]$, po predpostavki, da je funkcija zvezna, in da ima na tem intervalu natanko en lokalni minimum.
  2. Funkcijski vrednosti v krajiščih sta $y_a = f(a)$ in $y_b = f(b)$.
  3. Določite znotraj intervala dve točki $x_1$ in $x_2$, $x_1 < x_2$ tako, da velja:

$$x_1 = a + (b-a)\phi^2, \quad x_2 = a + (b-a)\phi.$$

  4. Izračunajte funkcijski vrednosti v novih točkah $y_1 = f(x_1)$ in $y_2 = f(x_2)$.
  5. Če je $y_1 < y_2$, potem postne $b = x_2$, $x_2 = x_1$, $y_2 = y_1$
     in izračunamo novo vrednost za $x_1$ in $f(x_1)$:

$$x_1 = a + (b-a)\phi^2, \quad y_1 = f(x_1).$$

  6. Sicer pa postane $a = x_1$, $x_1 = x_2$ in poiščemo novo točko $x_2$ in funkcijsko vrednost $f(x_2)$:

$$x_2 = a + (b-a)\phi, \quad y_2 = f(x_2).$$

  7. Nadaljujemo postopek s primerjanjem funkcijskih vrednosti $y_1$ in $y_2$.
  8. Ponavljamo, dokler, ni razlika $b - a$ dovolj majhna.

- Prednost te metode je, da razen v prvem koraku, izračunamo le eno novo funkcijsko vrednost.

**Funkcija**

```
#!/usr/bin/env python3
import matplotlib.pyplot as plt
import numpy as np

def postfix(func, x):
    oper = {'+': lambda x, y: x + y, '-': lambda x, y: y - x,
            '*': lambda x, y: x * y, '/': lambda x, y: y / x,
            '^': lambda x, y: y ** x}
    var  = {'x': x}
    stack = []
    tokens = func.split()
    for token in tokens:
        if token in oper:
            stack.append(oper[token](stack.pop(),stack.pop()))
        elif token in var:
            stack.append(var[token])
        else:
            stack.append(float(token))
    res = stack.pop()
    return res

if __name__ == '__main__':
    a = -2
    b = 1
    X = np.linspace(a, b)
    Y = [postfix('x 3 ^ 2 x * - 1 +', x) for x in X]
    plt.plot(X,Y)
    plt.show()
```

## Dualna števila

```python
#!/usr/bin/env python3
import numpy as np
import matplotlib.pyplot as plt

epsilon = '\u03B5'
eps = 1e-40

def _dual_(real, dual=1):
    return real + 1j*eps*dual

def real_(d):
    return np.real(d)

def dual_(d):
    return np.imag(d)/eps


if __name__ == '__main__':

def f(x):
    return x**3 + 2*x**2 + 1

x = np.linspace(-3, 2)
z = np.array([_dual_(u) for u in x])
y = f(z)

plt.plot(x, real_(y), x, dual_(y))
plt.show()
```

## Fibonaccijevo zaporedje in zlati rez

```python
#!/usr/bin/env python3
import math, time

def fibonacci_search(f, a, b, eps=1e-8):
    c = (a + b)/2
    minimum = f((a + b) / 2) < (f(a) + f(b)) / 2
    # Define the Fibonacci sequence F(k)
    F = [1, 1]
    k = 1
    while F[k] < (b-a) / eps:
        k += 1
        F.append(F[k-1] + F[k-2])

    # Initialize the variables x1, x2, y1, y2
    x1 = a + (b-a) * F[k-2] / F[k]
    x2 = a + (b-a) * F[k-1] / F[k]
    y1 = f(x1)
    y2 = f(x2)

    # Repeat the search until the interval [a, b] is smaller than eps
    while b - a > eps:
        if minimum:
            test = y1 < y2
        else:
            test = y1 > y2
        if test:
            b = x2
            x2 = x1
            y2 = y1
            x1 = a + (b-a) * F[k-3] / F[k-1]
            y1 = f(x1)
        else:
            a = x1
            x1 = x2
            y1 = y2
            x2 = a + (b-a) * F[k-2] / F[k-1]
            y2 = f(x2)
        k -= 1

    # Return the midpoint of the final interval as the maximum value
    return (a + b) / 2
```

```python
def golden_search(f, a, b, eps=1e-8):
    c = (a + b)/2
    minimum = f((a + b) / 2) < (f(a) + f(b)) / 2

    fi = (math.sqrt(5)-1)/2
    f1 = fi
    f2 = fi**2
    x1 = a + (b-a) * f2
    x2 = a + (b-a) * f1
    y1 = f(x1)
    y2 = f(x2)

    # Repeat the search until the interval [a, b] is smaller than eps
    while b - a > eps:
        if minimum:
            test = y1 < y2
        else:
            test = y1 > y2
        if test:
            b = x2
            x2 = x1
            y2 = y1
            x1 = a + (b - a) * f2
            y1 = f(x1)
        else:
            a = x1
            x1 = x2
            y1 = y2
            x2 = a + (b - a) * f1
            y2 = f(x2)

    # Return the midpoint of the final interval as the maximum value
    return (a + b) / 2
```

### Main

```
if __name__ == '__main__':
from fibs import *
import matplotlib.pyplot as plt
import numpy as np

if __name__ == '__main__':
    def f(x):
        return x**3 -3*x**2 +4

    a = -2
    b = 4
    x = np.linspace(a, b)
    y = f(x)

    a = 0
    xm = golden_search(f, a, b)

    a = -2
    b = 0
    xM = golden_search(f, a, b)

    plt.scatter([xm, xM],[f(xm), f(xM)], c = 'red')
    plt.plot(x, y)
    plt.show()
```

https://trinket.io/python3/889dfea984