Poročilo o reševanju prve naloge

Gašper Harej 28241126

3.3.2025

1 Uvod

V tem poročilu predstavljam analizo prve naloge. V nalogi obravnavamo različne metode aproksimacij funkcije ter njihove konvergenčne lastnosti.

2 Primerjava aproksimacij funkcije erf

Analiziramo različne metode aproksimacije funkcije napake (erf).

2.1 Potenčna vrsta

Prva aproksimacija je potenčna vrsta, kjer sem uporabil naslednjo enačbo:

$$\operatorname{erf}(z) = \frac{2}{\sqrt{\pi}} \sum_{n=0}^{\infty} (-1)^n \frac{z^{2n+1}}{n!\,(2n+1)}$$

Za implementacijo sem spisal naslednji program:

```
def potencna(z, niter, epsilon):
value = 0.
if z==0:
    return value
star = 0.
for n in range(0, niter+1):
    star = value
    val = (-1)**n * z**(2*n + 1) / (math.factorial(n) *
(2*n + 1))
    value += val * 2 / math.sqrt(math.pi)

    if abs(value - star) < epsilon * star or value ==
star or n==niter:
        break
return value
```

2.2 Asimptotska vrsta

Drugi uporabljen pribljižek je asimptotska vrsta:

$$z\sqrt{\pi}\,\exp(z^2)\,\mathrm{erfc}(z) \longrightarrow 1 + \sum_{m=1}^{\infty} \frac{(2m-1)!!}{(-2z^2)^m};$$

```
    def asimptotska(z, niter, epsilon):
    if z==0:
        return 0.0
    value = 1 - 1 / (z * math.sqrt(math.pi) * math.exp(z**2))
    star = 0.0
    erfc = 1 / (z * math.sqrt(math.pi) * math.exp(z**2))
    old_erfc = 0.0

    for m in range(1, niter + 1):
        star = value
        val = factorial2(2 * m - 1, exact=True) / (-2 * z**2) ** m
        old_erfc = erfc
        erfc = val / (2 * z * math.sqrt(math.pi) * math.exp(z**2))

        if abs(erfc) > abs(old_erfc):
            break

        value += erfc

        if abs(value - star) < epsilon * abs(star) or value == star or m == niter:
            break
    return value
```

2.3 Racionalna aproskimacija

Nazadnje sem za $z > 0$ uporabil še racionalno aproksimacijo:

$$\mathrm{erf}(z) = 1 - (a\,t + b\,t^2 + c\,t^3 + d\,t^4 + e\,t^5)\,\exp(-z^2) + \varepsilon(z),$$

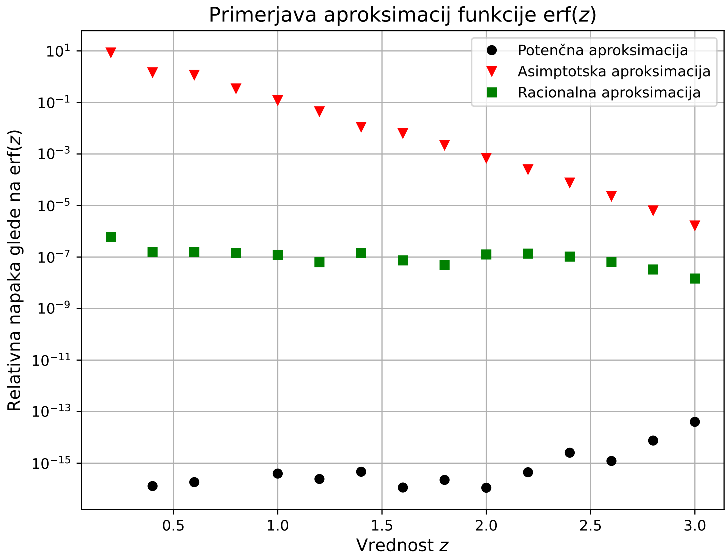
$$t = \frac{1}{1+pz}, \qquad \varepsilon(z) < 1.5 \cdot 10^{-7},$$

Koda za implementacijo:

```
    p = 0.3275911
a = 0.254829592
b = -0.284496736
```

```python
c = 1.421413741
d = -1.453152027
e = 1.061405429

def racaprox(z):
    if z==0:
        return 0
    t = 1 / (1 + p * z)
    return 1 - (a*t + b*t**2 + c*t**3 + d*t**4 + e*t**5) * math.exp(-z**2)
```

2.4 Primerjava



Slika 1: Primerjava aproksimacij funkcije erf(z).

Vidimo, da ima racionalna aproksimacija najmanjšo napako pri vseh vrednostih z, medtem ko potenčna hitro izgubi natančnost. Asimptotska metoda je uporabna le pri večjih vrednostih z.

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


tabela1 = [[potencna(i / 10, 100, 1e-16), asimptotska(i / 10, 100, 1e-16), racaprox(i / 10), math.erf(i / 10)]
```

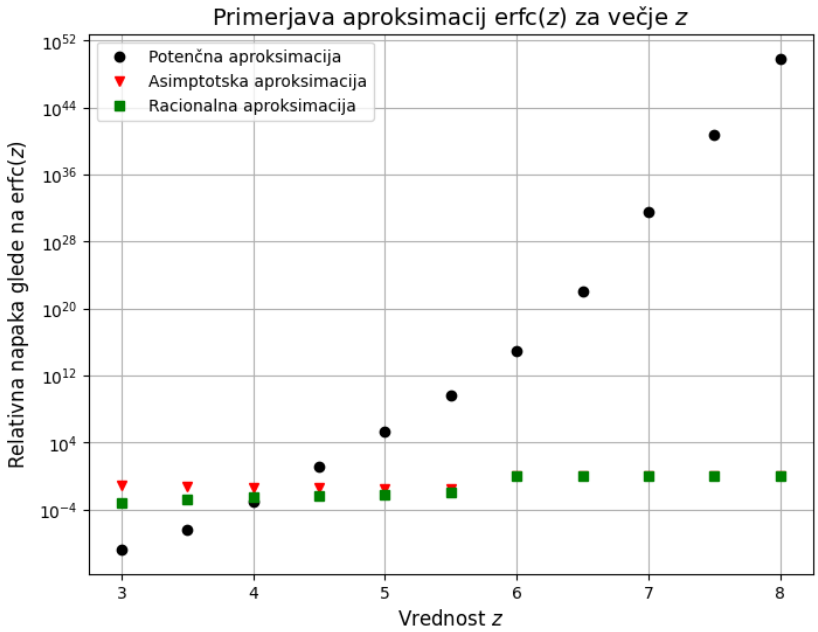
```
            for i in range(0, 32, 2)]

napake1 = [[abs(tabela1[i][j] - tabela1[i][3]) / tabela1[i][3] for j in range(3)] for i in range(1, len(tabela1))]

napake1_pot, napake1_asi, napake1_rac = zip(*napake1) if napake1 else ([], [], [])


plt.figure(figsize=(8, 6))
plt.plot(np.arange(0.2, 3.2, 0.2), np.array(napake1_pot), 'ko', label="Potencna aproksimacija")
plt.plot(np.arange(0.2, 3.2, 0.2), np.array(napake1_asi), 'rv', label="Asimptotska aproksimacija")
plt.plot(np.arange(0.2, 3.2, 0.2), np.array(napake1_rac), 'gs', label="Racionalna aproksimacija")
plt.yscale('log')
plt.xlabel('Vrednost $z$', fontsize=12)
plt.ylabel('Relativna napaka glede na $\\operatorname{erf}(z)$', fontsize=12)
plt.title('Primerjava aproksimacij funkcije $\\operatorname{erf}(z)$', fontsize=14)
plt.legend()
plt.grid(True)
plt,show()
```

Listing 1: Koda za izris grafa

3 Primerjava aproksimacij funkcije erfc

Podobno kot pri erf ocenimo metode aproksimacije komplementarne napake (erfc).

$$\operatorname{erfc}(z) = 1 - \operatorname{erf}(z)$$



Slika 2: Primerjava aproksimacij funkcije erfc(z).

Rezultati kažejo, da racionalna metoda ponuja najboljšo natančnost skozi celotno območje, medtem ko je asimptotska koristna pri večjih vrednostih z.

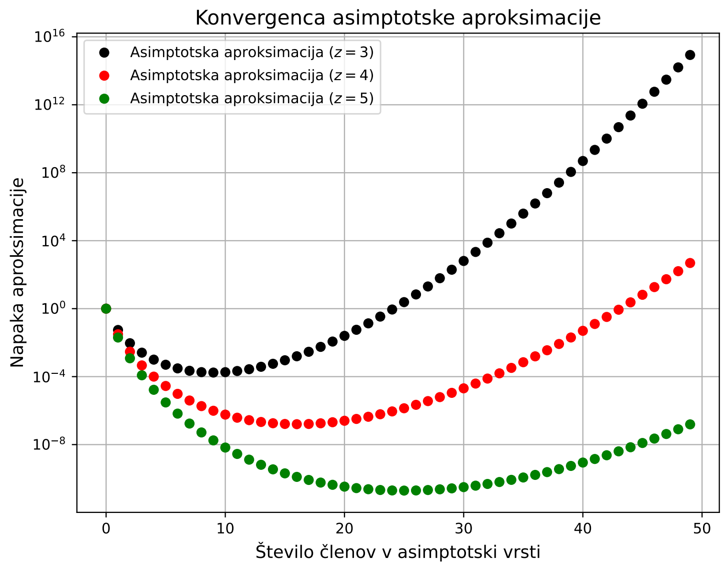
```
import numpy as np
import matplotlib.pyplot as plt
import math

tabela2 = [[potencna_erfc(i / 10, 100, 1e-16),
    asimptotska_erfc(i / 10, 100, 1e-16), racaprox_erfc(i /
    10), math.erfc(i / 10)]
            for i in range(30, 85, 5)]

napake2 = [[abs(tabela2[i][j] - tabela2[i][3]) / tabela2[i
    ][3] for j in range(3)] for i in range(len(tabela2))]
napake2_pot, napake2_asi, napake2_rac = zip(*napake2) if
    napake2 else ([], [], [])

plt.figure(figsize=(8, 6))
plt.plot(np.arange(3, 8.5, 0.5), np.array(napake2_pot), 'ko',
     label="Potencna aproksimacija")
plt.plot(np.arange(3, 8.5, 0.5), np.array(napake2_asi), 'rv',
     label="Asimptotska aproksimacija")
plt.plot(np.arange(3, 8.5, 0.5), np.array(napake2_rac), 'gs',
     label="Racionalna aproksimacija")
plt.yscale('log')
plt.xlabel('Vrednost $z$', fontsize=12)
```

```
plt.ylabel('Relativna napaka glede na $\\operatorname{erfc}(z
    )$', fontsize=12)
plt.title('Primerjava aproksimacij $\\operatorname{erfc}(z)$
    za vecje $z$', fontsize=14)
plt.legend()
plt.grid(True)
plt.show()
```

Listing 2: Koda za izris grafa

4 Konvergenca asimptotske aproksimacije

V nadaljevanju analiziramo hitrost konvergence asimptotske metode aproksimacije.



Slika 3: Konvergenca asimptotske aproksimacije.

Na grafu (Slika 3) vidimo, kako hitro se zmanjšuje napaka aproksimacije pri naraščanju števila členov v asimptotski vrsti. Kot pričakovano, večje število členov vodi do manjše napake, vendar postane izboljšanje manj izrazito pri pribljižno več kot 20 členih.

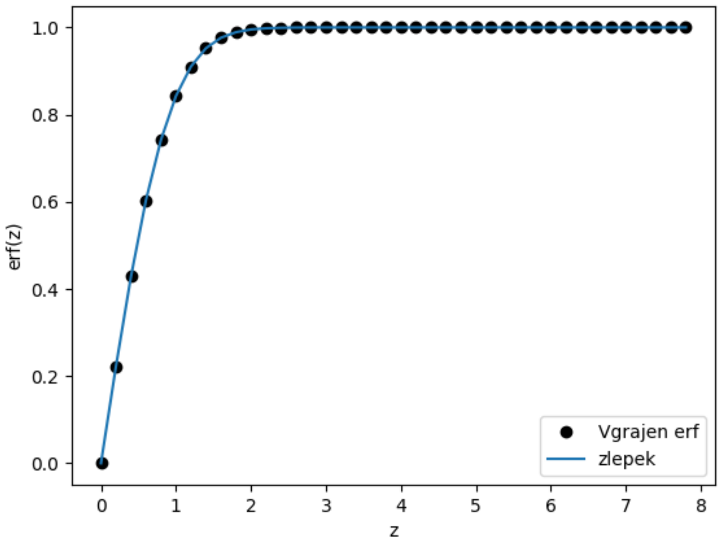
```
def askonv(z):
    nf = 1
    x = 0
    tabela3 = []
```

```
    for i in range(0, 50):
        nf = nf * (2 * i - 1)
        x = nf / ((-2 * z**2) ** i)
        tabela3.append(abs(x))
    return tabela3


plt.figure(figsize=(8, 6))
plt.plot(np.array(askonv(3)), 'ko', label="Asimptotska
    aproksimacija ($z=3$)")
plt.plot(np.array(askonv(4)), 'ro', label="Asimptotska
    aproksimacija ($z=4$)")
plt.plot(np.array(askonv(5)), 'go', label="Asimptotska
    aproksimacija ($z=5$)")
plt.yscale('log')
plt.xlabel('Stevilo clenov v asimptotski vrsti', fontsize=12)
plt.ylabel('Napaka aproksimacije', fontsize=12)
plt.title('Konvergenca asimptotske aproksimacije', fontsize
    =14)
plt.legend()
plt.grid(True)
plt.show()
```

Listing 3: Koda za izris grafa

5 Optimalna kombinacija aproksimacij

Za izboljšanje natančnosti čez celotno območje združimo različne metode.



Slika 4: Kombinirana aproksimacija funkcije na celotnem območju.

Ta graf prikazuje najboljšo kombinacijo aproksimacij, kjer uporabimo potenčno metodo za majhne vrednosti z, racionalno metodo za srednje vrednosti in asimptotsko metodo za velike vrednosti z. S tem dosežemo optimalno natančnost na celotnem intervalu.

```
def er(z, niter=100, epsilon=1e-5):
    if z < 4:
        return potencna(z, niter, epsilon)
    elif z >= 4 and z < 5.5:
        return racaprox(z)
    else:
        return asimptotska(z, niter, epsilon)

err = []
xerr = []
for i in range(0, 80, 2):
    xerr.append(i / 10)
    err.append(er(i / 10, 100, 1e-16))

erp = []
xerp = []
```

```
for i in range(0, 80, 2):
    xerp.append(i / 10)
    erp.append(math.erf(i / 10))

plt.plot(np.array(xerp), np.array(erp), 'ko', np.array(xerr),
     np.array(err))
plt.legend(('Vgrajen erf', 'zlepek'), loc='lower right')
plt.xlabel('z')
plt.ylabel('erf(z)')
plt.show()
```

Listing 4: Koda za izris grafa

6 Analiza konvergence Leibnizove vrste in Ramanujanove formule

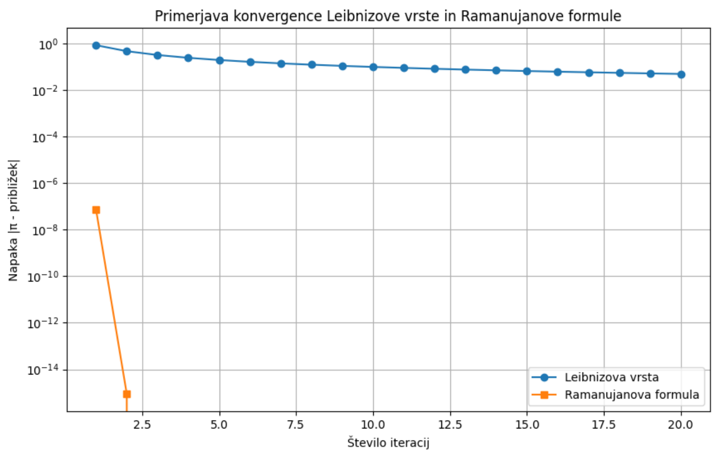
Leibnizova vrsta

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots$$

in Ramanujanova formula

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!\,(1103 + 26390k)}{(k!)^4\, 396^{4k}},$$

sta dve metodi za računanje vrednosti π. V nadaljevanju primerjamo njuno hitrost konvergence ter ocenjujemo njihovo praktično uporabnost.



Slika 5: Primerjava konvergence Leibnizove vrste in Ramanujanove formule.

Na grafu (Slika 5) je prikazana napaka aproksimacije π v odvisnosti od števila iteracij. Leibnizova vrsta konvergira zelo počasi, medtem ko Ramanujanova formula že po nekaj iteracijah doseže visoko natančnost.

```python
import numpy as np
import matplotlib.pyplot as plt
from math import factorial, pi
from decimal import Decimal, getcontext


getcontext().prec = 50
niter = 20


leibniz_values = []
leibniz_pi = 0
for n in range(1, niter+1):
    leibniz_pi += (1/(2*n-1)) * (-1)**(n+1) * 4
    leibniz_values.append(abs(leibniz_pi - pi))


ramanujan_values = []
ramanujan_sum = Decimal(0)
for k in range(0, niter+1):
    num = Decimal(factorial(4 * k)) * (1103 + 26390 * k)
    den = (Decimal(factorial(k)) ** 4) * (Decimal(396) ** (4 * k))
    ramanujan_sum += num / den
    pi_inverse = (Decimal(8).sqrt() / Decimal(9801)) * ramanujan_sum
    ramanujan_pi = 1 / pi_inverse
    ramanujan_values.append(abs(float(ramanujan_pi) - pi))


plt.figure(figsize=(10, 6))
plt.yscale("log")
plt.plot(range(1, niter+1), leibniz_values, marker="o", linestyle="-", label="Leibnizova vrsta")
plt.plot(range(1, niter+1), ramanujan_values[:niter], marker="s", linestyle="-", label="Ramanujanova formula")
plt.xlabel("Stevilo iteracij")
plt.ylabel("Napaka |\pi - priblizek|")
plt.title("Primerjava konvergence Leibnizove vrste in Ramanujanove formule")
plt.legend()
plt.grid(True)
plt.show()
```

Listing 5: Koda za izris grafa

7 Zaključek

Iz rezultatov lahko zaključimo:

- Ramanujanova formula konvergira hitreje kot Leibnizova vrsta.
- Asimptotska aproksimacija postane natančna pri velikih vrednostih, vendar ni uporabna za majhne vrednosti z.
- Racionalna metoda je splošno najboljša za aproksimacijo erf in erfc.
- Zlepek aproksimacij omogoča optimalno natančnost na celotnem intervalu.

Rezultati so uporabni pri numeričnih algoritmih, kjer je izbira metode ključnega pomena za natančnost in hitrost izračuna.