Laimonas Beniušis studento nr. 1410102
Kompiuterių mokslas 1 1gr

Optimizavimo metodai užduotis 1

Algoritmų palyginimas:

Algoritmas	Iteracijos	Funkcijos iškvietimai	Funkcijos iškvietimai kiekis	Vidurinio taško atstumas nuo minimumo	Intervalo dydis
Intervalo Dalinimas Pusiau	17	Iteracijos * 2 + 1	35	0.0000381470	0.0000762939
Auksinio Pjūvio	24	Iteracijos + 2	25	0.0000482244	0.0000964488

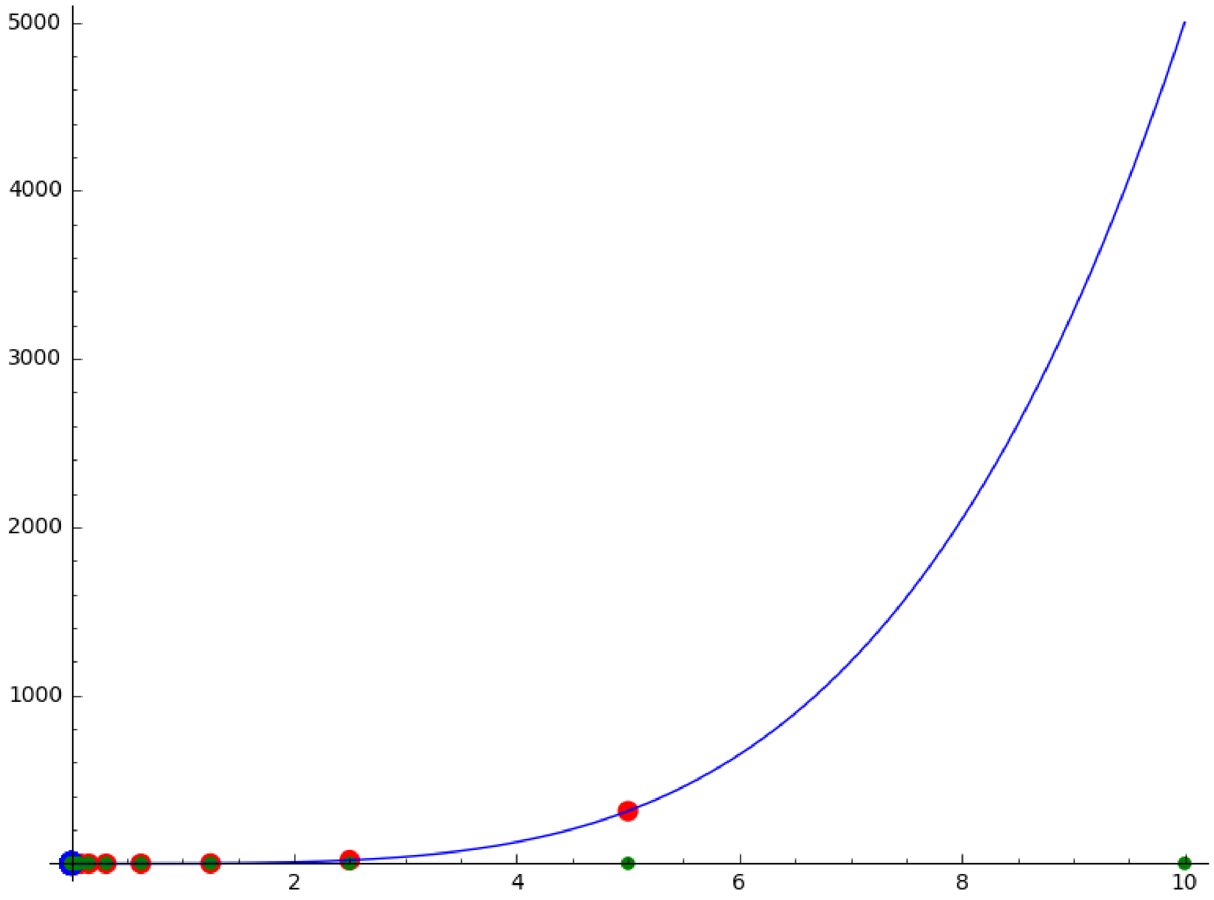
tikslo funkcija

$$f=\frac{(x^2-a)^2}{b-1}$$

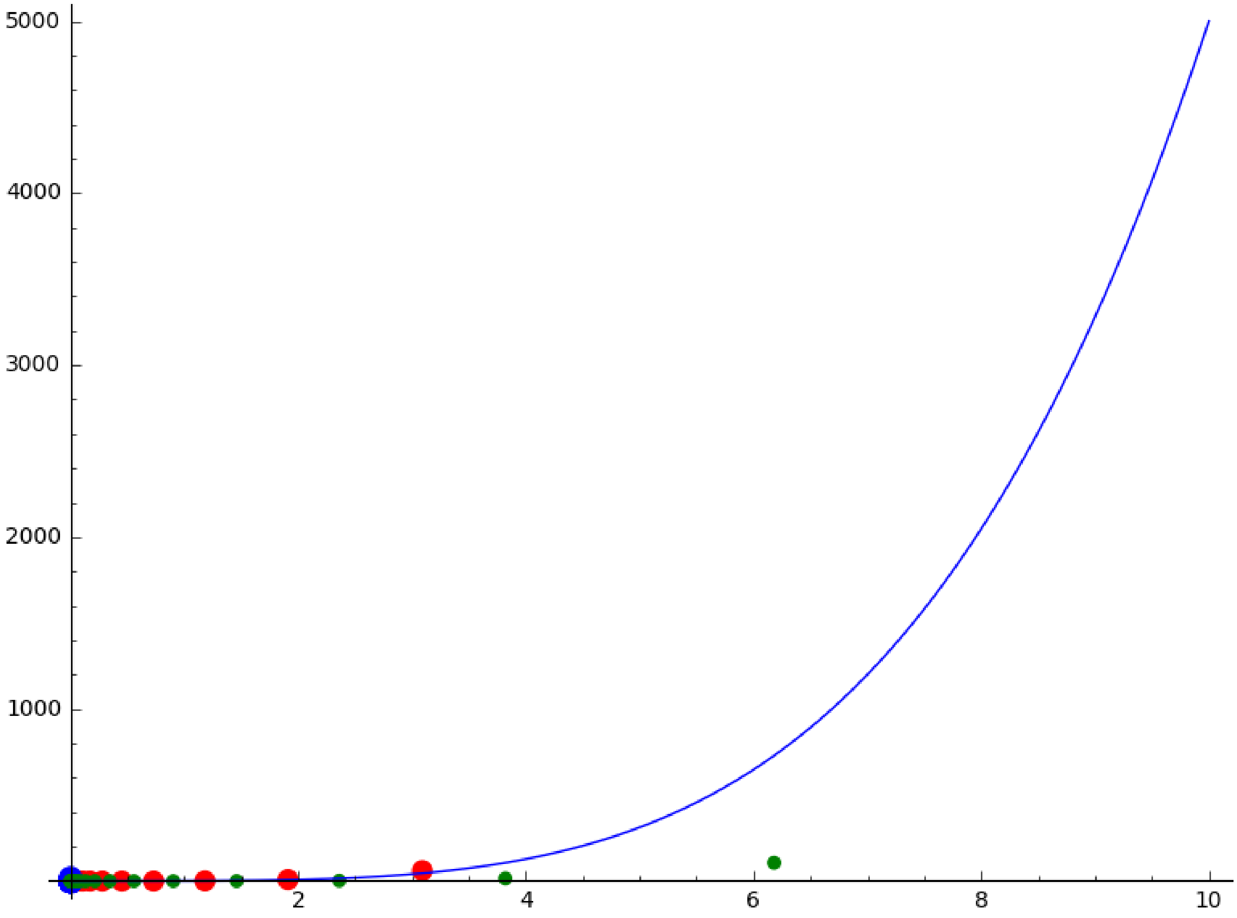
a = 0
b = 2

tikslumas = 0.0001
intervalas = [0,10]
Kaip matome, IDP (Intervalo Dalijimas Pusiau) metodas artėja greičiau prie minimumo nei AP (Auksinio Pjūvio) metodas, tačiau daugiau kartų iškviečia funkciją.

IDP bandymo taškų ir tikslo funkcijos visualizacija



AP bandymo taškų ir tikslo funkcijos visualizacija



Taškų spalvų reikšmės:
Mėlini = kairysis tiriamojo intervalo rėžis
Žali – dešinysis tiriamojo intervalo rėžis
Raudoni – tiriamojo intervalo vidurinysis taškas
Iteracijos
IDP:

```
1:       0.000000000000000 2.50000000000000 5.00000000000000
2:       0.000000000000000 1.25000000000000 2.50000000000000
3:       0.000000000000000 0.625000000000000 1.25000000000000
4:       0.000000000000000 0.312500000000000 0.625000000000000
5:       0.000000000000000 0.156250000000000 0.312500000000000
6:       0.000000000000000 0.0781250000000000 0.156250000000000
7:       0.000000000000000 0.0390625000000000 0.0781250000000000
8:       0.000000000000000 0.0195312500000000 0.0390625000000000
9:       0.000000000000000 0.00976562500000000 0.0195312500000000
10:      0.000000000000000 0.00488281250000000 0.00976562500000000
11:      0.000000000000000 0.00244140625000000 0.00488281250000000
12:      0.000000000000000 0.00122070312500000 0.00244140625000000
13:      0.000000000000000 0.000610351562500000 0.00122070312500000
14:      0.000000000000000 0.000305175781250000 0.000610351562500000
15:      0.000000000000000 0.000152587890625000 0.000305175781250000
16:      0.000000000000000 0.0000762939453125000 0.000152587890625000
17:      0.000000000000000 0.0000381469726562500 0.0000762939453125000
Vidurio taškas  0.0000381469726562500 Intervalo dydis 0.0000762939453125000
```

```
AP:
1:       0.000000000000000 3.09016994374947 6.18033988749895
2:       0.000000000000000 1.90983005625053 3.81966011250105
3:       0.000000000000000 1.18033988749895 2.36067977499790
4:       0.000000000000000 0.729490168751577 1.45898033750315
5:       0.000000000000000 0.450849718747372 0.901699437494743
6:       0.000000000000000 0.278640450004207 0.557280900008413
7:       0.000000000000000 0.172209268743165 0.344418537486330
8:       0.000000000000000 0.106431181261041 0.212862362522083
9:       0.000000000000000 0.0657780874821237 0.131556174964247
10:      0.000000000000000 0.0406530937789178 0.0813061875578356
11:      0.000000000000000 0.0251249937032076 0.0502499874064153
12:      0.000000000000000 0.0155281000757110 0.0310562001514221
13:      0.000000000000000 0.00959689362749838 0.0191937872549968
14:      0.000000000000000 0.00593120644821177 0.0118624128964235
15:      0.000000000000000 0.00366568717928750 0.00733137435857500
16:      0.000000000000000 0.00226551926892604 0.00453103853785208
17:      0.000000000000000 0.00140016791036324 0.00280033582072647
18:      0.000000000000000 0.000865351358562805 0.00173070271712561
19:      0.000000000000000 0.000534816551801320 0.00106963310360264
20:      0.000000000000000 0.000330534806762373 0.000661069613524745
21:      0.000000000000000 0.000204281745041612 0.000408563490083225
22:      0.000000000000000 0.000126253061718984 0.000252506123437968
23:      0.000000000000000 0.0000780286833244048 0.000156057366648810
24:      0.000000000000000 0.0000482243783945790 0.0000964487567891581
Vidurio taškas 0.0000482243783945790 Intervalo dydis 0.0000964487567891581
```

Naudota priemonė SageMath

Intervalo dalinimo pusiau kodas:

```
def toStr(*ob):
    s = ""
    for arg in ob:
        s+=" "+str(numerical_approx(arg))
    return s
a=0
b=2
f = ((x**2-a**2)**2)/b - 1
epsilon = 0.0001
right = 10
left = 0
xm = (right+left)/2
difference = right - left
fxm = f(xm)
var('x')
iteration=0
leftpoints,rightpoints,middlepoints = [],[],[]
while difference > epsilon:
    leftpoints.append((left,0))
    rightpoints.append((right,0))
    middlepoints.append((xm,fxm))
    x1 = left + difference/4
    x2 = right - difference/4
    fx1 = f(x1)
    fx2 = f(x2)
    if fx1 < fxm :
        right = xm
        xm = x1
        fxm = fx1
    elif fx2 < fxm:
        left = xm
        xm = x2
        fxm = fx2
    else:
        left = x1
        right = x2
    difference = right - left
    iteration += 1
    print(str(iteration)+":\t"+toStr(left,xm,right))
    xList = [left,right]
    yList = [0,0]
print("Vidurio taškas "+toStr(xm))
print("Intervalo dydis"+toStr(right-left))
showPoints1 = list_plot(middlepoints, color = "red", size = 90)
showPoints2 = list_plot(leftpoints, color = "blue", size = 120)
showPoints3 = list_plot(rightpoints, color = "green", size = 40)
show(showPoints1+showPoints2+showPoints3 + plot(f,(0,10)))
```

Auksinio pjūvio kodas:

```
def toStr(*ob):
    s = ""
    for arg in ob:
        s+=" "+str(numerical_approx(arg))
    return s
a=0
b=2
f = ((x**2-a**2)**2)/b - 1
fi = (-1 + sqrt(5))/2
epsilon = 0.0001
right = 10
left = 0
numerical_approx(fi)
difference = right - left
xL = right - fi*difference
xR = left + fi*difference
var('x')
iteration=0
fxR = f(xR)
fxL = f(xL)
leftpoints,rightpoints,middlepoints = [],[],[]
while difference > epsilon:
    if fxR < fxL :
        left = xL
        difference = right - left
        xL = xR
        fxL = fxR
        xR = left + fi*difference
        fxR = f(xR)
    else:
        right = xR
        difference = right - left
        xR = xL
        xL = right - fi*difference
        fxR = fxL
        fxL = f(xL)
    currentMiddlePoint =((left+right)/2,(fxR+fxL)/2)
    middlepoints.append(currentMiddlePoint)
    leftpoints.append((left,fxL))
    rightpoints.append((right,fxR))
    iteration+=1
    print(str(iteration)+":\t"+toStr(left,right))
print("Vidurio taškas "+str(numerical_approx((left+right)/2)))
print("Intervalo dydis"+toStr(right-left))
showPoints1 = list_plot(middlepoints, color = "red", size = 90)
showPoints2 = list_plot(leftpoints, color = "blue", size = 120)
showPoints3 = list_plot(rightpoints, color = "green", size = 40)
show(showPoints1+showPoints2+showPoints3 + plot(f,(0,10)))
```