Univerza v Ljubljani
Fakulteta za matematiko in fiziko
Univerzitetni študijski program
Finančna matematika

finančni praktikum

HORIZONTALNO PREKRIVANJE POLIGONOV

Avtorja: Luka Bizjak, Lucija Udir

Ljubljana, 2021

Kazalo

1 Navodilo **1**

2 Generiranje konveksnih poligonov **1**
2.1 Opis kode za generiranje poligonov 2

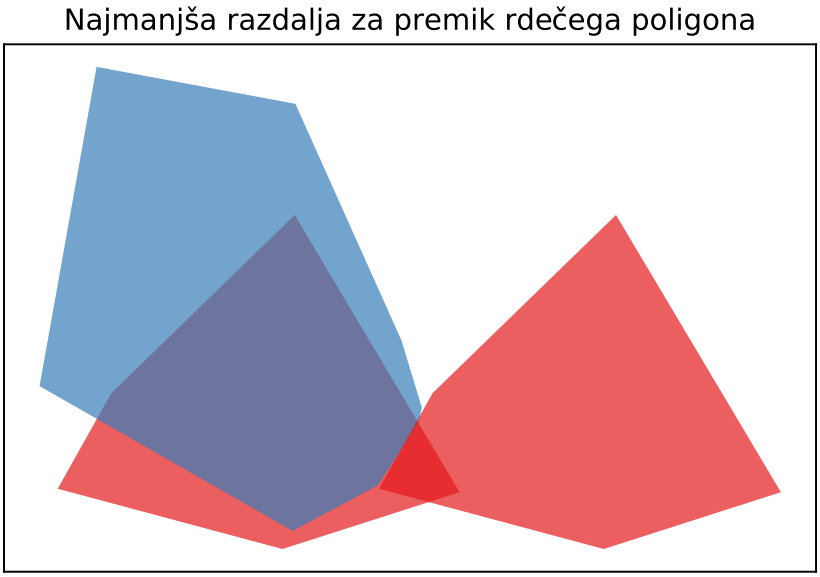
3 Iskanje minimalnega premika **4**

4 Primeri in ugotovitve **5**
4.1 Primeri generiranja 5
4.2 Iskanje minimalnega premika na primerih 5

5 Viri **7**

1 Navodilo

Generiraj moder in rdeč konveksen poligon v ravnini ter zapiši program, ki poišče potreben minimalni horizontalni premik enega izmed teh dveh poligonov, da postaneta disjunktna.



Slika 1: Premik rdečega poligona, da poligona postaneta disjunktna.

2 Generiranje konveksnih poligonov

Zgornji problem sva reševala v programskem jeziku Python. Konstruirala sva naključna konveksna poligona in izračunala njuno horizontalno prekrivanje. Konveksen poligon je enostaven mnogokotnik, katerega notranjost je konveksna množica. To pomeni, da vsaka daljica med dvema točkama, ki ležita v notranjosti ali na robu poligona, leži tudi v njem.

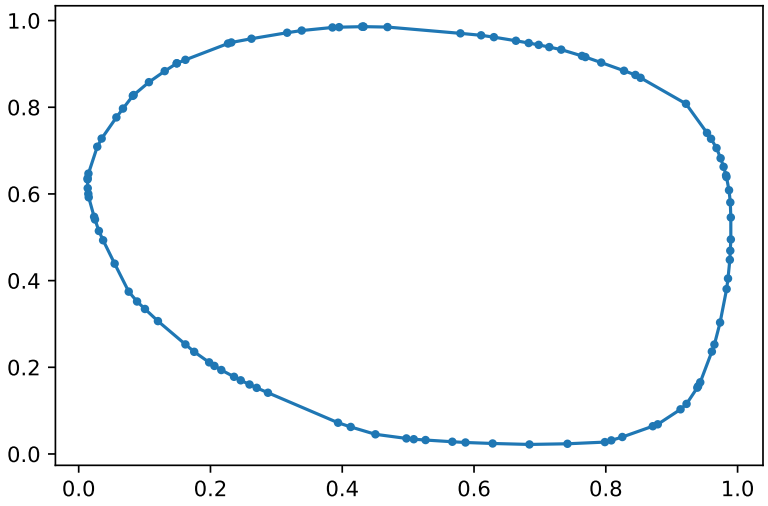
Njihovega generiranja sva se lotila na tri različne načine.

1. Način: generiranje s pomočjo algoritma, ki je opisan v članku “Probability that n random points are in convex position.”, ki ga je napisal Pavel Valtr.
2. Način: generiranje s pomočjo konveksne ovojnice.
3. Način: “brute-force” metoda.

2.1 Opis kode za generiranje poligonov

1. Način: Algoritem generira konveksne poligone s časovno zahtevnostjo $O(N \log(N))$ in je zato prmeren tudi za generiranje poligonov z velikim številom kotov (1000, 10000, itd.).

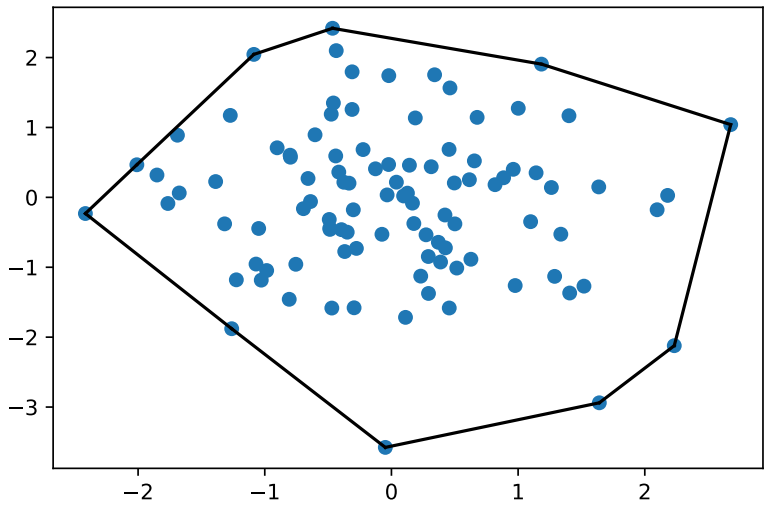
   - Algoritem generira dva seznama xs in ys z N naključnimi celimi števili med 0 in konstanto C.
   - Razvrsti xs in ys po velikosti in si zapomni najmanjša in največja elementa.
   - Ostale elemente naključno razdeli na dva nova seznama x1, x2 in y1, y2, s pomočjo funkcije _to_vectors_coordinates.
   - Vsakemu izmed teh seznamom dodamo pripadajoči minimalni in maksimalni element.
   - Poišče zaporedne razlike x1$[i+1]-$ x1$[i]$, ter obratno za drugi seznam x2$[i]-$ x2$[i+1]$. Te dva seznama shrani v nova seznama.
   - Iz novih seznamov naredi vectors_xs in vectors_ys in premeša vectors_ys.
   - Uredi vektorje po kotu oziroma smeri.
   - Premakne poligon nazaj na prvotne koordinate.



Slika 2: Primer generiranja 100 kotnika.

2. Način: s pomočjo knjižnice scipy.spatial in funkcije ConvexHull sva generirala konveksno ovojnico za podano število točk. Časovna zahtevnost algoritma je $O(N \log(N))$.

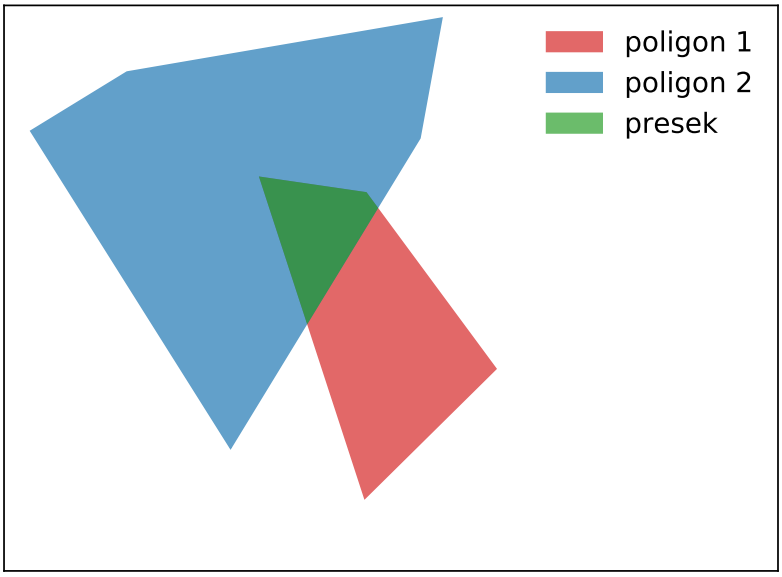
   - Za začetno točko izberemo tisto izmed generiranih točk, ki ima najmanjšo y koordinato, če je takih točk več, izberemo tisto, ki ima najmanjšo x koordinato.
   - Skozi začetno točko in vse ostale točke naredimo vektorje in izračunamo smer vektorja.
   - Vektorje uredimo po velikosti in sicer od 0 do $-\infty$ ter od ∞ do 0.
   - Naredimo pregled treh točk v negativni smeri. Če te tri točke predstavljajo desni obrat, kar pomeni da je kot med temi tremi točkami med π in 2π, se premaknemo naprej, sicer izločimo srednjo točko.



Slika 3: Primer generiranja konveksne ovojnice na 100 točkah.

3. Način: generiranje mnogokotnikov s pomočjo funkcij:

   - is_convex_polygon, ki preveri ali je dani poligon konveksen. Torej bo funkcija preverila ali je vsak notranji kot manjši ali enak π.
   - gen_rand_poly, ki zgenerira naključen konveksen poligon s poljubnim številom oglišč.
   - plot_polys, ki izriše poligon na koordinatni sistem.



Slika 4: Primer dveh konveksnih poligonov.

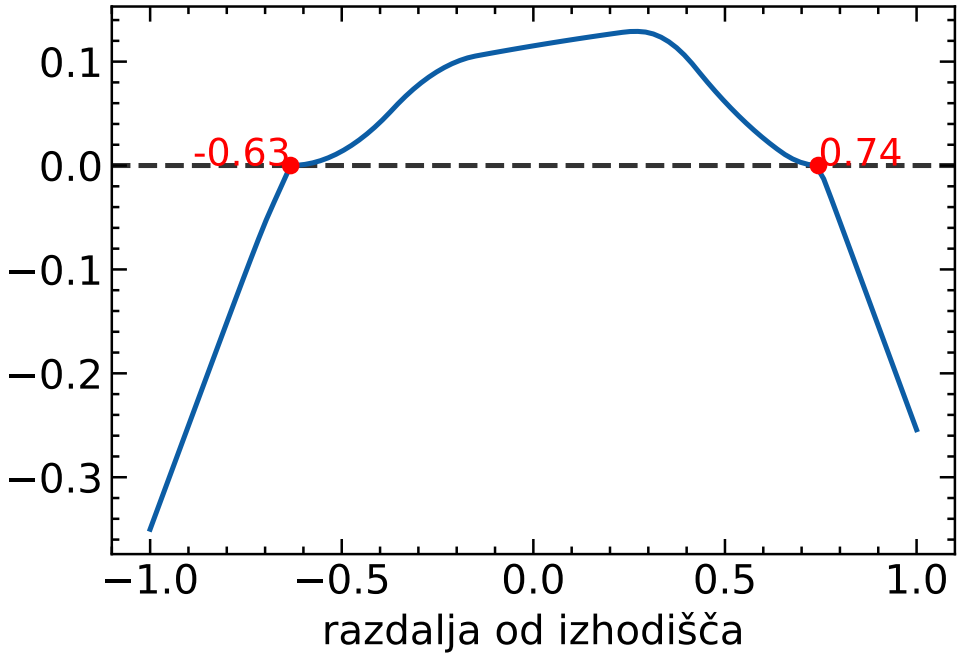
3 Iskanje minimalnega premika

Za iskanje minimalnega premika sva definirala naslednje funkcije:

- Funkcijo move_polygon(poly, x), ki sprejme matriko v kateri so shranjene koordinate oglišč poligona, ter začetni premik x za premikanje poligona v smeri abcisne osi. Funkcija x koordinatam poligona prišteje oziroma odšteje dano vrednost. Vrne premaknen poligon.

- optimize_fun(x, poly1, poly2) funkcija sprejme premik x po abcisni osi in dve matriki s koordinatami poligona. Ta premakne rdeč poligon za dani x in izračuna ploščino preseka premaknjenega rdečega in modrega poligona ter njuno oddaljenost. Funkcija bo vrnila razliko med njunim presekom in razdaljo. Ko bosta poligona disjunktna, bo ploščina njunega preseka enaka 0, razdalja med njima pa nenegativna. Z večanjem pomika rdečega poligona, bo tako vrednost te funkcije vedno bolj negativna. V primeru, ko je ploščina preseka teh dveh poligonov različna od 0, bo razdalja med njima enaka 0 in vrednost funkcije bo pozitivna.

- calc_min_distance(poly1, poly2) funkcija poišče potreben minimalni premik rdečega poligona v smeri abcisne osi, da rdeč in moder poligon postaneta disjunktna. Ko se to zgodi sta njuna razdalja in presek enaka 0, zato bova s pomočjo Newtonove metode poiskala ničlo prejšne funkcije. Newton-Raphson metoda uporabi sekantno metodo, če odvod ni podan. calc_min_distance(poly1, poly2) ima dve ničli: eno dobimo s premikom v desno (x0= 0.5) in drugo s premikom v levo (x0= -0.5). Za iskani minimalni premik izberemo najmanjšo po absolutni vrednosti.



Slika 5: Funkcija `f(x, RdecPoligon, ModerPoligon)`.

4 Primeri in ugotovitve

4.1 Primeri generiranja

Zanimala naju je posebnost generiranja pri vsakem izmed načinov.

1. Način: Že pri 15-kotniku lahko skoraj zanesljivo vidimo obliko elipse. Bolj kot povečujemo število točk in s tem število kotov bolj je oblika mnogokotnika podobna superpoziciji kvadrata in kroga.

2. Način: zanimalo naju je koliko kotnike dobimo iz podanega števila točk na katerih generiramo konveksno ovojnico. Za vsako število točk sva kodo za generiranje pognala dvajsetkrat in nato izračunala povprečno število kotov. Opazimo lahko, da se s povečevanjem števila točk povečuje tudi število točk v konveksni ovojnici. Vidimo, da se generirajo liki vedno bolj podobni elipsi.

Število točk:	10	100	1000	10000	100000	1000000
Povprečno število kotov:	5.7	9.1	10.8	14.9	17.6	20.5

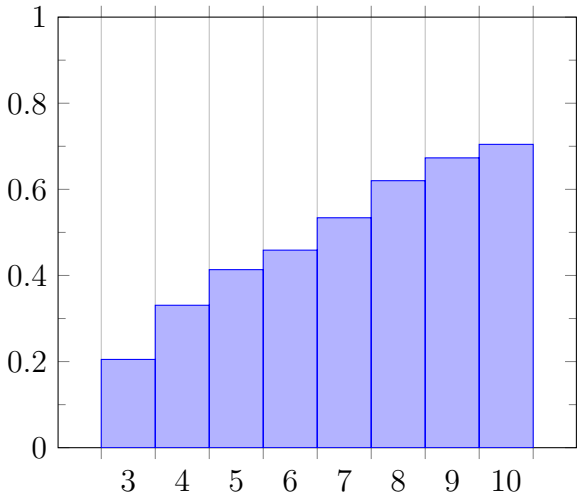
3. Način: to kodo lahko poganjamo le za poligone z majhnim številom kotov. Že za generiranju 12 kotnikov je koda prepočasna, zato oblike mnogogkotnikov z velikm številom stranc ne moramo zagotovo opisati a lahko sklepamo iz zgornjega, da bodo liki vedno bolj podobni elipsi.

4.2 Iskanje minimalnega premika na primerih

Zanimalo naju je kakšna je povezava med število kotov in povprečnim premikom. Genirirala sva dva poligona z istim številom kotov in s pomočjo zgoraj opisanih funkcij izračinala premik. Za vsako število kotov sva poskus ponavljala dvajsetkrat in izračunala povprečni minimalni premik poligona, da poligona postaneta disjunktna.

Število oglišč:	3	4	5	6	7	8	9	10
Povprečni min premik:	0.2050	0.3308	0.4135	0.4588	0.5340	0.6201	0.6731	0.7045

Histogram: povezava med številom oglišč in povprečnim minimalnim premikom



Iz dobljenih rezultatov vidimo da se s povečevanjem števila kotov potreben premik povečuje. Rast potrebnega povprečnega minimalnega premika spominja na korensko funkcijo. Za večje število kotov pa dobimo naslednje rezultate.

Število oglišč:	30	50	100	1000
Povprečni min premik:	0.90	0.95	0.99	0.999

Zelo hitro je ta premik skoraj 1, kar pomeni da sta mnogokotnika za velike N skoraj enaka in se skoraj v celoti prekrivata. To se zgodi, ker smo generirali točke na omejeni množici.

Za generiranje dveh 10000 kotnikov in izračun minimalnega premika, program potrebuje le 1 sekundo. Pri generiranje 1000000 kotnika 12 sekund. Za 10000000 kotnike pa že kar 2 minuti. Porabljen čas je predvsem za generiranje poligona. Kljub naraščanju časovne zahtevnosti je algoritem dovolj hiter, saj bi bilo generiranje elipse kateri je podoben mnogokotnik za velike N hitrejša, rezultat potrebnega minimalnega premika pa podoben.

5 Viri

1. Refubium: *Probability that n random points are in convex position.* Citirano: 27.11.2021.
   Dostopno na naslovu: https://refubium.fu-berlin.de/bitstream/handle/fub188/17874/1994_01.pdf?sequence=1&isAllowed=y

2. Nauk.si: *Konveksne ovojnice.* Citirano: 28.11.2021.
   Dostopno na naslovu: http://www.nauk.si/materials/6858/out/?printSlides=*