Háskóli Íslands

Töluleg greining

Stórt verkefni I

Atli Fannar Franklín
Hjörvar Logi Ingvarsson

Kennari:
Sigurður Freyr Hafstein

19. febrúar 2019

Öll hjálparföll má finna í viðauka ásamt öðrum kóða. Í þessu verkefni notuðumst við við sage sem forritunarmálið okkar. Sage er útvíkkun á python þ.e.a.s. python með numpy, scipy, sympy, osfrv. innihaldið sjálfkrafa.

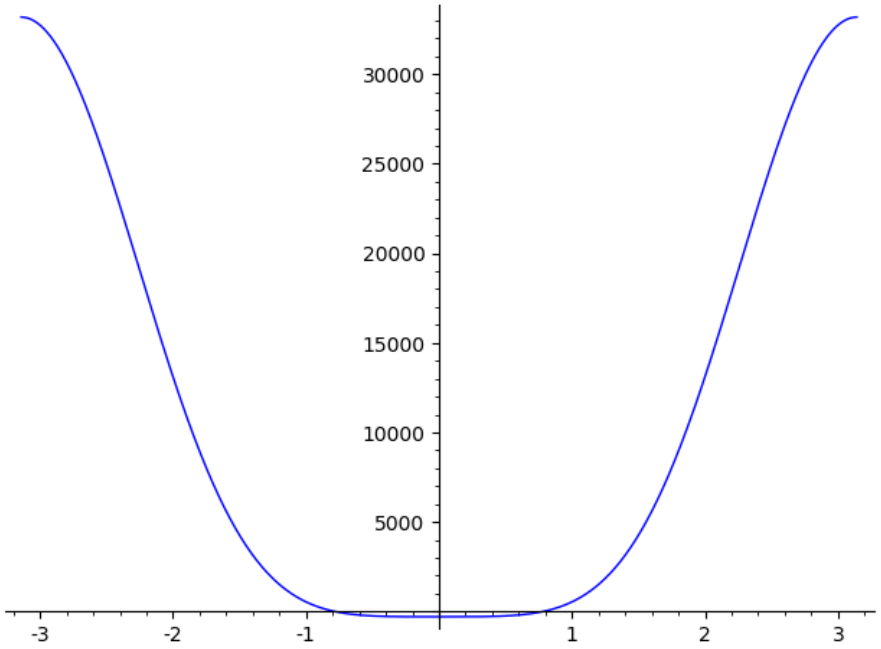
Fyrst er beðið um að forrita fall f sem tekur inn θ (liður 1). Þetta fall má finna í viðaukanum. Ef við prófum fallið á umbeðnu inntaki fæst eftirfarandi niðurstaða:

```
sage: x1, x2, y2 = 4, 0, 4
sage: L1, L2, L3 = 2, sqrt(2), sqrt(2)
sage: gam, p1, p2, p3 = pi/2, sqrt(5), sqrt(5), sqrt(5)
sage: f(pi/4)
0.000000000000000
sage: f(-pi/4)
0.000000000000000
```

Nú má teikna graf af f með allar breytur fastar nema θ með því að nota g hjálparfallið í viðaukanum og skipunina:

```
sage: h = g(4, 0, 4, 2, sqrt(2), sqrt(2), pi/2, sqrt(5), sqrt(5), sqrt(5))
sage: plot(h, (-pi,pi))
```

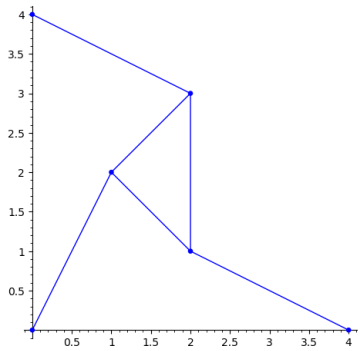
Þetta gefur myndina:

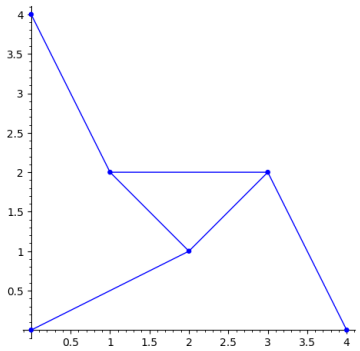


Nú getum við nýtt okkur `plotter` hjálparfallið í viðauka til að teikna myndirnar 15a og 15b í bók upp á nýtt með skipununum:

```
sage: plotter(4, 0, 4, 2, sqrt(2), sqrt(2), 2, 1, pi/4, pi/2)
sage: plotter(4, 0, 4, 2, sqrt(2), sqrt(2), 1, 2, -pi/4, pi/2)
```

Þetta gefur myndirnar:

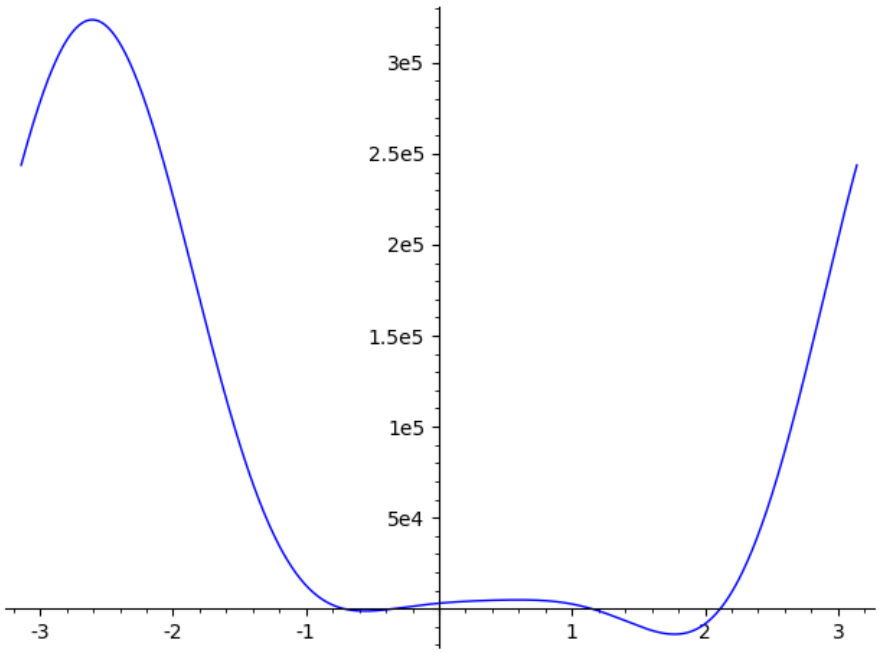




Nú getum við næst teiknað graf f með þessum nýju föstum (lið 4) með því að kalla á g úr viðauka með eftirfarandi skipunum:

```
sage: h = g(5, 0, 6, 3, 3*sqrt(2), 3, pi/4, 5, 5, 3)
sage: plot(h, (-pi,pi))
```

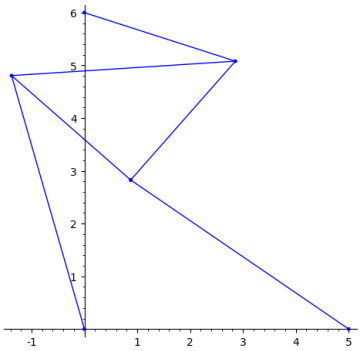
sem gefur myndina:

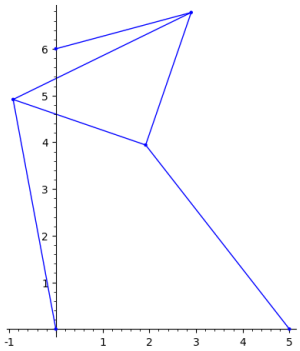


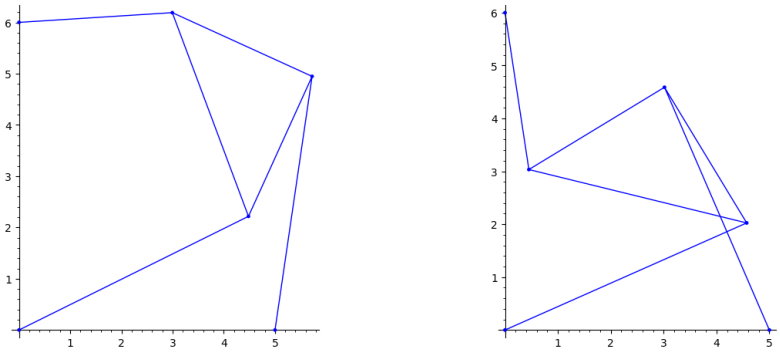
Getum nú fundið allar núllstöðvarnar sem f hefur með því að nýta okkur hjálparfallið `find_all_roots` í viðauka ásamt hjálparfallinu `xyfromth` til að finna x og y gildin sem samsvara lausnarhorninu θ. Þetta getum við allt gert ásamt því að teikna samsvarandi myndir með eftirfarandi skipunum:

```
sage: r = find_all_roots(g(5, 0, 6, 3, 3*sqrt(2), 3, pi/4, 5, 5, 3), -pi, pi)
sage: xys = [xyfromth(5, 0, 6, 3, 3*sqrt(2), 3, pi/4, 5, 5, 3, k) for k in r]
sage: for i in range(len(r)):
....:     plotter(5, 0, 6, 3, 3*sqrt(2), 3, xys[i][0], xys[i][1], r[i], pi/4)
```

Þá fáum við eftirfarandi fjórar myndir:



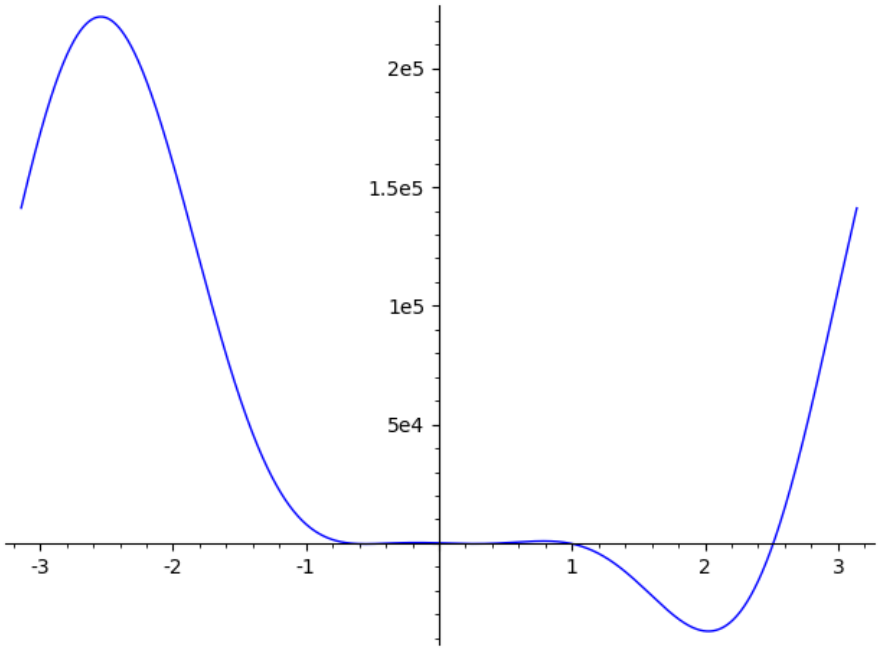




Auðvelt er að sjá útfrá myndunum að við fáum réttar lengdir á p_1,p_2 og p_3. Nú ef við breytum p_2 í 7 getum við fengið sama plot nema fyrir þá breytingu með því að kalla á skipunina:

```
sage: h = g(5, 0, 6, 3, 3*sqrt(2), 3, pi/4, 5, 7, 3)
sage: plot(h, (-pi,pi))
```

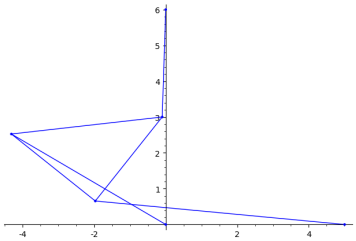
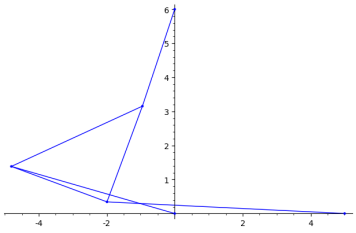
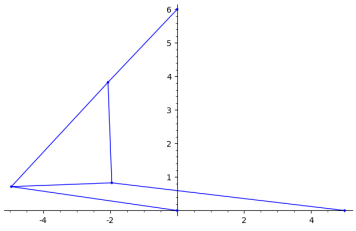
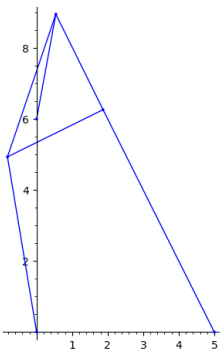
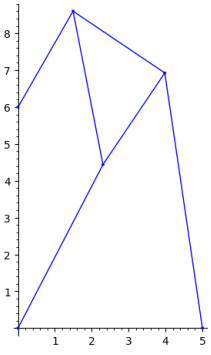
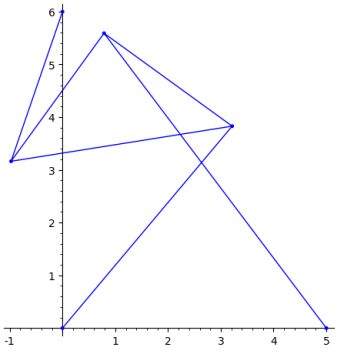
Fáum þá eftirfarandi mynd:



Getum nú fundið allar núllstöðvar og samsvarandi myndir svipað og áður með skipununum:

```
sage: r = find_all_roots(g(5, 0, 6, 3, 3*sqrt(2), 3, pi/4, 5, 7, 3), -pi, pi)
sage: xys = [xyfromth(5, 0, 6, 3, 3*sqrt(2), 3, pi/4, 5, 7, 3, k) for k in r]
sage: for i in range(len(r)):
....:     plotter(5, 0, 6, 3, 3*sqrt(2), 3, xys[i][0], xys[i][1], r[i], pi/4)
```

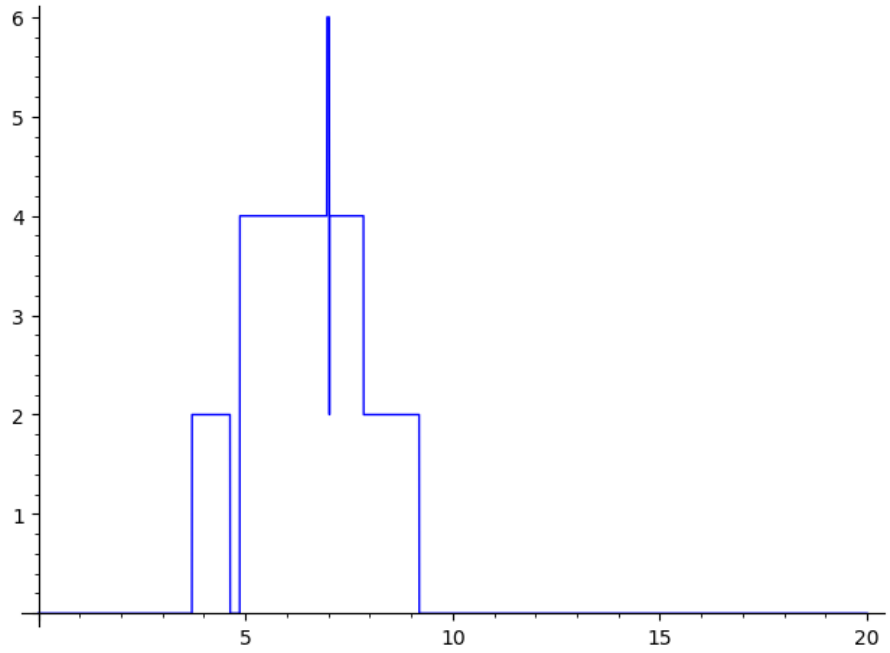
Þá fáum við eftirfarandi sex myndir:

Við getum nú teiknað graf af fjölda lausna $f(\theta) = 0$ sem fall af p_2 með því að nýta hjálparfallið `solsnumfromp2` og eftirfarandi skipun:

```
sage: h = solsnumfromp2(5, 0, 6, 3, 3*sqrt(2), 3, pi/4, 5, 3)
sage: plot(h, 0, 20)
```

Þetta gefur okkur eftirfarandi mynd:



Þetta sýnir að ef við viljum hafa nákvæmlega tvær lausnir er m.a. hægt að velja $p_2 = 4$. Nú til að ákvarða nákvæma endapunkta hvers fallsgildisbils getum við notað eftirfarandi forritskóða:

```
sage: curh = solsnumfromp2(5, 0, 6, 3, 3*sqrt(2), 3, pi/4, 5, 3)
sage: curv = curh(0)
sage: changes = []
sage: for i in range(1000):
....:     nxtv = curh((i + 1) / 100)
....:     if nxtv != curv:
....:         changes.append((i / 100), (i + 1) / 100))
....:     curv = nxtv
```

Þetta gefur okkur 8 lítil bil sem skiptipunktarnir liggja í. Til þess að fá nákvæmari (upp að 9 aukastöfum) skiptipunkta notum við eftirfarandi forritskóða:

```
sage: exactchanges = [float("-inf")]
sage: for c in changes:
....:     lpos, rpos = c[0], c[1]
....:     lval, rval = curh(lpos), curh(rpos)
....:     while rpos - lpos > 1e-9:
....:         mpos = ((lpos + rpos) / 2).n()
....:         mval = curh(mpos)
....:         if mval == lval:
....:             lpos = mpos
....:         else:
....:             rpos = mpos
....:     exactchanges.append((lpos + rpos) / 2)
sage: exactchanges.append(float("inf"))
```

Til þess að fá hvaða fallgildi eru á hvaða bilum getum við loks notað eftirfarandi forritskóða:

```
sage: intervals = []
sage: for i in range(len(exactchanges) - 1):
....:     intervals.append(((exactchanges[i], exactchanges[i + 1]),
....:         curh((exactchanges[i] + exactchanges[i + 1]) / 2)))
```

Sem gefur eftirfarandi niðurstöðu:

```
[((-inf, 3.71053114980459), 0),
 ((3.71053114980459, 4.63160214930773), 2),
 ((4.63160214930773, 4.86372385472059), 0),
 ((4.86372385472059, 6.96734398752451), 4),
 ((6.96734398752451, 7.02234040886164), 6),
 ((7.02234040886164, 7.03075937300920), 2),
 ((7.03075937300920, 7.84908692449331), 4),
 ((7.84908692449331, 9.19254726916552), 2),
((9.19254726916552, inf), 0)]
```

Að lokum áttum við að framlengju niðurstöður þannig þær gangi í þremur víddum (svokallað Stewart platform). Verkefnið má skilgreina þannig að við fáum gefnar lengdir allra brúna í óreglulegum áttflötung og við viljum finna hnit hornpunkta hliðar í hnitakerfi sem skilgreinist þannig að gagnstæð hlið liggi í XY-planinu.
Við getum kallað hornpunkta áttflötungsins A, B, C, X, Y, Z þar sem A, B, C

eru hnitin sem við þekkjum og X, Y, Z eru hnitin sem við viljum finna.
Ástæðan fyrir að við þekkjum A, B, C er að við getum valið einn þeirra sem núllpunktinn, $(0, 0, 0)$, og leitt út hina með hornaföllum (við þekkjum lengdirnar $|AB|, |BC|, |AC|$ þannig þyrftum bara beita kósínusarreglu).
Næst getum við notað lengdirnar $|BX|$ og $|CX|$ til þess að finna skurðferil kúlnanna með miðjur í B og C með þessa radíusa (í þessari röð). Þetta má einnig gera fyrir hin tvö hnitin, Y og Z, s.s. með því að skoða $|AY|$ og $|CY|$ með miðjur A og C og svo $|AZ|$ og $|BZ|$ með miðjur A og B.
Þá höfum við þrjá hringferla í $\mathbb{R}^3$ (með jákvæð z-hnit) og þurfum að finna hnit á þeim, X, Y, Z, þannig að lengdirnar $|XY|, |YZ|, |ZX|$ séu eins og inntök. Þetta verkefni má leysa með því að finna ystu gildin fyrir einhvern hringferil, t.d. X, þ.a. $|XY|$ og $|ZX|$ geti verið rétt lengd og svo helmingunarleita á því bili hvort $|YZ|$ gangi.
Verkefnið er s.s. að beita ítrekað leitunaraðferðum til að mæta sífellt harðari skorðum þangað til að hnitin X, Y, Z eru fundin.

Viðauki - kóði:

```
def f(th):
    A2 = L3*cos(th)-x1
    B2 = L3*sin(th)
    A3 = L2*cos(th+gam) - x2
    B3 = L2*sin(th+gam) - y2
    D = 2*(A2*B3-B2*A3)
    if D == 0:
        raise ValueError("D is 0")
    N1 = B3*(p2**2-p1**2-A2**2-B2**2)
    N1 -= B2*(p3**2-p1**2-A3**2-B3**2)
    N2 = A2*(p3**2-p1**2-A3**2-B3**2)
    N2 -= A3*(p2**2-p1**2-A2**2-B2**2)
    return (N1**2+N2**2-p1**2*D**2).n()

def g(x1, x2, y2, L1, L2, L3, gam, p1, p2, p3):
    def f(th):
        A2 = L3*cos(th)-x1
        B2 = L3*sin(th)
        A3 = L2*cos(th+gam) - x2
        B3 = L2*sin(th+gam) - y2
        D = 2*(A2*B3-B2*A3)
        if D == 0:
            raise ValueError("D is 0")
        N1 = B3*(p2**2-p1**2-A2**2-B2**2)
        N1 -= B2*(p3**2-p1**2-A3**2-B3**2)
        N2 = A2*(p3**2-p1**2-A3**2-B3**2)
        N2 -= A3*(p2**2-p1**2-A2**2-B2**2)
```

```
        return (N1**2+N2**2-p1**2*D**2).n()
    return f

def plotter(x1, x2, y2, L1, L2, L3, x, y, th, gam):
    res = circle((0,0),0.025,fill=True,rgbcolor=(0,0,1))
    res += circle((x1,0),0.025,fill=True,rgbcolor=(0,0,1))
    res += circle((x2,y2),0.025,fill=True,rgbcolor=(0,0,1))
    res += circle((x,y),0.025,fill=True,rgbcolor=(0,0,1))
    res += circle((x+L3*cos(th),y+L3*sin(th)),
        0.025,fill=True,rgbcolor=(0,0,1))
    res += circle((x+L2*cos(th+gam),y+L2*sin(th+gam)),
        0.025,fill=True,rgbcolor=(0,0,1))
    res += line([(0,0),(x,y),(x+L2*cos(th+gam),
        y+L2*sin(th+gam))],rgbcolor=(0,0,1))
    res += line([(x1,0),(x+L3*cos(th),y+L3*sin(th)),
        (x,y)],rgbcolor=(0,0,1))
    res += line([(x2,y2),(x+L2*cos(th+gam),y+L2*sin(th+gam)),
        (x+L3*cos(th),y+L3*sin(th))],rgbcolor=(0,0,1))
    return res

def xyfromth(x1, x2, y2, L1, L2, L3, gam, p1, p2, p3, th):
    A2 = L3*cos(th)-x1
    B2 = L3*sin(th)
    A3 = L2*cos(th+gam) - x2
    B3 = L2*sin(th+gam) - y2
    D = 2*(A2*B3-B2*A3)
    if D == 0:
        raise ValueError("D is 0")
    N1 = B3*(p2**2-p1**2-A2**2-B2**2)
    N1 -= B2*(p3**2-p1**2-A3**2-B3**2)
    N2 = A2*(p3**2-p1**2-A3**2-B3**2)
    N2 -= A3*(p2**2-p1**2-A2**2-B2**2)
    return (N1/D).n(), (N2/D).n()

def find_all_roots(f, a, b, eps=0.0000000001):
    roots = []
    intervals_to_check = [(a,b)]
    while intervals_to_check:
        start, end = intervals_to_check.pop()
        try:
            root = find_root(f, start, end)
        except RuntimeError:
            continue
        if root in roots:
            continue
        if abs(f(root)) < 1:
```

```
            roots.append(root)
        intervals_to_check.extend([(start, root-eps),
            (root+eps, end)])
    roots.sort()
    return roots

def solsnumfromp2(x1, x2, y2, L1, L2, L3, gam, p1, p3):
    def h(p2):
        def f(th):
            A2 = L3*cos(th)-x1
            B2 = L3*sin(th)
            A3 = L2*cos(th+gam) - x2
            B3 = L2*sin(th+gam) - y2
            D = 2*(A2*B3-B2*A3)
            if D == 0:
                raise ValueError("D is 0")
            N1 = B3*(p2**2-p1**2-A2**2-B2**2)
            N1 -= B2*(p3**2-p1**2-A3**2-B3**2)
            N2 = A2*(p3**2-p1**2-A3**2-B3**2)
            N2 -= A3*(p2**2-p1**2-A2**2-B2**2)
            return (N1**2+N2**2-p1**2*D**2).n()
        return len(find_all_roots(f, -pi, pi))
    return h
```