Ústav teoretickej fyziky a astrofyziky, Prírodovedecká fakulta
Masarykova Univerzita

Pasenie Kozy

Meno: Ema Šipková
Predmet: F5330 Základné numerické metódy
Dátum: 31.10.2023

1.1 Zadanie

Predstavme si kruhový trávnik s polomerom R homogénne zarastený trávou. Na okraji trávnika je zabodnutý kolík. Lanom o dĺžke L je ku kolíku priviazaná koza. Koza je nenásytná, takže spasie všetku trávu, ktorá je v jej dosahu. Nájdite dĺžku lana, pri ktorej koza zošere práve jednu polovicu plochy trávnika. Ulohu môžete aj rozšíriť a nájsť takú posloupnosť dĺžok $\{L_n\}_{n=1}^{N}$ aby koza spásla trávnik po N rovnakých dieloch.

1.2 Rozbor a matematická formulácia

Majme kružnicu k_2 s polomerom L, do ktorej stredu umiestnime počiatok sústavy súradníc (stred má súradnice $[0,0]$) a kružnicu k_1 s polomerom R, ktorá má stred na súradniciach $[R,0]$. Rovnica, ktorá popisuje kružnicu k_2 má predpis:

$$x^2 + y^2 = L^2,$$

a kružnica k_1 má predpis:

$$(x-R)^2 + y^2 = R^2.$$

Prienik týchto kružníc označím bodom A so súradnicami $[x_1,y_1]$. Riešením tejto sústavy rovníc dostaneme pre súradnice bodu A nasledujúce hodnoty:

$$x_1 = \frac{L^2}{2R} \tag{1.1}$$

$$y_1 = L\sqrt{1 - \frac{L^2}{4R^2}} \tag{1.2}$$

Náčrt problému je možné vidieť na obrázku 1.1. Oblasť, ktorú koza spasie je daná súčtom jednotlivých výrezov $S = S_1 + S_2$. Keďže sa jedná o problém, ktorý je symetrický podľa osi x, môžem pri riešení uvažovať len hornú polovicu a následne rozšíriť riešenie na celý problém násobením dvojkou.

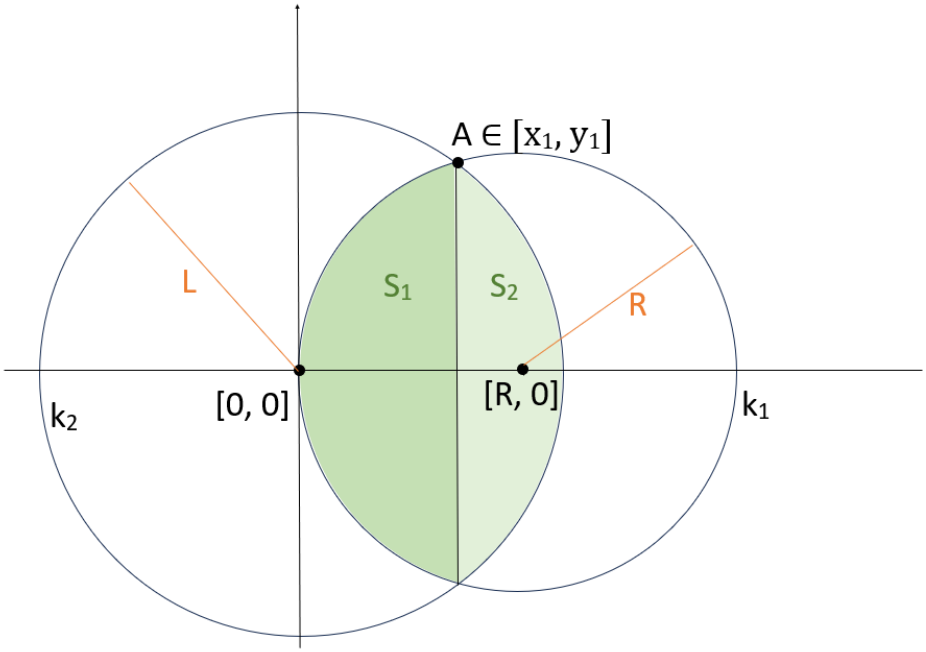


Figure 1.1: Schéma riešeného problému

Hľadaný obsah polovice kruhového odseku kružnice k_1 a k_2 vypočítam ako rozdiel obsahu kruhového výseku S_{o1}, S_{o2} so stredovým uhlom α_1, α_2 a obsahu trojuholníka S_{t1}, S_{t2}, ako je naznačené na obrázkoch 1.2.

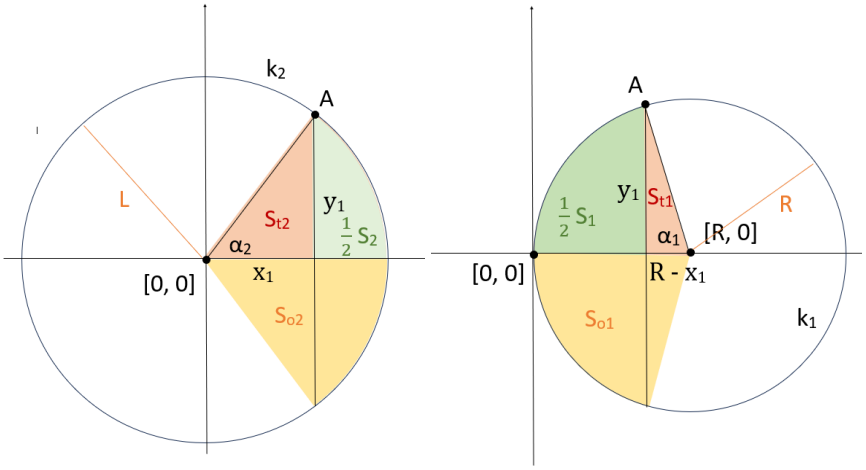


Figure 1.2: Schéma riešeného problému

Tento problém môžme zformulovať matematicky nasledovne:

$$S_1 = \alpha_1 R^2 - (R - x_1)y_1$$

$$S_2 = \alpha_2 L^2 - x_1 y_1$$

kde uhly α_1, α_2 sú vyjadrené v radiánoch s predpisom:

$$\alpha_1 = \arctan \frac{y_1}{R - x_1} \tag{1.3}$$

$$\alpha_2 = \arctan \frac{y_1}{x_1}. \tag{1.4}$$

Chceme aby sa súčet $S = S_1 + S_2$ rovnal polovici plochy trávnika, takže riešime rovnicu:

$$0 = 2\alpha_2 L^2 + (2\alpha_1 - \pi)R^2 - L\sqrt{4R^2 - L^2} \tag{1.5}$$

Aby sme našli postupnosť dĺžok $\{L_i\}_{i=1}^{N}$, tak, aby koza spásavala trávnik postupne po N dieloch, budeme riešiť rovnicu:

$$0 = 2\alpha_2 L^2 + (2\alpha_1 - \frac{i}{N}\pi)R^2 - L\sqrt{4R^2 - L^2} \tag{1.6}$$

1.3 Riešenie a opis použitej metódy

Riešenie vyššie uvedených transcendentných rovníc 1.5 a 1.6, je možné len pomocou numerických metód, keďže analytické riešenie rovníc s neznámou x v rôznych mocninách a v arctan neexistuje.

Na riešenie rovnice 1.5 a 1.6 použijem metódu bisekcie. Bisekcia, alebo tiež metóda polovičného delenia intervalu je iteračná metóda, ktorá rieši nelineárne rovnice v tvare $f(x) = 0$ definované na nejakom intervale $< a_0, b_0 >$. Pred výpočtom koreňov musím aspoň približne poznať interval, v ktorom sa hľadané korene danej funkcie nachádzajú.

Postup pri riešení nelineárnej rovnice bisekciou je znázornený na obrázku 1.3 a je možné ho zhrnúť nasledovne:

- Určíme intervalové hodnoty a_0 a b_0
- Pozrieme sa na ich znamienka - ak ide o interval s riešením, tak jedno znamienko bude + a druhé -
- Interval rozpolíme, určíme hodnotu $x_0 = \frac{1}{2}(b_0 - a_0)$
- Ďalší interval určíme tak, aby mali krajné hodnoty opačné znamienka
- Opakujeme

Chyba riešenia v n-tom kroku sa dá určiť pomocou vzťahu:

$$\varepsilon = \frac{b_0 - a_0}{2^n} \tag{1.7}$$

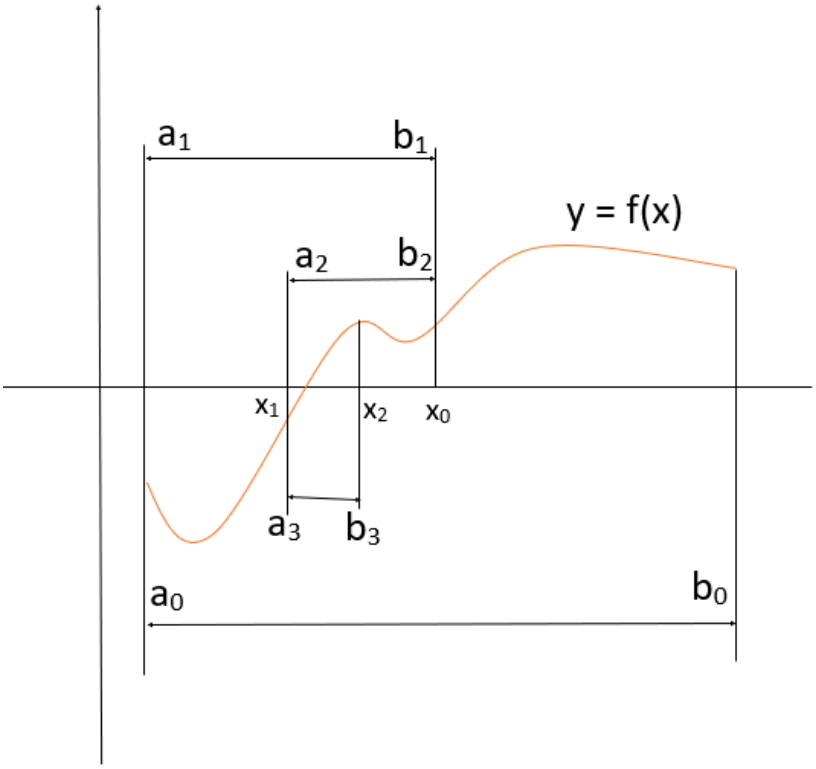


Figure 1.3: Schéma bisekcie

1.4 Výsledky a ich rozbor

Predtým, ako som rovnicu 1.5 vyriešila, som si vykreslila graf, ktorý ukazuje priebeh danej funkcie, aby som vedela zvoliť správnu dĺžku intervalu, v ktorom jej koreň budem hľadať. Táto funkcia je zobrazená na grafe 1.4

Interval som volila tak, aby som sa pri jeho konštrukcii vyhla asymptote a taktiež nule. Pre zvolenú hodnotu $R = 10$ som určila hodnotu dĺžky lana, ktorá pri daných parametroch bola $L = 11.587$. Chyba pri tomto výsledku je určená pomocou rovnice 1.7, a má hodnotu $\varepsilon = 1.585.10^{-12}$. Aby sme videli plochu, ktorú by koza spásla pri tejto dĺžke lana, vykreslila som tento problém do grafu 1.5.

Pre rôzne hodnoty R som tiež určila vzťah medzi parametrami R a L, ktorý som pomocou lineárneho fitu odvodila z vypočítaných hodnôt. Graf závislosti R na hodnote L je možné vidieť na obrázku 1.6. Závislosť medzi týmito veličinami sa dá približne vyjadriť pomocou vzťahu:

$$L = 1.15872847R \tag{1.8}$$

Pre určenie postupnosti dĺžok lana, tak aby koza vždy spásala rovnakú plochu som riešila rovnicu 1.6. V tejto úlohe som našla hodnoty $\{L_i\}_{i=1}^{N}$ pre $N = 10$ a

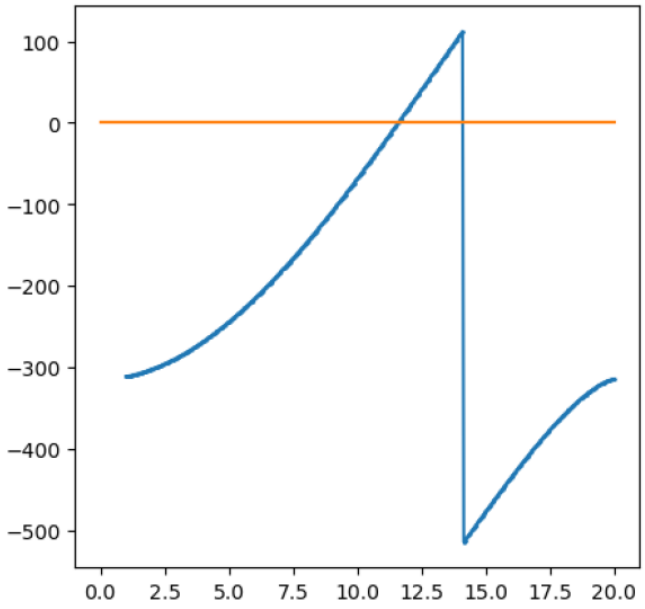


Figure 1.4: Priebeh funkcie na pravej strane rovnice 1.5. Asymptota je daná výrazom $L_{\text{asympt}} = \sqrt{2R^2}$

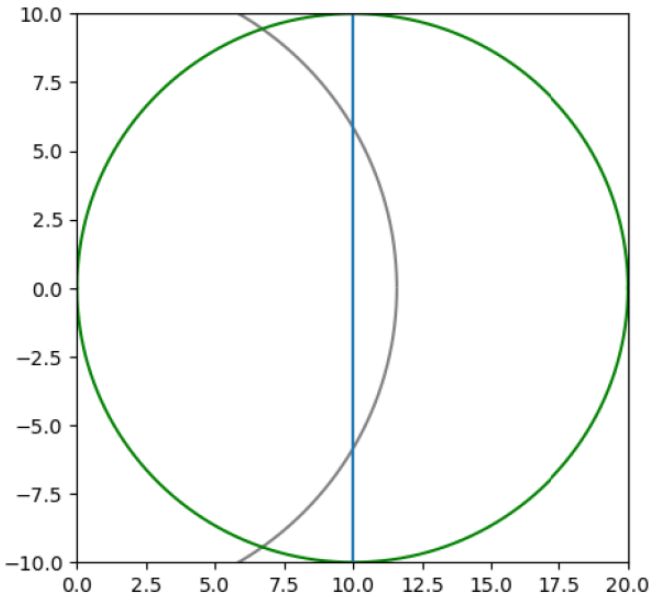


Figure 1.5: Priebeh funkcie na pravej strane rovnice 1.5. Asymptota je daná výrazom $L_{\text{asympt}} = \sqrt{2R^2}$

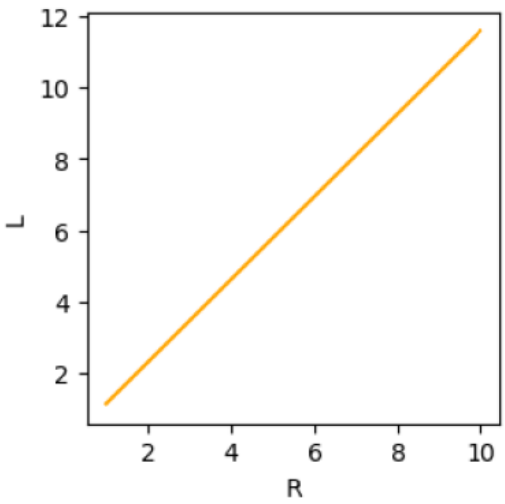


Figure 1.6: Graf závislosti R na L.

$R = 10$. Vykreslila som tento prípad do obrázka 1.7, pričom hodnoty polomerov jednotlivých kružníc sú vypísané v kóde v prílohe.

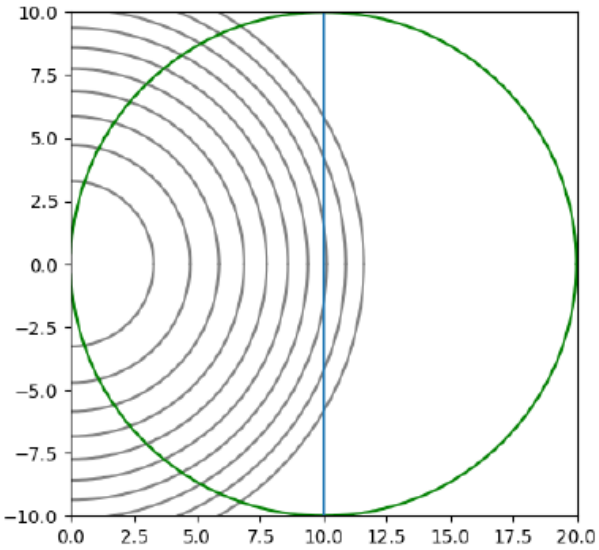


Figure 1.7: Graf závislosti R na L.

1.5 Iná aplikácia problému

Farmárčenie je len jedno z mnohých odvetví, kde sa tento problém dá aplikovať. Nakoľko je predmetom môjho štúdia astrofyzika, pokúsim sa predostrieť niekoľko možných scenárov, kde sa rozoberaný problém bude hodiť pri ich riešení.

1.5.1 Zbieranie materiálu

Prvým z takýchto scenárov je zbieranie vzoriek a extrahovanie materiálu protoplanetárneho disku hviezdy, prípadne akéhokoľvek iného hviezdneho objektu. Predstavme si, že sme schopní ťažiť materiál v okolí inej hviezdy. Vyslali sme vesmírnu loď, ktorá má na palube prístroj schopný zozbierať tento materiál a uložiť ho v útrobách lode. Tento prístroj bude fungovať ako "vysávač" s ramenami a nastavcom nastaviteľnej dĺžky, pričom na jeho konci bude pripevnená trubica, cez ktorú tento materiál potečie priamo do skladového priestoru lode. Nazvem ho Viktor (podľa robotického vysávača, ktorý máme doma).

Vesmírna loď zastaví na kraji disku - toto bude bod, z ktorého vyšleme Viktora, podobne ako keď je koza priviazaná na kraj trávnika. Skladové priestory lode môžu urdžať len určitý objem materiálu a ak predpokladáme konštantnú šírku a hustotu disku, vieme jednoducho určiť aký objem s Viktorom naberieme na jeden krát. Povedzme, že druhá vesmírna loď sa stará o vyprázdňovanie skladu, preto naša loď môže permanentne kotviť na jednom mieste. Môžme takto určiť postupnosť dĺžok spojovacieho tunelu tak, aby sme nabrali každý krát rovnaké množstvo materiálu, ktoré naplní sklad lode..

Samozrejme, ak by išlo o disk v okolí hviezdy, museli by sme s ťažbou prestať skôr, aby nedošlo k poškodeniu Viktora, preto by mal na sebe tepelné senzory, ktoré by ukončili jeho činnosť, ak by teplota príliš vzrástla.

1.5.2 Umývanie zrkadla

Ak by sme chceli umyť polovicu plochy zrkadla pomocou malého čistiaceho prístroja priviazaného na kraji tohto zrkadla, môžme využiť tento problém na určenie dĺžky lanka, ktoré bude tento prístroj spájať s konštrukciou tak, aby sa omylom nestratil v hlbokom vesmíre. Taktiež môžme takto určiť aj iné dĺžky, v závislosti na tom, koľko percent plochy zrkadla chceme mať stále vyčistených.

1.5.3 Trest na Enterprise

Disk vesmírnej lode Enterprise má kruhový tvar. Ak chce kapitán potrestať poddôstojníka, môže mu dať za úlohu, aby vyčistil polovicu tejto plochy.

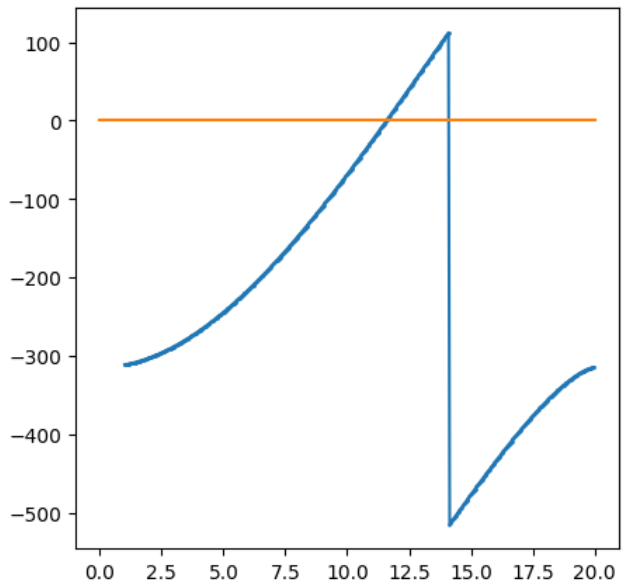
Poddôstojník musí byť vrámci bezpečnostných opatrení zakaždým pripevnený na kraji tohto disku, kde sa nachádzajú vstupné komory. Aby sa nemusel počas svojho trestu odopínať a opäť pripínať k inému úchytu, dá sa táto úloha využiť na určenie dĺžky úchytného lana, ktorým sa musí pripútať, aby na jeden krát splnil svoju úlohu.

In [1]:
```
# I load the libraries I need
import numpy as np
import matplotlib.pyplot as plt
import scipy.optimize as sco
```

In [2]:
```
# defining function that describes the relationship between R and L
def f(L):
    # point A:
    xa = L ** 2 / (2 * R)
    ya = L * np.sqrt(1 - (L / (2 * R))**2)

    # angles
    alpha1 = np.arctan(ya / (R - xa))
    alpha2 = np.arctan(ya / xa)

    return 2*alpha2 * L**2 - L * np.sqrt(4*R ** 2 - L**2) + R**2 * (2*alpha1 - np.pi)
```

In [3]:
```
# for a chosen value of R, I plot this function to see how the function behaves and where the function is 0
xval = np.linspace(1,50, 1000)
yval = np.zeros(1000)

R = 10

for i in range(1000):
    yval[i] = f(xval[i])

plt.figure(figsize=(5,5))

plt.plot(xval, yval, '-o', ms = 1)
plt.plot([0,20], [0,0])
#plt.ylim(-0.25,0.25)
plt.show()
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_7452\1090437321.py:5: RuntimeWarning: invalid value encountered in sqrt
  ya = L * np.sqrt(1 - (L / (2 * R))**2)
C:\Users\admin\AppData\Local\Temp\ipykernel_7452\1090437321.py:11: RuntimeWarning: invalid value encountered in sqr
t
  return 2*alpha2 * L**2 - L * np.sqrt(4*R ** 2 - L**2) + R**2 * (2*alpha1 - np.pi)
```



Asymptote of the function stems from denominator of alpha1 parameter, I need to take the interval in such way, so I won't accidentally go over the asymptote.

In [4]:
```
# x value of asymptote
asympt = np.sqrt(2 * R**2)
asympt
```

Out[4]:
```
14.142135623730951
```

In [5]:
```
# for the chosen value of R, I calculate the corresponding value of L using scipy function for bisection of the inte
result, info = sco.bisect(f, 0 + 0.1, asympt - 0.1, full_output=True)
result
```

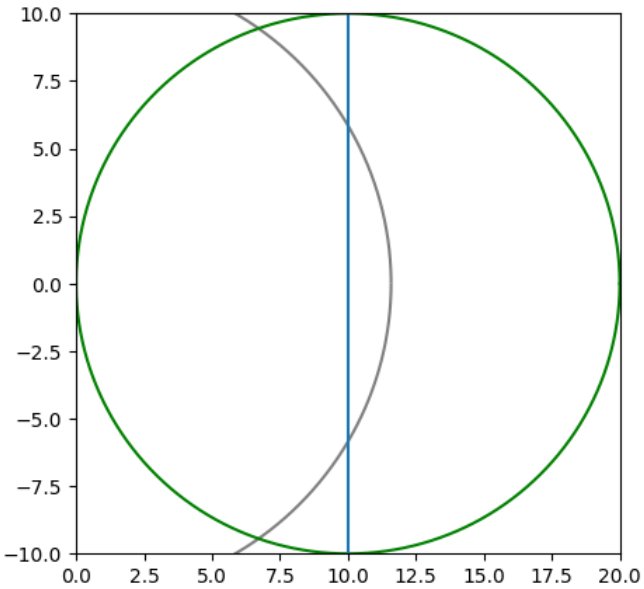
Out[5]:
```
11.58728473018217
```

In [6]:
```
info
```

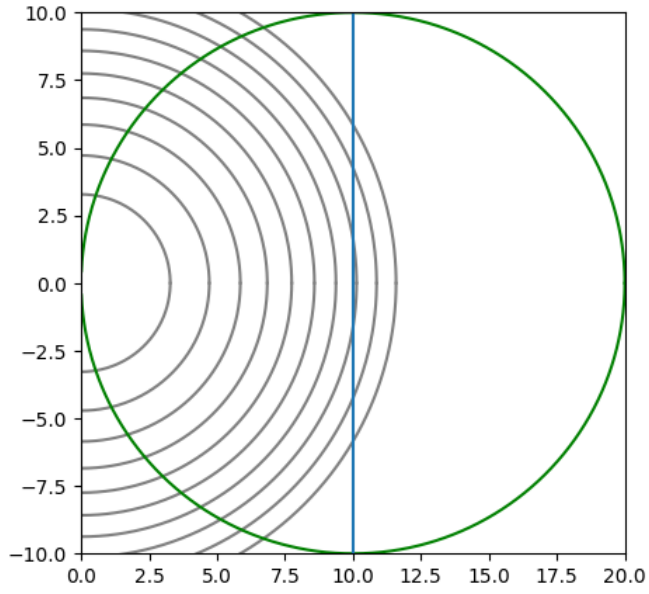
Out[6]:
```
      converged: True
           flag: 'converged'
 function_calls: 45
     iterations: 43
           root: 11.58728473018217
```

In [7]:
```
# for better visualisation, I plot the problem in a graph
x = np.linspace(0, 20, 10000)
y1 = np.zeros(10000) # values of y for the circle, that the goat can encompass
y2 = np.zeros(10000) # values of y for the grass circle

R = 10

for i in range(len(x)):
    y1[i] = np.sqrt(result**2 - x[i]**2)
    y2[i] = np.sqrt(R**2 - (x[i] - R)**2)

plt.figure(figsize=(5,5))

plt.plot(x,y1, color = 'grey')
plt.plot(x,-y1, color = 'grey')
plt.plot(x, y2, color = 'green')
plt.plot(x,-y2, color = 'green')
plt.plot([10, 10], [-R, R]) #
plt.xlim(0,20)
plt.ylim(-R, R)
plt.show()
```

```
C:\Users\admin\AppData\Local\Temp\ipykernel_7452\3618344343.py:9: RuntimeWarning: invalid value encountered in sqrt
  y1[i] = np.sqrt(result**2 - x[i]**2)
```

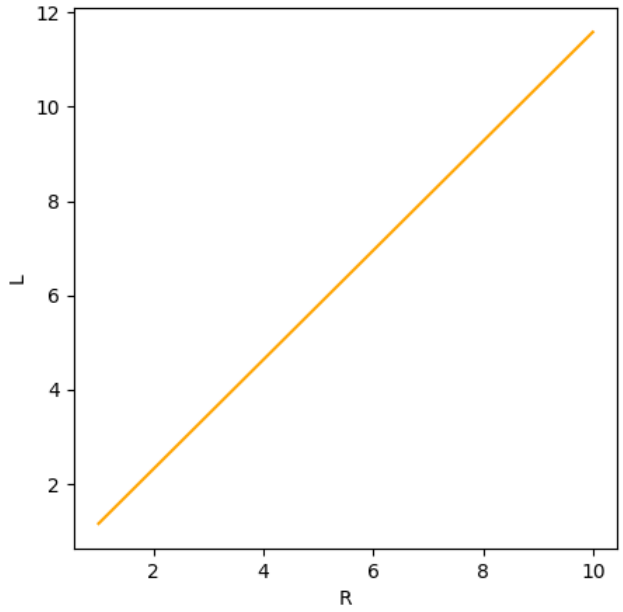


In [8]:
```
# I will calculate values of L, so the goat can eat the same ammout of grass in every step
plt.figure(figsize=(5,5))

N = 10 # number of steps
R = 10 # the radius of grass circle

result = np.zeros(N + 1)

# function that incomporates the individual steps
def f2(L):
    # point A:
    xa = L ** 2 / (2 * R)
    ya = L * np.sqrt(1 - (L / (2 * R))**2)

    # angles
    alpha1 = np.arctan(ya / (R - xa))
    alpha2 = np.arctan(ya / xa)

    return 2*alpha2 * L**2 - L * np.sqrt(4*R ** 2 - L**2) + R**2 * (2*alpha1 - np.pi / (N / i))

# calculating values of L for individual steps
for i in range(1, N + 1):
    result[i] = sco.bisect(f2, 0 + 0.1, asympt - 0.1)

    x = np.linspace(0, result[i], 10000)
    y1 = np.zeros(10000)

    # visualising the individual circles the goat will make
    for j in range(len(x)):
        y1[j] = np.sqrt(result[i]**2 - x[j]**2)

    plt.plot(x,y1, color = 'grey')
    plt.plot(x,-y1, color = 'grey')

# visualising the grass circle
x = np.linspace(0.01,20, 10000)
y2 = np.zeros(10000)

for j in range(len(x)):
    y2[j] = np.sqrt(R**2 - (x[j] - R)**2)

plt.plot(x, y2, color = 'green')
plt.plot(x,-y2, color = 'green')
plt.plot([10, 10], [-R, R])
plt.xlim(0,20)
plt.ylim(-R, R)
plt.show()

print(result)
```




```
[ 0.          3.27871517  4.71571466  5.85666793  6.8482563   7.74752946
  8.584046    9.3757074  10.13456573 10.86944062 11.58728473]
```

In [9]:
```
# I will calculate and plot the relationship between values of R and L
R_val = np.linspace(1, 10, 1000)
L_val = np.zeros(1000)

for i in range(len(R_val)):
    R = R_val[i]
    asympt = np.sqrt(2 * R**2)

    L_val[i] = sco.bisect(f, 0 + 0.1, asympt - 0.1)

plt.figure(figsize=(5,5))

plt.plot(R_val, L_val, color = 'orange')
plt.xlabel('R')
plt.ylabel('L')
plt.show()
```



In [10]:
```
# I fit the values with linear fit and derive the parameters of the line

def g(x, a, b):
    return a * x + b

param, pcov = sco.curve_fit(g, R_val, L_val)
print('a = {:.8f}, b = {:.8f}'.format(param[0], param[1]))
```

```
a = 1.15872847, b = 0.00000000
```

In [ ]: