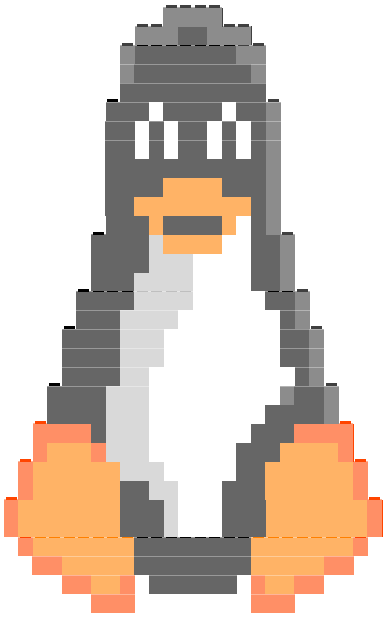
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΕΦΑΡΜΟΣΜΕΝΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΦΥΣ 140 Εισαγωγή στην Επιστημονική Χρήση Υπολογιστών
Χειμερινό Εξάμηνο 2023

Φώτης Πτωχός και Αλέξανδρος Αττίκης

Φροντιστήριο 11

21 Νοεμβρίου 2023
15:00 - 17:00



Φροντιστήριο 11

Παράδειγμα 1 Παράδειγμα εύρεσης ακροτάτων της συνάρτησης $f(x) = x^3 - 6x^2 - x + 1$ στο διάστημα $x = [0, 5]$. Η ακριβής λύση είναι $f_{min} = -10.16230$ για $x = 4.16025$:

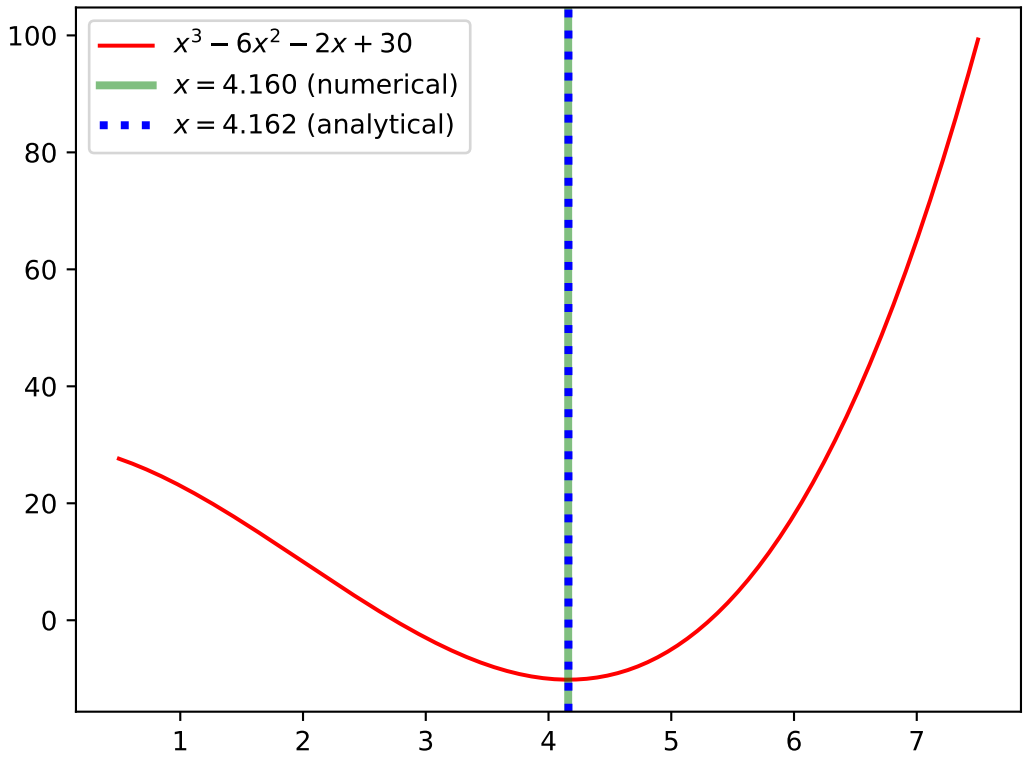
tutorial11/ex3.py

```
#!/usr/bin/python3
import numpy as np
from random import randrange, seed, random
import matplotlib.pyplot as p
import sympy as s

seed(123)
x  = s.Symbol('x')
F  = x**3 - 6*x**2 - 2*x + 30
DF = s.diff(F, x)
f  = s.lambdify(x, F)
df = s.lambdify(x, DF)
N  = 1001
xMin  = +1
xMax  = +7
xList = []
yList = []

# Find exact solution with sympy
eqn = s.Eq(DF, 0)
# Find critical points (CPs) & keep only those within range [xMin, xMax]
cpList = [p.evalf() for p in s.solve(eqn, x) if p.evalf() >= xMin and p.evalf()
    <= xMax]

for i in range(N):

    # Get random number in interval
    x = xMin + (xMax-xMin)*random()

    y = f(x)
    xList.append(x)
    yList.append(y)

fMin  = min(yList)
fMax  = max(yList)
fMinX = xList[yList.index(min(yList))]
fMaxX = xList[yList.index(max(yList))]

print("=== Analytical solution:\n\tf(min) = %.5f at x= %.5f" % (f(cpList[0]),
   cpList[0] ) )
print("=== Numerical approximation:\n\tf(min) = %.5f at x= %.5f" % (fMin, fMinX
   ) )

# Plot our function and mark the evaluation minimum
X = np.arange(xMin-0.5, xMax+0.5, 0.001)
Y = [f(x) for x in X]
p.plot( X, Y, "r-", label=r"$%s$" % s.latex(F) )
```

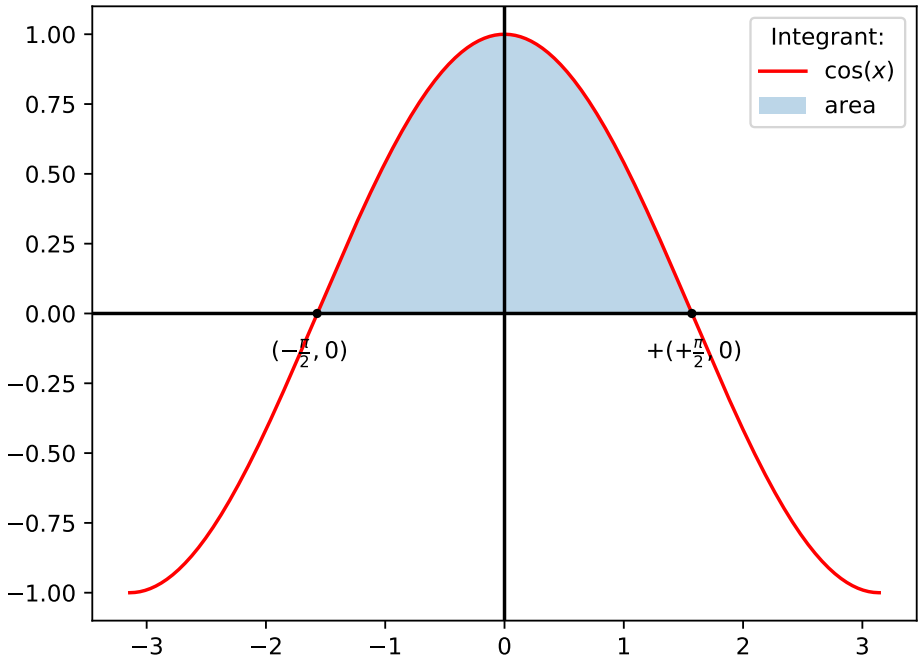
Παράδειγμα 1 συνεχίζεται...

```
45 p.axvline(cpList, linestyle="-" , color="g", alpha=0.5, lw=3, label=r"$x=%.3f$
       (numerical)" % (cpList[0]) )
46 p.axvline(fMinX , linestyle=":", color="b", alpha=1.0, lw=3, label=r"$x=%.3f$ (
       analytical)" % (fMinX) )
47 p.legend()
48 for ext in [".png", ".pdf"]:
49     p.savefig(__file__.split(".")[0] + ext)
50 p.show()
51 quit()
```

Αποτέλεσμα:



Παράδειγμα 2 Στο παρακάτω παράδειγμα δείχνουμε πως μπορούμε να υπολογίσουμε το ολοκλήρωμα $\int_{-\frac{\pi}{2}}^{+\frac{\pi}{2}} \cos(x)dx$ χρησιμοποιώντας την μέθοδο Monte Carlo μέσης τιμής.



tutorial11/ex2.py

```python
#!/usr/bin/python3
import numpy as np
import random as rndm

def f(x):
    return np.cos(x)

def intF(x):
    return np.sin(x)

def MCIntegrate(f, nMC, a, b):

    mySum = 0.0

    for i in range(nMC):

        # Get random number in interval [a, b]
        xRand = a + (b-a) * rndm.random()

        # Sum the value of the function for this random value of x
```

```
21          mySum+= f(xRand)
22
23      # Divide by the number of trials to get the mean
24      integral = (b-a) * mySum / nMC
25      return integral
26
27  def main():
28
29      rndm.seed(12345)
30      xMin = -np.pi/2
31      xMax = +np.pi/2
32
33      # Analytic solution of integral in range [xMin, xMax]
34      intAnalytic = intF(xMax) - intF(xMin)
35
36      # MC solution of integral in range [xMin, xMax]
37      nTries = 1000000
38      intMC  = MCIntegrate(f, nTries, xMin, xMax)
39      msg = "Integral of cos(x) in interval [%3.1f, %3.1f]:" % (xMin, xMax)
40      msg+= "\n\tMC Mean Value = %14.10f " % (intMC)
41      msg+= "\n\tAnalytic Solution = %14.10f"% (intAnalytic)
42      print(msg)
43
44  main()
45  quit()
```

Αποτέλεσμα:

```
Integral of cos(x) in interval [-1.6, 1.6]:
  MC Mean Value =   1.9989593864
  Analytic Solution =   2.0000000000
```

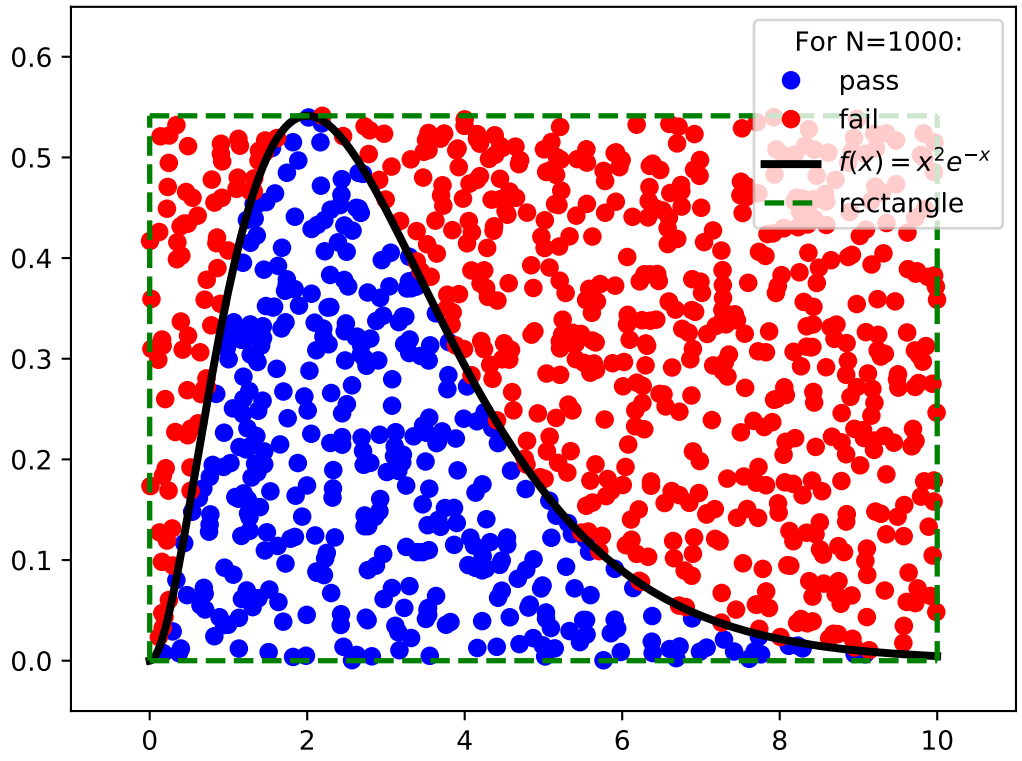
Παράδειγμα 3 Παράδειγμα υπολογισμού του εμβαδού της συνάρτησης $f(x) = x^2 e^{-x}$ (ολοκλήρωμα) στο διάστημα $x = [0, 10]$ με τη μέθοδο δειγματοληψίας.

tutorial11/ex4.py

```
#!/usr/bin/python3
import random as rndm
import matplotlib.pyplot as p
import sympy as s
import numpy as np

rndm.seed(123456)
x  = s.Symbol('x')
F  = x*x * s.exp(-x)
DF = s.diff(F, x)
f  = s.lambdify(x, F)
df = s.lambdify(x, DF)

# Find maximum  by solving df/dx = 0 and finding x. Then evaluate f at this x.
eq     = s.Eq(DF, 0)
xDF0   = [float(x) for x in s.solve(eq, x)]
yDF0   = [float(f(x)) for x in xDF0]
yMax   =  max(yDF0) #  = 0.55
yMin   =  0.0
xMin   =  0.0
xMax   = 10.0
nPass  =  0
nTries = 1000
x1List = []
x2List = []
y1List = []
y2List = []


for i in range(nTries):

    # Get random number in interval [0, 1]
    x = rndm.random()
    # Transform random number to interval [xMin, xMax]
    x = xMin + (xMax-xMin)*x

    # Get random number in range [0, yMax]
    y = yMin + yMax * rndm.random()

    # Our random point lies within the width 'x' (by construction) and height '
   y' of rectangle
    if f(x) > y:
        nPass+=1
        x1List.append(x)
        y1List.append(y)
    else:
        x2List.append(x)
        y2List.append(y)

```

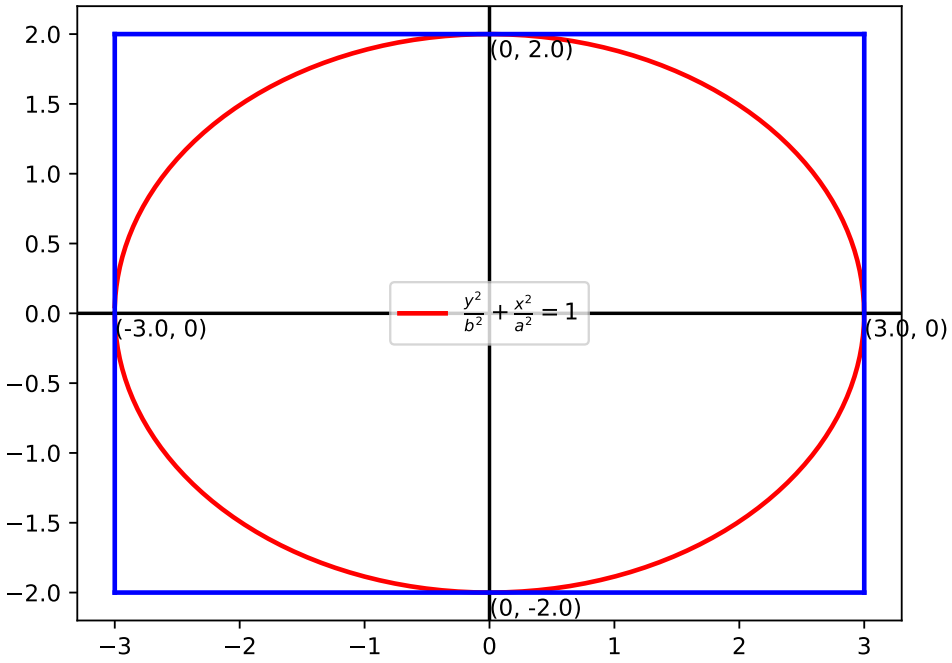
```
49
50 dy = yMax-yMin
51 dx = xMax-xMin
52 rArea = dy * dx
53 fArea = rArea* (nPass/nTries)
54 print("The area of the rectangle is:\n\tA(rectangle) = %.3f\nThe area of the
       function is:\n\tA(f) = %.3f" % (rArea, fArea) )
55
56 X = np.arange(xMin, xMax, 0.001)
57 Y = [f(x) for x in X]
58
59 # Plot the sampling points enclosed by function
60 p.plot( x1List, y1List, "bo", label="pass")
61
62 # Plot the sampling points not enclosed by function
63 p.plot( x2List, y2List, "ro", label="fail")
64
65 # Plot the function curve
66 p.plot( X, Y, "k-", lw=3, label=r"$f(x)=%s$" % s.latex(F) )
67
68 # Draw the rectangle enclosing the curve
69 p.plot( [xMin, xMax], [yMax, yMax], "g--", lw=2, label="rectangle")
70 p.plot( [xMin, xMin], [0, yMax], "g--", lw=2)
71 p.plot( [xMax, xMax], [0, yMax], "g--", lw=2)
72 p.plot( [xMin, xMax], [0, 0], "g--", lw=2)
73
74 # Set axes limits
75 p.ylim(-0.05, yMax*1.2)
76 p.xlim(xMin-1, xMax+1)
77
78 #p.axvline(fMinX, linestyle=":", color="k", label=r"$x=%s$" % (fMinX) )
79 p.legend(title="For N=%s:" % nTries)
80 for ext in [".png", ".pdf"]:
81     p.savefig(__file__.split(".")[0] + ext)
82 p.show()
83
84 quit()
```

Αποτέλεσμα:



Για να περικλείσουμε τη συνάρτηση στο διάστημα της ολοκλήρωσης απαιτείται η εύρεση της μέγιστης ή ελάχιστης τιμής της συνάρτησής μας. Για να το επιτύχουμε αυτό χρησιμοποιούμε την *sympy* για να βρούμε την παράγωγο της συνάρτησης, να λύσουμε την εξίσωση $\frac{df}{fx} = 0$, και έπειτα να υπολογίσουμε τη συνάρτηση στο σημείο του ακρώτατου.

Παράδειγμα 4 Μια έλλειψη ορίζεται από την εξίσωση $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$, όπου a και b ο μεγάλος και μικρός άξονάς της. Θα χρησιμοποιήσουμε τη μέθοδο του Monte Carlo με $N = 10$, 100, 1000, $10\,000$, και $100\,000$ προσπάθειες για να βρούμε το εμβαδό της έλλειψης A_{MC} του παρακάτω σχήματος που έχει $a = 3$ και $b = 2$. Η αναλυτική τιμή είναι το εμβαδό της έλλειψης υπολογιζόμενο θεωρητικά και που είναι ίσο με $A_{\text{ελ}} = \pi \cdot a \cdot b$. Τέλος, μπορούμε να υπολογίσουμε τη τιμή του π εξισώνοντας τα δύο εμβαδά $A_{\text{MC}} = A_{\text{ελ}}$.



tutorial11/ex1.py

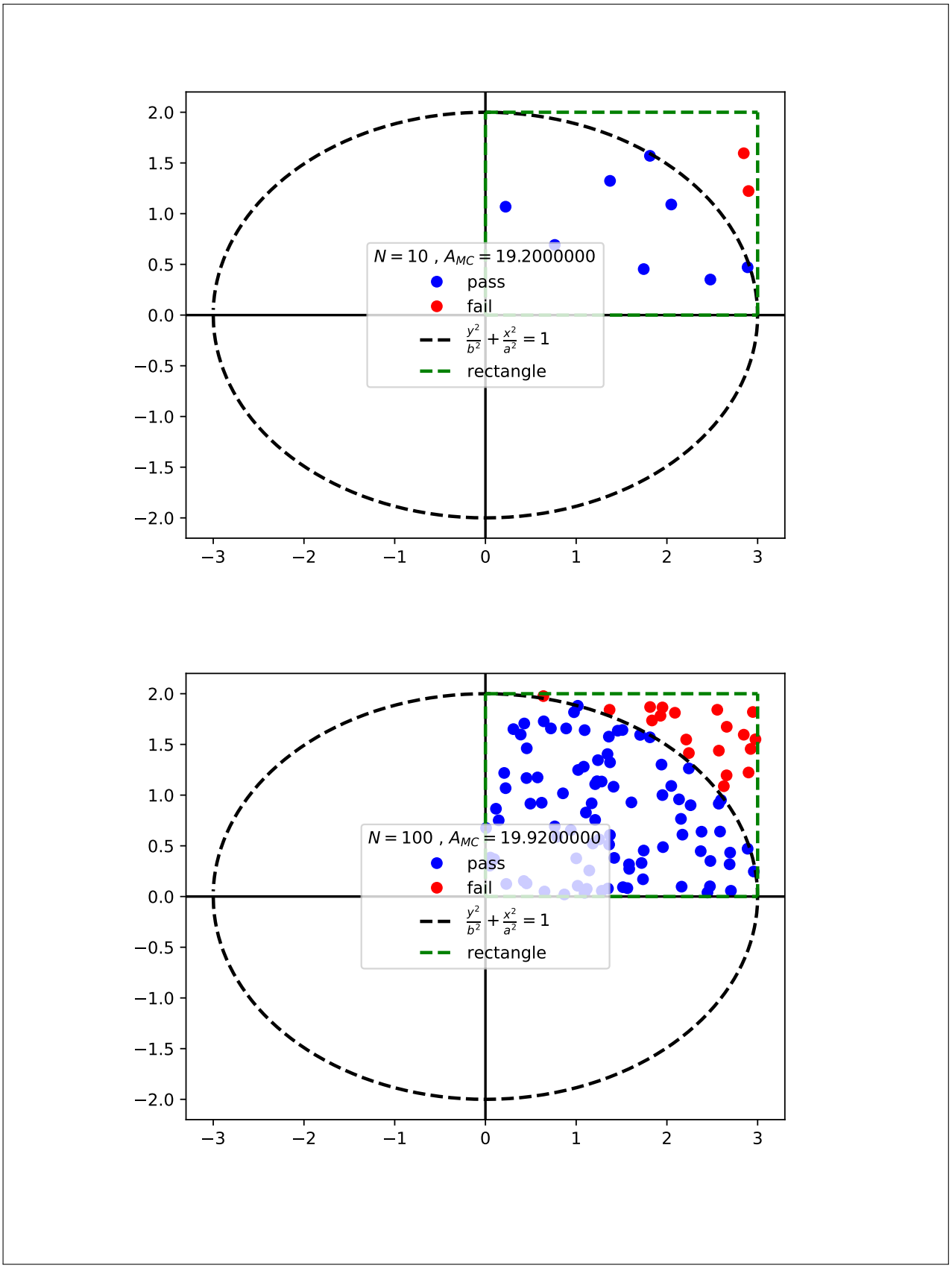
```python
#!/usr/bin/python3
import numpy as np
import random as rndm
import matplotlib.pyplot as p
import sympy as s

# Define ellipse dimensions
a = 3.0  # major axis
b = 2.0  # minor axis
r = 1.0  # radius
nList  = [10**i for i in range(1, 6, 1)]
x1List = []
y1List = []
x2List = []
y2List = []
rndm.seed(1234567)

```

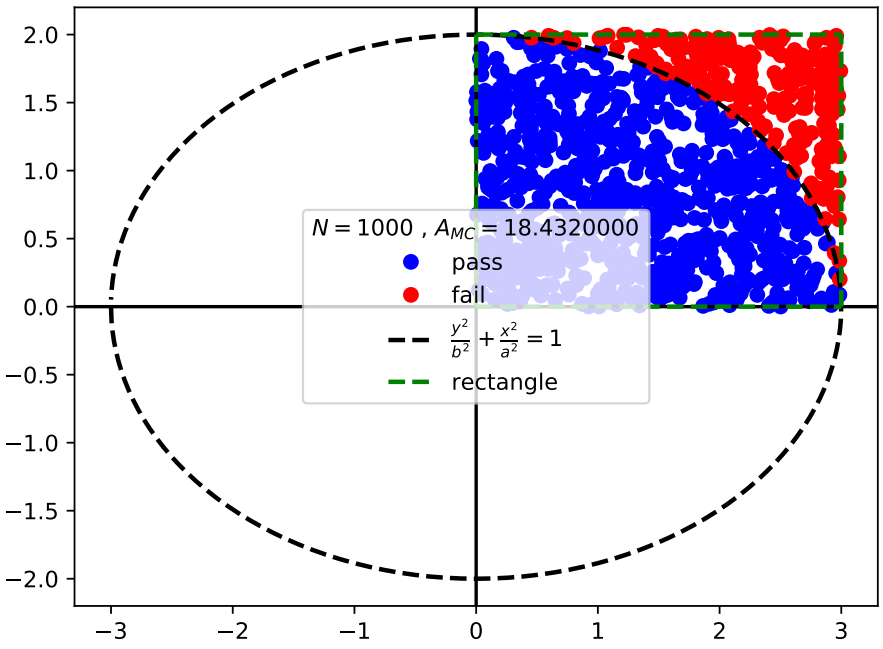
```
18
19 # For plotting the ellipse use sympy
20 x, y, aa, bb = s.symbols('x y a b')
21 # Define the equation of the ellipse
22 ellipseEq = s.Eq(x**2 / aa**2 + y**2 / bb**2, 1)
23 # Solve for y in terms of x
24 ellipseY  = s.solve(ellipseEq, y)
25 # Lambdify the ellipse equation for numerical evaluation
26 ellipseF  = s.lambdify((x, aa, bb), ellipseY)
27
28
29 # Loop over various values of experiments N
30 for i, N in enumerate(nList, 1):
31     nPass = 0
32     if N == 10:
33         print("%11s %14s %14s %12s %15s" % ("N","A (MC)", "A (Theory)", "dA", "
   Pi Estimate"))
34
35     # Loop over [0, N] to evaluate the random numbers for each experiment
36     for itry in range(N):
37
38         # Generate random number for x in range [0, a]
39         xRand = a * rndm.random()
40
41         # Generate random number for y in range [0, b]
42         yRand = b * rndm.random()
43
44         # Evaluate radius of ellipse
45         rRand = (xRand/a)**2 + (yRand/b)**2
46
47         # Check if
48         if rRand <= r**2:
49             nPass += 1
50             x1List.append(xRand)
51             y1List.append(yRand)
52         else:
53             x2List.append(xRand)
54             y2List.append(yRand)
55
56     # Evaluate integral estimate for given N
57     integral   = nPass / N
58     areaMC     = 4 * integral * a * b
59     areaTheory = np.pi * a * b
60     piEstimate = 4*integral
61     print( "%10i %14.6f %14.6f %14.7f %13.7f"% (N, areaMC, areaTheory, abs(
   areaMC-areaTheory), 4*integral) )
62     # areaMC = areaTheory
63     # => 4 * integral * a * b = pi * a * b
64     # => pi = 4 * integral
65
66     # Plot the sampling points enclosed/not enclosed by function
67     p.plot( x1List, y1List, "bo", label="pass")
68     p.plot( x2List, y2List, "ro", label="fail")
```

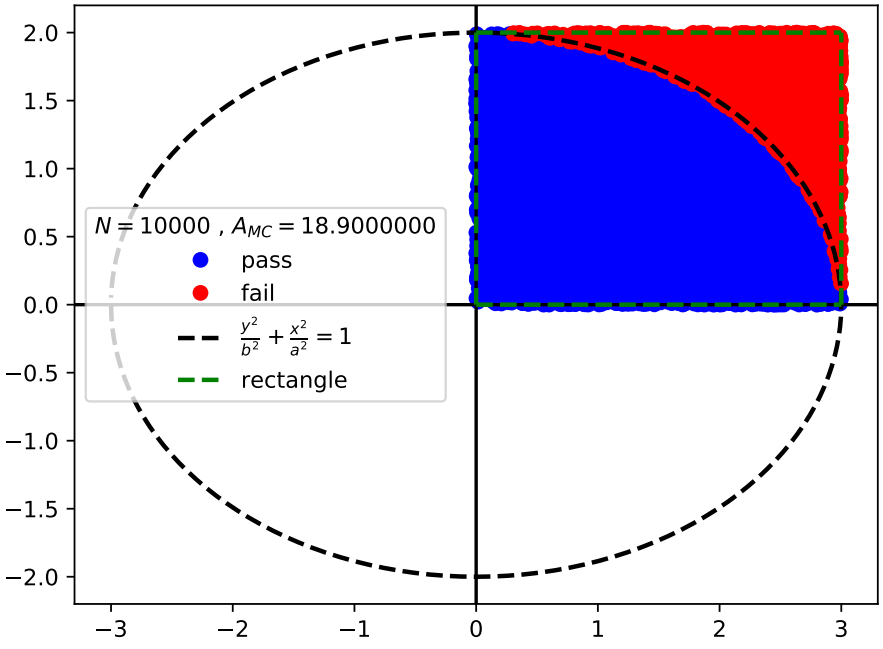
```
69
70      # Set axes limits & draw axes lines
71      p.ylim(-b*1.1, +b*1.1)
72      p.xlim(-a*1.1, +a*1.1)
73      p.axhline(0, color="k")
74      p.axvline(0, color="k")
75
76      # Plot the function curve
77      X = list(np.arange(-a, +a, 0.001))
78      # Y = [ellipseF(x, a, b) for x in X] # this causes duplicate label!
79      Y1 = [ellipseF(x, a, b)[0] for x in X]
80      Y2 = [ellipseF(x, a, b)[1] for x in X]
81      Y  = []
82      Y.extend(Y1)
83      Y.extend(Y2)
84      X.extend(reversed(X))
85      p.plot(X, Y, "k--", lw=2, label=r"$%s$" % s.latex(ellipseEq) )
86
87      # Draw the rectangle enclosing the curve
88      p.plot( [0, a], [b, b], "g--", lw=2, label="rectangle")
89      p.plot( [0, a], [0, 0], "g--", lw=2)
90      p.plot( [0, 0], [0, b], "g--", lw=2)
91      p.plot( [a, a], [0, b], "g--", lw=2)
92
93      # Draw the legend
94      p.legend(title=r"$N=%d$ , $A_{MC}=%.7f$" % (N, areaMC) )
95      for ext in [".png", ".pdf"]:
96          p.savefig(__file__.split(".")[0] + "_N" + str(N) + ext)
97      p.show()
98
99  quit()
```

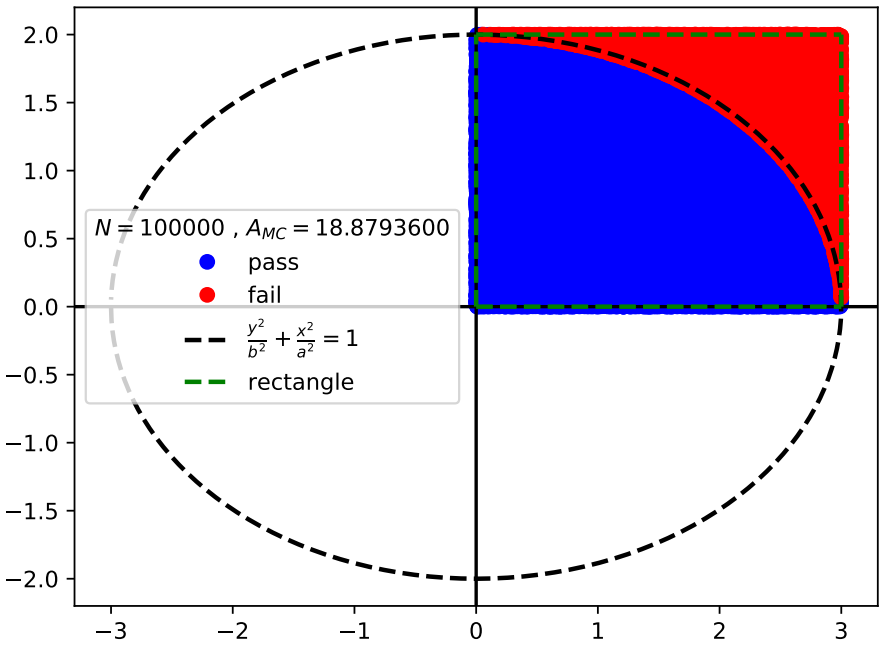
Αποτέλεσμα:

N	A (MC)	A (Theory)	dA	Pi Estimate
10	19.200000	18.849556	0.3504441	3.2000000
100	19.920000	18.849556	1.0704441	3.3200000
1000	18.432000	18.849556	0.4175559	3.0720000
10000	18.900000	18.849556	0.0504441	3.1500000
100000	18.879360	18.849556	0.0298041	3.1465600


$N = 10$, $A_{MC} = 19.2000000$
pass
fail
$\frac{y^2}{b^2} + \frac{x^2}{a^2} = 1$
rectangle
$N = 100$, $A_{MC} = 19.9200000$
pass
fail
$\frac{y^2}{b^2} + \frac{x^2}{a^2} = 1$
rectangle


$N = 1000$, $A_{MC} = 18.4320000$
pass
fail
$\frac{y^2}{b^2} + \frac{x^2}{a^2} = 1$
rectangle


$N = 10000$, $A_{MC} = 18.9000000$
pass
fail
$\frac{y^2}{b^2} + \frac{x^2}{a^2} = 1$
rectangle



Ορίζουμε ένα παραλληλόγραμμο με μήκος πλευράς $a = 3$ και $b = 2$, το οποίο περικλείει το ένα τέταρτο της έλλειψης. Το εμβαδόν του παραλληλογράμμου αυτού είναι $A_{\text{πα}} = a \cdot b$. Επιλέγουμε τυχαία σημεία x στο διάστημα $x = [0, 3]$ και τυχαία σημεία y στο διάστημα $y = [0, 2]$ τα οποία ζητούμε να ικανοποιούν την εξίσωση της έλλειψης:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1. \tag{1}$$

Το εμβαδό της έλλειψης θα είναι τότε:

$$A_{\text{MC}} = a \cdot b \cdot \frac{N_{\text{ελ}}}{N} \tag{2}$$

όπου N είναι ο συνολικός αριθμός τυχαίων σημείων (x, y) και $N_{\text{ελ}}$ είναι ο αριθμός των τυχαίων σημείων (x, y) που ικανοποιούν την Εξ. (1) και άρα βρίσκονται εντός της έλλειψης.