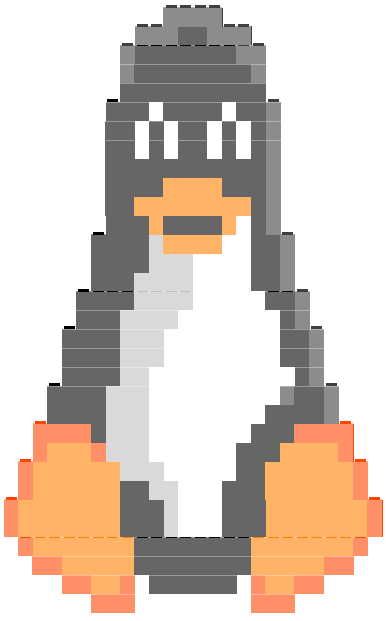
ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΚΑΙ ΕΦΑΡΜΟΣΜΕΝΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΦΥΣΙΚΗΣ

ΦΥΣ 140 Εισαγωγή στην Επιστημονική Χρήση Υπολογιστών
Χειμερινό Εξάμηνο 2023

Φώτης Πτωχός και Αλέξανδρος Αττίκης

Φροντιστήριο 8

31 Οκτωβρίου 2023
15:00 - 17:00



Φροντιστήριο 8

Παράδειγμα 1 Γραφική αναπαράσταση της κατανομής Gauss για ευρύ φάσμα τιμών μέσης τιμής μ και τυπικής απόκλισης σ:

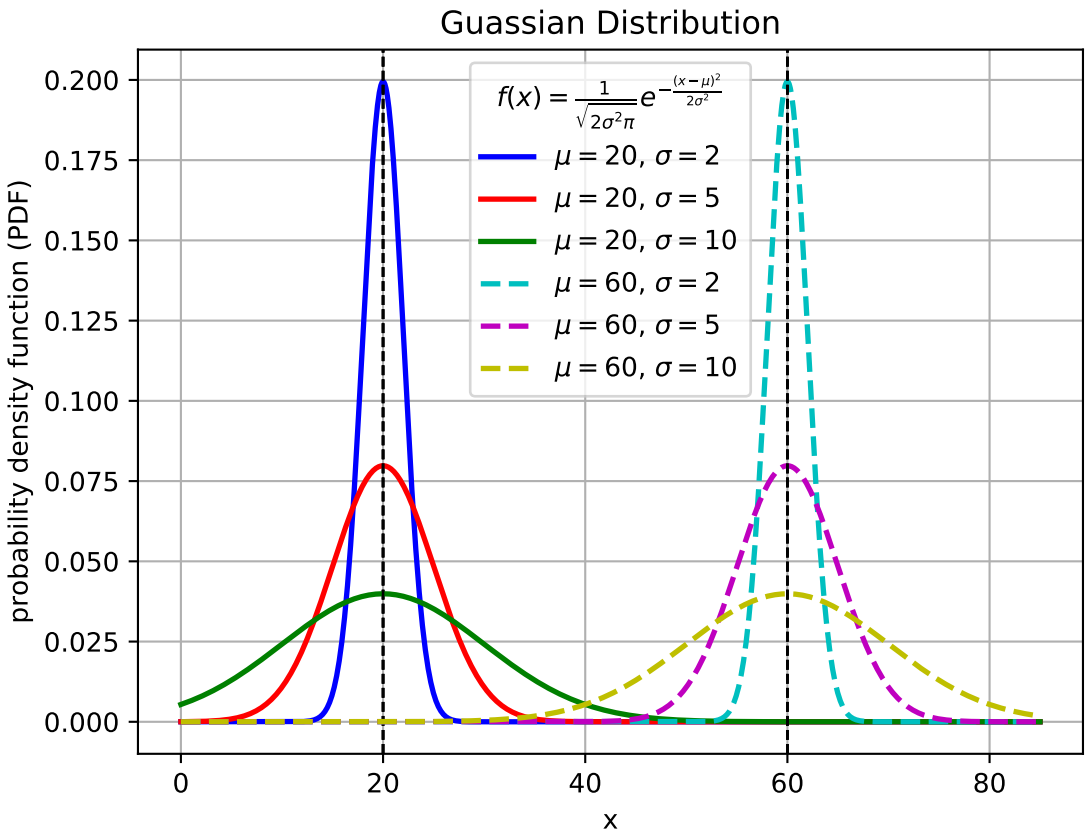
tutorial8/ex1.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex1.py
   python3 ex1.py
   script -q ex1.log python3 -i ex1.py


DESCRIPTION:
Examples of various Gaussian distributions
'''
import numpy as np
import matplotlib.pyplot as plt
import random as rndm

def getGauss(x, mu, sigma):
    return (1/(np.sqrt(2*np.pi)*sigma))*np.exp(-0.5*( (x-mu)/sigma)**2)

coList  = ['b', 'r', 'g', 'c', 'm', 'y', 'k']
stList  = ['-', '-', '-', '--', '--', '--']
index   = -1
means   = [20, 60]
sigmas  = [2, 5, 10]
xList   = np.arange(0.0, 85.0, 1/100)

for mean in means:
    for sigma in sigmas:
        index+= 1

        # Create a list with Gaussian function values for given parameters (mean, sigma)
        yList = [getGauss(v, mean, sigma) for v in xList]

        # Enable the grid?
        plt.grid(True)

        # Draw the plot on the canvas
        plt.plot(xList, yList, coList[index], ls=stList[index], lw=2, label=r'$\mu=%.0f$, $\sigma=%.0f$' % (mean, sigma) )

        # Add line at x=mean?
        plt.axvline(mean, color='k', linestyle="--", lw=1)

# Add information on canvas (labels, title)
plt.title("Guassian Distribution", size=12)
plt.xlabel("x")
plt.ylabel("probability density function (PDF)")
```

Παράδειγμα 1 συνεχίζεται...

```
46 plt.legend(title=r"$f(x) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{\left(x-\mu\
       right)^2}{2\sigma^2}}$")
47
48 # Save BEFORE you show the figure interactively
49 for ext in [".png", ".pdf"]:
50     plt.savefig("ex1" + ext)
51 plt.show()
52 quit()
```

Αποτέλεσμα:

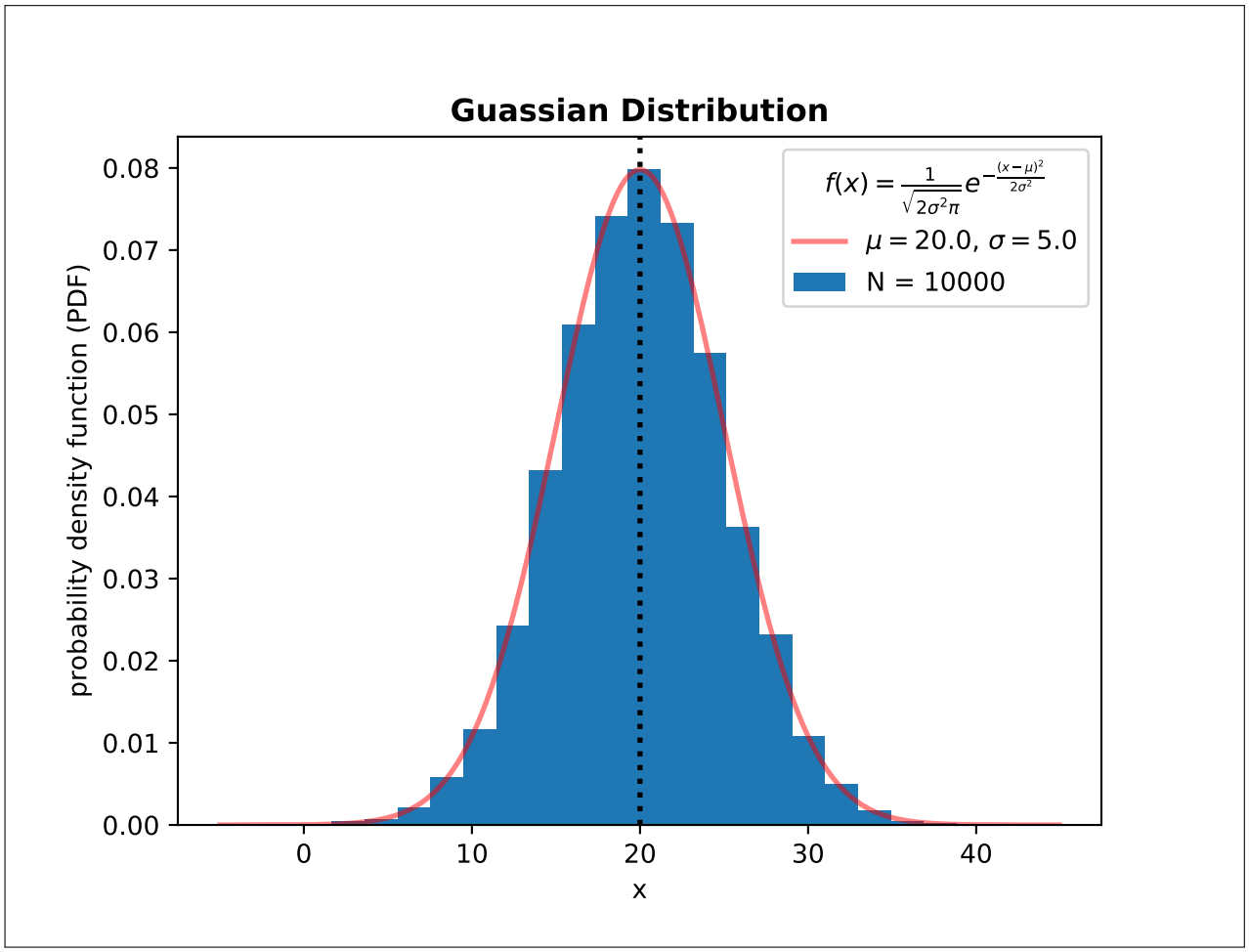


Παράδειγμα 2 Κατασκευή ιστογράμματος με *matplotlib* τυχαίων αριθμών καταναμημένων σύμφωνα με την κατανομή Gauss με N=10 000, μέση τιμή $\mu = 20$ και τυπική απόκλιση $\sigma = 5$:

tutorial8/ex2-simple.py

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

def myGauss(x, mu, sigma):
    return (1/(np.sqrt(2*np.pi)*sigma))*np.exp(-0.5*( (x-mu)/sigma)**2)

# Create histogram entries using a random number generator (sampling Gaussian
    with mean and sigma)
N     = 10000
mean  = 20
sigma = 5
hList = [rndm.gauss(mean, sigma) for i in range(N+1)]

# Plot the Gaussian function using appropriate lists
xMin  = mean-5*sigma
xMax  = mean+5*sigma
nBins = 20
xList = np.arange(xMin, xMax, 1/nBins)
yList = [myGauss(x, mean, sigma) for x in xList]

# Create the canvas and paint the plots
plt.figure()
content, binvalues, interm = plt.hist(hList, bins=nBins, density=True, label="N
    = %d" % (N) )
plt.plot(xList, yList, "r-", lw=2, alpha=0.5, label=r'$\mu=%.1f$, $\sigma=%.1f$
    ' % (mean, sigma) )
plt.title("Guassian Distribution", size=12, weight='bold')
plt.grid(False)
plt.xlabel("x")
plt.ylabel("probability density function (PDF)")
plt.legend(title=r"$f(x) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{\left(x-\mu\
    right)^2}{2\sigma^2}}$")
plt.axvline(mean, color='black', linestyle=":", lw=2)
for ext in [".png", ".pdf"]:
    plt.savefig("ex2-simple" + ext)
plt.show()
quit()
```

Αποτέλεσμα:


Guassian Distribution
probability density function (PDF)
x
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0
10
20
30
40
$f(x) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
$\mu = 20.0, \sigma = 5.0$
N = 10000

Παράδειγμα 3 Κατασκευή ιστογράμματος με *matplotlib* με τη δημιουργία 1 000 τυχαίων αριθμών καταναμημένων σύμφωνα με την κατανομή Gauss με μέση τιμή $\mu = 60$ και τυπική απόκλιση $\sigma = 5$, με επιπλέον στυλιστικές προσαρμογές. Επιπλέον, ορίζουμε διάφορες συναρτήσεις για να υπολογίσουμε τον μέσο όρο, την τυπική απόκλιση, και το εμβαδόν του ιστογράμματος μας:

tutorial8/ex2.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex2.py
   python3 ex2.py
   script -q ex2.log python3 -i ex2.py


DESCRIPTION:
Example usage of matplotlib pyplot library


LINKS:
https://matplotlib.org/2.2.5/api/_as_gen/matplotlib.pyplot.colors.html
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.fill_between.html
https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html
https://www.color-hex.com
https://www.askpython.com/python/examples/mean-and-standard-deviation-python
'''
import numpy as np
import matplotlib.pyplot as plt
import random as rndm

def myGauss(x, mu, sigma):
    return (1/(np.sqrt(2*np.pi)*sigma))*np.exp(-0.5*( (x-mu)/sigma)**2)

def getMean1(content, binvalues):
    '''
    Parameters:
    content ....: List of bin content (frequency) values.
    binvalues ..: List of bin edge values.

    \return the mean of the histogram
    '''
    # Use binvalues[:-1] to allow us to work with the bin edges of all but the
    last bin in the histogram.
    # binvalues[:-1] = make a copy of binvalues list and remove last element
    mean = sum(content * binvalues[:-1]) / sum(content)
    return mean

def getMean2(content, binvalues):
    '''
    Parameters:
    content ....: List of bin content (frequency) values.
    binvalues ..: List of bin edge values.
```

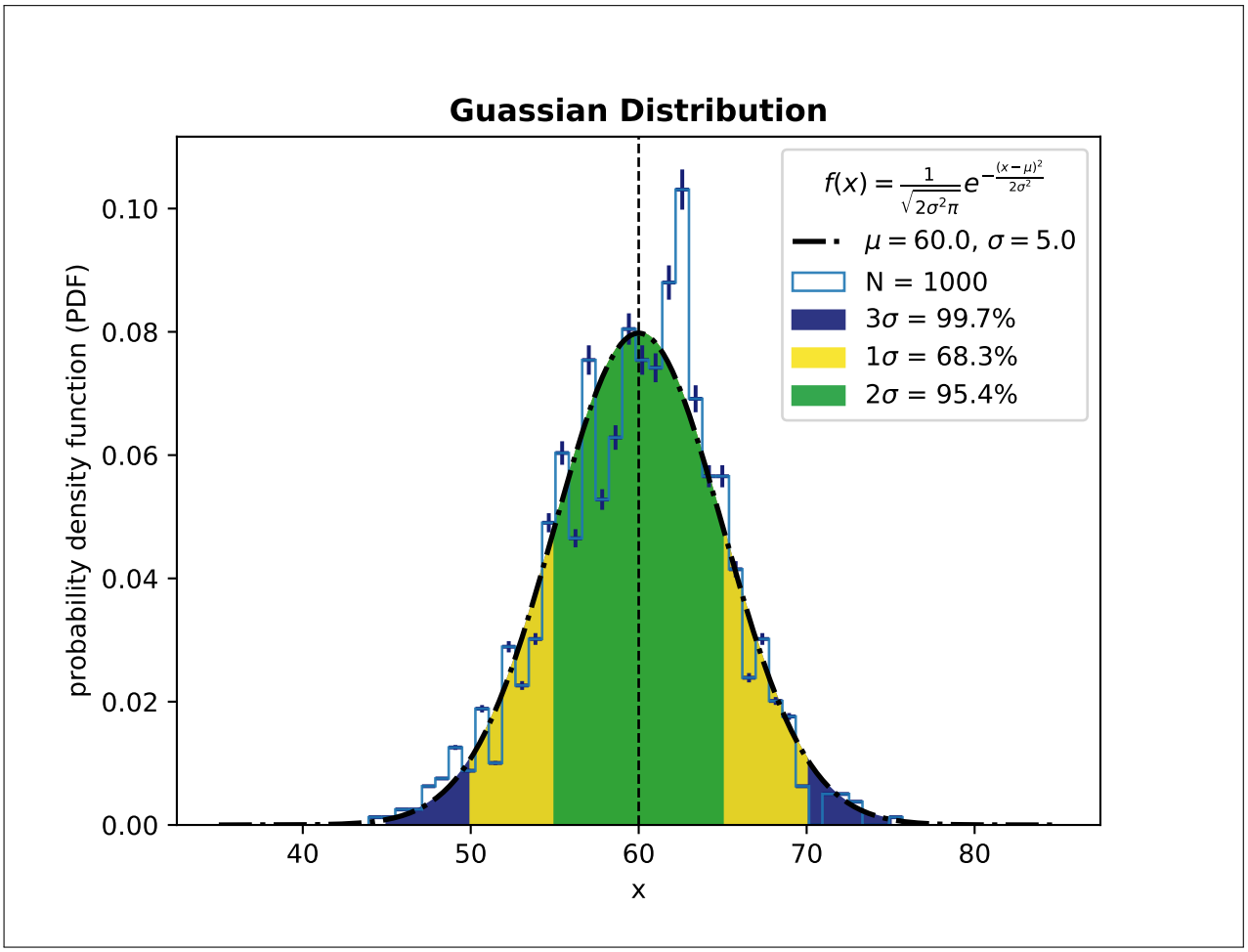
Παράδειγμα 3 συνεχίζεται...

```
45
46      \return the mean of the histogram
47      Sum_{i}^{all bins} x_{i} F_{i} = x_{i} * (f_{i} * dx_{i})
48
49      where:
50
51      f_{i} = F_{i}/dx_{i}      [prob. density]
52      F_{i} = f_{i} * dx_{i}    [frequency]
53      '''
54      xMid  = [0.5 * (binvalues[i-1] + binvalues[i]) for i in range(1, len(
        binvalues))]
55      mean  = 0.0
56
57      ### NB: len(binvalues) = len(content) + 1 !!!
58      ### This is because binvalues is a lsit containing the bin edges!
59      for i in range(0, len(content) ):
60          dx= binvalues[i+1] - binvalues[i] # constant bin width
61          x = xMid[i]
62          f = content[i]
63          mean += x * (f * dx)
64      return mean
65
66  def getIntegral(fList, binvalues):
67      '''
68      Parameters:
69      content ....: List of bin content (frequency) values
70      binvalues ..: List of bin edge values.
71
72      Sum_{i}^{all bins} f_{i} dx_{i}
73      '''
74      dx = binvalues[1]-binvalues[0] # assume constant bin width
75      integral = 0.0
76      for f in fList:
77          integral += f * dx
78      return integral
79
80  def getStDev(data):
81      '''
82      Easier to use the sum() built-in python function!
83      '''
84      return np.sqrt( getVariance(data) )
85
86  def getVariance(data):
87      '''
88      Easier to use the sum() built-in python function
89      with list comprehension.
90      '''
91      N = len(data)
92      mean = sum(data) / N
93      return sum((x - mean) ** 2 for x in data) / (N)
94
95  N, mean, sigma = input("Enter histogram population and Gaussian parameters (N,
      mean,standDev): ").strip().split(",")
```

```
96  N     = int(N)        # e.g. 50000
97  mean  = float(mean)   # e.g. 60
98  sigma = float(sigma) # e.g. 5
99
100 # Variable definition
101 xMin = mean-5*sigma
102 xMax = mean+5*sigma
103 kBrazilGold  = "#f8e31c"
104 kBrazilGreen = "#1d9e3a"
105 kBrazilBlue  = "#161f75"
106
107 # Create N histogram entries using a random number generator sampling from a
        Gaussian with custom mean and sigma
108 xList = [rndm.gauss(mean, sigma) for i in range(N+1)]
109
110 # Create the canvas
111 plt.figure()
112
113 # Create histogram (content = freq. density, binvalues= x-values in the defined
        intervals, list of intermediate calculations)
114 content, binvalues, interm = plt.hist(xList, bins=40, density=True, facecolor=
        kBrazilBlue, alpha=0.9, histtype='step', label="N = %d" % (N) )
115
116
117 # Plot the Gaussian function using appropriate lists
118 x = np.arange(xMin, xMax, 1/100)
119 y = [myGauss(v, mean, sigma) for v in x]
120 plt.plot(x, y, "k", ls="-.", lw=2, alpha=1.0, label=r'$\mu=%.1f$, $\sigma=%.1f$
        ' % (mean, sigma) )
121 plt.title("Guassian Distribution", size=12, weight='bold')
122 plt.xlabel("x")
123 plt.ylabel("probability density function (PDF)")
124
125
126 ## Evaluate intervals for the 1-sigma
127 xOneSigma = np.arange(mean-1*sigma, mean+1*sigma, 1/100)
128 yOneSigma = [myGauss(x, mean, sigma) for x in xOneSigma]
129 # yOneSigma = list(map(myGauss, xOneSigma, [mean]*len(xOneSigma) ,[sigma]*len(
        xOneSigma)) ) # same as previous line
130
131 ## Evaluate intervals for the 2-sigma
132 xTwoSigma   = np.arange(mean-2*sigma, mean+2*sigma, 1/100)
133 yTwoSigma   = [myGauss(x, mean, sigma) for x in xTwoSigma]
134 # yTwoSigma = list(map(myGauss, xTwoSigma, [mean]*len(xTwoSigma) ,[sigma]*len(
        xTwoSigma)) ) # same as previous line
135
136 xThreeSigma = np.arange(mean-3*sigma, mean+3*sigma, 1/100)
137 yThreeSigma = [myGauss(x, mean, sigma) for x in xThreeSigma]
138
139
140 # Fill the area behind the curve
141 plt.fill_between(x, y, 0, alpha=0.0, color="w")
142 plt.fill_between(xThreeSigma, yThreeSigma, 0, alpha=0.9, color=kBrazilBlue ,
```

```
    label=r'3$\sigma$ = 99.7%')
plt.fill_between(xTwoSigma  , yTwoSigma  , 0, alpha=0.9, color=kBrazilGold ,
    label=r'1$\sigma$ = 68.3%')
plt.fill_between(xOneSigma  , yOneSigma  , 0, alpha=0.9, color=kBrazilGreen,
    label=r'2$\sigma$ = 95.4%')

# Add error bars? find the mid-points of the bins and add them
xMid = [0.5*(binvalues[i-1]+binvalues[i]) for i in range(1, len(binvalues))]
xErr = 0.5*(binvalues[1]-binvalues[0])
yErr = [c/np.sqrt(N) for c in content]
plt.errorbar(xMid, content, yerr=yErr, xerr=xErr, fmt='none', color=kBrazilBlue
    )

# Add legend to plot
plt.legend(title=r"$f(x) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{\left(x-\mu\
    right)^2}{2\sigma^2}}$")

# Add line at x=mean?
plt.axvline(mean, color='black', linestyle="--", lw=1)

# Calculate integral, mean, variance, standard deviation
integral = getIntegral(content, binvalues)
variance = getVariance(xList)
stdDev   = getStDev(xList)
mean1    = getMean1(content, binvalues)
mean2    = getMean2(content, binvalues)
print("===Histogram calculations:" )
print("\tintegral = %.2f" % (integral))
print("\tmean1 = %.2f" % (mean1))
print("\tmean2 = %.2f" % (mean2))
print("\tvar = %.2f" % (variance))
print("\tstdDev = %.2f" % (stdDev))

for ext in [".png", ".pdf"]:
    plt.savefig("ex2" + ext)
plt.show()
quit()
```

Αποτέλεσμα:


Guassian Distribution
$f(x) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$
$\mu = 60.0, \sigma = 5.0$
N = 1000
3σ = 99.7%
1σ = 68.3%
2σ = 95.4%
probability density function (PDF)
0.00
0.02
0.04
0.06
0.08
0.10
40
50
60
70
80
x

Παράδειγμα 4 Παράδειγμα προσεγγιστικής μεθόδου εύρεσης ριζών της εξίσωσης $\sin(x)$:

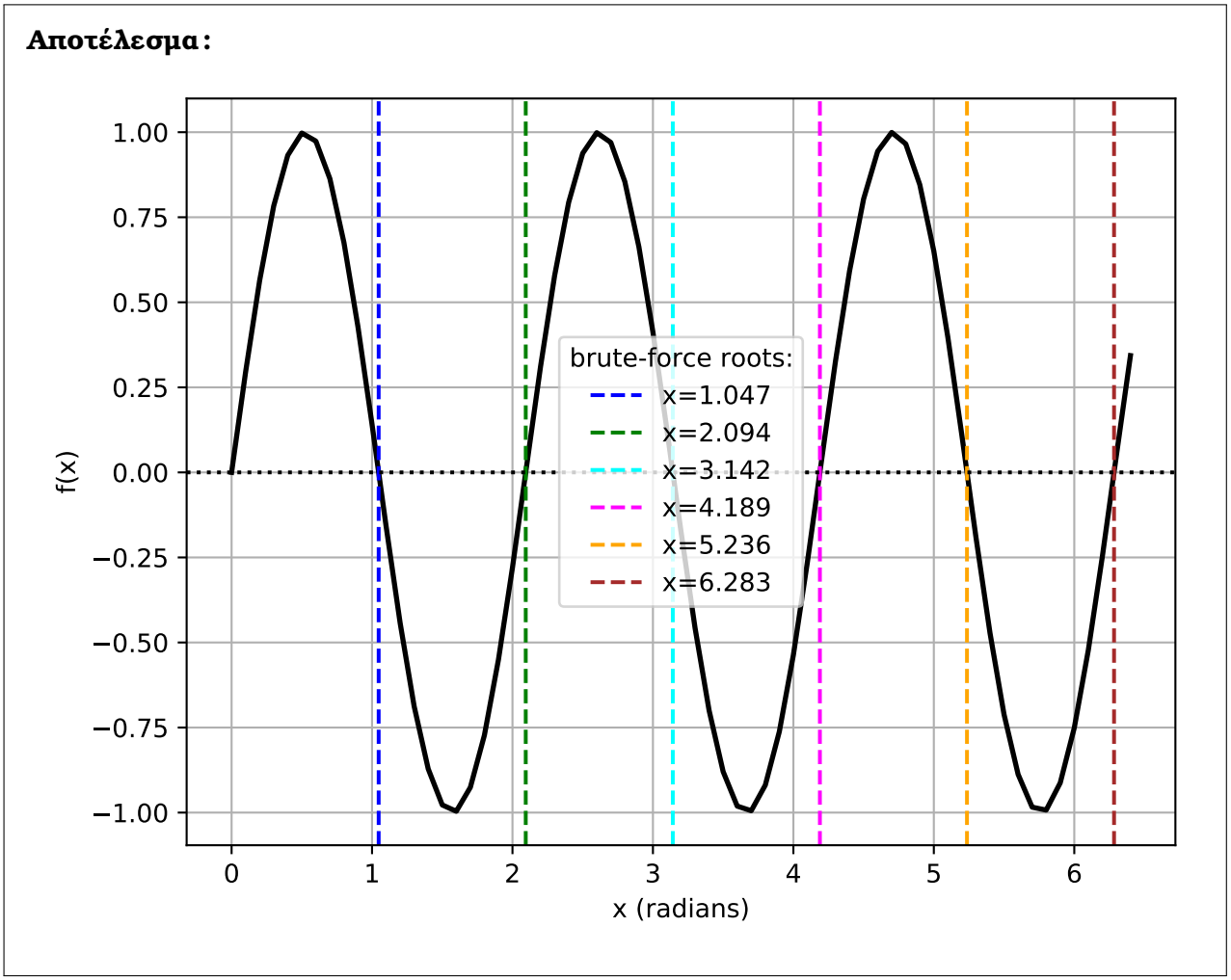
tutorial8/ex3.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex3.py
   python3 ex3.py
   script -q ex3.log python3 -i ex3.py
'''
import numpy as np
import matplotlib.pyplot as p

def bruteForceRootFinder(f, start, stop,  N):
    '''
    f .......: a function that given a param x returns another value f(x)
    start ...: the lower bound of the interval to be considered
    stop ....: the upper bound of the interval to be considered
    N .......: the total number of equally-spaced points to be created is N+1
    '''
    dx = (stop - start)/N
    xList = [start + x * dx for x in range(0, N+1, 1)]
    yList = list( map( f, xList) )
    rList = []

    # For-loop over all intervals
    for i in range(0, N-1, 1):
        root = xList[i] - ( xList[i+1] - xList[i] ) /(yList[i+1] - yList[i]) *
   yList[i]

        # Check if we have crossed the x-axis; y(i) and y(i+1) have different
   signs
        if yList[i] * yList[i+1] < 0:
            rList.append(root)
    return rList

def myFunc(x):
    return np.sin(3*x)

# Execute the program
rList = bruteForceRootFinder(myFunc, 0.0, 2.05*np.pi, 100)
xList = np.arange(0, 2.05* np.pi, 0.1)
yList = list( map(myFunc, xList) )
cList = ["blue", "green", "cyan", "magenta", "orange" , "brown", "yellow", "red
   ", "purple"]

### Visualise the results
p.figure()
p.xlabel("x (radians)")
p.ylabel("f(x)")
p.plot(xList, yList, 'k-', lw=2)
p.grid(True)
for i, root in enumerate(rList, 0):
```

Παράδειγμα 4 συνεχίζεται...

```
48      p.axvline(x=root, color=cList[i], linestyle='--', label="x=%.3f" % (root) )
49  p.axhline(y=0,color='k',linestyle=':')
50  p.tight_layout()
51  p.legend(title="brute-force roots:")
52
53  for ext in [".png", ".pdf"]:
54      p.savefig("ex3" + ext)
55  p.show()
56
57  quit()
```

Αποτέλεσμα:



Παράδειγμα 5 Απλό παράδειγμα για τη χρήση συμβολικών υπολογισμών στην Python με *Sympy* και γραφική αναπαράσταση των αποτελεσμάτων στο ίδιο γράφημα:

tutorial8/ex4.py

```
#!/usr/bin/python3
'''
USAGE:
   chmod +x ex4.py
   python3 ex4.py
   script -q ex4.log python3 -i ex4.py


DESCRIPTION:
Example usage of symbolic calculations


PREREQUISITES:
sudo pip install --upgrade pip
sudo pip install sympy


LINKS:
https://docs.sympy.org/latest/tutorials/intro-tutorial/basic_operations.html
https://docs.sympy.org/dev/modules/utilities/lambdify.html
'''
import sympy as s
import matplotlib.pyplot as plt
import numpy as np

# Define symbolic variable and expression
x   = s.Symbol("x")
f   = 2*x**2 + x*2 + x - 10
intF= s.integrate(f, x)
df  = s.diff(f, x)
ddf = s.diff(df, x)

# Define my functions using lambdify
F       = s.lambdify(x, f)
INTF    = s.lambdify(x, intF)
DF      = s.lambdify(x, df)
DDF     = s.lambdify(x, ddf)

# Define my lists
xList   = np.arange(0, 4, 0.005)
fList   = [ F(x) for x in xList ]
intList = [ INTF(x) for x in xList ]
dfList  = [ DF(x) for x in xList ]
ddfList = [ DDF(x) for x in xList ]

# Plot my functions
plt.figure(figsize=(14,7))
plt.subplot(1,1,1)
plt.xlabel('x')
```

```
50 plt.ylabel('f(x)')
51 plt.plot(xList, fList   , 'k-' , label=r'$f(x) = %s$' % s.latex(f)   ,
       linewidth=2)
52 plt.plot(xList, intList , 'r:' , label=r'$\int f(x) \, dx = %s$' % s.latex(intF
       ), linewidth=2)
53 plt.plot(xList, dfList  , 'm--', label=r'$\frac{df}{dx} = %s$' % s.latex(df)  ,
        linewidth=2)
54 plt.plot(xList, ddfList , 'g--', label=r'$\frac{d^{2}}{dx^2} = %s$' % s.latex(
       ddf) , linewidth=2)
55 plt.legend(title="Symbolic computations with Sympy")
56 plt.grid(True)
57 for ext in [".png", ".pdf"]:
58     plt.savefig("ex4" + ext)
59 plt.show()
60
61 print("=== Done!")
62 quit()
```

Αποτέλεσμα: