Σημειωματάριο Δευτέρας, 6 Νοε. 2017

Ένα πρόγραμμα για επίλυση ενός 2x2 γραμμικού συστήματος

Παρακάτω γράφουμε μια συνάρτηση `solve` η οποία βρίσκει τις λύσεις του γραμμικού συστήματος

$$aX + bY = e$$
$$cX + dY = f,$$

για τους αγνώστους X, Y.

Μπορείτε να δείτε μια σύντομη περιγραφή της λύσης του παραπάνω, με τη χρήση 2x2 οριζουσών,
[http://users.sch.gr/kpratsoli/images/documents/blyk/system_th_st.pdf]
(http://users.sch.gr/kpratsoli/images/documents/blyk/system_th_st.pdf).

Κεντρικό ρόλο παίζουν οι ορίζουσες D, Dx και Dy τις οποίες υπολογίζουμε πρώτα. Η εύκολη περίπτωση είναι όταν $D \neq 0$.

Επειδή ένα σύστημα όπως το παραπάνω δεν έχει πάντα μοναδική λύση η συνάρτησή μας επιστρέφει και ένα status του συστήματος που μας λέει αν το σύστημα έχει μοναδική λύση, αν είναι αδύνατο και αν είναι αόριστο. Γενικά επιστρέφουμε μια λίστα της μορφής

```
[x, y, status]
```

όπου τα x, y είναι πραγματικοί αριθμοί και το status είναι ένα string που μας περιγράφει την κατάσταση του συστήματος. Ανάλογα με την τιμή του status δίνουμε και ερμηνεία στις ποσότητες x, y σύμφωνα με τον παρακάτω πίνακα:

status	**x**	**y**
"nonsingular"	η μοναδική λύση του συστήματος	
"impossible"	αδύνατο σύστημα, δεν έχουν σημασία τα x, y	
"indefinite-X-from-Y"	αόριστο σύστημα, το X δίνεται από το Y από την $X = aY + b$, όπου a και b είναι αυτά που επιστρέφει η συνάρτηση στη θέση των x, y	
"indefinite-Y-from-X"	αόριστο σύστημα, το Y δίνεται από το X από την $Y = aX + b$, όπου a και b είναι αυτά που επιστρέφει η συνάρτηση στη θέση των x, y	
"all"	αόριστο σύστημα, κάθε x, y είναι λύση	

In [11]:

```
def solve(a, b, c, d, e, f):

    D = a*d-b*c # Υπολογίζουμε τις 3 ορίζουσες
    Dx = e*d - b*f
    Dy = a*f - e*c
    if abs(D)>1e-6: # Αν η ορίζουσα D δεν είναι 0 (πάντα το ελέγχουμε προσεγγιστ
ικά)
        x = Dx/D; y = Dy/D
        return [x, y, "nonsingular"]
    if abs(Dx) > 1e-6 or abs(Dy) > 1e-6: # Σύστημα είναι αδύνατο σε αυτή την περ
ίπτωση
        return [0, 0, "impossible"]
    # Από δω και κάτω και οι τρεις ορίζουσες είναι (προσεγγιστικά 0). Θα πρέπει
```

```
 να ελέγξουμε και τους
    # συντελεστές για να βγάλουμε συμπέρασμα.
    # Πρώτα φροντίζουμε ώστε η δεύτερη γραμμή του συστήματος να έχει τουλάχιστον
 ένα μη μηδενικό συντελεστή.
    if abs(a)<1e-6 and abs(b)<1e-6 and abs(e)<1e-6: # Ελέγχουμε αν η πρώτη γραμμ
ή είναι όλα μηδενικά
        firstrowzero=True
    else:
        firstrowzero=False
    if abs(c)<1e-6 and abs(d)<1e-6 and abs(f)<1e-6: # Ελέγχουμε αν η δεύτερη γρα
μμή είναι όλα μηδενικά
        secondrowzero=True
    else:
        secondrowzero=False
    if firstrowzero and secondrowzero: # Αν όλα είναι μηδενικά τότε τα πάντα είν
αι λύση
        return [0, 0, "all"] # Τα πάντα είναι λύση
    if secondrowzero: # Αν η δεύτερη γραμμή είναι όλο μηδέν τότε την εναλλάσουμε
 με την πρώτη ώστε να ξέρουμε από
                      # δω και κάτω ότι υπάρχει κάποιο μη μηδενικό στη δεύτερη γ
ραμμή. Αυτό δεν αλλάζει φυσικά τις λύσεις.
        a, c = c, a;   b, d = d, b;   e, f = f, e
    if abs(c)>1e-6: # Αν το c δεν είναι 0
        L = a/c # Για να μην είναι το σύστημα αδύνατο πρέπει η πάνω γραμμή του σ
υστήματος να είναι πολλαπλάσιο
                # της κάτω γραμμής. Ο λόγος (άνω προς κάτω) είναι το L. Παρακάτω
 ελέγχουμε αν ο ίδιος λόγος
                # ισχύει και για τις άλλες δύο στήλες, αλλά προσέχουμε μη διαιρέ
σουμε μήπως και είναι 0 τα d ή f.
        if abs(b-L*d)<1e-6 and abs(e-L*f)<1e-6:
            return [ -d/c, f/c, "indefinite-X-from-Y"]
                # Επιστρέφουμε στην 1η θέση το συντελεστή του Y, A, και
                # στη 2η θέση το σταθερό όρο, B, οπότε η γενική λύση
                # είναι X = A Y + B, με Y οτιδήποτε.
        else: # αδύνατο σύστημα
            return [0, 0, "impossible"]
    if abs(d)>1e-6: # Αν το d δεν είναι 0, κάνουμε αντίστοιχα ό,τι κάναμε ακριβώ
ς από πάνω
        L = b/d
        if abs(a-L*c)<1e-6 and abs(e-L*f)<1e-6:
            return [ c/d, -f/d, "indefinite-Y-from-X"]
                # Επιστρέφουμε στην 1η θέση το συντελεστή του X, A, και
                # στη 2η θέση το σταθερό όρο, B, οπότε η γενική λύση
                # είναι Y = A X + B, με X οτιδήποτε.
        else: # αδύνατο σύστημα
            return [0, 0, "impossible"]
    # Εδώ που φτάσαμε έχουμε c=d=0 και f !=0. Αυτό είναι αδύνατο.
    return [0, 0, "impossible"]

print(solve(1, 0, 0, 1, 2, 3))

print(solve(1, 1, 1, 1, 2, 3))

print(solve(1, 1, 2, 2, 2, 4))

print(solve(0, 0, 0, 0, 2, 3))

print(solve(0, 0, 0, 0, 0, 0))
```

```
[2.0, 3.0, 'nonsingular']
[0, 0, 'impossible']
[-1.0, 2.0, 'indefinite-X-from-Y']
[0, 0, 'impossible']
[0, 0, 'all']
```

Σχεδιασμός με τη βιβλιοθήκη matplotlib

Με τη βιβλιοθήκη `matplotlib` μπορούμε και σχεδιάζουμε ("γραφικά"). Με την εντολή

```
import matplotlib.pyplot as plt
```

εισάγουμε τη βιβλιοθήκη με το σύντομο (αυθαίρετο) όνομα `plt`, και οι συναρτήσεις αυτής καλούνται ως `plt.XXXX`.

Η βασική συνάρτηση σχεδιασμού που είδαμε σήμερα είναι η `plot` (καλείται ως `plt.plot`, αν `plt` είναι το όνομα που έχουμε επιλέξει για τη βιβλιοθήκη `matplotlib.pyplot`). Η `plot`, στη βασική της μορφή, παίρνει ως παραμέτρους δύο λίστες

```
plt.plot([x0, x1, ..., xN], [y0, y1, ..., yN])
```

και σχεδιάζει την πολυγωνική γραμμή που ξεκινάει από το σημείο (x_0, y_0) πάει (με ευθύγραμμο τμήμα) στο (x_1, y_1), μετά στο (x_2, y_2), κλπ, και καταλήγει στο (x_N, y_N).

Μπορείτε να δείτε μια πιο πλήρη περιγραφή της `matplotlib.pyplot.plot` στο
[https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.plot.html]
(https://matplotlib.org/devdocs/api/_as_gen/matplotlib.pyplot.plot.html)

Οι εντολές της `matplotlib` δεν έχουν άμεσο αντίκτυπο στην οθόνη μας παρά μόνο αφού κληθεί η συνάρτηση `show`, η οποία κάνει να φανούν στην οθόνη ό,τι έχουμε σχεδιάσει προηγουμένως.

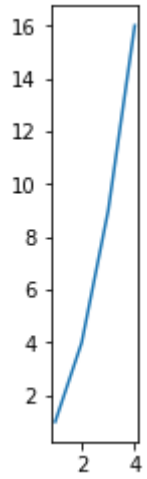
Η εντολή

```
plt.axes().set_aspect('equal')
```

είναι μια εντολή που επιβάλει στη μονάδα μέτρησης στους δύο άξονες να είναι η ίδια, ώστε π.χ. αν η απόσταση ανάμεσα στο 1 και στο 2 στον άξονα των x είναι 1cm το ίδιο να είναι και η απόσταση ανάμεσα στο 1 και στο 2 στον άξονα των y (αυτός ο λόγος των μονάδων λέγεται aspect ratio).

In [7]:

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3 ,4], [1, 4, 9, 16])

plt.axes().set_aspect('equal')
plt.show()
```



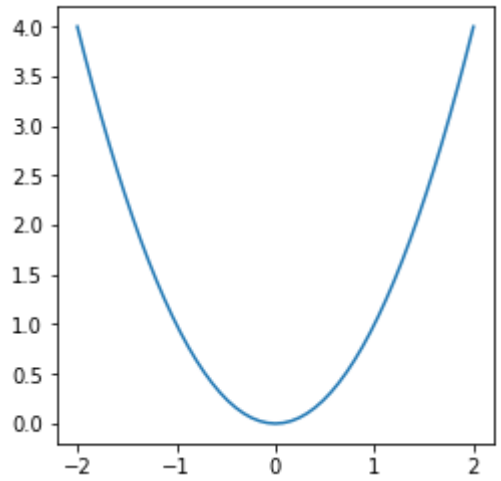
Έπειτα σχεδιάζουμε το γράφημα της συνάρτησης $y = x^2$ στο διάστημα $[-2, 2]$. Για να το κάνουμε αυτό με την `plot` θα πρέπει να σχεδιάσουμε μια αρκετά πυκνή πολυγωνική γραμμή που ενώνει σημεία πάνω στο γράφημα αυτό. Χωρίζουμε λοιπόν το διάστημά μας σε N ίσα διαστήματα, μήκους λοιπόν $h = 4/N$ το καθένα, και με αυτό τον τρόπο ορίζουμε τη διαμέριση του διαστήματος $[-2, 2]$ στα σημεία

$$x_i = -2 + ih, \quad (i = 0, 1, \ldots, N),$$

και τα αντίστοιχα y-σημεία δίδονται φυσικά από τον τύπο $y_i = x_i^2$. Φτιάχνουμε αυτές τις δύο λίστες τις οποίες και περνάμε στην `plot`.

In [15]:

```
import matplotlib.pyplot as plt

N = 50 # Η παράμετρος ποιότητας. Όσο πιο μεγάλο τόσο πιο πυκνή θα είναι η πολυγωνική γραμμή
h = 4/N # το μήκος των μικρών διαστημάτων
x = []; y = []
for i in range(N+1):
    xx = -2+i*h
    x.append(xx)
    y.append(xx*xx)

plt.plot(x, y)

plt.axes().set_aspect('equal')
plt.show()
```

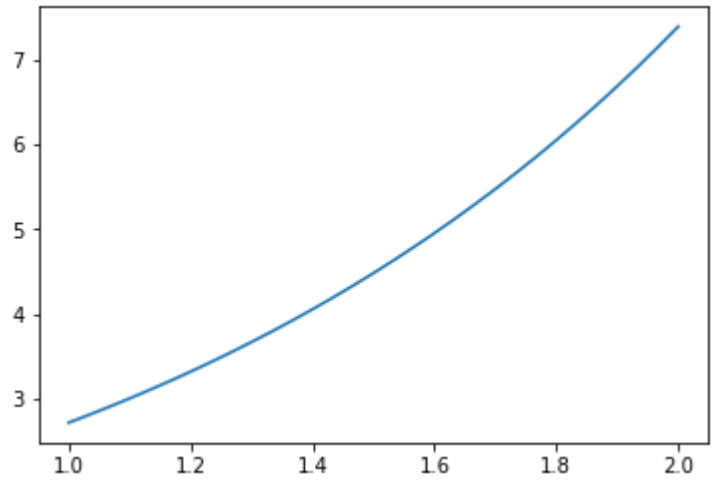


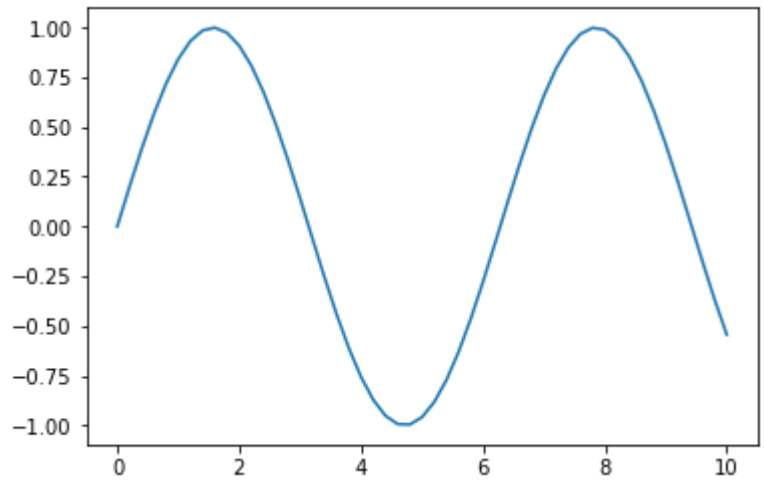
Φτιάχνουμε τώρα μια γενική συνάρτηση `plotfunction(a, b, f, N)` της οποίας σκοπός είναι να σχεδιάσει το γράφημα της συνάρτησης με όνομα `f` στο διάστημα `[a, b]` με μια πολυγωνική γραμμή με `N` σημεία.

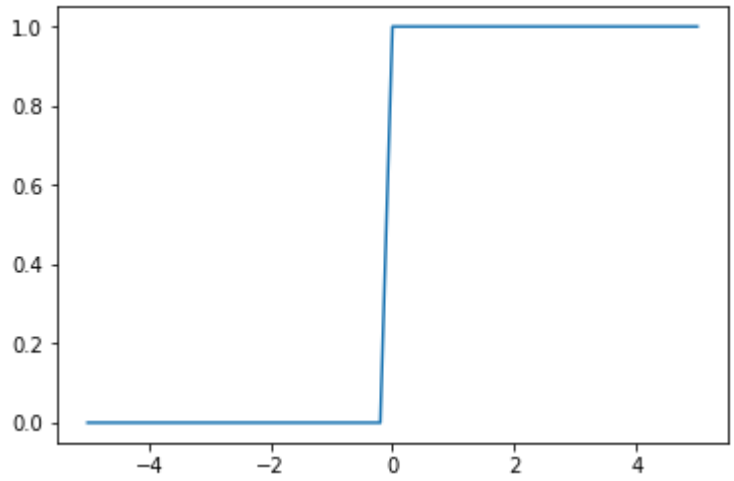
Τη χρησιμοποιούμε αυτή συνάρτηση για να σχεδιάσουμε τα γραφήματα τεσσάρων συναρτήσεων (δύο από τις οποίες είναι συναρτήσεις της βιβλιοθήκης `math`, και δύο είναι δικού μας σχεδιασμού και υπάρχουν μέσα στο πρόγραμμα οι ορισμοί τους.

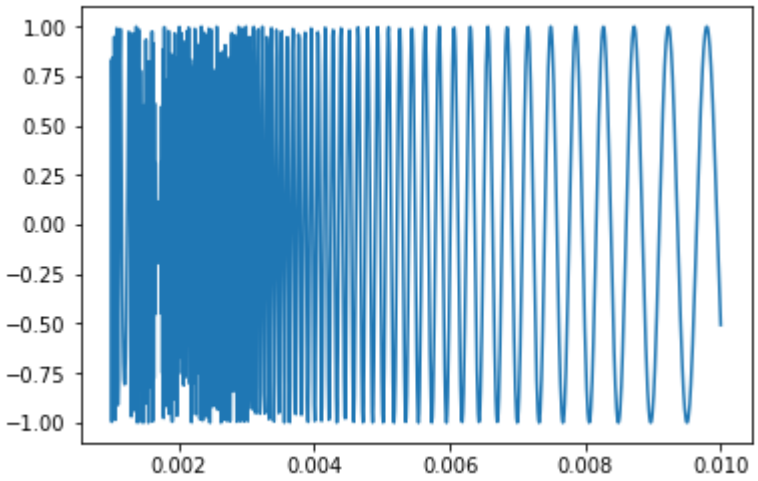
In [21]:

```
import matplotlib.pyplot as plt
import math

def step(x):
    if x<0:
        return 0
    return 1

def updown(x):
    return math.sin(1/x)

#plt.axis([0, 10, 0, 10])
def plotfunction(a, b, f, N):
    h = (b-a)/N
    x = [a+i*h for i in range(N+1)]
    y = [f(t) for t in x]
    plt.plot(x, y)
    plt.show()

plotfunction(1, 2, math.exp, 30)
plotfunction(0, 10, math.sin, 50)
plotfunction(-5, 5, step, 50)
plotfunction(1e-3, 0.01, updown, 1000)
```


7
6
5
4
3
1.0
1.2
1.4
1.6
1.8
2.0


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
2
4
6
8
10


1.0
0.8
0.6
0.4
0.2
0.0
−4
−2
0
2
4



Εδώ φτιάχνουμε μια συνάρτηση `plotparametric` για το σχεδιασμό μιας καμπύλης

$$(x(t), y(t)), \quad a \leq t \leq b$$

με N ενδιάμεσα σημεία.

Για παράδειγμα, για να σχεδιάσουμε τον μοναδιαίο κύκλο παίρνουμε $x(t) = \cos t,\ y(t) = \sin t$ για $0 \leq t \leq 2\pi$.

Με τις παραμετρικές συναρτήσεις `fx` και `fy` που έχουμε ορίσει μπορούμε να σχεδιάσουμε μια σπείρα.