Αριστοτελειο

Σχολη

# Στοχαστικά Πειράματα και Τυχαίοι Περίπατοι στη Στατιστική Φυσική

*Επιβλέπων:*
Ονοματεπώνυμο

*Συγγραφέας:*
Ονοματεπώνυμο

**Abstract**

Abstract

## Περίληψη

Περίληψη

# Περιεχόμενα

# Κεφάλαιο 1

# Εισαγωγή

ασδφασδφαδφασ [1]

# Κεφάλαιο 2

# Θεωρία

## 2.1 Τυχαίοι Περίπατοι

# Κεφάλαιο 3

# Πειράματα

## 3.1 Διακριτός Τυχαίος Περίπατος - Μέση τετραγωνική μετατόπιση

### 3.1.1 Σε μία διάσταση

Η πρώτη και απλούστερη προσομοίωση που θα κάνουμε αφορά ένα τυχαίο περίπατο σωματιδίου σε ένα ¨πλέγμα' μίας διάστασης. Ο σκοπός είναι να υπολογίσουμε την μέση τετραγωνική μετατόπιση 100000 προσομοιώσεων που ονομάζουμε runs όπως φαίνεται και στον κώδικα. Το κάθε run θα κάνει t=1000 τυχαία βήματα είτε αριστερά είτε δεξιά μεγέθους step=1.

Για να μοντελοποιήσουμε έναν τέτοιο τυχαίο περίπατο φανταζόμαστε την γραμμή των ακεραίων αριθμών και επιλέγουμε αυθαίρετα ένα σημείο εκκίνησης. Για απλότητα επιλέγουμε ως αρχική θέση την αρχή του άξονα δηλαδή position=0. Στη συνέχεια ένας βρόγχος επανάληψης t=1000 βημάτων ανανεώνει την θέση του σωματιδίου προσθέτοντας ή αφαιρώντας step=1 στη μεταβλητή position. Η τυχαία επιλογή του βήματος αριστερά ή δεξιά γίνεται με την εντολή random.choice που επιλέγει με ίση πιθανότητα ένα μέλος της λίστας [-step,+step]. Έτσι το βήμα αριστερά γίνεται αν στην θέση που βρίσκεται το σωματίδιο προσθέσουμε -1 ενώ το βήμα δεξιά αν προσθέσουμε +1. Επομένως ο αλγόριθμος ενός πειράματος έχει ως εξής:

```
//=================================================================//
set starting position

For t=1000 steps:
    choose randomly(uniformly) if the particle goes left or right
    update particle position
//=================================================================//
```

η ομοιόμορφη κατανομή εξασφαλίζει ότι σε κάθε βήμα το σωματίδιο έχει 1/2 πιθανότητα να μεταβεί στην θέση αριστερά του και 1/2 πιθανότητα να μεταβεί στη θέση δεξιά του.
Ο κώδικας που υλοποιεί το πείραμα:

```
position = 0
for j in range(t):
    move = random.choice([-step,+step])
    position = position + move
```

Έχοντας λοιπόν την τελική θέση είναι εύκολο να υπολογίσουμε την τετραγωνική μετατόπιση του πειράματος απλά υψώνοντας την μεταβλητή position στο τετράγωνο. Σκοπός όμως είναι να τρέξουμε το πείραμα 100000 φορές και να υπολογίσουμε την μέση τετραγωνική απόσταση για όλα τα runs. Άρα στον τελικό κώδικα αρχικοποιούμε μία μεταβλητή sd_sum=0(ακρωνύμιο του square distance sum) και σε κάθε run προσθέτουμε την τελική τετραγωνική μετατόπιση σε αυτή τη μεταβλητή. Έτσι υπολογίζουμε το άθροισμα όλων των τετραγωνικών μετατοπίσεων για τα 100000 πειράματα. Τέλος διαιρούμε αυτό τον αριθμό με τον αριθμό των πειραμάτων που τρέξαμε προκειμένου να βρούμε την μέση τετραγωνική μετατόπιση meam_sd.

```
start_time = time.time()

sd_sum = 0
runs = 100000
t=1000
step = 1
for i in range(runs):
    position = 0
    for j in range(t):
        move = random.choice([-step,+step])
        position = position + move
    sd_sum+=position**2
mean_sd = sd_sum/runs
```

```
print(f"The Mean Square Displacement is {mean_sd}.")
print(f"Execution Time: {(time.time() - start_time)} seconds.")
```

και το output:

```
The Mean Square Displacement is 995.37292.
Execution Time: 55.54194784164429 seconds.
```

### 3.1.2 Σε δύο διαστάσεις

Την ίδια λογική θα χρησιμοποιήσουμε στο πείραμα δύο διαστάσεων. Αυτή την φορά αντί να έχουμε μία γραμμή ακεραίων φανταζόμαστε δύο, κάθετες μεταξύ τους, με σημείο τομής το (0,0) όπως σε ένα καρτεσιανό σύστημα αξόνων. Η διαφορά είναι ότι ο χώρος μας αποτελείται από τα διακριτά ζεύγη των ακεραίων $(x, y) \in \mathbb{Z} \times \mathbb{Z}$. Έχουμε δηλαδή ένα πλέγμα δύο διαστάσεων πάνω στο οποίο το σωματίδιό μας θα εκτελέσει τον τυχαίο περίπατο. Η αρχική μας θέση πλέον έχει δύο συνιστώσες, την τετμημένη και την τεταγμένη.

Επομένως αρχικοποιούμε για το ένα run δύο μεταβλητές position_x=0 και position_y=0. Στη συνέχεια για την μία εκτέλεση του πειράματος t=1000 βημάτων, επιλέγουμε σε κάθε επανάληψη τυχαία αν το σωματίδιο θα κινηθεί αριστερά, δεξιά, πάνω ή κάτω. Ανανεώνουμε την τετμημένη και την τεταγμένη αναλόγως και έχουμε πλέον ως νέα θέση του σωματιδίου στο δισδιάστατο πλέγμα την (position_x, position_y)=(0,0). Συνεπώς ο αλγόριθμος ενός πειράματος έχει ως εξής:

```
//=====================================================================//
set starting position
For t=1000 steps:
    choose randomly(uniformly) if the particle goes
                                left, right, up or down
    update particle position
//=====================================================================//
```

Προφανώς η ομοιόμορφη τυχαία επιλογή μετάβασης δίνει 1/4 πιθανότητα για την κάθε κατεύθυνση που μπορεί να κινηθεί το σωματίδιο. Προφανώς η μετάβαση αριστερά σημαίνει πρόσθεση -1 στην τετμημένη και 0 στην τεταγμένη, η μετάβαση επάνω πρόσθεση +1 στην τεταγμένη και 0 στην τετμημένη κ.ο.κ. Ο κώδικας που υλοποιεί το ένα πείραμα είναι:

```
position_x = 0
position_y = 0
```

```
for j in range(t):
    move = random.choice([[-step,0],[step,0],[0,-step],[0,step]])
    position_x+= move[0]
    position_y+= move[1]
```

Έχοντας την τελική θέση του σωματιδίου μπορούμε να υπολογίσουμε την τετραγωνική μετατόπιση με χρήση του πυθαγόρειου θεωρήματος ως $d^2 = x^2 + y^2$.

Τρέχουμε λοιπόν το πρόγραμμα 100000 φορές και κάθε φορά προσθέτουμε την τετραγωνική μετατόπιση στην μεταβλητή sd_sum:

```
sd_sum+=position_x**2+position_y**2
```

Φυσικά στο τέλος διαιρούμε με τον αριθμό των πειραμάτων που εκτελέσαμε για να βρούμε την μέση τετραγωνική μετατόπιση των 100000 runs. Ο συνολικός κώδικας είναι ως εξής

```
start_time = time.time()

sd_sum = 0
runs = 100000
t=1000
step = 1
for i in range(runs):
    position_x = 0
    position_y = 0
    for j in range(t):
        move = random.choice([[-step,0],[step,0],[0,-step],[0,step]])
        position_x+= move[0]
        position_y+= move[1]
    sd_sum+=position_x**2+position_y**2
mean_sd = sd_sum/runs

print(f"The Mean Square Displacement is {mean_sd}.")
print(f"Execution Time: {(time.time() - start_time)} seconds.")
```

με output:

```
The Mean Square Displacement is 1009.55048.
Execution Time: 79.88995146751404 seconds.
```

## 3.2 Συνεχής τυχαίος περίπατος σε δύο διαστάσεις-Μέση Τετραγωνική Απόσταση

Στην συνέχεια των πειραμάτων με σωματίδιο σε τυχαίο περίπατο θα προσομοιώσουμε ένα συνεχή χώρο και όχι σε ένα πλέγμα με ακέραιες τιμές. Ο τρόπος να προσεγγίζουμε συνεχής χώρους υπολογιστικά είναι να δεχτούμε μικρές υποδιαιρέσεις των παραμέτρων. Ένα δισδιάστατο χώρο μπορούμε να τον παραμετροποιήσουμε με χρήση πολικών συντεταγμένων. Ο τρόπος με τον οποίο θα ανανεώνεται η θέση του σωματιδίου στον συνεχή χώρο είναι με την τυχαία επιλογή γωνίας. Χωρίζουμε το επίπεδο σε 360 ίσες διαμερίσεις ενός κυκλικού δίσκου. Επομένως, για ένα πείραμα, θα έχουμε μία επανάληψη χιλίων βημάτων και θα επιλέγουμε τυχαία μία γωνία. Συνεπώς ο αλγόριθμος θα έχει τη μορφή:

```
//==============================================================//
set starting position
For t=1000 steps:
    choose randomly(uniformly) an angle
    update particle position based on that angle
//==============================================================//
```

Εικόνα επεξ
γηματική

Για την τυχαία επιλογή της γωνίας εκμεταλλευόμαστε την εντολή random.randint(0,359) για την επιλογή σε μοίρες και μετατρέπουμε σε rad με χρήση του τύπου:

$$\frac{angle}{180} = \frac{rad}{\pi} \tag{3.1}$$

άρα:

```python
angle = (random.randint(0,359)/180)*np.pi
```

Η ανανέωση της θέσης γίνεται προβάλλοντας το ευθύγραμμο τμήμα μήκους step=1 στον άξονα $x$ πολλαπλασιάζοντας με το συνημίτονο της γωνίας, και στον άξονα $y$ πολλαπλασιάζοντας με το ημίτονο της γωνίας.

Συνεπώς για ένα πείραμα κώδικας έχει τη μορφή:

```python
position_x = 0
position_y = 0
for j in range(t):
    angle = (random.randint(0,359)/180)*np.pi
    position_x+=round(step*np.cos(angle),2)
    position_y+=round(step*np.sin(angle),2)
```

όπου η συνάρτηση round στρογγυλοποιεί τους υπολογισμούς των τριγωνομετρικών στο δεύτερο δεκαδικό. Εφόσον θέλουμε να υπολογίσουμε ξανά την μέση τετραγωνική μετατόπιση, τρέχουμε τον κώδικα για runs=100000 πειράματα και προσθέτουμε την τετραγωνική απόσταση από την αρχή των αξόνων(την αρχική μας θέση) σύμφωνα με το πυθαγόρειο θεώρημα. Στο τέλος διαιρούμε το άθροισμα με τον αριθμό των πειραμάτων για να βρούμε τη μέση τιμή. Ο συνολικός κώδικας έχει ως εξής:

```
start_time = time.time()

sd_sum = 0
runs = 100000
t=1000
step = 1
for i in range(runs):
    position_x = 0
    position_y = 0
    for j in range(t):
        angle = (random.randint(0,359)/180)*np.pi
        position_x+=round(step*np.cos(angle),2)
        position_y+=round(step*np.sin(angle),2)
    sd_sum+=position_x**2+position_y**2
mean_sd = sd_sum/runs

print(f"The Mean Square Displacement is {mean_sd}.")
print(f"Execution Time: {(time.time() - start_time)} seconds.")
```

με output:

```
The Mean Square Displacement is 999.8212898829913.
Execution Time: 963.7810792922974 seconds.
```

Τώρα θα υπολογίσουμε ξανά την μέση τετραγωνική μετατόπιση αλλά αυτή τη φορά θα πάρουμε τους μέσους όρους των 100000 πειραμάτων ανά 100 τυχαία βήματα. Δηλαδή θα βρούμε την μέση τετραγωνική μετατόπιση από 100000 πειράματα με 100 βήματα, την μέση τετραγωνική μετατόπιση από 100000 πειράματα με 200 βήματα κ.ο.κ. Η βασική δομή του κώδικα είναι όμοια με το προηγούμενο πείραμα σε συνεχές χώρο αλλά με κάποιες τροποποιήσεις.

Αρχικοποιούμε μία λίστα 10 θέσεων ονόματι averages με την τιμή μηδέν σε κάθε στοιχείο της. Σε αυτή τη λίστα θα προσθέτουμε τις τιμές της τετραγωνικής μετατόπισης για τα 100,200,300....1000 βήματα και θα διαιρούμε με τον αριθμό των πειραμάτων προκειμένου να έχουμε τις μέσες τιμές ανά αριθμό βημάτων.

Το σχήμα του αλγορίθμου ενός πειράματος έχει ως εξής:

```
//==========================================================================//
set starting position
create an empty list named values

For 1000 steps:
    choose randomly(uniformly) an angle
    update particle position based on that angle
    if steps are 100,200.....1000
              then fill the list values
              with the square displacement
//==========================================================================//
```

Ο κώδικας που υλοποιεί το ένα πείραμα είναι:

```python
position = [0,0]
values = []
for j in range(1,t+1):
    angle = (random.randint(0,359)/180)*np.pi
    position[0]+=round(np.cos(angle),2)
    position[1]+=round(np.sin(angle),2)
    if j%100==0:
        values.append(position[0]**2+position[1]**2)
```

Λεπτομερέστερα, το block κώδικα:

```python
if j%100==0:
    values.append(position[0]**2+position[1]**2)
```

ελέγχει πότε ο αριθμός των βημάτων είναι πολλαπλάσιο του 100 (κοιτώντας να μηδενίζεται το υπόλοιπο της διαίρεσης με το 100) και γεμίζει την λίστα values με τις 10 τετραγωνικές μετατοπίσεις όπως φαίνεται και στον ψευδοκώδικα. Ο λόγος που επιλέχθηκε το range(1,t+1) και όχι το range(0,t) είναι καθαρά για την ευκρίνεια του κώδικα και τη χρήση του mod100 αντί του mod99.
Έτσι για κάθε ένα πείραμα πρέπει να προσθέσουμε στην αρχικοποιημένη με μηδενικά λίστα averages τις τιμές της λίστας values διαιρεμένες με τον αριθμό των πειραμάτων προκειμένου να έχουμε στο τέλος τις μέσες τετραγωνικές μετατοπίσεις 100000 πειραμάτων ανά 100 βήματα του σωματιδίου.
Ο τελικός κώδικας θα είναι:

```python
start_time = time.time()

runs = 10000
```

```
t=1000
averages = [0]*10
for i in range(runs):
    position = [0,0]
    values = []
    for j in range(1,t+1):
        angle = (random.randint(0,359)/180)*np.pi
        position[0]+=round(np.cos(angle),2)
        position[1]+=round(np.sin(angle),2)
        if j%100==0:
            values.append(position[0]**2+position[1]**2)
    for j in range(10):
        averages[j]+=values[j]/runs

print(f"The averages are {averages}.")
print(f"Execution Time: {(time.time() - start_time)} seconds.")
```

με έξοδο:

```
The averages are [99.67193910999949,
                  201.10285323000062,
                  298.58902693000033,
                  399.82761215000033,
                  494.9311487600011,
                  597.2043227299978,
                  693.3227494999987,
                  791.3730152799965,
                  901.8665181699997,
                  1003.6801849000012].
Execution Time: 94.43011784553528 seconds.
```

Χρησιμοποιούμε την μέθοδο ελαχίστων τετραγώνων μέσω του πακέτου `sklearn.linear_model` και του αντικειμένου LinearRegression(). Αρχικά κατασκευάζουμε τα np.array αντικείμενα έτσι ώστε να μπορούν να εισαχθούν στις εντολές που θα χρησιμοποιήσουμε. Ως τετμημένες θα χρησιμοποιήσουμε τους αριθμούς των βημάτων που πραγματοποιήθηκαν στην μεταβλητή t. Η εντολή np.arange(100,1001,100).reshape(-1,1) ουσιαστικά κατασκευάζει ένα στηλοδιάνυσμα με τα πολλαπλάσια του 100 έως και το 1000. Ως τεταγμένες φυσικά τις μέσες τετραγωνικές μετατοπίσεις ανά 100 βήματα. Ο κώδικας:

```
t = np.arange(100,1001,100).reshape(-1,1)
av = np.array(averages)
```

Εν συνεχεία, κατασκευάζουμε το μοντέλο γραμμικής παλινδρόμησης μέσω των μεθόδων που αναφέραμε, και στη συνέχεια εκτελούμε το fitting, δηλαδή το την εύρεση της ευθείας ελαχίστων τετραγώνων. Στη μεταβλητή m τοποθετούμε την κλίση της ευθείας και στην μεταβλητή c την σταθερά, έτσι ώστε η ευθείας μας να γράφετε ως $y = \mu t + c$. Ο κώδικας:

```
reg = LinearRegression()
model = reg.fit(t,av)
c = model.intercept_
m = model.coef_[0]
print(f"y = {m}x+{c}")
```

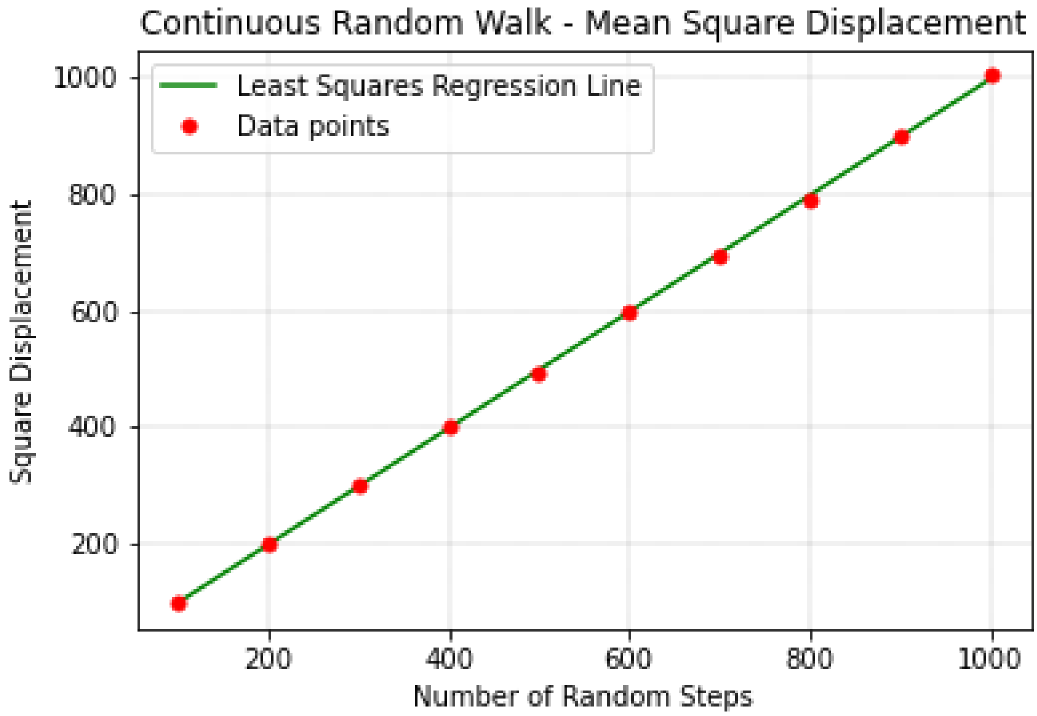
με έξοδο

```
y = 0.999278690573332x+-1.4463427393329766
```

Πλέον μένει να παραστήσουμε γραφικά την ευθεία και τα δεδομένα που χρησιμοποιήθηκαν προκειμένου να καταλήξουμε στα κατάλληλα συμπεράσματα. Για να το κάνουμε αυτό θα χρησιμοποιήσουμε το πακέτο matplotlib.pyplot με το ακρωνύμιο plt όπως συνήθως χρησιμοποιείτε. Χρησιμοποιούμε την εξίσωση της ευθείας και την αναπαριστούμε γραφικά με γραμμή, ενώ τα αρχικά δεδομένα με τελείες. Ονομάζουμε τους άξονες και τα χρώματα σύμφωνα με το πρόβλημά μας. Ο τελικός κώδικας:

```
y = m*t+c
plt.plot(t,y,'-',color ='green',label='Least Squares Regression Line')
plt.plot(t,av,'.',color = 'red',label='Data points',ms=10.0)
plt.legend()
plt.grid(color='0.25', linestyle='--', linewidth=0.2)
plt.ylabel('Square Displacement')
plt.xlabel('Number of Random Steps')
plt.title('Continuous Random Walk - Mean Square Displacement')
plt.show()
```

και το γράφημα:




Σχήμα 3.1: Μέση τετραγωνική απόσταση αν 100 βηματισμούς του σωματιδίου για συνεχή χώρο δύο διατάσεων

## 3.3 Αριθμός πλεγματικών θέσεων που το σωματίδιο επισκέφτηκε τουλάχιστον μία φορά

### 3.3.1 Σε μία διάσταση

### 3.3.2 Σε δύο διαστάσεις

## 3.4 Διακριτός τυχαίος περίπατος με παγίδες

### 3.4.1 Συγκέντρωση παγίδων ς=0.01

### 3.4.2 Συγκέντρωση παγίδων ς=0.001

### 3.4.3 Σύγκριση με προσέγγιση Ροσενστοcκ

# Κεφάλαιο 4

# Συμπεράσματα

# Παράρτημα Α΄

# Γλωσσάριο εντολών

# Παράρτημα Β′

# Jupyter Notebook

# Βιβλιογραφία

[1] Μ. Α. Νιελσεν ανδ Ι. Λ. ῆυανγ. *Χυαντυμ ὃμπυτατιον ανδ Χυαντυμ Ινφορματιον: 10τη Αννιερσαρψ Εδιτιον*. ᾶμβριδγε Υνιερσιτψ Πρεσ, 2010.