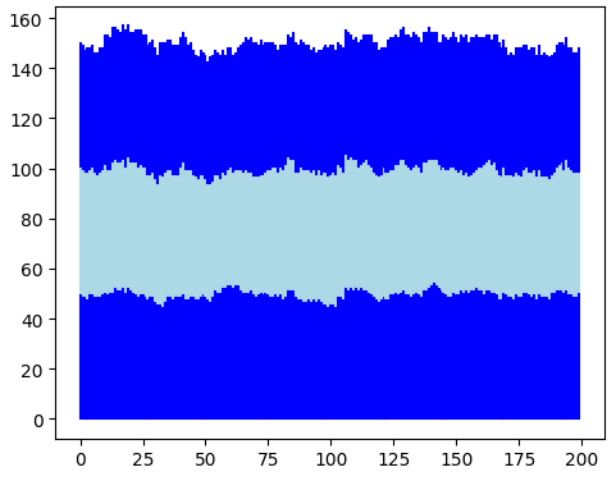
گزارش کار تمرین سوم شبیه‌سازی رایانه‌ای در فیزیک

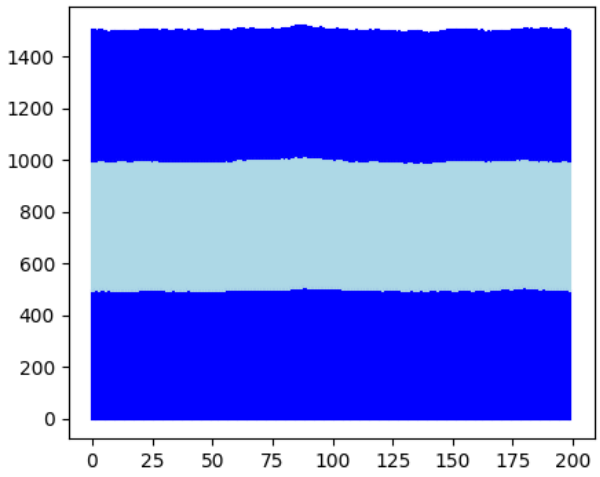
علی اکرامیان – 99100563

تمرین 3.2 (پایین نشست):

در این سوال نیز مانند سوال 3.1 عمل کرده‌ام و یک آرایه‌ی base گرفتم که اعداد0 تا 199 را دارد. سپس یک آرایه‌ی height تعریف کرده‌ام که همه‌ی درایه‌های آن صفر است (ارتفاع اولیه‌ی همه صفر است). سپس یک حلقه زده‌ام که N بار تکرار می‌شود و هر بار یک عدد رندوم بین 0 تا 199 انتخاب کرده و چک می‌کند که اگر برابر هرکدام بود، طول خودش را افزایش دهد (اولویت با خود آن عدد رندم است) و اگر این طور نبود مینیمم ارتفاع خودش، قبلی و بعدی‌اش را بگیرد و روی آن بنشیند. برای گرفتن مینیمم نیز از توابع argmin کتابخانه‌ی numpy استفاده کردم. آرگیومنت‌ها را در یک ارایه و سه تا همسایه را در یکی دیگر ریختم و پس از پیدا کردن مینیمم آن‌ها، از آرایه‌ی آرگیومنت‌ها، عدد ستون را گرفتم. برای تناوبی بودن نیز از این استفاده کردم که درایه‌ی 1- در پایتون یعنی آخرین درایه‌ی آن ارایه و برای گرفتن همسایه‌ی آخرین ستون نیز از باقی‌مانده‌ی آن به عداد ستون‌ها L استفاده کردم. حال آرایه‌ی نهایی ازین حلقه‌را کپی کرده و آن را صفر می‌کنیم تا یک بار دیگر این کار را انجام دهیم (لایه‌ی دوم) و برای لایه‌ی سوم نیز همین کار را می‌کنیم. حال ارتفاع‌های لایه‌ی جدید با لایه‌ی اول را که جمع کنیم، طول لایه‌ی دوم را داریم. برای لایه‌ی سوم نیز همین طور است.
حال میانگین و انحراف از معیار را حساب می‌کنیم برای هر یک از لایه‌ها و آنها را چاپ می‌کنیم. در آخر هم این لایه‌ها را می‌کشیم. کشیدن نیز این‌گونه است که نقطه‌ی بالایی ارتفاع از لایه‌را می‌گیریم و به نقطه‌ی انتهایی آن ستون در آن لایه وصل می‌کنیم (خط می‌کشیم). در نهایت چنین چیزهایی خواهیم داشت:



برای N=10000 نقطه



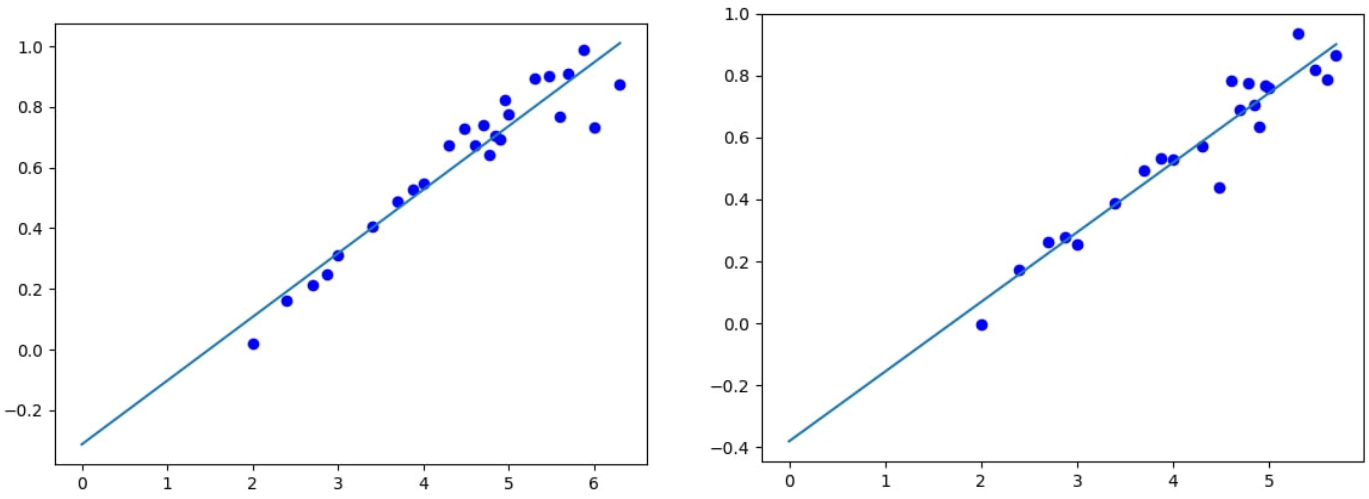
برای N=100000 نقطه

که بدیهی‌ست که وقتی تعداد نقاط بیشتر می‌شود، ناهمواری سطح به طور چشم‌گیری کاهش می‌یابد.
کد فایل را نیز در صفحه‌ی بعد می‌آورم (فایل مربوطه DBD.py است و فایل DBD-data.py مربوط به تحلیل داده‌هاست).
نکته: این N که من در کل این گزارش اشاره می‌کنم، تعداد هر لایه است یعنی وقتی N=1000 است، در واقع 3000 نقطه داریم.

simulation > HW3 > DBD.py > ...

```python
import matplotlib.pyplot as plt
import numpy as np
N=1000
L=200
base=np.arange(0,200)
height=np.zeros(L)
for i in range(1,N+1):
    rnd=np.random.randint(0,L)
    if height[rnd]==height[rnd-1] or height[rnd]==height[(rnd+1)%L]:
        height[rnd]+=1
    else:
        arg=np.array([rnd-1,rnd,(rnd+1)%L])
        narr=np.array([height[rnd-1],height[rnd],height[(rnd+1)%L]])
        min=np.argmin(narr)
        nrnd=arg[min]
        height[nrnd]+=1
height1=height.copy()
height=np.zeros(L)
for i in range(1,N+1):
    rnd=np.random.randint(0,L)
    if height[rnd]==height[rnd-1] or height[rnd]==height[(rnd+1)%L]:
        height[rnd]+=1
    else:
        arg=np.array([rnd-1,rnd,(rnd+1)%L])
        narr=np.array([height[rnd-1],height[rnd],height[(rnd+1)%L]])
        min=np.argmin(narr)
        nrnd=arg[min]
        height[nrnd]+=1
height2=height.copy()
height2+=height1
height=np.zeros(L)
for i in range(1,N+1):
    rnd=np.random.randint(0,L)
    if height[rnd]==height[rnd-1] or height[rnd]==height[(rnd+1)%L]:
        height[rnd]+=1
    else:
        arg=np.array([rnd-1,rnd,(rnd+1)%L])
        narr=np.array([height[rnd-1],height[rnd],height[(rnd+1)%L]])
        min=np.argmin(narr)
        nrnd=arg[min]
        height[nrnd]+=1
height3=height2+height
for i in range(0,L):
    plt.plot([base[i],base[i]],[0,height1[i]],'b')
    plt.plot([base[i],base[i]],[height1[i],height2[i]],'lightblue')
    plt.plot([base[i],base[i]],[height2[i],height3[i]],'b')
av_height1=sum(height1)/L
av_height2=sum(height2)/L
av_height3=sum(height3)/L
var_height1=np.var(height1)
var_height2=np.var(height2)
var_height3=np.var(height3)
print(av_height1,av_height2,av_height3)
print(var_height1,var_height2,var_height3)
plt.show()
```

حال در فایل DBD-data.py تحلیل داده‌ها را انجام می‌دهیم (همان کد قبلی‌است که قسمت گرافیکی ندارد و برای N ها و Lهای مختلف تست می‌شود.
در این کد یک آرایه‌ی دیتا برای N داریم و یک آرایه برای L که برای این اعداد حلقه داریم و تکرار می‌شود و سپس واریانس آنها را حساب کرده و به توان یک دوم می‌رسانم. حال این عدد که انحراف معیار است را برای N های مختلف بدست آورده و الگاریتم آنها را برحسب هم رسم می‌کنم و می‌بینم که چه موقع تقریبا صاف می‌شود (شیب = 0) و جایی که به صورت خطی است، شیب خط را حساب کرده و گزارش می‌کنم. با polyfit نیز این خط را فیت می‌کنم (کمترین مربعات) که نتیجه را در ادامه می‌بینیم:

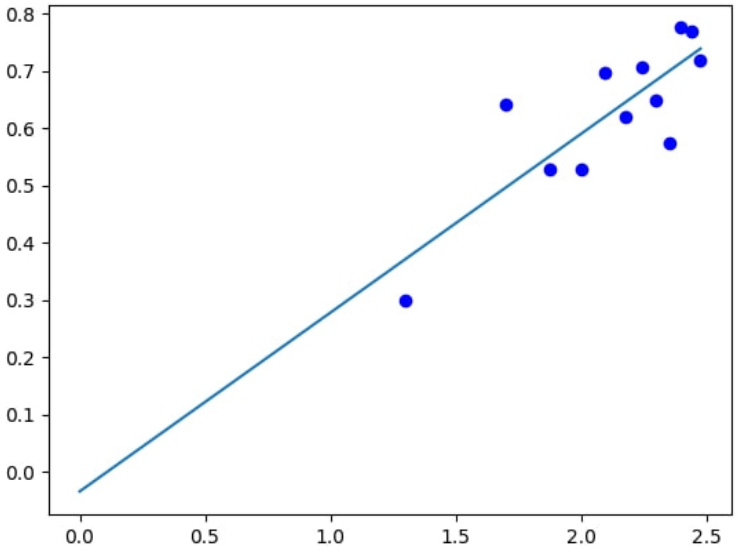


که داده‌ها برای اشباع تا 3 حلقه‌ی 3 میلیونی پیش رفته است.

معادله‌ی خط نیز به صورت زیر است:

$$y=0.21\ x - 0.3$$

که ضریب ایکس همان توان t است (چون محورها لگاریتمی است)
و اگر نمودار لگاریتم انحراف معیار را بر حسب طول رسم کنیم:

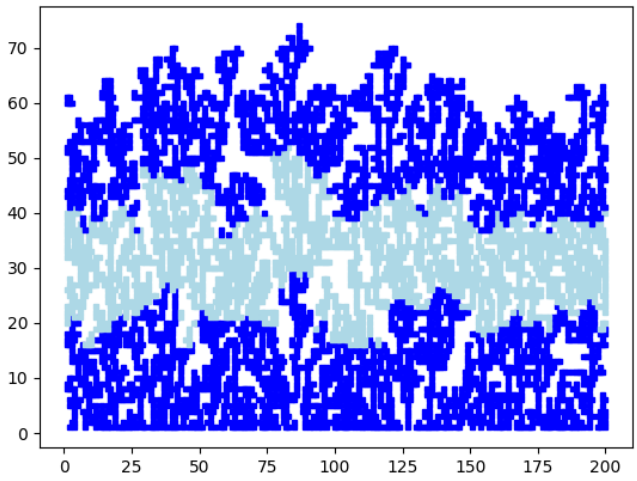


که معادله‌ي خط بدین صورت است:

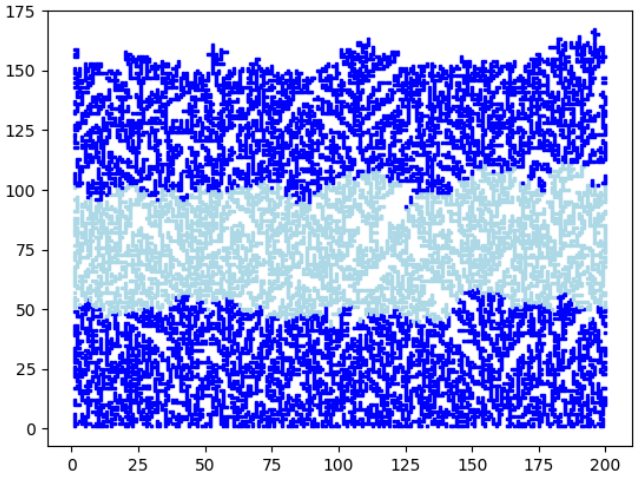
$$y=0.3\ x - 0.3$$

تمرین 3.3 (کنار نشست):

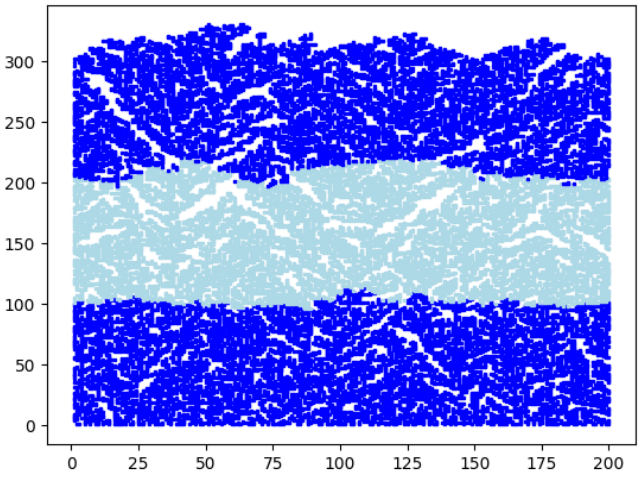
در این سوال همه‌ی کارهایم شبیه به سوال 32 است و فقط شروط آن را عوض کرده‌ام:
شرط: اگر ارتفاع ستون رندوم انتخاب شده از همسایه‌ی چپ و راستش بیشتر یا مساوی بود، یکی روی خودش بگذارد. اگر این طور نیز نبود، ارتفاع همسایه‌ی بلندتر را بگیرد و ارتفاع این ستون رندوم را برابر با ارتفاع ستون بلندتر بگذارد و در آن محل یک نقطه‌ی مربعی بکشد. در ضمن چون قرار است نقطه به نقطه رسم کنیم، دستور پلات کردن در خود حلقه‌ها و شروط آمده است. خروجی را می‌بینیم: (N تعداد نقاط هر لایه است)



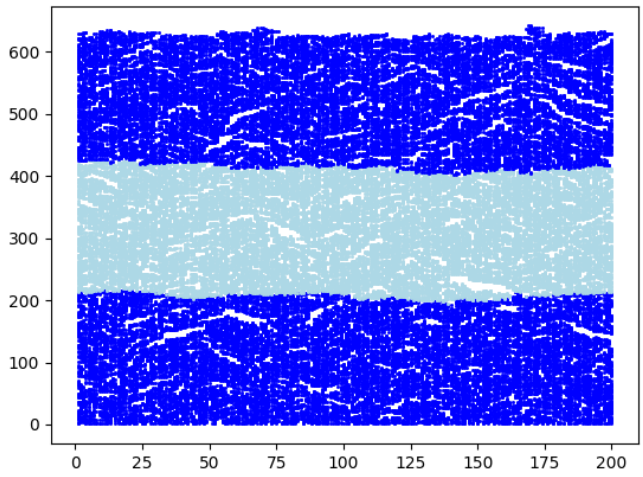
برای N=2000 نقطه



برای N=5000 نقطه



برای N=10000 نقطه



برای N=20000 نقطه

می‌بینیم که با زیاد شدن تعداد نقاط ناهمواری سطح کاهش می‌یابد و سطح به هموار شدن نزدیک‌تر می‌شود.
در ضمن میانگین و واریانس را نیز برای هر لایه حساب کرده و چاپ می‌کنم (در خود کد!)

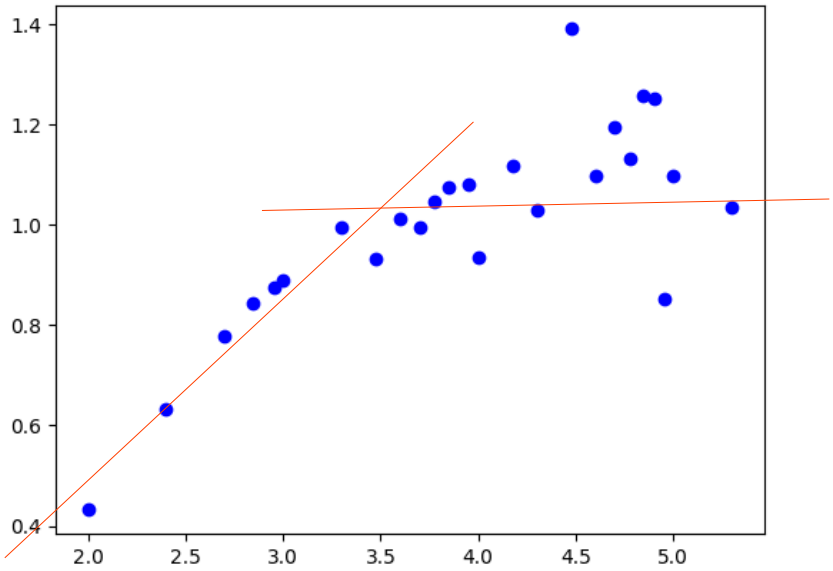
کد را نیز در زیر می‌بینیم:

simulation > HW3 > SBD.py > ...

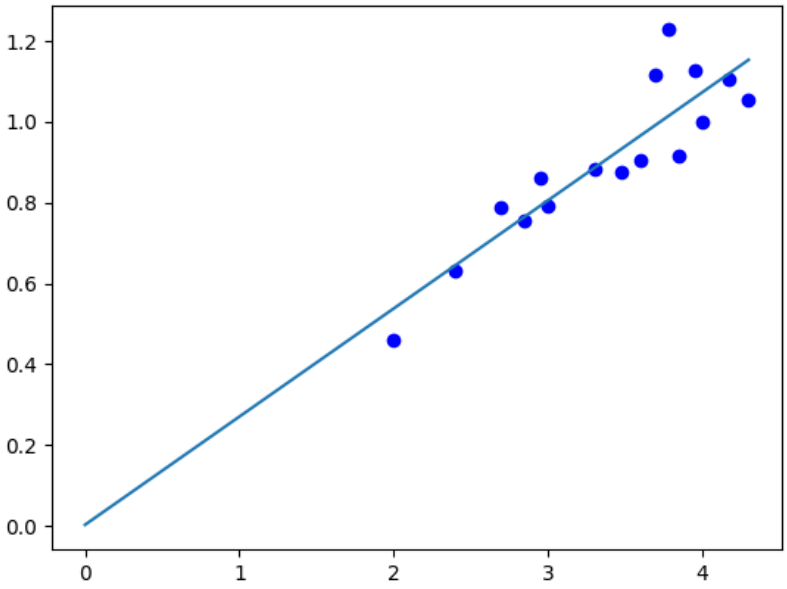
```python
import matplotlib.pyplot as plt
import numpy as np
N=20000
L=200
markersize=1
base=np.arange(1,L+1)
height=np.zeros(L)
for i in range(1,N+1):
    rnd=np.random.randint(0,L)
    if height[rnd]>=height[rnd-1] and height[rnd]>=height[(rnd+1)%L]:
        height[rnd]+=1
    else:
        arg=np.array([rnd-1,(rnd+1)%L])
        narr=np.array([height[rnd-1],height[(rnd+1)%L]])
        max=np.argmax(narr)
        nrnd=arg[max]
        height[rnd]=height[nrnd]
    plt.plot(base[rnd],height[rnd],c='b',marker='s',markersize=markersize)
height1=height.copy()
for i in range(1,N+1):
    rnd=np.random.randint(0,L)
    if height[rnd]>=height[rnd-1] and height[rnd]>=height[(rnd+1)%L]:
        height[rnd]+=1
    else:
        arg=np.array([rnd-1,(rnd+1)%L])
        narr=np.array([height[rnd-1],height[(rnd+1)%L]])
        max=np.argmax(narr)
        nrnd=arg[max]
        height[rnd]=height[nrnd]
    plt.plot(base[rnd],height[rnd],c='lightblue',marker='s',markersize=markersize)
```

```python
height2=height.copy()
height2+=height1
for i in range(1,N+1):
    rnd=np.random.randint(0,L)
    if height[rnd]>=height[rnd-1] and height[rnd]>=height[(rnd+1)%L]:
        height[rnd]+=1
    else:
        arg=np.array([rnd-1,(rnd+1)%L])
        narr=np.array([height[rnd-1],height[(rnd+1)%L]])
        max=np.argmax(narr)
        nrnd=arg[max]
        height[rnd]=height[nrnd]
    plt.plot(base[rnd],height[rnd],c='b',marker='s',markersize=markersize)
height3=height2+height
av_height1=sum(height1)/L
av_height2=sum(height2)/L
av_height3=sum(height3)/L
std_height1=np.var(height1)**0.5
std_height2=np.var(height2)**0.5
std_height3=np.var(height3)**0.5
print(av_height1,av_height2,av_height3)
print(std_height1,std_height2,std_height3)
print(height1,height2,height3)
plt.show()
```

حال در فایل SBD-data.py تحلیل داده‌ها را انجام داده‌ام و مانند سوال قبلی به ازای طول‌های متفاوت و تعداد نقاط رندوم متفاوت رسم کرده ام نقاط را و منحنی را کشیده‌ام:



که قسمت خطی از قسمتی که اشباع می‌شود قابل تشخیص است. حال به قسمت خطی آن یک خط فیت می‌کنیم تا توان تی در رابطه را بدست آوریم:



که معادله‌ی خط بدین صورت است:

y = 0.26x - 0.02

که یعنی توان t عدد 0.26 است. چون شیب این خط توان تی را می‌دهد (لگاریتم انحراف معیار بر حسب لگاریتم زمان)

وابستگی کنار نشست به ال نیز در کد آمده است و به طورت خطی‌است که من در کد آن را نیز آورده‌ام.

تمرین 3.4 (رشد گیاه):

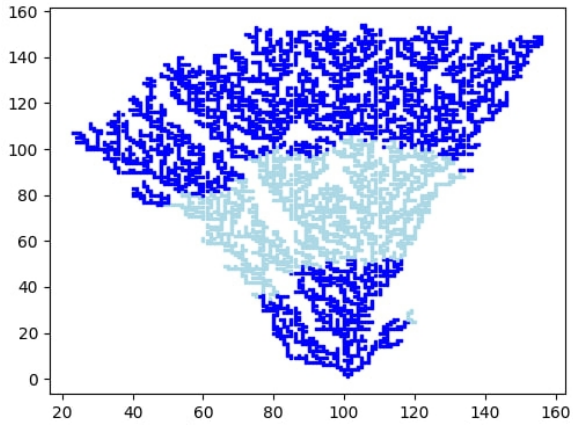
در این تمرین من ابتدا ستون وسط را ب ارتفاع دو رساندم و سپس حلقه را مانند سوالات قبلی زدم. البته شروط آن نیز این گونه است:

1. اگر ارتفاع ستون انتخاب شده (به صورت رندوم) یا هر یک از همسایگانش صفر بود، به عدد بعدی در حلقه برود (حلقه را ادامه دهد و کاری نکند)
2. اگر ارتفاعش از هریک از همسایگانش بیشتر بود، خودش یک واحد بیشتر شود
3. اگر هیچ یک از این‌ها نبود مانند کنار نشست ماکزیمم ارتفاع دو همسایه‌اش را پیدا کرده و ارتفاعش همان ارتفاع بلندترین همسایه‌اش بشود.

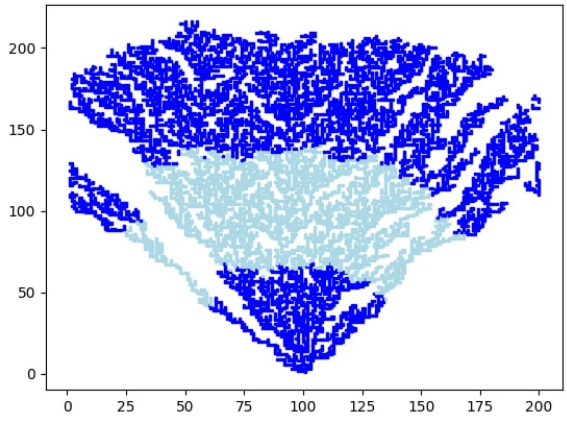
هر بار که حلقه نیز تکمیل می‌شود (موارد 2 و 3) آن نقطه‌ی جدید را به صورت مربعی رسم می‌کنم. (با markersize) مشخص می‌کنم اندازه‌ی این مربع چقدر باشد.

حال مانند بقیه‌ی کدهایم این را نیز 3 بار تکرار می‌کنم (3 لایه شود) و نتیجه را می‌کشم. دیگر خروجی‌های کد نیز میانگین، واریانس و انحراف معیار برای ارتفاع‌های هر لایه است.

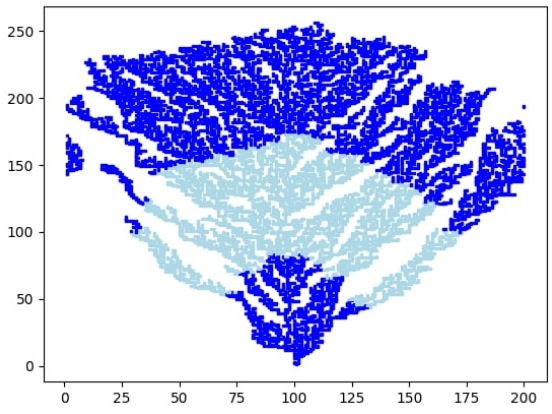
حال برای چند N مختلف (تعداد نقاط کل در واقع سه برابر N است) نتیجه را می‌بینیم:



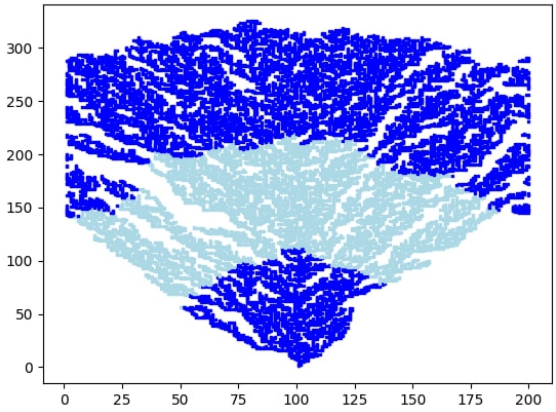
برای N=5000 نقطه



برای N=7000 نقطه



برای N=8000 نقطه



برای N=10000 نقطه

که در 10000 نقطه واضح است که به اشباع رسیده و کل طول را پوشانده است.

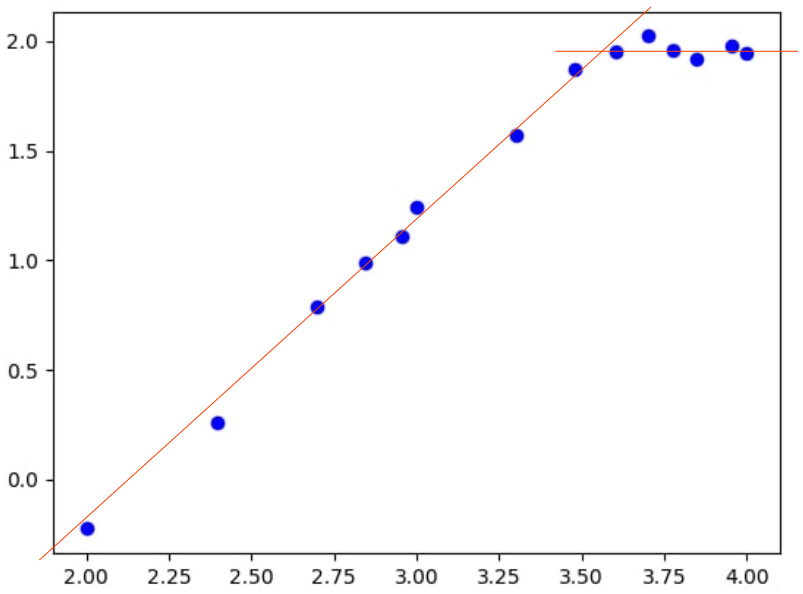
مارکر سایز را نیز در تعداد مختلف تغییر داده‌ام تا جواب‌ها زیباتر شوند!

کد را نیز می‌بینیم:

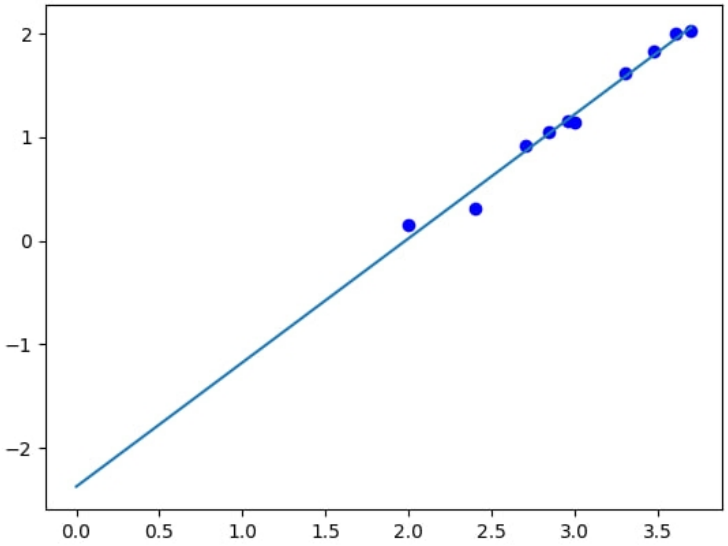
simulation > HW3 > herb.py > ...

```python
import matplotlib.pyplot as plt
import numpy as np
N=10000
L=200
markersize=1
base=np.arange(1,L+1)
height=np.zeros(L)
height[L//2]+=1
plt.plot(base[L//2],height[L//2],c='b',marker='s',markersize=markersize)
height[L//2]+=1
plt.plot(base[L//2],height[L//2],c='b',marker='s',markersize=markersize)
for i in range(1,N+1):
    rnd=np.random.randint(0,L)
    #print("ran: ",rnd+1)
    if height[rnd]==0 and height[rnd-1]==0 and height[(rnd+1)%L]==0:
        continue
    elif height[rnd]>=height[rnd-1] and height[rnd]>=height[(rnd+1)%L]:
        height[rnd]+=1
    else:
        arg=np.array([rnd-1,(rnd+1)%L])
        narr=np.array([height[rnd-1],height[(rnd+1)%L]])
        max=np.argmax(narr)
        nrnd=arg[max]
        height[rnd]=height[nrnd]
    #print(height)
    plt.plot(base[rnd],height[rnd],c='b',marker='s',markersize=markersize)
height1=height.copy()
for i in range(1,N+1):
    rnd=np.random.randint(0,L)
    if height[rnd]==0 and height[rnd-1]==0 and height[(rnd+1)%L]==0:
        continue
    elif height[rnd]>=height[rnd-1] and height[rnd]>=height[(rnd+1)%L]:
        height[rnd]+=1
    else:
        arg=np.array([rnd-1,(rnd+1)%L])
        narr=np.array([height[rnd-1],height[(rnd+1)%L]])
        max=np.argmax(narr)
        nrnd=arg[max]
        height[rnd]=height[nrnd]
    plt.plot(base[rnd],height[rnd],c='lightblue',marker='s',markersize=markersize)
height2=height.copy()
height2+=height1
for i in range(1,N+1):
    rnd=np.random.randint(0,L)
    if height[rnd]==0 and height[rnd-1]==0 and height[(rnd+1)%L]==0:
        continue
    elif height[rnd]>=height[rnd-1] and height[rnd]>=height[(rnd+1)%L]:
        height[rnd]+=1
    else:
        arg=np.array([rnd-1,(rnd+1)%L])
        narr=np.array([height[rnd-1],height[(rnd+1)%L]])
        max=np.argmax(narr)
        nrnd=arg[max]
        height[rnd]=height[nrnd]
    plt.plot(base[rnd],height[rnd],c='b',marker='s',markersize=markersize)
height3=height2+height

av_height1=sum(height1)/L
av_height2=sum(height2)/L
av_height3=sum(height3)/L
std_height1=np.var(height1)**0.5
std_height2=np.var(height2)**0.5
std_height3=np.var(height3)**0.5
print(av_height1,av_height2,av_height3)
print(std_height1,std_height2,std_height3)
print(height1,height2,height3)

plt.show()
```

حال در فایل herb-data.py تحلیل داده‌ها را دقیقا مانند سوال قبلی به ازای طول‌های متفاوت و تعداد نقاط رندوم متفاوت رسم کرده ام نقاط را و منحنی را کشیده‌ام:



که حالت اشباع آن به وضوح مشخص است. (خط قرمز کشیده‌ام!)
حال به قسمت خطی یک خط فیت می‌کنم.

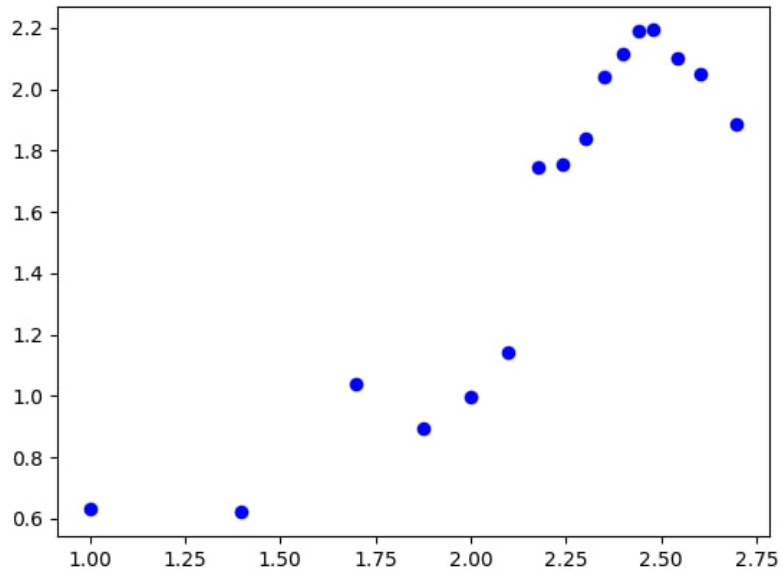


که معادله‌ی خط بدین صورت است:

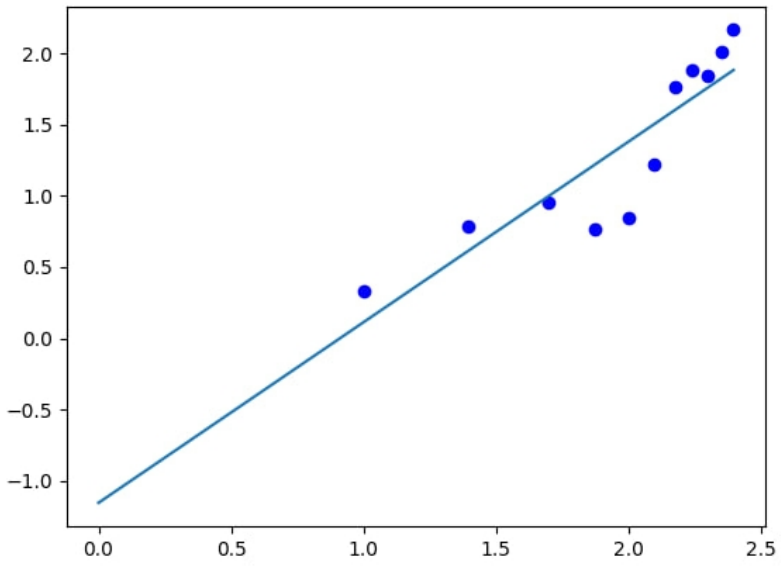
$$y = 1.199\ x - 2.37$$

که یعنی توان t عدد 1.199 است. چون شیب این خط توان تی را می‌دهد (لگاریتم انحراف معیار بر حسب لگاریتم زمان)

حال وابستگی به ال را بررسی می‌کنیم:



و برای قسمت اول که تقریبا می‌توان آن را خطی گرفت (!) توان L را بدست می‌آوریم:



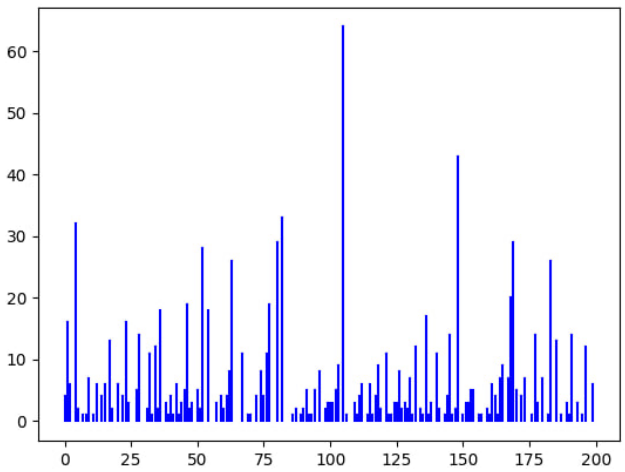
که معادله‌ی خط بدین صورت است:

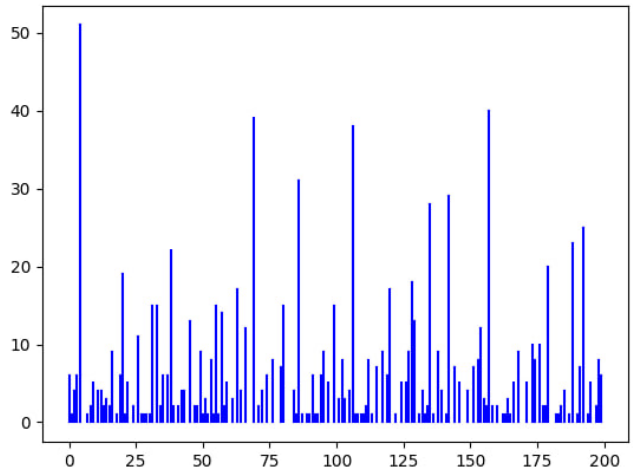
$$y = 1.26\ x - 1.15$$

که یعنی توان L عدد 1.26 است. چون شیب این خط توان L را می‌دهد (لگاریتم انحراف معیار بر حسب لگاریتم L)

تمرین 3.5 (رشد سوزنی):

در این سوال من یک زاویه تعریف کردم که از آن زاویه قرار است مثلا توپ‌هایی که قرار است پرتاب شوند، پرت شوند! من این‌گونه زدم که ابتدا ارتفاع ستون انتخاب شده (به صورت رندوم) را در نظر می‌گیرم. حال از زاویه‌ی angle درجه از انتهای سمت چپ کادر (قبل از ستون اول) این توپ را با این زاویه پرت می‌کنم. حال ارتفاع این توپ پرت شده را در ارتفاع‌های هر ستون داریم پس می‌توان دید آیا برخورد می‌کند یا نه. پس حال شرط را این‌گونه گذاشتم که اگر این ارتفاع بدست آمده (یک حلقه می‌زنیم تا تمامی ارتفاع ستون‌ها را تا ستون انتخاب شده چک کند) کمتر از ارتفاع توپ در آن ستون بود یک عدد به خود ستون رندوم انتخاب شد اضافه شود و اگر کمتر بود، به اولین ستون سمت چپ که به آن برخورد می‌کند، اضافه می‌شود. حال نتیجه را می‌بینیم:
برای 1000 نقطه دو نمونه می‌آورم: (زاویه نیز 60 درجه است)

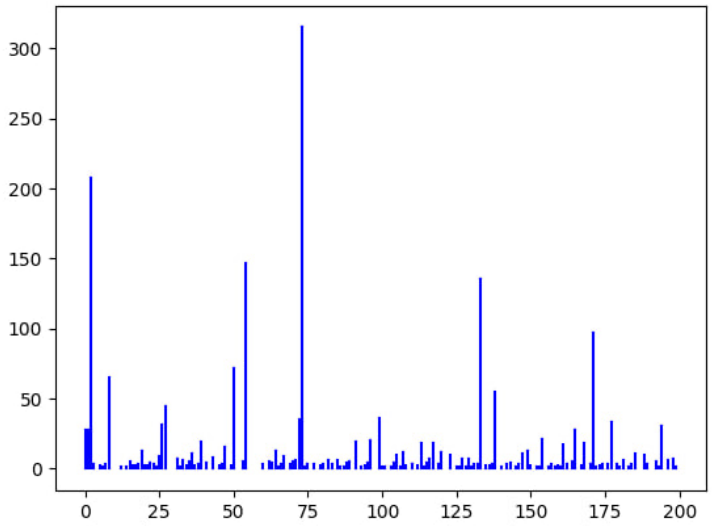




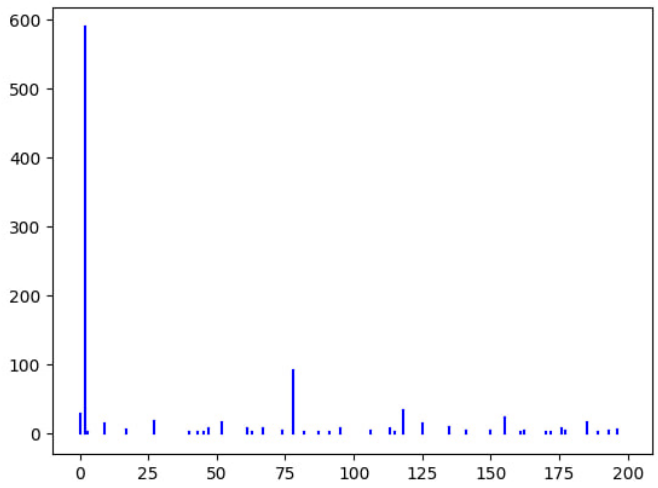
که برای داده‌های دوم انحراف معیار و میانگین بدین صورت است:

STD : 8.577 / Mean : 5

برای 2000 نقطه این اختلاف‌ها خیلی شدیدتر می‌شود (آنهایی که بالاتر بوده‌اند خیلی بالاتر می‌روند و پایینی‌ها پایین می‌مانند و رشد چندانی نمی‌کنند):



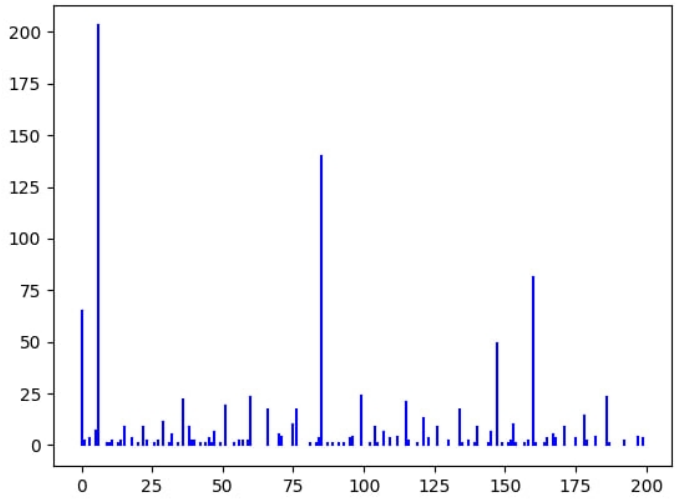
حال زاویه‌ی 30 درجه را در نظر می‌گیریم: (همان 1000 نقطه را می‌گیریم)



که بدیهی است چون زاویه کمتر شده اختلاف‌ها بیشتر می‌شود.
که برای 1000 تا داده و زاویه‌ی 30 درجه انحراف معیار و میانگین بدین صورت است:

STD : 37.78 / Mean : 5

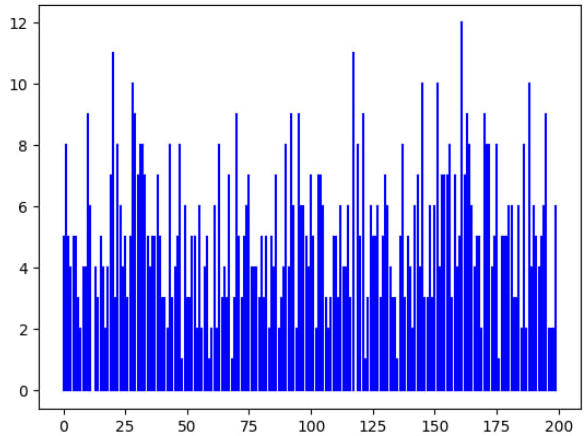
حال برای زاویه‌ی 45 درجه نیز که بین این است هم این کار را انجام می‌دهیم:



که برای 1000 تا داده و زاویه‌ی 45 درجه انحراف معیار و میانگین بدین صورت است:

STD : 19.36 / Mean : 5

حال اگر زاویه را 90 بگیریم انتظار داریم همان ول نشست را ببینیم که از شکل واضح است:



STD : 2.315
Mean : 5

simulation > HW3 > needle-growth.py > ...

```python
import matplotlib.pyplot as plt
import numpy as np
N=1000
L=200
markersize=10
angle=np.pi/6
base=np.arange(0,L)
height=np.zeros(L)
for j in range(0,N):
    rnd=np.random.randint(0,L)
    for i in range(0,rnd+1):
        height_ball_i=height[rnd]+((rnd-i)*np.tan(angle))
        #print(height_ball_i,height[i])
        if height_ball_i<=height[i]:
            height[i]+=1
            break
    #print(rnd,height)
for i in range(0,L):
    plt.plot([base[i],base[i]],[0,height[i]],'b')
av_height=sum(height)/L
std_height=np.var(height)**0.5
print('av: ',av_height,'  std: ',std_height)
plt.show()
```