گزارش تمرین سری ششم

شبیه سازی رایانه ای در فیزیک

علیرضا رضایی

97100762

1 فهرست

2 تمرین 6.1

2.1 مقدمه

نمودار توزیع فراوانی N عدد تولید شده با تابع np.random.randint که در بازه ی [0, 9] قرار دارند را رسم می کنیم.

این کار را چهار بار به ازای چهار N مختلف انجام می دهیم تا در نهایت بتوانیم نتیجه گیری بهتری داشته باشیم.

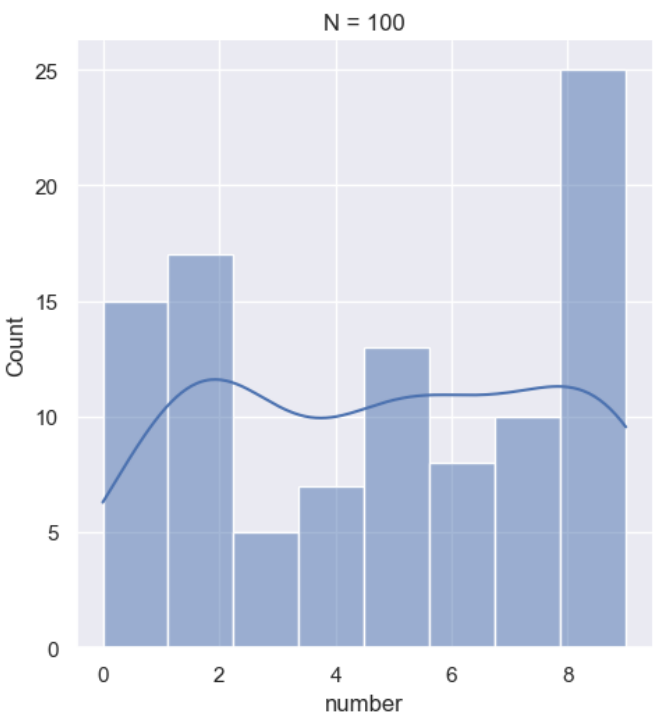
این دو کار در تابع generate انجام می شوند که به توضیح بیشتری نیاز ندارد.

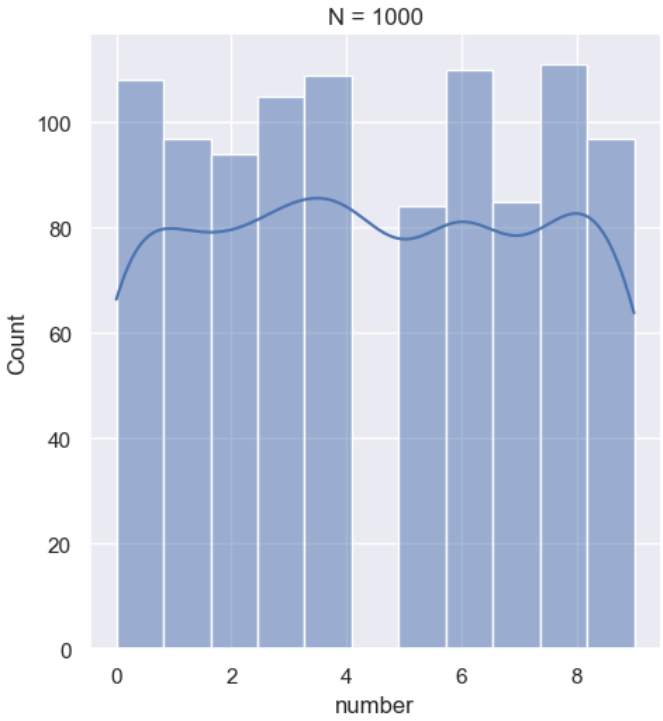
در قسمت بعدی سوال از ما میخواهد که انحراف معیار استاندارد را به ازای یک سری N بدست اورده و با $\frac{1}{\sqrt{N}}$ مقایسه کنیم.

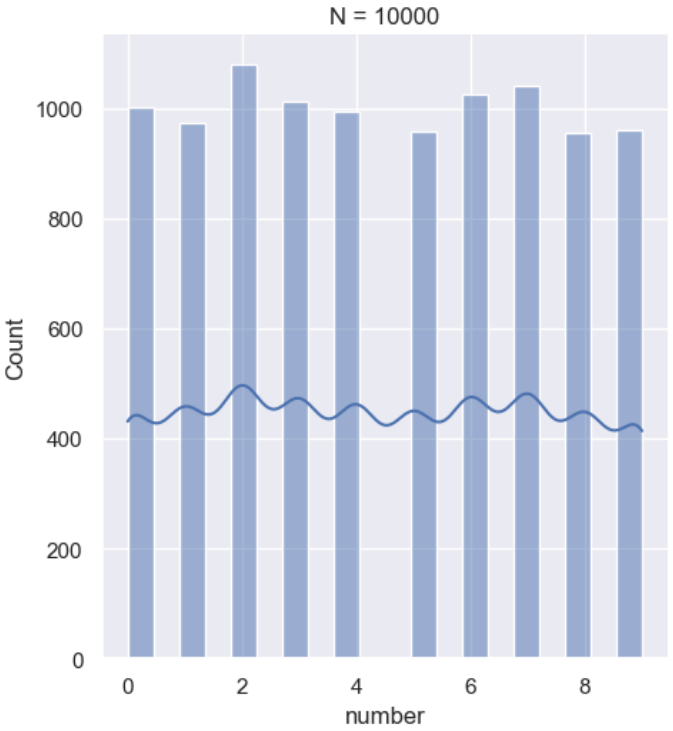
این کار را در تابع calculate انجام می دهیم که یک ورودی به اسم max_N از کار بر میگیریم و به ازای N های از 1 تا max_N نمودار های خواسته شده را در یک شکل رسم کرده و با هم مقایسه می کنیم.

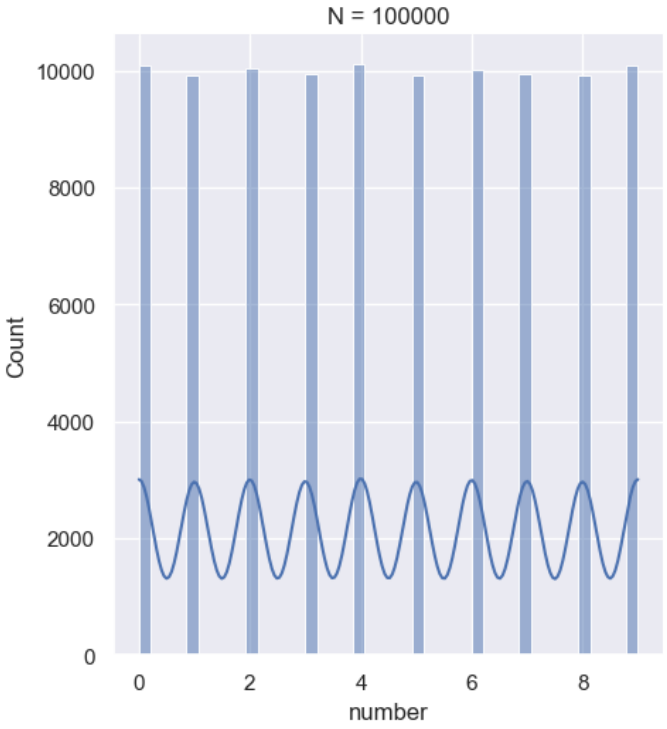
2.2 نتایج

در باره ی قسمت اول درباره ی N های کوچک این یکنواختی در توزیع مشاهده نمی شود ولی هرچه تعداد نمونه بیشتر می شود این یکنواختی بیشتر می شود و در N=100,000 همانطور که از تابع توزیع مشخص است تقریبا همه ی اعداد به یک تعداد تولید شده اند. (N/10)

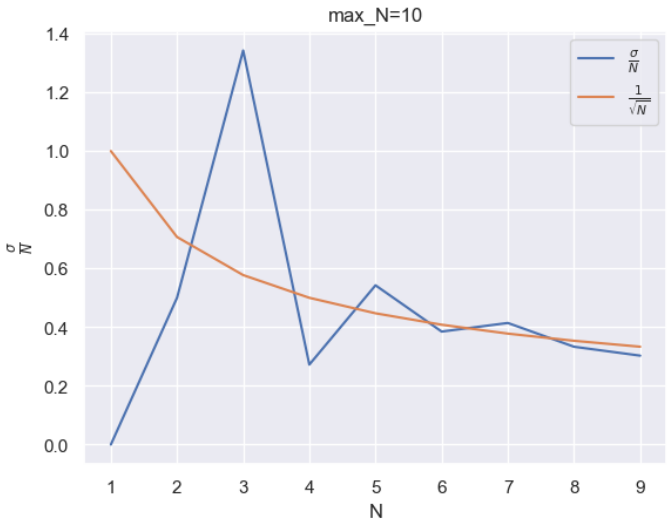


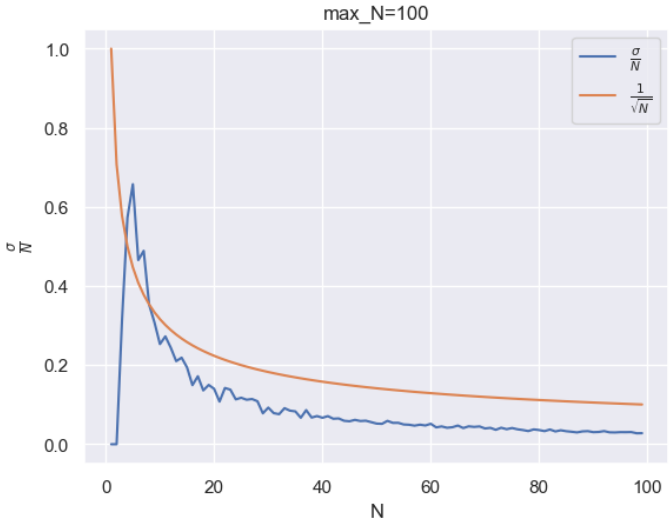


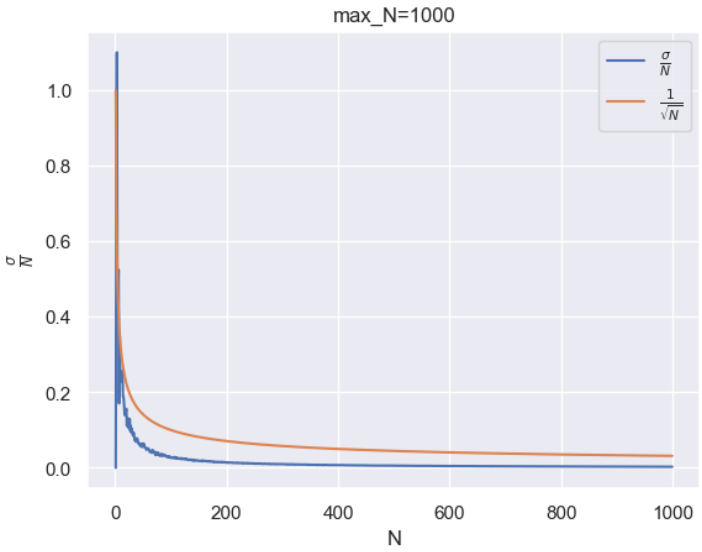


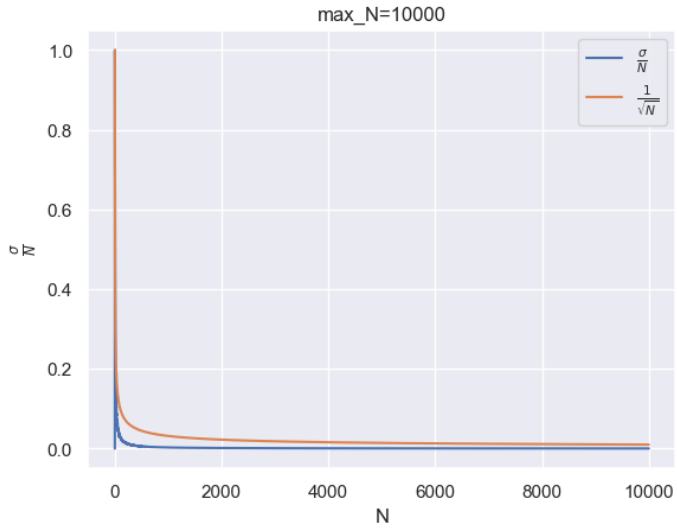


برای قسمت دوم هم نمودار های زیر را داریم که در این قسمت هم مشاهده میکنیم که هر چه max_N بزرگ تر شود به خواسته ی سوال نزدیک تر می شویم:









در باره ی شباهت این تمرین و تمرین ول نشست هم باید گفت که بله نمودار های قسمت اول این تمرین دقیقا مثل نمودار های تمرین ول نشست اند که بعد از مدتی ارتفاع همه ی خانه ها تقریبا برابر است چون احتمال نشستن ذره در هر خانه برابر با بقیه ی خانه هاست.

3 تمرین 6.2

3.1 مقدمه

میخواهیم همان تمرین قبل را تکرار کنیم فقط با این تفاوت که این بار فقط اعدادی را بر میداریم که عدد قبلی آنها چهار بوده باشد و نمودار تابع توزیع این اعداد را رسم می کنیم.

کار اصلی در تابع generate انجام می شود پس به توضیح این تابع بسنده می کنیم.

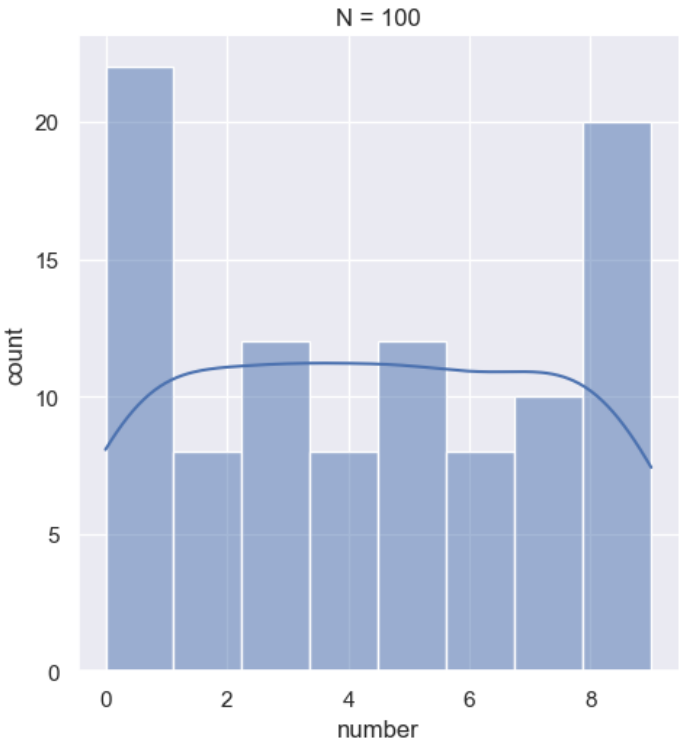
در این تابع یک حلقه را N بار موثر تکرار می کنیم که N تعداد اعداد تصادفی ایست که می خواهیم داشته باشیم.

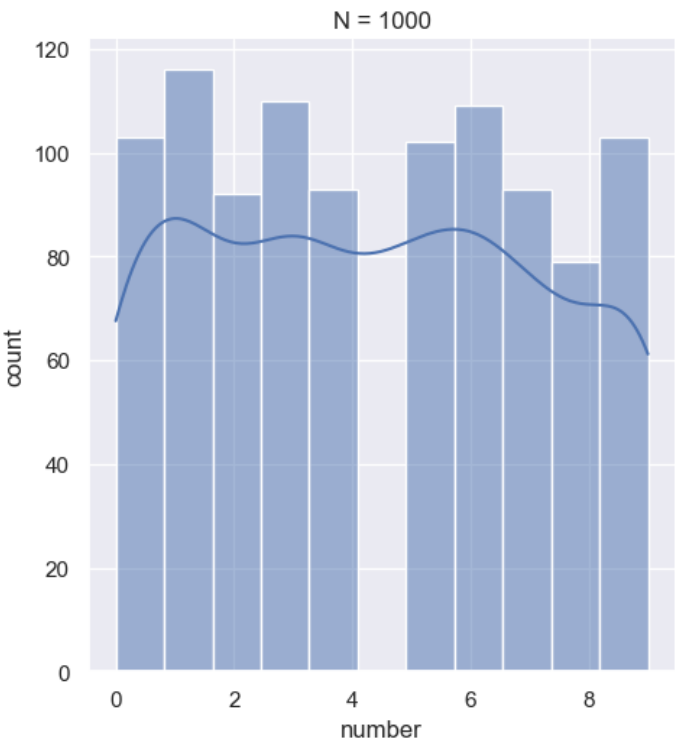
در هر قدم یک عدد تصادفی تولید میکنیم ، اگر این عدد چهار بود یک عدد تصادفی دیگر تولید کرده و در آرایه ای که از قبل ساخته ایم ذخیره می کنیم و این گام حلقه را یک گام موثر به حساب اورده و یک واحد به شمارنده ی حلقه اضافه می کنیم. ولی اگر عدد اول تولید شده چهار نبود بدون تغییر در شمارنده ی حلقه یک بار دیگر حلقه را از اول آغاز می کنیم و یک عدد تصادفی دیگر تولید می کنیم و چک میکنیم که آیا چهار است یا نه که بر اساس آن بقیه ی کار را ها مطابق قبل انجام دهیم.

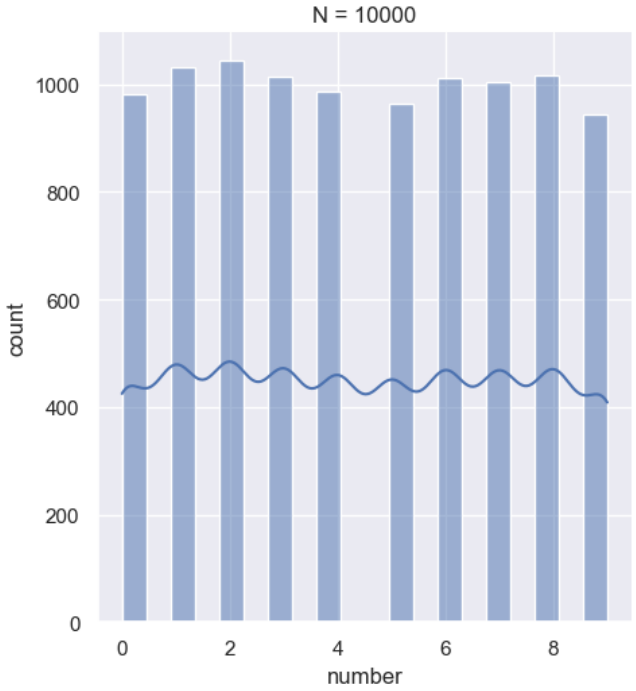
در نهایت N عدد تصادفی با شرایط خواسته شده داریم که می توانیم نمودار تابع توزیع آنها را رسم کنیم.

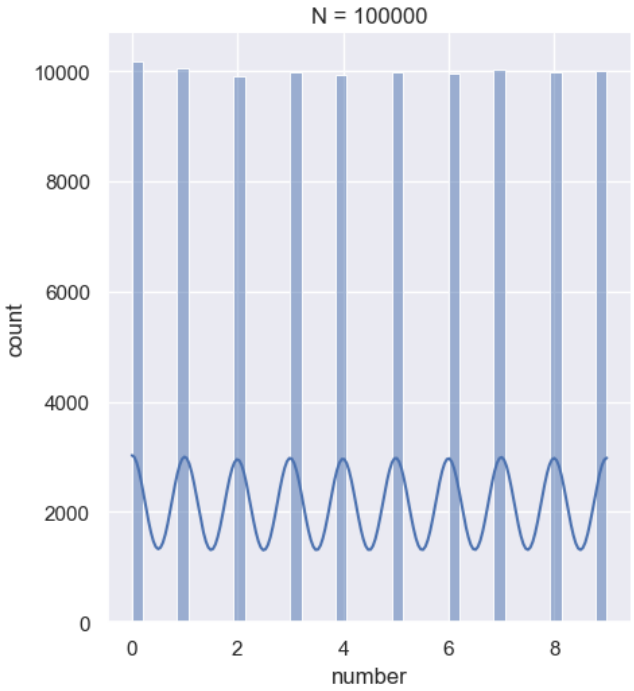
3.2 نتایج

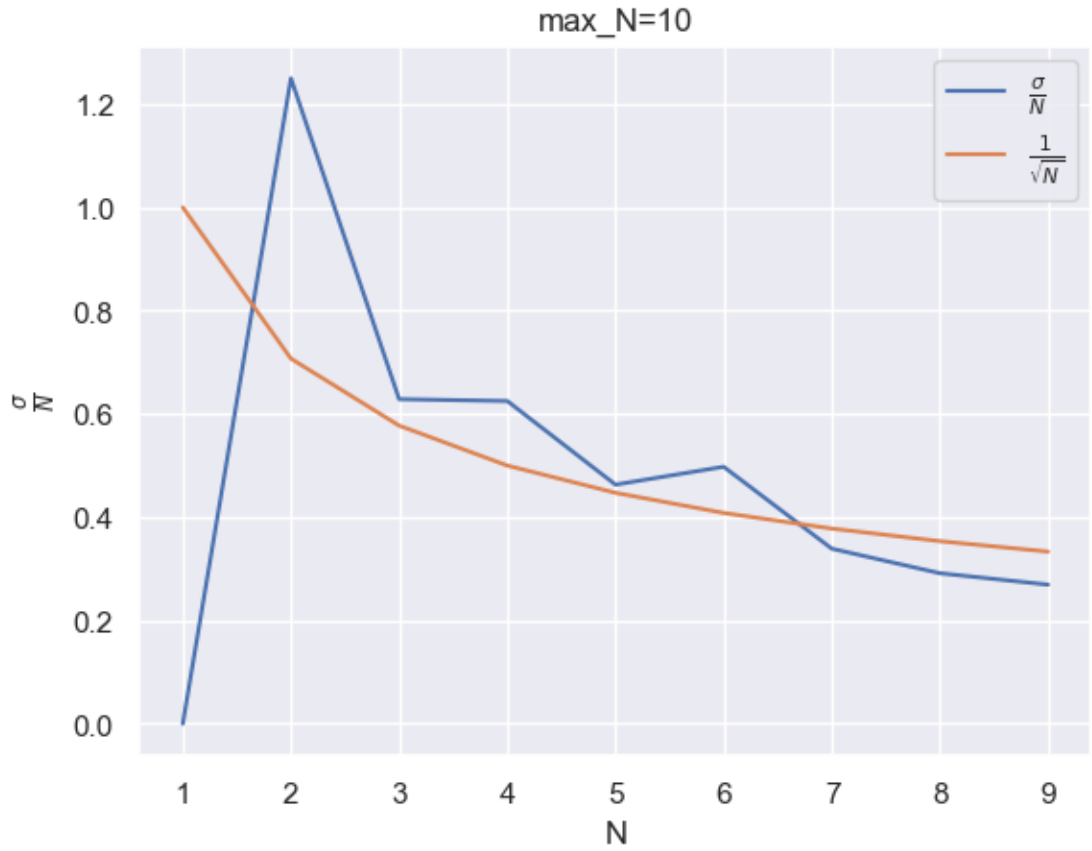
در این تمرین هم مشاهده می کنیم که هر چه تعداد نمونه ها (N) بیشتر شود تابع توزیع یکنواخت تر می شود و در N های بزرگ می توان گفت تقریبا یکنواخت است.

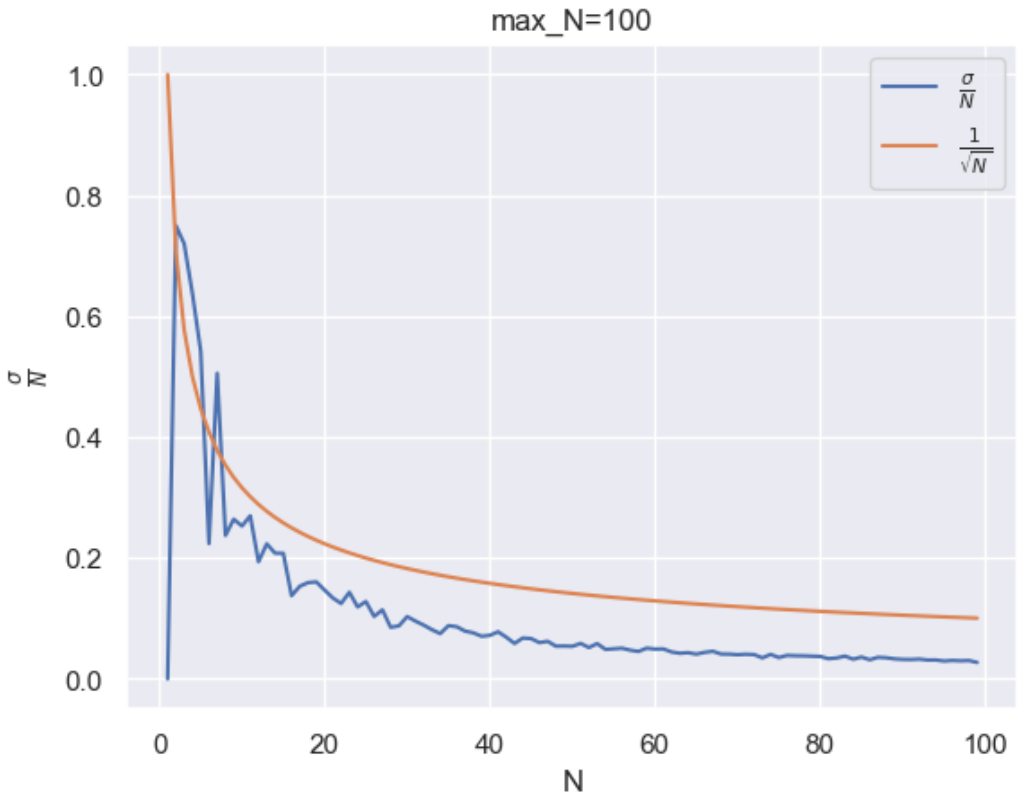


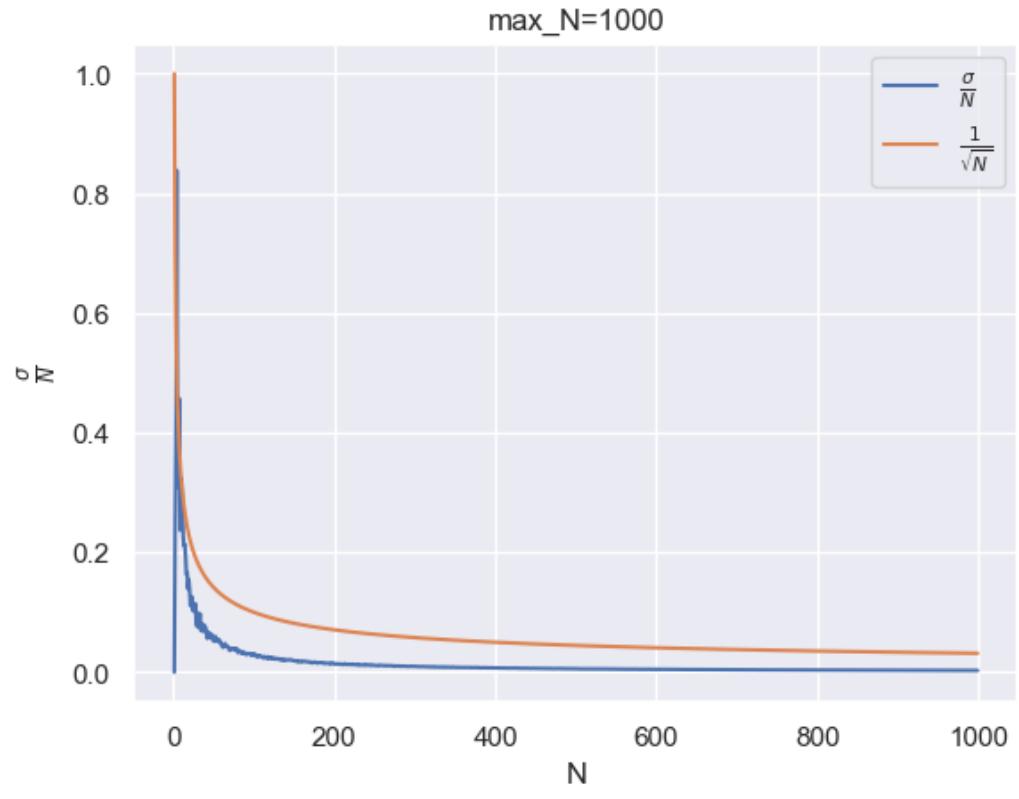



N = 10000
count
number
0
200
400
600
800
1000
0
2
4
6
8


N = 100000
count
number
0
2000
4000
6000
8000
10000
0
2
4
6
8


max_N=10
$\frac{\sigma}{N}$
$\frac{1}{\sqrt{N}}$
$\frac{\sigma}{N}$
N
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1
2
3
4
5
6
7
8
9


max_N=100
1.0
0.8
0.6
0.4
0.2
0.0
$\frac{\sigma}{N}$
0
20
40
60
80
100
N
$\frac{\sigma}{N}$
$\frac{1}{\sqrt{N}}$

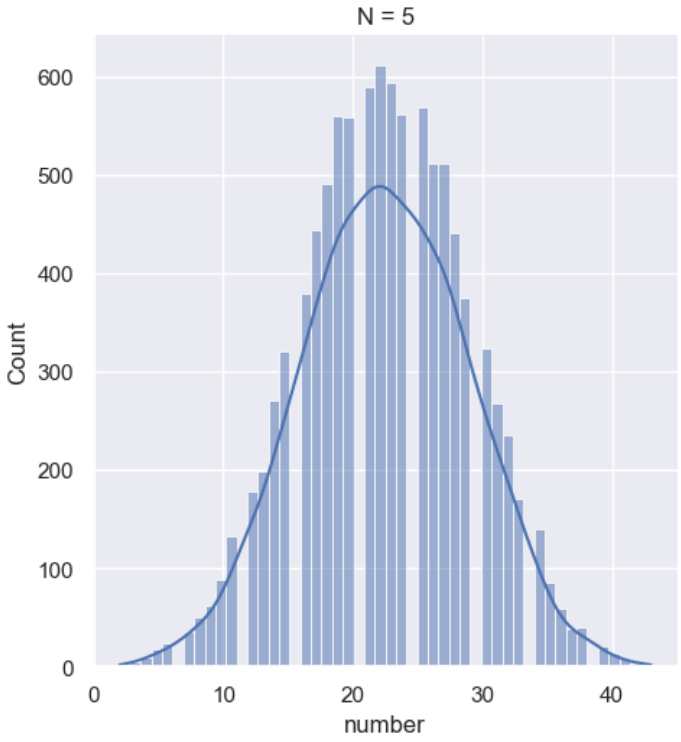

max_N=1000
1.0
0.8
0.6
0.4
0.2
0.0
$\frac{\sigma}{N}$
0
200
400
600
800
1000
N
$\frac{\sigma}{N}$
$\frac{1}{\sqrt{N}}$

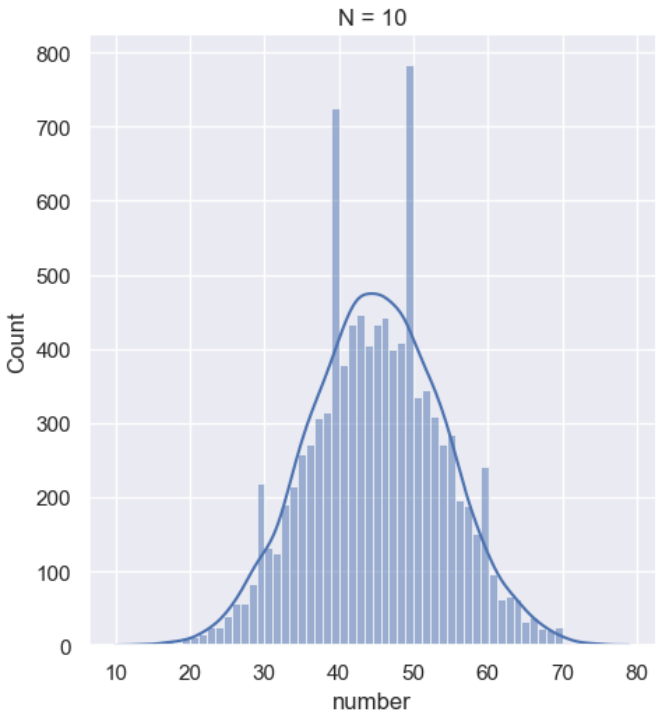
4 تمرین 6.3

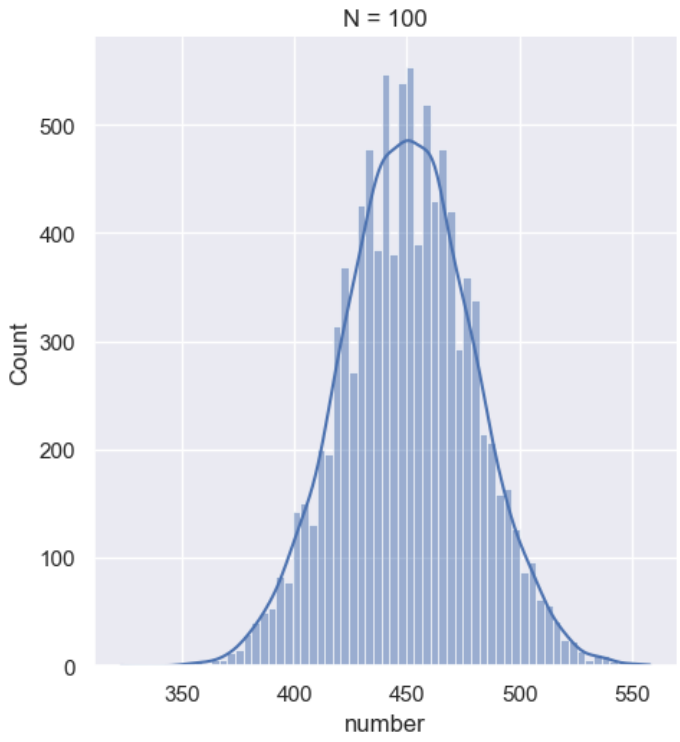
4.1 مقدمه

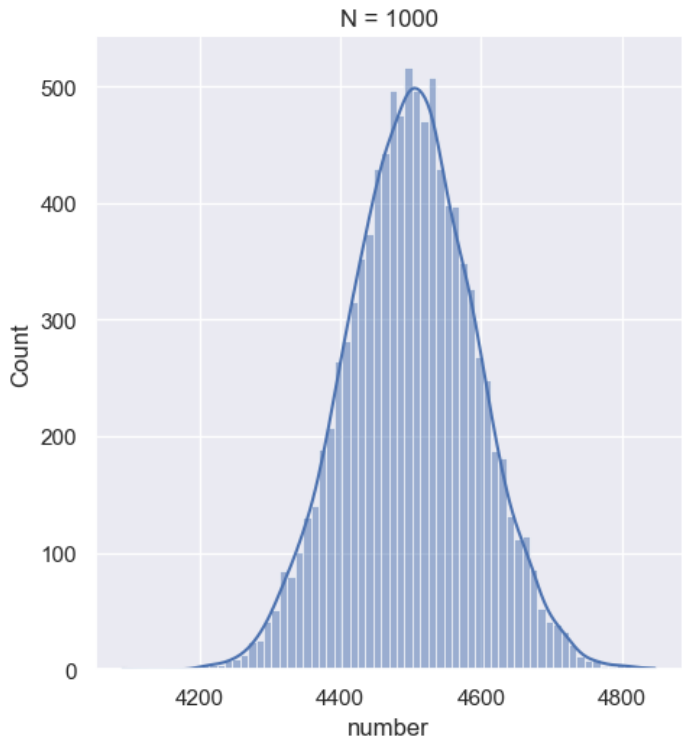
در این تمرین می خواهیم صحت قضیه ی حد مرکزی را بررسی کنیم یعنی تعدادی عدد تصادفی تولید کنیم و N تا N تا آنها را با هم جمع کنیم و ببینیم که آیا تابع توزیع جدید ، تابع توزیعی گاوسی می شود یا خیر.

برای این کار number_of_samples*N عدد تصادفی تولید می کنیم و سپس آنها را در ماتریسی متشکل از number_of_samples سطر و N ستون قرار می دهیم و سپس اعداد موجود در هر سطر را با هم جمع می زنیم و این گونه number_of_samples عدد داریم که هر کدامشان مجموع N عدد تصادفی دیگرند.









صحت و دقت قضیه ی حد مرکزی را می توانیم در نتایج بدست آمده مشاهده کنیم.

در باره ی شباهت این تمرین و تمرین ول گشت و ول نشست می توان گفت که در ولگشت هم اگر بخواهیم نمودار مکان نهایی ولگرد را بعد از تعداد گام N رسم کنیم آن نمودار هم نموداری گاوسی خواهد شد چون عملا جمع N عدد رندوم است که طبق این تمرین و قضیه حد مرکزی می دانیم که تابع توزیع مجموع N عدد رندوم تابع توزیعی گاوسی خواهد شد.

5 تمرین 6.4

5.1 مقدمه

در این قسمت می خواهیم تابع تبدیلی بدست بدست بیاوریم که اگر آن را بر اعداد رندوم بدست آمده از تابع توزیع یکنواخت اثر دهیم و تابع توزیع اعداد بدست آمده ی جدید را رسم کنیم ، نمودار حاصل نموداری گاوسی باشد.

برای قسمت کد ، توضیح خاصی لازم نیست و تمام توضیحات تئوری هم در قسمت محاسبات (بخش بعد) موجود است.

البته میتوان به طور کلی توضیحاتی هم در مورد کد به این صورت داد که:

با هر بار صدا زدن تابع draw این تابع مقادیر سیگما و تعداد نمونه ها را می گیرد و دو توزیع یکنواخت با این پارامتر ها با نام های x1 , x2 رسم می کند و از روی آن ها با کمک تابع تبدیل ، تابع توزیع های گاوسی را می سازد و رسم می کند.

5.1.1 محاسبات

$-\frac{9}{4}$

ص ۱ از نظیری ۲۲ کتاب داریم:

$$f(r,\theta)=\frac{1}{2\pi\sigma^2}e^{-\frac{r^2}{2\sigma^2}}r\,dr\,d\theta$$

$f_r(r)$ و $f_\theta(\theta)$ را اینگونه انتخاب می‌کنیم:

$$f_r(r)=\frac{1}{\sigma^2}e^{-\frac{r^2}{2\sigma^2}}r$$

$$f_\theta(\theta)=\frac{1}{2\pi}$$

که این‌ها تابع توزیع‌هایی در مختصات قطبی‌اند. حال با کمک هرکدام از این‌ها سعی می‌کنیم توزیع‌هایی برای θ و r پیدا کنیم:

$$q_1=\int_0^{\theta}f_\theta(\theta)\,d\theta=\frac{\theta}{2\pi}\Rightarrow q_1=\frac{\theta}{2\pi}\Rightarrow \boxed{\theta=2\pi q_1}$$

$$q_2=\int_0^{r}f_r(r)\,dr=\int_0^{r}\frac{1}{\sigma^2}e^{-\frac{r^2}{2\sigma^2}}r\,dr$$

با تغییر متغیر داریم:

$$u=-\frac{r^2}{2\sigma^2}\Rightarrow du=-\frac{2r}{2\sigma^2}dr\Rightarrow dr=-\frac{\sigma^2}{r}du$$

$$\Rightarrow q_2=-\int e^u du=-e^u\Rightarrow \boxed{q_2=1-e^{-\frac{r^2}{2\sigma^2}}}$$

$$e^{-\frac{\rho^2}{2\sigma^2}} = 1-u_2 \xrightarrow{Ln} -\frac{\rho^2}{2\sigma^2} = Ln\,1-u_2$$

$$\Rightarrow \boxed{\rho^2 = -2\sigma^2 Ln\,1-u_2 = -2\sigma^2 Ln\,u_2}$$

بخاطر نوع تابع توزیع u_2 که یکنواخت است و از ۰ تا ۱

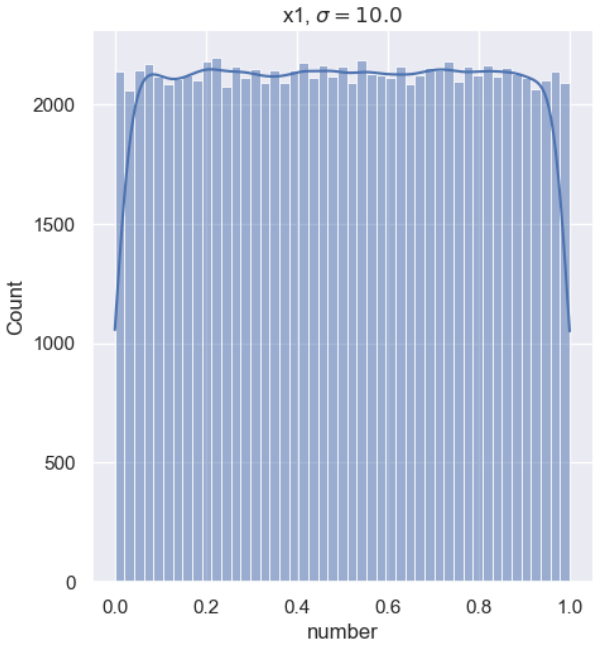
حال دو توزیع θ و r را داریم که با تابع توزیع های θ و r بدست آمده اند.

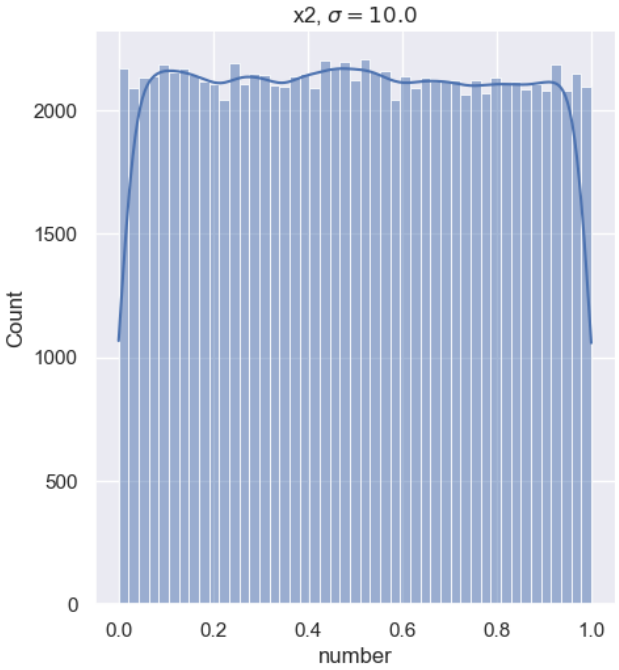
پس کافی است که دستگاه معنصات را از قطبی به دکارتی ببرد:

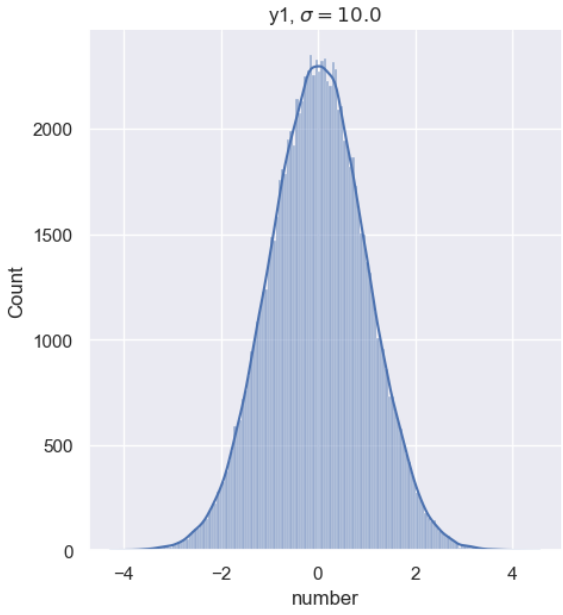
$$\begin{cases} x_1 = \rho \sin\theta \\ x_2 = \rho \cos\theta \end{cases} \Longrightarrow \text{که} \begin{cases} \rho = \sqrt{-2\sigma^2 Ln\,u} \\ \theta = 2\pi u \end{cases}$$

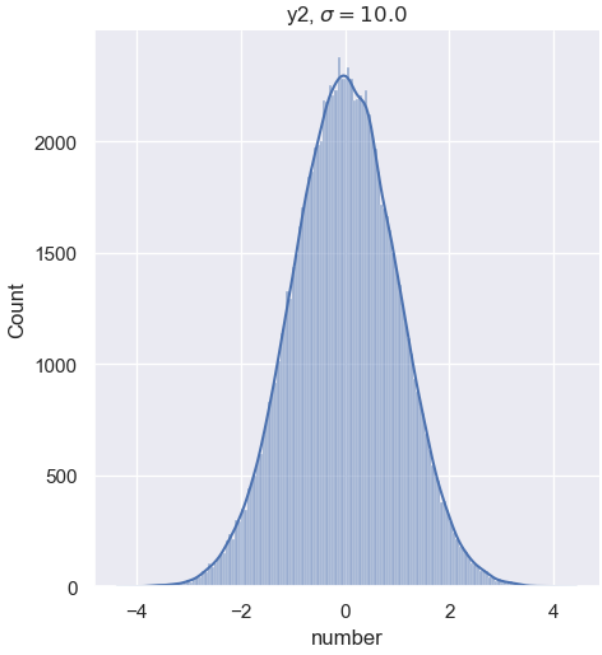
5.2 نتایج

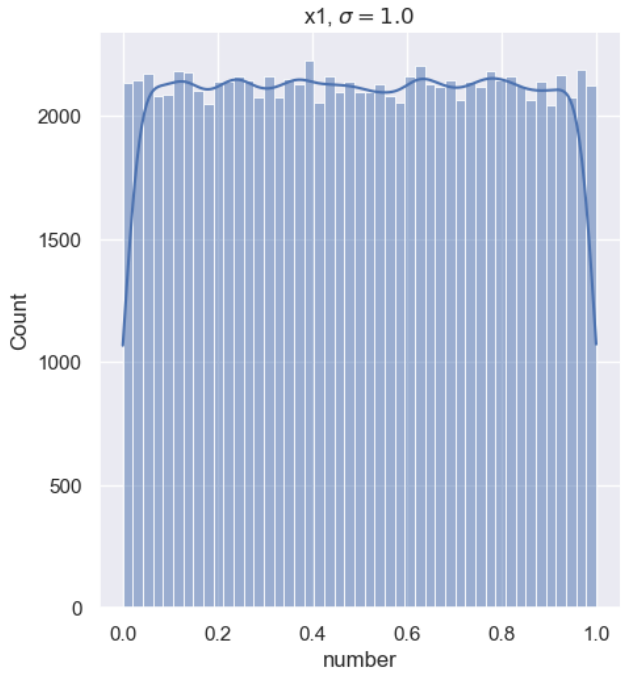
- تعداد کل نمونه ها در همه ی این نتایج: 100,000 تا
- برای 3 سیگمای مختلف هم نتایج رسم شده اند.

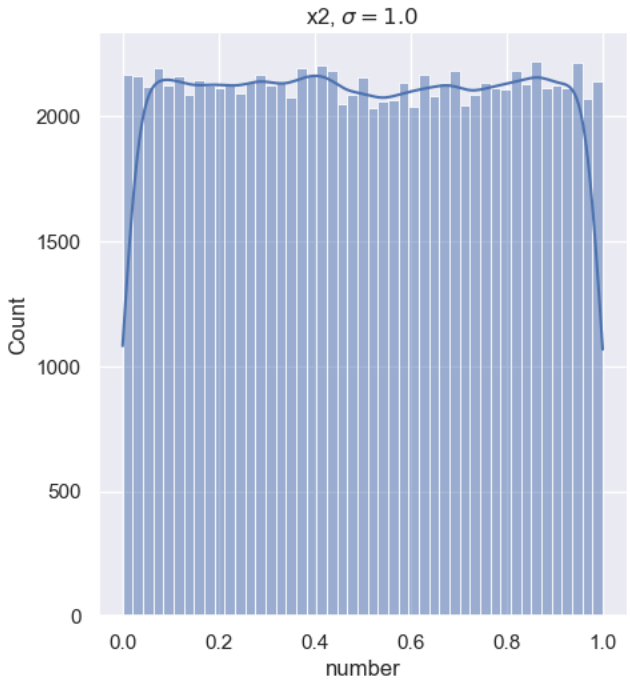


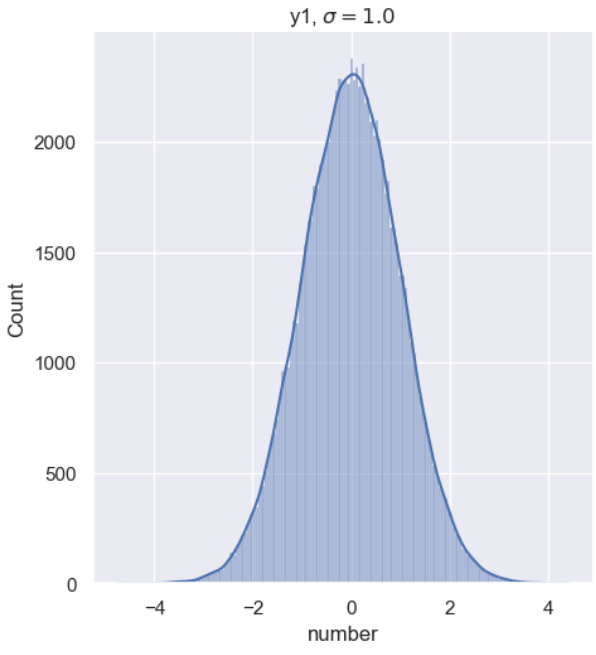


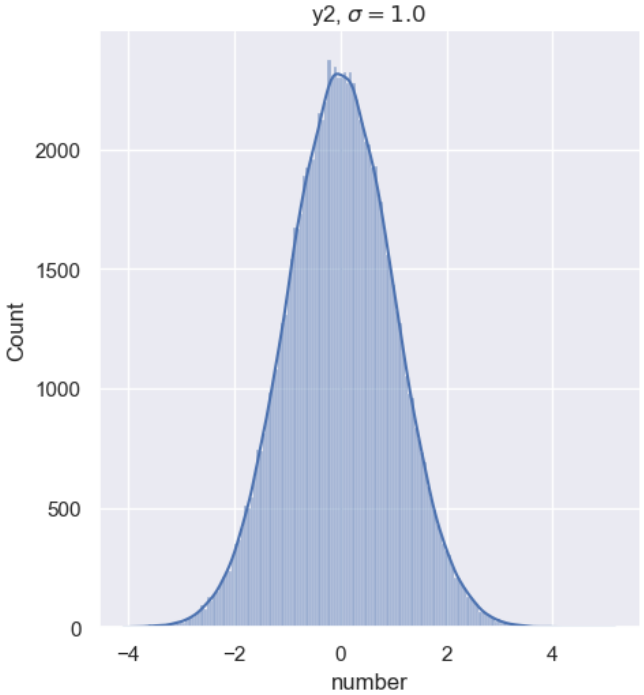


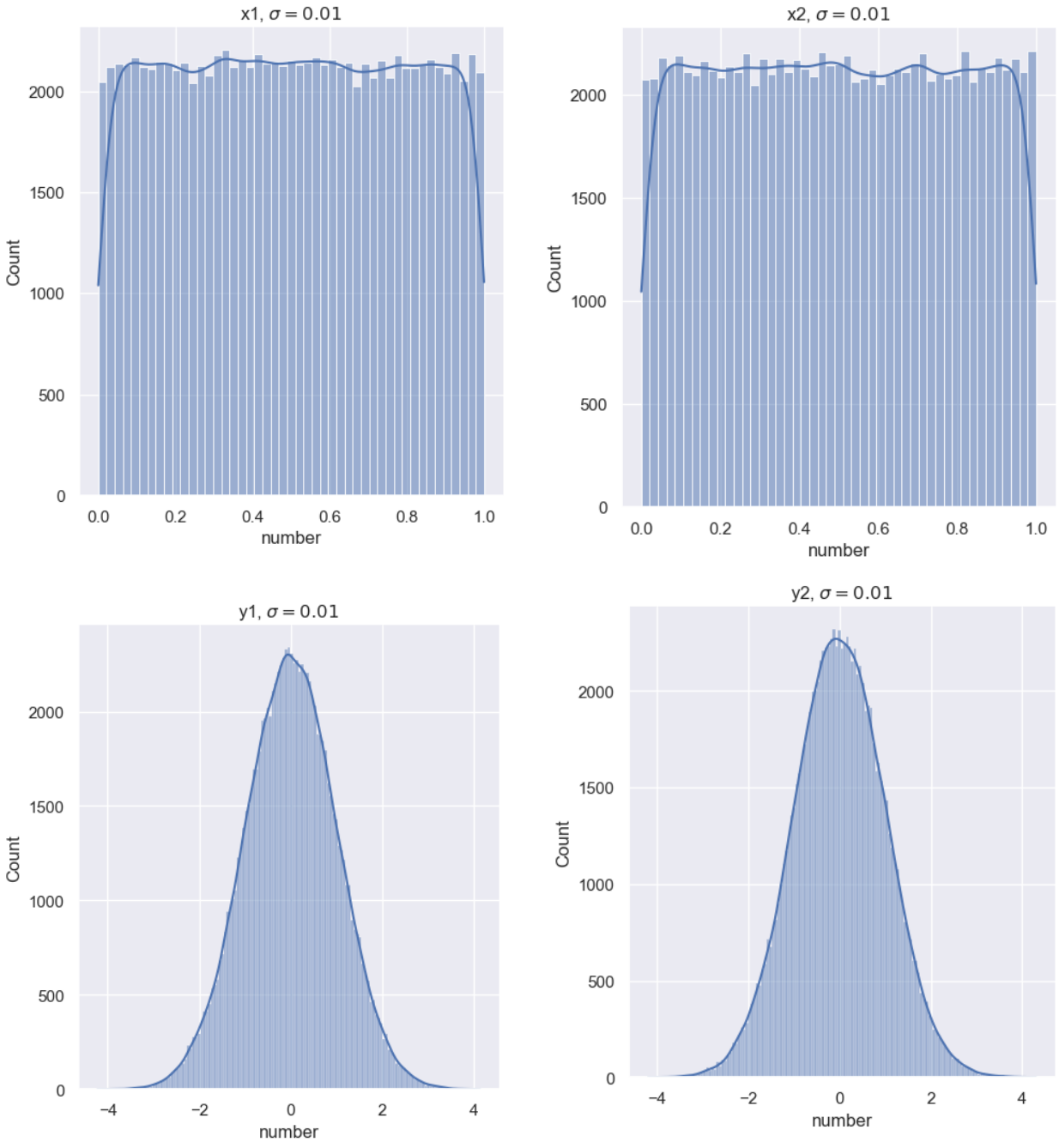



x1, $\sigma = 1.0$
Count
number
0
500
1000
1500
2000
0.0
0.2
0.4
0.6
0.8
1.0


x2, $\sigma = 1.0$
Count
number
0
500
1000
1500
2000
0.0
0.2
0.4
0.6
0.8
1.0


y1, $\sigma = 1.0$
Count
number
0
500
1000
1500
2000
−4
−2
0
2
4


y2, $\sigma = 1.0$
Count
number
0
500
1000
1500
2000
−4
−2
0
2
4



همانطور که مشاهده می کنید هر دو توزیع y1, y2 گاوسی شده اند و تغییر سیگما هم تاثیری در اصل گاوسی شدنشان نداشته و فقط شکل تابع گاوسی را کمی تغییر داده.

6 تمرین 7.1

6.1 مقدمه

در این تمرین میخواهیم انتگرال خواسته شده در صورت سوال را به روش های نمونه برداری ساده و هوشمند بدست آوریم و نتایج این دو روش را با هم مقایسه کنیم.

در کد سه تابع برای انتگرال گیری داریم:

1- simple_monte_carlo برای انتگرال گیری مونت کارلو به روش نمونه بر داری ساده
2- important_sampling_monte_carlo برای انتگرال گیری مونت کارلو به روش نمونه برداری هوشمند
3- scipy_integral برای انتگرال گیری به کمک کتابخانه ی scipy

در تابع اول یک سری عدد رندوم با تابع توزیع یکنواخت در بازه ی (2 ,0] تولید می کنیم (x) و میانگین f(x) ها را ضربدر طول بازه (2) می کنیم و به عنوان نتیجه ی انتگرال خروجی می دهیم.

در تابع دوم هم یک سری عدد رندوم با تابع توزیع یکنواخت در بازه ی (1 ,0] تولید می کنیم و با کمک تابع تبدیلمان عدد های y را با تابع توزیع گفته شده در صورت سوال در بازه ی (2 ,0] تولید کرده و مقدار (f(y)/g(y)).mean() را به عنوان جواب انتگرال خروج می دهیم که g(y) تابع توزیع گفته شده در سوال است و دلیل این جواب هم به این صورت اثبات می شود:

(دقت می کنیم که به علت نرمالیزاسیون ، انتگرال g(x) در بازه ی [2 ,0] باید برابر با 1 باشد.)

۱۷- بخش اول:

$$\int_{-\infty}^{\infty} f(x)\,dx = 1 = \int_{0}^{2} Ae^{-x}$$

$$= -Ae^{-x}\Big|_0^2 = A\left(1 - e^{-2}\right) = 1 \Rightarrow \boxed{A = \frac{1}{1-e^{-2}}}$$

حال به کمک تابع توزیع x های جدید تولید می‌کنیم. (از همان x جدید است):

$$\int_0^y Ae^{-x}\,dx = -Ae^{-x}\Big|_0^y = -Ae^{-y} + A = x$$

$$\Rightarrow 1 - e^{-y} = \frac{x}{A} \Rightarrow e^{-y} = 1 - \frac{x}{A} \Rightarrow -y = \ln\left(1 - \frac{x}{A}\right)$$

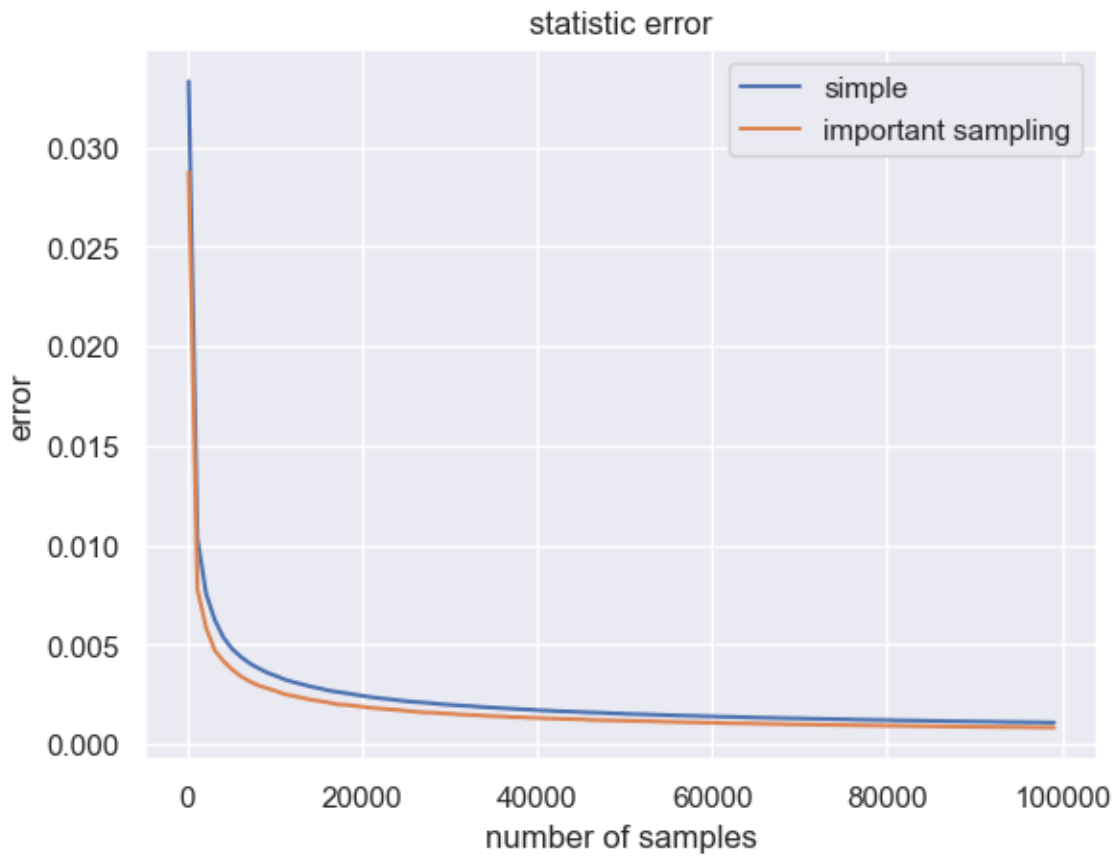
$$\Rightarrow \boxed{y = -\ln\left(1 - \frac{x}{A}\right)}$$

در این حل فرض شده که x از تابع توزیع یکنواخت در بازه‌ی $[0, 1]$ آمده است.

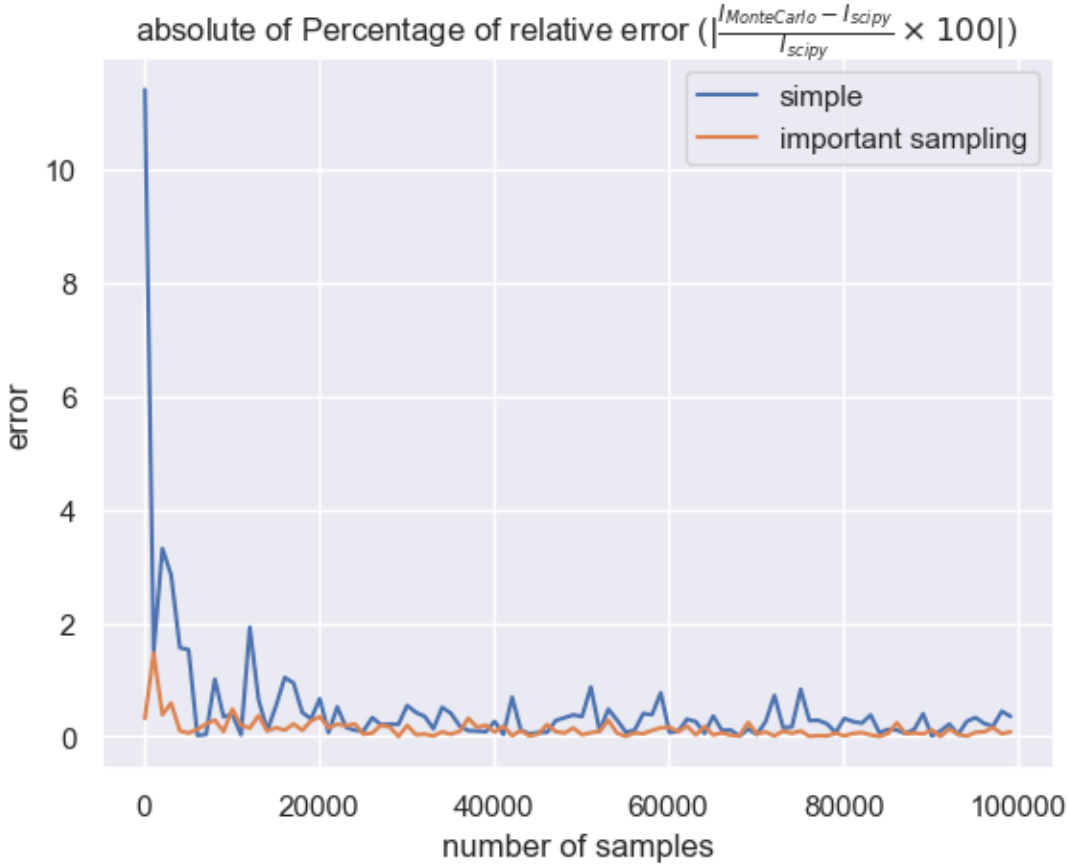
در قسمت test the random generator هم رندوم جنریتورمون رو تست می کنیم که آیا تابع توزیعی که میخواهیم را برایمان تولید می کند یا خیر.

در تابع main هم با کمک توابع قبلی محاسبات انجام شده و نمودار ها رسم می شوند و به توضیح اضافه ای نیاز نیست.

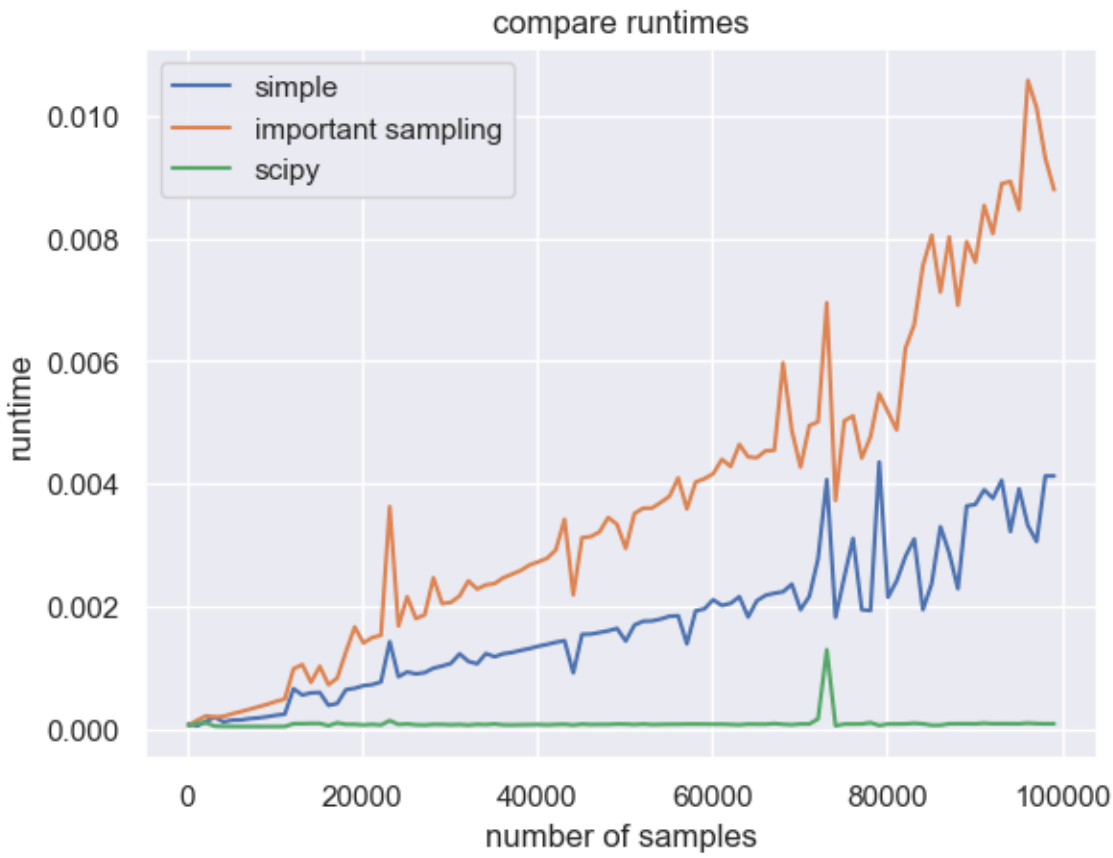
6.2 نتایج



همانطور که مشاهده می کنید خطای آماری روش نمونه برداری هوشمند به طور کلی کمتر از روش نمونه برداری ساده است.

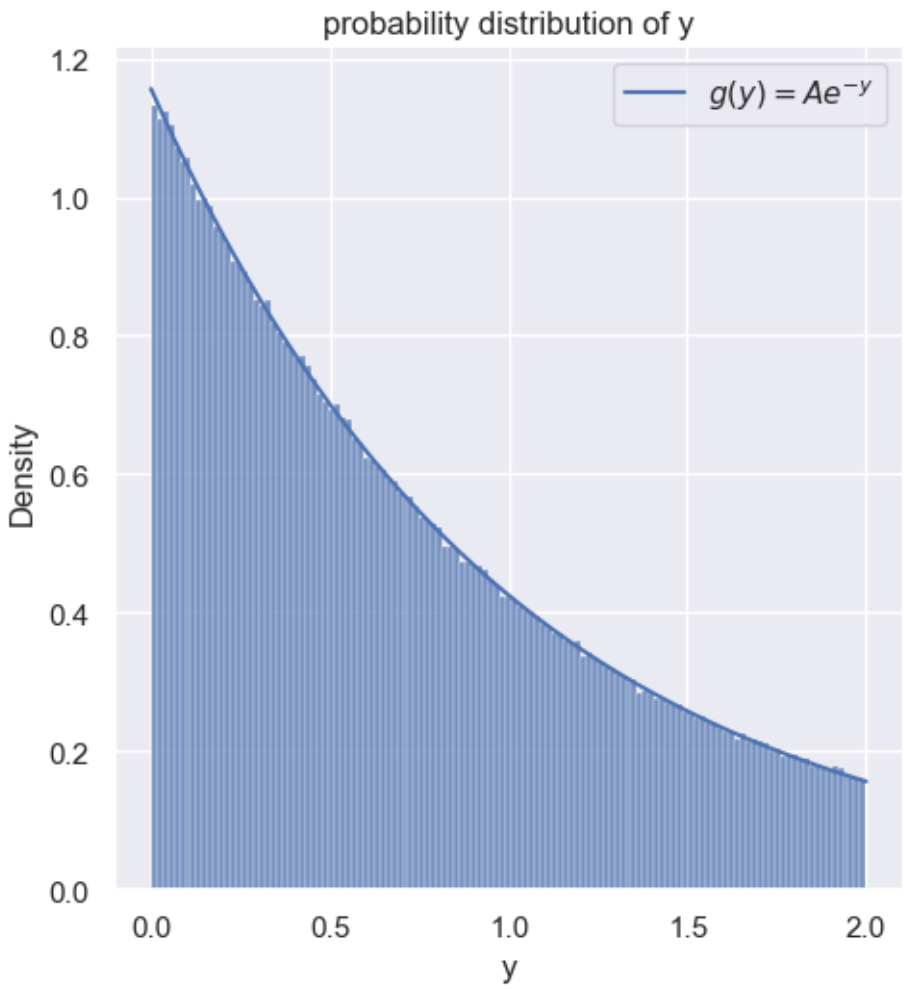


در مورد خطای واقعی هم خطای نمونه برداری هوشمند با تعداد کمتری نمونه به صفر همگرا می شود.



در مورد زمان اجرا هم همانطور که می توان انتظار داشت به ازای تعداد برابری از نمونه ، نمونه برداری هوشمند زمان بیشتری برای محاسبه ی انتگرال نیاز دارد چون یک مرحله برای بدست آوردن تابع توزیع جدید از تابع توزیع قبلی نیاز دارد که این خودش به یک زمانی برای محاسبه احتیاج دارد ولی از آنجایی که مقدار انتگرال در نمونه برداری هوشمند به ازای تعداد کمتری نمونه همگرا می شود ، میتواند الگوریتم بهینه تری باشد و حتی در زمان کمتری ما را به مقدار انتگرال و خطای مطلوب برساند.

کتابخانه ی سایپای هم چون هیچ پارامتری ازش تغییر نمی کند تقریبا به ازای هر تعداد نمونه (تعداد نمونه بر آن اثر نمی گذارد و خودش کار خودش را می کند) در یک زمان ثابتی خروجی را به ما می دهد. (آن نوسان ها در سرعتش هم می توانند به علت تغییر سرعت محاسبات در سیستم ما باشند.)



نتیجه ی تست تابع توزیع هم حاکی از این است که محاسباتمان برای تابع تبدیل به درستی انجام شده است.

[36]:
```
data.head(12)
```

[36]:

	تعداد نمونه ها	مقدار واقعی انتگرال	مقدار انتگرال با نمونه گیری ساده	مقدار انتگرال با نمونه گیری هوشمند	خطای آماری نمونه گیری ساده	خطای آماری نمونه گیری هوشمند	خطای واقعی نمونه گیری ساده	خطای واقعی نمونه گیری هوشمند	زمان اجرای نمونه گیری ساده	زمان اجرای نمونه گیری هوشمند
0	100	0.882081	0.781425	0.879273	0.033304	0.028750	11.411208	0.318393	0.000081	0.000056
1	1100	0.882081	0.868856	0.895039	0.010350	0.007785	1.499380	1.469034	0.000048	0.000139
2	2100	0.882081	0.911339	0.878681	0.007568	0.005862	3.316918	0.385444	0.000122	0.000210
3	3100	0.882081	0.907193	0.887267	0.006243	0.004715	2.846874	0.587935	0.000187	0.000189
4	4100	0.882081	0.868268	0.881208	0.005355	0.004174	1.566020	0.098959	0.000111	0.000202
5	5100	0.882081	0.868586	0.881551	0.004784	0.003753	1.529904	0.060149	0.000141	0.000245
6	6100	0.882081	0.882265	0.883220	0.004391	0.003412	0.020812	0.129120	0.000143	0.000279
7	7100	0.882081	0.882420	0.880140	0.004071	0.003162	0.038386	0.220102	0.000164	0.000319
8	8100	0.882081	0.873147	0.884638	0.003821	0.002957	1.012857	0.289831	0.000177	0.000358
9	9100	0.882081	0.885144	0.881318	0.003599	0.002817	0.347193	0.086523	0.000196	0.000399
10	10100	0.882081	0.878607	0.877826	0.003439	0.002676	0.393877	0.482406	0.000220	0.000445
11	11100	0.882081	0.881826	0.883846	0.003257	0.002521	0.028976	0.200012	0.000238	0.000486

7 تمرین 7.2

7.1 مقدمه

در این تمرین میخواهیم مرکز جرم کره ای که چگالی جرمی اش در راستای عمودی از بالا تا پایین به صورت خطی کم می شود را حساب کنیم.

طبق شهودی که از مسئله داریم و تقارن موجود در آن ، مرکز جرم قطعا بر روی محور Z ها قرار دارد پس کارمان ساده تر شد و فقط نیاز داریم که محل آن را روی محور Z بدست بیاوریم:

$$Z_{com} = \frac{\iiint z\rho(z)\, dxdydz}{\iiint \rho(z)\, dxdydz}$$

پس باید مقدار دو انتگرال را در حجم کره حساب کرده و بر هم تقسیم کنیم.

برای این کار از الگوریتم مونت کارلو به روش نمونه برداری ساده استفاده می کنیم.

در این سوال چون انتگرال گیری در سه بعد است فرض کنید که در حالت کلی نیاز داریم انتگرال $\iiint f(x,y,z)\ dxdydz$ را حساب کنیم.

پس کافیست اعداد رندومی با تابع توزیع یونیفورم به فرم (x, y, z) تولید کنیم و میانگین f را در این نقاط حساب کرده و در حجم کل ضرب کنیم.

ولی در این سوال باز هم بخاطر تقارن های موجود و اینکه تابع فقط روی محور z ها مقدارش تغییر می کند پس می توانیم اعدادمان را فقط روی محور z ها تولید کنیم و محاسبات را انجام دهیم. (مهم نیست که به ازای هر z ، در چه x و y ای قرار داریم و مقدار تابع فقط به مقدار z بستگی دارد)

پس با این گفته ها کدمان را می نویسیم و توضیحاتش به این صورت می شود:

در قسمت configs مقادیر ثابت را به صورت زیر تعریف می کنیم:

```
rho_0 = 1

R = 1

V = 4/3 * np.pi * R**3
```

تابع چگالی هم به صورت زیر بدست می آید:

چون می دانیم تابع چگالی خطی است پس:

$$\rho(z) = az + b$$

اگر Z=0 را در پایین ترین نقطه ی کره تعریف کنیم طبق فرض های سوال خواهیم داشت:

$$\rho(0) = 0 \rightarrow b = \rho_0$$

و

$$\rho(2R) = 2aR + \rho_0 = 2\rho_0 \rightarrow a = \frac{\rho_0}{2R}$$

پس خواهیم داشت:

$$\rho(z) = \frac{\rho_0}{2R} z + \rho_0$$

پس آن را در کد به این صورت تعریف می کنیم:

```
rho = lambda z: (rho_0/(2*R))*z + rho_0
```

در قسمت functions هم دو تابع f1, f2 را به این صورت تعریف می کنیم که f1 تابع صورت کسر و f2 تابع مخرج کسر باشد.

```
f1 = lambda z: rho(z)*z
```

```
f2 = lambda z: rho(z)
```

تابع simple_monte_carlo را هم مانند تمرین قبل تعریف می کنیم که از تابع هایمان انتگرال بگیرد و تابع main هم که محاسبات را انجام داده و مقدار انتگرال را خروجی می دهد.

7.2 نتایج

- در بخش نتایج هم چند تست که نتیجه ی شان را می دانیم انجام داده ایم که از صحت نتایجمان مطمئن تر شویم.
- Z=0 را هم در پایین ترین نقطه ی کره تعریف کرده ایم و چون R را برابر با 1 در نظر گرفته ایم پس مرکز کره در Z=1 قرار می گیرد.

خواسته ی سوال

```
[6]: main()
```

```
number of random numbers:  1000000
Z_com = 1.1107262403997604
```

تاثیر نداشتن مقدار مطلق چگالی روی نتیجه

```
[7]: rho_0 = 50
```

```
[8]: main()
```

```
number of random numbers:  1000000
Z_com = 1.1114188666725515
```

اگر چگالی کره یکنواخت باشد مرکز جرم باید در مرکز باشد

```
[9]: rho = lambda z: z*0 + rho_0
```

```
[10]: main()
```

```
number of random numbers:  1000000
Z_com = 0.9998880823589794
```