方法二. 分类讨论再合并

① $Np < L$

可以认为是独立的

即每人只要失败则退回

step	0	1	2	3	……
	1	$1-p$	$(1-p)^2$	$(1-p)^3$	……

$$sum = \sum_{0}^{\infty} (1-p)^i = \frac{1}{p}$$

$$\therefore P(step = k) = \frac{(1-p)^k}{\frac{1}{p}} = p(1-p)^k$$

②. $NP > L$

引入概念"最大到达分布" $m(x)$

即在T回合里到达 step $= x$

的次数。

$m(x)$ 中位数可以这样求：

即平衡时，得到 = 失去，

得到 = 前进 $= N$

失去 $= (\mu+1)L$

$\therefore \mu = \frac{N}{L} - 1$

对于目标分布 $f(x)$，即在 x 上的概率

分布，与 $m(x)$ 有如下关系，

$$\underbrace{\frac{T \cdot f(x)}{\text{在}x\text{上数量}}} = \underbrace{\int_x^{+\infty}} m(x) \cdot \frac{T}{\frac{N}{L} - 1}$$

平均需要 $\frac{N}{L}-1$ 次到达最大步数点

从 x 到 $+\infty$ 的最大步数都对其有贡献，因都经过该点、

$$\therefore \left(\frac{N}{L} - 1\right) \cdot f'(x) = -m(x)$$

然后猜测 $m(x)$ 与正态分布类似。（μ 值高，两头低）

而且 $f'(x) \sim m(x)$ 也说明了一定关系、

然后因为 $f(x)$ 在足够大时

$$\frac{f(x+1)}{f(x)} = 1-p$$ 。因为一失就回.

通过机器用二分法猜测 σ 值.

使 $\frac{f(a)}{f(a-1)} = 1-p$，其中 $\underbrace{f(a) = \int_0^{\infty} f(x)\,dx = \frac{1}{N}}_{\text{不一定要这样}}$.

最后有 $m(x) \sim \mu, \sigma$

$$f(x) = \frac{T}{\frac{N}{2}-1} \cdot \int_x^{+\infty} m(x)$$

③ 合并

记算在二项分布 Np, $Np(1-p)$ 下，$P(X<L)$ 合 $P(X>L)$

用 $P(X<L)\times①+P(X>L)\times②$ 得到最后解

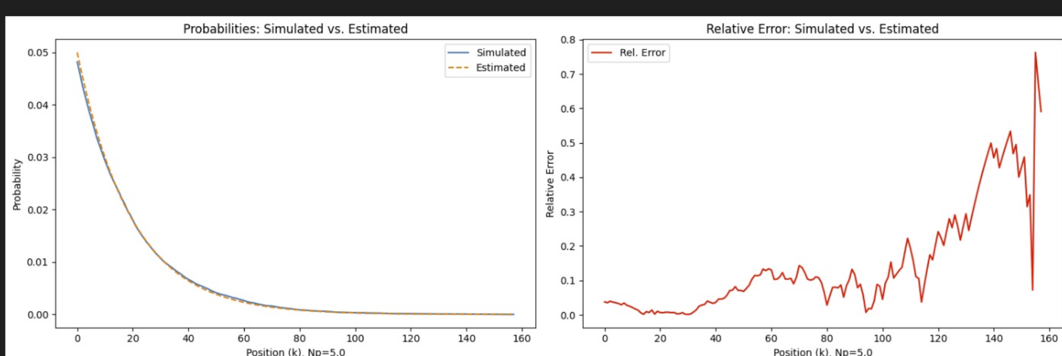
检验：

$Np<L$

```python
L = 10
N = 100
p0=0.05
T = 100000
method_2(N,L,p0,T=100000)
```

✓ 6.0s — Python

```
8.999999999991815
0.9999999291990661 7.080093389344455e-08
```

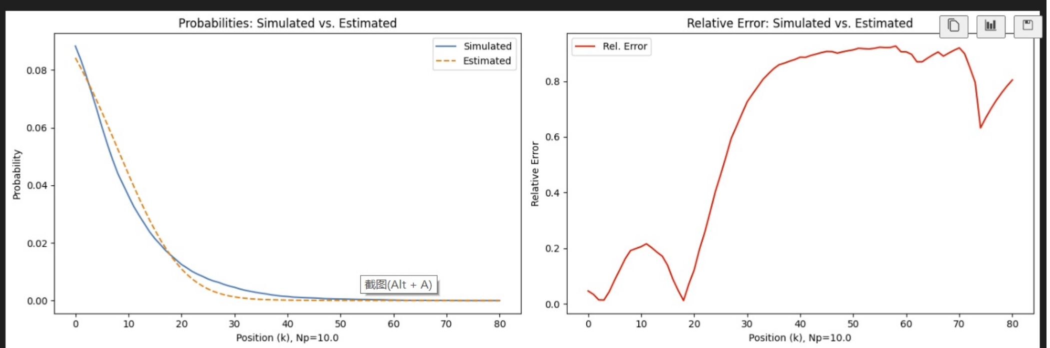


$NP = L$

```
L = 10
N = 100
p0=0.1
T = 100000
method_2(N,L,p0,T=100000)
```

✓ 6.2s

```
8.999999999991815
0.5 0.5
```

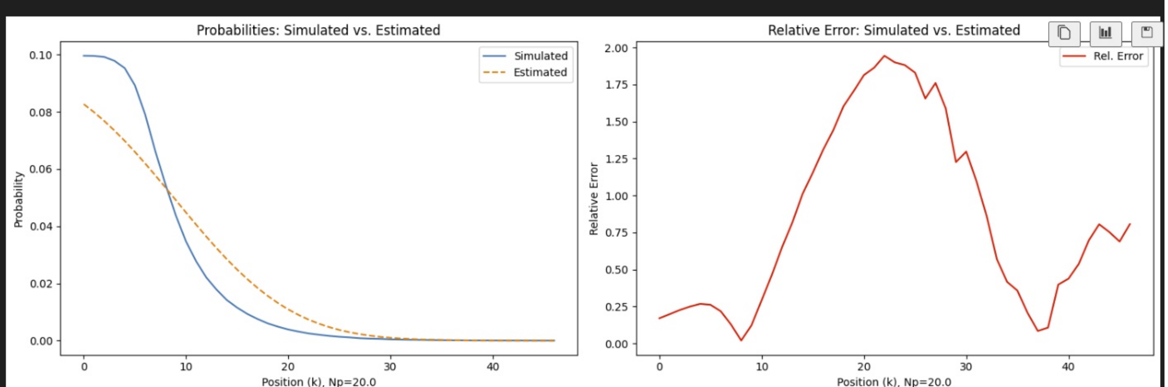


$NP > L$

```
L = 10
N = 100
p0=0.2
T = 100000
method_2(N,L,p0,T=100000)
```

✓ 7.3s

```
8.999999999991815
0.0008890252991084321 0.9991109747008916
```

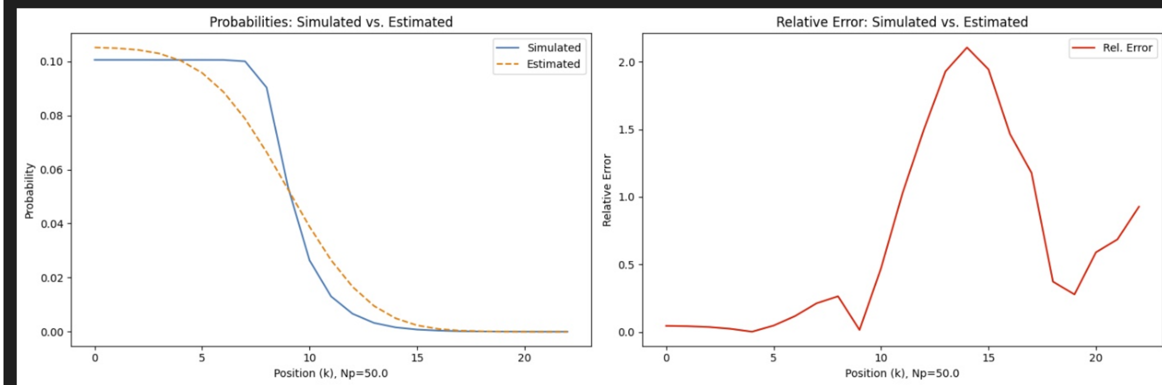


$Np >> L$

```python
L = 10
N = 100
p0=0.5
T = 100000
method_2(N,L,p0,T=100000)
```

✓ 5.9s

```
2.98953693326348
6.22096057427174e-16 0.9999999999999993
```

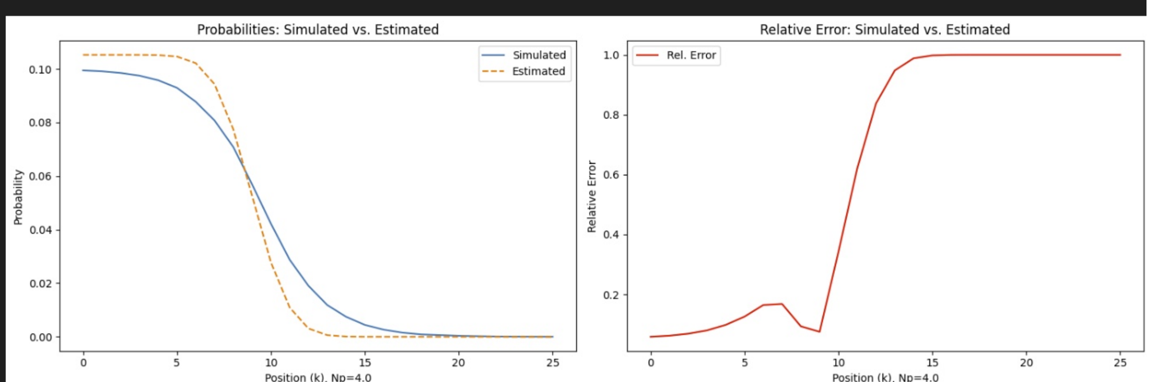


N 较小

```python
L = 1
N = 10
p0=0.4
T = 100000
method_2(N,L,p0,T=100000)
```

✓ 3.6s

```
1.5871340343628617
2.0522634252189396e-10 0.9999999997947736
```

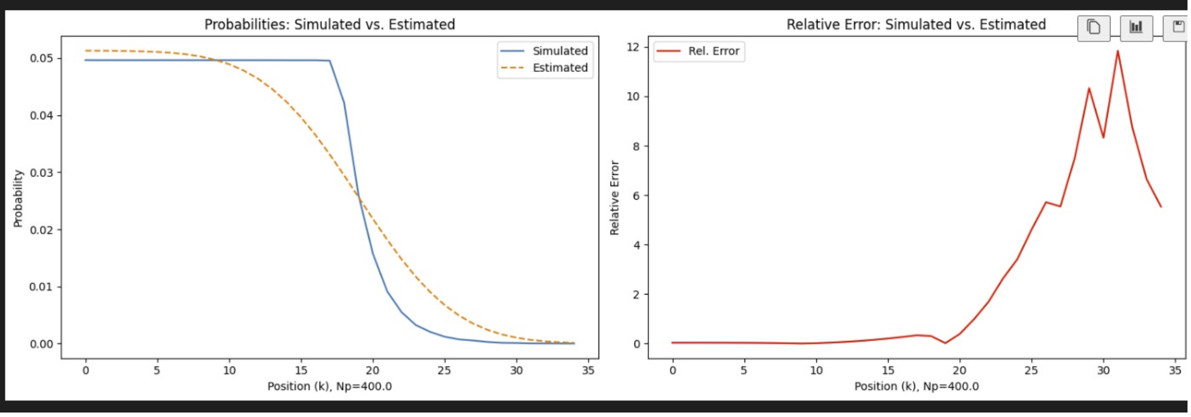


N较大

```
L = 50
N = 1000
p0=0.4
T = 100000
method_2(N,L,p0,T=100000)
```

✓ 14.7s

```
5.355380205856818
1.5307192759192892e-13 0.9999999999998469
```



方法三.

是近似函数解

即把 $g(x)=p(1-p)^x$ 在 $x>\frac{N}{2}-1$ 的部分和其统 $g(\frac{N}{2}-1)$ 做点对称拼接起来就行.

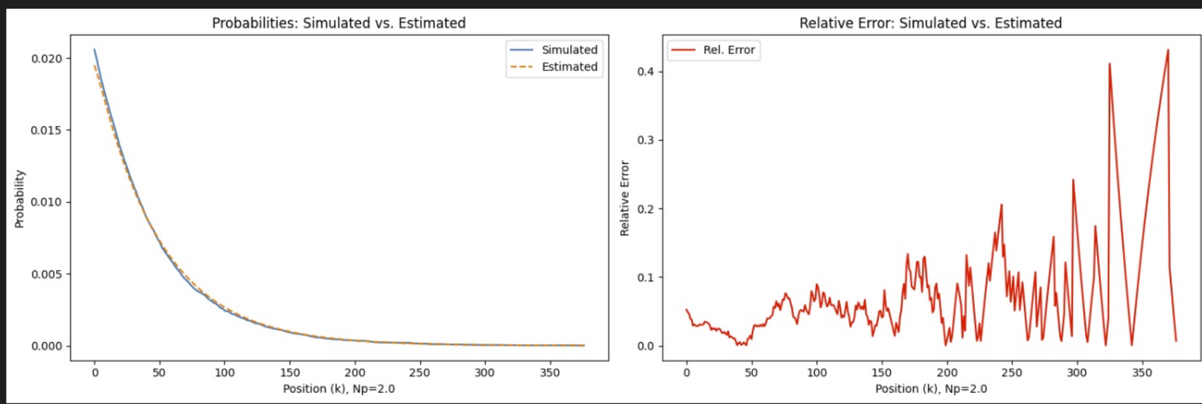
验证:

$NP<2$

$Np < L$

```python
L = 10
N = 100
p0=0.02
T = 100000
method_3(N,L,p0,T=100000)
```

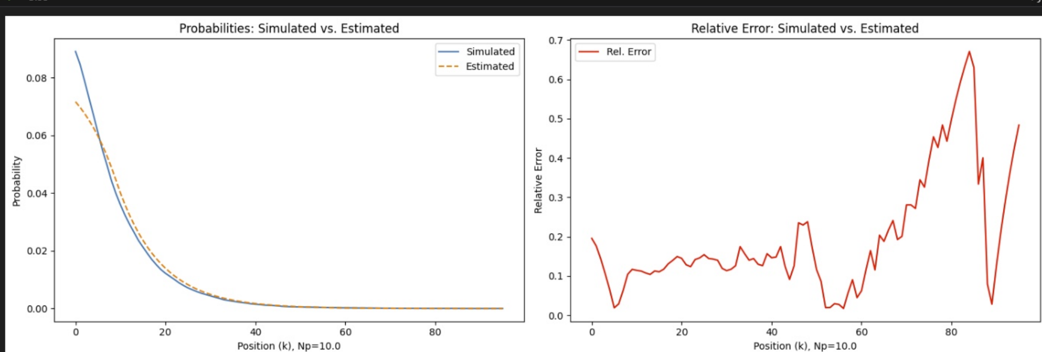
✓ 2.2s



$Np = L$

```python
L = 10
N = 100
p0=0.1
T = 100000
method_3(N,L,p0,T=100000)
```

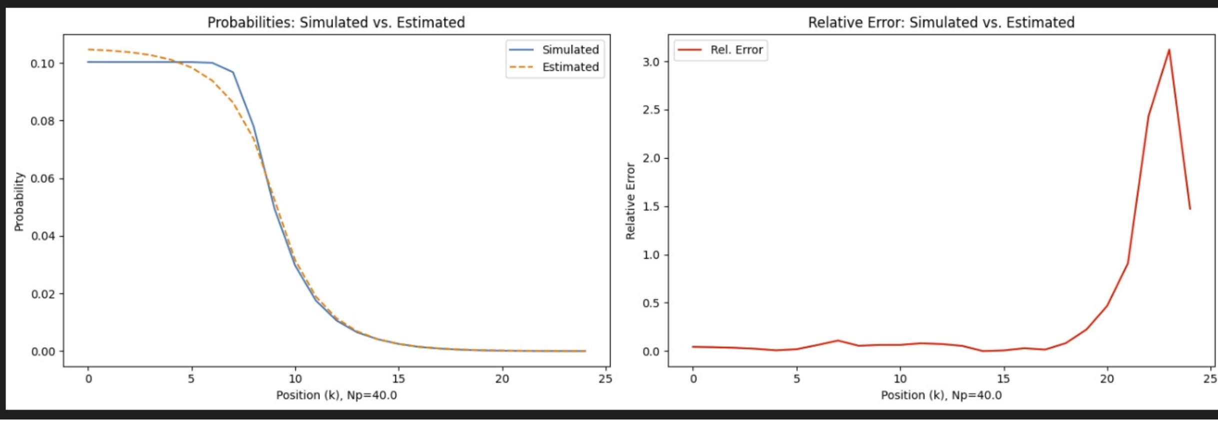
✓ 3.9s



$Np > L$

```python
L = 10
N = 100
p0=0.4
T = 100000
method_3(N,L,p0,T=100000)
```

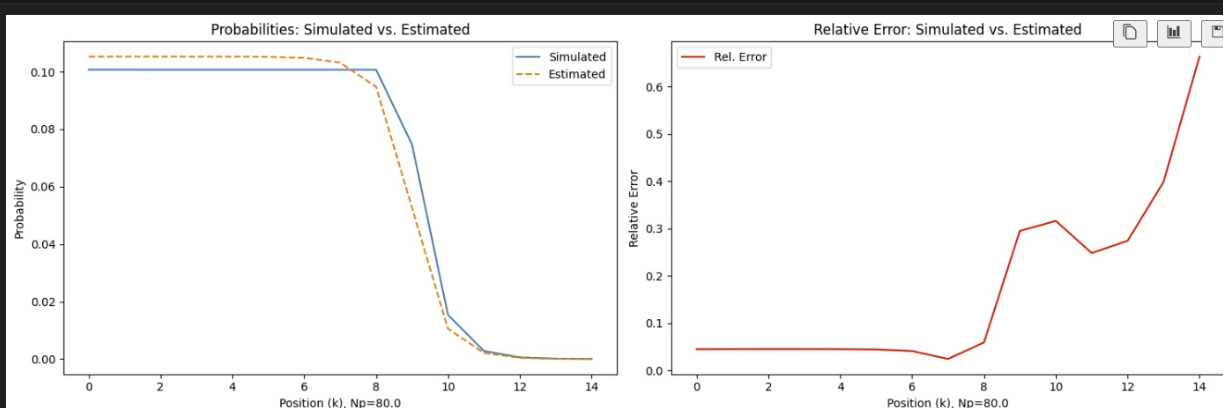
✓ 2.6s



P较大

```python
L = 10
N = 100
p0=0.8
T = 100000
method_3(N,L,p0,T=100000)
```

✓ 4.2s

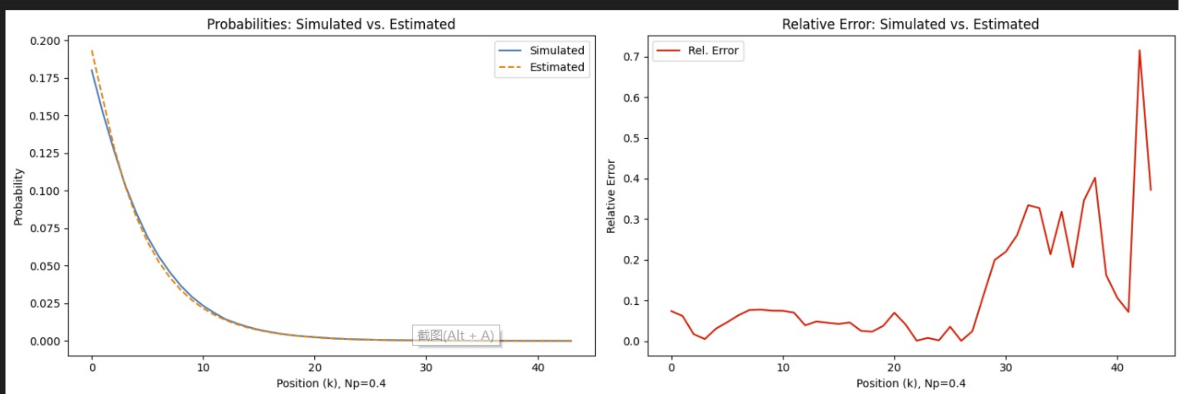


在N很小时也有很好的性能

N=2，p小时

```
L = 1
N = 2
p0=0.2
T = 100000
method_3(N,L,p0,T=100000)
```

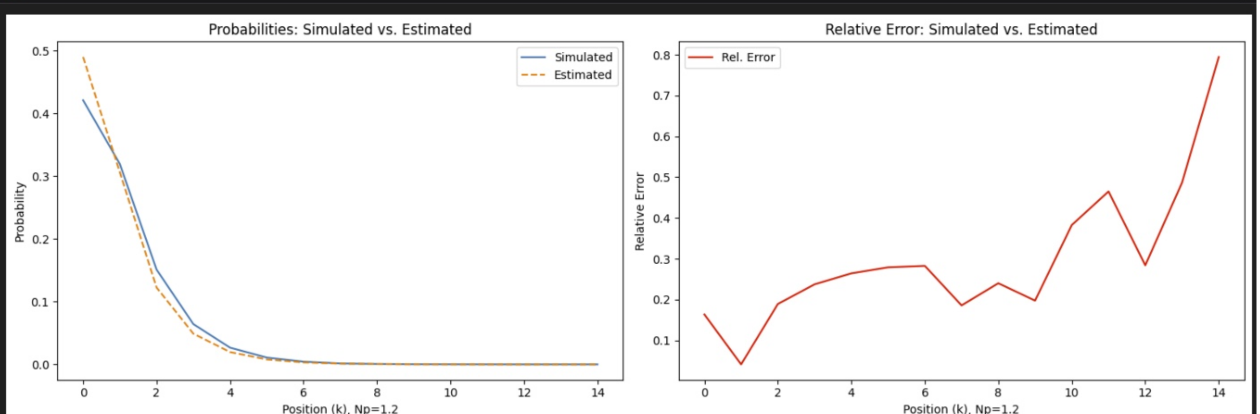
1.3s



N=2 p大时

```
L = 1
N = 2
p0=0.6
T = 100000
method_3(N,L,p0,T=100000)
```

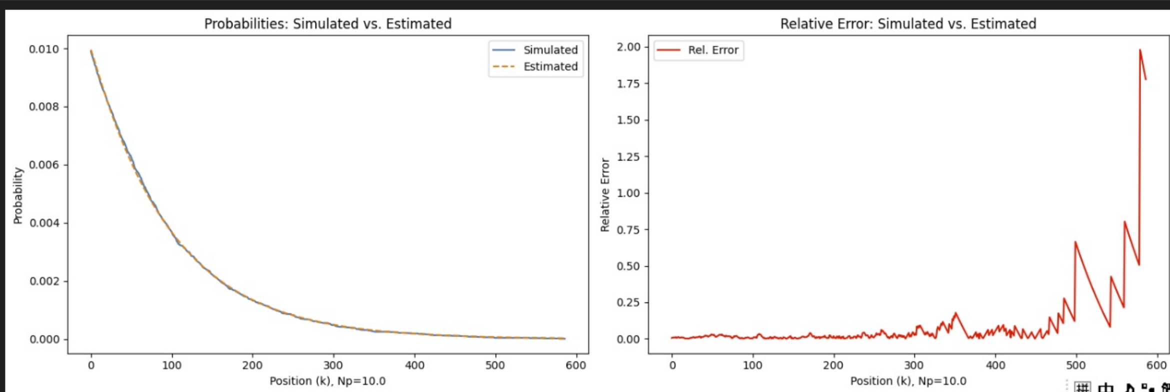
1.7s



N大 p小

```python
L = 100
N = 1000
p0=0.01
T = 100000
method_3(N,L,p0,T=100000)
```

✓ 9.3s



N大 p大

```python
L = 100
N = 1000
p0=0.4
T = 100000
method_3(N,L,p0,T=100000)
```

[25] ✓ 11.3s