Mètodes numèrics II

Simulació 2

Números irracionals

GENÍS LÁINEZ MORENO 1420606
ORIOL TEIXIDÓ GARCIA 1426827

Abstract: En el present *report*, ens proposarem aconseguir numèricament a partir de fenòmens estocàstics 3 números irracionals força coneguts en el món de les matemàtiques e, π i Φ.

Universitat Autònoma de Barcelona - 31 de Desembre de 2017

"Everything existing in the universe is the fruit of chance and necessity"

DEMOCRITUS

1 Introducció

Els nombres irracionals estan sovint relacionats amb fenòmens estocàstics que tenen lloc en diverses situacions físiques. Farem, doncs, models d'aquests sistemes que involucren conceptes de probabilitat que ens permetran obtenir certs nombres irracionals.

A continuació explicarem per separat l'obtenció de cada nombres irracional junt amb el model que ens permet tal cosa. Els números que ens proposem determinar són e, π i Φ.

2 El número e

Fonament teòric

Per tal d'obtenir el número e utilitzarem la probabilitat de desintegració d'una certa partícula en un determinat interval de temps. Si estudiem un sistema d'N partícules d'aquest tipus podem deduir que es compleix, per a un cert nombre arbitràriament elevat de partícules N

$$dN = -p \cdot N dt \quad ,$$

on p és la probabilitat que es desintegri cada partícula en un interval de temps dt. Per tant, separant convenientment les variables i integrant podem deduir

$$ln\left(\frac{N}{N_0}\right) = -p\Delta t \quad \Rightarrow \quad N(t) = N_0 e^{-p\Delta t} \quad .$$

Per tant si imposem $p \equiv (\Delta t)^{-1}$ podrem treure el número e de la relació N_0/N al cap d'un temps Δt. És a dir

$$e^{-p\Delta t} = e^{-1} = \frac{N}{N_0} \tag{1}$$

Implementació

Construirem per tant un programa en $C++$ que simuli un sistema d'N partícules cada una amb una probabilitat $p \equiv (\Delta t)^{-1}$ de desintegrar-se cada iteració. Aquestes iteracions esmentades correspondran a un interval de temps finit δt que intentarà simular el continu i, per tant, serà escollit arbitràriament petit. El programa relativament senzill és el següent:

```
int n, x, f, t=200, N0=7000000, N=N0, s; double e;
srand(time(NULL));

for(s=1; s<=t; s++){
for(n=1; n<=N; n++){

x=rand()%t;
if(x==0){ N=N-1;}        }}

e=N0*pow(N,-1);
```

la funció $srand(time(Null))$ ens garanteix nombres atzarosos que siguin diferents cada vegada que executem el programa. El primer for ens permet fet tantes iteracions com divisions, δt, hem fet del temps total Δt i el segon s'ocupa de què cada una de les partícules tingui la probabilitat corresponent de desintegrar-se a més, és clar, de restar una unitat a N cada cop que una desintegració es produeix.

Resultats i discussió

Amb el programa exposat obtenim:

e obtingut	N_0	Δt	$\lvert 1 - \frac{e_{\text{obt}}}{e} \rvert$
2.7144	$7 \cdot 10^6$	200	0.14%
2.8210	$7 \cdot 10^4$	$2 \cdot 10^3$	3.78%
2.8185	$7 \cdot 10^5$	$2 \cdot 10^3$	3.68%
2.7218	$7 \cdot 10^4$	200	0.13%
2.7123	$7 \cdot 10^4$	200	0.22%

Observem uns resultats notablement aproximats al valor que volíem obtenir. Variant els paràmetres N_0 i Δt podem obtenir millor resultats com s'exposa en la taula anterior. A l'augmentar qualsevol d'aquests dos el temps de càlcul del programa augmenta considerablement i no pas de forma lineal. No obstant això, donat els bons resultats podem dir que no urgeix la necessitat d'emprar un temps arbitràriament gran. Una observació qualitativa d'interès és que obtenim millors resultats quan augmentem N_0 que quan augmentem Δt. Això és degut al fet de que per un Δt molt gran es pot donar el cas en què les partícules restants (sense desintegrar) acaben sent molt escasses i deixa d'ésser una bona mostra per a aplicar la probabilitat de desintegració que és, per construcció, aquest cop més petita ($p \equiv \Delta t^{-1}$). La darrera observació és l'obtenció de resultats diferents per a parells de paràmetres, N_0 i Δt, fixats com es pot apreciar a les dues últimes files de la taula, això es deu, com s'ha explicat anteriorment, a la funció $srand(time(Null))$.

3 El número π

En el cas de l'obtenció del número pi, presentarem dos procediments diferents per mostrar el poder de la idea d'obtenció d'un número sobre la precisió del resultat. És a dir, quan per un procediment ja no és factible intentar reduir més l'error, en l'altre el podem reduir en moltes menys iteracions.

Mètode de Montecarlo

Fonament teòric

El mètode usat per a la primera obtenció del número π és el mètode de Montecarlo. El mètode porta aquest nom en referència al Casino de Montecarlo per ser la capital del joc de l'Atzar. El sistema en què estudiarem aquesta llei de probabilitat consta d'un quadrat de costat 1 amb una circumferència circumscrita de diàmetre 1.

Si tirem una pedra en aquest sistema sense mirar, la probabilitat de què passi per dins la circumferència s'equival a la raó entre les àrees. Per assignar un valor a aquesta probabilitat, ens basarem en l'assignació tipus freqüencial

$$p = \lim_{n\to\infty} \frac{\#\text{casos favorables}}{\#\text{casos totals}} = \frac{\text{Àrea circumferència}}{\text{Àrea quadrat}} = \frac{\pi}{4} \tag{2}$$

Per tant, la idea és utilitzar un programa com el c++ per crear parelles de nombres aleatoris i contar els que han caigut (han quedat situats) dins de la circumferència.

Implementació

Primer de tot, hem vist que agafant la circumferència amb radi 1, el quadrat de costat igual 2 i quedant-nos amb el primer quadrant d'aquest sistema, és a dir, dividint l'àrea de cada figura entre 4, ens queda la mateixa raó d'àrees i podem fer servir la mateixa relació de probabilitat.

$$p = \frac{\pi 1^2/4}{2^2/4} = \frac{\pi}{4} \tag{3}$$

Per tant, ara només ens caldrà generar parelles de nombres aleatoris entre el 0 i l'1 i contar com a cas favorable aquell que la suma dels quadrats d'aquesta parella de nombres sigui menor o igual a 1.
Per generar nombres entre el 0 i 1, hem demanat al programa que ens doni nombres aleatoris entre el 0 i una potència de 10 per dividir-los entre la mateixa potència de 10. D'aquesta manera, podem tindre nombres aleatoris entre el 0 i l'1 amb tantes xifres decimals com l'ordre de la potència de 10. El nombre de casos totals l'hem fixat a 5000000, tot i que també estudiarem els resultats amb diferent nombre de casos.

```
int num1,num2,n,M=5000000; float x, y,a,s=0,k,prob,pi;
srand(time(NULL));

for(n=1;n<=M;n=n+1){
        num1= rand()%(1000-0); x=num1*pow(1000,-1);
        num2= rand()%(1000-0); y=num2*pow(1000,-1);

if(x*x+y*y<1){   a=1;}
else{    a=0;}

s=s+a; }
```

On s és el nombre de casos favorables. Com hem dit, el número de casos favorables entre casos totals, equivaldrà a la freqüència d'aquest succés. Quan el nombre de casos és prou elevat, aquesta freqüència la podem tractar com a la probabilitat del succés. Així doncs, multiplicant aquest valor de la probabilitat per 4, ens donarà el nombre π.

```
pi=4*s/M;
```

Resultats i discussió

Amb aquest mètode per l'obtenció del número π, hem trobat els següents resultats en funció del nombre de decimals dels nombres aleatori x i y i del nombre d'iteracions

π obtingut	Decimals	iteracions n	Temps de càlcul (s)	$\lvert 1 - \frac{\pi_{\text{obt}}}{\pi}\rvert \cdot 100$
3.16565	3	$5 \cdot 10^6$	1.22	0.76%
3.16485	3	$5 \cdot 10^6$	1.20	0.74%
3.18312	2	$5 \cdot 10^6$	1.23	1.32%
3.27288	4	$5 \cdot 10^6$	1.25	4.18%
3.16648	3	$5 \cdot 10^4$	0.38	0.79%
3.16517	3	$5 \cdot 10^8$	44.94	0.75%

Primer de tot, s'observa que el nombre idoni de decimals per l'obtenció de π per aquest mètode és 3. Recordem que el nombre de decimals el determinem en el moment que li demanem al programa que ens

doni nombres aleatoris entre 0 i una potència de 10. L'ordre d'aquesta potència de 10 és la que ens determinarà el nombre de decimals. Per tant, sembla que al programa li costa donar nombres aleatoris tan grans i d'aquesta manera hi ha més punts que cauen dins la circumferència quan demanem nombres aleatoris d'ordre elevat.

En segon lloc, veiem que tal com ens esperàvem, podem notar que com més gran sigui el nombre d'iteracions, més ens aproparem al valor esperat de π. També podem remarcar que a partir d'un nombre d'iteracions d'ordre 6, ja no és tan notable la millor aproximació al nombre π. Alhora, ens podem fixar que per un nombre d'iteracions d'ordre 8, ens hem d'esperar 44.94 segons per obtenir un valor de pi semblant al d'ordre 6. Per tant, podríem dir que el nombre d'iteracions que ens dóna un millor resultat i en menys poc temps és proporcional a una potència de 10 a la 6.

Mètode dels nombres coprimers

Fonament teòric

Aquesta vegada, per a trobar el número π, emprarem un truc conegut en el món de la teoria de números que es deriva de la manipulació de la identitat d'Euler (o equivalentment i més general de la funció $\zeta(s)$ de Riemann)

$$S \equiv \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6} = \zeta(2) \quad . \tag{4}$$

Aquesta sèrie és convergent i absolutament sumable, propietats necessàries per a la manipulació que il·lustrarem a continuació. Primerament agafem la sèrie i la multipliquem per $\frac{1}{2^2}$,

$$S = \quad \tfrac{1}{1^2} + \tfrac{1}{2^2} + \tfrac{1}{3^2} + \tfrac{1}{4^2} + \tfrac{1}{5^2} + \tfrac{1}{6^2} + \dots \tag{5}$$

$$\frac{1}{2^2}S = \qquad \tfrac{1}{2^2} + \qquad \tfrac{1}{4^2} + \qquad \tfrac{1}{6^2} + \dots \tag{6}$$

Fent una simple resta obtindrem

$$\left(1 - \frac{1}{2^2}\right) S = \frac{1}{1^2} + \frac{1}{3^2} + \frac{1}{5^2} + \frac{1}{7^2} + \frac{1}{9^2} + \frac{1}{11^2} + \dots \quad , \tag{7}$$

és a dir, els termes senars d'S. Si procedim de la mateixa manera, multiplicant ara per un factor $\frac{1}{3^2}$ la nova sèrie, obtindrem

$$\left(1 - \frac{1}{3^2}\right)\left(1 - \frac{1}{2^2}\right) S = \frac{1}{1^2} + \frac{1}{5^2} + \frac{1}{7^2} + \frac{1}{11^2} + \frac{1}{13^2} + \frac{1}{17^2} + \dots \quad , \tag{8}$$

és a dir, termes de la sèrie S els inversos dels quals no són ni múltiples de 2 ni de 3. Iterant aquest procediment infinitament obtenim l'expressió

$$\prod_{p,primer} \left(1 - \frac{1}{p^2}\right) S = 1 \quad \text{i, per tant,} \quad \frac{6}{\pi^2} = \prod_{p,primer} \left(1 - \frac{1}{p^2}\right) \tag{9}$$

Paral·lelament podem deduir que la probabilitat de què un nombre n sigui divisible entre un nombre primer p és $P(n = \dot{p}) = 1/p$. La probabilitat de què ho siguin dos nombres n i m és $P(m, n = \dot{p}) = 1/p^2$ i la probabilitat de què no siguin múltiples ambdós alhora del nombre p és

$$P(m, n \neq \dot{p}) = 1 - \frac{1}{p^2} \tag{10}$$

Sabent que dos nombres són primers entre si (coprimers) si no tenen cap divisor comú, la probabilitat de què, escollint 2 nombres a l'atzar, aquests siguin primers entre ells és

$$P(m,n) = \prod_{p,primer} \left(1 - \frac{1}{p^2}\right) = \frac{6}{\pi^2} \tag{11}$$

Implementació

Primerament trobem els nombres primers amb el mètode d'Eratòstenes fins a un nombre N. Descartem un nombre cada 2 posicions partint del número 2 (no inclòs), seguidament un nombre de cada 3 posicions partint del número 3, seguidament descartem un nombre cada 5 posicions (doncs és el següent nombre després del 3 no descartat)... iterant aquest procediment restaran els nombres primers.

```
for(i=1; i<=N ; i++ ){ n[i]=i;} construim una cadena de nombres naturals
while(p<=N){   P[s]=p;   s=s+1;
        for(i=p; i<=N ; i=i+p ){ n[i]=0;} anul.lem els multiples de p
        while(n[p]==0 && p<=N){ p=p+1;}} trobem el seguent nombre no descartat
```

A continuació mirem si dos nombres escollits a l'atzar entre 1 i N són primers entre ells verificant si algun primer és un divisor d'ambdós i repetim el procés t vegades per a obtenir una bona mostra. Amb una funció *while* introduïm les condicions

- La resta de la divisió entre els nombres generats i un nombre primer p_i és major a 0 per algun dels nombres generats.
- Els nombres primers p_i provats són menors a cada un dels nombres generats.

tal que, provant primers p_i de forma creixent, si es deixa de complir la primera abans que la segona tenim un parell de nombres no primers entre ells i, en canvi, seran coprimers en el cas contrari.

```
srand(time(NULL));

for(s=1; s<=t; s++){i=1;

        n1=rand()%N+1, n2=rand()%N+1;

        while(n1%P[i]+n2%P[i]>0 && P[i]<=n2 && P[i]<=n1){i=i+1;}

        if(n1%P[i]==0||n1==n2){no=no+1;}//no son primers      }

pr=1-no*pow(t,-1);
pi=pow(6*pow(pr,-1),0.5);
```

Resultats i discussió

Amb el programa presentat anteriorment obtenim els resultats següents:

π obtingut	N nombres	t parells	$\lvert 1 - \frac{\pi_{obt}}{\pi}\rvert \cdot 100$
3.1418	$20 \cdot 10^4$	$2 \cdot 10^6$	$6.6 \cdot 10^{-3}\%$
3.1410	$20 \cdot 10^4$	$4 \cdot 10^6$	$1.9 \cdot 10^{-2}\%$
3.1423	$25 \cdot 10^4$	$4 \cdot 10^6$	$2.3 \cdot 10^{-2}\%$
3.1518	1000	$20 \cdot 10^4$	$3.2 \cdot 10^{-1}\%$
3.1362	$20 \cdot 10^4$	1000	$1.7 \cdot 10^{-1}\%$

Observem que aconseguim una bona aproximació del nombre π. Una possible font d'error és que hem reduït la nostra presa de nombres atzarosos a un interval de nombres naturals i no pas al conjunt infinit d'aquests per evident límit en les eines computacionals. Aquest fet fa que la concentració de nombres primers sigui més gran i, per tant, la probabilitat de trobar dos nombres coprimers augmenti. És a dir,

$$\frac{6}{\pi^2_{\text{obtingut}}} \approx P(m,n) > \frac{6}{\pi^2} \quad \Rightarrow \quad \pi_{\text{obtingut}} < \pi \quad . \tag{12}$$

No obstant això, observem a la taula que per a parells t i N arbitràriament elevats obtenim valors més grans que l'esperat i, per tant, podem dir que l'ordre de l'error que cometem és major que el que ens crea la mancança explicada.

Per altra banda, el temps de càlcul per a intervals de nombres naturals arbitràriament gran es fa, de seguida, feixuc i ardu o, fins i tot, el programa és incapaç de compilar a causa del gran nombre de dades que ha de manejar alhora. La poca diferència entre els resultats obtinguts convida a pensar que, si bé hem introduït les dades N i t a conveniència i tan elevades com ens era possible, aquest mètode no es donarà un resultat millor si no millorem també les tècniques computacionals. Observem, no obstant això, que és un mètode notablement més precís que el mètode de Montecarlo utilitzat en aquesta mateixa pràctica per al càlcul del nombre π però, per contra, el temps de càlcul és molt més elevat.

4 El nombre d'or Φ

Fonament teòric

Per a trobar tal número a partir de la probabilitat farem ús de la coneguda Successió de Fibonacci. Una possible definició del número d'or a partir de la successió de Fibonacci $\{a_n\}$ és

$$\Phi \equiv \lim_{n\to\infty} \frac{a_{n+1}}{a_n} \quad . \tag{13}$$

Recrearem aquesta successió a partir d'un sistema d'elements que anomenarem conills d'ara endavant per a fer amena la metodologia i per l'evident analogia que ens permet pensar els elements com a entitats reproduïbles. Aquests conills compleixen unes regles senzilles que són les següents:

- Per a cada iteració cada parella de conills adults pot tenir un nombre de fills amb una certa probabilitat.
- En cada iteració els fills que han tingut els adults de la darrera generació esdevenen adolescents
- En cada iteració els adolescents de l'anterior generació esdevenen adults i, per coherència amb la primera, adquireixen amb aquesta condició l'habilitat per reproduir-se.

Les probabilitats de reproducció dels adults, p_i de tenir n_i fills, compleixen

$$\sum_i p_i \cdot n_i = 2 \quad \Rightarrow \quad \langle n \rangle = 2 \tag{14}$$

Trivialment es pot comprovar que si és exactament 2 llavors el nombre de conills b_n en cada n-èssima iteració està relacionat amb la sèrie de Fibonacci a_n de la forma $b_n = 2a_n$.*
D'aquesta manera aconseguirem que mentre avança el temps (iteracions) i els conills són més nombrosos el nostre nombre de conills b_n en cada n-èssima iteració sigui

$$\lim_{n\to\infty} \frac{b_{n+1}}{b_n} \approx \lim_{n\to\infty} \frac{2a_{n+1}}{2a_n} = \Phi \quad . \tag{15}$$

*Si tenim 2 conills adolescents, en la primera iteració tenim una parella ($b_1 = 2a_1 = 2$) i en la segona també, tot i que ara seran adults, ($b_2 = 2a_2 = 2$); en la tercera aquesta parella té 2 fills i per tant són 4 conills en total ($b_3 = 2a_3 = 4$); en la quarta iteració els 2 conills adults tenen 2 fills més i els fills de la iteració anterior passen a ser adolescents ($b_4 = 2a_4 = 6$)...

Implementació

Construïm altra vegada el nostre sistema a partir del llenguatge de programació $C++$ de la següent manera:

```
srand(time(NULL));

for(s=1; s<=t; s++){

        N1=N1+N2; N2=N3; N3=0;

for(n=1; n<=N1/2; n++){

          x=rand()%4;
        if(x==0){ N3=N3+2;}
        if(x==1||x==2){ N3=N3+3;}          }}

phi=(N1+N2+N3)*pow(N1+N2,-1);
```

on hem fet servir de nou la funció $srand(time(NULL))$ i hem establert la correspondència $N1 \equiv$ #conills adults, $N2 \equiv$ #conills adolescents, $N1 \equiv$ #fills. En el nostre cas les probabilitats relacionades amb tenir fills de les parelles adultes són

$$p_1\,(2 \text{ fills}) = \frac{1}{4} \qquad p_2\,(3 \text{ fills}) = \frac{1}{2} \qquad p_3\,(0 \text{ fills}) = \frac{1}{4} \tag{16}$$

que compleixen clarament la relació 14.

Resultats i discussió

Amb el programa exposat obtenim:

Φ obtingut	$b_0 = N1_0$	iteracions n	$\lvert 1 - \frac{\Phi_{\text{obt}}}{\Phi} \rvert \cdot 100$
1.6203	20	20	0.97%
1.6181	40	20	$1.6 \cdot 10^{-3}$%
1.6177	20	25	0.02%
1.6209	30	15	0.17%
1.6155	30	15	0.16%

Aconseguim doncs el propòsit d'obtenir el número d'or amb un error relatiu considerablement petit. Variant els paràmetres que són el nombre de conills inicial (tots adults) i el nombre d'iteracions aconseguim aproximar-nos més al valor que busquem. Com passa en l'*obtenció del número e*, en aquest cas també trobem una notable millora dels resultats augmentant, preferiblement, un dels dos paràmetres que és, en aquest cas, el nombre inicial de conills. Augmentant les iteracions també millorem el resultat però relacionant el temps de càlcul que emprem en augmentar aquest paràmetre convida a ser cautelosos al fer aquesta operació.

5 Conclusions

- Hem pogut obtenir els nombres e, π i Φ satisfactòriament. Per ser un mètode estadístic per obtenir nombres irracionals, veiem que per a un major nombre d'iteracions, ens aproximem més al valor esperat del número que volíem obtenir reafirmant així el teorema dels grans números.
- D'altra banda, en augmentar el número d'iteracions també augmenta notablement el temps de càlcul. Per conseqüent, ha urgit establir un criteri arbitrari de temps elevat. Per a aquest temps arbitrari (aproximadament un minut) hi ha a cada taula un valor, del nombre irracional, que considerem per tant el millor que pot obtenir el programa.
- També anotem com a conclusió que el mètode per obtenir π basat en les propietats dels nombres primers, ens dóna un resultat més precís. Però el temps de càlcul d'aquest segon mètode, és més elevat.

Further research

En aquest treball, ha sigut, per a nosaltres, una prioritat treure resultats amb un error arbitràriament baix. Aquest fet s'ha traduït en uns programes que empraven molt temps de càlcul per a cada resultat. Per tant, això ha sigut un impediment a l'hora de treure gran quantitat de dades per a presentar aquestes en forma, per exemple, de gràfic. Per tant, una millora significativa de l'estudi que presentem seria optimitzar aquest temps i millorar les eines computacionals utilitzades per tal d'obtenir més conclusions significatives.

Bibliografia

[1]**Carles Navau** *Lectures sobre mètodes numèrics II. UAB*
[2]**Genís Láinez i Oriol Teixidó** *LáinezTeixidó.pdf, Equacions Diferencials*