KM 15(1984)
pag 76 - 86

HET SIMULEREN VAN EEN NORMALE VERDELING MET APL

- EEN VERGELIJKEND ONDERZOEK -

R. Tijssen*

Samenvatting

APL is een programmeertaal die, door middel van krachtige operatoren, een snelle programmering mogelijk maakt van methoden die een normale verdeling kunnen simuleren. In dit artikel worden de APL-algoritmes van een vijftal bekende simulatiemethoden op hun resultaten vergeleken. Hierbij werden zowel de rekentijden van de algoritmes als de kwaliteit van de geproduceerde normale verdelingen in ogenschouw genomen. Uit dit onderzoek bleken de Box-Muller methode en de Centrale Limiet Stelling de meest efficiënte simulatiemethoden te zijn.

*Vakgroep Methoden en Technieken, Subfaculteit der Psychologie, Rijksuniversiteit Leiden, Hooigracht 15, 2312 KM Leiden, tel. 071 - 148333, tst. 5126.

1. INLEIDING

In het wetenschappelijk onderzoek kan men in bepaalde situaties behoefte hebben aan een of meer normaal verdeelde grootheden. Vanwege de gunstige statistische eigenschappen is deze specifieke verdeling ondermeer zeer bruikbaar voor de toetsing van karakteristieken van (nieuwe) statistische- en methodologische programmatuur in de vorm van Monte-Carlo studies. De simulatie van een dergelijke verdeling kan op een aantal duidelijk verschillende manieren worden uitgevoerd. Overzichten hiervan worden gegeven door Muller (1959), Ahrens & Dieter (1972), Atkinson & Pierce (1976), Kinderman & Ramage (1976) en Payne (1977).
De meeste methoden hebben als uitgangspunt de uniforme verdeling-U(0,1), die getransformeerd wordt in een benadering van de standaard-normale verdeling-N(0,1), waarbij het aantal benodigde U(0,1)-verdelingen evenwel per methode varieert. In dit artikel zullen we een vijftal methoden vergelijken op hun nauwkeurigheid en efficiency.

De desbetreffende algoritmes zijn in de programmeertaal APL geschreven. APL is een taal voor interactief computergebruik en ondermeer zeer toepasbaar voor statistische bewerkingen. Deze taal is bij uitstek geschikt voor de bewerking van matrices. Het voordeel van APL, in vergelijking met andere computertalen, is de beschikbaarheid van een reeks ingebouwde krachtige operatoren die een zeer snelle programma-opbouw mogelijk maken.

Het genereren van de steekproeven uit de U(0,1)-verdeling is verricht met behulp van de APL-operator die pseudo-aselecte getallen genereert (APL Language, 1978).
Aan de basis van deze operator staat een multiplicatieve congruentiële generator, gebaseerd op een algoritme van D.H. Lehmer (1951), die als volgt gedefinieerd is:

$$Xi + 1 = aXi \pmod{m} \qquad \text{met } i = 1,2,3,\ldots$$

$$Xi = 16807 \qquad a = 16807 \qquad m = 10^{10}$$

Aanvullende karakteristieken van deze random-generator vindt men onder andere in Fuller (1976).
Voor uitgebreide informatie over het genereren van aselecte getallen en aanverwante onderwerpen kan men terecht bij Knuth (1969), Sowey (1972) en Sahai (1980).

De daadwerkelijke rekentijd van APL, uitgedrukt in CPU-seconden, is over het algemeen groter dan vergelijkbare FORTRAN- of PASCAL-programma's. Wij zijn van mening dat genoemde voordelen van APL ruim-

schoots opwegen tegen dit nadeel als men op een snelle en efficiente wijze een beperkt aantal simulaties van een verdeling wil verkrijgen.

De opzet van dit onderzoek was om tot een vergelijking te komen van de CPU-tijden van deze algoritmes, de kwaliteiten van de geproduceerde N(0,1)-verdelingen en de hiervoor benodigde U(0,1)-verdelingen. Deze resultaten kunnen, voor de APL-gebruiker in het bijzonder, een nuttige wegwijzer zijn bij de keuze van een methode voor het genereren van steekproeven uit een standaard-normale verdeling.

Voor dit onderzoek is de APL standaardversie toegepast met een AMDAHL 7VB computer. De CPU-tijden van de APL-algoritmes zijn machine-afhankelijk.

2. SIMULATIE-METHODEN

2.1 De Centrale Limiet Stelling (CLS)

Uit de Centrale Limiet Stelling volgt dat voor een gegeven aantal onafhankelijke U(0,1)-verdeelde grootheden U1,U2,...,Un geldt dat

$$((S\ Un) - .5n)/(n/12) * .5$$

bij benadering een N(0,1)-verdeling heeft.
Zelfs bij een betrekkelijk klein aantal U(0,1)-verdelingen geeft deze methode een redelijk goede benadering van N(0,1) en is door Maritsas (1973) als een zeer bruikbare methode aanbevolen. Om vergelijkingen met andere onderzoeksresultaten mogelijk te maken en om berekeningen verder te vereenvoudigen is gekozen voor n = 12.

2.2 Inverse methode (INV)

Deze methode is gebaseerd op de inverse van de cumulatieve verdelingsfunctie. Als U een U(0,1)-verdeling heeft dan heeft $N = F^{-1}(U)$ een verdeling met cumulatieve verdelingsfunctie F (Muller, 1958; Gebhart, 1964). Het APL-algoritme is gebaseerd op de volgende benadering van Hastings (1955) voor de inverse van de cumulatieve verdelingsfunctie van de standaard-normale verdeling.

$$t = (\ln 1/u^2)\ 0.5$$

Voor $0 < u < .5$

$$N = t - ((C0 - C1t + C2t^2)/(1 + C3t + C4t^2 + C5t^3))$$

met de constanten:

C0 = 2.515517	C3 = 1.432788
C1 = 0.802853	C4 = 0.189269
C2 = 0.01328	C5 = 0.001308.

Het teken van N wordt afhankelijk van de waarde van de corresponderende U(0,1)-waarde; groter dan .5 - positief, kleiner dan .5 - negatief.

2.3 Box-Muller Transformatie (B-M)

Dit is eveneens een vrij simpele en directe methode die door Box en Muller (1958) geïntroduceerd is, die wiskundig-exacte transformaties oplevert.

Twee onafhankelijke U(0,1)-grootheden worden getransformeerd in twee N(0,1)-verdeelde grootheden N1, N2:

$$N1 = ((-2 \ln (u1)) * .5) * \cos(2 \pi u2)$$
$$N2 = ((-2 \ln (u1)) * .5) * \sin(2 \pi u2)$$

Onderzoek naar de resultaten van deze methode is ondermeer verricht door Golder & Settle (1976) en Neave (1973), die vermelden dat de resultaten bij een multiplicatieve congruentiële generator met een relatief kleine vermenigvuldiger ($a < 1000$) onnauwkeurigheden kunnen bevatten. Chay, Fardo & Mazumbar (1975) hebben een aantal aanpassingen van de Box-Muller methode voorgesteld die dit probleem kan ondervangen.

2.4 Marsaglia's Polar methode (Polar)

Marsaglia (1962) heeft een iteratieve aanpassing van de Box-Muller Transformatie geconstrueerd waarbij, d.m.v. de volgende stappen, de berekening van de goniometrische functies achterwege kan blijven.

1) Genereer twee onafhankelijke U(0,1)-verdelingen: u1 en u2.
2) $v1 = 2u1 - 1$
   $v2 = 2u2 - 1.$
3) Als $v1^2 + v2^2 > 1$ herhaal 1 en 2, anders
   $N1 = v1(-2 \ln ((v1^2 + v2^2)/(v1^2 + v2^2))) * .5$
   $N2 = v2(-2 \ln ((v1^2 + v2^2)/(v1^2 + v2^2))) * .5$.

2.5 Kinderman-Ramage methode (K-R)

Uit een drietal onafhankelijke uniform verdeelde stochastische grootheden wordt een halve-normale verdeling benaderd, die opgebouwd is uit: een driehoekige dichtheid van de vorm $f1(x) = c1(a - \{x\})$, drie lineaire dichtheden die het resultaat zijn van de verschillen tussen de normale dichtheid Q(x) en f(x) over drie intervallen binnen (-a,a) en een staartverdeling waarvoor de normale dichtheid zelf genomen wordt. De verschillen $ci(Q(x) - f1(x))$ zijn bij benadering lineair over drie subintervallen binnen (-a,0) of (0,a). Door middel van de constanten

ci wordt een halve normaalverdeling opgebouwd.
Bij a = 2.2160358 en g(x) = Q(x) - 0.180025191 (a - I x I) heeft het iteratieve algoritme van Kinderman & Ramage (1976) de volgende vorm:

1) Genereer u1
   Als u1 < .884070402298758 genereer dan u2 en produceer
   x = a(1.13113163544180u1 + u2 - 1).
2) Als u1 < .973310954173898 ga naar 4.
3) - Staartverdeling -
   Genereer u2 en u3 zodanig dat $u^2 < a^2(a - 2\ln(u3)) * .5$
   Als u1 < .986655477086949 produceer dan $x = (a^2 - 2\ln(u3)) * .5$
   anders $x = -(a^2 - 2\ln(u3)) * .5$
4) Als u1 < .958720824790463 ga naar 6.
5) Genereer u2 en u3.
   Definieer t = a -(.63083480192160min(u2,u3)).
   Als max(u2,u3) < .755591531667601 ga naar 9.
   Als .034240503750111(|u2 - u3|) < g(t) ga naar 9; herhaal anders 5.
6) Als u1 < .911312780288703 ga naar 8.
7) Genereer u2 en u3.
   Definieer t = .479727404222441 + (1.10547366102207min(U2,U3)).
   Als max(u2,u3) < .8722834796671790 ga naar 9.
   Als .049264496373128(|u2 - u3|) < g(t) ga naar 9; herhaal anders 7.
8) Genereer u2 en u3.
   Definieer t = .47972740422241-(.595507138015940min(u2,u3)).
   Als max(u2,u3) < .805577924423817 ga naar 9; herhaal anders 8.
9) Als u2 < u3 produceer x = t; anders x = -t.

3. TOETSEN

De Kolmogorov-Smirnov toets onderzoekt in hoeverre een verdeling overeenkomt met een gespecificeerde verdelingsvorm. Zowel de kwaliteit van de gesimuleerde normaalverdeling als de afwijkingen van de uniformiteit van de U(0,1)-verdelingen zijn met de Kolmogorov-Smirnov toets onderzocht.

De onafhankelijkheid van de gegenereerde uniforme verdelingen is tevens onderzocht met een zogenaamde Runs-toets. Deze bepaalt in hoeverre monotone subreeksen de uniformiteit van de totale verdeling aantasten.
Gekozen is voor de Runs-toets van Levene & Wolfowitz (1944), die gebruik maakt van de aantallen en lengten van zowel dalende als stijgende monotone subreeksen.Onder een ' run ' verstaan we in dit verband de

lengte van een monotone subreeks minus één.
Het aantal runs van bepaalde lengten en de bijbehorende verwachte waarde in een reeks van onafhankelijke grootheden worden aan elkaar gerelateerd. Het aantal runs van verschillende lengten is echter negatief gecorreleerd. Levene & Wolfowitz hebben een covariantiematrix afgeleid die in combinatie met de aantallen subreeksen van bepaalde lengten en hun verwachte waarden een asymptotische chi-kwadraat verdeelde toetsgrootheid oplevert (Wolfowitz, 1944).
De aselecte reeksen zijn onderzocht op runs met lengten 1,2,3,4,5 en 6 of meer. De toetsgrootheid heeft dan 6 vrijheidsgraden.
Kennedy & Gentle (1980) raden deze Runs-toets in het bijzonder aan voor de toetsing van onafhankelijkheid in een reeks getallen. Zowel Knuth (1969) als Wedderburn (1976) waarschuwen overigens tegen het gebruik van andere run-toetsen waarbij een covariantiematrix buiten beschouwing wordt gelaten.

4. RESULTATEN EN CONCLUSIES

Voorop gesteld moet worden dat dit onderzoek evenzeer een uitspraak doet over APL als over de toepasbaarheid en kwaliteiten van de gehanteerde simulatiemethoden. De genoemde kwaliteiten van APL komen in het ene algoritme beter tot zijn recht dan in het andere. Deze verschillen hebben de CPU-tijden dan ook vrij sterk beïnvloed.

De CPU-tijden van de algoritmes zijn geregistreerd voor drie onafhankelijke, in lengte variërende, steekproeven uit uniform verdeelde stochastische grootheden.
Elke registratie is vijfmaal herhaald. De CPU-seconden van de algoritmes bleken stabiel te zijn tot in de tweede decimaal.

Tabel I: CPU-seconden algoritmes (gemiddelde over 5 replicaties).

	LENGTE STEEKPROEF		
METHODE	n=250	n=500	n=1000
B-M	.07	.11	.19
CLS	.16	.22	.34
K-R	.43	.81	1.58
POLAR	.65	1.27	2.55
INV	.98	1.81	3.62

De relatief grote verschillen tussen de CPU-tijden van de algoritmes worden vooral veroorzaakt door het aantal loop's en door de complexiteit van de uit te voeren bewerkingen. Als deze bewerkingen minimaal in aantal zijn komen, in de eenvoudige matrixbewerkingen, de positieve APL-karakteristieken beter tot hun recht. De bovenstaande CPU-tijden van de Box-Muller methode en de Centrale Limiet Stelling zijn hiervan een voorbeeld.
Ahrens & Dieter (1972) hebben met een IBM-360 computer de rekentijd vergeleken van FORTRAN-versies van ondermeer de Box-Muller methode, de Centrale Limiet Stelling (n=12) en Marsaglia's Polar methode. Deze bedroegen respectievelijk .1093, .0575 en .0803 CPU-seconden voor de simulatie van een steekproef van 1000 uit de N(0,1)-verdeling. In het meest gunstige geval, de Box-Muller methode, blijkt de CPU-tijd van het APL-algoritme een factor 2 groter te zijn dan de FORTRAN-uitvoering. Bij meer bewerkelijke algoritmes komt de grotere snelheid van FORTRAN nog duidelijker naar voren; bij Marsaglia's Polar methode bedraagt het verschil een factor 32.

De kwaliteit van de N(0,1)-verdelingen is per methode met 30 replicaties onderzocht. Deze replicaties kunnen dienen als een indicatie van de kwalitatieve stabiliteit van de N(0,1)-verdelingen. Van de eerste drie metingen zijn tevens de U(0,1)-verdelingen bekeken om eventuele relaties tussen de kwaliteiten van N(0,1) en U(0,1) te kunnen vaststellen.

De uitkomsten van de Kolmogorov-Smirnov toets (K-S) en de Runs-toets van Levene & Wolfowitz (L-W) zijn in Tabel II aangegeven met de p-waarden voor iedere toets van de eerste drie N(0,1)-verdeelde grootheden en de benodigde U(0,1)-verdeelde grootheden. Het gemiddelde-X en de standaarddeviatie-S D van de p-waarden van de Kolmogorov-Smirnov toetsen van N(0,1) over de 30 metingen worden eveneens in Tabel II vermeld.

De p-waarden kunnen als volgt geïnterpreteerd worden:

Kolmogorov-Smirnov toets - p=1/p=0 : De verdeling komt exact/niet overeen met de gespecificeerde verdeling.

Runs-toets - p=1/p=0 : De getallen in een reeks zijn onafhankelijk/afhankelijk van elkaar.

De toetsresultaten zijn gebaseerd op de grootheden met lengte n = 250.

Tabel II: Toetsresultaten.

	METING										
	1 - 30		1			2			3		
TOETS	K-S op N		K-S	K-S	L-W	K-S	K-S	L-W	K-S	K-S	L-W
METH.	X	S D	op N	op U	op U	op N	op U	op U	op N	op U	op U
B-M	.525	.286	.324	.586	.900	.077	.760	.001	.695	.400	.250
			.872	.549	.100	.956	.175	.001	.765	.999	.750
				.138	.500		.053	.500		.479	.250
				.791	.001		.531	.001		.775	.900
				.663	.750		.299	.001		.310	.100
				.801	.001		.387	.500		.795	.001
				.516	.750		.229	.001		.916	.100
CLS	.560	.285	.775	.317	.025	.343	.811	.001	.565	.770	.001
				.091	.500		.639	.900		.819	.250
				.913	.750		.211	.001		.828	.250
				.708	.025		.260	.750		.750	.900
				.491	.001		.703	.025		.072	.900
				.566	.250		.946	.001		.400	.500
				.540	.050		.534	.975		.676	.500
				.544	.500		.565	.001		.464	.750
K-R	.541	.301	.090	.185	.250	.566	.110	.001	.004	.982	.001
				.274	.001		.998	.900		.087	.001
POLAR	.514	.338	.660	.445	.500	.124	.909	.500	.288	.267	.050
			.203	.708	.025	.808	.973	.025	.667	.612	.001
INV	.482	.347	.960	.909	.500	.176	.180	.900	.974	.977	.500

De methoden blijken alle in staat goede tot zeer goede benaderingen van N(0,1) te simuleren. Toch zijn er binnen de methoden aanzienlijke verschillen in de kwaliteit van N(0,1). De resultaten van de Kinderman-Ramage methoden geven hiervan een duidelijke illustratie.
Uit de bevindingen blijkt dat alle methoden de in Muller (1959) genoemde precisiegrenzen voor N(0,1) veelvuldig overschrijden. Deze verschillen zijn natuurlijk deels verklaarbaar als statistische afwijkingen in de benadering van N(0,1), maar zijn veel meer het gevolg van fluctuaties in de aselectheid van de gegenereerde U(0,1)-grootheden.
De kwaliteit van die uniforme verdelingen en derhalve de APL random-generator zijn toch redelijk goed te noemen. De resultaten van de Kolmogorov-Smirnov toets wijzen uit dat de U(0,1)-verdelingen over het algemeen voldoende uniform zijn. De p-waarden wijken niet significant af van hetgeen statistisch verwacht mocht worden (slechts 4 van de 60 gegenereerde verdelingen hebben een $p < .1$).

De resultaten van de Runs-toets van Levene & Wolfowitz tonen echter aan dat de uniforme verdelingen regelmatig te kampen hebben met monotone subreeksen, die qua lengte en aantal teveel afwijken van hetgeen verwacht mocht worden. Deze eigenschap van een uniform verdeelde grootheid heeft, gezien onze resultaten, geen zichtbaar nadelig effect op de kwaliteit van de N(0,1)-verdelingen.
De eigenschappen van de N(0,1) over een reeks van 30 simulaties wijzen erop dat de verschillende methoden kwalitatief vergelijkbare reeksen N(0,1)-grootheden weten te simuleren. Hierbij zijn wel enkele kanttekeningen te plaatsen. Zo zijn de gebruikte complexere of omvangrijkere methoden, over het geheel, niet in staat gebleken betere benaderingen van N(0,1) te produceren. De CLS methode blijkt met het gebruik van slechts 12 U(0,1)-grootheden al concurrerend te zijn met de overige methoden. Gezien de asymptotische eigenschappen van de Centrale Limiet Stelling kunnen we aannemen dat een groter aantal dan 12 deze methode dan ook, qua resultaten, zeker boven de overige zal uitstijgen.

Samenvattend mogen we stellen dat de Box-Muller methode en de Centrale Limiet Stelling redelijk goede en snelle methoden zijn om een standaard-normaal verdeelde grootheid te simuleren. Beide methoden zijn in APL zeer eenvoudig te programmeren en bieden de gebruiker derhalve een efficiente en goedkope mogelijkheid een reeks normaal verdeelde grootheden te produceren.

NAWOORD

De auteur is R. Visser en één van de reviewers dankbaar voor enkele waardevolle opmerkingen met betrekking tot een eerdere versie van dit artikel.

LITERATUUR

Ahrens, J.H. & Dieter, U., (1972), Computer Methods for Sampling from the Exponential and Normal Distributions, *Communications of American Computing Machinery*, *15*, 873-882.

APL-Language (1978), IBM-systems no. BC 26-3847-14, IBM.

Atkinson, A.C. & Pearce, M.C., (1976), The Computer Generation of Beta, Gamma and Normal Random Variables, *Journal of the Royal Statistical Society (A)*, *139*, 431-461.

Box, G.E.P. & Muller, M.E., (1958), A note on Generation of Normal Deviates, *Annals of Mathematical Statistics*, *28*, 610-611.

Chay, S.C., Fardo, R.D. & Mazumba, M., (1975), On using the Box-Muller Transformation with Mulitplicative Congruential Pseudo-Random Generators, *Applied Statistics*, *24*, 132-135.

Gebhardt, F., (1964), Generating normally distributed random numbers by inverting the normal distribution. *Mathematical Computing*, *18*, 302-306.

Golder, E.R. & Settle, J.C., (1976), The Box-Muller method for generating pseudo-random normal deviates. *Applied Statistics*, *25*, 12-20.

Hastings, C., (1955), *Approximations for digital Computers*. Princeton N.Y.: Princeton University Press.

Kennedy, W.J. & Gentle, J.E., (Eds.), (1980), *Statistical Textbooks and Monographs, vol. 33. Statistical Computing*. New York: Marcel Dekker.

Kinderman, A.J. & Ramage, J.G., (1976), Computer Generation of Normal Random Variables. *Journal of the American Statistical Association*, *71*, 893-896.

Knuth, D.E., (1969), *The art of Computer Programming, vol. 2: Seminumerical Algorithmes*. Reading, Mass.: Addison-Wesley.

Lehmer, D.H., (1951), *Mathematical Methods in Large-scale Computing Units, Proceedings of the Second Symposium on Large-scale Digital Computing Machinery*. Cambridge: Harvard University Press, 141-146.

Levene, H. & Wolfowitz, J., (1944), The covariance matrix of Runs Up and Down. *Annals of Mathematical Statistics*, *15*, 58-69.

Maritsas, D.G., (1973), A high speed and accuracy Digital Gaussian Generator of Pseudorandom numbers. *IEEE Trans. on Computers*, *c22*, 629-634.

Marsaglia, G., (1962), Random Variables and Computers, Information Theory Statistical Decision Functions Random Process: Transactions of the Third Praque Conference. In: J. Kozesnik (Ed.). Praque, Czechoslovak Academy of Sciences, 499-510.

Muller, M.E., (1958), An inverse method for generation of random normal deviates on Large-scale computers. *Mathematical Tables and other aids to Computation*, *12*, 167-174.

Muller, M.E., (1959), A comparison of methods for generating normal deviates on digital computers. *Journal of the Association of Computing Machinery*, *6*, 376-383.

Neave, H.R., (1973), On using the Box-Muller Transformation with Multiplicative Congruential Pseudo-Random Generators. *Applied Statistics*, *24*, 893-896.

Payne, W.H., (1977), Normal Random Numbers: Using Machine Analysis to choose the Best Algorithm. *Transactions on Mathematical Software*, *3*, 346-358.

Sahai, H., (1980), A supplement to Sowey's bibliography on random number generation and testing. *Biometrical Journal*, *22*, 447-461.

Sowey, E.R., (1972), A cronical and classified bibliography on random number generation and related topics. *International Statistical Review*, *40*, 355-371.

Wedderburn, R.W.M., (1976), A remark on Algorithm A829, 'The Runs Up and Down Test'. *Applied Statistics*, *25*, 193.

Wolfowitz, J., (1944), Asymptotic distribution of Runs up and Down. *Annals of Mathematical Statistics*, *15*, 163-172.

Ontvangen: 29-2-84