

Pembangkitan Bilangan Acak dengan Memanfaatkan Fenomena Fisis

Otniel 13508108

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia

¹if18108@students.if.itb.ac.id

Pembangkitan bilangan acak yang menggunakan komputasi secara matematis tidak dapat menciptakan deretan angka yang benar-benar acak. Hal itu disebabkan semua fungsi dalam matematika hanya dapat memetakan ke satu nilai saja, sedangkan yang diharapkan dari bilangan acak adalah ketika suatu karakter, katakanlah nilai a di acak, akan menghasilkan nilai yang beda tiap kali di acak. Hal tersebut tidak dapat diakomodasi oleh pembangkitan bilangan acak yang hanya menggunakan komputasi matematis saja.

Dalam dunia nyata, kejadian acak dapat terjadi seperti pelemparan dadu, gerakan debu, kecepatan gerakan tangan, dan sebagainya. Jika kita bisa menghubungkan kejadian fisis yang terdapat di dalam dunia nyata sebagai generator bilangan acak, maka bilangan true random dapat dibuat. Prinsipnya seperti menghubungkan pengocokan dadu pada computer, seperti mengocok dadu lewat mouse yang memanfaatkan variasi kecepatan drag mouse, namun medianya tidak harus menggunakan mouse.

Index Terms—bilangan acak, fungsi, pelemparan dadu, kejadian fisis, generator.

I. INTRODUCTION

Dalam penyembunyian informasi, terkadang kita membutuhkan sebuah pengacak nilai, namun pengacak nilai tersebut harus sulit dipecahkan. Pengacak nilai yang ada sekarang kebanyakan menggunakan metode pseudorandom, yaitu pembangkitan bilangan random yang sepenuhnya menggunakan komputasi matematis. Jadi hasil dari pseudorandom pasti selalu sama untuk suatu nilai, oleh karena itu pseudorandom banyak digunakan untuk pengenkripsian data karena data bisa di dekripsikan kembali.

Fungsi yang memetakan suatu angka ke banyak variasi angka kita katakan true random. Namun, sebenarnya terminology fungsi tidak sesuai dengan pembangkitan true random, karena pada dasarnya, dalam matematika, fungsi hanya dapat memetakan suatu nilai ke satu nilai tunggal. Jika dikatakan true random, maka sequence angka yang keluar tidak dapat diprediksi. Hal ini berguna untuk membuat kombinasi password pada web. Banyak aplikasi web, yang menyediakan fasilitas reset password dengan menggunakan bilangan acak. Dengan true random, kombinasi karakter password tidak akan memiliki pola.

Ada beberapa teknik yang dapat digunakan untuk membangkitkan bilangan random. Metode tersebut antara lain dengan menggunakan chaos theory dan Gaussian random. Namun keduanya sama-sama membutuhkan rutin yang membangkitkan bilangan secara acak.

II. TEKNIK RANDOM SECARA MATEMATIS

A. Chaos Theory

Banyak kejadian dalam kehidupan nyata yang tidak terprediksi kemunculannya. Misalnya, kejadian seseorang meninggal, kecelakaan kendaraan, dan sebagainya. Bahkan, kejadian alam seperti bencana tsunami, gempa, dan letusan gunung merapi muncul secara acak meskipun bisa diprediksi kemunculannya.

Jika dilihat, yang dilakukan ahli geologi mirip dengan yang dilakukan kriptanalis. Jika ahli geologi memprediksi terjadinya bencana dengan parameter keadaan alam, kriptanalis memanfaatkan frekuensi kemunculan karakter yang dienkrispikan dengan pseudorandom. Tentunya hal tersebut belum benar2 acak. Namun, yang terjadi pada kejadian alam murni merupakan kejadian yang acak.

Kejadian acak pernah dijelaskan dalam suatu teori yang dinamakan chaos theory. Chaos theory menggambarkan perilaku system dinamis nirlinjar yang menunjukkan fenomena chaos (bisa diartikan kekacauan atau acak). Chaos theory menjelaskan, salah satu karakteristik dari chaos theory adalah peka pada nilai awal (initial condition).

Sistem chaos deterministic dan tidak memiliki parameter acak. Bahkan, system chaos ini bisa dimodelkan dengan persamaan matematis yang disebut persamaan logistic

$$f(x) = rx(1 - x) \quad (1)$$

Dalam bentuk persamaan iterative menjadi

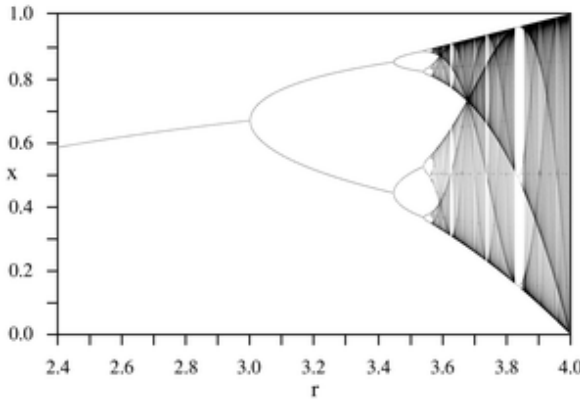
$$x_{i+1} = x_i r(1 - x_i) \quad (2)$$

Dengan r : laju pertumbuhan ($0 \leq r \leq 4$)

x : nilai chaos ($0 \leq x \leq 1$)

Persebaran nilai dari persamaan logistic diatas digambarkan dengan diagram bifurcation seperti di bawah

ini



Gambar 1 Bifurcation Diagram

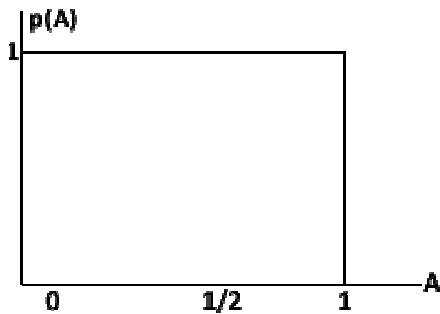
Masalah yang terdapat pada persamaan logistic tersebut adalah, variasi angka bukan merupakan parameter acak. Nilai tersebut harus didefinisikan secara heuristic oleh orang yang akan membangkitkan bilangan acak.

Selain dari logistic map, ada juga persamaan matematis henon map dan arnold's cat map.

B. GAUSSIAN RANDOM

Output dari random generator dikenal dengan nama random variable. Random variable memiliki range dari 0 sampai 1. Namun di computer, mustahil untuk menyajikan nilai dari 0 sampai 1 secara kontinu.

Fungsi dari uniform probability density untuk random variable A dinyatakan dengan $p(A)$. Random variable A memiliki mean yang nilainya $\frac{1}{2}$.



Gambar 2 Grafik dari fungsi peluang $p(A)$

Area di bawah $p(A)$ dikatakan probability distribution function dari random variable a dan didefinisikan sebagai integral dari minus tak hingga sampai a dari fungsi peluang p. Secara matematis dirumuskan dengan

Dan $F(A)$ memiliki range nilai dari 0 sampai 1.

Misalkan terdapat random variable C dengan fungsi probability distribution $F(C)$ yang memiliki range antara 0

sampai 1, akan dilakukan pembangkitan uniform distributed random variable A dalam range 0 sampai satu jika kita menset $F(C) = A$. Untuk mendapatkan nilai dari random variable C kita bisa lakukan inverse fungsi $F(C)$ dengan persamaan . Namun akan muncul permasalahan ketika ingin membangkitkan bilangan random yang memiliki normal distribution function.

Noise yang ditemui pada physical system sering kali dikarakteristikan dengan normal atau Gaussian probability distribution. Fungsi probabilitasnya adalah

$$= \quad (3)$$

dengan , dimana merupakan variansi dari C yang merupakan ukuran dari sebaran fungsi kerapatan probabilitas $p(C)$. Fungsi distribusi probabilitas $F(C)$ adalah area di bawah $p(C)$ dalam range

$$(4)$$

Namun, pengintegralan di atas tidak bisa diexpresikan dalam fungsi yang sederhana. Hal tersebut menimbulkan kesulitan. Tapi ada cara lain untuk mengatasi masalah tersebut. Dari teori probabilitas, diketahui (Rayleigh distributed) bahwa random variable R, dengan fungsi distribusi probabilitas

$$= \quad (5)$$

Berhubungan dengan pasangan Gaussian Random variable C dan D, dengan transformasi

$$(6)$$

$$(7)$$

dimana adalah distributed variable yang uniform pada interval . Parameter adalah variansi dari C dan D. Maka persamaan (4) dapat di inversekan untuk mendapatkan nilai R menjadi

$$= \quad (8)$$

Dimana A adalah uniform distributed random variable pada interval (0,1). Sekarang, jika kita membangkitkan uniform distributed random variable B yang kedua dan mendefinisikan

$$(10)$$

maka persamaan (6) dan (7), akan didapatkan dua statistically independent Gaussian distributed variable C dan D.

III. TRUE RANDOM

A. Kemunculan Nilai

True random memiliki kemunculan yang tidak terdefinisi. Tentu saja peluangnya bergantung pada semesta kejadian yang terlibat. Misalnya, peluang kemunculan mata dadu 1 pada dadu, dalam 6 kali pelemparan tidak akan menghasilkan peluang 1. Jika kembali pada teori kekacauan (chaos theory), maka kejadian tersebut bersesuaian dengan yang dikatakan pada chaos theory karena peluang tersebut tidak pernah bisa mencapai 1, karena dari tak hingga pelemparan dadu ada peluang untuk angka dadu satu tidak keluar. Penyebabnya tentu saja akibat nilai chaos yang selalu berubah. Untuk tiap pelemparan, seharusnya nilai chaos pada kejadian fisis akan selalu berubah. Oleh karena itu, peluang kemunculan angka dadu seharusnya tidak pernah bisa dimodelkan secara matematis.

Seperti telah dijelaskan di atas, peluang kemunculan dadu bermata satu seharusnya tidak terprediksi hanya dengan mengatakan “dengan 6 kali lemparan akan keluar angka 6”. Hal tersebut yang diharapkan terjadi pada pembangkitan bilangan true random. Kemunculannya tidak terduga. Dengan menggunakan satu sample nilai, dapat membangkitkan banyak nilai baru.

Permasalahan baru muncul, ketika pembangkitan secara true random dilakukan pada computer. Pada pseudorandom, saat eksekusi program, misalnya melakukan random dengan seed x , nilainya akan selalu a untuk tiap eksekusi. Sekarang, jika true random ingin diimplementasikan pada suatu program, tiap eksekusi untuk pembangkitan bilangan random akan menghasilkan nilai yang bervariasi, misal urutan keluaran angka per eksekusi, 1,4,2,3,4,4,5,6,6,7,2,4,7,5,9.

B. Prinsip Kerja

Prinsipnya dengan menggabungkan kejadian fisis yang terjadi diluar program. Misalnya, pada sebuah game yang melibatkan pengocokan dadu dengan memanfaatkan gerakan mouse dan kecepatan pergerakan. Kecepatan gerakan dan posisi tangan manusia tentunya akan bervariasi, akibatnya, hasil kocokan dadu akan tak terprediksi, atau dengan kata lain merupakan true random. Namun, hal tersebut akan sangat merepotkan jika tiap kali ingin membangkitkan bilangan acak, pengguna harus menggerakkan mouse terlebih dahulu.

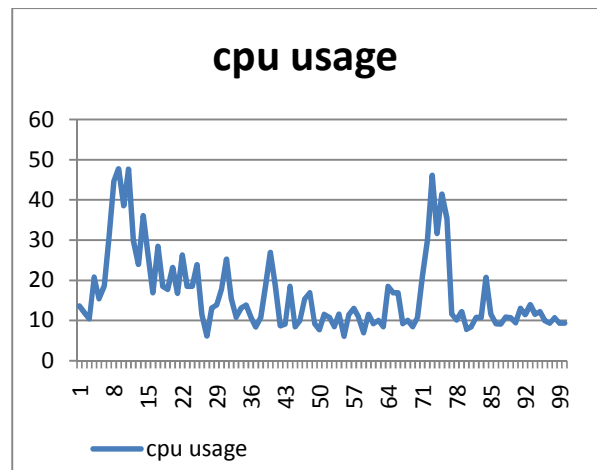
Prosesor computer memiliki kinerja yang berbeda-beda tiap waktunya. Hal tersebut dapat dimanfaatkan untuk membuat elemen random. Mirip dengan contoh pengocokan dadu seperti yang telah dijelaskan di atas, prosesor pun mengeluarkan persentase performa secara acak akibat dari ketidakteraturan dari lingkungan sekitar.

Pada persamaan logistic yang sudah dijelaskan di atas, terdapat bilangan chaos x . Bilangan tersebut bisa di inialisasi secara acak tiap awal eksekusi. Jadi, tiap eksekusi program untuk membangkitkan bilangan acak, nilai x tersebut akan selalu berbeda untuk tiap eksekusi.

C. Distribusi Nilai Performa CPU

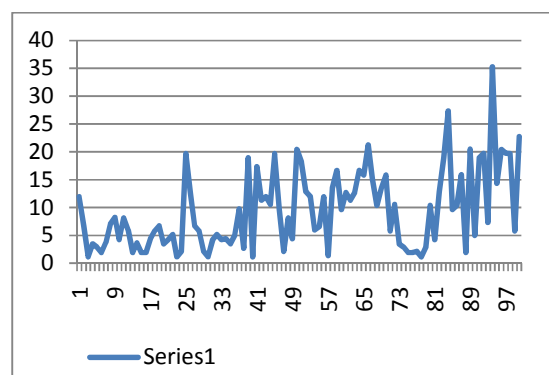
Performa dari sebuah CPU berbeda-beda setiap waktunya. Hal disebabkan karena keadaan lingkungan yang berubah-ubah pula. Gejala ini menunjukkan bahwa performa CPU merupakan sesuatu yang nilainya acak, namun sifatnya true random, bukan pseudo random. Nilainya tidak bisa dirumuskan kedalam suatu fungsi.

Berikut ini adalah hasil percobaan pembangkitan nilai performa CPU per detik. Sumbu mendatar, memiliki rentang nilai 1-100 yang memiliki arti pengujian dilakukan dalam waktu 100 detik. Kemudian sumbu tegak menunjukkan nilai performa CPU yang dicapai setiap detiknya.



Gambar 3 CPU Performance Distribution

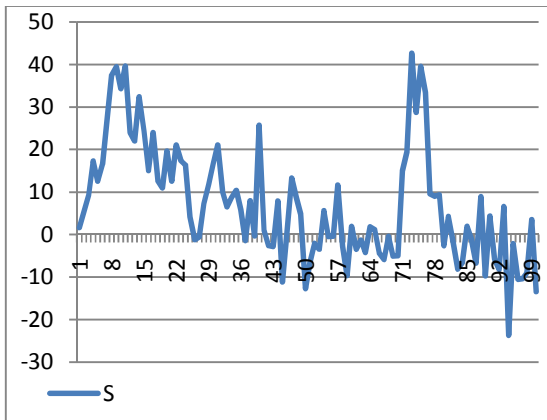
Dari gambar di atas terlihat tidak ada pola yang teratur dalam kemunculan nilai performa CPU. Semuan nilai dari detik pertama hingga detik terakhir tidak ada yang sama (nilai performa CPU dalam float). Kemudian dilakukan ujicoba lagi dengan keadaan yang berbeda. Kali ini, dilakukan task pada computer yang lebih sedikit dari pengujian yang pertama.



Gambar 4 Pengujian Performa dengan sedikit task

Terlihat pada nilai awalnya, memiliki nilai performa yang tidak terlalu jauh berbeda, Namun secara eksak, keduanya memiliki perbedaan. Pada pengujian yang pertama, nilai

awal yang dihasilkan adalah sebesar 13.5949, namun pada pengujian yang kedua nilai awalnya menjadi 11.95995. Berikut ini adalah grafik selisih nilai yang dihasilkan dari dua pengujian (pengujian pertama-pengujian kedua) tersebut.



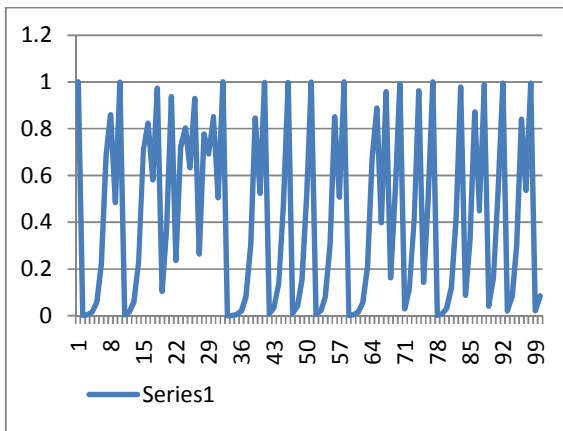
Gambar 5 Selisih antara performa test 1 dan test 2

Dari gambar di atas, selisih antara performa pengujian pertama dengan pengujian kedua tidak ada yang menghasilkan 0. Semuanya lebih dari 0 atau kurang dari 0. Namun, jika nilai performa dikonversikan ke int maka akan terdapat cukup banyak nilai yang sama antar pengujian 1 dan pengujian 2 di waktu yang bersamaan. Artinya, pada grafik selisih nilai performa 1 dan performa 2 akan ada yang menghasilkan nilai 0.

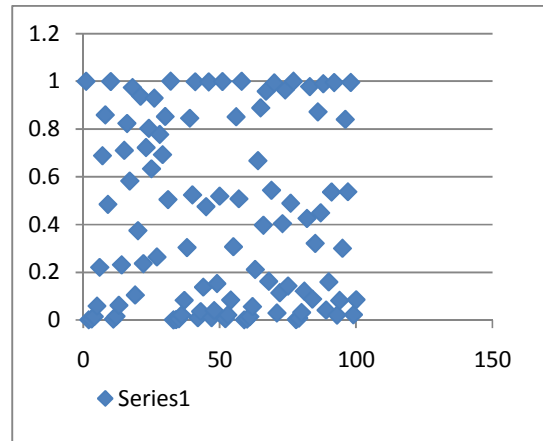
D. Implementasi Pada Persamaan Logistik

Pembangkit variable true random kita misalkan dengan nama rand(). Fungsi rand() ini nantinya akan mengembalikan nilai acak untuk menjadi inisial state pada nilai chaos.

Sekarang, persamaan (2) dimodifikasi menjadi $x_{i+1} = x_i r(1 - x_i)$ dengan $i = 0, x_0 = rand()$. Pemanggilan fungsi rand() tersebut dilakukan sekali ketika program dijalankan. Hasilnya, nilai x awal memiliki nilai true random.

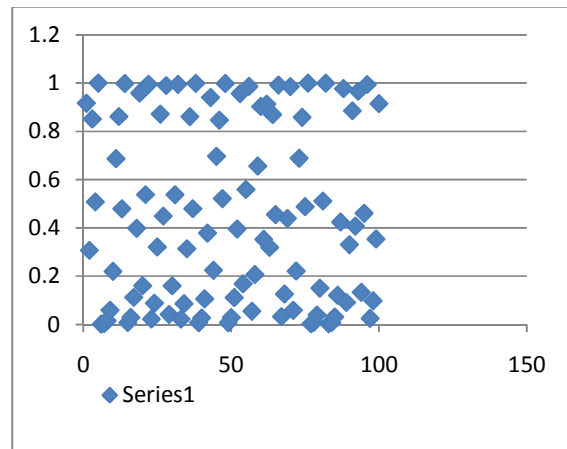


Gambar 6 Hasil dari penggunaan elemen true random pada persamaan logistic

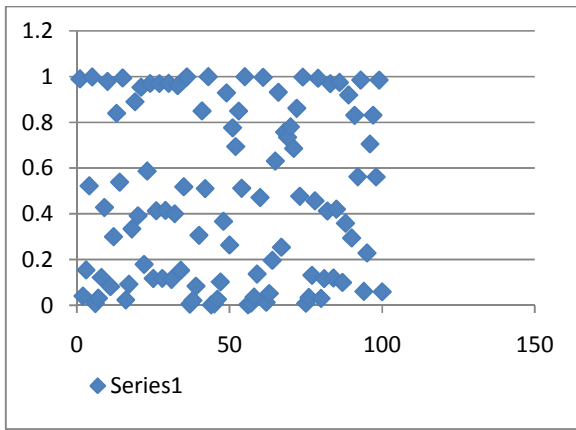


Gambar 7 Sebaran nilai true random dengan chaos

Sebenarnya, hasil yang dikeluarkan murni hasil dari persamaan chaos. Namun perbedaannya, metode pemilihan nilai awal dibangkitkan secara random. Ini yang membuat bentuk sebarannya selalu berbeda-beda pada tiap eksekusi. Berikut ini akan ditampilkan 2 bentuk sebaran dari eksekusi yang berbeda-beda.

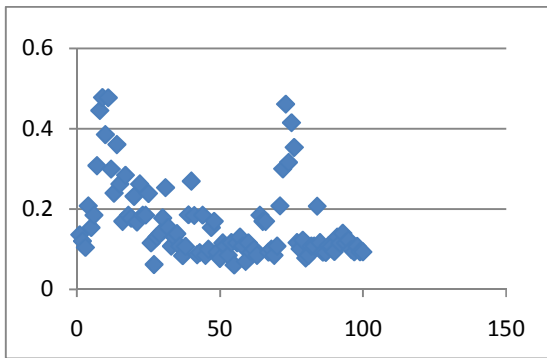


Gambar 8 Sebaran nilai dari eksekusi kedua

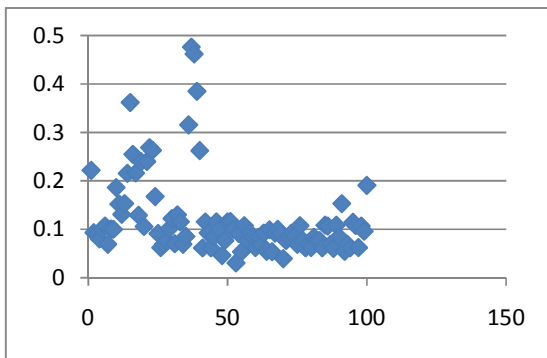


Gambar 9 Sebaran nilai dari eksekusi ketiga

Dari ketiga gambar tersebut, jika diperhatikan pada nilai pertamanya, terjadi variasi yang berbeda satu dengan yang lainnya. Hal ini disebabkan dari variasi true random yang dibandingkan dari nilai clock cpu yang sedang terpakai. Jika disajikan dalam bentuk diagram sebaran seperti diatas, hasil dari pembangkitan nilai acak dari performa cpu akan menjadi seperti gambar di bawah ini



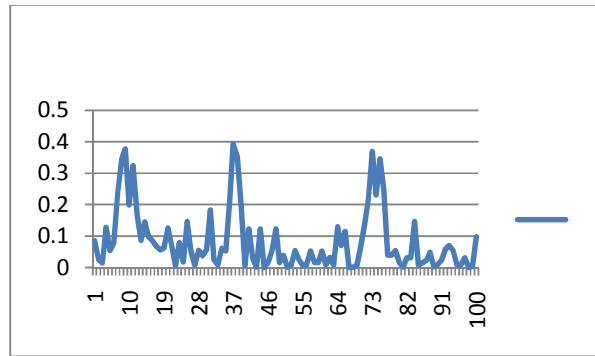
Gambar 10 Distribusi nilai true random 1



Gambar 11 Distribusi nilai true random 2

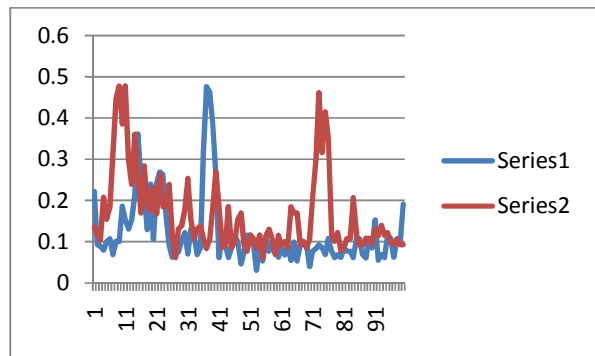
Dari kedua gambar di atas, terlihat dengan jelas bahwa variasi sebaran nilai memiliki pola yang jauh berbeda. Namun tetap ada kemiripan di pada bentuk sebaran nilainya. Pada gambar 10 dan gambar 11, terlihat sebaran

nilai memiliki kerapatan yang cukup tinggi pada interval antara detik ke 50 – 100. Hal tersebut dikarenakan penggunaan resource CPU menurun setelah eksekusi awal. Akibatnya nilai hamper terkonsentrasi dalam satu area. Meskipun begitu, tetap tidak ada nilai yang berhimpitan antara satu dengan yang lain. Hal tersebut juga bisa dibuktikan dengan menggunakan pendekatan grafik yang menggabungkan dua himpunan nilai dari gambar di atas.



Gambar 12 Grafik selisih performa pada gambar 10 dan gambar 11

Berikut Ini adalah grafik penggabungan dari 2 data pengujian yang digambarkan dalam gambar 10 dan gambar 11.



Gambar 13 Perbedaan nilai random pada kedua test

Garis yang berwarna biru menunjukkan data pada percobaan kedua dan garis yang berwarna merah menunjukkan dat pada percobaan pertama. Terlihat bahwa tidak ada titik-titik yang berhimpitan pada gambar tersebut. Hal itu menunjukkan bahwa true random generator berjalan dengan benar. Hasil yang muncul pada true random seharusnya memiliki peluang yang kecil untuk tiap proses dapat menghasilkan beberapa data saja yang sama. Meskipun hanya dua titik yang muncul berhimpitan, tetap memiliki peluang yang kecil. Karena data disajikan dalam pecahan yang memiliki ketelitian 1/1000.000, oleh karena itu, jika terdapat 100 data untuk kedua grafik, maka kemungkinan keluar paling tidak satu data yang sama antara grafik pertama denga grafik kedua

sebesar $\left(\frac{1}{1000.000} \times 1/100\right)^2$. Nilai tersebut sudah hampir menuju 0.

E. Perbandingan

Jika diperhatikan, bilangan yang dihasilkan dari true random yang menggunakan performance CPU memiliki sebaran yang agak terkonsentrasi di beberapa titik. Persebaran luas hanya terjadi pada detik-detik pertama. Namun setelah itu nilai-nilai yang dihasilkan hampir bersinggungan. Hal tersebut dikarenakan performansi yang hampir stabil. Fenomena tersebut bisa dilihat pada gambar 10 dan gambar 11. Jika diperhatikan, nilai persebaran pada detik ke 50 sampai dengan detik ke 100 memiliki densitas yang cukup tinggi. Bahkan, dari hasil analisis grafik, selisih (delta) dari nilai yang dihasilkan pada grafik 10 dan grafik 11, khususnya pada detik ke 50 sampai dengan 100, cukup banyak yang nilainya mendekati 0.

Untuk pemanfaatan random variable di persamaan logistic, sebaran yang dihasilkan sudah cukup tersebar. Setidaknya, dari grafik, terlihat bahwa nilai-nilai acak yang sudah dibangkitkan tidak terlalu terkonsentrasi pada suatu titik, namun tersebar ke beberapa area. Sebaran tersebut ditunjukkan pada gambar 7, 8, dan 9. Ketiganya memiliki pola yang sama, namun terjadi sedikit variasi karena perbedaan nilai awal.

Jika dibandingkan antara penggunaan bilangan acak yang dibangkitkan hanya dari pengambil nilai performa CPU dengan gabungan antara persamaan logistic dengan performa CPU, hasil sebaran yang terbaik adalah yang hasil dari penggabungan performansi CPU dengan persamaan logistic. Hal tersebut dikarenakan terjadi variasi yang cukup merata untuk setiap titik pada grafik sebaran nilai yang menggunakan metode gabungan tersebut.

Pengimplementasian dilakukan dengan menggunakan persamaan dari chaos theory (logistic equation) karena fungsinya relative sederhana dan hanya melibatkan perkalian, meskipun perkalian yang dilakukan menggunakan nilai mengambang. Berbeda dengan penggunaan Gaussian distribution (normal distribution). Pada normal distribution, penghitungan lebih rumit karena melibatkan perhitungan akar dan logaritma natural.

IV. KESIMPULAN

Pembangkitan rutin bilangan true random memang sangat membantu untuk menghasilkan nilai yang benar-benar acak bukan secara rumus, namun secara fisis seperti variasi performa yang dihasilkan CPU. Namun, dalam prakteknya, tetap diperlukan persamaan matematis untuk dapat menghasilkan bilangan yang sebarannya cukup bervariasi. Persamaan logistic telah memberikan distribusi yang cukup baik terhadap persebaran nilai di tiap satuan waktu. Hal tersebut sekaligus memperbaiki persebaran yang dihasilkan hanya dari elemen true

random saja.

Penggunaan true random dengan hanya memanfaatkan variasi performa CPU tidak terlalu baik karena ketika CPU berjalan normal atau hanya terjadi kenaikan atau penurunan performa yang sedikit, hasilnya apabila dikonversikan ke bilangan integer akan sering merujuk ke suatu nilai yang sama.

Sebaran nilai yang hanya menggunakan nilai random dari performa computer tidak terlalu bagus, karena persebaran nilainya tidak merata. Jika dibandingkan dengan yang menggunakan metode campuran, yakni penggabungan antara elemen true random yang dibangkitkan dari performa computer dengan persamaan logistic, nilai persebarannya lebih merata yang menggunakan metode campuran ini. Hal tersebut sebenarnya sudah ditunjukkan pada pola persebaran persamaan logistic. Berapapun nilai awal yang diberikan akan selalu memberikan pola yang sama. Hal tersebut terlihat dari gambar 7, 8, 9. Nilai yang penyebarannya merata lebih baik dari yang relative terkonsentrasi pada suatu daerah (contoh pada gambar 10 dan 11). Artinya, pada suatu waktu kemunculan suatu nilai (jika rentang nilai yang diambil besar) akan semakin kecil karena persebaran yang merata.

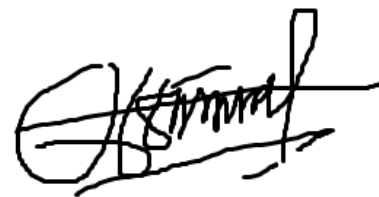
REFERENCES

- Proakis John G, "Digital Signal Processing," Edition 2, J. New Jersey: Prentice Hall, 1996, pp. 909-914.
Rinaldi Munir, Slide Kuliah Kriptografi
http://en.wikipedia.org/wiki/Normal_distribution
<http://cnx.org/content/m11250/latest/>
<http://www.taygeta.com/random/gaussian.html>
http://en.wikipedia.org/wiki/Chaos_theory

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 8 Mei 2011



ttd

Otniel 13508108