

MODELADO Y LA SIMULACION I

TAREA N° 2

Batería de pruebas de uniformidad e independencia

Fecha de entrega: miércoles 29 de febrero de 2012 antes de las 6 p.m. por e-mail

Considere los siguientes generadores de números aleatorios:

$$X_n = 16807X_{n-1} \bmod (2^{31} - 1) \quad r_n = \frac{X_n}{2^{31} - 1}$$

$$Y_n = 48271Y_{n-1} \bmod (2^{31} - 1) \quad s_n = \frac{Y_n}{2^{31} - 1}$$

$$Z_n = 69621Z_{n-1} \bmod (2^{31} - 1) \quad t_n = \frac{Z_n}{2^{31} - 1}$$

Se debe efectuar pruebas de uniformidad usando Kolmogorov-Smirnov (KS) y de independencia usando correlación serial (CS) con desplazamiento entre 1 y 5, con $\alpha = 0.05$, al primer generador (X_n). Estas pruebas se le harán a muestras de este generador. El tamaño de una muestra será uniforme entre 40 y 55. Para generar una variable aleatoria uniforme entera en $[m, n]$ se usa $\lfloor m + (n - m + 1)u \rfloor$ donde u es un número aleatorio. Este tamaño se generará usando la secuencia Y_n . Es decir, usaremos $\text{Trunc}(40 + (55 - 40 + 1)s_n)$ o $\text{Trunc}(40 + 16s_n)$. La semilla de la muestra se generará usando Z_n .

Con más detalle. Al comenzar el programa se pide el número de pruebas y las semillas de las secuencias Y_n y Z_n (ver ejemplo en la pagina siguiente). Para X_n no hay que pedir semilla ya que esta se generará usando Z_n . Luego, para las primeras pruebas generamos el tamaño de la muestra usando $\text{Trunc}(40 + 16s_1)$ y usamos Z_1 como la semilla X_0 (los t_n no se usan). Generamos los valores de la muestra (x_n) y se hacen las pruebas. Para las segundas pruebas generamos el tamaño de la muestra usando $\text{Trunc}(40 + 16s_2)$ y usamos Z_2 como la próxima semilla X_0 . Generamos los valores de la muestra y se hacen las pruebas. Así sucesivamente.

Los 3 generadores deben ser implementados usando el método de **Schrage**, tal como se vio en clase y aparece en las notas, en donde se expresa $ax \bmod m$ como $g(x) + mh(x) \dots$ (pagina IV-5).

Dado que se debe ordenar la muestra para hacer la prueba K-S, puede usar:

```
procedure sort(var v:array of real; n:integer);
{ Metodo más primitivo para ordenar un arreglo
  v: es dirección en memoria del vector a ser ordenado
  n: es el numero de elementos a ser considerados en el vector.
  Ojo, independientemente de como este definido el vector que se este pasando
  Como parámetro, dentro de la rutina sort el primer elemento es v[0], ya que
  no se esta especificando el rango de los índices. Por ejemplo, si
  externamente se esta definiendo datos: array[10..50] of real y luego se
  llama sort(datos,20), el primer elemento dentro de sort es v[0] y no v[10],
  y solo se ordenaran los primeros 20 elementos de datos}

var i,j:integer;
    x:real;
begin
  for i:=0 to n-2 do
    for j:=i+1 to n-1 do
      if v[i]>v[j] then begin
        x:=v[j];
        v[j]:=v[i];
        v[i]:=x;
      end;
    end;
end;
```

Hay que mostrar los resultados de las pruebas e indicar cuantas fueron pasadas y cuantas no tal como se muestra en la figura siguiente (usted debería obtener exactamente los mismos valores mostrados para las entradas dadas - las 3 primeras líneas). Dcal es el valor calculado de la prueba y Dtab es el valor teórico que por ser las muestras mayores a 35 se calcula fácilmente como $\frac{1.36}{\sqrt{n}}$. Para la prueba de correlación serial, si uno de los intervalos de confianza para alguno de los desplazamientos no incluye el cero, considere que la muestra no pasa la prueba

```

C:\ArchivosNuestros\Berti\Simulacion\B200
Numero de muestras: 2
Semilla generador auxiliar 1: 113
Semilla generador auxiliar 2: 29

Semilla: 2019009 Valores: 40
Dcal: 0.190 Dtab: 0.215 OK

Distancia Autocovarianza Intervalo de Confianza 95%
k Rk Limite Inferior Limite Superior
-----
1 0.0207402 -0.0053874 0.0468678 OK
2 0.0180445 -0.0084246 0.0445136 OK
3 -0.0069174 -0.0337418 0.0199071 OK
4 0.0041006 -0.0230938 0.0312951 OK
5 0.0087067 -0.0188735 0.0362869 OK

Semilla: 978988534 Valores: 49
Dcal: 0.149 Dtab: 0.194 OK

Distancia Autocovarianza Intervalo de Confianza 95%
k Rk Limite Inferior Limite Superior
-----
1 -0.0056564 -0.0292075 0.0178946 OK
2 0.0344401 0.0106398 0.0582404
3 0.0099837 -0.0140740 0.0340413 OK
4 -0.0067437 -0.0310672 0.0175797 OK
5 0.0144931 -0.0101052 0.0390914 OK

Numero de veces que paso KS: 2 <100.0%> no paso: 0 < 0.0%>
Numero de veces que paso CS: 1 < 50.0%> no paso: 1 < 50.0%>

```

Se debe **enviar solo el código fuente** (el “.dpr”) a hhoeger@ula.ve. **NOTA:** no enviar ejecutables (.exe) ya que el servidor de correos de la ULA no los deja pasar. Recuerde que:

- Cuando se reciba la tarea, se le devolverá un correo indicando que la misma fue recibida. Mantenga este correo (no lo borre) ya que es el único medio de probar que fue enviada. Sin el no hay posibilidad de reclamar que usted la envió y no fue corregida.
- El código debe cumplir con estilo de programación (principalmente indentación) y ser modular
- Estar debidamente identificado y documentado.
- Pueden formar un grupo de un **máximo de 3 (tres)** personas para la realización de la tarea.