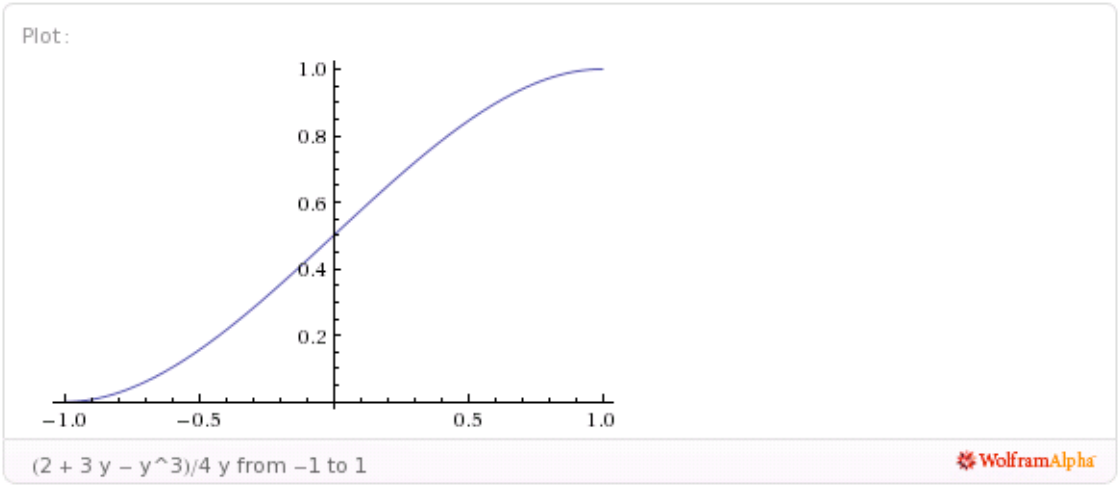
Alumno: Pau Gispert Victory
Métodos Estocásticos – Profesor Raúl Toral
6.12.2011

5. - Utilizar el método de inversión numérica para generar números aleatorios distribuidos según la f.d.p. $f_{\hat{x}}(x) = \frac{3}{4}(1-x^2)$ para $x \in (-1,1)$. Comprobar que los números generados satisfacen dicha distribución comparando la distribución exacta con el histograma obtenido numéricamente.

Para empezar, vamos a encontrar la función acumulativa:

$$F(x) = \int_{-1}^{x} \frac{3}{4}(1-y^2)dy = \ldots = \frac{1}{4}(-x^3+3y+2)$$



Para aplicar el método de la inversión numérica debemos de encontrar las M raices de la ecuación $F(x_i)=i/M$ para poder tabular $x_i=F^{-1}(i/M)$. Con ello, conseguimos discretizar a un número M el continuo hipotéticamente "infinito" de soluciones que supondría calcular cada vez, para cada número U(0,1) el valor de F^{-1}.

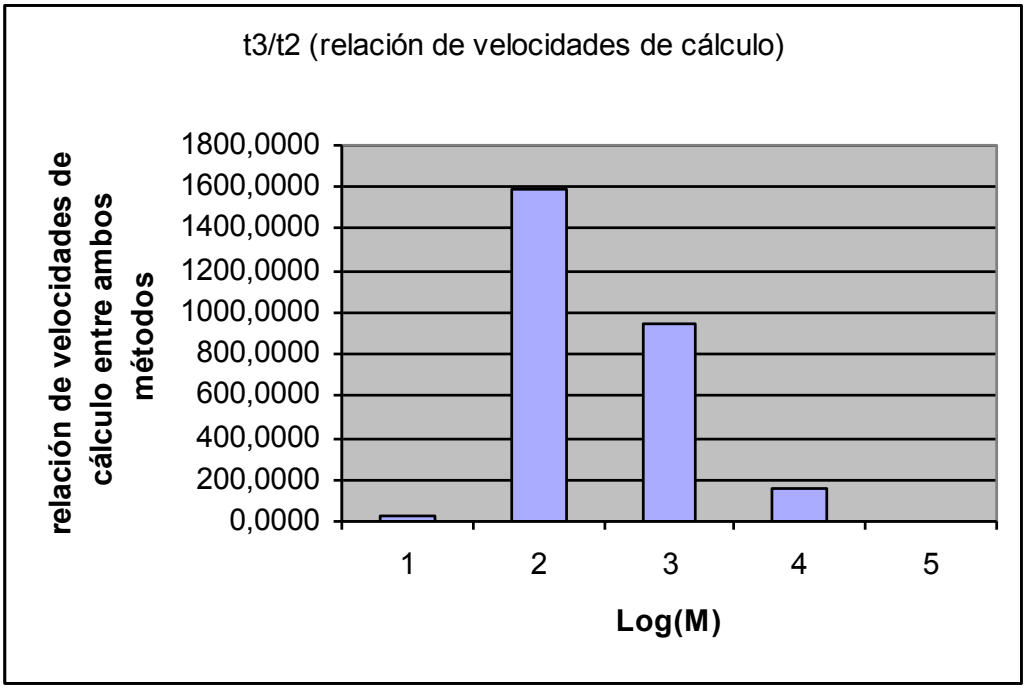
Cuando hayamos hecho este paso del ejercicio, podremos aplicar la interpolación lineal mediante la fórmula:

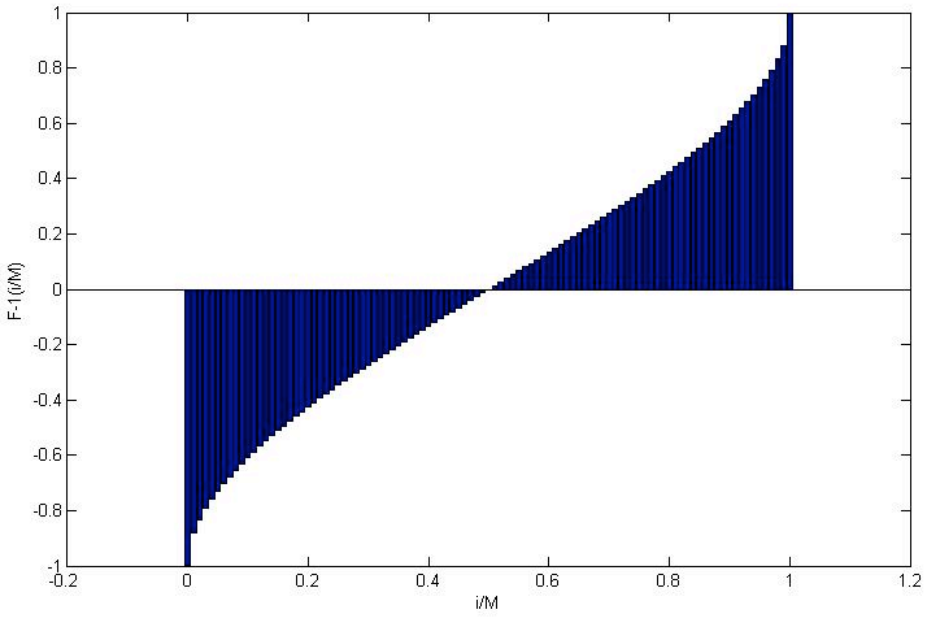
$$x = (M\hat{\mathbf{u}} - i)x_{i+1} + (i+1-M\hat{\mathbf{u}})x_i$$

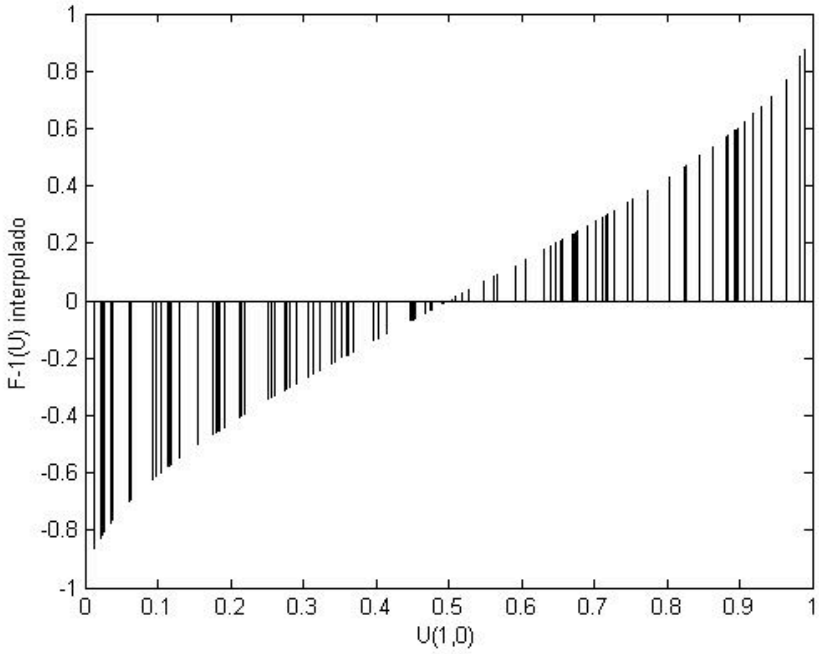
El ejercicio nos pide comparar los datos para el caso exacto (he usado el método de Newton Raphson con una tolerancia de `0.0000001)` con el caso interpolado. Para ello, he calculado el coeficiente de determinación lineal R^2 entre el caso exacto y el caso interpolado (para el mismo conjunto de U(1,0)):

Resultados:

log(M)	1	2	3	4	5
t1, tiempo de Calculo de xi=F-1(i/M) (s)	0,0488	0,2627	2,3420	22,7456	251,3161
t2, tiempo de interpolación para U(1,0) (s)	0,0013	0,0002	0,0025	0,1411	41,9654
t3, tiempo de calculo para U(1,0) (s)	0,0304	0,2411	2,3267	22,5906	255,8971
t4, tiempo de cálculo para otros U(1,0) (s)	0,0337	0,2446	2,2740	22,9583	254,7455
R^2	0,7413	0,9968	1,0000	1,0000	1,0000
t3/t2 (relación de velocidades de cálculo)	24,0269	1584,5446	940,0326	160,1280	6,0978

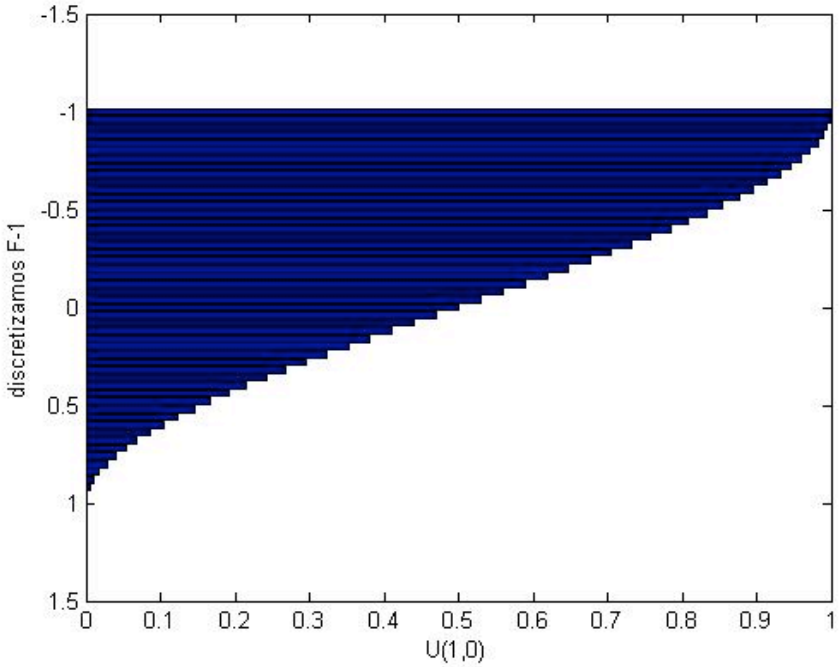






La velocidad relativa del método interpolado con respecto al método exacto nos da un pico para M=100 siendo R^2 prácticamente 1. Sin embargo,es posible que para según qué calculos puede que este valor sea pequeño. Vemos que la velocidad para M superiores va descendiendo considerablemente, pero en todo caso , indica que mejoramos la agilidad de nuestros programas con el método de inversión numérica y interpolación.

Ahora, vamos a estudiar qué ocurre si en vez de equispaciar los valores de F lo hacemos para F^{-1}.



En esta situación no hace falta calcular la inversa, pues basta con permutar los ejes, pero para interpolar debemos de buscar en cada ocasión la posicion del intervalo de interpolación necesario(antes era M*u directamente);veamos qué ocurre con los tiempos:

log(M)	1	2	3	4	5
tiempo para discretizar x y calcular F	0,0001	0,0007	0,0036	0,1483	40,8628
tiempo para interpolar x para buscar F-1	0,0014	0,0016	0,0359	2,5815	271,9746
relación velocidades metodo interpolación correcto/incorrecto	1,1076	10,5143	14,5045	18,2983	6,4809

Vemos que el método correcto, como no podia ser de otra manera, es más rapido , por tanto, aunque invirtamos un tiempo relativamente considerable al principio, al calcular los M valores de F-1(i/M), después podemos de forma rapidísima, interpolar con R^2 rozando 1 cualquier valor de U, ya que no hay que buscar la posición del intervalo, pues viene dado por el producto M*u.

Algoritmos en MATLAB:

```
for k=1:5;
x=sym('x');
f=(2+3*x-x^3)/4;
fprima=3/4*(1-x^2);
```

```
f=inline(f);
fprima=inline(fprima);

    t1=tic;
tol=0.0000001;
M=10^k;

for i=1:M+1;% newton-raphson para M puntos equidistantes, solo se
calcula 1 vez
x0=0.00000001;
j=0;
ejex(i)=(i-1)/M;
x1=x0-(f(x0)-(i-1)/M)/fprima(x0);
while abs(x0-x1)>tol;
j=j+1;
x0=x1;
x1=x0-(f(x0)-(i-1)/M)/fprima(x0);
end
g(i)=x1;
end;
t2=toc(t1);

t1=tic;
 for i=1:M+1; % realizamos interpolación lineal para los M's U(0,1)
 u(i)=rand();
int=floor(M*u(i))+1;
interpRand(i)=(M*u(i)-int)*g(int+1)+(int+1-M*u(i))*g(int);
end;
t3=toc(t1);

t1=tic;
for i=1:M+1; %realizamos newton raphson para los anteriores M's U(0,1)
    x1=0;
x0=0.00000032;
j=0;
x1=x0-(f(x0)-u(i))/fprima(x0);
while abs(x0-x1)>tol;
j=j+1;
x0=x1;
 x1=x0-(f(x0)-u(i))/fprima(x0);
end;
h(i)=x1;
end;
t4=toc(t1);

t1=tic;
 for i=1:M+1; %realizamos newton raphson para M randoms cualquiera
    x1=0;
x0=0.00000032;
 j=0;
p=rand();
x1=x0-(f(x0)-p)/fprima(x0);
while abs(x0-x1)>tol;
j=j+1;
x0=x1;
 x1=x0-(f(x0)-p)/fprima(x0);
end
hh(i)=x1;
 t5=toc(t1);
```

```
end;

 %calculamos el coeficiente de determinacion lineal R^2
ycal=sort(interpRand); % ordenamos de menor a mayor
ym=sort(h);
 n = length(ym);
yav = mean(ym);
 s1 = 0;
s2 = 0;
for(i=1:1:n);
s1 = s1 + (ym(i)-yav)^2;
s2 = s2 + (ym(i)-ycal(i))^2;
end;
z(k) = 1 - s2/s1

tiempos(1,k)=t2 ;%representa el tiempo para el calculo de los M puntos
de F-1
tiempos(2,k)=t3 ;%representa el tiempo para la interpolacion de M
randoms
tiempos(3,k)=t4 ;%representa el tiempo para el calculo de M randoms
anteriores
tiempos(4,k)=t5 ;%representa el tiempo para el calculo de M randoms
cualquiera
 end;
```

Y este último algoritmo es el método erróneo, en el que discretizamos directamente F^{-1}

```
%numero de pasos m; generamos la funcion F

k=5;
t1=tic;
m=10^k;
for n=1:m+1;
    y(n)=2*(n-1)/m-1; %esto representa F-1
x(n)=(2+3*y(n)-y(n)^3)/4; % y esto F
end;
t2=toc(t1)
t1=tic;
% construimos la interpolación;

for i=1:m;
u(i)=rand();
% buscamos el intervalo donde vamos a interpolar;
for n=1:m;
    if u(i)==0;
        p=1;
    end;
    if x(n)<u(i)&&x(n+1)>=u(i);
        p=n;

    end;
end;


randx(i)=((y(p+1)-y(p))/(x(p+1)-x(p)))*(u(i)-x(p))+y(p);

end;
t3=toc(t1)
```