toto-share.com

sharing everything

WEB HOSTING MONEY ONLINE ASCII TABLE ABOUT ME

You are here: Home / 2011 / January / *C/C++ create random noise (gaussian noise/white noise)*

C/C++ create random noise (gaussian noise/white noise)

By totosugito on January 28, 2011

I was amazed when use randn command at Matlab. randn command will generate random data every we call that command. After I search at google, I found how to make this happen. I get this code at seismic unix source code. This code will generate random noise or white noise with Gaussian method. Code for main.c is :

```
#include <stdio.h>
#include <stdlib.h>
#include "frannor.h"

int main(int argc, char **argv)
{
    double *noise=NULL;
    int i;
    int ndata;
    unsigned int seed;

    if(argc!=2)
    {
        fprintf(stderr, "usage : random 5");
        exit(0);
    }

    ndata = atoi(argv[1]);

    if (-1 == (seed = (unsigned int) time((time_t *) NULL
    {
        fprintf(stderr, "time() failed to set seed");
        exit(0);
    }

    srannor(seed);  //seed random number generator
    noise = (double*) calloc (ndata, sizeof(double));
    for(i=0; i<ndata; i++)  //create random data
    {
        /* Compute noise vector elements in [-1, 1] */
        /* GAUSS METHOD. frannor gives elements in N(0,1)
        noise[i] = (double) frannor();
    }

    for(i=0; i<ndata; i++)
        printf("%3.2f    ", noise[i]);
```

Search

Categories

.NET (3)
Android (1)
C (42)
C++/Qt (47)
freelance (6)
Internet (18)
Linux (51)
Matlab (9)
Notes (4)
Python (7)
Windows (11)

```
    printf("\n");

    free(noise);  //free allocated memory
    return(1);
}
```

This is code for frannor.h :

```
#ifndef FRANNOR_H_
#define FRANNOR_H_

#include <math.h>
#define ABS(x) ((x) < 0 ? -(x) : (x))
#define AA 12.37586
#define Bsu 0.4878992
#define Csu 12.67706
#define C1 0.9689279
#define C2 1.301198
#define PC 0.01958303
#define XN 2.776994
#define OXN 0.3601016
#define NBITS 24

/* macro to generate a random number uni uniform on [0,1)
#define UNI(uni) \
    uni = u[i]-u[j]; if (uni<0.0) uni += 1.0; u[i] = uni;
    if (--i<0) i = 16;  if (--j<0) j = 16

float frannor(void);
void srannor (int seed);

#endif /* FRANNOR_H_ */
```

This is code for frannor.c :

```
/* Copyright (c) Colorado School of Mines, 2010.*/
/* All rights reserved.                       */

/*********************** self documentation ***********
/*****************************************************
FRANNOR - functions to generate a pseudo-random float nc
        with N(0,1); i.e., with zero mean and unit varia

frannor     return a normally distributed random float
srannor     seed random number generator for normal dist

******************************************************
Function Prototypes:
float frannor (void);
void srannor (int seed);

******************************************************
frannor:
Input:      (none)
Returned:   normally distributed random float

srannor:
Input:
seed        different seeds yield different sequences of

******************************************************
Notes:
Adapted from subroutine rnor in Kahaner,  Moler and Nash
which in turn was based on  an algorithm by
Marsaglia and Tsang (1984).

******************************************************
References:
"Numerical Methods and Software", D.
Kahaner, C. Moler, S. Nash, Prentice Hall, 1988.

Marsaglia G. and Tsang, W. W., 1984,
A fast, easily implemented method for sampling from decr
unimodal density functions:  SIAM J. Sci. Stat. Comput.,
p. 349-359.

********************************************************
Author:  Dave Hale, Colorado School of Mines, 01/21/89
********************************************************
/**************** end self doc *************************

#include "frannor.h"

static float v[]={
    0.3409450, 0.4573146, 0.5397793, 0.6062427, 0.663169
    0.7136975, 0.7596125, 0.8020356, 0.8417227, 0.879210
    0.9490791, 0.9820005, 1.0138492, 1.0447810, 1.074925
    1.1332738, 1.1616530, 1.1896010, 1.2171815, 1.244451
    1.2982650, 1.3249008, 1.3514125, 1.3778399, 1.404221
    1.4569915, 1.4834526, 1.5100121, 1.5367061, 1.563571
    1.6179680, 1.6455802, 1.6735255, 1.7018503, 1.730604
    1.7896223, 1.8200099, 1.8510770, 1.8829044, 1.915583
    1.9839239, 2.0198430, 2.0571356, 2.0959930, 2.136645
    2.2245175, 2.2725185, 2.3239338, 2.3795007, 2.440221
    2.5834658, 2.6713916, 2.7769943, 2.7769943, 2.776994
};

/* internal state variables for uniform random number ge
static  int i=16,j=4;
static  float u[]={
    0.8668672834288,  0.3697986366357,  0.8008968294805,
    0.4173889774680,  0.8254561579836,  0.9640965269077,
    0.4508667414265,  0.6451309529668,  0.1645456024730,
    0.2787901807898,  0.06761531340295, 0.9663226330820,
    0.01963343943798, 0.02947398211399, 0.1636231515294,
    0.3976343250467,  0.2631008574685
};

float frannor(void)
/*******************************************************
return a normally distributed random float
 *******************************************************
Returned:   normally distributed random float
 *******************************************************
{
    int k;
    float uni,vni,rnor,x,y,s,bmbx,xnmx;

    /* uni is uniform on [0,1) */
    UNI(uni);

    /* vni is uniform on [-1,1) */
    vni = uni+uni-1.0;

    /* k is in range [0,63] */
    k = ((int)(u[i]*128))%64;

    /* fast part */
    rnor = vni*v[k+1];
    if (ABS(rnor)<=v[k]) return rnor;

    /* slow part */
    x = (ABS(rnor)-v[k])/(v[k+1]-v[k]);
    UNI(y);
    s = x+y;
    if (s<=C2) {
        if (s<=C1) return rnor;
        bmbx = Bsu-Bsu*x;
        if (y<=Csu-AA*exp(-0.5*bmbx*bmbx)) {
            if (exp(-0.5*v[k+1]*v[k+1])+y*PC/v[k+1] <=
                    exp(-0.5*rnor*rnor)) return rnor;
            do {
                UNI(y);
                x = OXN*log(y);
                UNI(y);
            } while (-2.0*log(y)<=x*x);
            xnmx = XN-x;
            return (rnor>=0.0 ? ABS(xnmx) : -ABS(xnmx));
        }
    }
    bmbx = Bsu-Bsu*x;
    return (rnor>=0.0 ? ABS(bmbx) : -ABS(bmbx));
}

void srannor (int seed)
/*******************************************************
seed random number generator
 ******************************************************
Input:
seed        different seeds yield different sequences of
 ******************************************************
{
    int ii,jj,ia,ib,ic,id;
    float s,t;

    i = 16;
    j = 4;
    ia=ABS(seed)%32707;
    ib=1111;
    ic=1947;
    for (ii=0; ii<17; ii++) {
        s = 0.0;
        t = 0.5;
        for (jj=0; jj<64; jj++) {
            id = ic-ia;
            if (id<0) {
                id += 32707;
                s += t;
            }
            ia = ib;
            ib = ic;
            ic = id;
            t *= 0.5;
        }
        u[ii] = s;
    }
}
```

To compile this source code, use command :

gcc main.c frannor.c -lm -o random

This is output from program :

```
toto@toto-laptop:/home/toto/$ ./random 6
0.08   1.78   -1.35   -1.03   1.62   0.07
toto@toto-laptop:/home/toto/$ ./random 6
1.12   1.32   0.73   -0.34   0.25   -0.58
```

Source : http://www.cwp.mines.edu/cwpcodes/index.html

totosugito

Hi, This is my notes when I try something. Maybe, this is NOT THE BEST SOLUTION for your problems. If you have other method or idea that better with my post, please share in this blog. Thank for visiting my site.

More Posts - Website

Related posts:

1. **Matlab : Create Random Data in Range**

2. **C++ create 2D array**
3. **C Create Dynamic 2D array**
4. **Create C/C++ shared library with MATLAB Compiler**

Posted in C, C++/Qt | Tagged create random number c, gaussian noise, randn matlab, random noise, random numbers c, white noise

One Response

Link Building Services
March 12, 2011 at 3:33 am | Permalink

Great Websie…

I loved this great post I saw today….

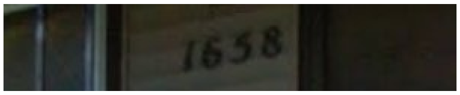
Leave a Reply

Name *

Email *

Website

Comment



Type the text

Privacy & Terms

Post Comment

← Previous

Next →