USER MANUAL

UNURAN for Windows

VERSION 0.6.0

ARVAG

UNURAN for Windows is a user-friendly implementation of algorithms for generating uniform and non-uniform random numbers wrapped around the following two distributions :

- *PRNG = Pseudo-Random Number Generator*
  (version 3.0.2)
  A portable, high-performance ANSI-C implementations of pseudorandom number generators such as linear congruential, inversive congruential, and explicit inversive congruential random number generators (called LCG, ICG and EICG, respectively) created by Otmar Lendl (versions >= 3 are being developed and maintained by Josef Leydold)
  http://statistik.wu-wien.ac.at/prng/index.html

- *UNU.RAN = Universal Non-Uniform RANdom number generators*
  (version 0.6.0)
  A collection of algorithms for generating non-uniform pseudorandom variate as a library of C functions.
  It provides an consistent and flexible interface to universal algorithm as well as to generators for standard distribution. It is applicable for continuous and discrete, univariate and multivariate distributions and for (re)sampling from empirical distributions.
  UNU.RAN is developed and maintained by Josef Leydold, Wolfgang Hörmann, Erich Janka, Günter Tirler and Roman Karawatzki
  http://statistik.wu-wien.ac.at/unuran/index.html

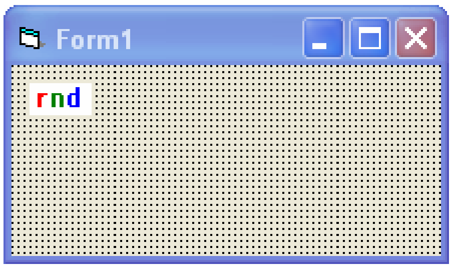
The currnent release of **UNURAN for Windows** includes mutually independent parts :

- An ActiveX control (unuran-0.6.0.ocx)

- A dynamic link library (unuran-0.6.0.dll)
  + declaration module (unuran-0.6.0.bas)

providing a very flexible tool for random number generation in the windows environment.
For remarks, problems, questions and/or suggestions please contact unuran@statistik.wu-wien.ac.at

Place an UNURAN ActiveX control onto your Form :



Assuming that the name of your ActiveX control is unuran1, add the initialization call to your source-code :

```
api_string = "normal(0,1)"
return_init = unuran1.init(api_string)
```

The initialization routine return 0 upon successfull initialization, any other value indicate an error in the api_string and or incopatible settings of parameter values, sampling method etc — in this case additional info on the error condition is written to the `unuran.log` file.

Optional seeding of the generator may be performed using the call

```
return_seed = unuran1.set_seed(seed_value)
```

To obtain a random variate x sampled from a distribution defined by the api_string, we can now write

```
x = unuran1.sample
```

Each succesive call to the sample-method will now yield another random variate from the given distribution.

Add the following declaration block to your project (can be found in the file `unuran-0.6.0.bas`) :

```
    Public Declare Function unuran_init Lib "unuran-
    0.6.0.dll" (ByVal s As String) As Long
    Public Declare Function unuran_sample Lib
    "unuran-0.6.0.dll" () As Double
    Public Declare Function unuran_set_seed Lib
    "unuran-0.6.0.dll" (ByVal seed As Long) As Long
```

Initialize your random number generator

```
    api_string = "normal(0,1)"
    return_init = unuran_init(api_string)
```

The initialization routine return `0` upon successfull initialization, any other value indicate an error in the api_string and or incopatible settings of parameter values, sampling method etc — in this case additional info on the error condition is written to the `unuran.log` file.

Optional seeding of the generator may be performed using the call

```
    return_seed = unuran_set_seed(seed_value)
```

To obtain a random variate x sampled from a distribution defined by the api_string, we can now write

```
    x = unuran_sample()
```

Each succesive call to the sample-method will now yield another random variate from the given distribution.

Following are some api_string examples. For a more detailed description please consult the original UNU.RAN manual as well as the documentation for the PRNG package ...

`"uniform(0,1)"`
: Standard uniform random number generator.

`"uniform(0,1) & urng=mt19937(163)"`
: Explicitly choosing the Mersenne-Twister with given seed.

`"gamma(5)"`
: Gamma-distribution with shape parameter.

`"gamma(5,1,7)"`
: Gamma-distribution with shape, scale and location parameter.

`"normal(0,1) & method=arou"`
: Choosing AROU as sampling method for the normal distribution

`"normal(0,1); domain=(1,2)"`
: Restricting the domain — sample only in the given interval.

`"distr=discr; pv=(0.5,0.2,0.3)"`
: Discrete distribution with given probability vector.

`"distr=cont; pdf='1-x*x'; domain=(-1,1) & method=tdr"`
: User-defined distribution.