Using UNU.RAN with Microsoft Visual C

UNU.RAN – Universal Non-Uniform RANdom number generators
Version 1.7.1, 26 April 2010

Josef Leydold
Wolfgang Hörmann
Engin Durmaz

Using UNU.RAN with Microsoft Visual C

This document describes how to use Microsoft Visual Studio to create a C or C++ project and link the target with UNU.RAN.

UNU.RAN (Universal Non-Uniform RAndom Number generator) is a collection of algorithms for generating non-uniform pseudorandom variates as a library of C functions designed and implemented by the ARVAG (Automatic Random VAriate Generation) project group in Vienna, and released under the GNU Public License (GPL).

This binary distribution also has support for Pierre L'Ecuyers RngStreams library. This library is already compiled into the distributed DLL, the corresponding header file is included in this package.

The DLL is linked against the C run-time library '`MSVCRT.lib`'. Thus the file '`MSVCR80.DLL`' must be available at run time for applications linked with UNU.RAN and '`MSVCRT.lib`' (i.e., included in the envirenment variate `PATH`). For details see the Microsoft Visual C++ manual.

Installation of Binary Distribution

1. Download unuran-1.7.1-win32.zip from the UNU.RAN download page.
2. Unpack unuran-1.7.1-win32.zip. It creates directory '`unuran`' which contains the following files:

   '`unuran.h`'
   : Header file to be included in your application source file. (Contains function prototypes, typedefs and macros.)

   '`RngStream.h`'
   '`unuran_urng_rngstreams.h`'
   : Header files required for using the '`RngStream`' library.

   '`libunuran17.def`'
   : Module-definition file. (Declares all functions exported the UNU.RAN library.)

   '`libunuran17.dll`'
   '`libunuran17.dll.manifest`'
   : DLL (Dynamic link library) and its manifest file (must always be located in the same directory).

   '`libunuran17.lib`'
   : Import library file.

   '`libunuran17.exp`'
   : Library export file.

   '`unuran.pdf`'
   : UNU.RAN User Manual.

   '`unuran_win32.pdf`'
   : Short description for building your own application using UNU.RAN with Microsoft Visual C.

   '`example1.c`'
   '`example2.c`'
   : Two example files.

3. *Optional:* Move directory '`unuran`' into an appropriate place, e.g. '`C:\unuran`' (or maybe the folder where you build your application).

Throughout this document, the UNU.RAN installation folder is referred to as '<UNURANDIR>'. For example, if UNU.RAN has been installed in the folder 'C:\unuran', references to '<UNURANDIR>\libunuran17.lib' represent 'C:\unuran\libunuran17.lib'.

4. *Optional:* Add '<UNURANDIR>' to the PATH environment variable. Thus your application can find the location of libunuran17.dll when it is started.

Uniform Random Number Generator

This binary distribution uses Pierre L'Ecuyers RngStreams library as its default uniform random number generator. This library uses a package seed. Thus you should add the following piece of code at the beginning of your application (at least before you call any of the UNU.RAN functions):

```
unsigned long seed[] = {111u, 222u, 333u, 444u, 555u, 666u};
RngStream_SetPackageSeed(seed);
```

where 111u, . . . , 666u should be replaced by your seeds.

Remark: UNU.RAN works with any source of uniform random numbers. We refer to the UNU.RAN for details if you want to use your own uniform random number generator instead of the 'RngStreams' library.

Building Your Own Project in Visual Studio

Note: The information below applies to the Visual C++ .NET 2005.

Let us assume that you want to build a target named 'example.exe' and have:

- a source file named 'example.c' which uses the C API of the UNU.RAN library (or alternatively a C++ file 'example.cpp');
- a folder where this file is located and which we refer to as '<MYAPPDIR>'.

One way to build your project is to create a *Solution* named 'example.sln' as described here.

1. Start Microsoft Visual Studio .NET 2005.
2. Build the 'example.sln' solution:

   From the 'File' menu, select 'New', and then 'Project'.

   When the 'New Project' dialog box appears then
   - In the 'Project Types' pane, select 'Visual C++ Win32 Projects'.
   - In the 'Templates pane', select the 'Win32 Console Project' icon.
   - Fill in the project name ('example').
   - If necessary, correct the location of the project (to '<MYAPPDIR>').
   - Click 'OK'.

   When the Win32 Application Wizard appears then
   - Click on 'Application Settings'.
   - Select 'Console Application' as application type.
   - Make sure that 'Empty Project' is checked in 'Additional Options'.
   - Click 'Finish'.

   This creates a solution, 'example', with a single project, 'example'. You can view the contents of the solution by selecting 'Solution Explorer' in the 'View' menu.
3. Add your source file to the project. From the 'Project' menu, choose 'Add Existing Item':

   - Move to folder '<MYAPPDIR>' and select 'example.c'.
   - Click 'Add'.

4. Set some options so that the project knows where to find the UNU.RAN include files and the UNU.RAN libraries.
   - From the 'Project' menu, choose 'example Properties'. The 'example Property Pages' dialog box appears.
   - In the 'Configuration' drop-down list, select 'Release'.
   - Select 'C/C++' in the 'Configuration Properties' tree.
   - Select 'General'.
   - In the 'Additional Include Directories' field
     . add directory '<UNURANDIR>';
     . choose 'No' for 'Detect 64-bit Portability Issues'.
   - Select 'Linker' in the 'Configuration Properties' tree.
     . Select 'General' and then select 'Additional Library Directories'.
     . Add directory '<UNURANDIR>'.
     . Select 'Input' and then select 'Additional Dependencies'. Add library file 'libunuran17.lib'.
   - Click 'OK' to close the 'example Property Pages' dialog box.
5. Set the default project configuration.
   - From the 'Build' menu, select 'Configuration Manager'.
   - Select 'Release' in the 'Active Solution Configuration' drop-down list.
   - Click 'Close'.
6. Finally, to build the project, from the 'Build' menu, select 'Build Solution'.

After completion of the compiling and linking process, the target is created. The full path of the executable is '<MYAPPDIR>\example\Release\example.exe'. Notice that, if you want to run the 'example.exe' by clicking on it, you need to locate the DLL file in the same directory or adjust the PATH environment variable for the DLL (see [Installation], page 1).

Building Your Own Project on the Command Line

First you have to set the appropriate environment variables to enable 32-bit command-line builds by means of the 'vcvars32.bat' file:

1. At the command prompt, change to the '\bin' subdirectory of your Visual C++ installation.
2. Run 'vcvars32.bat' by typing VCVARS32.

Then change to your application folder '<MYAPPDIR>') that contains your source file(s) (C or C++). Assume you have one C source file 'example.c'. Then you can compile and link your executable by

```
cl /O2 /W3 /MD /I<UNURANDIR> example.c libunuran17.lib /link /LIBPATH:<UNURANDIR>
```

which creates the file 'example.exe'.

When you want to run 'example' then the location of the DLL libunuran17.dll (and that of the C run-time library 'msvcr80.dll') have to be included in the environment variable PATH (or you need to locate these DLLs in the same directory).

Two Examples

Here we give small examples. These show

- How to seed the seed for the RngStreams library.
- How to create an UNU.RAN generator for a given target distribution. For more detailed information we refer to the UNU.RAN manual.
- How to use an independent stream of uniform random numbers for one of these UNU.RAN generators.
- We demonstrate the usage of the easy to use String API.
- We furthermore show the same example with the more flexible C API.

The String API

```
/* ------------------------------------------------------------ */
/* File: example1.c                                             */
/* ------------------------------------------------------------ */

/* Include UNURAN header files.                                 */
#include <unuran.h>
#include <unuran_urng_rngstreams.h>

/* ------------------------------------------------------------ */

int main(void)
{
  int    i;               /* loop variable                     */
  double x; int k;        /* will hold the random number       */

  /* Declare UNU.RAN objects.                                   */
  UNUR_GEN  *gen1, *gen2, *gen3; /* generator objects           */

  /* Declare objects for uniform random number generators.      */
  UNUR_URNG  *urng2;      /* uniform RN generator object       */

  /* -- Optional: Set seed for RNGSTREAMS library ------------- */

  /* The RNGSTREAMS library sets a package seed.                */
  unsigned long seed[] = {111u, 222u, 333u, 444u, 555u, 666u};
  RngStream_SetPackageSeed(seed);

  /* -- Example 1 --------------------------------------------- */
  /* Beta distribution with shape parameters 2 and 3.           */
  /* Use method 'AUTO' (AUTOmatic).                             */

  /* Create generator object.                                   */
  gen1 = unur_str2gen("beta(2,3)");

  /* It is important to check if the creation of the generator  */
  /* object was successful. Otherwise 'gen1' is the NULL pointer */
  /* and would cause a segmentation fault if used for sampling. */
  if (gen1 == NULL) {
     fprintf(stderr, "ERROR: cannot create generator object\n");
     exit (EXIT_FAILURE);
  }

  /* Now you can use the generator object 'gen1' to sample from */
  /* the target distribution. Eg.:                              */
  for (i=0; i<10; i++) {
    x = unur_sample_cont(gen1);
    printf("%f\n",x);
  }
```

```
  /* -- Example 2 ------------------------------------------- */
  /* Student's t distribution with 3 degrees of freedom.      */
  /* Use method 'TDR' (Transformed Density Rejection) with    */
  /* "immediate acception"                                    */
  gen2 = unur_str2gen("student(3) & method=TDR; variant_ia");
  if (gen2 == NULL) exit (EXIT_FAILURE);

  /* However, this time we use a (new) independent stream of  */
  /* uniformrandom numbers.                                   */
  urng2 = unur_urng_rngstream_new("urng2");
  unur_chg_urng( gen2, urng2 );

  /* Draw a sample.                                           */
  for (i=0; i<10; i++) {
    x = unur_sample_cont(gen2); printf("%f\n",x);
  }

  /* -- Example 3 ------------------------------------------- */
  /* Discrete distribution with given probability vector.     */
  /* Use method 'DGT' (Discrete Guide Table method).          */
  gen3 = unur_str2gen("discr; pv=(0.5,1.5,1.0,0.3) & method=DGT");
  if (gen3 == NULL) exit (EXIT_FAILURE);

  /* we use the default URNG again. So there is nothing to do.   */

  /* Draw a sample. Notice that we get integers!              */
  for (i=0; i<10; i++) {
    k = unur_sample_discr(gen3); printf("%d\n",k);
  }

  /* -- Call destructor ------------------------------------- */
  /* When generators are not needed any they can be destroyed.   */

  unur_free(gen1);
  unur_free(gen2);  unur_urng_free(urng2);
  unur_free(gen3);

  exit (EXIT_SUCCESS);
} /* end of main() */

/* ------------------------------------------------------------ */
```

The C API

```
/* ------------------------------------------------------------ */
/* File: example2.c                                             */
/* ------------------------------------------------------------ */

/* Include UNURAN header files.                                 */
#include <unuran.h>
#include <unuran_urng_rngstreams.h>

/* ------------------------------------------------------------ */

int main(void)
{
  int    i;           /* loop variable                        */
  double x; int k;    /* will hold the random number          */

  /* Declare UNU.RAN objects.                                  */
  UNUR_DISTR *distr;             /* distribution object       */
  UNUR_PAR   *par;               /* parameter object          */
  UNUR_GEN  *gen1, *gen2, *gen3; /* generator objects         */

  /* Declare objects for uniform random number generators.     */
```

```
UNUR_URNG  *urng2;       /* uniform RN generator object        */

/* -- Optional: Set seed for RNGSTREAMS library -------------- */

/* The RNGSTREAMS library sets a package seed.                 */
unsigned long seed[] = {111u, 222u, 333u, 444u, 555u, 666u};
RngStream_SetPackageSeed(seed);

/* -- Example 1 ---------------------------------------------- */
/* Beta distribution with shape parameters 2 and 3.            */
/* Use method 'AUTO' (AUTOmatic).                              */

/* Create distribution object.                                 */
{
  double fparams[] = {2., 3.};
  distr = unur_distr_beta( fparams, 2 );
}

/* Choose a method: 'AUTO'.                                    */
par = unur_auto_new(distr);

/* Create the generator object.                                */
gen1 = unur_init(par);

/* It is important to check if the creation of the generator   */
/* object was successful. Otherwise 'gen1' is the NULL pointer */
/* and would cause a segmentation fault if used for sampling.  */
if (gen1 == NULL) {
   fprintf(stderr, "ERROR: cannot create generator object\n");
   exit (EXIT_FAILURE);
}

/* It is possible to reuse the distribution object to create   */
/* another generator object. If you do not need it any more,   */
/* it should be destroyed to free memory.                      */
unur_distr_free(distr);

/* Now you can use the generator object 'gen1' to sample from  */
/* the target distribution. Eg.:                               */
for (i=0; i<10; i++) {
  x = unur_sample_cont(gen1);
  printf("%f\n",x);
}

/* -- Example 2 ---------------------------------------------- */
/* Student's t distribution with 3 degrees of freedom.         */
/* Use method 'TDR' (Transformed Density Rejection) with       */
/* "immediate acception"                                       */

/* Create distribution object.                                 */
{
  double fparams[] = {3.};
  distr = unur_distr_student( fparams, 1 );
}

/* Choose a method: 'TDR'.                                     */
par = unur_tdr_new(distr);
/* ... and change to immediate acceptance.                     */
unur_tdr_set_variant_ia(par);

/* However, this time we use a (new) independent stream of     */
/* uniformrandom numbers.                                      */
urng2 = unur_urng_rngstream_new("urng2");
unur_set_urng( par, urng2 );
```

```
  /* Create the generator object.                                  */
  gen2 = unur_init(par);
  if (gen2 == NULL) exit (EXIT_FAILURE);

  /* Destroy distribution object. (We do not need it any more.)  */
  unur_distr_free(distr);

  /* Draw a sample.                                                */
  for (i=0; i<10; i++) {
    x = unur_sample_cont(gen2); printf("%f\n",x);
  }

  /* -- Example 3 ------------------------------------------------ */
  /* Discrete distribution with given probability vector.        */
  /* Use method 'DGT' (Discrete Guide Table method).             */

  /* Create distribution object.                                   */
  {
    double probs[] = {0.5, 1.5, 1.0, 0.3};
    distr = unur_distr_discr_new();
    unur_distr_discr_set_pv(distr, probs, 4);
  }

  /* Choose a method: 'DGT'.                                       */
  par = unur_dgt_new(distr);

  /* we use the default URNG again. So there is nothing to do.   */

  /* Create the generator object.                                  */
  gen3 = unur_init(par);
  if (gen3 == NULL) exit (EXIT_FAILURE);

  /* Destroy distribution object. (We do not need it any more.)  */
  unur_distr_free(distr);

  /* Draw a sample. Notice that we get integers!                   */
  for (i=0; i<10; i++) {
    k = unur_sample_discr(gen3); printf("%d\n",k);
  }

  /* -- Call destructor ------------------------------------------ */
  /* When generators are not needed any they can be destroyed.   */

  unur_free(gen1);
  unur_free(gen2);  unur_urng_free(urng2);
  unur_free(gen3);

  exit (EXIT_SUCCESS);
} /* end of main() */

/* ------------------------------------------------------------- */
```