Readme:

1. This program is a demo of a polar sampler (online phase) in C++ language. It focuses on the rules of the polar sampler as well as speed tests. The data type of most probabilistic values is made to be double. The C++ library <random> is used to generate basic random numbers. So there is no constant-time guarantee. The offline phase is finished in MATLAB by which the data for online phase are produced and saved in .mat format.
2. File list in folder SCsampler
   1. “Bata\_BDMC\_sigmas=3\_Levelx\_ny.mat”: Battacharyya parameters computed offline in MATLAB and stored in mat file; x means the i th level of binary partition; y indicates block length N=2^y.
   2. “InpDistri\_sigmas=3.mat”: contains the transition probabilities of each level
   3. “BECconstruct.cpp”, “BECconstruct.h ”: define the high and low entropy sets; define the bit-reversal function.
   4. “decoding.cpp”: main function
   5. “polar.cpp”, “polar.h”: define polar sampler class
   6. “use.h”, “use.cpp”: define other functions
3. decoding.cpp
   1. decoding.cpp includes the main function;
   2. this program demonstrates how to use polar sampler to produce discrete Gaussian samples with 0 centre and Gaussian width s=3\*sqrt(2\*pi);
   3. this distribution is generated from a domain of 64 integer points, and the target discrete probability density function table is as follows.

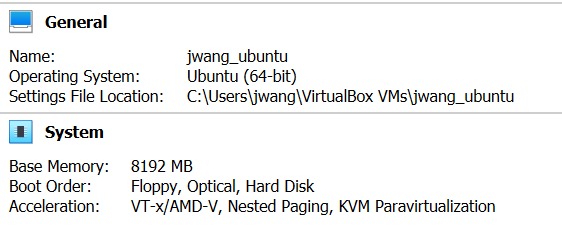
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0~20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| Prob. | ≈0 | 5.14e-4 | 1.5e-3 | 3.8e-3 | 8.7e-3 | 1.8e-2 | 3.32e-2 | 5.47e-2 | 8.07e-2 | 0.1065 |
| Index | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 |
| Prob. | 0.1258 | 0.1330 | 0.1258 | 0.1065 | 8.07e-2 | 5.47e-2 | 3.32e-2 | 1.8e-2 | 8.7e-3 | 3.8e-3 |
| index | 40 | 41 | 42~63 |  |  |  |  |  |  |  |
| Prob. | 1.5e-3 | 5.14e-4 | ≈0 |  |  |  |  |  |  |  |

* 1. parameters:
     1. Blk: block length N; it can be 1024, 8192, 16384, 32768. (changeable)
     2. Lev: the number of levels; there are 6 levels in this demo. (fixed)
     3. width: the number of valid integers after tail cut; it is 64 in this demo. (fixed)
     4. beta: between 0 and 0.5; large beta gives better accuracy. (changeable)
     5. sigma: a Gaussian parameter; is 3 in this demo. (fixed)
     6. numiter: the number of repetitions to invoke a polar sampler. (changeable)
     7. outputs:

If the following two code blocks are not disabled, you can see the statistics of discrete Gaussian samples.

|  |  |
| --- | --- |
| /\*\*\*\*\*\*statistics output\*\*\*\*\*\*\*\*\*\*/  for (UINT i = 0; i < Blk; i++)  {  histArray[preX[i]] ++;  }  cout << endl; | /\*\*\*\*\*\*statistics output \*\*\*\*\*\*\*\*\*\*/  cout << endl;  cout << "Histogram value at index..." << endl;  for (UINT i = 0; i < pow(2,Lev); i++)  {  cout << i << ":" << histArray[i] / (Blk \*numiter)<< endl;  } |

1. examples
   1. Environment：intel [i7-8550U@1.8GHz](mailto:i7-8550U@1.8GHz), VirtualBox 6.0, ubuntu 64bit，Base Memory 8G.



* 1. Commands:

command 1 : $export LD\_LIBRARY\_PATH=/usr/local/MATLAB/R2016b/bin/glnxa64

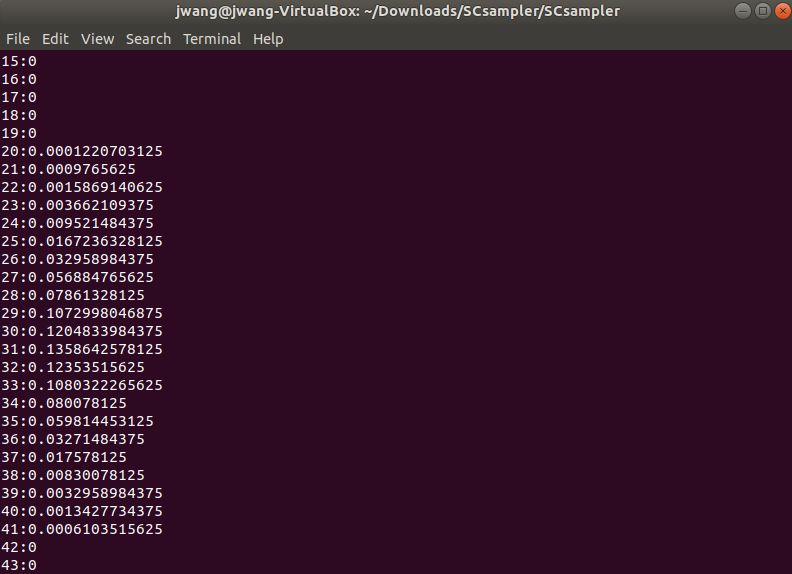
command 2 : $g++ -Ofast decoding.cpp BECconstruct.cpp polar.cpp use.cpp -o SCsampler -I/usr/local/MATLAB/R2016b/extern/include -L/usr/local/MATLAB/R2016b/bin/glnxa64 -lmat -leng -lmx -lmex

command 3 : $./SCsampler

To build and run the codes, you are suggested to install MATLAB (any version after MATLAB2016 is supposed to work correctly). Please find the folder *glnxa64* and the folder *include* in the installation directory and use command 1 and 2 to build. Please make sure the path in command 1 and 2 are correct according to your installation directory.

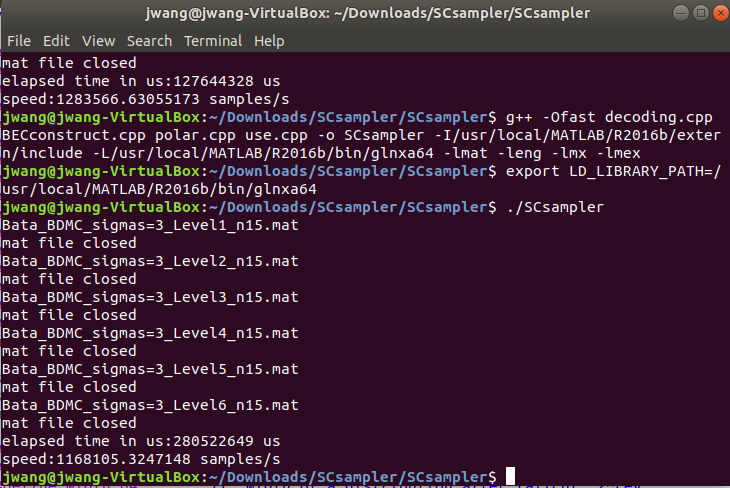
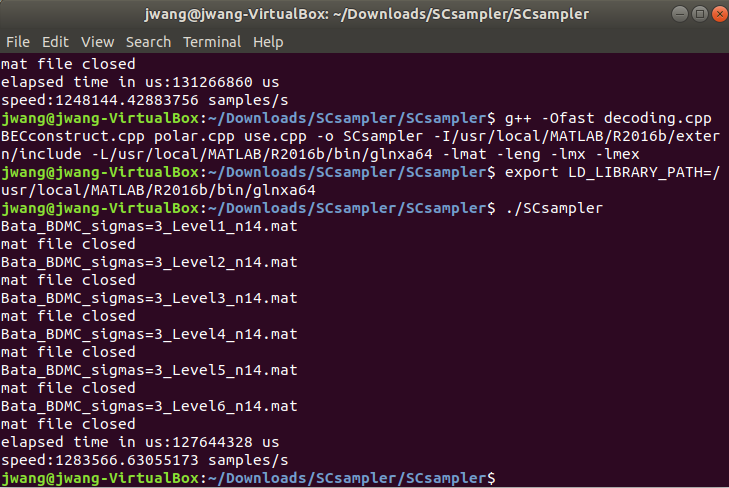
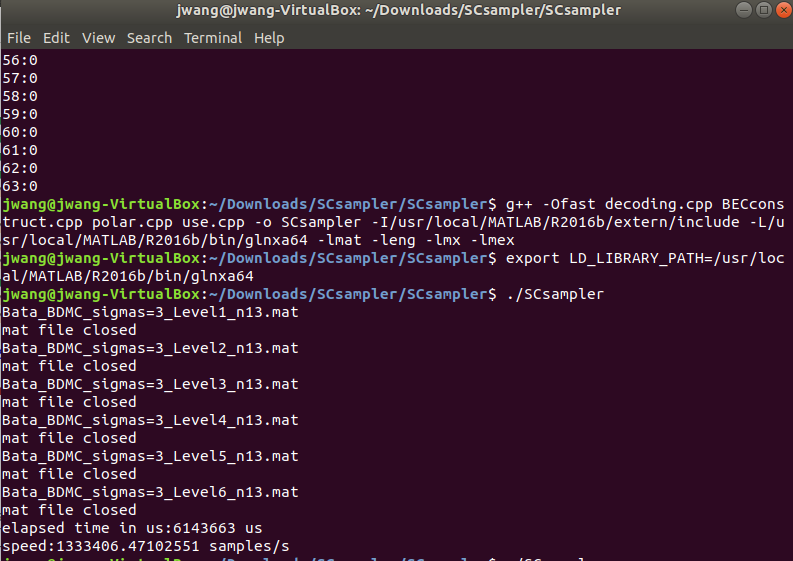
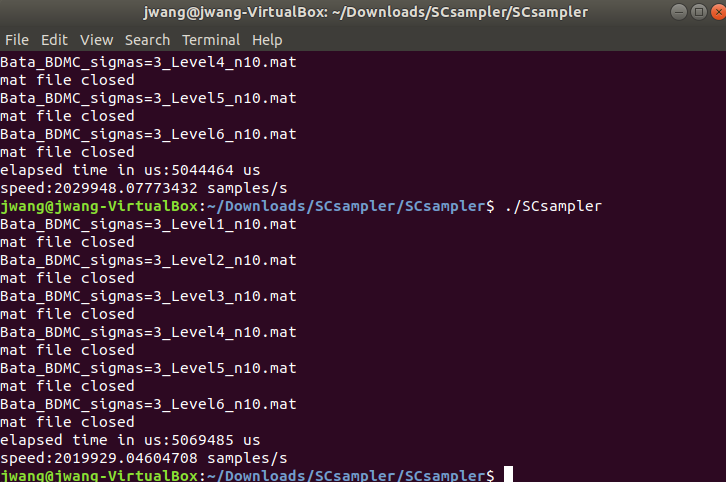
The reason we use the library of MATLAB is that the calculations in offline phase are done in MATLAB and the data including Bhattacharyya parameters and transition probabilities are save in the MATLAB data format. To read these data by C++, you need to install the library of MATLAB.

* 1. Statistic output：(Settings: Blk=8192, Lev=6, width=64, beta=0.48, sigma=3, numiter=1000)



* 1. Speed output：(Settings: Lev=6, width=64, sigma=3, numiter=1000)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lev | Sigma  =s/sqrt(2pi) | Polar sampler(samples/second) | | | |
| 6 | 3 | Blk=1024,  beta=0.48 | Blk=8192,  beta=0.48 | Blk=16384,  beta=0.45 | Blk=32768,  Beta=0.42 |
| 2.020E6 | 1.333E6 | 1.283E6 | 1.168E6 |



1. A full benchmark test of speed

This table shows the results of benchmark test of sampling speed which is conducted on a PC with a CPU [i9-9900K@3.6GHz](mailto:i9-9900K@3.6GHz), 32G memory and Ubuntu 64bit OS.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Lev | Sigma  =s/sqrt(2pi) | Polar sampler(samples/second) | | | |
| Blk=1024 | Blk=8192 | Blk=16384 | Blk=32768 |
| 6 | 3 | beta=0.487  2.459E6/s | beta=0.487  1.798E6/s | beta=0.4535  1.589E6/s | beta=0.4244  1.409E6/s |
| 7 | 8 | beta=0.4876  2.127E6 | beta=0.4876  1.578E6/s | beta=0.454  1.396E6/s | beta=0.425  1.233E6/s |
| 9 | 32 | beta=0.488  1.750E6 | beta=0.488  1.229E6/s | beta=0.4544  1.078E6/s | beta=0.4252  0.960E6/s |
| 12 | 256 | beta=0.4885  1.336E6 | beta=0.4885  0.896E6/s | beta=0.455  0.794E6/s | beta=0.4257  0.734E6/s |

Remark 1: We make beta to be the smallest value such that the security level is at least 64 for every choice of block length. But the third column is an exception where Blk is 1024 and it cannot meet the security level 128.

Remark 2: Normally, the larger the block length is, the smaller beta is required. A large Blk can make the polarization effect relatively deep so that a small beta suffices to achieve the target approximation error.

Remark 3. A smaller beta contributes to the increase of sampling speed because it makes the fraction of polarized sets larger. However, the block length Blk dominates the overall speed.