Clear all symbols from previous evaluations to avoid problems

In[ ]:=
```
Clear["Global`*"]
```

Generating random variables with a specified distribution

Special methods for the normal distribution

The normal distribution $N(\mu, \sigma^2)$

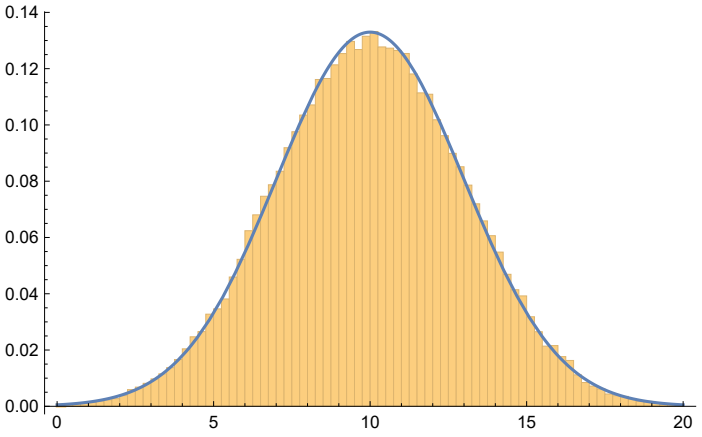
$$\rho(x) = \frac{1}{\sigma \sqrt{2\,\pi}} \exp\left(-(x-\mu)^2 / 2\,\sigma^2\right) \tag{1}$$

can be efficiently generated for example using the central limit theorem. If we sum up 12 uniformly distributed ($\mu = 1/2,\ \sigma^2 = 1/12$) random variables we get $N(6, 1)$. Thus by shifting to 0 and then by multiplying by σ and adding μ we get the general normal distribution:

In[ ]:=
```
myNormalSums[n_, par_] := Module[{out = ConstantArray[0.0, {n}]},
   Do[
    out[[i]] = 0.0;
    Do[
     out[[i]] = out[[i]] + RandomReal[],
     {j, 1, 12}];
    out[[i]] = par[[1]] + par[[2]] (out[[i]] - 6.0),
    {i, 1, n}
   ];
   Return[out]
  ];
```

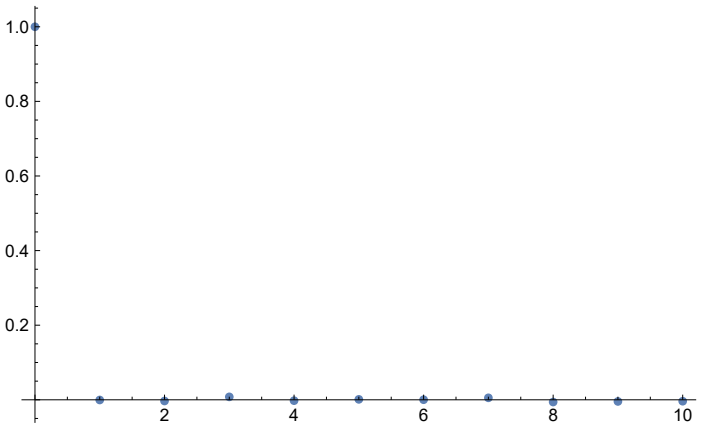
In[ ]:=
```
μ = 10; σ = 3;
n = 10^5;
points = myNormalSums[n, {μ, σ}];
Show[Histogram[points, {0, 20, 1 / 4}, "ProbabilityDensity"],
 Plot[PDF[NormalDistribution[μ, σ], x], {x, 0, 20}]]
expectedValue = Total[points] / n ;
variance = Total[points^2] / n - expectedValue^2 ;
Print["Expected value and standard deviation should be: ", N[μ], ", ", N[σ]];
Print["Expected value and standard deviation obtained:  ",
  expectedValue, ", ", Sqrt[variance]];
kmax = 10; corr = ConstantArray[1.0, kmax + 1];
Do[
  corr[[k + 1]] =
   {k, (points[[1 ;; n - k]].points[[k + 1 ;; n]] / (n - k) - expectedValue^2) / variance},
  {k, 0, kmax}
 ];
ListPlot[corr, PlotRange → All]
```

Out[ ]=




```
Expected value and standard deviation should be: 10., 3.
Expected value and standard deviation obtained:  10.0014, 3.00507
```

Out[ ]=



Another efficient way of generating the normal distribution is to consider a Gaussian distribution in two dimensions and use the polar coordinates to transform it to a product of exponential and uniform distribution:

$$\exp\left(-\left(x_1^2+x_2^2\right)/2\right) dx_1\, dx_2 \sim \exp\left(-r^2/2\right) r\, dr\, d\theta \sim \exp(-u)\, du\, d\theta \tag{2}$$

If u is generated between 0 and ∞ with an exponential distribution using

$$y(u) = \int_0^u \exp(-t)\, dt = 1 - \exp(-u) \implies u(y) = -\ln(1-y) \tag{3}$$

and θ is uniformly distributed between 0 and 2π then

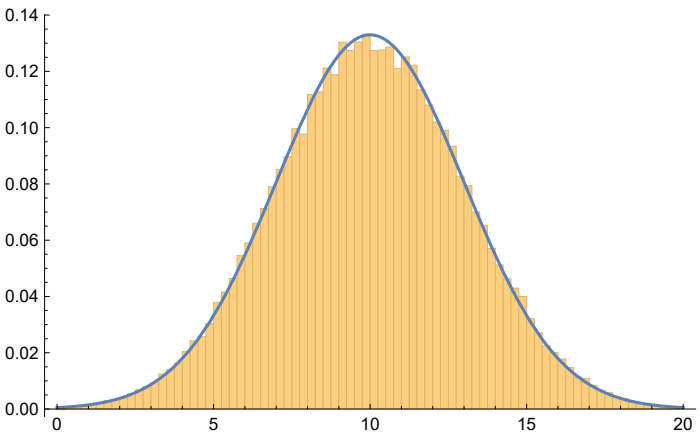
$$x_1 = \sqrt{2u}\,\cos(\theta),\ x_2 = \sqrt{2u}\,\sin(\theta) \tag{4}$$

are distributed normally.

In[ ]:=
```
myNormal2D[n_, par_] := Module[{r, θ, out = ConstantArray[0.0, {n + Mod[n, 2]}]},
   Do[
    r = Sqrt[-2 Log[1 - RandomReal[]]];
    θ = 2 π RandomReal[];
    out[[i]] = par[[1]] + par[[2]] r Cos[θ];
    out[[i + 1]] = par[[1]] + par[[2]] r Sin[θ],
    {i, 1, n + Mod[n, 2], 2}
   ];
   Return[out[[1 ;; n]]]
  ];
```

In[ ]:=
```
μ = 10; σ = 3;
n = 10^5;
points = myNormalSums[n, {μ, σ}];
Show[Histogram[points, {0, 20, 1 / 4}, "ProbabilityDensity"],
 Plot[PDF[NormalDistribution[μ, σ], x], {x, 0, 20}]]
expectedValue = Total[points] / n ;
variance = Total[points^2] / n - expectedValue^2 ;
Print["Expected value and standard deviation should be: ", N[μ], ", ", N[σ]];
Print["Expected value and standard deviation obtained:  ",
  expectedValue, ", ", Sqrt[variance]];
kmax = 10; corr = ConstantArray[1.0, kmax + 1];
Do[
  corr[[k + 1]] =
   {k, (points[[1 ;; n - k]].points[[k + 1 ;; n]] / (n - k) - expectedValue^2) / variance},
  {k, 0, kmax}
 ];
ListPlot[corr, PlotRange → All]
```

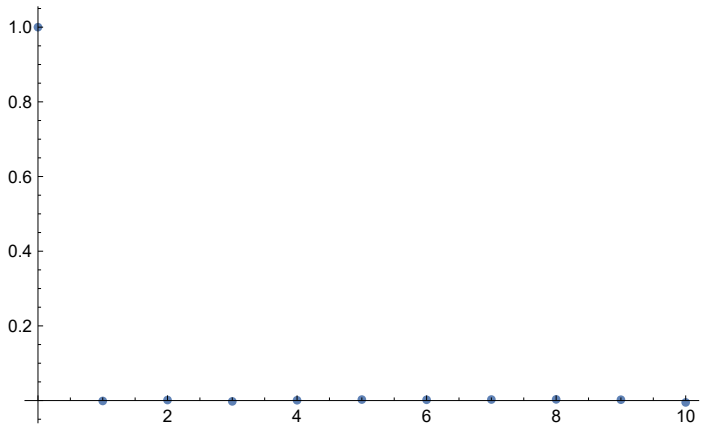
Out[ ]=



```
Expected value and standard deviation should be: 10., 3.

Expected value and standard deviation obtained:  10.0064, 3.01374
```

Out[ ]=



von Neumann method

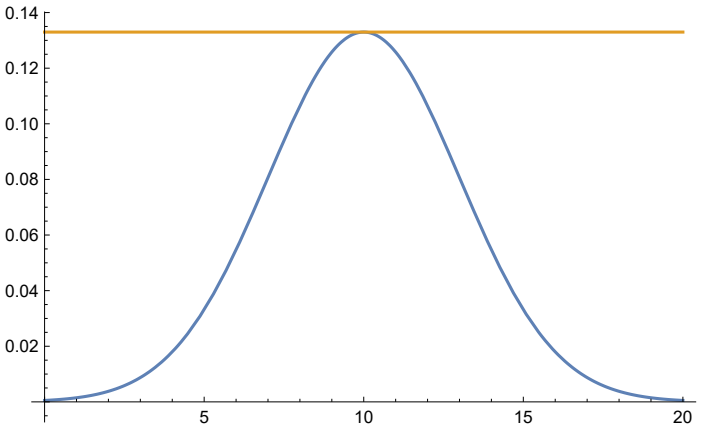
In[ ]:=

```
myRandomNeumann[ρ_, a_, b_, R_, n_] := Module[{g, x, y, rejected = 0, out},
  out = ConstantArray[0.0, n];
  Do[
   g = 1;
   While[g == 1,
    x = a + RandomReal[] (b - a);
    y = R RandomReal[];
    If[y < ρ[x], g = 0, rejected++]
   ];
   out[[i]] = x,
   {i, 1, n}
  ];
  Print[rejected, " rejected points to generate ", n, " points."];
  Return[out]
 ];
```

Normal distribution generated by von Neumann method:

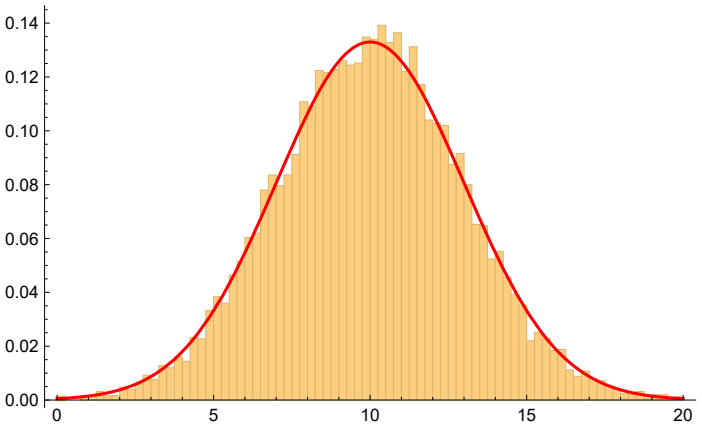
In[ ]:=
```
a = 0; b = 20;
μ = 10;  σ2 = 3;
ρ := PDF[NormalDistribution[μ, σ2], #] &;
R := PDF[NormalDistribution[μ, σ2], μ];
Plot[{ρ[x], R}, {x, a, b}]
n = 10^4;
points = myRandomNeumann[ρ, a, b, R, n];
Show[Histogram[points, {a, b, 1 / 4}, "ProbabilityDensity"],
 Plot[PDF[NormalDistribution[μ, σ2], x], {x, a, b}, PlotStyle → Red]]
expectedValue = Total[points] / n ;
variance = Total[points^2] / n - expectedValue^2 ;
Print["Expected value and standard deviation should be: ", N[μ], ", ", N[σ]];
Print["Expected value and standard deviation obtained:  ",
  expectedValue, ", ", Sqrt[variance]];
kmax = 10;  corr = ConstantArray[1.0, kmax + 1];
Do[
  corr[[k + 1]] =
   {k, (points[[1 ;; n - k]].points[[k + 1 ;; n]] / (n - k) - expectedValue^2) / variance},
  {k, 0, kmax}
 ];
ListPlot[corr, PlotRange → All]
```

Out[ ]=



16 580 rejected points to generate 10 000 points.

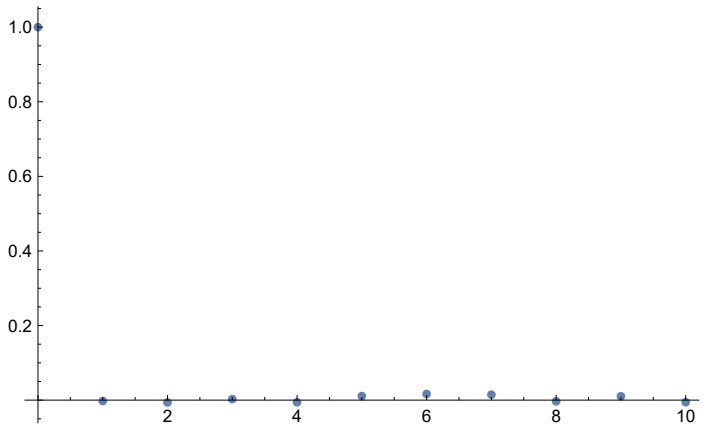
Out[ ]=



```
Expected value and standard deviation should be: 10., 3.

Expected value and standard deviation obtained:  10.0068, 2.9505
```

Out[ ]=

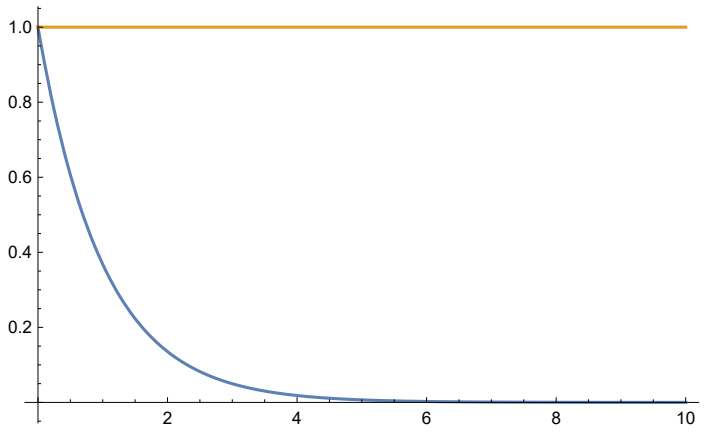


Exponential distribution generated by von Neumann method (not very efficient):

In[ ]:=

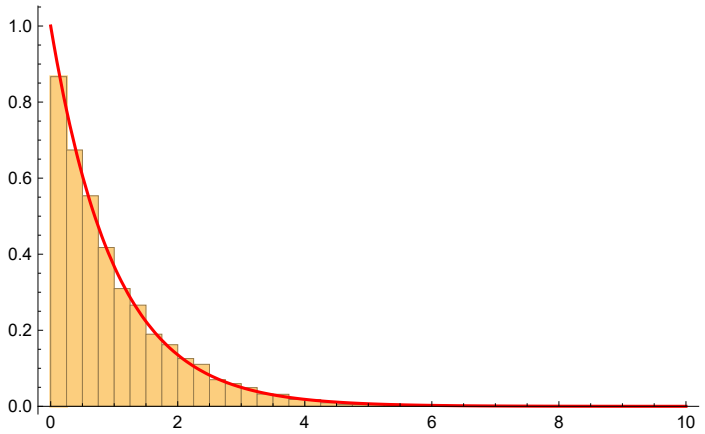
```
a = 0; b = 10;
ρ := Exp[-#] &;
R := 1;
Plot[{ρ[x], R}, {x, a, b}]
n = 10^4;
points = myRandomNeumann[ρ, a, b, R, n];
Show[Histogram[points, {a, b, 1 / 4}, "ProbabilityDensity"],
 Plot[Exp[-x], {x, a, b}, PlotRange → {0, 1}, PlotStyle → Red]]
expectedValue = Total[points] / n ;
variance = Total[points^2] / n - expectedValue^2 ;
Print["Expected value and standard deviation obtained:  ",
  expectedValue, ", ", Sqrt[variance]];
kmax = 10; corr = ConstantArray[1.0, kmax + 1];
Do[
  corr[[k + 1]] =
   {k, (points[[1 ;; n - k]].points[[k + 1 ;; n]] / (n - k) - expectedValue^2) / variance},
  {k, 0, kmax}
 ];
ListPlot[corr, PlotRange → All]
```

Out[ ]=



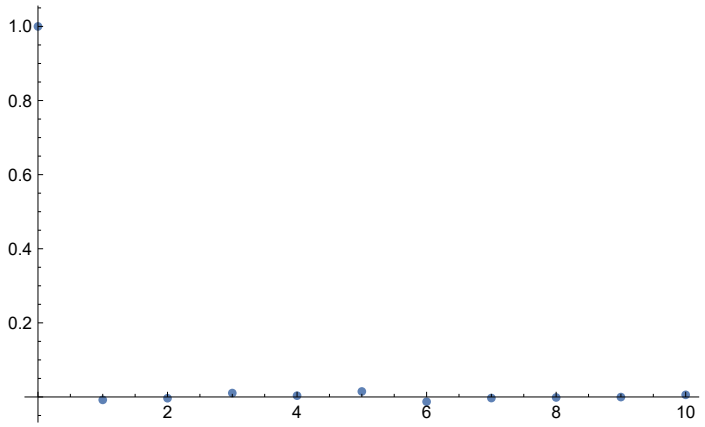

```
89669 rejected points to generate 10000 points.
```

Out[ ]=




```
Expected value and standard deviation obtained:  1.00744, 0.989543
```

Out[ ]=



Metropolis algorithm

One dimensional case, ρ must be a function of x:

In[ ]:=

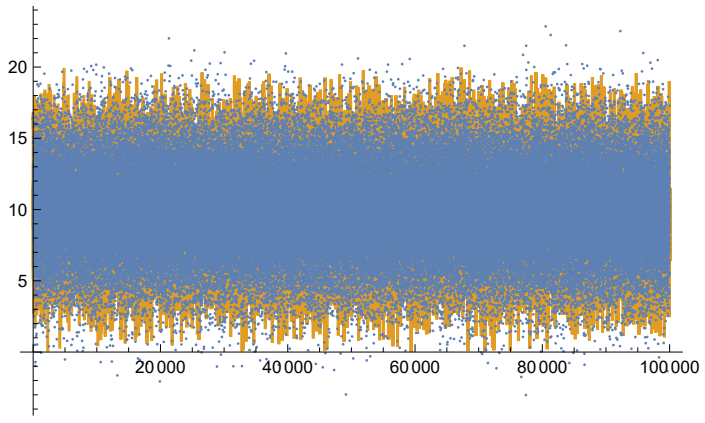
```
myRandomMetropolis1D[n_, ρ_, a_, b_, x0_, δ_] :=
  Module[{x = x0, xtrial, ratio, ρx, out = ConstantArray[0.0, n], accepted = 0},
   ρx = ρ[x];
   Do[
    (* try a new random step *)
    xtrial = x + δ RandomReal[{-1, 1}];
    (* check if we are still in the given region, if not, adjust *)
    If[xtrial < a, xtrial = a + δ RandomReal[]];
    If[xtrial > b, xtrial = b - δ RandomReal[]];
    ratio = ρ[xtrial] / ρx;
    (* decide whether to make this step according to the distribution *)
    If[ratio ⩾ 1 || ratio > RandomReal[], x = xtrial;
     ρx = ρx ratio; accepted++];
    out[[i]] = x,
    {i, 1, n}
   ];
   Print["Number of accepted trials: ", accepted, " from ", n];
   Return[out]
  ];
```

Applied to the normal distribution:

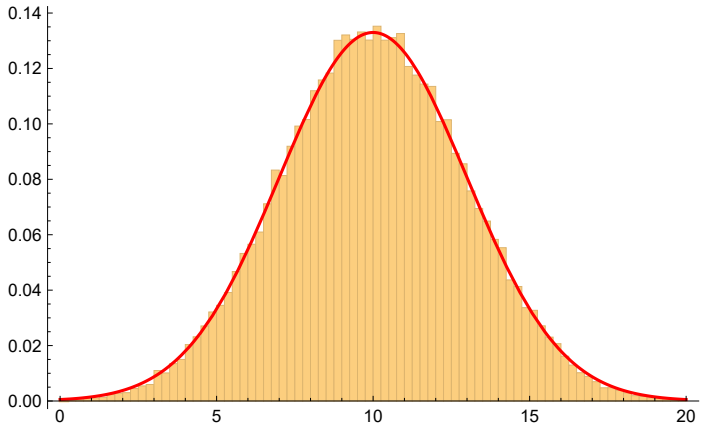
In[ ]:=
```
n = 10^5;
a = 0; b = 20;
μ = 10; σ = 3;
ρ := PDF[NormalDistribution[μ, σ], #] &;
x0 = 10.0; step = 5.0;
points = myRandomMetropolis1D[n, ρ, a, b, x0, step];
data = RandomReal[NormalDistribution[μ, σ], {n}];
ListPlot[{data, points}, Joined → {False, True}]
Show[Histogram[points, {a, b, 1 / 4}, "ProbabilityDensity"],
 Plot[PDF[NormalDistribution[μ, σ], x], {x, a, b}, PlotStyle → Red]]
expectedValue = Total[points] / n ;
variance = Total[points^2] / n - expectedValue^2 ;
Print["Expected value and standard deviation should be: ", N[μ], ", ", N[σ]];
Print["Expected value and standard deviation obtained:  ",
  expectedValue, ", ", Sqrt[variance]];
kmax = 10; corr = ConstantArray[1.0, kmax + 1];
Do[
  corr[[k + 1]] =
   {k, (points[[1 ;; n - k]].points[[k + 1 ;; n]] / (n - k) - expectedValue^2) / variance},
  {k, 0, kmax}
 ];
ListPlot[corr, PlotRange → All]
```

```
Number of accepted trials: 69 211 from 100 000
```

Out[ ]=



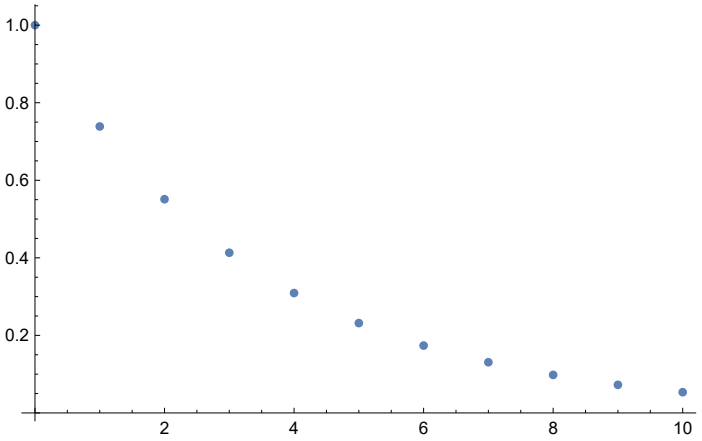
Out[ ]=



```
Expected value and standard deviation should be: 10., 3.

Expected value and standard deviation obtained:  9.99835, 2.93575
```

Out[ ]=

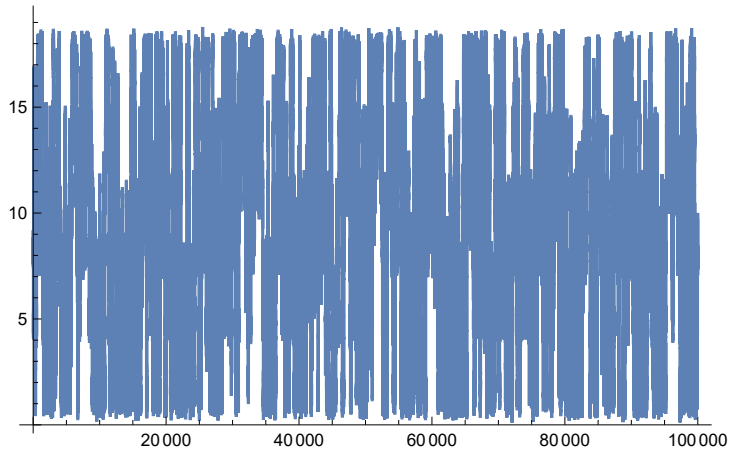


Applied to the distribution with several maxima (the choice of δ plays an important role in whether we get reasonable results):

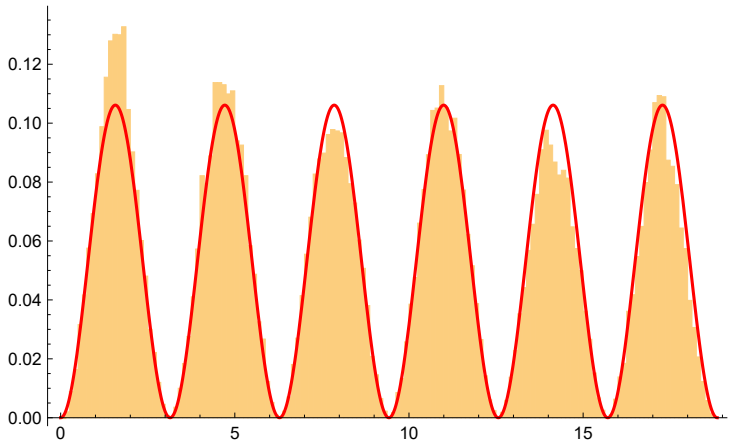
In[ ]:=
```
n = 100 000;
a = 0; b = 6 π;
ρ := 2 Sin[#]^2 / b + 0.00001 &;
x0 = (a + b) / 2; step = (b - a) / 10;
points = myRandomMetropolis1D[n, ρ, a, b, x0, step];
ListPlot[{points}, Joined → {True}]
Show[Histogram[points, {a, b, 1 / 8}, "ProbabilityDensity"],
 Plot[ρ[x], {x, a, b}, PlotStyle → Red]]
```

```
Number of accepted trials: 56 931 from 100 000
```

Out[ ]=



Out[ ]=



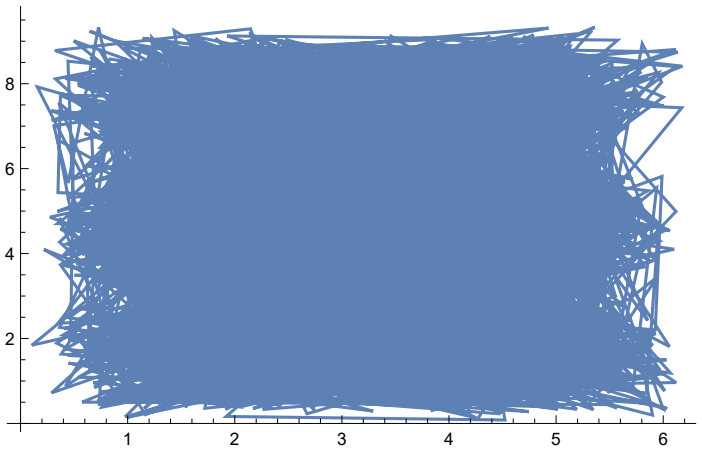
Metropolis algorithm for *n*-dimensional case:

In[ ]:=
```
myRandomMetropolisND[n_, dim_, ρ_, a_, b_, x0_, δ_] :=
  Module[{x = x0, xtrial, ratio, ρx, out = ConstantArray[0.0, {n, dim}], accepted = 0},
   ρx = ρ[x];
   Do[
    (* try a new random step *)
    xtrial = x + δ RandomReal[{-1, 1}, dim];
    (* check if we are still in the given region, if not, adjust *)
    Do[
     If[xtrial[[d]] < a[[d]], xtrial[[d]] = a[[d]] + δ[[d]] RandomReal[]];
     If[xtrial[[d]] > b[[d]], xtrial[[d]] = b[[d]] - δ[[d]] RandomReal[]],
     {d, 1, dim}
    ];
    (* decide whether to make this step according to the distribution *)
    ratio = ρ[xtrial] / ρx;
    If[ratio ⩾ 1 || ratio > RandomReal[], x = xtrial;
     ρx = ρx * ratio, accepted++];
    out[[i]] = x,
    {i, 1, n}
   ];
   Print["Number of accepted trials: ", accepted, " from ", n];
   Return[out]
  ];
```

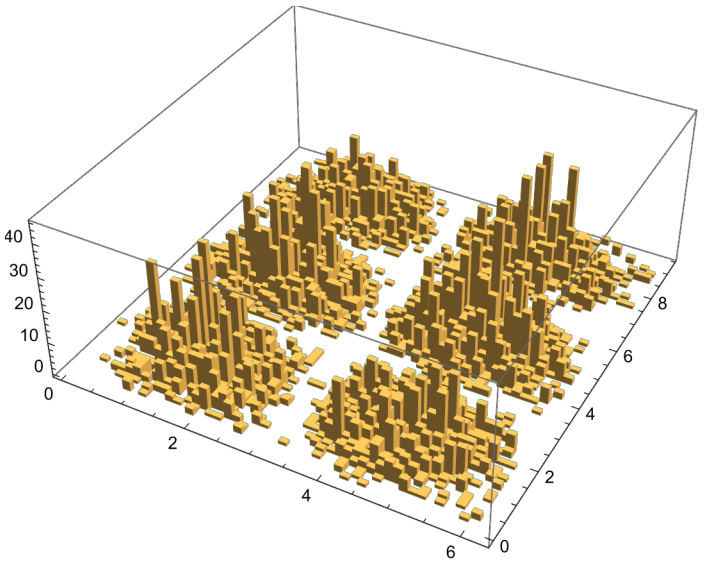
In[ ]:=
```
n = 10 000;
a = {0, 0}; b = {2 π, 3 π};
(* ρ:=2Sin[#1]^2Sin[#2]^2/Times@@b &; *)
ρ := 2 Sin[#[[1]]]^2 Sin[#[[2]]]^2 / (Times @@ b) + 0.00001 &;
x0 = {3, 4}; δ = {5, 5};
points = myRandomMetropolisND[n, 2, ρ, a, b, x0, δ];
ListPlot[{points}, Joined → {True}, PlotRange → All]
Histogram3D[points, {{a[[1]], b[[1]], 1 / 8}, {a[[2]], b[[2]], 1 / 8}}]
```

```
Number of accepted trials: 5857 from 10 000
```

Out[ ]=

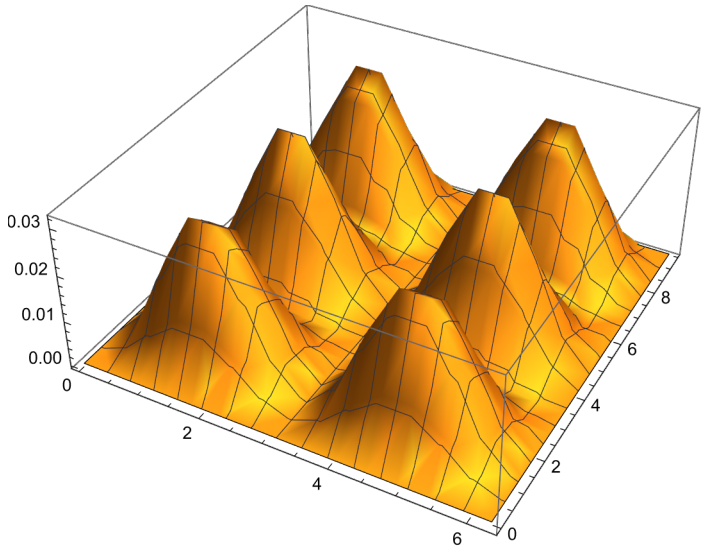


Out[ ]=



In[ ]:=
```
Plot3D[ρ[{x, y}], {x, a[[1]], b[[1]]}, {y, a[[2]], b[[2]]}, PlotRange → All]
```

Out[ ]=



Barker algorithm

One dimensional case, ρ must be a function of x:

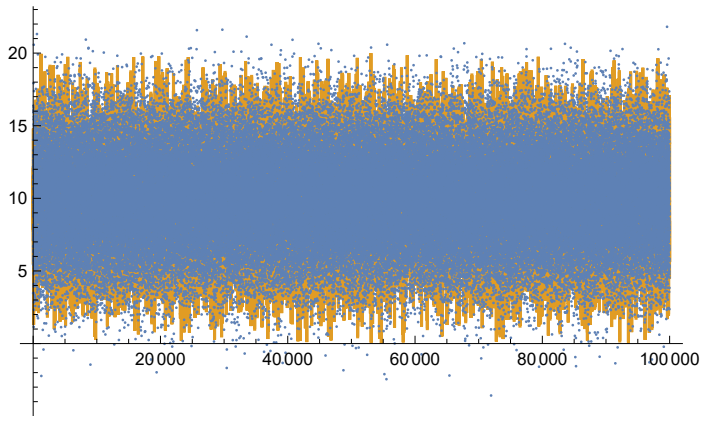
In[ ]:=
```
myRandomBarker1D[n_, ρ_, a_, b_, x0_, δ_] :=
  Module[{x = x0, xtrial, ratio, ρx, ρtrial, out = ConstantArray[0.0, n], accepted = 0},
   ρx = ρ[x];
   Do[
    (* try a new random step *)
    xtrial = x + δ RandomReal[{-1, 1}];
    (* check if we are still in the given region, if not, adjust *)
    If[xtrial < a, xtrial = a + δ RandomReal[]];
    If[xtrial > b, xtrial = b - δ RandomReal[]];
    ρtrial = ρ[xtrial];
    ratio = ρtrial / (ρx + ρtrial);
    (* decide whether to make this step according to the distribution *)
    If[ratio > RandomReal[], x = xtrial; ρx = ρtrial; accepted++];
    out[[i]] = x,
    {i, 1, n}
   ];
   Print["Number of accepted trials: ", accepted, " from ", n];
   Return[out]
  ];
```

Applied to the normal distribution:

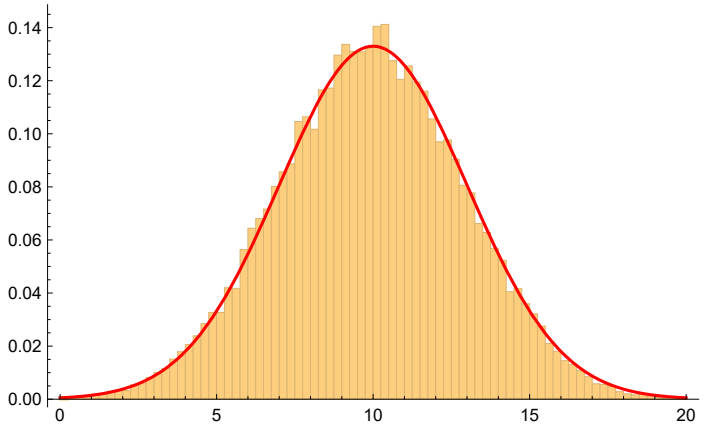
In[ ]:=
```
n = 10^5;
a = 0; b = 20;
μ = 10;  σ = 3;
ρ := PDF[NormalDistribution[μ, σ], #] &;
x0 = 10.0;  step = 5.0;
points = myRandomBarker1D[n, ρ, a, b, x0, step];
data = RandomReal[NormalDistribution[μ, σ], {n}];
ListPlot[{data, points}, Joined → {False, True}]
Show[Histogram[points, {a, b, 1 / 4}, "ProbabilityDensity"],
 Plot[PDF[NormalDistribution[μ, σ], x], {x, a, b}, PlotStyle → Red]]
expectedValue = Total[points] / n ;
variance = Total[points^2] / n - expectedValue^2 ;
Print["Expected value and standard deviation should be: ", N[μ], ", ", N[σ]];
Print["Expected value and standard deviation obtained:  ",
  expectedValue, ", ", Sqrt[variance]];
kmax = 10;  corr = ConstantArray[1.0, kmax + 1];
Do[
  corr[[k + 1]] =
   {k, (points[[1 ;; n - k]].points[[k + 1 ;; n]] / (n - k) - expectedValue^2) / variance},
  {k, 0, kmax}
 ];
ListPlot[corr, PlotRange → All]
```

```
Number of accepted trials: 41798 from 100000
```

Out[ ]=



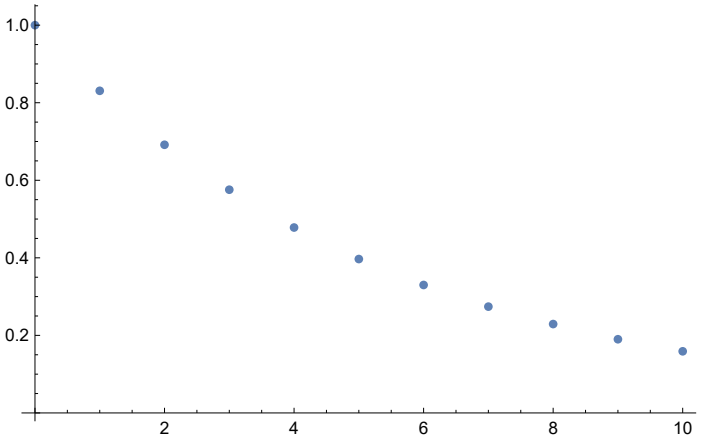
Out[ ]=



```
Expected value and standard deviation should be: 10., 3.

Expected value and standard deviation obtained:  9.93025, 2.9474
```

Out[ ]=

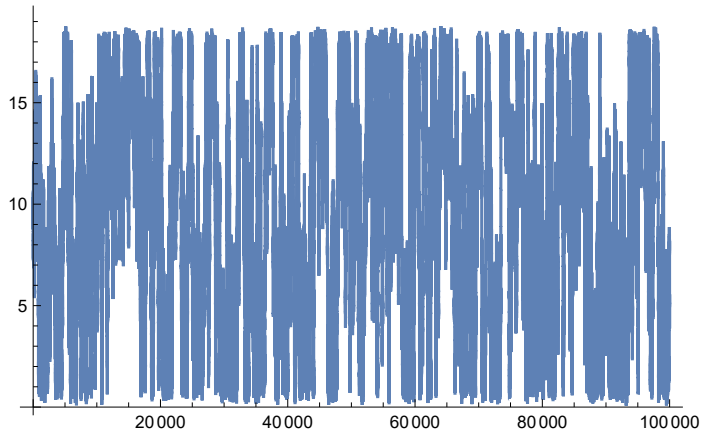


Applied to the distribution with several maxima (the choice of δ plays an important role in whether we get reasonable results):

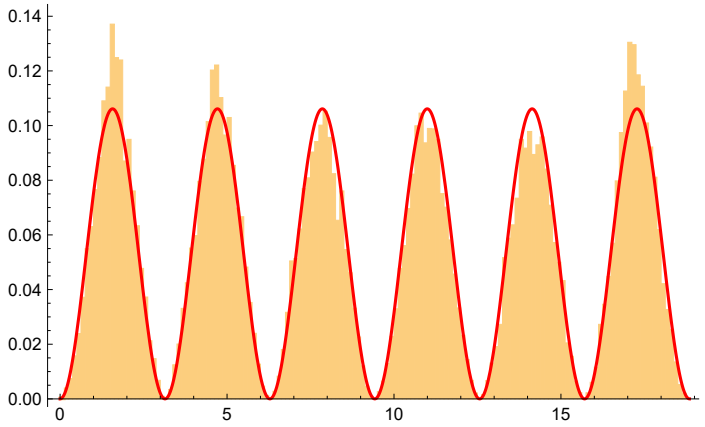
In[ ]:=
```
n = 100 000;
a = 0; b = 6 π;
ρ := 2 Sin[#]^2 / b + 0.00001 &;
x0 = (a + b) / 2; step = (b - a) / 10;
points = myRandomBarker1D[n, ρ, a, b, x0, step];
ListPlot[{points}, Joined → {True}]
Show[Histogram[points, {a, b, 1 / 8}, "ProbabilityDensity"],
 Plot[ρ[x], {x, a, b}, PlotStyle → Red]]
```

```
Number of accepted trials: 34 933 from 100 000
```

Out[ ]=



Out[ ]=



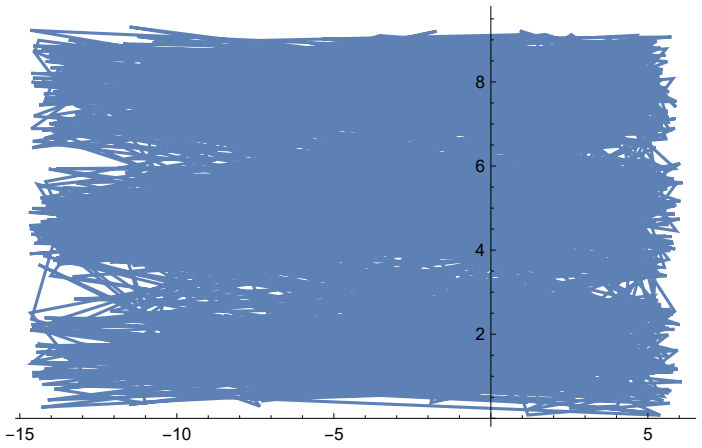
Barker algorithm for *n*-dimensional case:

In[ ]:=
```
myRandomBarkerND[n_, dim_, ρ_, a_, b_, x0_, δ_] := Module[
    {x = x0, xtrial, ratio, ρx, ρtrial, out = ConstantArray[0.0, {n, dim}], accepted = 0},
    ρx = ρ[x];
    Do[
     (* try a new random step *)
     xtrial = x + δ RandomReal[{-1, 1}, dim];
     (* check if we are still in the given region, if not, adjust *)
     Do[
      If[xtrial[[d]] < a[[d]], xtrial[[d]] = a[[d]] + δ[[d]] RandomReal[]];
      If[xtrial[[d]] > b[[d]], xtrial[[d]] = b[[d]] - δ[[d]] RandomReal[]],
      {d, 1, dim}
     ];
     (* decide whether to make this step according to the distribution *)
     ρtrial = ρ[xtrial];
     ratio = ρtrial / (ρx + ρtrial);
     If[ratio > RandomReal[], x = xtrial; ρx = ρtrial, accepted++];
     out[[i]] = x,
     {i, 1, n}
    ];
    Print["Number of accepted trials: ", accepted, " from ", n];
    Return[out]
   ];
```

In[ ]:=
```
n = 10000;
a = {0, 0}; b = {2 π, 3 π};
(* ρ:=2Sin[#1]^2Sin[#2]^2/Times@@b &; *)
ρ := 2 Sin[#[[1]]]^2 Sin[#[[2]]]^2 / (Times @@ b) + 0.00001 &;
x0 = {3, 4}; δ = {21, 2};
points = myRandomBarkerND[n, 2, ρ, a, b, x0, δ];
ListPlot[{points}, Joined → {True}, PlotRange → All]
Histogram3D[points, {{a[[1]], b[[1]], 1 / 8}, {a[[2]], b[[2]], 1 / 8}}]
```

```
Number of accepted trials: 7464 from 10000
```

Out[ ]=



Out[ ]=