Diffusion quantum Monte Carlo method

Clear all symbols from previous evaluations to avoid problems

```
Clear["Global`*"]
```

1D harmonic oscillator

Starting wave function, local energy, diffusion coefficient

In[19]:=
```
ϕ0[x_, α_] := Exp[-α * x * x];
exactEnergy[α_] =
  Assuming[α > 0, Integrate[ϕ0[x, α] * (-D[ϕ0[x, α], x, x] + x^2 ϕ0[x, α]) / 2,
     {x, -Infinity, Infinity}] / Integrate[ϕ0[x, α]^2, {x, -Infinity, Infinity}]];
locEn[x_, α_] = Simplify[1 / 2 (-D[ϕ0[x, α], x, x] + x^2 ϕ0[x, α]) / ϕ0[x, α]];
Dif[x_, α_] = D[ϕ0[x, α], x] / ϕ0[x, α];
Print["ϕ0(x,α) = ", ϕ0[x, α], ", ϵ(x) = ", locEn[x, α], ", D(x) = ", Dif[x, α]]
```

$\phi 0(x,\alpha) = e^{-x^2 \alpha}$, $\epsilon(x) = \alpha + x^2 \left(\frac{1}{2} - 2\alpha^2\right)$, $D(x) = -2 x \alpha$

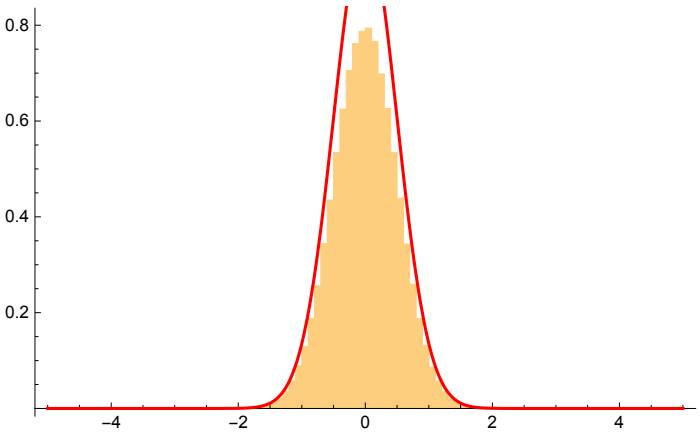
Get initial n points with distribution given by |ϕ0(x)|^2

In[238]:=
```
αini = 1.0; (* exact for α = 1/2 *)
nini = 1000000;
a = -5.0; b = 5.0;
ρ[x_] := ϕ0[x, αini]^2;
x0 = 0.0; δ = 4.0;
(* use in Matropolis each 10th point of random walk to avoid correlation*)
points = MyRandomMetropolis1D[nini, ρ, a, b, x0, δ, 10];
ϵ[x_] := locEn[x, αini];
energy = MyMCIntegration[points, ϵ];
Print["Alpha = ", αini, ", mean energy = ",
 energy[[1]], ", exact energy = ", exactEnergy[αini]]
```

```
Alpha = 1., mean energy = 0.6255774056, exact energy = 0.625
```

In[264]:=
```
Histogram[points[[1 ;; nini]], {a, b, (b - a) / 100}, "PDF",
 Epilog → First@Plot[ρ[x], {x, a, b}, PlotRange → All, PlotStyle → Red]]
```

Out[264]=



Time evolution of random points:

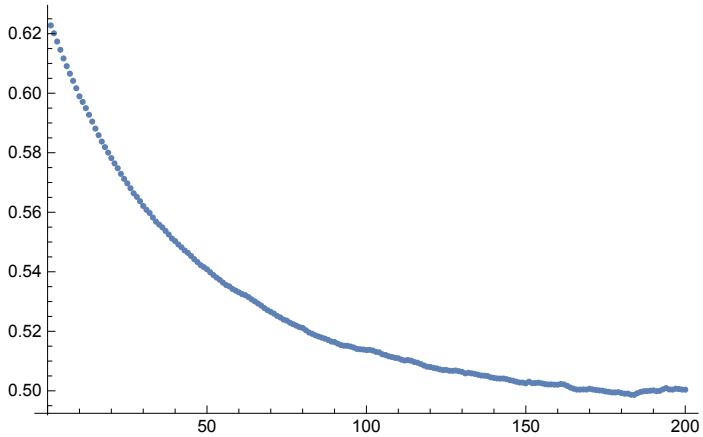
```
nt = 200; (* number of iterations *)
dt = 0.01; (* time step *)
En = energy[[1]];

xx = ConstantArray[0.0, 2 * nini];
xx[[1 ;; nini]] = points;
nn = nini;
meanEnergy = ConstantArray[0.0, nt];
Do[
  j = 1;
  While[j ≤ nn,
   xx[[j]] = xx[[j]] + Dif[xx[[j]], αini] * dt + MyRandomNormal[{0.0, Sqrt[dt]}];
   expEn = Exp[- (locEn[xx[[j]], αini] - En) * dt];
   If[
    expEn < 1.0,
    (* then *)
    If[expEn < RandomReal[], (* delete the point (replace with the last one) *)
     xx[[j]] = xx[[nn]];
     nn--;
     j--
    ],
    (* else *)
    If[expEn - 1.0 > RandomReal[], (* add another point *)
     xx[[nn + 1]] = xx[[j]];
     nn++
    ]
   ];
   j++
  ];
  {En, error} = MyMCIntegration[xx[[1 ;; nn]], ϵ];
  meanEnergy[[i]] = En;
  If[Mod[i, 10] == 0,
   Print["Iteration ", i, ", En = ", En, ", number of points: ", nn]],
  {i, 1, nt}
 ];
```

```
Iteration 10, En = 0.598947038, number of points: 1000540
Iteration 20, En = 0.5781802486, number of points: 1000757
Iteration 30, En = 0.5621188988, number of points: 1001035
Iteration 40, En = 0.5503177047, number of points: 1001261
Iteration 50, En = 0.5408323593, number of points: 1001155
Iteration 60, En = 0.5330903707, number of points: 1001466
Iteration 70, En = 0.5265180904, number of points: 1001962
Iteration 80, En = 0.5211336954, number of points: 1002375
Iteration 90, En = 0.5163872639, number of points: 1002570
Iteration 100, En = 0.5136911006, number of points: 1002640
Iteration 110, En = 0.5109239429, number of points: 1002825
Iteration 120, En = 0.5080083968, number of points: 1003173
Iteration 130, En = 0.5063741592, number of points: 1002775
Iteration 140, En = 0.5043457968, number of points: 1003328
Iteration 150, En = 0.5025355815, number of points: 1003928
Iteration 160, En = 0.5020546432, number of points: 1003691
Iteration 170, En = 0.5006839447, number of points: 1003910
Iteration 180, En = 0.4992380917, number of points: 1004281
Iteration 190, En = 0.5001416902, number of points: 1004120
Iteration 200, En = 0.500377161, number of points: 1004166
```

In[256]:=
```
Print["Average energy ", Mean[meanEnergy[[nt / 2 ;; nt]]]];
ListPlot[meanEnergy]
```

```
Average energy 0.5041422797
```

Out[257]=



In[263]:=
```
Histogram[{xx[[1 ;; nn]], points[[1 ;; nini]]}, {a, b, (b - a) / 100}, "PDF",
 Epilog → First@Plot[ρ[x], {x, a, b}, PlotRange → All, PlotStyle → Red]]
```

Out[263]=