HW1 – RNG's with specified distribution

Enrique Gómez Cruz
Facultad de Ciencias, Universidad Nacional Autónoma de México.

1 Random number generation with specified distribution

1.1 Numerical inversion Method

We have a distribution function $w(x)$ normalized between x_1 and x_2. The interval $[x_1, x_2]$ is divided in M equal subintervals of size

$$I = \frac{x_2 - x_1}{M}$$

For each subinterval we apply the recurrence relation in equation 1 with steps of size Δy.

$$x_{i+1} = x_i + \frac{\Delta y}{w(x_i)} \tag{1}$$

The step size should be less than I. For example, we chose

$$\Delta y = \frac{I}{M} = \frac{x_2 - x_1}{M^2}$$

As an example, take $w(x)$ to be as in equation 2, normalized in the interval $[0, \pi]$.

$$w(x) = \frac{\sin^2(2x) + \cos^2(x)}{\pi} \tag{2}$$

We use algorithm 1 to get the random variable X with distribution w, and then we use X to calculate the histogram and cdf of w. The outputs produced with gnuplot are shown in figures 1 and 2. The former use 100 intervals and the latter 1000 intervals.

Algorithm 1 Numerical inversion method

$x \leftarrow x_1$ ▷ Start at lower bound
for each index $k \in [1, 2, \ldots, M]$ **do** ▷ M subintervals, each with upper bound Ik
    $x \leftarrow x + \Delta y / w(x)$
    while $x \leq Ik$ **do** ▷ Until end of interval is reached
        X.push(x)
        $x \leftarrow x + \Delta y / w(x)$
    end while
    $x \leftarrow Ik$ ▷ Start at beginning of next interval
end for

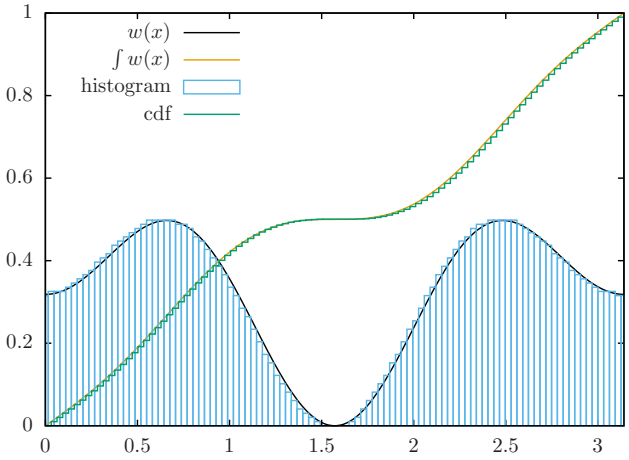

Figure 1: Inversion method with 100 equal intervals.

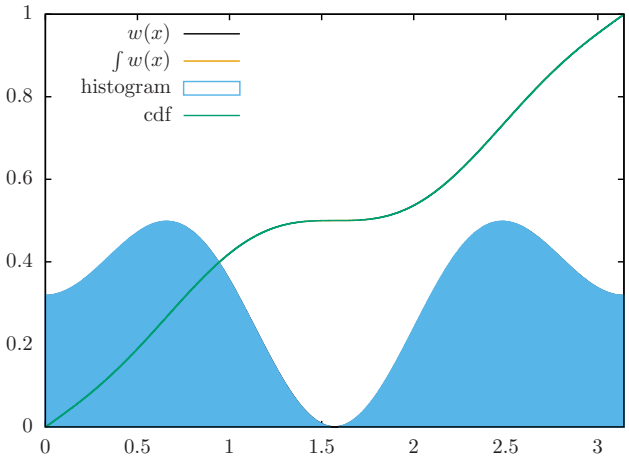

Figure 2: Inversion method with 1000 equal intervals.

Initially, the method applied equation 1 to the whole interval $[x_1, x_2]$ instead of applying it to individual subintervals of the form $[x_i - \epsilon, x_i + \epsilon]$. This made the method useless because when $w(x) \to 0$, x jumped out of the region of integration. Dividing the region of integration into M equally spaced intervals and applying equation 1 in each interval made the method work.

Now, looking at figures 1 and 2 we can see that dividing into a bigger number of subintervals, M, we get a better approximation of the density function.

1.2 von Neumann rejection method

This method uses one distribution $w(x)$ and one function $w'(x)$. Function $w'(x)$ is an upper bound of $w(x)$ for all x.

The region of integration is divided in M subintervals of size

$$\Delta x = \frac{x_2 - x_1}{M}$$

For each value $x = x_1 + i\Delta x$ $(i = 0, 1, \ldots, M)$ a random uniform value, η, is generated. If η is less than the probability of acceptance, $w(x)/w'(x)$, then the value x is accepted.

Algorithm 2 Rejection/Acceptance method

```
1: x ← x_1
2: while x ≤ x_2 do
3:     η ← uniform random between 0 and 1
4:     if η ≤ w(x)/w'(x) then
5:         X.push(x)
6:     end if
7:     x ← x + Δx
8: end while
```

Once again equation 2 is used as the example. Figures 3 and 4 show the outputs produced with gnuplot.

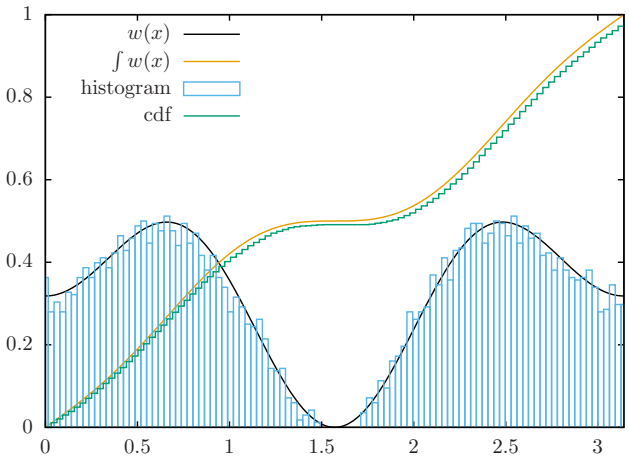

Figure 3: Rejection method with 100 equal intervals.

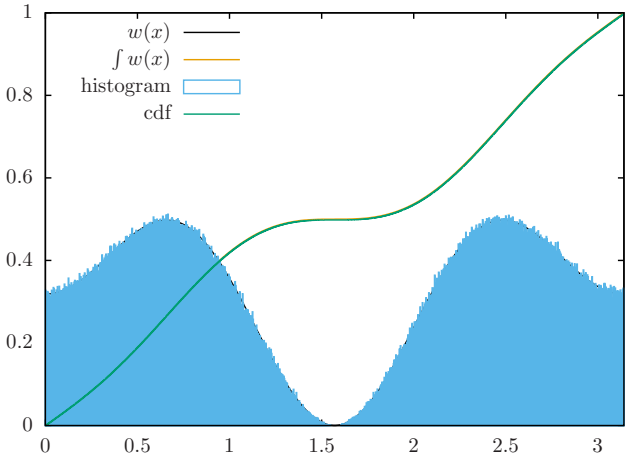

Figure 4: Rejection method with 1000 equal intervals.

There's a clear difference between the rejection method and the inversion method. The inversion method makes a more precise approxi mation of the density function because it is not using a random uniform value that produces noise. Nonetheless, both methods achieve almost the same looking plots with the same partition of the region of integration.

1.3 Metropolis method

The region of integration, $[x_1, x_2]$, was divided in M equally spaced subintervals of size ΔM. Each point in the partition is given by $x_1 + i\Delta M$ where $i = 0, 1, \ldots, M$. There are $M + 1$ walkers each positioned in a different point of the parition. Every walker is going to sample, in s steps, a percentage of the region of integration by making random steps (using η_1, a random uniform number between 0 and 1) of size δ.

Algorithm 3 Metropolis method

```
for each index i ∈ [0, 1, ..., M] do
    X.push(x_1 + iΔM)
    for each index j ∈ [1, 2, ..., s] do
        X_n ← X.back()                    ▷ Get last item in X and save it as X_n
        X_t ← X_n + (2η_1 − 1)δ           ▷ Trial step
        if w(X_t)/w(X_n) > η_2 then       ▷ η_2 random uniform number in [0, 1]
            X.push(X_t)
        end if
    end for
end for
```

This was the heaviest method of the three, taking 20 seconds longer to output the plot with 1000 walkers. As the density of the walkers become greater the approximation also becomes better. And also as the walk becomes larger the approximation to the distribution function becomes better. It was found experimentally that the step size, δ, was better off being

$$\delta = \Delta M$$

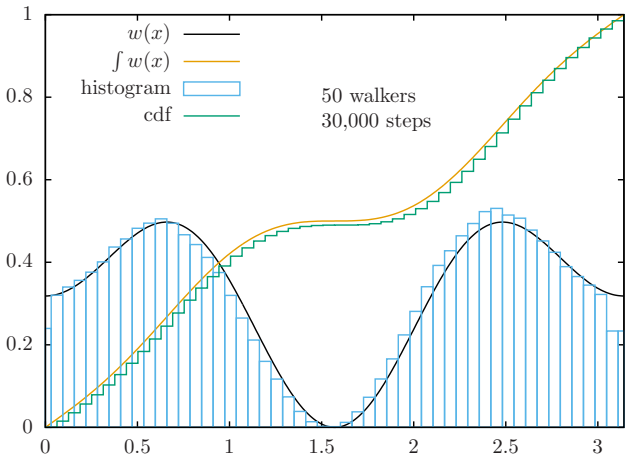

Figure 5: Metopolis method with 50 walkers and 30,000 steps.

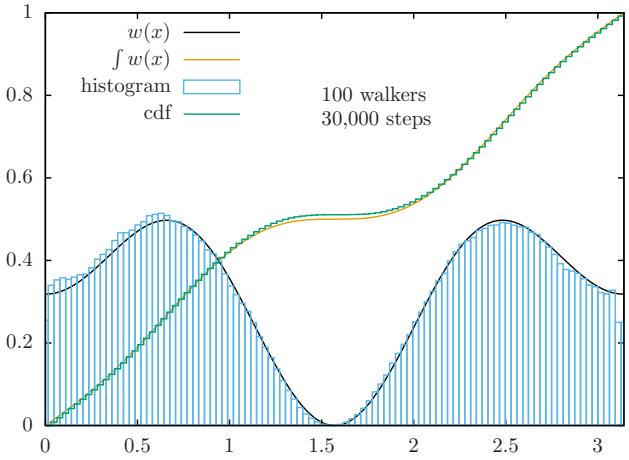

Figure 6: Metopolis method with 100 walkers and 30,000 steps.

As the walk size tends to infinity the metropolis algorithm gets to equilibrium. This process of equilibrium is plotted in figures 7 and 8. It can be seen that the bigger the number of walkers becomes, the bigger the walk size must be. This is seen due to the fact that 50 walkers with a walk size of 1000 is better than 100 walkers with a walk size of 1000.

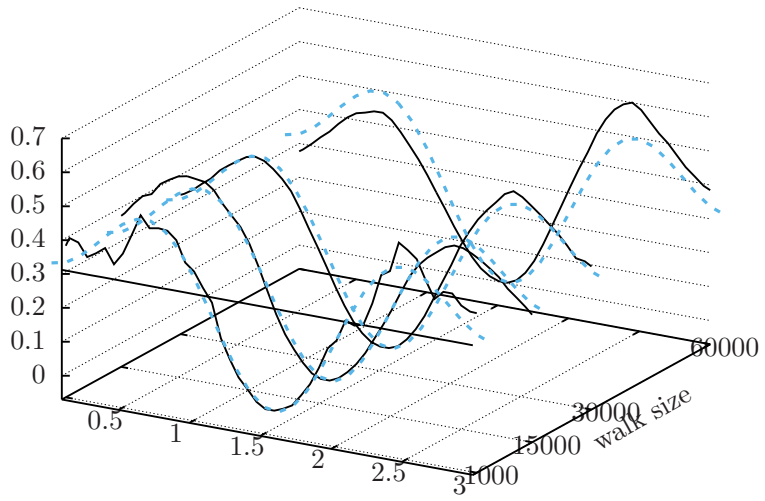

Figure 7: Equilibrium process in metropolis algorithm with 50 walkers and walk sizes of 1, 1000, 15000, 30000, 60000 steps.

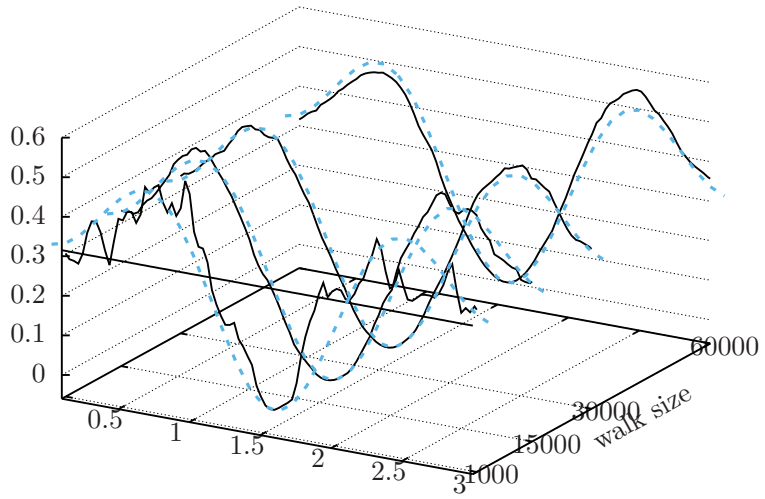

Figure 8: Equilibrium process in metropolis algorithm with 100 walkers and walk sizes of 1, 1000, 15000, 30000 and 60000 steps.

1.4 A Gaussian distribution

$$w(x) = \frac{1}{\sqrt{\pi}} e^{-x^2}$$

This distribution is normalized in the interval of $[-\infty, \infty]$ but since it quickly goes to zero we don't need to sample the whole interval.

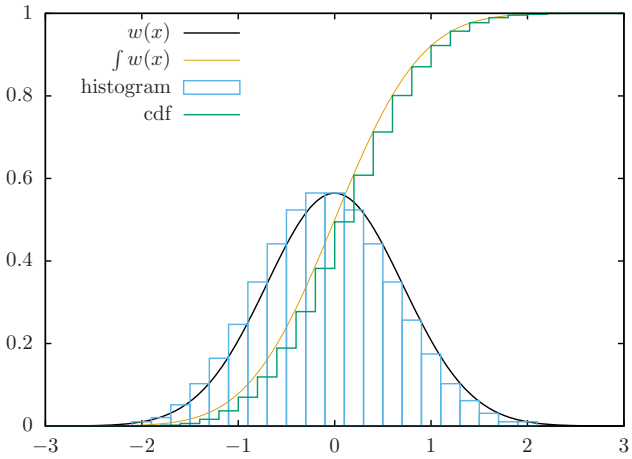

Figure 9: Inversion method with 100 equal intervals.

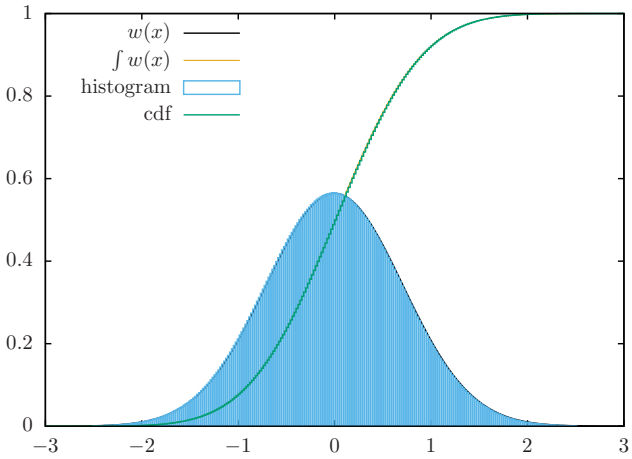

Figure 10: Inversion method with 1000 equal intervals.

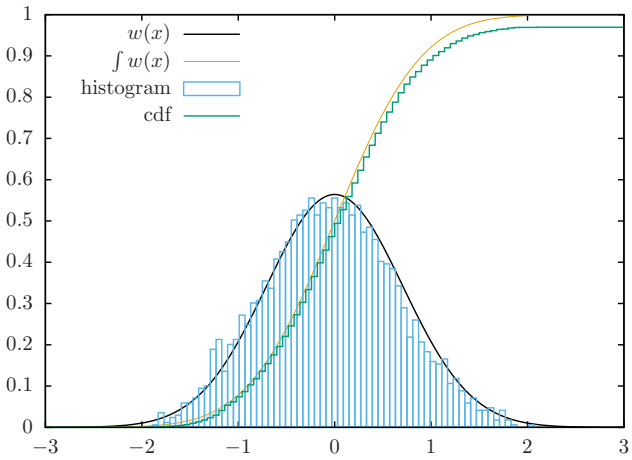

Figure 11: Rejection method with 100 equal intervals.

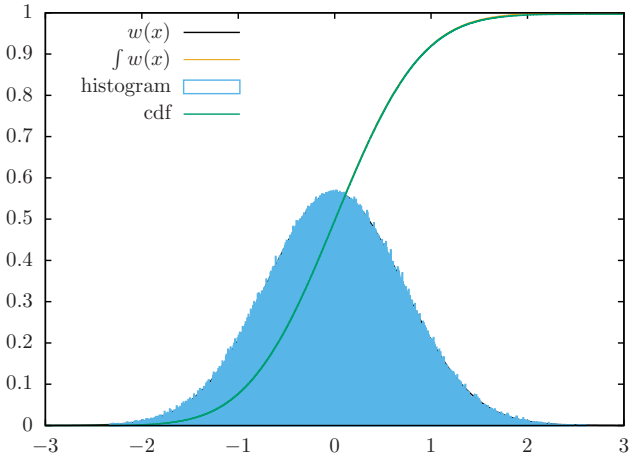

Figure 12: Rejection method with 1000 equal intervals.

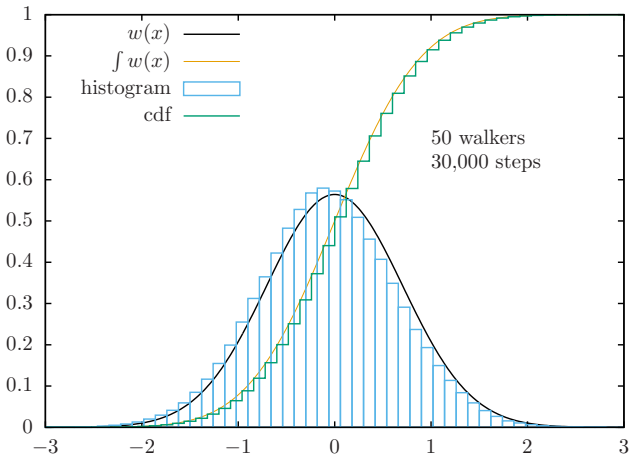

Figure 13: Metropolis method with 50 walkers and 30,000 steps.

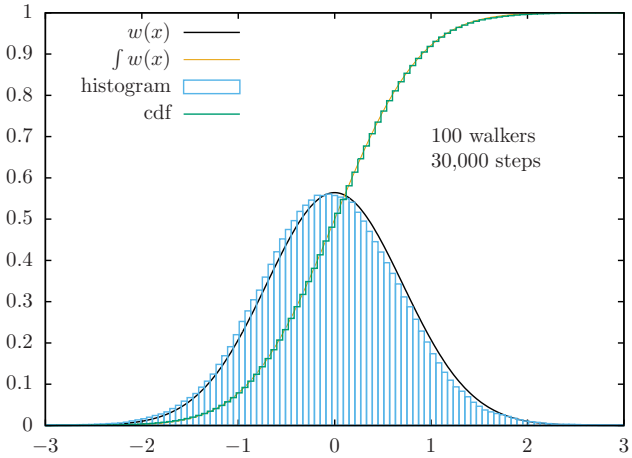

Figure 14: Metropolis method with 100 walkers and 30,000 steps.

The equilibrium process of the metropolis algorithm is shown in the next figures.

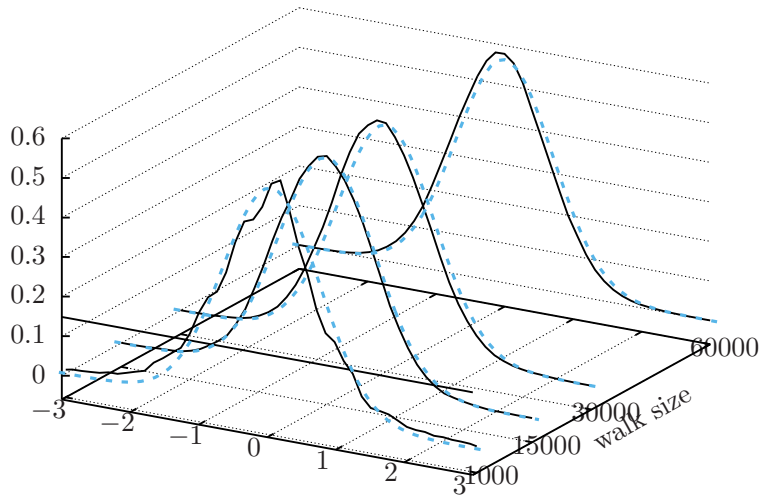

Figure 15: Equilibrium process in metropolis algorithm with 50 walkers and walk sizes of 1, 1000, 15000, 30000, 60000 steps.

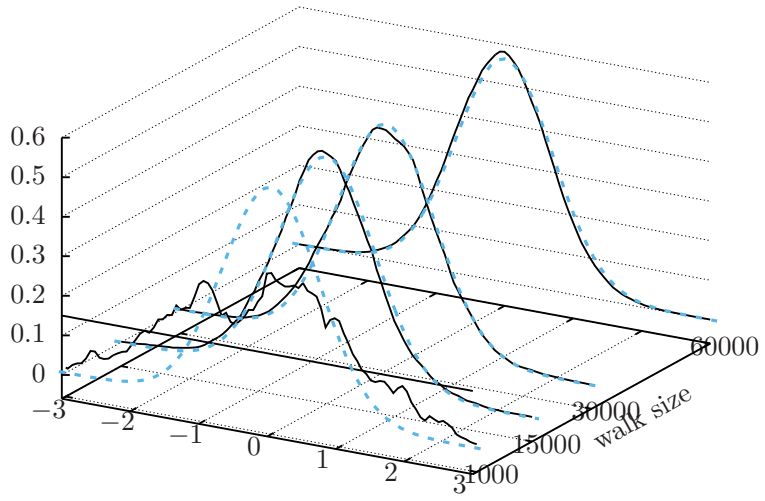

Figure 16: Equilibrium process in metropolis algorithm with 100 walkers and walk sizes of 1, 1000, 15000, 30000 and 60000 steps.