# MA2000: OTML

*Madhab Barman & Nachiketa Mishra*

*Indian Institute of Information Technology, Design & Manufacturing, Kancheepuram*

# Monte Carlo Integration

- Assume that we want to evaluate the integral of $f(x)$ over $[0, 1]$

$$I = \int_0^1 f(x)\, dx$$

- Given a uniform random variable $X_i$ over $[0, 1]$, then Monte Carlo Integration is

$$\boxed{\int_0^1 f(x)\, dx \approx \frac{1}{N} \sum_{i=1}^{N} f(X_i) = \frac{1}{N} \sum_{i=1}^{N} f_i := I_N, \quad \text{(say)}}$$

- The variance

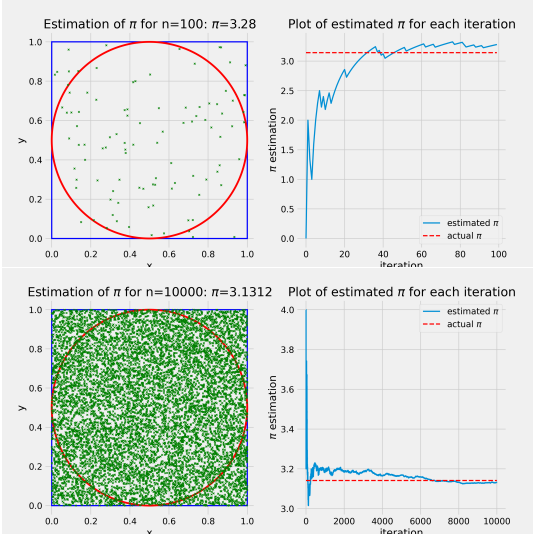
$$\boxed{\sigma_f^2 = \left[ \frac{1}{N} \sum_{i=1}^{N} f_i^2 - \left( \frac{1}{N} \sum_{i=1}^{N} f_i \right)^2 \right]}$$

- Approximate the value of the integral as $I = I_N \pm \dfrac{\sigma_f}{\sqrt{N}}$
- The accuracy of Monte Carlo integration increases with the number of sampling points N

# **Monte Carlo Integration**: Estimating $\pi$

- $A_s$:= a unit square Area
- Draw a circle inside this square with a radius of 0.5. The Area of this circle is $A_c = \frac{\pi}{4}$
- We create n random points inside the square
- Therefore,
  $\frac{A_c}{A_s} = \frac{\pi}{4} \Rightarrow \pi = 4 \times \frac{A_c}{A_s}$
- Thus

$$\pi \approx 4 \times \frac{\text{Total number of points in circle}}{\text{Total number of points}}$$



# Importance Sampling

- In the simple Monte Carlo integration, we have used the uniform sampling
- For an integrand $f(x)$ that varies rapidly in a narrow region such as a sharp peak:
  - the only sampling points that are important are near the peak, the sampling points far outside will contribute less.
- Thus, it seems that a lot of unnecessary sampling points are wasted.
- There are two main ways to use the sampling points more effectively, and they are change of variables and importance sampling.

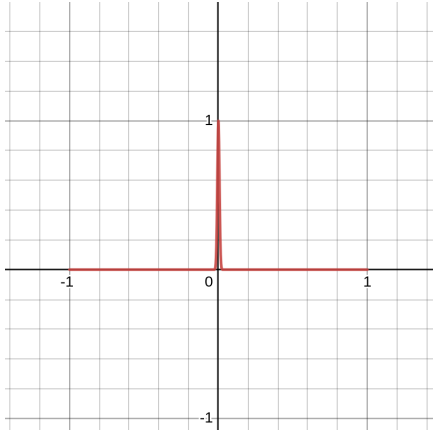



Figure: $f(x) = e^{-(100x)^2}$

# Change of Variables

- The change of variables uses the integrand itself so that it can be transformed to a more uniform (flat) function. For example, the integral

$$I = \int_a^b f(u)\, du$$

- Transform by using a known function $u = g(v)$, which yields $du = g'(v)dv$

$$I = \int_a^b f(u)\, du = \int_{a_v}^{b_v} f[g(v)]g'(v)\, dv$$

- The idea is to make sure that the new integrand is or close to a constant $A$

$$\phi(v) = f[g(v)]g'(v) = A, \quad \text{where } v = g^{-1}(u)$$

- This means that a uniform sampling can be used for $\phi$
- The new integration limits are $a_v = g^{-1}(a)$ and $b_v = g^{-1}(b)$

# Example

- An exponential function $f(u) = e^{-\alpha u}$
- Transform $u = g(v) = -\frac{1}{\alpha} \ln(\alpha v)$
- Compute $f(u)g'(v) = -1 = A$
- We then hav

$$I = \int_a^b f(u)\, du = \int_{\exp(-\alpha b)/\alpha}^{\exp(-\alpha a)/\alpha} (-1)\, dv$$

- Then, we can use a uniform sampling set to estimate the integral.
- Both these methods are limited either to the case when $g^{-1}$ exists uniquely and has explicit expressions in terms of basic functions, or to the case when $f(x)$ are flat in subregions.
- Otherwise, it is not easy, or even impossible, to make such transformations or domain-decomposition.
- A far more efficient method is to use the importance sampling, rejection sampling

- As the integration is the area (or volume) under a curve (or surface), the region(s) with higher values of integrand will contribute more, thus, we should put more weights on these important points.
- Importance sampling is just the method for doing such weighted sampling.
- The integral of interest is often rewritten as the weighted form or a product such that

$$I = \int_a^b f(x)\,dx = \int_a^b \frac{f(x)}{p(x)} p(x)\,dx = \int_a^b h(x)p(x)\,dx = \langle h \rangle_p, \text{ where } h(x) = \frac{f(x)}{p(x)}$$

- Obviously, it is required that $p(x) \neq 0$ in $[a, b]$
- Here the function $p(x)$ acts as the probability density function whose integration over the region $[a, b]$ should be always equal to 1. i.e

$$\int_a^b p(x)\,dx = 1$$

- The evaluation of the integral becomes the estimation of the expected value of

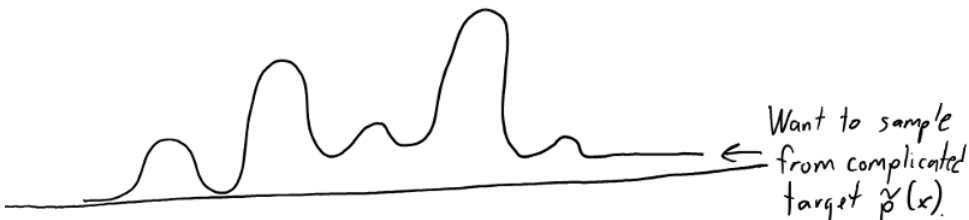
$$E[h] = \langle h(x) \rangle = \left\langle \frac{f(x)}{p(x)} \right\rangle_p$$

- The idea is to choose a function $p(x)$ such that the sampling points become more important when $f(x)$ is in the region with higher values.
- That is equivalent to a weighted sum with $p(x)$ as the weighting coefficients.
- The choice of $p(x)$ should make $h(x) = f(x)/p(x)$ as close to constant as possible.
- In a special case: $p(x) = 1/(b-a)$, we have the following:
- Monte Carlo integration of $f(x)$ on $[a, b]$ is defined by

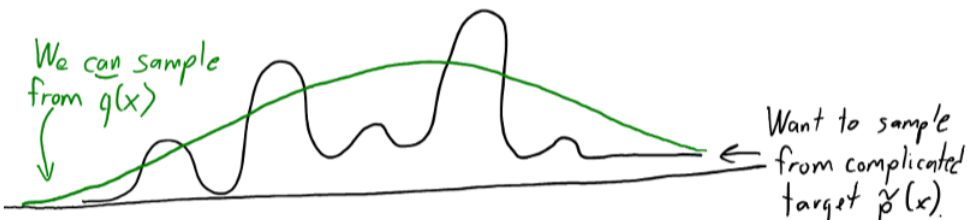
$$\int_a^b f(x)\,dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(X_i) = \frac{b-a}{N}\sum_{i=1}^{N} f_i,$$

where uniform random variable $X_i$ over $[a, b]$

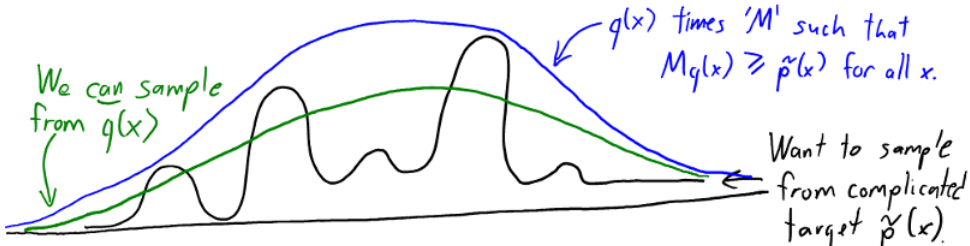
# Rejection sampling



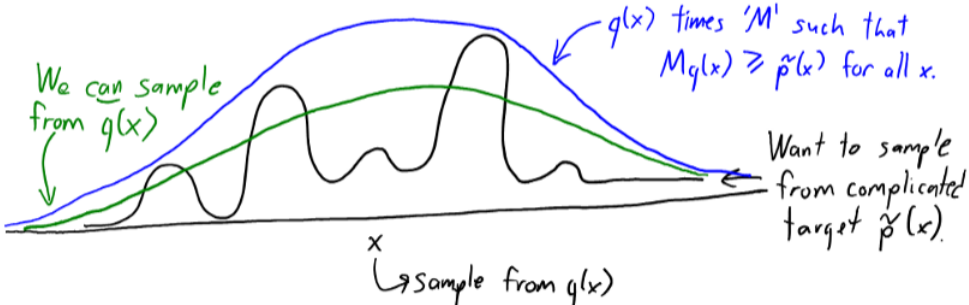
# Rejection sampling



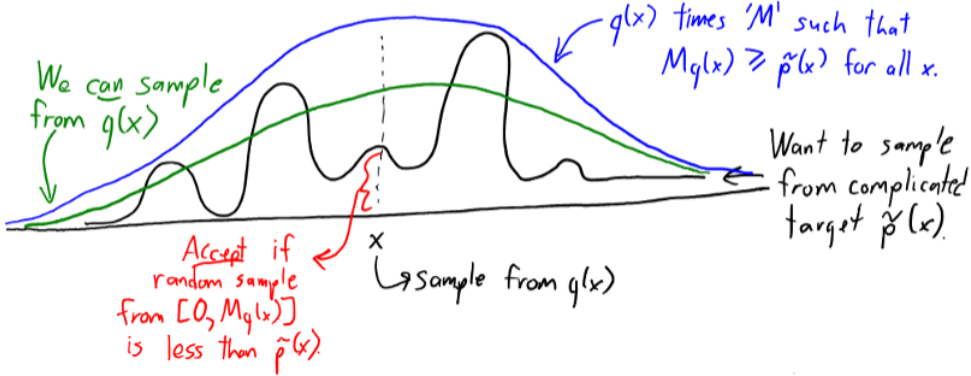
# Rejection sampling



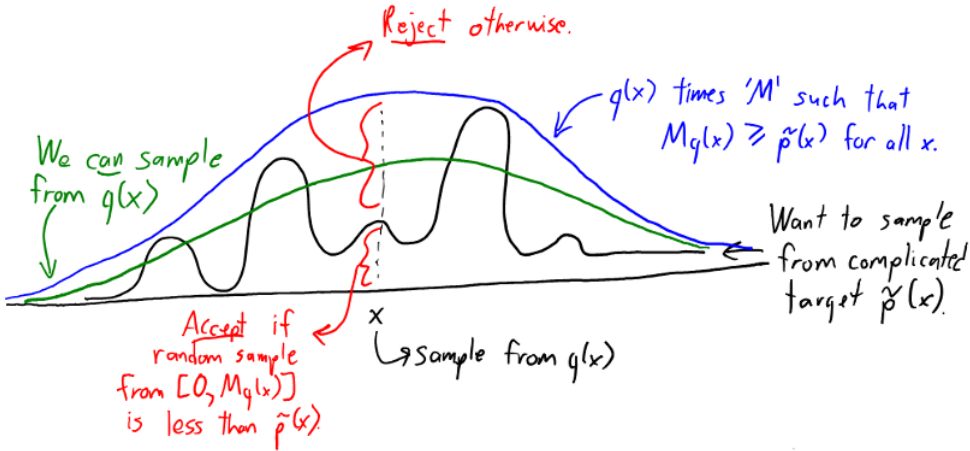
# Rejection sampling



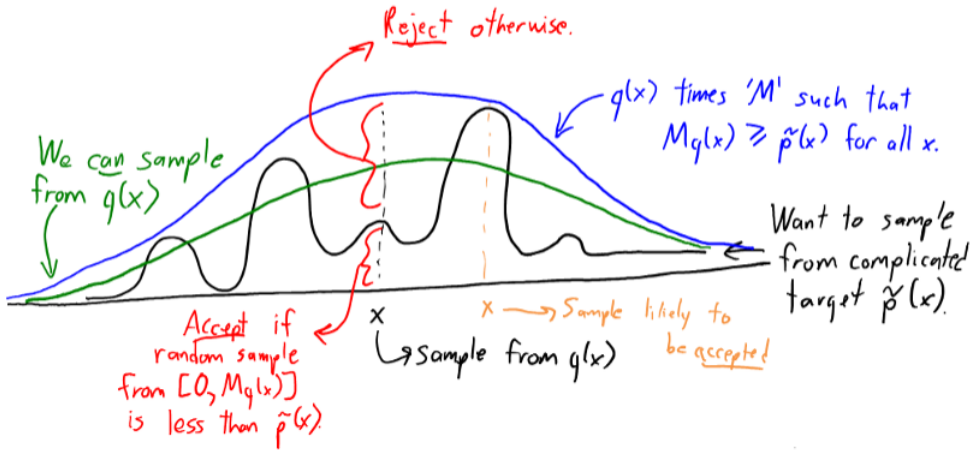
# Rejection sampling



# Rejection sampling



# Rejection sampling



# Rejection sampling

- Assume, a distribution $q$ that is easy to sample from
- an upper bound $M$ on $\tilde{p}(x)/q(x)$, where

$$\frac{\tilde{p}(x)}{Z} = p(x),$$

$Z$ is normalization part

**Rejection algorithm:**

1. sample from $q(x)$
2. sample $u$ from a uniform distribution $\mathcal{U}(0, 1)$
3. keep the sample (accept) if $u \leq \frac{\tilde{p}(x)}{Mq(x)}$, otherwise reject

**Drawbacks:**

- You may reject a large number of samples.
  - Most samples are rejected for high-dimensional complex distributions
- You need to know M.

# Recap: Conditional Probability

- The conditional probability of an event $A$, given that the other event $B$ has happened,

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

- Suppose $A$ and $B$ are independent, $P(A,B) = P(A)P(B)$,

$$P(A|B) = \frac{P(A,B)}{P(B)} = \frac{P(A)P(B)}{P(B)} = P(A)$$

# MCMC

## Definition (Markov chain)

A series of random variables $X_1, X_2, \ldots, X_t, \cdots \sim f(x)$ is a Markov chain if

$$P(X_{i+1} = y | X_i, X_{i-1}, \ldots, X_1) = P(X_{i+1} = y | X_i).$$

Example: random walk $X_{i+1} = X_i + \Delta x_i$ where $\Delta x_i$ are i.i.d is a Markov chain.

## Definition (Stationary)

A Markov chain is stationary if

$$P(X_{i+1} = y | X_i = x)$$

is independent of $i$.

- How do we construct a Markov chain whose stationary distribution is our target distribution, $f(x)$ ?
- Metropolis et al. (1953) illustrated how.
- Then, the method was generalized by Hastings (1970): Metropolis-Hastings algorithm

# Metropolis-Hastings algorithm

- **Idea:** To define a Markov chain over possible values, in such a way that the stationary distribution of the Markov chain is in fact $f(x)$.

---

At each iteration $t+1$

Step-1. sample $y \sim q(y|x^{(t)})$, where $y$ is the candidate point and $q$ is the proposal distribution

Step-2. compute the probability

$$\alpha(x^{(t)}, y) = \min\left\{1, \frac{f(y)q(x^{(t)}|y)}{f(x^{(t)})q(y|x^{(t)})}\right\}$$

Step-3. Draw $u$ from $\mathcal{U}(0,1)$

Step-4. **If** $\alpha \geq u$, then accept $x^{(t+1)} = y$;

**else** reject, $x^{(t+1)} = x^{(t)}$.

---

- This special case of the Metropolis-Hastings algorithm, in which the proposal distribution is symmetric $q(x^{(t)}|y) = q(y|x^{(t)})$, is referred to as the Metropolis algorithm.
- **Useful feature of the MH algorithm:** it can be implemented even when $f(x)$ is known only up to a constant: that is, $f(x) = ch(x)$ for some known $h$, but unknown constant $c$.

# Example of MH

- Here, we implement the algorithm to sample from an exponential distribution:

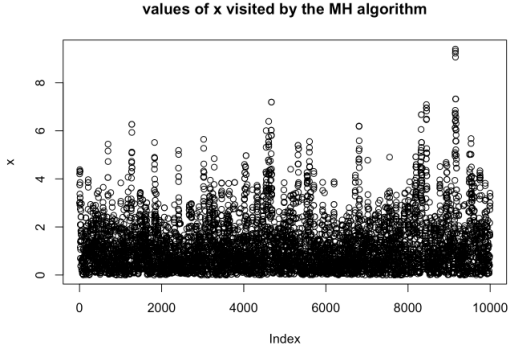
$$f(x) = \exp(-x), \quad x \geq 0$$

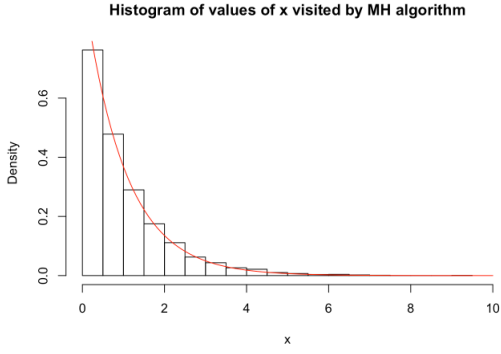
- So, **target** distribution is $f(x) = \exp(-x)$

```
target = function(x){
  return(ifelse(x<0,0,exp(-x)))
}
```

```
x = rep(0,10000)
x[1] = 3     #initialize; I've set arbitrarily set this to 3
for(i in 2:10000){
  current_x = x[i-1]
  proposed_x = current_x + rnorm(1,mean=0,sd=1)
  A = target(proposed_x)/target(current_x)
  if(runif(1)<A){
    x[i] = proposed_x       # accept move with probabily min(1,A)
  } else {
    x[i] = current_x        # otherwise "reject" move, and stay where we are
  }
}
```

# contd.




- Here $q$ is symmetric, therefore, the ratio of $q$'s become 1
- **Remember!** we designed this algorithm to sample from an exponential distribution.
- The histogram of x should look like an exponential distribution

# Markov Chain And Optimization

- **Stochastic approach:** pick items randomly $x_1, x_2, \ldots, x_N$ from your search space $X$ and return $\arg \max_{i=1,2,\ldots,N} f(x_i)$ or $\arg \min_{i=1,2,\ldots,N} f(x_i)$
- What probability distribution should we use to pick these points?
- Pick x uniformly from $X$ which is simple (bad) distribution.
  - **Problem**: we might spend most of the time sampling junk.
- Great distribution: Softmax $p(x) = \frac{e^{f(x)/T}}{Z}$ where T is a parameter and $Z = \sum_x e^{f(x)/T}$ is the normalization term.
- It is not easy to calculate $Z$ when dimension increases.
- To solve this problem we use MCMC (Markov Chain Monte Carlo) sampling.

# Simulated Annealing

- Unconstrained optimization

$$\min_x f(x)$$

- **Simulated annealing**:
  1. Start from an initial point
  2. Repeatedly consider various new solution points
  3. Accept or reject some of these solution candidates by
  4. Converge to the optimal solution
- It is based on "similarities" and "analogies" with the way that alloys manage to find a nearly global minimum energy level when they are cooled slowly
- **Applications:** the knapsack problem, the traveling salesman problem, Air traffic optimization, etc..

# Local optimization vs. Simulated annealing

**Simulated annealing**

1. Start from an initial point
2. Repeatedly consider various new solution points
3. Accept/reject new solution using probability at each iteration
4. Converge to the optimal solution

**Local optimization**

1. Start from an initial point
2. Repeatedly consider various new solution points
3. Reduce cost function at each iteration
4. Converge to the optimal solution

# Simulated Annealing

Let the objective function $f(X)$, $X \in \mathbb{R}^n$

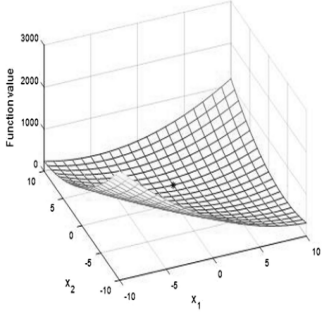
- Initialize initial temperature $T_0$ and initial guess $X = X^{(0)}$ and $i = 0$
- Set minimum temperature $T_{min}$ and max number of iterations $N$, and cooling rate $\alpha$
- Define cooling schedule $T \to \alpha T$ where $0 < \alpha < 1$.
- **While** ($T < T_{min}$ and $i < N$)
  - Move randomly to new locations $X^{(i+1)} = X^{(i)} + \texttt{rand}$
  - Calculate $\Delta f = f(X^{(i+1)}) - f(X^{(i)}) = f^{(i+1)} - f^{(i)}$
  - Accept the new solution if better i.e., $f(X^{(i+1)}) < f(X^{(i)})$
  - **If** not improved i.e., $f(X^{(i+1)}) \geq f(X^{(i)})$
    - Generate a random number $r$
    - Accept if $p = \exp(-\Delta f / T) > r$
  - **End if**
  - Update the best $X_*$ and $f_*$
  - i = i+1
- **End While**

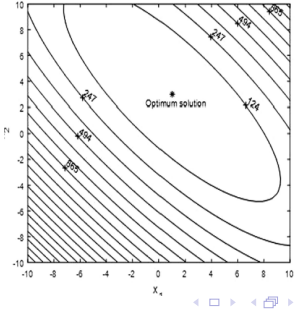
# Example.

Find the minimum to the objective function

$$f(x_1, x_2) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2,$$

where $-10 \leq x_1, x_2 \leq 10$. $(\min(f) = 0 \text{ at } x_1 = 1, x_2 = 3)$



# Example

Step-1 **Initialization**
Initial guess $X^{(0)} = (0, 0)$; Initial temperature $T = 1000$ and the initial value of iteration counter $i = 0$. So $f(X^{(0)}) = 74$, Choose $\alpha = 0.5$, $T_{min} = 1, i = 1$.

Step-2 **Generation of a new point in the nbd. of current point**
Pick a new point $X^{(1)} = X^{(0)} + \texttt{rand}$ such that $X^{(1)} \in [-10, 10]$
$X^{(1)} = (-7.337, 0.53718)$ and $f(X^{(1)}) = 542.11$;
Compute $\Delta f = f(X^{(k+1)}) - f(X^{(k)}) = 542.11 - 74 = 468.11$

Step-3 **Checking the Acceptance of the new point**
Since $f(X^{(1)}) > f(X^{(0)})$, new solution is not improved. So, we calculate the probability of accepting the new point $X^{(1)}$: $p = e^{-\Delta f/T} = e^{-468.11/1000} = 0.6262$.
Next, we pick a random number in between 0 and 1, say, $r = 0.43091$
Now, since $p > r$ we **accept** the new point $X^{(1)} = (-7.337, 0.53718)$.

Step-4 **Lowering the temperature as per cooling schedule**
The new temperature $T = \alpha T = 0.5 * 1000 = 500$ and since $T > T_{min} = 1$, we go to **Step-2**. Repeat!

# Example

- Now $i = i + 1 = 2$

Step-2 We pick $X^{(2)} = (-11.2205, 1.8084)$ which is not belongs to the region $[-10, 10]$. So $X^{(2)}$ remains same as $X^{(1)}$ i.e., $X^{(2)} = (-7.337, 0.53718)$. Go to **Step-4**.

Step-4 $T = 0.5 * 500 = 250$ and since $T > T_{min}$, go to **Step-2.**

- Now $i = i + 1 = 3$

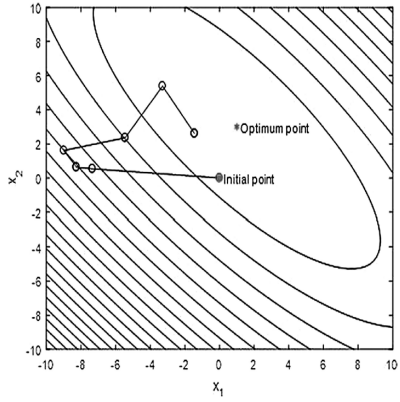
Step-2 $X^{(3)} = (-8.2649, 0.6315)$ and $f(X^{(3)}) = 632.79$; Compute $\Delta f = f(X^{(3)}) - f(X^{(1)}) = 632.79 - 542.11 = 90.68$

Step-3 Compute $p = e^{-90.68/250} = 0.69578$ and select random $r \in (0, 1), r = 0.25622$ Now, since $p > r$ we **accept** the new point $X^{(3)} = (-8.2649, 0.6315)$.

Step-4 $T = 0.5 * 250 = 125$ and since $T > T_{min}$, go to **Step-2.** and **Repeat!**

# First 10 iterations

| Iteration | Values of the Variables |
|---|---|
| $i = 0$ | $X^{(0)} = (0, 0)$ |
| $i = 1$ | $X^{(1)} = (-7.337, 0.53718)$ |
| $i = 2$ | $X^{(2)} = (-7.337, 0.53718)$ |
| $i = 3$ | $X^{(3)} = (-8.2649, 0.6315)$ |
| $i = 4$ | $X^{(4)} = (-8.2649, 0.6315)$ |
| $i = 5$ | $X^{(5)} = (-8.9895, 1.6117)$ |
| $i = 6$ | $X^{(6)} = (-5.4466, 2.3564)$ |
| $i = 7$ | $X^{(7)} = (-5.4466, 2.3564)$ |
| $i = 8$ | $X^{(8)} = (-3.2893, 5.386)$ |
| $i = 9$ | $X^{(9)} = (-1.4429, 2.6077)$ |
| $i = 10$ | $X^{(10)} = (-1.4429, 2.6077)$ |




- If we run this algorithm in MATLAB/Python/Etc., we will get $X^* = (0.97556, 3.0229)$. Therefore, required approximated minimum of $f$ is equal to 0.00113.

# Exercise

A. Apply SA to find the minimum of the function

$$f(x_1, x_2) = (1 - x_1)^2 + 100(x_2 - x_1^2)^2$$

on $[-10, 10] \times [-10, 10]$.

B. Python code: `https://machinelearningmastery.com/simulated-annealing-from-scratch-in-python/`