Maxwell's Demon : Statistical Mechanics Project

Ammar Ali Hakeem,[1,*] Javaria Ghafoor,[1,†] and Zohran Ali[1,‡]

[1]*Department of Physics, School of Natural Sciences, NUST, H-12 Islamabad, Pakistan*

Who would not like to have a wheel that runs on its own without doing any work? This project report emphasizes on the notion of a Maxwell Demon and how it challenges the second law of thermodynamics. The paper includes the details about various approaches and efforts that have been carried out to analyze the idea. This report also includes a brief study of the different types and definitions of entropy and its statistical basis.

INTRODUCTION

Maxwell's Demon is an essential and a debatable being in thermodynamics. It has celebrated its 152nd birthday this year. Originally, born in 1867 in the papers of Maxwell and was named by Thomson who did not conceive the creature to be hostile. Its significance arises from the truth that it's believed to hypothetically violate the famous second law of thermodynamics - that is considered inviolable, absolute and beyond question. Since birth, the demon created controversies mocking physicists each who favor it and who oppose it alike. Many different demons got here on the scene making the difficulty of inviolability of the second law debate a never-ending task. Since our motive is to analyze the challenge it proposes towards the second law of thermodynamics, we do no longer delve more into the historic aspects however rather emphasize on a particular and direct explanation of its relationship with the second law of thermodynamics. Again, seeing that our method is solely thermodynamics based, we do not concern ourselves in this report, with the works that explain the issue from quantum mechanical points of view.

2nd Law of Thermodynamics

It is not possible for a process to have as its sole result the transfer of heat from a cooler body to a hotter one. The Second Law of Thermodynamics can be described via incorporating the concept of entropy. Which is,

$$\triangle S = dQ/dT \tag{1}$$

Defined this way, the Second Law can be restated as: In any closed system, the entropy of the system will either remain constant or increase. By "closed system" it means that every part of the process is included when calculating the entropy of the system.

Maxwell's distribution and the statistical nature of the second Law

$$f(v) = \left(\frac{2}{\pi}\right)^{1/2} \left(\frac{m}{kT}\right)^{3/2} v^2 \exp\left[-\frac{mv^2}{2kT}\right] \tag{2}$$

This formula gives the probability of a molecule in a gas having a certain speed v.

Three graphs of this probability distribution (at different temperatures) are shown in Fig. 1. The probability is highest at a particular point, the most probable speed, which is linked with the temperature. But on either side of this point there are many molecules with lower and higher speeds. The area under each of the curves is 1, because this is a distribution of probabilities and every possibility must be accounted for, and this explains the lower curves for higher temperatures.
This distribution could also predict the second law of thermodynamics. Suppose you have a hot substance with many fast moving molecules and you bring it into contact with a cold substance containing slow moving molecules. The fast molecules will hit the slow molecules accelerating them, as they do this they will slow down. This is the same as in the second law, 'heat flows from a hot body to a colder body'.

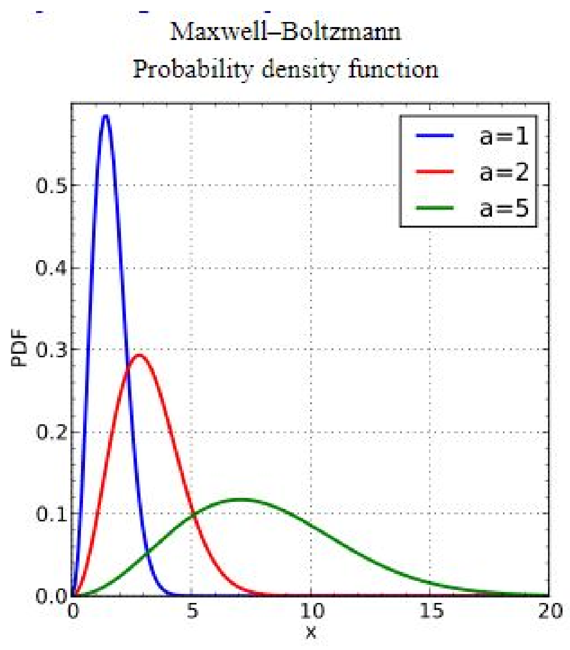

FIG. 1: Maxwell Probability Distribution

What was so revolutionary about this? The Maxwell-Boltzmann statistical distribution. It was what was most likely to happen. But just occasionally every single molecule in the hot substance might be hit from the side by a molecule in the cold substance. This would mean heat flowing from cold to hot.

When Maxwell published his "Theory of Heat" in 1871, which explained ideas of heat and molecular kinetics in a manner simple enough for a student to understand, he introduced at the end a thought experiment of a demon to explain this.

" ... if we conceive of a being whose faculties are so sharpened that he can follow every molecule in its course, such a being, whose attributes are as essentially finite as our own, would be able to do what is impossible to us. For we have seen that molecules in a vessel full of air at uniform temperature are moving with velocities by no means uniform, though the mean velocity of any great number of them, arbitrarily selected, is almost exactly uniform. Now let us suppose that such a vessel is divided into two portions, A and B, by a division in which there is a small hole, and that a being, who can see the individual molecules, opens and closes this hole, so as to allow only the swifter molecules to pass from A to B, and only the slower molecules to pass from B to A. He will thus, without expenditure of work, raise the temperature of B and lower that of A, in contradiction to the second law of thermodynamics. " (Theory of Heat, 1871)

ON MEASURING THE VELOCITIES

Using Light signals

A very famous argument against the demon was given by Brillouin [3]. The argument showed that because the system was at constant temperature the demon could not see the molecules due to black body radiation. And if the demon can not see the molecules it can not operate the trap door.

Now if we introduce a source of light, so as the demon can see the molecules. We shall find that the total entropy increases. What follows is the argument that Brillouin gave.

In order to discuss an entropy balance, we first consider an isolated system. The system is composed of:

1. A charged battery and an electric bulb, representing source of light.

2. A gas at constant temperature T_0, contained in Maxwell's enclosure, with a partition dividing the vessel into two portions and a hole in the partition.

3. The demon operating the trap door at the hole.

The whole system is insulated and closed. The battery heats the filament at a high temperature T_1. Now to make the particles visible,

$$T_1 \gg T_0 \tag{3}$$

thus,

$$h\upsilon_1 \gg k_B T_0 \tag{4}$$

During the experiment, the battery yields a total energy E and no entropy. The filament radiates E and an entropy S_f,

$$S_f = E/T_1 \tag{5}$$

If the demon does not intervene, the energy E is absorbed in the gas at temperature T_0, and we observe a global increase of entropy,

$$S_f = E/T_1 > S_f > 0 \tag{6}$$

Now let us investigate the work of the demon. He can detect a molecule when at least one quantum of energy $h\upsilon_1$ is scattered by the molecule and absorbed in the eye of the demon (e.g. in a photoelectric cell etc). This represents a final increase of entropy.

$$\triangle S_d = h\upsilon_1/T_0 = k_B c \tag{7}$$

where $\triangle S_d$ is the change of entropy in demon and c is a constant (using the condition $h\upsilon_1 \gg k_B T_0$).

Once the information is obtained, it can be used to decrease the entropy of the system. The entropy of the system is,

$$S_0 = k_B \ln P_0 \tag{8}$$

where P_0 represents the total number of microscopic configurations os the system. After the information has been obtained, the system is more completely specified. P is decreased by an amount of p and $P_1 = P_0 - p$

$$\triangle S_i = S - S_0 = k_B \ln(1 - p/P_0) \tag{9}$$

using Taylor expansion of $\ln(1+x) \approx x$ for small x we get ,

$$\triangle S_i = -k_B(p/P_0) \tag{10}$$

It is obvious that $p \ll P_0$ in all practical cases. The total balance of entropy is ,

$$\triangle S_d + \triangle S_i = -k_B(c - (p/P_0)) > 0 \tag{11}$$

since $c \gg 1$ and $p/P_0 \ll 1$. The final result is still an increase of entropy in the isolated system.

Use of Doppler-shifted Radiation

While considering the arguments in detecting the particle using light signals we missed one important detail. This detail was pointed out by Jack Denur [7]. He pointed out that the equilibrium black body radiation, like all radiation, is, in general, Doppler shifted if it is emitted and/or reflected from a moving body. He argued that this fact, in principle could save the demon. Thus, the demon could in principle violate the Second Law of Thermodynamics.

The argument was like follows. He would instruct this demon to open a gate between two parts of a container of thermal particles whenever the fluctuation of thermal radiation received by him seems to indicate that a particle is moving towards the gate in the direction of intended particle transport.

We are not mentioning the details of his arguments because it had its own flaws which were corrected by H.Motz [6]. He thus showed that the demon even using doppler shifted radiation would not violate the second law of thermodynamics.

Other techniques

As we saw in the previous section that if we take the statement of "seeing" the individual particles seriously (e.g. use of light) we saw that one could not in principle detect the particle. However, this argument is restricted as there can be other ways of detecting the particle which does not need "light".

Some of these were suggested by people in different people. For example detecting molecules via their magnetic moments or by purely mechanical ways, or even using van der Waals forces.

Since all these arguments showed that the second law of thermodynamics is not violated we do not go into these arguments. We assume that these would be similar to the one given by light signal and thus prove only the point that using known techniques for detecting particles, the demon would not violate the Second Law of thermodynamics.

FEYNMAN'S RACHET AND PAWL DEVICE

The basic motive behind introducing such a device was to actually understand the mechanics of theoretically well explained and documented fact that there is a maximum amount of work that can be extracted from a system. Carnot did mathematically described about such an efficient engine yet it lack an elementary notion that is it would have been nice to find out an explanation that we could see what is happening physically.

So, let's assume that we have built a device that defies second law of thermodynamics, that is this gadget would be able to extract work from a heat reservoir with everything at the same temperature. Let's build such a gadget(or at least try building it). Let us say we have a box of gas that has a vane and an axle attached to it, at certain temperature ($T_1 = T_2 = T$, say). The collisions of the gas molecules with the vane may result in oscillations and the vane might jiggle as well. Now in order to make sure that we allow just one directional motion we attach a ratchet and pawl and a wheel to the other end of the axle [5](Fig. 2). When the shaft tries to jiggle one way, it will not turn and if it jiggles the other way, it will. Then the wheel will slowly turn. We can even attach a flea onto a string hanging by a pulley to the shaft and lift the flea.

Now let us ask if this is possible. According to Carnot's concept it is impossible but of we just look at it, it seems quite possible. So, let us analyze deeply what is actually happening.First we have assumed that we have an idealised ratchet that is as simple as possible but it still has a pawl and spring must be attached to the pawl to make it able to restore its original position after coming of the tooth. Another feature of the device is that suppose it were made of perfectly elastic parts. After the pawl is lifted off the tooth end it will be pushed back by the spring at it will continue bouncing against the wheel. When another fluctuation is induced on the vane it is probable that the wheel turns backwards or you can say the tooth could get underneath in the same time when the pawl is lifted up!.

So, an essential part of the irreversibility of the wheel is damping of the spring. When this happens, the energy that was in the pawl goes to the wheel and shows up as heat. So as it turns, the wheel gets hotter and hotter. This is the reason that this device does not work in perpetual motion. When the vanes get kicked, sometimes the pawl lifts up and goes over the end. But sometimes, when it tries to turn the other way the pawl has already lifted up due to the fluctuations on he wheel side, and the wheel goes back the other way. The net result is zero. When temperature on both sides is equal there won't be any net average motion of the wheel. It will no doubt do a lot of jiggling this way and that way but it would not turn just one way as we wanted.

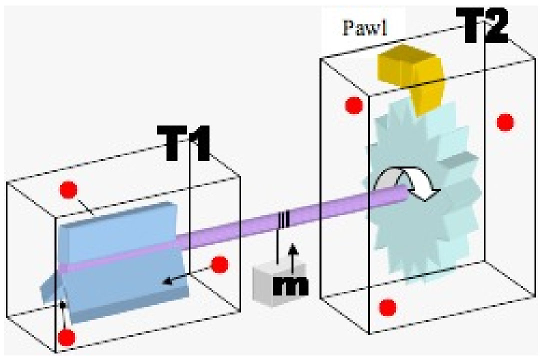

FIG. 2: Rachet and Pawl Device

The reason is simple, it is necessary to do work against the spring to lift the pawl. Let us say the energy required to do so is $'E'$. The chance that the system can accumulate enough energy , $'E'$, to the pawl over the top is $e^{-E/kT}$. Simultaneously, the probability that the pawl will accidently be up is also $e^{-E/kT}$. So the number of times that the pawl is up and the wheel can turn backwards freely is equal to the number of times that we have enough energy to turn it forward when the pawl is down. We thus turned to go nowhere.

Its relation with Demon: If we build a finite sized demon, that the demon may himself get warm cannot see very well after a while. The simplest possible demon would be the one that we have described earlier(trap door held over by a spring). This is just like the ratchet and pawl in another form. Ultimately, the mechanism will heat up just like our system did. If we assume realistically say that the specific heat of the demon is not infinite, it must heat up. So it cannot get rid of the extra heat that it gets from observing the molecules. It would start to shake from Brownian motion so much that it cannot tell whether the molecules are is coming or going. let alone observing their speeds. So it does not work.

LINK OF ENTROPY WITH INFORMATION

Szilard's Engine

Up till now we have seen that the specific measuring techniques are not capable of measuring the velocities of the particles without production of entropy(that overall is greater than or equal to the decrease in entropy). Now we shall consider a quite different argument, first given by Leo Szilard [2], which links entropy with information.

Szilard assumed a being that is capable of making measurements that are superior to our methods and thus the previous arguments about measuring does not apply to this being. What Szilard showed was that when such beings make measurements that might cause a permanent decrease in entropy, the measurement themselves are necessarily accompanied by production of entropy. Note here, that these argument apply not to specific measuring technique but in general any type of measurement.

Szilard defined "measurement" as 'success in coupling a value of a parameter y (for instance the position co-ordinate of a pointer of a measuring instrument) at one moment with the simultaneous value of a fluctuating parameter x of a system, in such a way that, from the value y, we can draw conclusions about the value that x had at that moment'.

We could not fully understand the arguments given by Szilard which showed that such measurement would necessarily produce entropy. Thus, we just mention the result, that yes it does. We mention this because this paper by

Szilard is considered a very important paper with regards to entropy and information.

Landuaer Principle

As we saw in the previous section that the process of measurement generates entropy and thus kills the demon in a sense. R.Landauer [4] further strengthened the argument that the demon can not exist. R.Landuaer argued that whatever the demon might be, it would have finite memory. And since it would have finite memory it would have to erase the existing memory to add in the new measurements. It is in this process of erasing information that additional entropy is generated.

To demonstrate this idea we would use a very simple example. We can safely assume that the information stored in general can be represented by ones and zeros. Now to label a state as one or zero it must be in a stable equilibrium. We can consider a bi-potential well in which we can represent one if the particle is in right well and zero if it is in the left well (see Fig. 3). Further consider that there are many such bi-potential well such that we can store information.

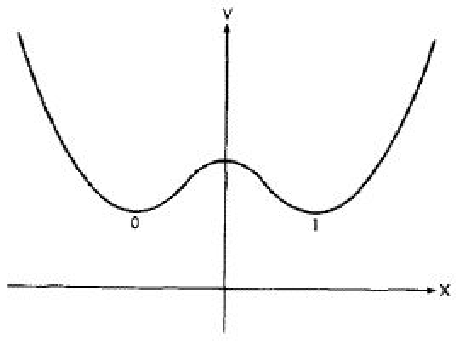

FIG. 3: Bi-stable Potential Well

Now suppose that the initial state was all zeros and then we stored information. Now to store new information we must change some of the bits to zero. To do this we would have to provide the particle with enough energy that it reaches the top of the hill. Now one might think that this energy provided is sufficient for the particle to move from state one to zero. However, it is important to realize that to keep the particle from moving into a new well one must extract energy while the particle goes down. It is this energy extraction (which can be thought of as friction) that causes the increase in entropy.

A detailed calculation which takes bi-potential system as a two-spin system gives us the least value of entropy increase for such a erasing process [4]. We do not go into those calculations since they also involve some knowledge of information theory. However, the concept is same as provided above.

ENGINEERED MAXWELL'S DEMON

In 2014, Zhiyue Lu, Dibyendu Mandal, and Christopher Jarzynski [8] presented a simple model illustrating the operating principles of an information engine, a mechanical device that mimics the behavior of Maxwell's demon by converting heat into information plus work. He aims to realize this model that systematically withdraws energy from a single thermal reservoir and delivers that energy to lift a mass against gravity while writing information to a memory register.

He builds up his model by considering the demon to be a ring, with a blade attached to it (radially inward), which can rotate both clockwise and anti clockwise.

If the demon is mechanical in nature, so must the memory of the demon be. For the memory he assumes a set of paddles that represent the bits 0 or 1 provided that the paddles are oriented to be i range 0 to π or π to 2π respectively.

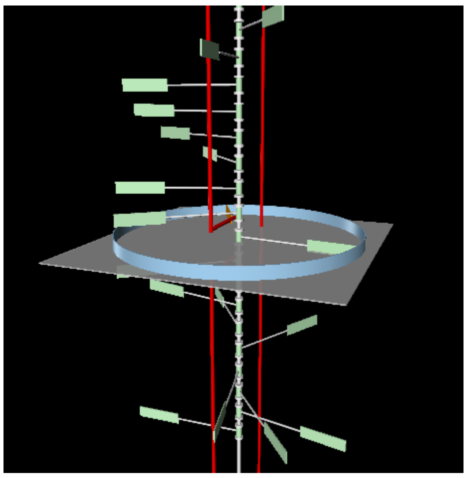

FIG. 4: Engineered Demon

These paddles are attached to an axle which moves without friction in the downward direction though the center of the demon. There are two static vertical bars at $\theta = 0$ and $\theta = \pi$ radians. Note that there is a gap for the paddle to cross $\theta = \pi$ region within the range of the demon. (Fig. 4)

Let the angle of the demon be given by θ_{D} (this angle can go from 0 to 2π). Now, if the angle of the paddles is given by θ_{P} say (this angle can go from 0 to π other than in the range of the demon where it can go from 0 to 2π), then considering the angles to be conventionally counter clockwise, the demon would have a probabilistic net counter clockwise rotation. This is explained as follows:

When the paddle enters the range of the demon, θ_{P} can be anywhere between 0 and π. Consider two possible cases:

- $\theta_{\mathrm{P}} < \theta_{\mathrm{D}}$
  Note that in such a scenario, the blade of the demon can complete a full counter clockwise rotation. Furthermore, θ_{D} can be anywhere between 0 and 2π, the instant the paddle enters the range of the demon.
- $\theta_{\mathrm{P}} > \theta_{\mathrm{D}}$
  Note that in such a scenario, the blade of the demon cannot complete a full counter clockwise rotation. Furthermore, θ_{D} can be anywhere between 0 and π, the instant the paddle enters the range of the demon.

The first case occurs more frequently and thus is more probabilistic than the second case thus the demon has a net biased counter clockwise rotation.

This biased counter clockwise rotation can be used to perform any kind of work for example lifting of a mass with the help of a pulley against gravity.

Now, you must have two questions in your mind, how are the paddles and demon moving. Do note that this setup is placed in an isothermal gas in which particles are undergoing brownian motion. The collision of these gas particles with the paddles makes their angular momentum change and hence makes them rotate. The axle is continuously moving downward at a constant rate. The particles also collide with the blade of the demon and hence are responsible for random movement of the blade with expected value of movement due to particle collision assumed to be zero. This can be achieved by assuming the speeds of all the particles to form a gaussian distribution with mean equal to zero. The blade paddle collision also results in momentum transfer and hence the angular velocities of the demon and the blade exchange after collision.

So far so good. Energy is being consumed from a single thermal reservoir to produce work, but where is the memory being stored? The axle is purposefully made to cross the demon so that according to information of the position and angular velocity of the incoming paddle, the demon would either push it towards the π to 2π region or trap it within the 0 to *pi* region and hance the incoming stream of 0's would be stored in the memory (outgoing stream) as a combination of 0's and 1's hence increasing the entropy of the stream bits.

We have not indulged deep in information entropy (Shannon Entropy) but we do agree with this argument that due to lesser correlation between the outgoing stream of bits with the incoming stream of bits, entropy of the bits has increased. This presented model seems to be violating the 2nd law but it in reality does not violate the 2nd law as due to the increase in information entropy, there is a net entropy increase. At least this is what paper [8] concludes. The paper also discusses the reversibility of this proposed model and argues that if the demon operates in reverse, the incoming stream of random bits would be arranged such that the outgoing stream of bits would be more correlated to the original sequence 000000.

Speaking of the original sequence, this reference sequence can be any sequence. And both Zhiyue Lu and Christopher Jarzynski discuss in detail both Autonomous and Non-Autonomous Programmable Mechanical Maxwell's demon in a paper they recently published in 2019 [9].

We have included the code to the Engineered Maxwell's Demon we wrote in VPython in the Appendix. The links to the almost complete codes (with a small logical error bugging us for days) for Autonomous and Non-Autonomous Programmable Demon are also mentioned in the Appendix.

Shannon Entropy

The formula For Shannon Entropy is:

$$\mathrm{H(X)} = \Sigma \mathrm{p(x)} * \log 2(\mathrm{p(x)}) \tag{12}$$

where H(X) is the entropy associated with random variable X, P(x) is the probability of occurrence of outcome x of variable X, and Log(P(x)) is the information encoded in outcome x of variable X.

Entropy Difference: Cleanness of Demon's Memory

Let our incoming stream be the reference stream of bits to quantify the cleanness of the Demon's Memory. Further let the reference bits be represented with a bar on the bit ($\overline{0}$ and $\overline{1}$). Here our reference sequence is ($\overline{000000}$). The binary state of the incoming bits in general is $b \in \{0,1\}$, this might or might not be the same as the corresponding reference bit $g \in \{\overline{0},\overline{1}\}$. In our case the incoming stream of bits is (000000) and hence it matches the reference sequence.

The cleanness of the incoming bit sequence $(\cdots b_{n-1}, b_n, b_{n+1} \cdots)$ is characterized by the degree to which it matches the fixed reference sequence $(\cdots g_{n-1}, g_n, g_{n+1} \cdots)$. If the binary state of each incoming memory bit matches that of the accompanying gate, i.e., if $b_n = g_n \forall n$, then the memory is considered to be perfectly clean. If the incoming sequence contains mismatches between memory and reference bits, then these mismatches are considered to be impurities that pollute the memory sequence.

Now, let $P_{in}(same)$ denote the fraction of incoming bits that are correctly matched to our reference sequence, ($0\overline{0}$ or $1\overline{1}$). $P_{in}(diff)$ is the fraction that the incoming bits are mismatched with the reference bit, ($0\overline{1}$ or $1\overline{0}$). Quantifying the cleanness of the incoming memory by the excess ratio of clean bits:

$$\delta = P_{\text{in}} \text{ (same)} - P_{\text{in}} \text{ (diff)} \in [-1, +1] \tag{13}$$

Introducing a logical variable L that is the Boolean equality between the states of the bit variable B and the gate variable G (having values l, b, and g respectively):

$$\begin{array}{llllll} l = & \text{true} & \equiv & \text{same} & \text{if} & b = g \\ l = & \text{false} & \equiv & \text{diff} & \text{if} & b \neq g \end{array} \tag{14}$$

The Shannon Entropy, per bit is given by:

$$S_{\mathrm{L,in}} = -\left[P_{\mathrm{in}}(\text{ same }) \log P_{\mathrm{in}}(\text{ same }) + P_{\mathrm{in}}(\text{ diff }) \log P_{\mathrm{in}}(\text{ diff })\right] \in [0, \log 2] \tag{15}$$

For the outgoing bit stream, with similar definition to $P_{out}(same)$ and $P_{out}(diff)$:

$$S_{\mathrm{Lout}} = -\left[P_{\mathrm{out}}(\mathrm{same}) \log P_{\mathrm{out}}(\mathrm{same}) + P_{\mathrm{out}}(\mathrm{diff}) \log P_{\mathrm{out}}(\mathrm{diff})\right] \tag{16}$$

The difference $\Delta S_L = S_{\mathrm{L,out}} - S_{\mathrm{L,in}}$ quantifies the cleanness of the memory sequence, per bit, due to the interactions between the memory bits and ring. In our scenario "Engineered Maxwell's Demon", $P_{in}(diff) = 0$.

CONCLUSION

As we have seen, Maxwell's demon was invented to illustrate the statistical nature of the second law of thermodynamics. It challenged great minds and produced import concepts about entropy. Such as relating entropy with measurement and relating information to thermodynamics. We have not gone into those arguments but the demon has important connections with quantum mechanics and biology as well.

Thus, up till now the demon has not been realized, even in principle, as we saw through the arguments given above. And hence, the second law is safe.

One of the importance of basing our project on Maxwell's demon was that if the demon could exist in principle it would have allowed us to answer the question of irreversibility in nature. The question that all the laws of physics, mechanics, electricity and magnetism are reversible in time so why is it that entropy is not reversible? We haven't found the exact answer to that but we would continue searching for it.

* Electronic address: `ammar.ali.hakeem@gmail.com`
† Electronic address: `javariaghafoor@gmail.com`
‡ Electronic address: `zohranali55@gmail.com`

BIBLIOGRAPHY

[1] Andrew F Rex. *Maxwell's Demon Entropy, Information.* (1990).
[2] Szilard. *On the decrease of entropy in a thermodynamic system by the intervention of intelligent beings.* (1929).
[3] Brillouin. *Maxwell's Demon Cannot Operate: Information and Entropy.* (1951).
[4] Landauer. *Irreversibility and Heat Generation in the Computing Process.* (1961).
[5] Feynamn. *Feynman Lectures Volume 1, chapter 46.*
[6] Motz. *The Doppler demon exorcised.* (1983).
[7] Denur. *The Doppler demon.* (1981).
[8] Zhiyue Lu, Dibyendu Mandal, and Christopher Jarzynski. *Engineering Maxwell's demon.* (2014).
[9] Zhiyue Lu, and Christopher Jarzynski. *A Programmable Mechanical Maxwell's Demon.* (2019).

APPENDIX

VPython Code of the Engineered Maxwell's demon:

```
from random import random
from math import exp, sqrt, pi, floor, cos, sin
from numpy.ma import arange
from vpython import scene, cylinder, color, box, compound, vector, vertex, quad, rate

def gaussian(x, sigma=1.0, meu=0.0):
    return exp(-0.5 * (x - meu / sigma) ** 2) / (sigma * sqrt(2 * pi))

dx = 0.001
N = 100
i = 1
xmax = 3
xlast = -xmax
integral = 0
bins = []
iterno = []

for x in arange(-xmax, dx / 2, dx):
    integral += gaussian(x, xmax/3, 0) * dx     # xmax lies within 3*sigma
    if integral >= i / N:
        k = (x - xlast) / 10
        for j in range(1, 11):
            bins.append(xlast + j * k)
        xlast = x
        i += 1
    iterno.append(x)
# print(len(iter))

# print(bins)
lbin = len(bins)    # 490 entries

for i in range(lbin - 1, -1, -1):  # to make sure average value of bins==meu==0 (adding symmetric values about 0)
    bins.append(-bins[i])

lbin = len(bins)    # 980 entries

def choose():  # returns a number between -3 and 3 (-3 and 3 sigma), gaussian distribution
    r = floor(lbin * random())  # random() gets the next random number in the range [0.0, 1.0).
    return bins[r]

scene.forward = vector(-0.15, -0.3, -1)
scene.range = 1

Dia_axle = 0.03  # diameter of axle
Dia_paddle = 0.05  # diameter of paddle bearing
R = 1  # radius of demon
H = 0.1  # height of demon
dr = R / 100  # thickness of demon cylinder, blade, and paddle
paddle_color = vector(0.8, 1, 0.8)
Len_paddle = 0.4 * R
h = H / 5
d = 1.5 * Dia_paddle

"""
Axes in VPython:
x-axis: right
y-axis: up
z-axis: out
"""

def make_paddle():
    vcyl = cylinder(pos=vector(0, -H / 2, 0), size=vector(H, Dia_paddle, Dia_paddle), axis=vector(0, 1, 0),
                    color=paddle_color, theta=0, omega=0)
    #   size is always set as if our axis is the default axis
    hcyl = cylinder(size=vector(R - dr / 2 - Len_paddle, Dia_paddle / 3, Dia_paddle / 3), color=color.white, theta=0,
                    omega=0)
    #   default pos=(0, 0, 0), axis=(1, 0, 0)
    blade = box(pos=vector(R - dr / 2 - Len_paddle / 2, 0, 0), size=vector(Len_paddle, H, dr), color=paddle_color,
                theta=0, omega=0)
    offset = 0.51 * H
    bearing1 = cylinder(pos=vector(0, offset, 0), size=vector(h, d, d), axis=vector(0, 1, 0), color=color.white,
                        theta=0, omega=0)
    bearing2 = cylinder(pos=vector(0, -offset, 0), size=vector(h, d, d), axis=vector(0, -1, 0), color=color.white,
                        theta=0, omega=0)
    paddle = compound([vcyl, hcyl, blade, bearing1, bearing2])
    paddle.rotate(angle=pi / 2, axis=vector(0, 1, 0), origin=vector(0, 0, 0))
    paddle.theta = 0
    paddle.omega = 0
    return paddle

def make_demon():
    # VPython 6 has an extrusion object unlike VPython 7 (which we're using)
    n = 4 * 90
    dtheta = 2 * pi / n
    theta = 0
    demon_color = vector(0.67, 0.86, 1)
    quads = []
    # subscript i: inside, o: outside
    vilower = []
    viupper = []
    volower = []
    voupper = []
    vibottom = []
    vitop = []
    vobottom = []
    votop = []
    while theta < 2*pi-dtheta/2:
        ray1i = R * vector(cos(theta), 0, sin(theta))
        ray2i = R * vector(cos(theta + dtheta), 0, sin(theta + dtheta))
        ray1o = (R + dr) * vector(cos(theta), 0, sin(theta))
        ray2o = (R + dr) * vector(cos(theta + dtheta), 0, sin(theta + dtheta))

        vilower.append(vertex(pos=ray1i + vector(0, -0.5 * H, 0), normal=-ray1i, color=demon_color))
        viupper.append(vertex(pos=ray1i + vector(0, 0.5 * H, 0), normal=-ray1i, color=demon_color))
        volower.append(vertex(pos=ray2o + vector(0, -0.5 * H, 0), normal=ray1o, color=demon_color))
        voupper.append(vertex(pos=ray2o + vector(0, 0.5 * H, 0), normal=ray1o, color=demon_color))

        vibottom.append(vertex(pos=ray1i + vector(0, -0.5 * H, 0), normal=vector(0, -1, 0), color=demon_color))
        vitop.append(vertex(pos=ray1i + vector(0, 0.5 * H, 0), normal=vector(0, 1, 0), color=demon_color))
        vobottom.append(vertex(pos=ray2o + vector(0, -0.5 * H, 0), normal=vector(0, -1, 0), color=demon_color))
        votop.append(vertex(pos=ray2o + vector(0, 0.5 * H, 0), normal=vector(0, 1, 0), color=demon_color))

        theta += dtheta
    vilower.append(vilower[0])
    viupper.append(viupper[0])
    volower.append(volower[0])
    voupper.append(voupper[0])
    vibottom.append(vibottom[0])
    vitop.append(vitop[0])
    vobottom.append(vobottom[0])
    votop.append(votop[0])
    for n in range(n):
        quads.append(quad(v0=vilower[n], v1=viupper[n], v2=viupper[n + 1], v3=vilower[n + 1]))
        quads.append(quad(v0=volower[n], v3=voupper[n], v2=voupper[n + 1], v1=volower[n + 1]))
        quads.append(quad(v0=vibottom[n], v1=vobottom[n], v2=vobottom[n + 1], v3=vibottom[n + 1]))
        quads.append(quad(v0=vitop[n], v1=votop[n], v2=votop[n + 1], v3=vitop[n + 1]))
    quads.append(box(pos=vector(0, 0, -(R - Len_paddle / 2)), size=vector(dr, H, Len_paddle), color=color.orange))
    box(pos=vector(0, -H / 2 - dr / 2, 0), size=vector(2.2 * R, dr, 2.2 * R), color=color.white, opacity=0.7)
    obj = compound(quads)
    obj.omega = 0
    obj.theta = 0
    return obj

demon = make_demon()
Demon_theta = [3 * pi / 2]      # x-z plane
demon_theta0 = Demon_theta[0]
demon_omega0 = 0
demon.theta = demon_theta0
demon.omega = demon_omega0
demon.rotate(angle=demon.theta, axis=vector(0, 1, 0), origin=vector(0, 0, 0))

# make the rods and axle

c = color.gray(0.9)
w = 0.05 * R
rod = cylinder(pos=vector(0, -4, -0.55 * R), radius=Dia_axle / 2, axis=vector(0, 8, 0), color=color.red)
rodupperV = cylinder(pos=vector(0, 1.5 * H, 0.55 * R), radius=Dia_axle / 2, axis=vector(0, 4, 0), color=color.red)
rodupperH = cylinder(pos=vector(0, 1.5 * H + Dia_axle / 2, 0.55 * R), radius=Dia_axle / 2, axis=vector(0, 0, -0.45 * R),
                     color=color.red)
rodlowerV = cylinder(pos=vector(0, -1.5 * H, 0.55 * R), radius=Dia_axle / 2, axis=vector(0, -4, 0), color=color.red)
rodlowerH = cylinder(pos=vector(0, -1.5 * H - Dia_axle / 2, 0.55 * R), radius=Dia_axle / 2,
                     axis=vector(0, 0, -0.45 * R), color=color.red)
axle = cylinder(pos=vector(0, -4, 0), radius=Dia_axle / 2, axis=vector(0, 8, 0), color=color.white)

def true_angle(a):  # convert an angle to be between 0 and 2pi
    a %= 2 * pi
    if a < 0:
        a += 2 * pi
    Demon_theta.append(a)
    return a

dt = 0.01

def displace_paddle(paddle, deltat):
    dtheta = paddle.omega * deltat
    if dtheta == 0:
        return
    thetai = paddle.theta
    thetaf = thetai + dtheta
    if thetaf >= 2 * pi or thetaf <= 0:
        # if collision with rodVo
        paddle.omega = -paddle.omega         # direction swap
        if dtheta > 0:
            thetaf = 2 * pi - (thetaf - 2 * pi)
        else:
            thetaf = -thetaf
    elif (thetai <= pi <= thetaf or thetaf <= pi <= thetai) and not (-H < paddle.pos.y < H):
        # if collision with rodVpi
        paddle.omega = -paddle.omega         # direction swap
        if dtheta > 0:
            thetaf = pi - (thetaf - pi)
        else:
            thetaf = pi + (pi - thetaf)
    paddle.rotate(angle=thetaf - thetai, axis=vector(0, 1, 0), origin=vector(0, 0, 0))
    paddle.theta = thetaf

def displace_demon(deltat):
    dtheta = demon.omega * deltat
    if dtheta == 0:
        return
    thetai = demon.theta
    thetaf = thetai + dtheta
    demon.rotate(angle=thetaf - thetai, axis=vector(0, 1, 0), origin=vector(0, 0, 0))
    demon.theta = thetaf

paddles = []
pdy = 2 * H     # distance between paddles
Npaddles = 20   # no of paddles
theta0 = pi / 2
omega0 = 0
maxpaddley = 3 * H + Npaddles * pdy

for y in arange(1.5 * H, maxpaddley + pdy / 2, pdy):  # create a set of paddles
    P = make_paddle()
    P.pos.y = y
    P.theta = theta0
    P.omega = omega0
    P.rotate(angle=P.theta, axis=vector(0, 1, 0), origin=vector(0, 0, 0))
    paddles.append(P)

def hit():  # use a Gaussian distribution ranging from -3 to 3 to hit a paddle or blade
    blow = 0
    if random() <= 0.5:
        blow = choose()
    return 2 * blow

gamma = 0.03  # http=//en.m.wikipedia.org/wiki/Langevin_dynamics
Ld = 10
kld = sqrt(2 * gamma * Ld)

def update_omega(obj):
    term = kld * hit()
    obj.omega += -gamma * obj.omega + term      # special case assumption

dy = H / 200  # how much to drop the axle for each iteration
t = 0

while True:
    rate(50)  # clamp to no more than 50 iterations per second
    collision = False
    for p in paddles:
        update = True
        if -H < p.pos.y < H:
            if p.omega != demon.omega:
                deltat = (true_angle(demon.theta) - p.theta) / (p.omega - demon.omega)
                if 0 <= deltat <= dt:  # there will be a collision
                    collision = True
                    displace_demon(deltat)
                    omega = demon.omega  # swap angular velocities
                    demon.omega = p.omega
                    displace_demon(dt - deltat)
                    update_omega(demon)
                    displace_paddle(p, deltat)
                    p.omega = omega
                    displace_paddle(p, dt - deltat)
                    update_omega(p)
                    update = False
        if update:
            displace_paddle(p, dt)
            update_omega(p)

        p.pos.y -= dy
        if p.pos.y < -(Npaddles / 2) * pdy:  # move a paddle from the bottom to the top
            p.pos.y += (1 + Npaddles + 1) * pdy
            p.axis = vector(-sin(p.theta), 0, -cos(p.theta))
            p.theta = theta0
            p.omega = 0
    if not collision:
        displace_demon(dt)
        update_omega(demon)
    t += dt
```

Links to Programmable Mechanical Maxwell's Demon VPython Code:

1. https://github.com/Javaria-Ghafoor/Maxwells-Demon/blob/master/Programmable%20Maxwell's%20Demon/DemonAutonomous.py
2. https://github.com/Javaria-Ghafoor/Maxwells-Demon/blob/master/Programmable%20Maxwell's%20Demon/DemonNonAutonomous.py