Schrodinger equation using Monte Carlo

Issar Amro

Density Matrix.

The density matrix is defined as

$$\rho(x, x', \beta) = \sum_n \psi_n(x) e^{-\beta E_n} \psi_n^*(x')$$

For a free particle, i.e. a particle that is not subject to a potential, the Hamiltonian operator is given by, taking $\frac{\hbar^2}{m} = 1$,

$$H_f \psi = -\frac{1}{2} \frac{\partial^2 \psi}{\partial x^2} = E\psi$$

Which is the time-independent Schrodinger equation, an eigenvalue problem whose solution represents the wavefunction of the particle.

Putting the particle in a box of length L with periodic boundary conditions, we would expect the solution to be periodic functions, namely cosines and sines. The solution, naturally, can also be a complex exponential. Since the Schrodinger equation is linear, the total solution would be a superposition of normalized sines and cosines solutions where

$$\psi_c = \sqrt{\frac{2}{L}} \cos\left(\frac{2n\pi}{L} x\right)$$

$$\psi_s = \sqrt{\frac{2}{L}} \sin\left(\frac{2n\pi}{L} x\right)$$

Or a superposition of the normalized complex solution

$$\psi_i = \sqrt{\frac{1}{L}} e^{\frac{i2n\pi}{L} x} \; \forall n \in (-\infty, \infty)$$

Applying the free Hamiltonian operator on any of these solutions, we get

$$E_n = \frac{2n^2\pi^2}{L^2}$$

Using the complex solution, we can now plug in these values in the free particle density matrix

$$\rho(x,x',\beta) = \frac{1}{L}\sum_{-\infty}^{\infty} e^{\frac{i2n\pi}{L}(x-x')} e^{-\frac{2\beta n^2 \pi 2}{L^2}}$$

Using the change of variable y = $\frac{2n\pi}{L}$, the sum becomes

$$\rho(x,x',\beta) = \frac{1}{2\pi}\sum_{y} e^{iy(x-x')} e^{-\frac{\beta}{2}y^2}$$

Taking the sum to the continuous limit, we get

$$\rho(x,x',\beta) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{iy(x-x')} e^{-\frac{\beta}{2}y^2}\, dy$$

Which is Gaussian integral that gives us the final form of the normalized density matrix that we will use

$$\rho(x,x',\beta) = \sqrt{\frac{1}{2\pi\beta}}\, e^{-\frac{(x-x')^2}{2\beta}}$$

If we introduce a potential, the density matrix becomes

$$\rho(x,x',\beta)_f = e^{-\frac{\beta V(x)}{2}}\ \rho(x,x',\beta)_f\ e^{-\frac{\beta V(x)}{2}}$$

Which can be proved by Taylor expanding the exponentials and getting $\rho(x,x',\beta) = e^{-\beta H}$ where $H = H_f + V(x)$.

Problem 1.

procedure matrix-square
input $\{x_0,\ldots,x_K\}, \{\rho(x_k,x_l,\beta)\}$ (grid with step size Δ_x)
for $x = x_0,\ldots,x_K$ **do**
  { **for** $x' = x_0,\ldots,x_K$ **do**
    { $\rho(x,x',2\beta) \leftarrow \sum_k \Delta_x \rho(x,x_k,\beta)\,\rho(x_k,x',\beta)$
output $\{\rho(x_k,x_l,2\beta)\}$

Algorithm 3.3 matrix-square. Density matrix at temperature $1/(2\beta)$ obtained from that at $1/\beta$ by discretizing the integral in eqn (3.32).

We first implement a function that returns the free density matrix for given 2 arrays x and x′ and β

```
def density(x1,x2,beta):
    n = len(x1)
```

```
    matrix = np.zeros((n, n))
    for i in range(n):
        for j in range(n):
            matrix[i,j] = x1[i] - x2[j]
    return 1/(np.sqrt(2*np.pi*beta)) * np.exp(-(matrix**2)/(2*beta))
```

Now, we implement a function that takes 1 array and use the free density function for x = x′ and accounts for a potential of the form $V(x) = \frac{1}{2}kx^2$ that is associated with a harmonic oscillator

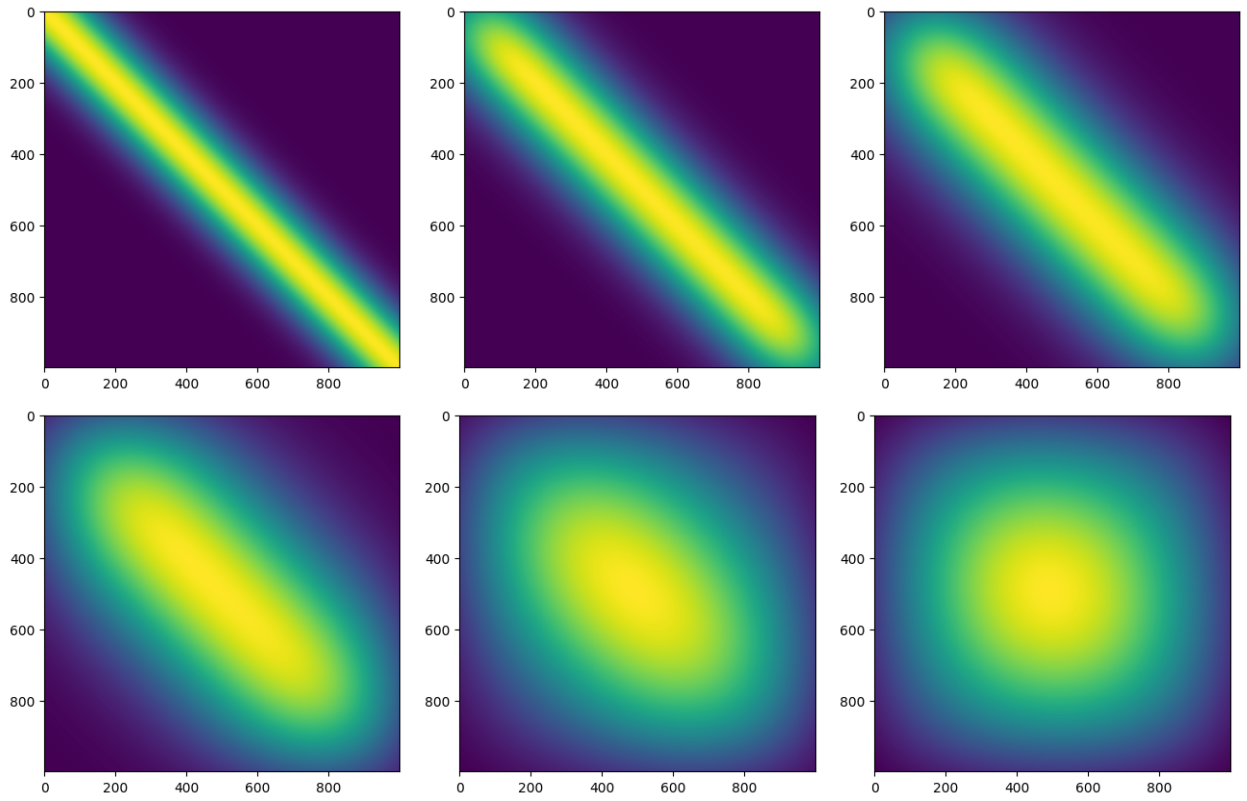
```
def harmonic(x,beta,k):
    return np.exp(-(1/2)*k*x) * density(x,x,beta) * np.exp(-0.5*k*x)
```

We introduce the array x and define β and k

```
x = np.linspace(0,1,1000)
beta = 0.01
k = 0.005
dx = x[1]-x[0]
h = harmonic(x,beta,k)
```

Taking the step size (dx) to be the distance between 2 consecutive x values, we watch how the system evolves by plotting 6 images of the density matrix of the harmonic oscillator, each while modifying the density (h) to h = dx*(h@h). At each iteration, we are performing the matrix squaring multiplied by the step size

```
for i in range(5):
    plt.imshow(h)
    plt.show()
    h = dx*(h@h)
plt.imshow(h)
plt.show()
```

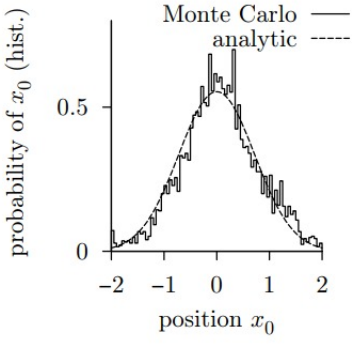


We see how the system eventually converges to a 2D projection of a Gaussian where the concentration in the middle corresponds to the highest probabilities.

Problem 2.

procedure `naive-harmonic-path`
input $\{x_0, \ldots, x_{N-1}\}$
$\Delta_\tau \leftarrow \beta/N$
$k \leftarrow \texttt{nran}\,(0, N-1)$
$k_\pm \leftarrow k \pm 1$
if $(k_- = -1) k_- \leftarrow N$
$x'_k \leftarrow x_k + \texttt{ran}\,(-\delta, \delta)$
$\pi_a \leftarrow \rho^{\text{free}}\left(x_{k_-}, x_k, \Delta_\tau\right) \rho^{\text{free}}\left(x_k, x_{k_+}, \Delta_\tau\right) \exp\left(-\frac{1}{2}\Delta_\tau x_k^2\right)$
$\pi_b \leftarrow \rho^{\text{free}}\left(x_{k_-}, x'_k, \Delta_\tau\right) \rho^{\text{free}}\left(x'_k, x_{k_+}, \Delta_\tau\right) \exp\left(-\frac{1}{2}\Delta_\tau x_k'^2\right)$
$\Upsilon \leftarrow \pi_b/\pi_a$
if $(\texttt{ran}\,(0,1) < \Upsilon) x_k \leftarrow x'_k$
output $\{x_0, \ldots, x_{N-1}\}$

Algorithm 3.4 `naive-harmonic-path`. Markov-chain sampling of paths contributing to $Z^{\text{h.o.}} = \int \mathrm{d}x_0\, \rho^{\text{h.o.}}(x_0, x_0, \beta)$.



We define the free density matrix function but this time only for 2 values (x_0, x_1)

```
def rho(x0,x1,beta):
     return 1/(np.sqrt(2*np.pi*beta)) * np.exp(-(x1-x0)**2/(2*beta))
```

We then define the random walk function. The time step is defined by $\frac{\beta}{n}$. We define k as a random number ranging over the indices until n-2, and we introduce a perturbation on the kth element of x. We now need to check whether this perturbed point should be accepted. For that, we introduce $k_+ = k+1$ and $k_- = k-1$ and calculate the existence probabilities associated with the new positions, which are defined by

$$\pi_a = \rho_f(x_{k_-}, x_k, dt)\ \rho_f(x_k, x_{k_+}, dt)\ e^{-0.5dtx_k^2}$$

$$\pi_b = \rho_f(x_{k_-}, x'_k, dt)\ \rho_f(x'_k, x_{k_+}, dt)\ e^{-0.5dtx_k'^2}$$

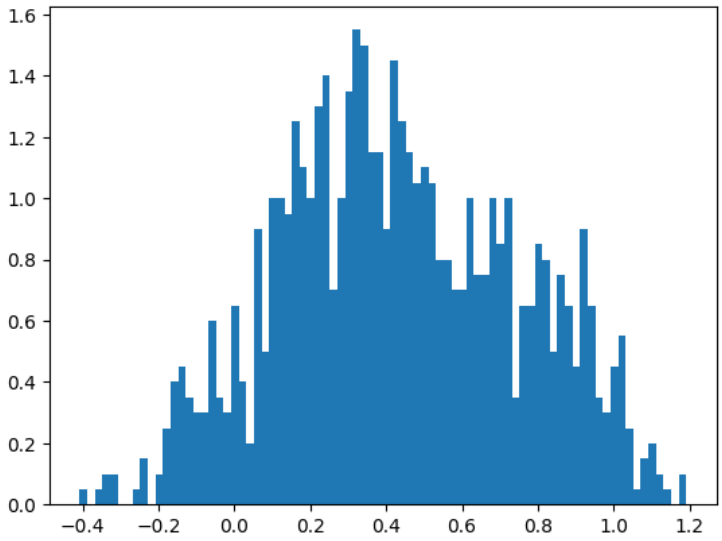
Where x'_k is the perturbed new point. The condition is that if a random measurement from a uniform distribution on (0,1) is less than the ratio $\frac{\pi_b}{\pi_a}$, then the new/perturbed point is accepted and the kth element is replaced by the perturbed x_k in the x array.

```
def randomWalk(x,beta):
    d=0.1
    n = len(x)
    dt = beta/n
    k = np.random.randint(0, n-2)
    xkp = x[k] + np.random.uniform(-d, d)
    kplus = k+1
    kminus = k-1
    if kminus == -1:
        kminus = n-1
    pia = rho(x[kminus], x[k], dt)*rho(x[k],x[kplus],dt)*np.exp(-0.5*dt*x[k]**2)
    pib = rho(x[kminus], xkp, dt)*rho(xkp, x[kplus],dt)*np.exp(-0.5*dt*xkp**2)
    if np.random.uniform(0,1)<pib/pia:
        x[k] = xkp
    return x
```

We now introduce the array x and define β, then we go over a million iterations for k values and apply the random walk function for each random k value.

```
x = np.linspace(0,1,1000)
beta = 4
for i in tqdm(range(1_000_000)):
    x = randomWalk(x,beta)
```

Finally, we plot the histogram given by the updated values of the x array, getting



Which is a Gaussian, something we expect the solution to the Schrodinger equation to look like.