Variational Monte Carlo method

Ilaria Carpi

Importance Sampling

- The expectation value of some function $h(x)$ can be calculated using an importance sampling probability distribution function (PDF) $f(x)$:

$$\langle h \rangle = \int h(x)dx = \int f(x)\left[\frac{h(x)}{f(x)}\right]dx$$

- The resulting expectation value is more precise than without importance sampling

Markov Chains and Metropolis Algorithm

- A Markov chain is a discrete set of states $\{s_1, \dots, s_N\}$ with a rule that defines the transition probability between states
- To generate this set of random numbers from a known PDF we use the Metropolis algorithm:
  - Generate a starting state s_i with probability P_i
  - Generate a random step s_j
  - Metropolis question:
    - If $P_j > P_i$, the transition is accepted
    - If $P_j < P_i$, we evaluate the state with probability P_j/P_i, meaning that:
      - If $\frac{P_j}{P_i} > rand()$, the transition is accepted
      - If $\frac{P_j}{P_i} < rand()$, the transition is rejected and the state s_i becomes the next step in the Markov chain
- Thermalization: while carrying out a random walk starting from s_i, we can choose to only collect the state s_{i+n}, discarding the intermediate steps. This minimizes the correlations among successive states

Markov Chains and Metropolis Algorithm

```
xOld = step * (np.random.rand() - 0.5)
list_x.append(xOld)

for k in range(0,NX):
  xNew = xOld + step*(np.random.rand()-0.5)
  if ((WF(xNew,BestPar)/WF(xOld,BestPar))**2>1 or (WF(xNew,BestPar)/WF(xOld,BestPar))**2 > np.random.rand() ):
    xOld = xNew
    list_x.append(xOld)
    NAcc = NAcc +1
  else:
    xOld = xOld
    list_x.append(xOld)
```

An example of Metropolis algorithm to generate a random set of coordinates. The probability is given by the 1D harmonic oscillator wave function

Variational Principle

- Given a Hamiltonian H and a trial wave function Ψ, an upper bound to the ground state energy is

$$\langle H \rangle = \frac{\int \Psi^* H \Psi dx}{\int |\Psi|\text{^}2 dx}$$

- We can define a PDF and a local energy:

$$f(x) = \frac{|\Psi(\mathrm{x})|^2}{\int |\Psi(x)|^2 dx} \qquad E_L(x) = \frac{1}{\Psi(x)} H\, \Psi(x)$$

- The expectation value becomes

$$\langle H \rangle = \int f(x) E_L(x) dx$$

Variational Monte Carlo

- Consider $f(x)$ as the importance sampling distribution function
- The values of $E_L(x)$ in the integral of $\langle H \rangle$ are weighted by the Metropolis algorithm
- We use the variational Monte Carlo method in order to find the best parameters to represent the 1D harmonic oscillator and the He4 ground state wave functions
- The ground state is found by searching the energy minimum, varying the parameters of a chosen trial wave function

Variational Monte Carlo

- To overcome fluctuations around the minimum we use a technique called correlated sampling: assume a fixed parameter β (or set) near the minimum, and compute the energy for a random parameter α with the following equality:

$$E_\alpha = \frac{\int |\Psi(\beta,x)|^2 \left[\frac{|\Psi(\alpha,x)|^2}{|\Psi(\beta,x)|^2} E_L(\alpha,x)\right] dx}{\int |\Psi(\beta,x)|^2 \left[\frac{|\Psi(\alpha,x)|^2}{|\Psi(\beta,x)|^2}\right] dx}$$

- This way, the importance sampling function is fixed and is $|\Psi(\beta,x)|^2$.

1D Harmonic Oscillator

- Hamiltonian:

$$H = \frac{1}{2}\left(-\frac{d^2}{dx^2} + x^2\right)$$

- Trial wave function:

$$\Psi(\alpha, x) = e^{-\alpha^2 x^2/2}$$

- Local energy:

$$E_L(\alpha, x) = \frac{H\Psi(x)}{\Psi(x)} = \frac{1}{2}[\alpha^2 + x^2(1 - \alpha^4)]$$

- We want to find the minimum of:

$$\bar{E}_{gs} = \int_{-\infty}^{+\infty} |\Psi|^2 E_L(x) dx$$

1D Harmonic Oscillator

- Start by defining the local energy and the wave function, and by initializing the necessary parameters:

```
NAcc = 0
NP = 900
NX = 10000
step = 1.0
step1 = 0.3

# parameter variables for VMC
BestPar = 1
Par = 0.4
Par_new = 0

np.random.seed(12231)

# wave function
def WF(x,alpha):
  return exp(-0.5*alpha*alpha*x*x)

# local energy
def LocalEnergy(x,alpha):
  return 0.5*x*x*(1-alpha**4) + 0.5*alpha*alpha
```

- Then, find a set of steps with the Metropolis algorithm, by using $|\Psi(\alpha, x)|^2 = e^{-\alpha^2 x^2}$ to define the probabilities for each step x_i

```
xOld = step * (np.random.rand() - 0.5)
list_x.append(xOld)

for k in range(0,NX):
  xNew = xOld + step*(np.random.rand()-0.5)
  if ((WF(xNew,BestPar)/WF(xOld,BestPar))**2>1 or (WF(xNew,BestPar)/WF(xOld,BestPar))**2 > np.random.rand() ):
    xOld = xNew
    list_x.append(xOld)
    NAcc = NAcc +1
  else:
    xOld = xOld
    list_x.append(xOld)
```

To construct the Markov chain, we use the value of α in the minimum, which is $\alpha = 1$

1D Harmonic Oscillator

- Then, start the Markov chain to find the energy minimum, starting from $\alpha = 0.4$ (the expected value in the minimum is $\alpha = 1$)
- Algorithm:
  - Generate each value of α as a random step

```
Par_new = Par + (np.random.rand()-0.5)*step1
```

  - Calculate the energy and variance using correlated sampling

```
for j in range(0,NX):
  E_loc = E_loc + LocalEnergy(list_x[j],Par_new)*(WF(list_x[j],Par_new)/WF(list_x[j],BestPar))**2
  E_loc_2 = E_loc_2 +(LocalEnergy(list_x[j],Par_new)*(WF(list_x[j],Par_new)/WF(list_x[j],BestPar))**2)**2
  D = D + (WF(list_x[j], Par_new)/WF(list_x[j], BestPar))**2

E_mean = E_loc/D
Var2 = (E_loc_2/D - (E_loc/D)**2)/D
```

1D Harmonic Oscillator

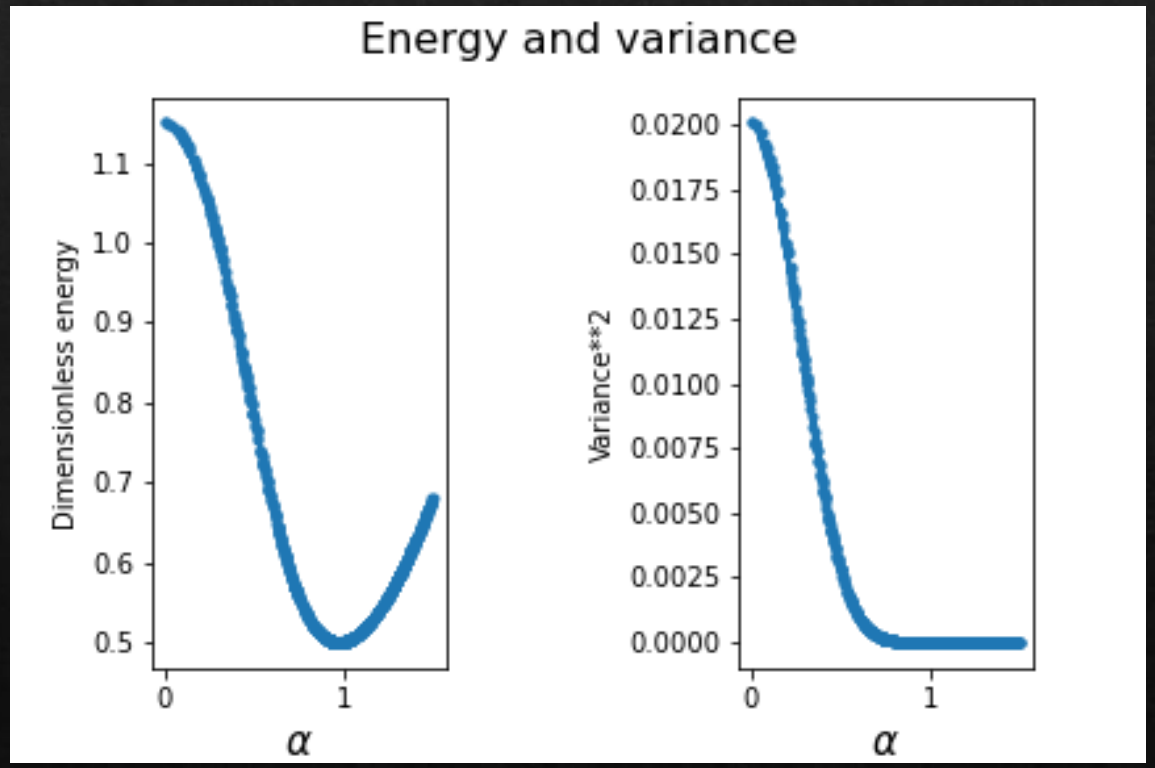
- Ask the Metropolis question:
  - If $\bar{E}_{gs}(\alpha_{old}) > \bar{E}_{gs}(\alpha_{new})$, the move is accepted; if instead $\bar{E}_{gs}(\alpha_{old}) > \bar{E}_{gs}(\alpha_{new})$, we evaluate the step α_{new} using as probability $\frac{\bar{E}_{gs}(\alpha_{new})}{\bar{E}_{gs}(\alpha_{old})}$
  - We also store the value of the minimum energy and update it in every cycle

```
if (list_Eall[l-1]>E_mean or list_Eall[l-1]/E_mean>np.random.rand()):

  if (minimumE>E_mean):
    minimumE = E_mean
    minimumAlpha = Par
    minimumVariance2 = Var2
  else:
    minimumE = minimumE
    minimumAlpha = minimumAlpha
    minimumVariance2 = minimumVariance2

  Par = Par_new
  list_alpha.append(Par)
  list_Eall.append(E_mean)
  list_E.append(E_mean)
  list_variance2.append(Var2)

else:
  Par = Par
  list_Eall.append(list_Eall[l-1])
```

1D Harmonic Oscillator: results

- From these graphs we see that the minimum energy is around $\alpha = 1$, and the value at the minimum is approximately $\bar{E}_{gs} = 0.5$

- The variance gets smaller as the value of α approaches the minimum



Ground State of Helium – 4

- Hamiltonian:

$$H = -\frac{h^2}{2m}\frac{d^2}{d\boldsymbol{r}^2} + V(r_1, \dots, r_4)$$

- Trial wave function:

$$\Psi(r_1, \dots, r_4, \gamma, a, \beta) = \prod_{i<j}^{4} \left(e^{-\gamma\, r_{ij}^2} + a\, e^{-(\beta+\gamma) r_{ij}^2} \right)$$

- Local energy:

$$E_L(r_1, \dots, r_4, \gamma, a, \beta) = -\frac{h^2}{2m}\left(\frac{d^2\Psi}{d\boldsymbol{r}^2}\right)\frac{1}{\Psi} + V(r_1, \dots, r_4)$$

Ground State of Helium – 4

- The potential is an effective potential that reproduces the spin-isospin-independent interaction between two nucleons at low energies [3]:

$$V(r) = 1000\, e^{-3\, r^2} - 163.35\, e^{-1.05\, r^2} - 21.5\, e^{-0.6\, r^2} - 83\, e^{-0.8\, r^2} - 11.5\, e^{-0.4\, r^2}$$

- The trial wave function, with 3 free parameters, is a Jastrow correlation factor multiplying an uncorrelated state:

$$\Psi(r_1, \dots, r_4) = \prod_{i<j}^{4} \left(1 + a\, e^{-\beta r_{ij}^2}\right) e^{-\alpha^2 (r_1^2 + \cdots + r_4)/2}$$

which can be written as the product:

$$\prod_{i<j}^{4} g(r_{ij}) = \prod_{i<j}^{4} \left(e^{-\gamma\, r_{ij}^2} + a\, e^{-(\beta+\gamma) r_{ij}^2}\right)$$

where $\gamma = \alpha^2/8$

Ground State of Helium – 4

- Wave function →
  - The parameters are $\mathrm{Par}[0] = \gamma$, $\mathrm{Par}[1] = \mathrm{a}$, $\mathrm{Par}[2] = \beta$
  - The function $d(R, i, j)$ is the square distance between particles
- Potential →
- Kinetic energy →
  - The second-order differential in the kinetic energy is approximated with a centered second difference, and the constant is $C = \frac{h^2}{2m} \simeq 20.72$

```
def d(R,i,j):
  return np.sum((R[:, i] - R[:, j])**2)

def WF(R, Par):
  wf = 1
  for i in range(0,A-1):
    for j in range(i+1,A):
      wf = wf * (exp(-Par[0]*d(R,i,j))+Par[1]*exp(-Par[2]*d(R,i,j)))
  return wf

def V(R):
  V = 0
  for i in range(0,A-1):
    for j in range(i+1,A):
      V = V + 1000*exp(-3*d(R,i,j))-165.35*exp(-1.05*d(R,i,j))
      -21.5*exp(-0.6*d(R,i,j))
      -83*exp(-0.8*d(R,i,j))-11.5*exp(-0.4*d(R,i,j))
  return V

def Diff (R, i, j, h, Par):
  R_P = R.copy()
  R_M = R.copy()
  R_P[i,j] = R[i,j]+h
  R_M[i,j] = R[i,j]-h
  return (WF(R_P, Par)+WF(R_M, Par)-2*WF(R, Par))/h**2

def K(R, Par):
  K = 0
  for j in range(0,A):
    for i in range(0,3):
      K = K + Diff(R, i, j, h, Par )*C
  return K
```

Ground State of Helium – 4

- As the final goal for parameters, we take the values from ref. [1]

```
BestPars = [0.08597 , -0.7191 , 2.13796 ]
```

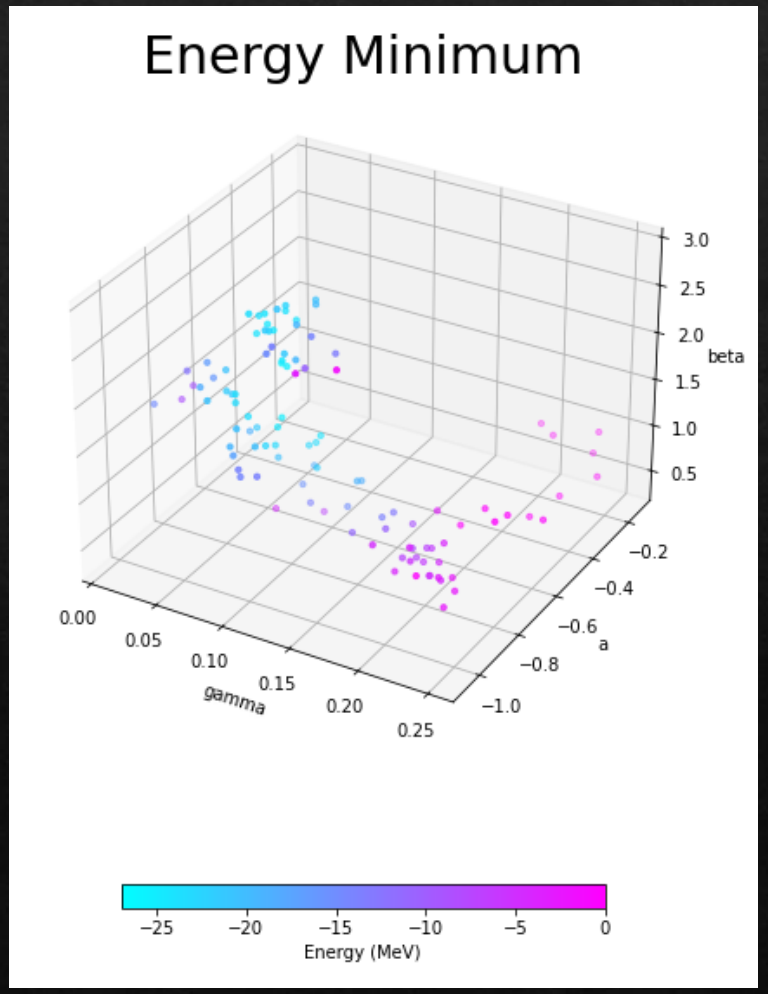
- The code structure is the same as in the 1d harmonic oscillator case, with the difference that now we are varying 3 parameters:

```
Par_new[0] = Par[0].copy() + (np.random.rand()-0.5)*step1
Par_new[1] = Par[1].copy() + (np.random.rand()-0.5)*step2
Par_new[2] = Par[2].copy() + (np.random.rand()-0.5)*step3
```

Ground State of Helium – 4: results

- Varying the parameter steps and the number of iterations of the VMC cycle, the results are visibly different

- The starting parameters that optimize the precision and the running time are:
  - step1 = 10/100
  - step2 = 25/100
  - Step3 = 50/100



References

1. R. Guardiola, "Monte Carlo Methods in Quantum Many Body Theories", 1998, DOI: 10.1007/BFb0104529
2. R. F. Bishop, E. Buendia, M. F. Flynn and R. Guardiola, "Diffusion Monte Carlo Determination of the Binding Energy of the 4He Nucleus for Model Wigner Potentials", 1992, DOI: 10.1088/0954-3899/18/2/002
3. I. R. Afnan, Y. C. Tang, "Investigation of Nuclear Three- and Four-Body Systems with Soft-Core Nucleon-Nucleon Potentials", 1968, DOI: 10.1103/PhysRev.175.1337
4. A. Gezerlis, "Numerical Methods in Physics with Python", ch. 7, 2020, DOI: 10.1017/9781108772310