(*CODE EXAMPLE FOR N1_OQC*)

```
(*Import the PDB file as a list of strings*)
pdbData = Import["DIRECTORY\\OQCs_files_codes\\N1.pdb", "Lines"];

(*Select only lines that start with "ATOM"*)
atomLines = Select[pdbData, StringStartsQ[#, "ATOM"] &];

(*Filter for only Carbon (C) atoms*)
carbonLines = Select[atomLines, StringTrim[StringTake[#, {14, 16}]] == "C" &];

(*Function to extract XYZ coordinates (columns 7,8,and 9 in standard PDB format)*)
extractXYZ[line_] := Module[{values}, values = StringSplit[line];
   ToExpression[values[[6 ;; 8]]]];
(*Apply extraction function to all Carbon ATOM lines*)
pts = extractXYZ /@ carbonLines;
(*Output the XYZ coordinates of Carbon atoms*)


COORDS = Sort[Round[Transpose[{pts[[All, 3]], pts[[All, 1]]}], 0.01]];
translated = Table[
   {COORDS[[i]][[1]] + Max[COORDS[[All, 1]]] - 5.5, COORDS[[i]][[2]]}, {i, 1, Length[COORDS]}];

linterne = Drop[TakeSmallest[
    DeleteDuplicates[Round[Flatten[DistanceMatrix[COORDS]], 0.01]], 300], 1];
lexterne =
  Sort[DeleteDuplicates[Round[Flatten[DistanceMatrix[COORDS, translated]], 0.01]]];
(*Show the result*)
distlist = TakeWhile[linterne, # < 3.7 &]
distlistphases = TakeWhile[lexterne, # < 3.7 &]

Export["COORDS_N1.csv", COORDS, "CSV"]
```

```
Export["translated_N1.csv", translated, "CSV"]
```
exporta

```
Show[ListPlot[COORDS, Frame → True, Axes → False, PlotMarkers → {Automatic, 0.02}],
```
mos·· gráfico de uma lista d·· quadro verd··· eixos falso marcadores do grá· automático

```
 ListPlot[translated, Frame → True, Axes → False, PlotStyle → Red,
```
gráfico de uma lista de val··· quadro verd··· eixos falso estilo do gráf··· vermelho

```
  PlotMarkers → {Automatic, 0.02}], PlotRange → All, AspectRatio → .2, ImageSize → 800]
```
marcadores do grá· automático intervalo do g·· tudo quociente de aspecto tamanho da imagem

```
ClearAll[kx]
```
apaga tudo

```
B = Table[0, {i, 1, Length[COORDS]}, {j, 1, Length[COORDS]}];
```
tabela comprimento comprimento

```
A = Table[0, {i, 1, Length[COORDS]}, {j, 1, Length[COORDS]}];
```
tabela comprimento comprimento

```
 (*PARAMETERS*)
t1 = -3.3; (*Hopping parameter*)
β = 3.6; (*Decay parameter*)
onsite = -0.27; (*Onsite energy*)
tol = 0.01;
```

```
Do[
```
repete

```
  If[
```
se

```
   Abs[EuclideanDistance[COORDS[[i]], COORDS[[j]]] - o] ≤ tol,
```
val·· distância euclidiana

```
   ndist = SetAccuracy[EuclideanDistance[COORDS[[i]], COORDS[[j]]], 3];
```
define exatidão distância euclidiana

```
   B[[i, j]] = t1 * Exp[β * (1 - Round[ndist / distlist[[1]], 0.01])];
```
exponencial arredondamento

```
  ], {o, distlist}, {i, 1, Length[COORDS]}, {j, 1, Length[COORDS]}];
```
comprimento comprimento

```
Do[
```
repete

```
  If[
```
se

```
   Abs[EuclideanDistance[COORDS[[i]], translated[[j]]] - o] ≤ tol,
```
val·· distância euclidiana

```
   ndist = SetAccuracy[EuclideanDistance[COORDS[[i]], translated[[j]]], 3];
```
define exatidão distância euclidiana

```
A⟦i, j⟧ =

  t1 * Exp[β * (1 - Round[ndist / distlist⟦1⟧, 0.01])] * Exp[I * kx];
       exponencial  arredondamento                        ex⋯  unidade imaginária

A⟦j, i⟧ =

  t1 * Exp[β * (1 - Round[ndist / distlist⟦1⟧, 0.01])] * Exp[- I * kx];
       exponencial  arredondamento                        exp⋯  unidade imaginária


], {o, distlistphases}, {i, 1, Length[COORDS]}, {j, 1, Length[translated]}];
                                  comprimento                   comprimento

H = A + B + IdentityMatrix[Length[A]] * onsite;
            matriz identidade  comprimento


hamiltonian[kx_] = Table[H⟦i, j⟧, {i, 1, Length[COORDS]}, {j, 1, Length[COORDS]}];
                   tabela               comprimento                comprimento
```