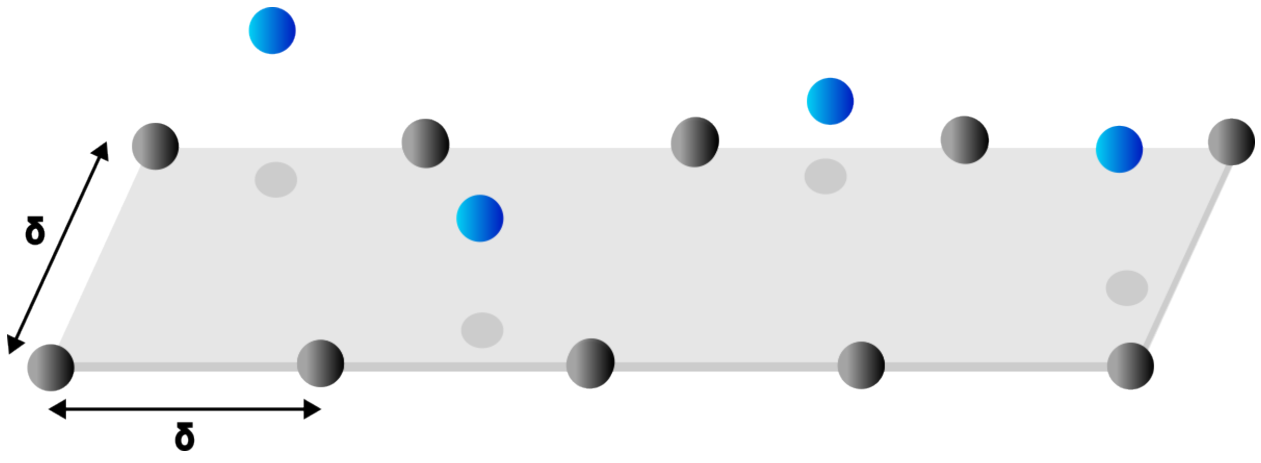
Fortran Pool
— Technical Report—

By Erwan Le Doeuff
M1 CompuPhys
23/02/2025



« Building of a multidimensional potential energy surface applied to the gas/surface reactivity »

Supervised by Ludovic Martin-Gondre

Table of contents

Introduction

The adsorption of gas-phase particles onto a surface is a fundamental process widely studied in surface physics and chemistry. It plays a key role in applications such a pollutant capture and catalysis, where surfaces help speed up chemical reactions.

Understanding how nitrogen gas interacts with a tungsten surface is important for controlling surface properties and optimizing catalytic efficiency. These interactions, driven by electronic effects, can be described using potential energy surfaces (PES), which map out how the energy of the system varies with atomic positions.

In this work, we aim to develop a set of tools to generate these energy landscapes using simple physical models, providing a practical way to study gas-surface interactions.

1 User Manual

1.1 Goal of the program

The purpose of this program is to generate the potential energy surface (PES) of adsorption sites for nitrogen gas on a tungsten surface (referred to as N-W interactions). The program is divided into three parts.

The first part is the **3DPES** software, which generates the PES by iterating over (X, Y, Z) coordinates — (X, Y) being bounded to the unit cell. However, due to technical limitations discussed in the following sections, this approach is not always practical. To address this, we provide two additional programs, **2DCUT** and **1DCUT**, designed to generate specific subsets of data more efficiently.

Additionally, we include various Python scripts for data visualization.

1.2 Input data

In the following sections, (X, Y, Z) are considered real numbers, while `Npts` is a positive integer.

1.2.1 3DPES

The **3DPES** software requires the number of points, `Npts`, as input before performing the iteration over the unit cell. The grid is constructed by stepping in the X and Y direction with increments of `1/Npts`, ensuring a uniform sampling of the surface.

The Z coordinate, which represents the distance between the nitrogen atom and the tungsten surface is constrained to the range (0,10) Å. This limitation ensures that the computation focuses on physically revelant adsorption positions, avoiding unreasonable large distances where the physical model has no more meaning.

1.2.2 2DCUT

The **2DCUT** software generates its own data similarly to **3DPES**, but instead of iterating over the (X, Y, Z) space, it keeps the Z coordinate fixed at a user-defined value. Upon launching the program the user is prompted to enter the number of points, `Npts`, which determines the resultion in the X and Y directions, as well as the Z coordinate in angstroms. This allows for the construction of a two-dimensional energy surface at a specified height above the tungsten surface

1.2.3 1DCUT

The **1DCUT** program generates its own data by prompting the user to specify (X, Y) coordinates, the number of points `Npts` and the desired range for Z in angstroms. This allows for a one-dimensional analysis of the potential energy surface along the Z direction at a fixed position on the tungsten surface.

1.3 Output data and visualization tools

1.3.1 3DPES

The program generates a file named ‘`3dpes.txt`‘, which contains the computed potential energy surface data. Each line in this file follows the format,

```
X Y Z V(X,Y,Z)
```

where $V(X, Y, Z)$ represents the potential energy at the given (X, Y, Z) coordinates.

Since this file is often quite large (see the **limitations section**), it is not well-suited for quick visualization. Therefore, we strongly recommend using the **2DCUT** and **1DCUT** programs for more efficient data extraction and analysis.

1.3.2 2DCUT

This program is responsible for generating a ‘`2dcut.txt`‘ file, which contains the computed potential energy surface as described in **section 1.2.2**.

The first line of the file specifies the number of points used for the computation, as well as Z coordinate at which the computation was performed. Each subsequent lines follows the format,

```
X Y V(X,Y,Z)
```

Two complementary Python scripts are provided for visualization.

The `2dmap.py` script visualizes the data as a 3D surface plot. It reads the `2dcut.txt` file, extracts the (X, Y, V) values, and generates a triangular surface plot. The colormap magma is used for better contrat and the axes are labeled accordingly.

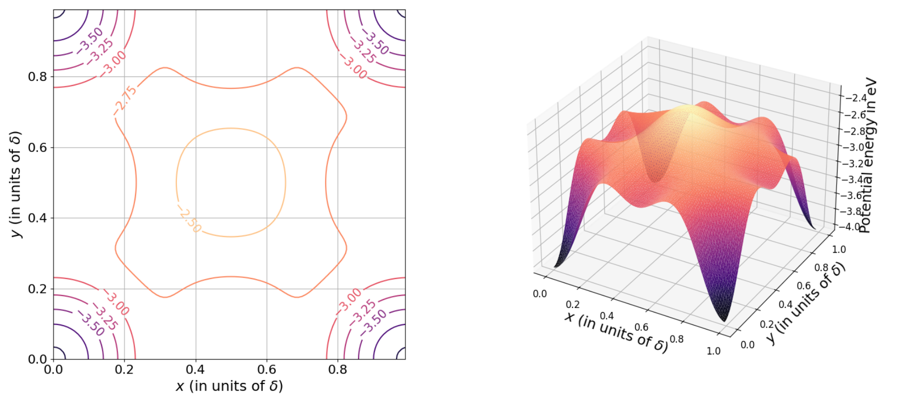


Figure. Illustration. (Left) A typical contour plot produced by `2dcontour.py`. (Right) The 3D map generated by `2dmap.py`.

The `2dcontour.py` script provides a contour plot representation of the potential energy surface. Unlike the 3D surface plot, this method highlights regions of equal potential energy, making it easier to identify energy minima and adsorption sites at a glance. The script reads the `2dcut.txt`, extracts the (X, Y, V) values and generates the contour lines. The contours are then labeled for better readability, and the magma colormap is used to enhance the contrast.

1.3.3 1DCUT

The **1DCUT** software computes the potential energy along the Z direction and stores the result in a file ‘`1dcut.txt`‘. The first line of the file serves as a header, specifying the (X, Y) coordinates of the selected site and the number of points used for the computation. Each subsequent lines are in the format,

```
Z V(X,Y,Z)
```

The script `1dcut.py` is used to plot the data.

1.4 Limitations

All real values are stored as double-precision floating-point numbers (8 bytes per value). For a large number of points, this can result in significant memory usage, both in volatile memory (RAM) and persistent storage (disk space). If `Npts` is set too high, the computational load may exceed your system's capabilities leading to slowdowns or failures.

To ensure execution, it is recommand to keep `Npts` at a reasonable value. Additionally, the spatial parameters should respect the following constraints:

- (X, Y) should remain within the range [0,1], corresponding to the unit cell.
- Z must be positive and not excessively large, as high values may correspond to unphysical adsorption states.

For more technical details, please refer to the developer's reference.

2 Developer's reference

2.1 Physical models

2.1.1 Base model

Several models have been proposed to describe atomic interactions. A commonly used and convenient approach is the Morse potential, defined as,

$$V(Z) = D\left[\exp\{-2\alpha(Z - r^{\text{eq}})\} - 2\exp\{-\alpha(Z - r^{\text{eq}})\}\right] \qquad (1)$$

where

- D (in eV) is the depth of the potential well, representing the bond dissociation energy. It quantifies the strength of the interaction, with larger values indicating stronger bonding.

- α (in Å^{-1}) is a stiffness parameter, controlling how rapidly the potential changes with distance. A high α leads to a steeper potential well.

- r^{eq} (in Å) is the equilibrium distance, at which the potential reaches its minimum and the interaction force is zero.

- Z (in Å) represents the distance between the nitrogen atom and the tungsten surface.

The Morse potential (represented in **Fig. 1**) is particularly useful for modeling adsorption interactions, as it captures both the attractive and repulsive components of the interaction in a physically meaningful way.

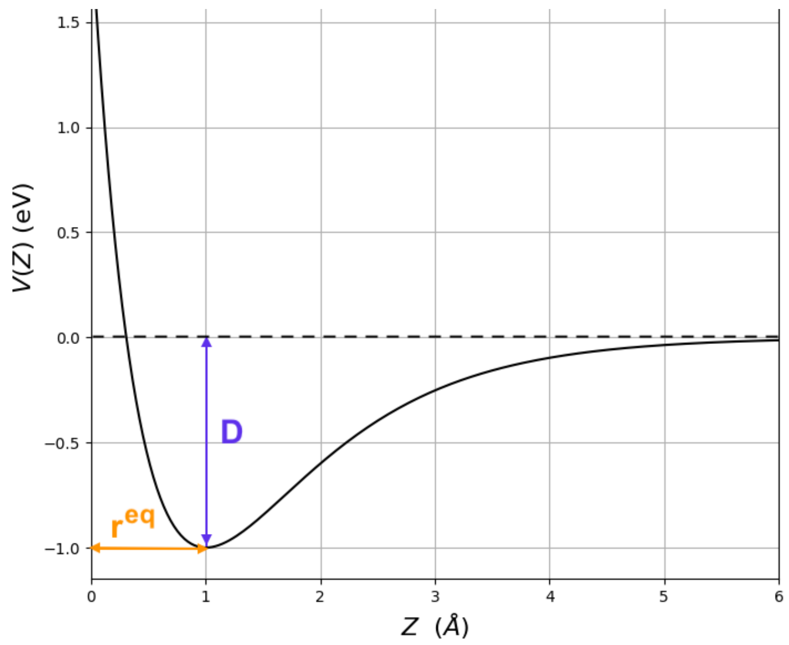


Figure 1. Morse potential with the parameters $D = 1\,\text{eV}$, $r^{eq} = 1\,\text{Å}$ and $\alpha = 1\,\text{Å}^{-1}$. Both the depth of the potential well and the equilibrium distance are reported on the figure.

The tungsten surface is modeled as the (100) plane of a body-centered cubic (BCC) crystal composed of monoatomic tungsten. This choice provides a well-defined and symmetric adsorption surface, where the atomic arrangement follows a square lattice structure with a lattice parameter δ.

Sites	(X,Y) **in units of** δ
Top	$(0,0)$
Bridge	$(\frac{1}{2},0),(0,\frac{1}{2})$
Hollow	$(\frac{1}{2},\frac{1}{2})$

Sites (abbrev.)	(X,Y) **in units of** δ
Top-Bridge *(TB)*	$(\frac{1}{4},0),(0,\frac{1}{4})$
Top-Hollow *(TH)*	$(\frac{1}{4},\frac{1}{4})$
Bridge-Hollow *(BH)*	$(\frac{1}{4},\frac{1}{2}),(\frac{1}{2},\frac{1}{4})$

Table 1. Symmetry sites locations on the (100) plane.

Symmetry sites on the (100) plane of the body-centered cubic crystal structure, where δ is the lattice parameter. We also introduce a shorthand notation for low-symmetry sites.

The (100) plane in a BCC crystal exhibits a layered atomic configuration and contains nine symmetry sites, classified as follows:

- High-symmetry sites: Top, Bridge and Hollow.
- Low-symmetry sites: Top-Bridge, Top-Hollow and Bridge-Hollow.

Each site involving a bridge is counted twice due to the symmetry of the squre lattice in both the *X* and *Y* directions. This means that the Bridge, Top-Bridge and Bridge-Hollow sites effectively appear in duplicate within the unit cell. For a visual description of these adsorption sites see **Fig. 2** and for their respective locations, refer to **Table 1**. These sites play a crucial role in determining adsorption energy landscapes and diffusion pathways for nitrogen atoms on the tungsten surface.

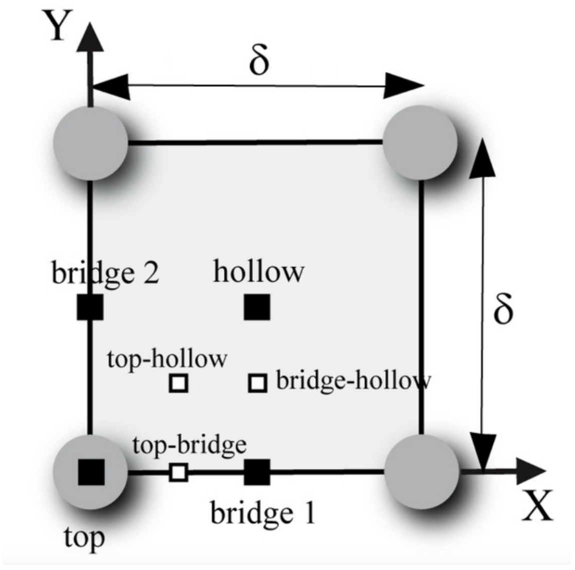


Figure 2. Symmetry sites on the (100) plane of the body-centered cubic crystal structure. A tungsten atom is sitting on the Top site. Adapted from the *Numerical Project* assignment sheet.

The arrangement of tungsten atoms on the surface introduces corrugation, meaning that the adsorption potential – modeled by the Morse potential – is not uniform across the surface. Instead, the potential energy energy varies depending on the adsorption site, implying that the Morse potential must depend on X and Y. Furthermore, due to the periodic nature of the crystal, this dependence must also be periodic, ensuring that the potential accurately reflects the repeating structure of the lattice.

This periodic dependence is introduced by the use of a Fourier expansion of Morse potential's parameters, up to the fifth order. For a parameter $P(X,Y)$, we can write,

$$\begin{aligned}
P(X,Y) &= P_0 + P_1(\cos\{2\pi X\} + \cos\{2\pi Y\}) \\
&+ P_2(\cos\{2\pi(X+Y)\} + \cos\{2\pi(X-Y)\}) \\
&+ P_3(\cos\{4\pi X\} + \cos\{4\pi Y\}) \\
&+ P_4(\cos\{2\pi(2X+Y)\} + \cos\{2\pi(X+2Y)\} + \cos\{2\pi(2X-Y)\} + \cos\{2\pi(X-2Y)\}) \\
&+ P_5(\cos\{4\pi(X+Y)\} + \cos\{4\pi(X-Y)\})
\end{aligned} \tag{2}$$

where X and Y are given in units of δ. Using the location of the symmetry sites, one can formulate a system of linear equations to determine the Fourier coefficients P_i, provided that the parameters are known at different (X,Y) positions.

Finally, taking the dependencies into account, the Morse potential is written,

$$\begin{aligned}
V(X,Y,Z) &= D(X,Y)\big[\exp\{-2\alpha(X,Y)(Z - r^{\mathrm{eq}}(X,Y))\} \\
&- 2\exp\{-\alpha(X,Y)(Z - r^{\mathrm{eq}}(X,Y))\big]
\end{aligned} \tag{3}$$

2.1.2 Further refinements

The model described in the previous section can be refined by adjusting the stiffness of the potential $\alpha(X,Y)$. By introducing a dependence in Z, we can have more control on the steepness of the potential for attractive and repulsive interactions,

$$\begin{aligned}
\alpha(X,Y,Z) &= (1 - f(X,Y,Z))(\alpha^0_{\mathrm{rep}}(X,Y) + \alpha^1_{\mathrm{rep}}(X,Y) \times Z) \\
&+ f(X,Y,Z)(\alpha^0_{\mathrm{att}}(X,Y) + \alpha^1_{\mathrm{att}}(X,Y) \times Z)
\end{aligned} \tag{4}$$

$$f(X,Y,Z) = \frac{1}{2}(1 + \tanh\{a(X,Y) \times Z + b(X,Y)\})$$

with $\alpha^i_{\mathrm{rep/att}}$, a and b being new parameters that we need to Fourier expand. We call the base model plus these refinements the **LEPS model**.

2.2 Algorithms

2.2.1 Fitting procedure

To solve the linear systems mentioned in **section 2.1.1**, the values of the coefficients at different sites must be known. Fortunately, the positions of the symmetry sites significantly simplify the expression of the system derived from expression (2). Moreover, we have access to CRP — a competing method to the one presented in this document — computation data, which we will use for fitting.

These CRP data provide the potential energy at a given site for different Z values, allowing us to reconstruct the potential described by expression (1). The data are extracted using a Python script with the `NumPy` and `SciPy` libraries. The fitting process is performed using the `scipy.optimize.curve_fit` function.

Initial parameters estimates can be made on the physical model described in **section 2.1.1**. Additionally, constraints can be applied, such as requiring D to be positive or ensuring that a and b remain within reasonable bounds, as the hyperbolic tangent function is effectively saturated for absolute values greater than approximately 3.

Once the fit is performed on every high- and low-symmetry sites, Fourier coefficients can be determined.

2.2.2 Fourier coefficients

The system can be expressed, for a given parameter P, as,

$$\begin{pmatrix} A_{00} & \cdots & A_{05} \\ \vdots & \ddots & \vdots \\ A_{50} & \cdots & A_{55} \end{pmatrix} \begin{pmatrix} P_0 \\ \vdots \\ P_5 \end{pmatrix} = \begin{pmatrix} P^{\mathrm{Top}} \\ \vdots \\ P^{\mathrm{BH}} \end{pmatrix}$$

which, using a common algebraic notation, simplifies to

$$Ax = B \tag{5}$$

Such system is often numerically solved by applying the so-called LU decomposition method. In this approach, the matrix A is expressed as the composition of a lower triangular matrix L and an upper triangular matrix U, such that,

$$A = \mathrm{LU}$$

This decomposition simplifies the solution of linear systems by breaking into two steps: first, solving for an intermediate vector through forward substitution in eqn (5),

$$\begin{aligned} \mathrm{LU}x &= B \\ \Leftrightarrow Ly &= B \end{aligned}$$

and then using backward substitution to determine x,

$$Ux = y$$

This method is particularly advantageous from a numerical standpoint. While decomposing the matrix A is an $O(n^3)$ operation, it only needs to be performed once, since A depends solely on (X, Y) and not on the parameter being considered.

As a result, the LU decomposition can be reused multiple times, for each parameter, without the need to recompute it. The subsequent substitution steps — forward and backward substitution — are only $O(n^2)$ operations, in contrast to the traditionnal Gaussian-pivot, which remains $O(n^3)$ for every new system.

Another advantage is that this algorithm has been well-established for a long time, and highly optimized implementations for both LU decomposition and substitution can be found in libraries such as LAPACK. These implementations are efficiently designed and rigorously tested, making them reliable for numerical computations. For this reason, we opted to use them in our approach.

However, to simplify its usage and as the matrix A is very small, we have written a small Fortran module that abstracts away the sometimes complex preparations required to call the LAPACK routines. This module handles the necessary setup, allowing for a more seamless and user-friendly implementation.

```fortran
subroutine solve(N, A, B)
   integer, intent(in) :: N
   double precision, dimension(N,N), intent(inout) :: A
   double precision, dimension(N), intent(inout) :: B

   integer, dimension(N) :: IPIV
   integer :: info

   ! compute the LU factorization
   call dgetrf(N, N, A, N, IPIV, info)
   call handle_error(info, "LU factorization misses")

   ! solve the system Ax=B
   ! stores the result in B
   call dgetrs('N', N, 1, A, N, IPIV, B, N, info)
   call handle_error(info, "Solving linear equations misses")
end subroutine solve
```

This subroutine takes 3 arguments: the integer N, which is the dimension of the A matrix, an array of double-precision floating-point number A and the vector B. The LU decomposition is stored in A and the computed solutions to P_i are stored in B. Subroutines `dgetrf` and `dgetrs` belong to LAPACK.

In practice, the user simply needs to define the matrix A and the vector B for each parameter, where B contains the fitted values for each site. Once these are set up, the user can call the solve subroutine, which will perform the LU decomposition and solve the system.

```
use math, only : solve
integer, parameter :: N = 6
double precision, dimension(N,N) :: A
double precision, dimension(N) :: BD, BREQ

BD = [4.833511, 6.377278, 7.382040, 5.149473, 5.394620,
6.817161]

BREQ = [1.816304, 1.142394, 0.621643, 1.575456, 1.384302,
0.791984]

call init(N, A)
call solve(N, A, BD)
print *, "D"
call disp(N, BD)

call init(N, A)
call solve(N, A, BREQ)
print *, "REQ"
call disp(N, BREQ)
```

Here, `init` and `disp` are convenient subroutines designed to, respectively, reinsert the coefficients into the A matrix and display the results in a compact and readable format to the user.

2.2.3 Mapping the surface

Once Fourier coefficients are determined, mapping the potential energy surface is pretty straightforward. We are iterating over unconstrained coordinates with a given step and compute the corresponding Morse potential expression (3). The whole is done in Fortran. For illustration, here is the iterative loop for the **3DPES**,[1]

1. We also compute the minimum global minimum in the **3DPES** software by continuously tracking the smallest value of the `Morse` function during the computation.

```
loopX: do i=1, Npts
     X = (i-1) * stepXY
     loopY: do j=1, Npts
        Y = (j-1) * stepXY
        loopZ: do k=1, Npts
           Z = (k-1) * stepZ
           PES(i,j,k) = Morse(X, Y, Z)
           if(Vmin > PES(i,j,k)) then
              Vmin = PES(i,j,k)
              Xmin = X
              Ymin = Y
              Zmin = Z
           end if
        end do loopZ
     end do loopY
  end do loopX
```

The function Morse is the expression (3),

```
double precision function Morse(X, Y, Z)
   double precision, intent(in) :: X, Y, Z
   Morse = exp(- 2 * alpha_mod(X,Y,Z) * (Z - req(X,Y)))
   Morse = Morse - 2 * exp( - alpha_mod(X,Y,Z) * (Z - req(X,Y)))
   Morse = D(X,Y) * Morse
 end function Morse
```

where all parameters are util functions keeping track of the Fourier expansion. For example, here is *D* and the corresponding Fourier expansion up to the fifth order,

```
function Fourier(Fp, X, Y)
    double precision, dimension(6), intent(in) :: Fp
    double precision, intent(in) :: X, Y
    double precision :: Fourier

    Fourier = Fp(1) + Fp(2) * (cos(2 * pi * X) + cos(2 * pi
* Y)) &
         + Fp(3) * (cos(2 * pi * (X + Y)) + cos(2 * pi * (X
- Y))) &
         + Fp(4) * (cos(4 * pi * X) + cos(4 * pi * Y)) &
        + Fp(5) * (cos(2 * pi * (2 * X + Y)) + cos(2 * pi *
(X + 2 * Y)) &
        + cos(2 * pi * (2 * X - Y)) + cos(2 * pi * (X - 2 *
Y))) &
         + Fp(6) * (cos(4 * pi * (X + Y)) + cos(4 * pi * (X
- Y)))
end function Fourier


double precision function D(X, Y)
  double precision, intent(in) :: X, Y
  D = Fourier(FD, X, Y)
end function D
```

The dummy variable FD contains a reference to an array containing *D*'s Fourier coefficients. Finally, the map is stored in a plain text file, each software following the according format described in **section 1.3**.

2.3 Internals

2.3.1 File architecture

The project is split in two versions. The first, named α-version, truncates some approximations made by the model described in **section 2.1**. It does implement Fourier expansion up to second order — fit of the parameters are only done on high-symmetry sites – and does not include the refinements proposed in **section 2.1.2**. The other version, called β-version, is a complete implementation of the physical model.

We report the following file structure,

- **alpha** — contains files for the α-version.
  - **CRP-LEPS** – contains plots and scripts comparing both methods.
  - **fit-CRP** — contains python scripts to fit coefficients on relevant symmetry sites.
  - **fortran-scripts** — contains Fortran code.
    - **3d-script** — contains source code for **3DPES, 2DCUT** and **1DCUT** software, along with visualization tools.
    - **fourier-coefficients** — contains source code for solving linear systems and determine Fourier coefficients.
- **beta** — contains files for the β-version.
  - **CRP-LEPS** – contains plots and scripts comparing both methods.
  - **fit-CRP** — contains python scripts to fit coefficients on relevant symmetry sites.
  - **fortran-scripts** — contains Fortran code.
    - **3d-script** — contains source code for **3DPES, 2DCUT** and **1DCUT** software, along with visualization tools.
    - **fourier-coefficients** — contains source code for solving linear systems and determine Fourier coefficients.
- **rsrc** — contains illustrations for this technical report.

2.3.2 Building the project

The compiler `gfortran` is needed in order to build the different parts of the project. Each directory within the fortran-scripts folder contains a `Makefile` for building the software. The **3DPES**, **2DCUT** and **1DCUT** softwares can be built separately using the following commands,

```
make 3dpes

make 2dcut

make 1dcut
```

The Fourier coefficient computation program has a `Makefile` that users may need to modify. Specifically, the `math` module — described in **section 2.2.2** — relies on the LAPACK library. Since LAPACK must be installed on the system, the corresponding arguments in the `Makefile` need to be adjusted accordingly. To install LAPACK, please refer to `https://www.netlib.org/lapack`.

We succesfully built the software on macOS Sequoia 15.3.1. On macOS, LAPACK linking is done using the `-framework Accelerate` flag, whereas on Linux, it should be replaced with `-llapack -L/path/to/lapack`.

The visualization tools require `Python3` along with the `Matplotlib, Numpy` and `SciPy` libraries to be installed. Those can be installed using the `pip` package manager.

3 Quality tests

3.1 User inputs

The **3DPES**, **2DCUT** and **1DCUT** software rely on user inputs. Therefore, it is essential to ensure the validity of these inputs. To enforce this, we impose the following restrictions,

3DPES

```
Npts > 1
```

2DCUT

```
Npts > 1
Z >= 0
```

1DCUT

```
Npts > 1
1 >= X >= 0
1 >= Y >= 0
Zmin >= 0
Zmax >= Zmin
```

We validate these inputs using do while loops, in the following manner,

```
X = -1.
do while ( X < 0. .OR. X > 1. )
  print *, "X (1 >= X= > 0)"
  read *, X
end do
```

3.2 Solving a linear system

We aim to verify that the the code for the linear system of Fourier coefficients produces reliable result. We therefore propose to solve a system that can be easily solved by hand,

$$\begin{pmatrix} 3 & 1 \\ 2 & 5 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} 15 \\ 36 \end{pmatrix}$$

yielding $\alpha = 3$ and $\beta = 6$. We use the `solve` function we implemented in the `math` module,

```
program main
  use math, only : solve
  implicit none

  double precision, dimension(2,2) :: A
  double precision, dimension(2) :: B

  A = reshape([3., 2., 1., 5.], [2,2])
  B = [15., 36.]
  call solve(2, A, B)

  print *, B
end program main
```

The results matches $\alpha = 3.0$ and $\beta = 6.0$ we calculated by hand.

3.3 Comparison with the CRP model

α-version	**ME**	**MAD**
Top	12.19	32.69
Bridge	-0.16	0.59
Hollow	-0.08	0.21

β-version	**ME**	**MAD**
Top	-0.24	0.62
Bridge	0.00(2)	0.01
Hollow	0.00(2)	0.01

α-version	**ME**	**MAD**
TB	-5.52	5.94
TH	-1.42	1.66
BH	0.03	0.40

β-version	**ME**	**MAD**
TB	0.02	0.07
TH	0.03	0.05
BH	0.00(2)	0.01

Table 2. Mean Error (ME) and Mean Absolute Deviation (MAD) for different symmetry sites. Values expressed in eV and rounded to two decimal places.

CRP data have been the foundation of our programs, as they allowed us to determine the Morse potential parameters by performing a fit on the different symmetry sites. The α-version exhibited very approximate predictions, as shown in **Fig. 3**, whereas refinements in the β-version resulted in significantly improved predictions quality.

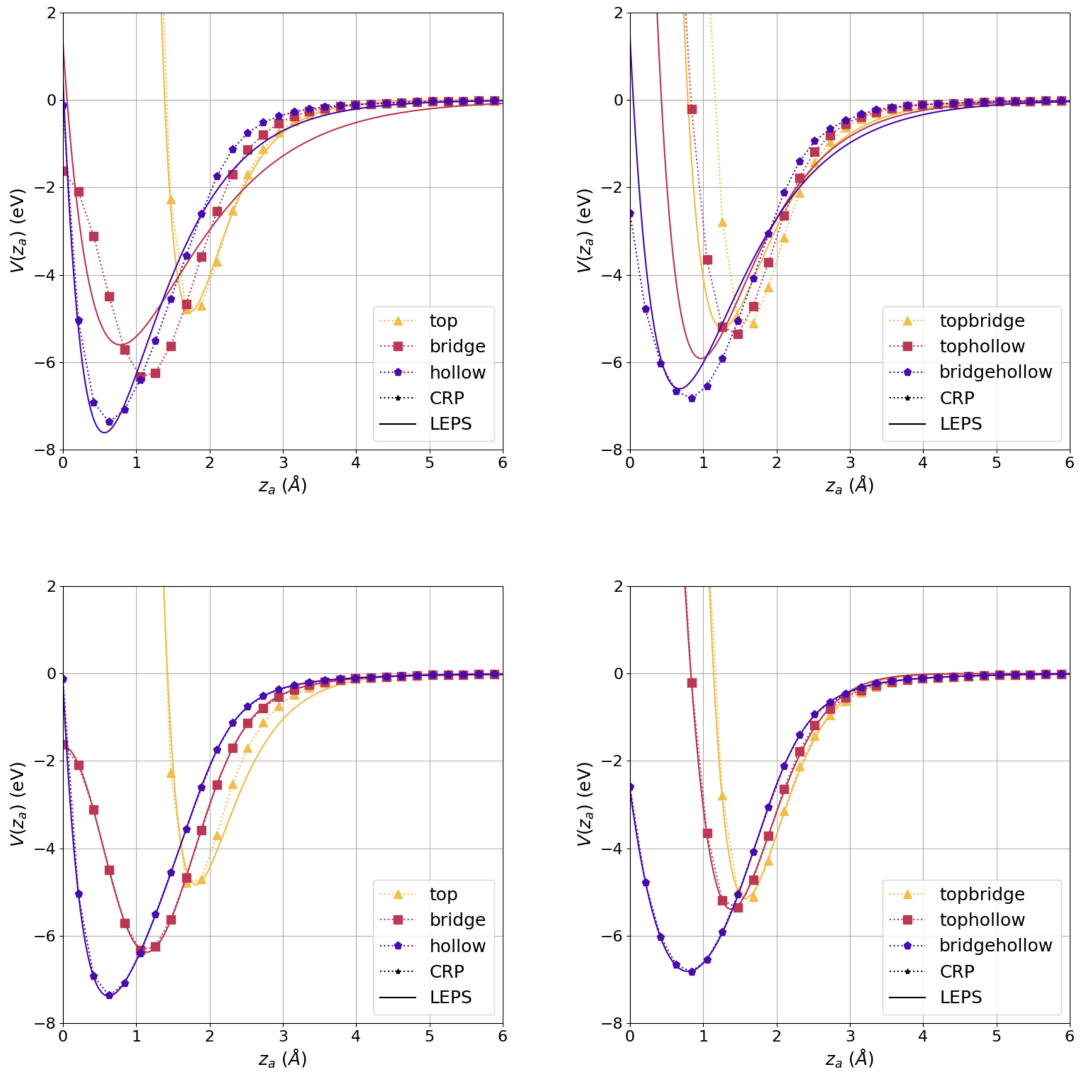


Figure 3. Comparison of CRP Data with LEPS model predictions for symmetry sites : α-version (top) vs. β-version (bottom).

The coordinate z_a is the distance between the nitrogen atom and the surface. We can see better agreements between β-version and CRP data than for the α-version.

To quantify qualitative observations, we computed the mean error (ME), defined as

$$\text{ME} = \text{avg}(\text{LEPS} - \text{CRP})$$

and the mean absolute deviation (MAD), given by,

$$\text{MAD} = \text{avg}(|\text{LEPS} - \text{CRP}|)$$

on the different symmetry sites. The results are presented in **Table 2.** We estimated that the β-version achieves on average a 100-fold improvement in agreement with CRP data compared to the α-version. The predictions for the top site remain relatively modest. During the fitting process for this site, we were only able to achieve a satisfactory fit when the distance between the nitrogen atom and the tungsten surface at this point was at least twice the Bohr radius, which is a value comparable to the radius of a tungsten atom sitting at this spot.

3.4 Limit cases

We chose not to impose constraints on Z in the **3DPES**, **2DCUT** and **1DCUT** programs, as there was no physical justification for doing so. However, it is important to note that, given the approximations involved — such as the Fourier expansion and the development of α — significant numerical errors may arise.

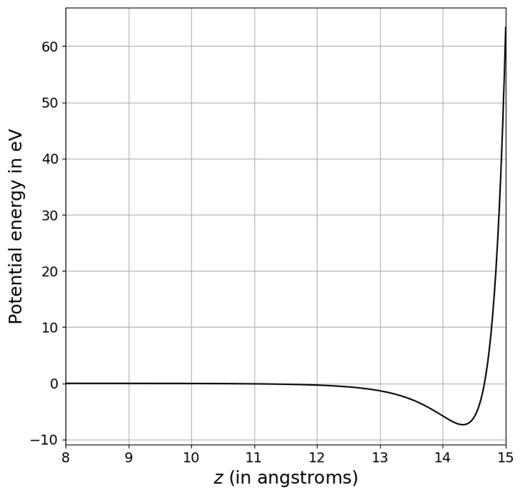


Figure 4. Morse potential for high values of *Z*.

As $(Z \longrightarrow +\infty)$ in expression (1), the potential should naturally converges to zero. However, as shown in **Fig. 4**, the potential becomes completely symmetric and resembles a potential well — highly repulsive for even larger Z. This behavior is not physically expected — periodic conditions may have been assumed in the development of the model — and user should keep Z on the lower side.

4 Physical discussions

*Throughout this section, we will assume the use of the β-version, as it has demonstrated greater accuracy (see **section 3.3**).*

4.1 Adsorption mechanism

The model assumes a gas of nitrogen atoms suspended above a tungsten surface (see **Fig. 5**), where nitrogen atoms do not interact with each other. As a nitrogen atom approaches the surface within a certain distance (~4Å), it begins to experience the potential energy landscape generated by its interactions with the tungsten surface. At this stage, these interactions are primarily electrostatic.

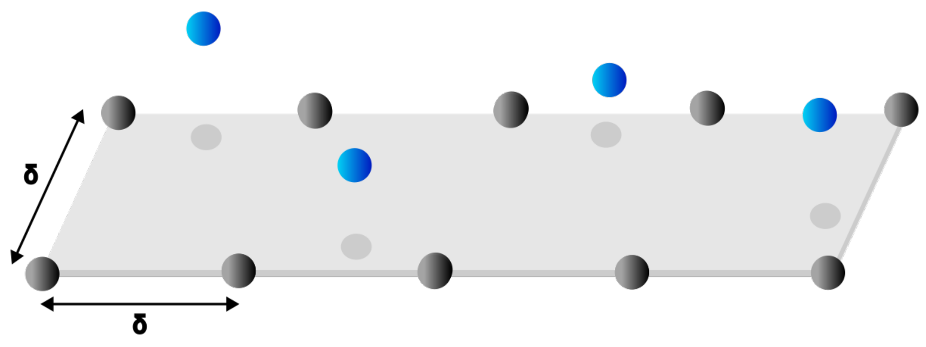


Figure 5. Adsorption model of nitrogen gas on a tungsten surface.

At each height Z, a potential energy curve can be associated with each specific symmetry site on the surface. As shown in **Fig. 6**, at larger distances ($Z > 4Å$), the potential energy curves converge, suggesting that the distinction between adsorption sites becomes negligible. However, at shorter distances, these potential wells start to compete, influencing the nitrogen atom's final adsorption site.

The adsorption site is primarily determined by the depth of the potential well, with the nitrogen atom preferentially binding to the site corresponding to the lowest energy minimum. In this regard, the hollow site exhibits the deepest energy minimum, making it the most favorable adsorption site. At the bottom of this well, the distance between the tungsten and nitrogen atoms (< 1Å) becomes short enough for covalent-like bonding interactions to emerge. These interactions further stabilize the adsorption and likely explain the steepness of the energy well observed in **Fig. 6**.

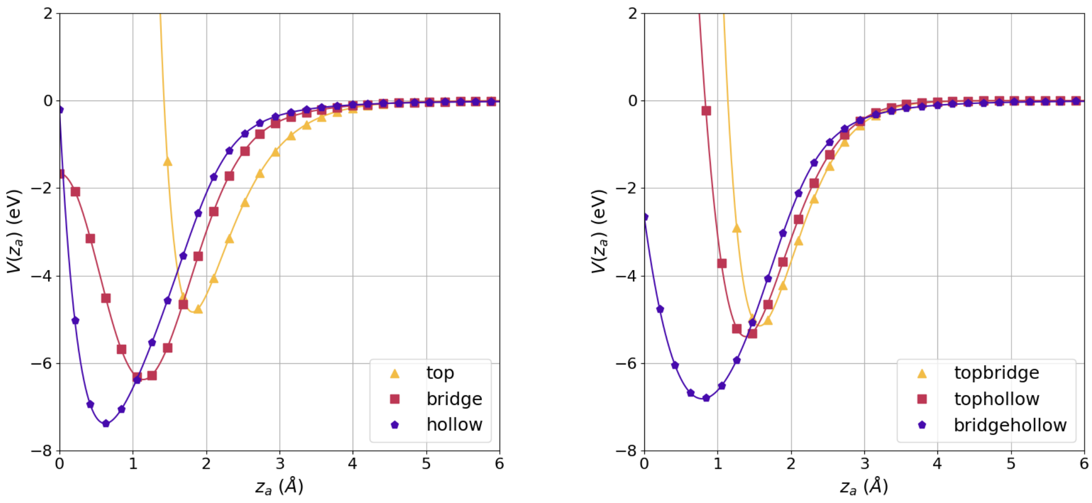


Figure 6. Potential profile of symmetry sites as a function of the height *Z*.

At even smaller *Z*, repulsive effects dominate, preventing the nitrogen atom from penetrating the tungsten structure. This strong repulsion is precisely what distinguishes adsorption from absorption — the nitrogen atom remains bound to the surface without diffusing into the bulk.

4.2 Barriers and diffusion pathways

Once adsorbed onto the tungsten surface, a nitrogen atom can be removed through two distinct processes: desorption, where it escapes into the gas phase, or diffusion, where it migrates between adsorption sites. For clarity, we assume the atom is initially located at a hollow site, as it is the most likely adsorption site according to **section 4.1**, though the same principles apply to any adsorption site.

To assess the likelihood of these processes, we first examine the energy scales involved. The thermal energy fluctuations, approximatively $kT \approx 0.02$eV at room temperature, are much lower than the energy barriers typically associated with diffusion and desorption (> 1eV). This indicates that, under ambient conditions, spontaneous activation of these processes is improbable — external excitation is required to induce significant atomic motion.

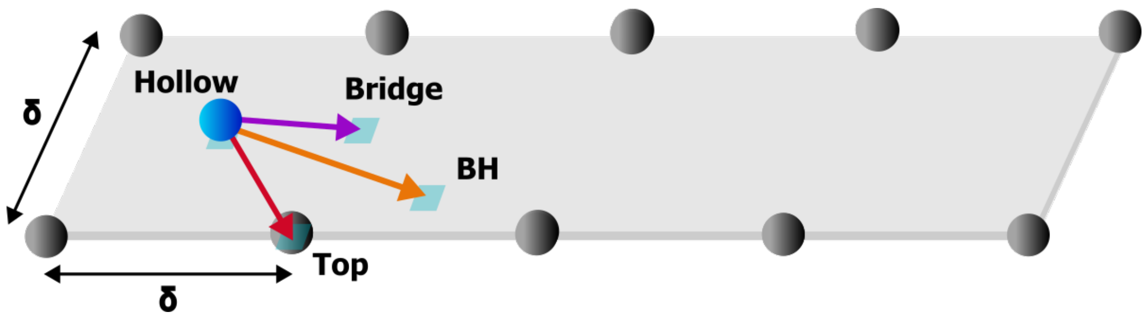


Figure 7. Some possible diffusion pathways for a nitrogen atom adsorbed onto a tungsten surface.

Among the two mechanisms, desorption is the most intuitive. For an atom to detach from the surface, it must acquire enough energy to escape its adsorption potential well. The desorption energy corresponds to the depth of the potential well, which we estimate using the D coefficients listed in **Appendix B** (see **Fig. 1**). On average, this energy is around 6eV.

By contrast, diffusion requires significantly less energy. Using a simple diffusion model, we assume that a nitrogen atom can jump between neighboring sites with a probability given by the Arrhenius equation,

$$P \propto \exp\{-\Delta E / kT\}$$

where ΔE represents the energy barrier between two sites. If $\Delta E > 0$, the transition requires activation energy; if $\Delta E < 0$, it occurs spontaneously.

Target Site	***Energy Barrier (eV)***
Top	2.548529
Bridge	1.004762
Topbridge	2.232567
Tophollow	1.987420
Bridgehollow	0.564879

Table 3. Energy barriers for nitrogen diffusion from the hollow site.

Since hollow sites are the most stable adsorption sites, we focus on their diffusion pathways (see **Table 3**). Because the hollow site is the lowest energy minimum, all diffusion barriers are positive, implying that energy input is necessary for migration. The highest barrier corresponds to hollow → top transition, making it an unlikely event. In contrast, diffusion towards bridge-related sites is much more accessible, as these pathways involve lower activation energies.

Increasing the temperature enhances thermal fluctuations, making diffusion more frequent. However, even at elevated temperatures, diffusion remains significantly more probable than desorption. While diffusion barriers typically range from 0.5 to 2.5 eV, the desorption energy is around 6eV, making it energetically far more difficult for the atom to escape the surface than to move across it.

Conclusion

We successfully reconstructed relevant adsorption data from the CRP model for a nitrogen gas interacting with a tungsten surface, enabling the development of a software tool to generate potential energy surfaces.

We used these surfaces to enlighten atomic diffusion dynamics and gas-surface interactions. While these preliminary results offer valuable insights, further refinements are needed to fully characterize reaction pathways and extend the model to more complex surface phenomena, such as studying the impact of defects to diffusion.

Additionnaly, we regret not having enough time to investigate the behavior of the N_2 molecule, which would be crucial for a more realistic description of nitrogen adsorption in experimental conditions.

A Coefficients for the α-version

	Top	Bridge	Hollow
D	4.843306	5.603632	7.613337
α	1.991931	0.950659	1.252572
r^{eq}	1.736781	0.785308	0.565774

Table. Morse potential parameters fitted on CRP data.

	P_0	P_1	P_2
D	5.915976762771606	-0.69250774383544922	0.15617239475250244
α	1.2864552140235901	0.18483975529670715	0.16789811849594116
r^{eq}	0.96829275786876678	0.29275174438953400	9.1492377221584320E-002

Table. Fourier coefficients determined.

B Coefficients for the β-version

	Top	Bridge	Hollow	Topbridge	Tophollow	Bridgehollow
D	4.833511	6.377278	7.382040	5.149473	5.394620	6.817161
r^{eq}	1.816304	1.142394	0.621643	1.575456	1.384302	0.791984
a	4.945993	1.195805	1.439399	1.726881	2.583052	1.281390
b	-0.728908	-3.349518	-2.889758	-1.300011	0.000000	-2.618644

	Top	Bridge	Hollow	Topbridge	Tophollow	Bridgehollow
a^0_{rep}	2.022437	0.541444	1.100547	1.136776	0.786462	0.722033
a^1_{rep}	-0.069871	0.474494	-0.084422	0.727781	0.334543	0.089125
a^0_{att}	1.730317	1.771598	2.060582	1.119403	0.962540	1.977717
a^1_{att}	0.031893	-0.043323	-0.143909	0.331250	0.352351	-0.115216

Table. Morse potential parameters fitted on CRP data.

	P_0	P_1	P_2
D	5.9009452164173126	-0.73548811674118042	-6.7375600337982178E-2
r^{eq}	1.2331064455211163	0.34520062059164047	1.9144885241985321E-2
a	1.9463933408260345	0.54969701170921326	0.49922275543212891
b	-1.6245201490819454	0.59976450353860855	0.38504625111818314

	P_0	P_1	P_2
a^0_{rep}	0.92418475449085236	0.21892200410366058	0.25501200556755066
a^1_{rep}	0.33753068558871746	0.16148287430405617	-0.13791012763977051
a^0_{att}	1.4732959344983101	-0.25586162507534027	3.0962869524955750E-2
a^1_{att}	0.12967987963929772	0.13359175436198711	-3.1712511554360390E-3

	P_3	P_4	P_5
D	0.21197667717933655	4.9177914857864380E-2	-4.1185960173606873E-2
r^{eq}	-5.0904575735330582E-2	-2.3267690092325211E-2	2.4693211540579796E-2
a	-9.7200363874435425E-2	0.16347573697566986	0.22112892568111420
b	-0.64485638961195946	-2.9775988310575485E-2	0.16740368492901325

	P_3	P_4	P_5
a^0_{rep}	6.6251501441001892E-2	5.7752653956413269E-3	-2.6098713278770447E-3
a^1_{rep}	-3.3967308700084686E-2	-7.8922562301158905E-2	-3.5461156629025936E-2
a^0_{att}	0.21774593740701675	8.6647696793079376E-2	-3.7632044404745102E-2
a^1_{att}	-0.10050412779673934	-4.4820626266300678E-2	1.0831437306478620E-2

Table. Fourier coefficients determined.