Supplementary information

Interaction models and molecular simulation systems of steel - organic friction modifier interfaces

Arkadii Pominov* Johannes Mueller-Hillebrand*
Johannes Traeg* Dirk Zahn†, *

*Lehrstuhl für Theoretische Chemie/ Computer Chemie Centrum
Friedrich-Alexander Universität Erlangen-Nürnberg
Nägelsbachstraße 25, 91052 Erlangen, Germany
†dirk.zahn@fau.de

Source Code 1: The source code in Python for generation of the OFM monolayers with various number of molecules.

```python
from math import isclose

class Lattice2D:
    """Class to describe a 2D periodic lattice
    """

    def __init__(self, basis, constants, lims):
        """Sets up the parameters, necessary
        to define a 2D lattice and generates the lattice

        Arguments:
            basis {list of (x,y)-tuples} -- in unit cell coordinates,
                                            e.g. [(0, 0), (0.5, 0.5)]
            constants {dict with keys ("a" and "b") and floats as values} --
            e.g. {"a": 1.0, "b": 1.0}
            the user must ensure these fit integer number of times
            in both directions
            lims {dict with keys ("x" and "y") and tuples as values} --
            {"x": (x_min, x_max), "y": (y_min, y_max)},
            describe the boundaries of the slab/surface
        """
        self.basis = basis
        self.constants = constants
        self.xlims = lims["x"]
        self.ylims = lims["y"]
        # List of tuples (x,y) for all points
        self.coordinates = None
        # Number of points on the lattice
        self.number = None
        # hexagonal criterion
        # (lattice_a_constant**2+lattice_b_constant**2)**0.5/2/lattice_a_constant
        # describes deviation from the hexagonal lattice
        # and ==1 if a and b equal
        self.hexcrit = None
        self.__generate()

    def __generate(self):
        """
        Generates coordinates of points for Lattice2D
```

```
        """
        x_min, x_max = self.xlims
        x_cur = x_min
        y_min, y_max = self.ylims
        self.coordinates = []
        for xbasis, ybasis in self.basis:
            # to prevent going beyond the given boundaries
            while not isclose(x_cur, x_max, rel_tol=0.01):
                y_cur = y_min
                while not isclose(y_cur, y_max, rel_tol=0.01):
                    self.coordinates.append(
                        (
                            x_cur + xbasis * self.constants["a"],
                            y_cur + ybasis * self.constants["b"],
                        )
                    )
                    y_cur += self.constants["b"]
                x_cur += self.constants["a"]
            x_cur = x_min
        self.number = len(self.coordinates)
        self.hexcrit = (
            (self.constants["a"] ** 2 + self.constants["b"] ** 2) ** 0.5
            / 2
            / self.constants["a"]
        )


# define surface dimensions
x_dim = 100.71
y_dim = 34.887
# only discrete lattice constants are possible when periodicity is maintained
nmax_x = 35  # 1/nmax_x is the smallest periodic interval in x
nmax_y = 12  # 1/nmax_y is the smallest periodic interval in y
lattice_a_constants = [x_dim / (i + 1) for i in range(1, nmax_x)]
lattice_b_constants = [y_dim / (i + 1) for i in range(1, nmax_y)]

# to choose only sensible lattices
criterions = {
    "lo_number": 1,
    "hi_number": 190,
    "hex_criterion_lo": 0.8,
    "hex_criterion_hi": 1.2,
}
hex_basis = [(0, 0), (0.5, 0.5)]
```

```
84
85  lattices = []
86
87  for a in lattice_a_constants:
88      for b in lattice_b_constants:
89          lattice = Lattice2D(
90              hex_basis, {"a": a, "b": b}, {"x": (0, x_dim), "y": (0, y_dim)}
91          )
92          # only accept the generated lattice if criterions are fulfilled
93          if (
94              criterions["lo_number"] < lattice.number < criterions["hi_number"]
95          ) and (
96              criterions["hex_criterion_lo"]
97              < lattice.hexcrit
98              < criterions["hex_criterion_hi"]
99          ):
100             lattices.append(lattice)
101         else:
102             continue
103 # sort the generated lattices by the number of points in them
104 sorted_lattices = sorted(lattices, key=lambda lattice: lattice.number)
```

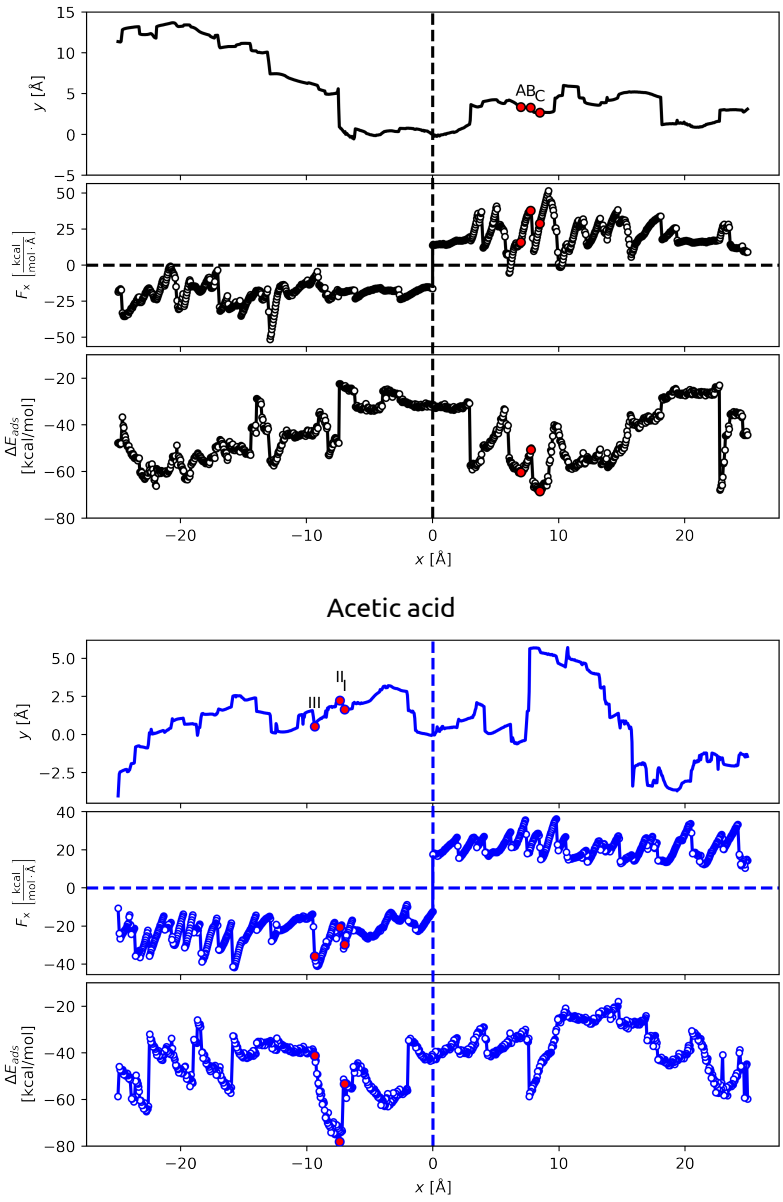
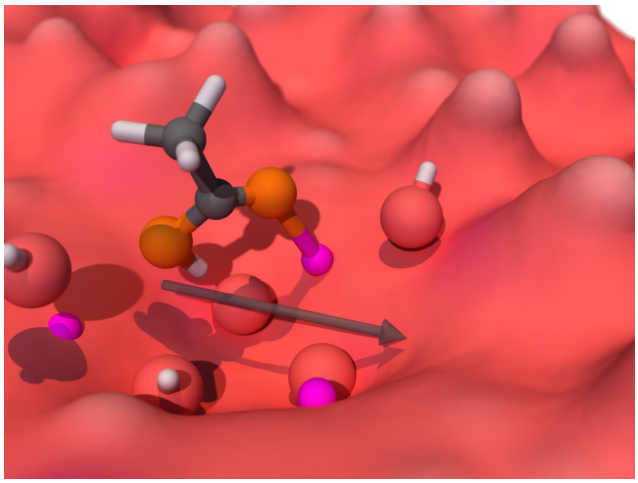
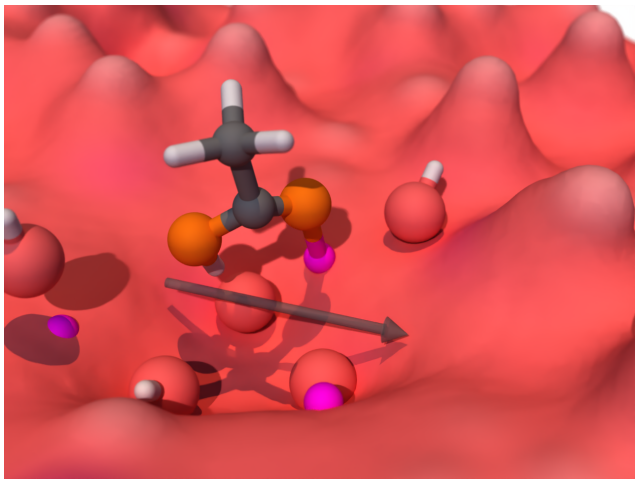


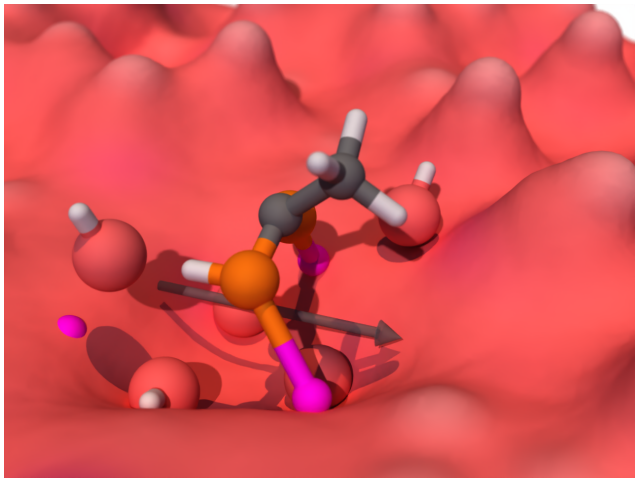
Figure s1: Pulling the molecules along the surface was performed from $x = 0$ to $x = 2.5$ nm and from $x = 0$ to $x = -2.5$ nm. Profiles of the orthogonal (y) coordinate, pulling force and adsorption energy are shown as functions of x. The indices A-C and I-III denote the configurations which are illustrated in fig. s2 and s3, respectively.



A

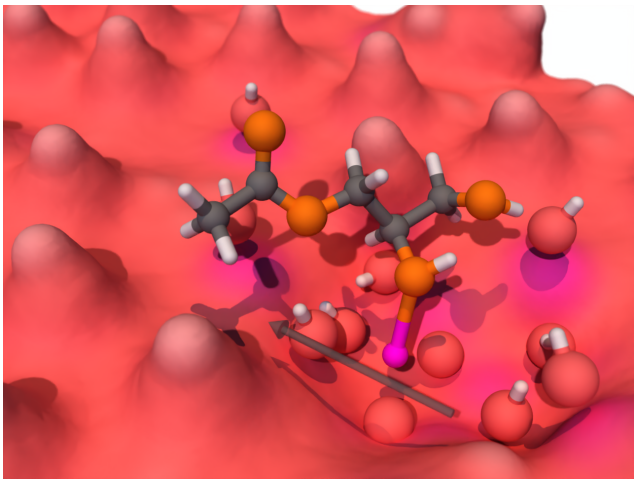


B

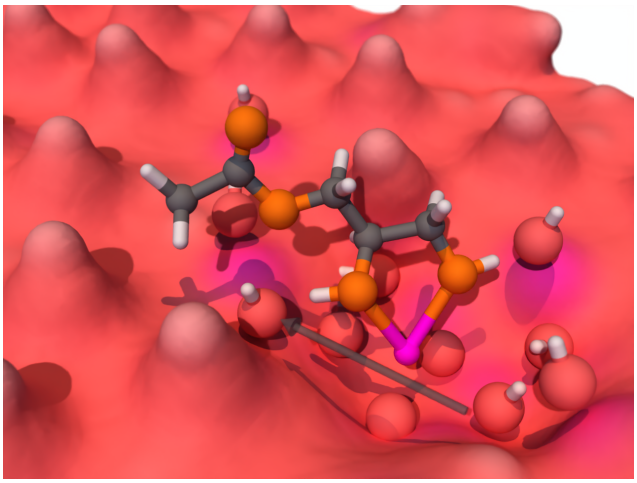


C

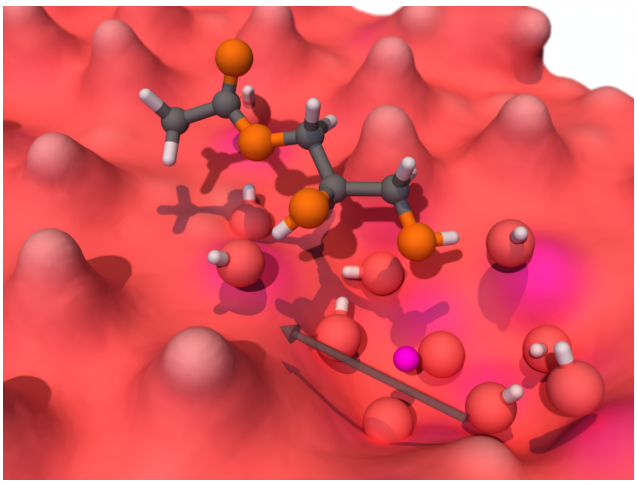
Figure s2: Bond stretching (A→B) and reorganization (B→C) of acetic acid upon pulling along the surface. The arrows indicate the pulling direction.



I



II



III

Figure s3: Bond stretching (I→II) and reorganization (II→III) of glycerol monoacetate upon pulling along the surface. The arrows indicate the pulling direction.