Supplementary Information for

Molecular origin of slippery behavior in tethered liquid layers

Fabio Rasera,* Antonio Tinti, and Alberto Giacomello†

Dipartimento di Ingegneria Meccanica e Aerospaziale,

Sapienza Università di Roma, 00184, Rome, Italy

Isaac J. Gresham* and Chiara Neto‡

School of Chemistry, The University of Sydney, NSW 2006, Australia and

The University of Sydney Nano Institute,

The University of Sydney, NSW 2006 Australia

(Dated: January 30, 2025)

I. SIMULATIONS

A. Simulations details

CG-MD simulations of polydisperse PDMS layers were conducted by modelling interactions with the most recent MARTINI force field (MARTINI 3 [1]) in which, on average, two to four heavy atoms and associated hydrogens are mapped into one CG bead. This framework enables the computation of systems larger than those accessible to all-atom models and for longer times, with an overall speedup of at least two orders of magnitude [2], while maintaining sufficient chemical and spatial resolution. Given its underlying building block principle, the MARTINI force field has been widely used for simulating polymeric systems[3, 4], including grafted polymers[5, 6]. This approach allowed to simulate for long times sufficiently large surface areas grafted with chemically realistic PDMS [7] molecules, while scanning a number of relevant constructive and environmental parameters. The simulations were carried at grafting densities σ=0.1, 0.2, 0.3, 0.6, and 1.0 gps/nm^2. Systematic simulations were performed for smaller domains of size 10 nm by 50 nm, using three replicas at each grafting density, to take into account different arrangements of the PDMS layers. Large individual P1, P2, and P3 surfaces of size 80 nm by 80 nm were also simulated.

The average layer thickness d was determined by dividing the systems into a 2D grid with a resolution of 1 nm^2. The highest atom position within each grid cell was identified, and the arithmetic average of these positions was calculated. Results for both polydisperse and monodisperse layers are presented in Fig. S2.

Volume fractions along a given coordinate are valuable for visualizing the packing of a grafted layer and the shape of its interface with a solvent. As shown in Fig. S3, the volume fractions of PDMS layers in water become more defined as the grafting density increases. Monodisperse layers exhibit a box-shaped volume fraction, indicating a flat interface with water, while polydisperse layers display curved volume fractions, reflecting their irregular surfaces. This contrast is particularly evident when comparing M300 and P3, as P3 features

* These authors contributed equally
† alberto.giacomello@uniroma1.it
‡ chiara.neto@sydney.edu.au

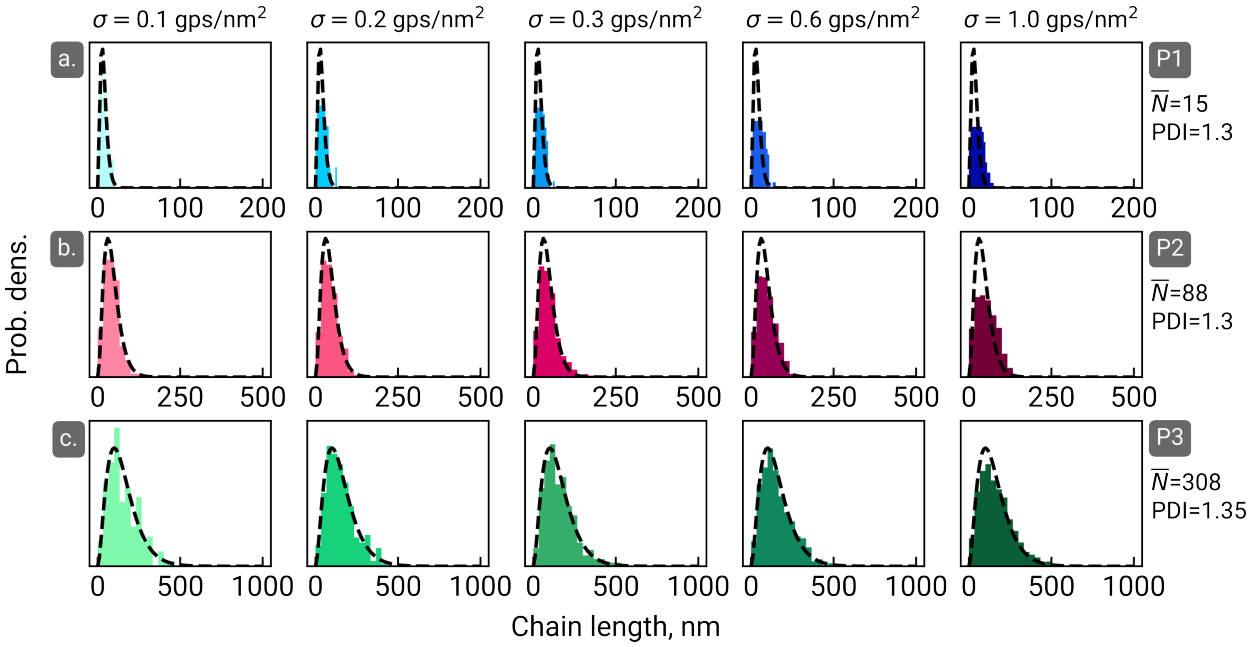

Fig. S1. **Chain length distributions.** The length distribution of simulated chains (histogram) compared to the Schulz-Zimm distribution obtained from experiments in [8] (line) for a) P1, b) P2 and c) P3 samples, in the 0.1-1.0 gps/nm^2 grafting density range.

a wavy surface.

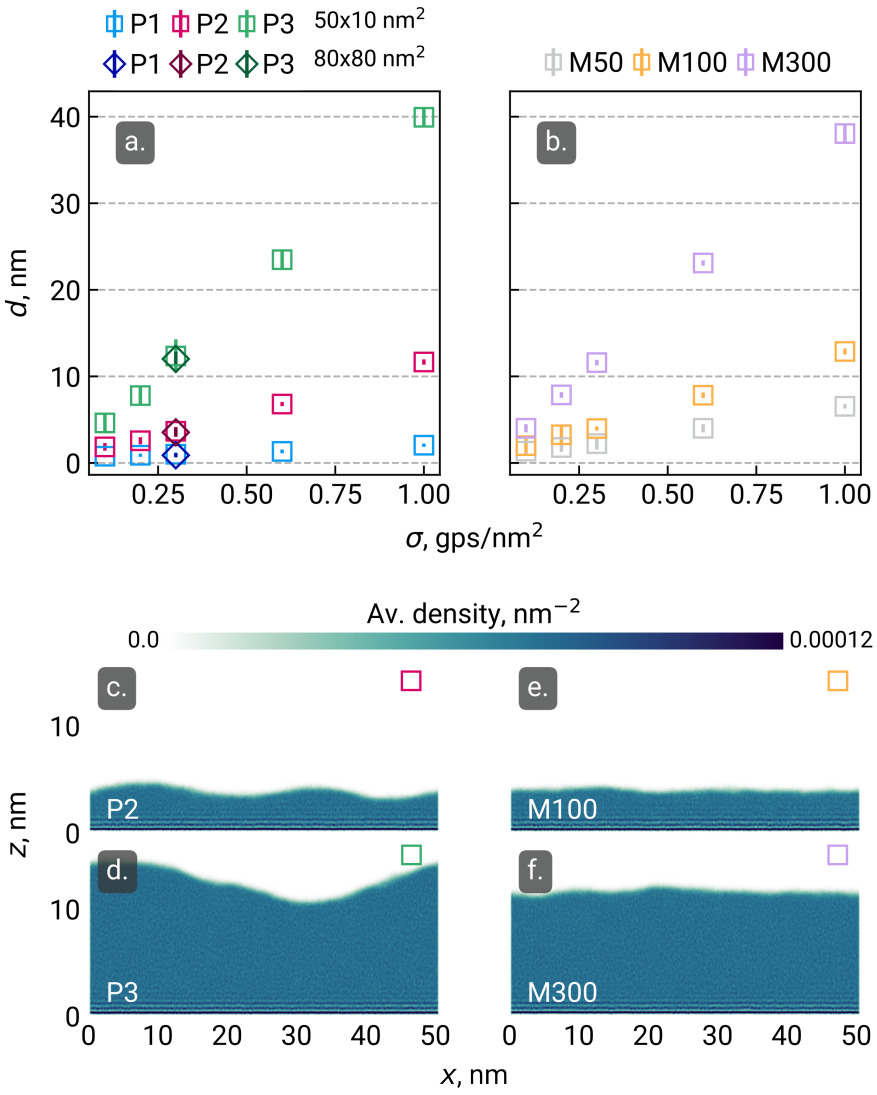

Fig. S2. **Layer Thickness**: average layer thickness d of simulated PDMS layers. Mean layer thickness of PDMS chains with a) polydisperse and b) monodisperse chain length distribution. The average layer thickness is primarily influenced by the mean length of the composing chains. At an intermediate grafting density (0.3 gps/nm^2), the PDMS density during 1 ns of a MD trajectory reveals that (c) P2 and (d) P3 surfaces exhibit a wavy morphology, while (e) M100 and (f) M300 surfaces are relatively flatter, despite having similar average thickness. The P2 surface shows shorter wavelengths and smaller bump heights compared to P3, resembling a monodisperse layer.

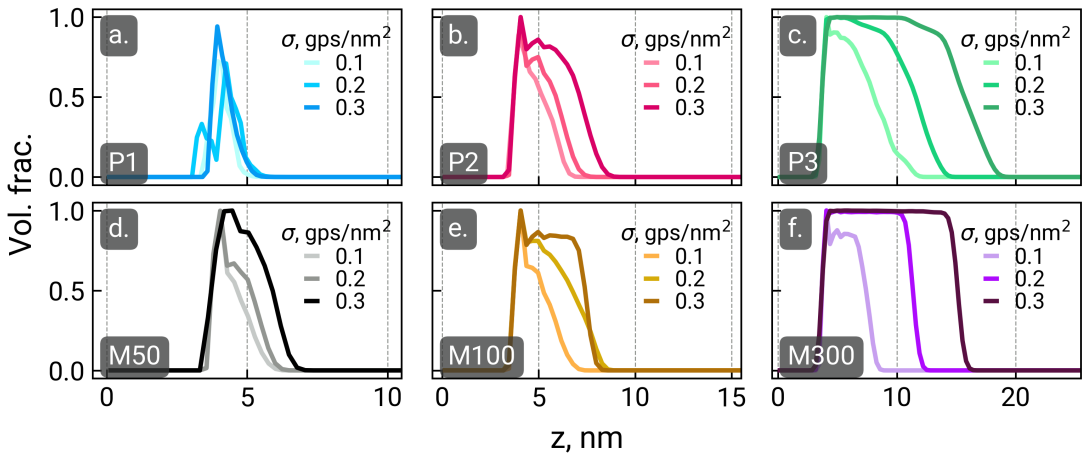

Fig. S3. **PDMS volume fraction from simulations**: volume fractions of polydisperse (P1, P2, P3) and monodisperse layers (M50, M100, M300) in water were analyzed over a grafting density range of 0.1-0.3 gps/nm^2. For the polydisperse cases, volume fractions were averaged across three different realizations, while in the monodisperse cases, a single system was simulated for each chain length and grafting density. In monodisperse samples, the volume fraction assumes a box-like shape once there is sufficient material to form a fully covered surface. Conversely, polydisperse samples exhibit a gradual decrease in volume fraction, reflecting the irregularity of their surface.

B. Vertical phase separation

As discussed in the main paper, the vertical segregation of chains by length in this work (Figure 3) is very similar to that produced by the numerical self-consistent field theory of de Vos and Leermakers [9]. For ease of comparison, the distributions of different chain fractions from our work and that of de Vos are plotted together in Figure S4. The profiles of de Vos are for a polydisperse brush in a good solvent.

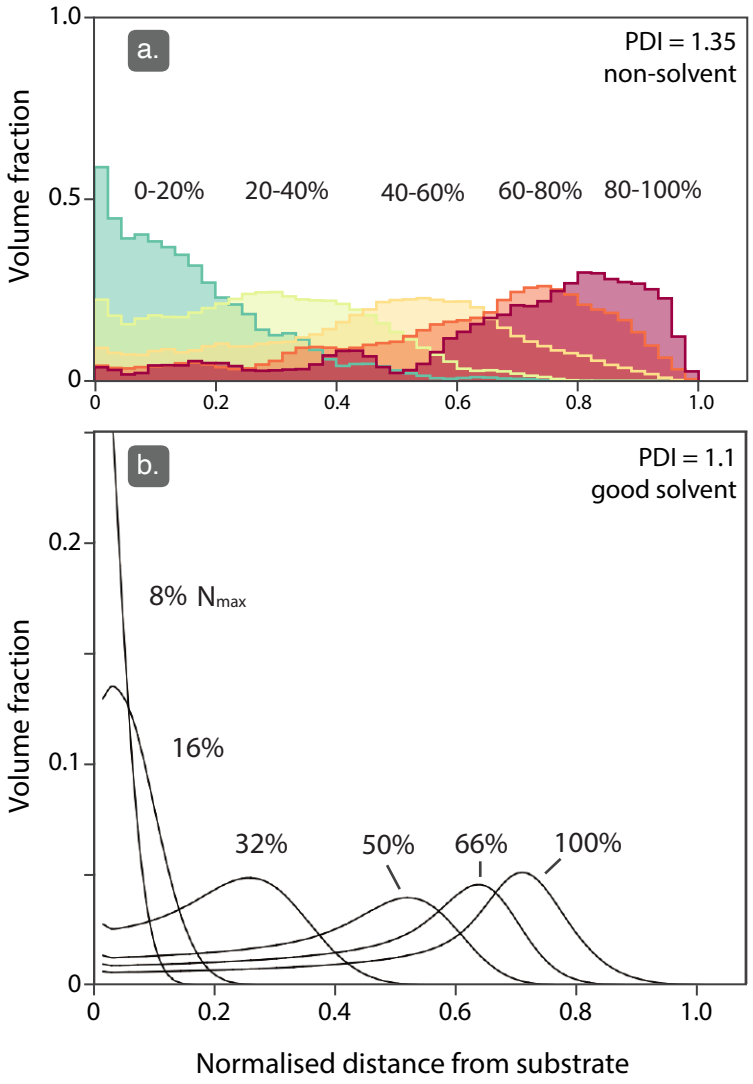

Fig. S4. **Vertical phase separation**: vertical segregation of polymer chains by length, as predicted by the (a) MARTINI coarse-grained molecular-dynamics used here (Figure 3), and (b) the numerical self-consistent field theory of de Vos and Leermakers [9]. Both approaches predict that longer chains segregate to the layer periphery, and shorter chains concentrate at the base of the layer. The distributions given by de Vos are for a swollen layer (i.e., polymer in a good solvent — $\chi = 0$), while those from the current work are for a collapsed layer (i.e., in poor solvent).

Reduced grafting density

The reduced grafting density Σ, as defined in Eq. 1 of the main text, can be written for unsolvated polymers as (see Gresham et al.[8] for the derivation):

$$\Sigma = \sigma k \left(\frac{\overline{N} M_m}{\rho} \right)^{2/3} , \tag{S.1}$$

where k is a constant equal to $\pi \left(\frac{3}{4\pi N_A} \right)^{2/3}$ and σ, $\overline{N}$, M_m and ρ are the grafting density, average monomer number of the chains, molecular mass and density. For simulated samples, ρ was calculated using the double cubic lattice method[10], as implemented in the SASA tool from GROMACS.

Substrate deformation

Fig. S5 shows the deformation of two polydisperse (P2 and P3) and two monodisperse (M100 and M300) substrates, quantified by the angle formed between the local orientation of the surface before and after equilibration with water. The polydisperse P3 exhibit the most pronounced ridge formation, resulting in a significant change of local curvature in comparison with other samples. This is indicative of stronger contact line pinning and greater surface deformation, which correlates with higher contact angle hysteresis (CAH) [11]. In contrast, the monodisperse surfaces (M100 and M300), as well as the polydisperse P2, display a ridge formation that is more symmetrical, resulting in a smaller deformation angle, suggesting that these surfaces experience weaker pinning and, possibly, lower energy dissipation in comparison with P3.

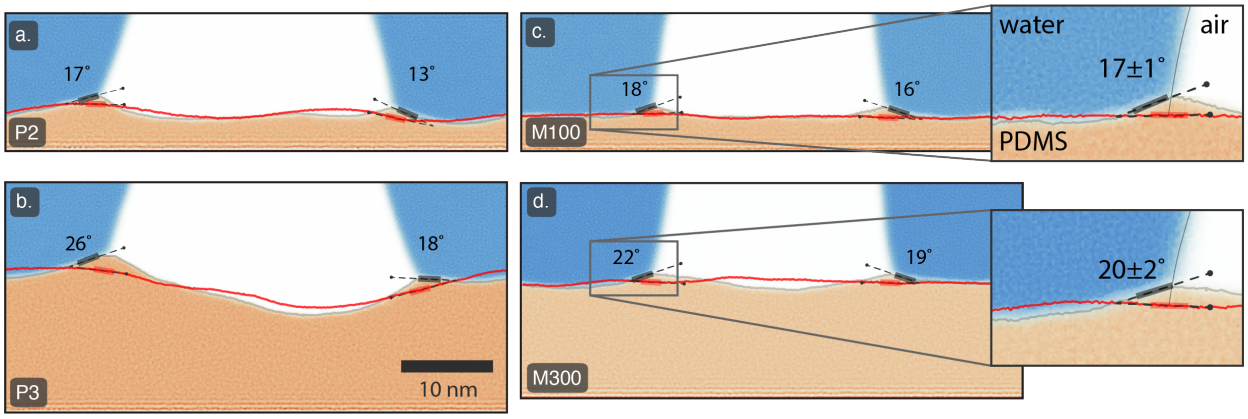

Fig. S5. Deformation of the PDMS layer at the three-phase contact line for (a) P2, (b) P3, (c) M100 and (d) M300 substrates. Insets are provided for monodisperse layers for direct comparison with Fig. 4; as in Fig. 4 average values are reported in the inset.

C. A_{sub} and A_{PDMS} calculation

The marching cubes algorithm[12], as implemented in the scikit image library[13], was used to calculate the surface area of the PDMS layers A_{PDMS}. It works by dividing the atom coordinates into a grid of cubes and determining the intersection of a desired isosurface

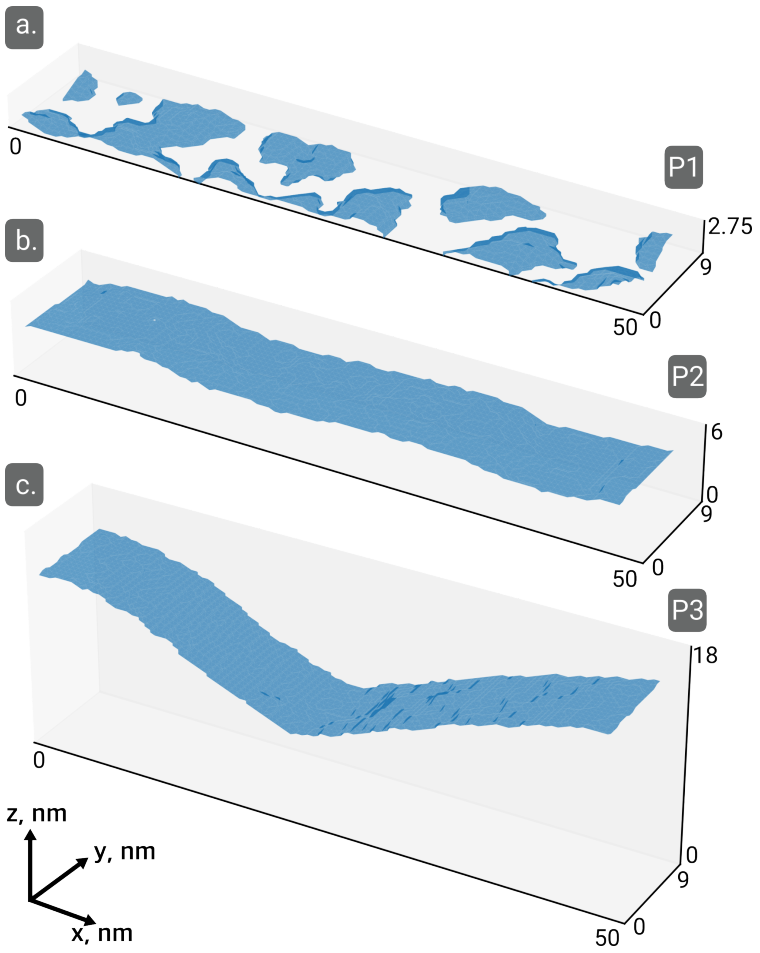

Fig. S6. **Reconstruction of the PDMS surface**: Surfaces reconstruction using marching cubes algorithm. a) P1 realization, rich in chemical defects. b) P2 realization, with approximately flat surface. c) P3 realization, with accentuated waviness.

with each cube, creating triangles that approximate the surface topography. A_{sub} is then calculated as the base area A minus the projected area of the PDMS surface onto the plane parallel to the substrate surface. Figure S6 shows 3 surfaces reconstructed by coloring the many triangles that compose the surface, where the sum of all triangles' areas is A_{PDMS}.

II. MODELLING

A. Hysteresis induced by nanodefects

In order to connect the characteristics of nanoscale defects and macroscopic contact angle hysteresis (CAH), we build upon the framework developed in Ref. [14]. There, the free-energy profile connected with the advancement or retreat of a liquid front across nanoscale defects of chemical or topographical nature was computed. Classical density functional simulations showed that, at such small sizes, the free-energy profile is typically sigmoidal, accounting for either advancing or receding defects, depending on their nature (e.g., chemical defects more hydrophobic than the substrate or more hydrophilic, respectively). Such nanoscale defects can withstand some external force coming from an external pressure or from the unbalanced Young force $\gamma(\cos\theta_{\rm app} - \cos\theta_Y)$ of a droplet having an apparent contact angle $\theta_{\rm app}$ different from the Young contact angle θ_Y of the undefected surface. The maximum (minimum) value of such defect force is dictated by the inflection point in the free-energy profile for the advancing/receding process; after this spinodal point is reached, the liquid front jumps beyond the defect and the deformations of the triple line are relaxed [14].

While capturing the detailed shape of the liquid front and accounting for subnanoscale defects require microscopic calculations, an estimate of the free-energy jump $\delta\Omega^{\rm def} = \Omega_{\rm wet} - \Omega_{\rm dry}$ for overcoming an individual defect can be supplied by macroscopic capillarity. For a chemical defect this reads:

$$\delta\Omega^{\rm chem} = \gamma(\cos\theta_Y - \cos\theta_{\rm def})A_{\rm def}$$

where γ is the liquid-vapor surface tension, θ_Y and $\theta_{\rm def}$ the Young contact angles characterizing the undefected surface and the defect, respectively, and $A_{\rm def}$ is the area of the defect. Similarly, for a topographical defect, one has:

$$\delta\Omega^{\rm top} = -\gamma\cos\theta_Y A_{\rm exc}$$

where $A_{\rm exc}$ is the excess area of the defect associated to topographic roughness, i.e., the additional area as compared to the undefected surface. The previous expressions are used in the main text, assuming that the undefected surface coincides with a perfectly flat PDMS

layer ($\theta_Y = \theta_{\mathrm{PDMS}}$) and that chemical defects have the wetting characteristics of the silica substrate ($\theta_{\mathrm{def}} = \theta_{\mathrm{sub}} = 75°$).

The second step in the CAH theory was recognizing that it is possible to estimate the force exerted by a dilute random distribution of defects on a macroscopic triple line, e.g., that of a droplet, from the characteristics of individual defects [15]. In order to do this, one integrates the forces coming individual defects at all relative distances between the defect and the unperturbed triple line, taking into account only the stable positions of the triple line, i.e., before the spinodal jumps [14]. This leads to the following expression for the unbalanced Young force acting on the macroscopic triple line:

$$\gamma(\cos\theta_{\mathrm{app}} - \cos\theta_Y) = -n(\Omega_{\mathrm{max}} - \Omega_{\mathrm{min}})$$

where n is the surface density of defects and the subscripts *max* and *min* denote the free energies computed at the maximum and minimum stable position of the triple line, respectively. Using the expression above for the advancing and receding processes, one obtains

$$\cos\theta_{\mathrm{r}} - \cos\theta_a = n\frac{|\delta\Omega_{\mathrm{sp}}^{\mathrm{def}}|}{\gamma}$$

where the free energy dissipated when the triple line jumps across each advancing or receding defect, $|\delta\Omega_{\mathrm{sp}}^{\mathrm{def}}|$, is calculated at the spinodal, i.e., at the configuration of the triple line where the maximum or minimum force is attained before the triple line snaps beyond the defect. The expression above allows one to compute CAH given the defect characteristics and their surface density.

The dissipated energy $|\delta\Omega_{\mathrm{sp}}^{\mathrm{def}}|$ should be calculated at the inflection point in the free-energy profile. In the main text, since we do not have an explicit calculation of such quantity for each defect, we simply assume that the free-energy profile is a sigmoidal function with height $\delta\Omega^{\mathrm{def}}$ such that the inflection point is simply achieved at the midpoint and $|\delta\Omega_{\mathrm{sp}}^{\mathrm{def}}| \approx |\delta\Omega^{\mathrm{def}}|/2$. Furthermore, since we only have defects of chemical or topographical nature, the integration over all defects of each kind simply corresponds to summing over the areas in which the substrate is exposed, $A_{\mathrm{def}} = A_{\mathrm{sub}}$, for chemical defects and over the excess areas for topographical ones, $A_{\mathrm{exc}} = A_{\mathrm{PDMS}} - (A - A_{\mathrm{sub}})$. The latter expression takes into account the fact that chemical and topographical defects can coexist, in which case the excess area should be computed with respect to only the portion of the projected surface which is covered

by PDMS, $(A - A_{\mathrm{sub}})$.

III. EXPERIMENTS

A. Imaging CALS with atomic force microscopy

The meniscus force mapping (MFM) approach described in the work allows for the quantitative and reproducible imaging of tethered liquid surfaces. The primary reasons why MFM was used instead of conventional tapping-mode atomic force microscopy is that tracking the surface of a thin liquid layer with tapping mode proved to be difficult on the available instrumentation. In short, the range of 'tapping force' (increased either through decreasing the set point or increasing the drive amplitude) required to adequately track the surface of the liquid layer is narrow. If the tapping force is too low, the tip no longer tracks the surface, if it is too high, the tip instead tracks the silicon substrate. Consequently, it difficult to track the liquid surface, with the tracked surface often switching mid-image as in Figure S7 without a change in scan parameters. Further complicating matters, both the silicon substrate and PDMS surface can be exceptionally smooth, which makes differentiating the two surfaces difficult during imaging. As such, we found that the MFM technique was much more robust than tapping mode.

Additional reasons MFM was used over tapping mode included the ability to measure both the thickness of the liquid layer and variations in tip-layer adhesion. Furthermore, the Bruker Peakforce mode allowed for force maps to be acquired at approximately the same rate as conventional tapping mode on the same instrument (approximately 5 minutes an image).

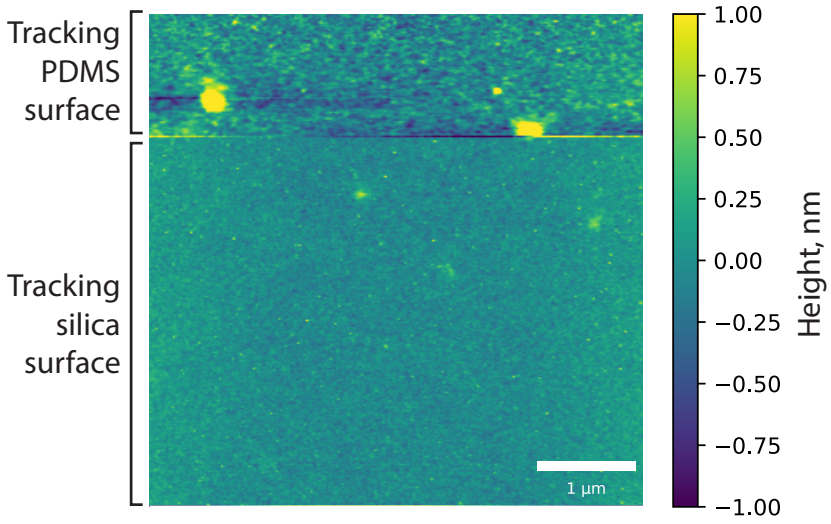

Fig. S7. **Tapping mode AFM**: 5 μm tapping mode atomic force microscope scan of a 4 nm thick PDMS CALS taken on an Asylum MFP 3D. During the scan, the tip transitioned from tracking the PDMS surface (top) to tracking the silica substrate (bottom). When tracking the PDMS surface, the texture appears similar to the wavy surfaces observed in Figure 2.

B. Complete output for AFM measurements

Additional data extracted from meniscus force measurements for the surfaces shown in Figure 2 are presented in Figures S8 to S14. For each surface, 500 and 1000 nm scans are provided. In each figure, maps of a number of parameters are produced:

Topography: The silica substrate topography as defined by the z-position at which the force equals to the peakforce (here 5 nN). The substrate is indicated in the force curves by shaded grey regions.

Retract max adhesion force: The maximum force exerted on the cantilever upon tip retraction, indicated by the horizontal red line. This is used as a measure of chemical heterogeneity in Figure 2c. These measurements were taken consecutively using the same AFM tip, so a direct comparison of the adhesion values is valid.

Jump-in: The point at which the cantilever begins to feel meniscus attraction to the sample, indicated by the vertical light-blue line. The jump-in point is taken to correspond to the surface of the tethered-liquid surface. To produce the images in Figure 2b, the jump-in point is added to the substrate topography.

Jump-off: The point at which the cantilever ceases to interact with the substrate, indicated by the dark blue line. It has been taken to be proportional to polymer molecular weight in other work [16].

Work of attraction: The integral of the approach curve between the points of jump-in and net-repulsion.

Work of adhesion: The integral of the retract curve between the points of jump-in and net-repulsion. The work of adhesion is used in Fig. 4e as a proxy for the energy required for meniscus formation.

Start of net repulsion (approach): separation at which the force acting on the cantilever becomes repulsive (positive) on approach. Values should be close to zero; values far from zero either indicate an elastic substrate or a problem with the measurement (e.g., poorly calibrated optical sensitivity).

Start of net repulsion (retract): separation at which the force acting on the cantilever

becomes repulsive (positive) on retract.

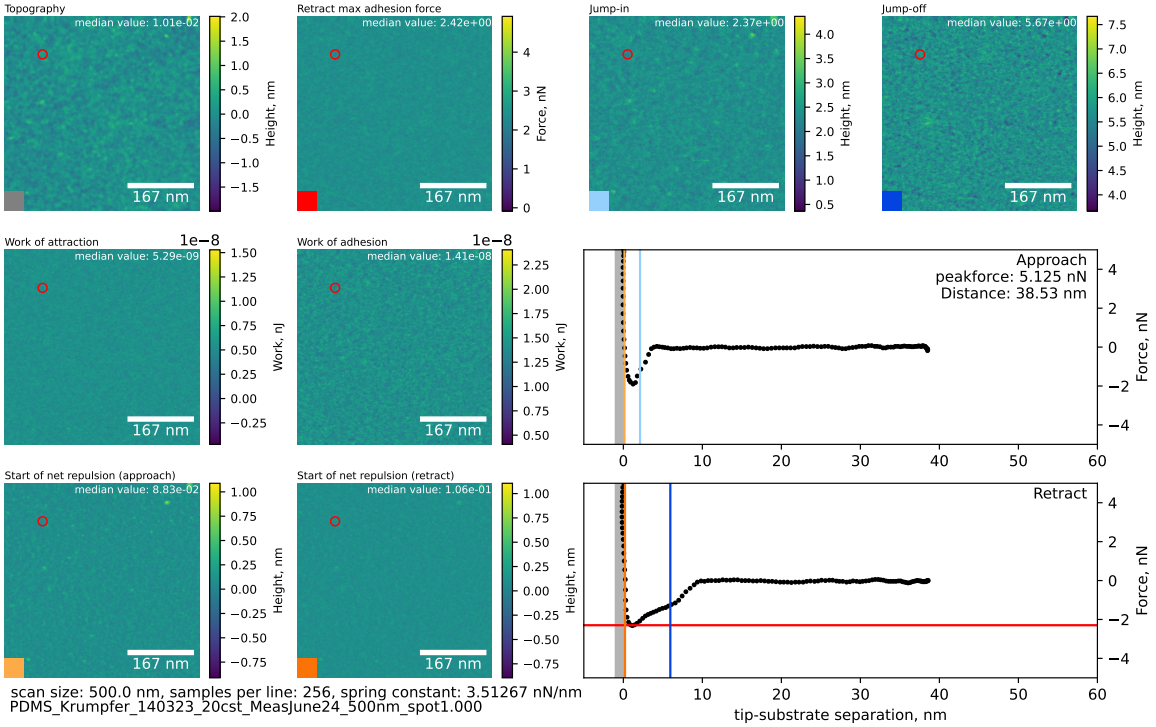

Fig. S8. Meniscus force measurements for sample P1 (1 nm thick), 500 nm scan

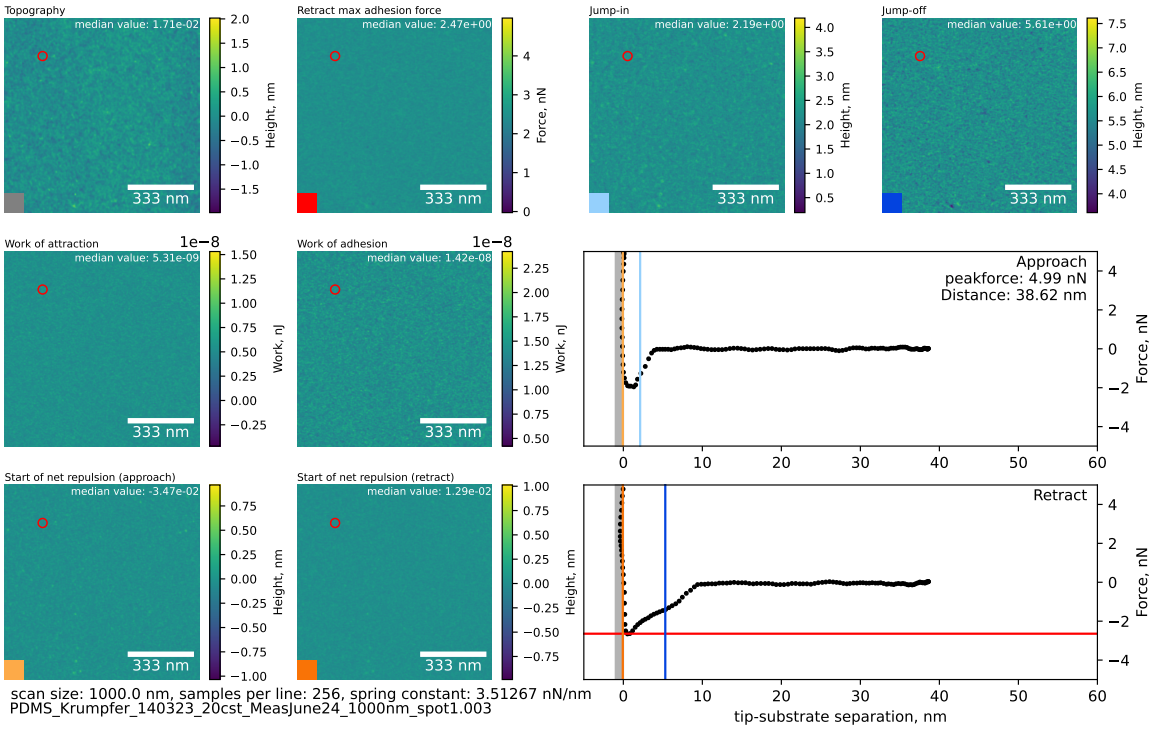

Fig. S9. Meniscus force measurements for sample P1 (1 nm thick), 1000 nm scan

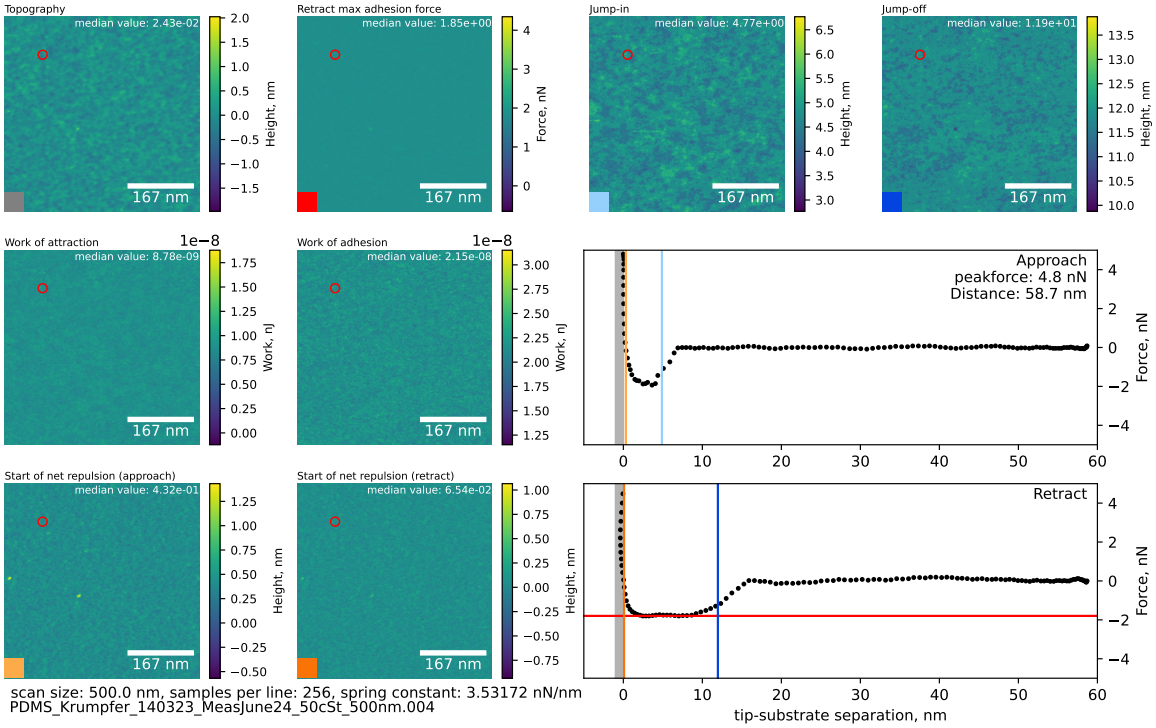

Fig. S10. Meniscus force measurements for sample P2 (3 nm thick), 500 nm scan

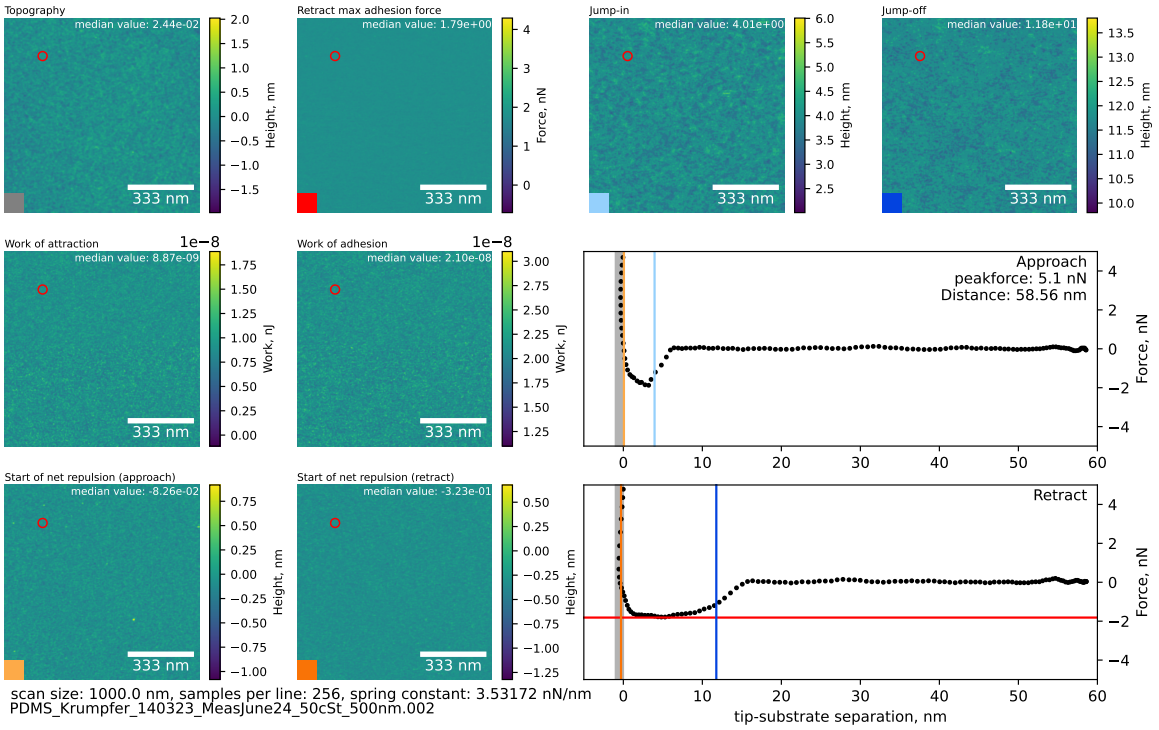

Fig. S11. Meniscus force measurements for sample P2 (3 nm), 1000 nm scan

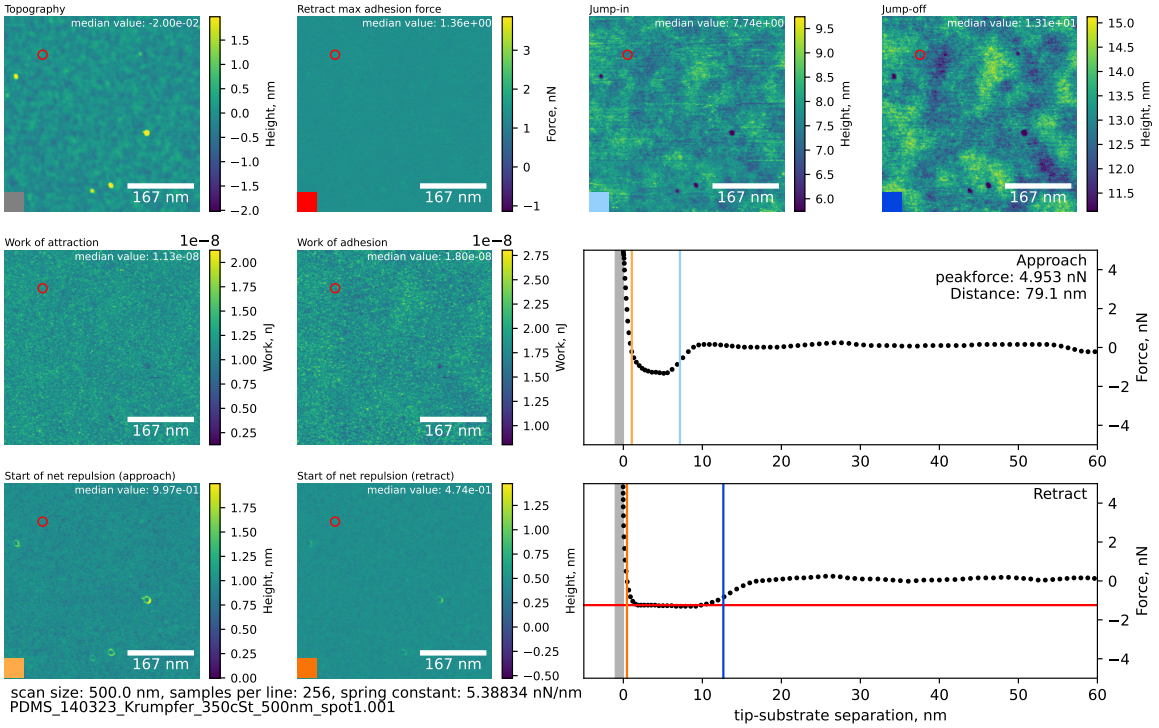

Fig. S12. Meniscus force measurements for sample P2 (5 nm thick), 500 nm scan

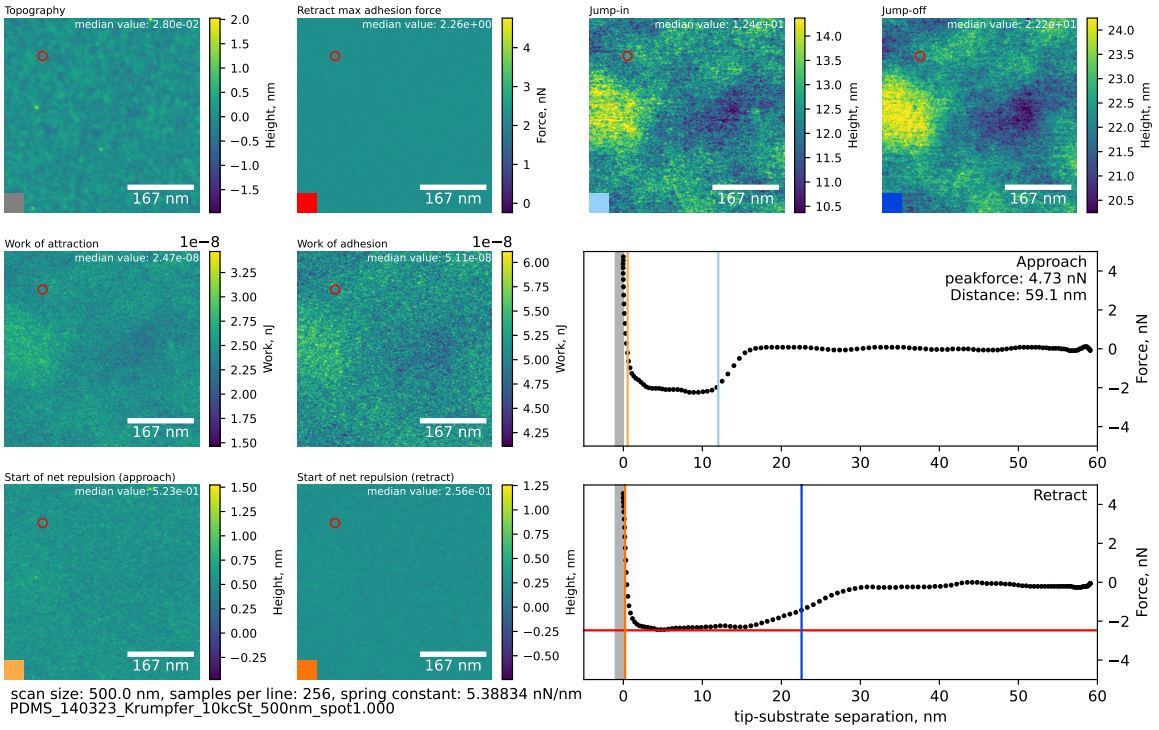

Fig. S13. Meniscus force measurements for sample P3 (8 nm thick), 500 nm scan

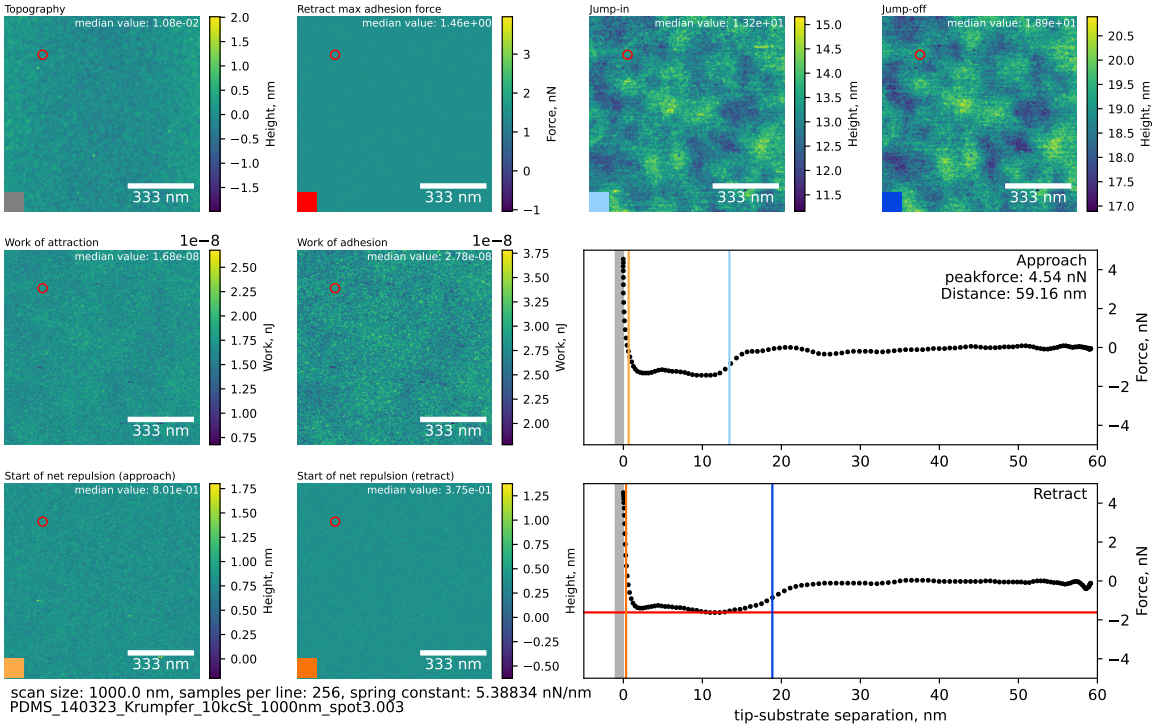

Fig. S14. Meniscus force measurements for sample P3 (8 nm thick), 1000 nm scan

C. Code used to process PeakForce data

The code used to process the PeakForce data into the images presented in Fig. 2b and Fig. S8-S14 is presented below. The code is also hosted on GitHub (github.com/igresh/AFMtools), alongside additional functions to assist in data visualisation.

The code reads a Bruker PeakForce files (with extension *.pfc*) via the *PeakForceImport* function, which produces a numpy binary of the force curve data, a numpy binary of the image provided by the AFM (termed 'topography' below) and a *.csv* file of relevant metadata. These output files are saved in a folder in the output directory; the folder shares the name of the original *.pfc* file. The force curves provided in the *.pfc* file have already been converted to force vs. separation curves, with zero separation being defined as the point at which the peakforce was reached. This means that all values calculated from the force curves are relative to the substrate; the substrate topography is also provided in the *.pfc* file.

These initial output files are then processed by the *processForceMap* function, which calculates the parameters below. Where colours are given in parentheses, they correspond to the lines in the force relevant force curves of Fig. S8-S14.

- Maximum attractive (i.e., negative) force on approach
- Maximum attractive (i.e., negative) force on retract (red)
- Jump-in point on approach, interpreted as the start of the liquid layer (light blue)
- Pull-off point on retract (dark blue)
- Point of net repulsion on approach (light orange)
- Point of net repulsion on retract (dark orange)
- 'Work of attraction', via the integral of the force vs. displacement curve between the points of jump-in and net repulsion on approach.
- Work of adhesion, via the integral of the force vs. displacement curve between the points of pull-off and net repulsion on retract.

Imports and helper functions

```
import numpy as np
import os
import csv
import copy
from scipy.signal import savgol_filter

from nanoscope import files
from nanoscope.constants import FORCE, METRIC, VOLTS, PLT_kwargs, RAW

import sys
import ForceCurveFuncs
import ImageFuncs

def save_array(data, name, directory,savecsv=True):
    np.save(f'{directory}/{name}.npy', data)
    if savecsv:
        np.savetxt(f'{directory}/{name}.csv', data, delimiter=',')

def get_bounds(A, B):
    bmin = np.min([np.mean(A) - np.std(A), np.mean(B) - np.std(B)])
    bmax = np.max([np.mean(A) + np.std(A), np.mean(B) + np.std(B)])
    return bmin, bmax
```

Opening .pfc files

```
def PeakforceImport(Filename, output_dir='./Output'):
    """
    Uses the nanoscope module (provided by bruker) to open the peakforce data files.

    Creates a new directory in the output_dir folder using the name of the peakforce
    file. The directory contains the following files:
        -   image.npy (numpy binary) the raw image from the 'image channel'.
            A NxN array containing the topography (height) of the substrate, where N
            is the scan size

        -   qnmcurves.npy (numpy binary) the force curves for the image.
            A NxNxYx2 array where N is the scan size and Y is the number of datapoints
            in each force curve.

        -   values.csv (csv file) Relevant imaging parameters.
    """
    arr = []

    if not os.path.exists(output_dir):
        print ('Creating output directory...')
        os.mkdir(output_dir)

    name = '.'.join(os.path.basename(Filename).split('.')[:-1])
    print (name)

    if not os.path.exists(f'{output_dir}/{name}'):
        os.mkdir(f'{output_dir}/{name}')

    with files.PeakforceCaptureFile(Filename) as file:
        image_channel = file.image_channel
        fv_image, ax_properties = image_channel.create_image(METRIC)

        spl = image_channel.samples_per_line
        lines = image_channel.number_of_lines

        fc_channel = file.force_curves_channel
        fv_pixels = fc_channel.number_of_force_curves

        for pix_idx in range(fv_pixels):
            fz_plot_bl, _ = fc_channel.create_force_z_plot(pix_idx, FORCE)
            fs_plot_bl = fc_channel.compute_separation(fz_plot_bl, FORCE)

            arr.append([fs_plot_bl.trace.x,
                        fs_plot_bl.trace.y,
                        fs_plot_bl.retrace.x,
```

```
                        fs_plot_bl.retrace.y])

        values_of_interest = {'image scan size':image_channel.scan_size,
                              'image scan size unit':image_channel.scan_size_unit,
                              'image line number':image_channel.number_of_lines,
                              'samples per line':image_channel.samples_per_line,
                              'spring constant':image_channel.spring_constant,
                              'optical sensitivity?':image_channel.z_scale_in_sw_units}

    arr = np.array(arr, dtype=np.float16 )

    np.save(f'{output_dir}/{name}/qnmcurves', arr)
    save_array(data=fv_image, name='image', directory=f'{output_dir}/{name}')

    with open(f'{output_dir}/{name}/values.csv', 'w', newline="", encoding='utf-8') as f:
        writer = csv.DictWriter(f, fieldnames=values_of_interest.keys())
        writer.writeheader()
        writer.writerow(values_of_interest)
```

Processing force curves

```python
def processForceMap(direc):
    with open(f'{direc}/values.csv', "r", encoding='utf-8') as infile:
        reader = csv.DictReader(infile)

        for row in reader:
            val_dict = row

    scan_size = float(val_dict['image scan size'])
    scan_unit = val_dict['image scan size unit']

    arr = np.load(f'{direc}/qnmcurves.npy')

    ExtendsForce = copy.deepcopy(arr[:,0:2])
    RetractsForce = copy.deepcopy(arr[:,2:4])

    baseline_av = np.average(ExtendsForce[:,1,:50], axis=1)
    points_per_line = int(np.sqrt(len(arr)))

    ExtendsForce[:,1] = (ExtendsForce[:,1].T - baseline_av.T).T
    RetractsForce[:,1] = (RetractsForce[:,1].T - baseline_av.T).T

    ExtendsForce[:,1] = ExtendsForce[:,1,::-1]
    ExtendsForce[:,0] = ExtendsForce[:,0,::-1]


    sav_params = {'window_length':5, 'polyorder':1}
    ExtendsForce[:,1] = savgol_filter(ExtendsForce[:,1], **sav_params)
    RetractsForce[:,1] = savgol_filter(RetractsForce[:,1], **sav_params)


    jump_in = []
    pull_off = []
    wadh_in = []
    wadh_off = []
    rep_on  = []
    rep_off = []

    for idx, [EF, RF] in enumerate(zip(ExtendsForce, RetractsForce)):
        # Calculate jump in
        mask = EF[1]<np.min(EF[1,:50])*0.5
        mask[70:] = False
        if np.sum(mask)==0:
            jump_in.append(0)
        else:
            jump_in.append(np.quantile(EF[0][mask],q=0.98))
```

```
46
47          # Calculate work of attraction
48          mask = EF[0] < jump_in[-1]+10
49          newEF = np.copy(EF)
50          newEF[1][newEF[1]>0]=0
51
52          if np.sum(mask)==0:
53              wadh_in.append(0)
54          else:
55              wadh = np.trapz(newEF[1], newEF[0])
56              if not wadh == np.inf:
57                  wadh_in.append(wadh)
58              else:
59                  print ('inf')
60                  wadh_in.append(0)
61
62
63          # Calculate the location of repulsive onset
64          mask = np.logical_and(EF[0] < jump_in[-1], EF[1]>0)
65
66          if np.sum(mask)==0:
67              rep_on.append(0)
68          else:
69              idx_at_int = np.argwhere(mask)[-1][0]
70              intercept = np.interp(x=0, xp=EF[1,idx_at_int:idx_at_int+2], fp=EF[0,idx_at_int
    :idx_at_int+2])
71              rep_on.append(intercept)
72
73
74          # calculate the point of pull-off
75          mask = RF[1]<np.min(RF[1,:50])*0.5
76          mask[70:] = False
77          if np.sum(mask)==0:
78              pull_off.append(0)
79          else:
80              pull_off.append(np.quantile(RF[0][mask],q=0.98))
81
82
83          # Calculate work of adhesion
84          mask = RF[0] < pull_off[-1]+10
85          newRF = np.copy(RF)
86          newRF[1][newRF[1]>0]=0
87
88          if np.sum(mask)==0:
89              wadh_off.append(0)
90          else:
```

```
            wadh = np.trapz(newRF[1], newRF[0])
            if not wadh == np.inf:
                wadh_off.append(wadh)
            else:
                print ('inf')
                wadh_off.append(0)


        # Calculate the location of repulsive offset
        mask = np.logical_and(RF[0] < pull_off[-1], RF[1]>0)
        if np.sum(mask)==0:
            rep_off.append(0)
        else:
            idx_at_int = np.argwhere(mask)[-1][0]
            intercept = np.interp(x=0, xp=EF[1,idx_at_int:idx_at_int+2], fp=EF[0,idx_at_int
    :idx_at_int+2])
            rep_off.append(intercept)


    jump_in  =  np.array(jump_in, dtype=np.float64)
    pull_off =  np.array(pull_off, dtype=np.float64)
    wadh_in  = - 1e-9 * np.array(wadh_in, dtype=np.float64) # report value in nJ
    wadh_off = - 1e-9 * np.array(wadh_off, dtype=np.float64) # report value in nJ
    rep_on   =  np.array(rep_on, dtype=np.float64)
    rep_off  =  np.array(rep_off, dtype=np.float64)


    ExtendsForce = np.reshape(ExtendsForce,(points_per_line,points_per_line,2,-1))
    RetractsForce = np.reshape(RetractsForce,(points_per_line,points_per_line,2,-1))


    ExtendsAdh = -np.min(ExtendsForce, axis=3)[:,:,1]
    RetractsAdh = -np.min(RetractsForce, axis=3)[:,:,1]
    wadh_in_arr = np.squeeze(np.reshape(wadh_in,(points_per_line,points_per_line,-1)))
    wadh_off_arr = np.squeeze(np.reshape(wadh_off,(points_per_line,points_per_line,-1)))
    jump_in_arr = np.squeeze(np.reshape(jump_in,(points_per_line,points_per_line,-1)))
    pull_off_arr = np.squeeze(np.reshape(pull_off,(points_per_line,points_per_line,-1)))
    rep_on_arr = np.squeeze(np.reshape(rep_on,(points_per_line,points_per_line,-1)))
    rep_off_arr = np.squeeze(np.reshape(rep_off,(points_per_line,points_per_line,-1)))

    save_array(data=ExtendsAdh, name='extends_adhesion', directory=direc)
    save_array(data=RetractsAdh, name='retracts_adhesion', directory=direc)
    save_array(data=wadh_in_arr, name='work_of_attraction', directory=direc)
    save_array(data=wadh_off_arr, name='work_of_adhesion', directory=direc)
    save_array(data=jump_in_arr, name='jump_in', directory=direc)
    save_array(data=pull_off_arr, name='jump_off', directory=direc)
```

```
    save_array(data=rep_on_arr, name='net_repulsion_in', directory=direc)
    save_array(data=rep_off_arr, name='net_repulsion_off', directory=direc)

    save_array(data=ExtendsForce, name='extend_force_curves', directory=direc, savecsv=
    False)
    save_array(data=RetractsForce, name='retract_force_curves', directory=direc, savecsv=
    False)
```

---

[1] P. C. Souza, R. Alessandri, J. Barnoud, S. Thallmair, I. Faustino, F. Grünewald, I. Patmanidis, H. Abdizadeh, B. M. Bruininks, T. A. Wassenaar, *et al.*, Martini 3: a general purpose force field for coarse-grained molecular dynamics, Nat. Methods **18**, 382 (2021).

[2] D. H. de Jong, S. Baoukina, H. I. Ingólfsson, and S. J. Marrink, Martini straight: Boosting performance using a shorter cutoff and gpus, Comput. Phys. Commun. **199**, 1 (2016).

[3] R. Alessandri, F. Grünewald, and S. J. Marrink, The martini model in materials science, Adv. Mater. **33**, 2008635 (2021).

[4] G. Rossi, L. Monticelli, S. R. Puisto, I. Vattulainen, and T. Ala-Nissila, Coarse-graining polymers with the martini force-field: polystyrene as a benchmark case, Soft Matter **7**, 698 (2011).

[5] S. Cambiaso, F. Rasera, A. Tinti, D. Bochicchio, Y. Grosu, G. Rossi, and A. Giacomello, Local grafting heterogeneities control water intrusion and extrusion in nanopores, Commun. Mater. **5**, 100 (2024).

[6] G. Rossi, I. G. Elliott, T. Ala-Nissila, and R. Faller, Molecular dynamics study of a martini coarse-grained polystyrene brush in good solvent: structure and dynamics, Macromolecules **45**, 563 (2012).

[7] S. Cambiaso, F. Rasera, G. Rossi, and D. Bochicchio, Development of a transferable coarse-grained model of polydimethylsiloxane, Soft Matter **18**, 7887 (2022).

[8] I. J. Gresham, S. G. Lilley, A. R. Nelson, K. Koynov, and C. Neto, Nanostructure explains the behavior of slippery covalently attached liquid surfaces, Angew. Chem. **62**, e202308008 (2023).

[9] W. M. de Vos and F. A. Leermakers, Modeling the structure of a polydisperse polymer brush, Polymer **50**, 305 (2009).

[10] F. Eisenhaber, P. Lijnzaad, P. Argos, C. Sander, and M. Scharf, The double cubic lattice method: Efficient approaches to numerical integration of surface area and volume and to dot surface contouring of molecular assemblies, J. Comput. Chem. **16**, 273 (1995).

[11] C. Extrand and Y. Kumagai, Contact angles and hysteresis on soft surfaces, J. Colloid Interface Sci. **184**, 191 (1996).

[12] W. E. Lorensen and H. E. Cline, Marching cubes: A high resolution 3d surface construction algorithm, in *Seminal graphics: pioneering efforts that shaped the field* (1998) pp. 347–353.

[13] S. Van der Walt, J. L. Schönberger, J. Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, and T. Yu, scikit-image: image processing in python, PeerJ **2**, e453 (2014).

[14] A. Giacomello, L. Schimmele, and S. Dietrich, Wetting hysteresis induced by nanodefects, Proc. Nat. Acad. Sci. **113**, E262 (2016).

[15] J. F. Joanny and P. G. de Gennes, A model for contact angle hysteresis, J. Chem. Phys. **81**, 552 (1984).

[16] X. Zhou, Y. Wang, X. Li, P. Sudersan, K. Amann-Winkel, K. Koynov, Y. Nagata, R. Berger, and H.-J. Butt, Thickness of nanoscale poly(dimethylsiloxane) layers determines the motion of sliding water drops, Adv. Mater. **36**, 2311470 (2024).