```
SURFACE_DIMENSION 1

DEFINE edge attribute ksp_linear real
DEFINE edge attribute nematic_stiffness real
DEFINE edge attribute rx0 real
DEFINE edge attribute ry0 real
DEFINE edge attribute ksp_nonlinear real
DEFINE edge attribute length0 real
DEFINE edge attribute sqcurve_string_mark integer
DEFINE edge attribute edge_border_left integer
DEFINE edge attribute edge_border_right integer
DEFINE edge attribute edge_border_down integer
DEFINE edge attribute edge_border_up integer

DEFINE vertex attribute x_ref real
DEFINE vertex attribute y_ref real
DEFINE vertex attribute vertex_border_left integer
DEFINE vertex attribute vertex_border_right integer
DEFINE vertex attribute vertex_border_down integer
DEFINE vertex attribute vertex_border_up integer
DEFINE vertex attribute vertex_PG_boundary_left integer
DEFINE vertex attribute vertex_PG_boundary_right integer
DEFINE vertex attribute vertex_PG_boundary_down integer
DEFINE vertex attribute vertex_PG_boundary_up integer

#include "PS_inputs.txt"

// LINEAR Spring Elastic Energy.
// The energy of the associated edge has the form: E =  (length - length0)^2.
// To make this energy scale invariant we have to consider the element modulus 'ksp_linear' is the form
'ksp_linear:=ksp/(length0^2)'.
quantity spring_energy_linear energy method hooke2_energy global element_modulus ksp_linear

// NON-LINEAR Spring Elastic Energy: Worm Like Chain Model (WLC).
// In the current form of the energy, the element modulus 'ksp_nonlinear' must have the form:
'ksp_nonlinear:=(length0^2)*ksp',
// where 'ksp' is conventional spring constant.
// We will use an intrinsic strain: epsilon=(sqrt(x4^2 + x5^2 + x6^2)/length0 -1)
// To avoid an infinite energy we will use the transformation:
// epsilon=(epsilon/(1+exp((epsilon-epsilon_yield)/epsilon_delta))); where epsilon_delta is a very small
number.
// Therefore: epsilon=((sqrt(x4^2 + x5^2 + x6^2)/length0 -1)/(1+exp(((sqrt(x4^2 + x5^2 + x6^2)/length0
-1)-epsilon_yield)/epsilon_delta)))
quantity spring_energy_nonlinear energy method edge_general_integral global element_modulus
ksp_nonlinear
scalar_integrand:
        (epsilonNLS^2)*( \
                                (1/2)*( ((sqrt(x4^2 + x5^2 + x6^2)/length0 -1)/(1+exp(((sqrt(x4^2 + x5^2 +
x6^2)/length0 -1)-epsilon_yield)/epsilon_delta)))/epsilonNLS )^2   \
                               + (1/4)*( \
                                        epsilonNLS/(epsilonNLS-((sqrt(x4^2 + x5^2 + x6^2)/length0
-1)/(1+exp(((sqrt(x4^2 + x5^2 + x6^2)/length0 -1)-epsilon_yield)/epsilon_delta))))   \
                                      - ((sqrt(x4^2 + x5^2 + x6^2)/length0 -1)/(1+exp(((sqrt(x4^2 + x5^2 +
x6^2)/length0 -1)-epsilon_yield)/epsilon_delta)))/epsilonNLS - 1   \
      ) \
```

```
                        ) \

// NEMATIC Elastic energy.
// To make this energy scale invariant we have to consider the element modulus 'nematic_stiffness' in the form 'nematic_stiffness:=ksp/(length0^2)'.
quantity nematic_energy energy method edge_general_integral global element_modulus nematic_stiffness
scalar_integrand:
    (1/2)*( (x4*ry0-rx0*x5) )^2

// BENDING Energy of Glycans.
// The value of this energy per vertex is E=4*(1-cos(theta))/(S1+S2).
// Up to second order we have E=2*theta^2/(S1+S2).
// The bending_energy_glycan.modulus must follows the formula:
// When activated bending_energy_glycan.modulus:=0.5 (instead of the usual 1.0).
// The energy modulus must follows the formula:
// bending_energy_glycan.modulus:=0.5*3.75*kg*(Lg^3), where kg is the glycan spring constant and Lg is the glycan natural length.
quantity bending_energy_glycan energy method sqcurve_string_marked global

constraint 2 //left-boundary of the frame
formula: x-frame_position_left = 0

constraint 3 //rigth-boundary of the frame
formula: x-frame_position_right = 0

constraint 4 //lower-boundary of the frame
formula: y-frame_position_down = 0

constraint 5 //upper-boundary of the frame
formula: y-frame_position_up = 0

#include "network_datafile(N=200).txt"

read

#include "PS_models.txt"
#include "PS_calc.txt"
#include "PS_saving.txt"
#include "PS_loading.txt"
#include "PS_comands.txt"
#include "config_directories.txt"
#include "PS_moduli.txt"
```