```
//__________________________________________________________
//  1.- look_convergence
//--------------------------------------------------
look_convergence:= {
delta_energy_loop:=total_energy; g 10; delta_energy_loop:=delta_energy_loop-total_energy;
IF abs(total_energy) >= energy_tolerance THEN { IF abs(delta_energy_loop/total_energy) <=
convergence_ratio \
                                   THEN {convergence:=1} ELSE {convergence:=0} } \
                              ELSE { convergence:=1 };
}//_________________________________________________________


//__________________________________________________________
//  2.- fast_convergence
//--------------------------------------------------
fast_convergence:= {
HESSIAN_SEEK; HESSIAN_SEEK; HESSIAN_SEEK;
HESSIAN_SEEK; HESSIAN_SEEK; HESSIAN_SEEK;
}//_________________________________________________________


//__________________________________________________________
//  3.- set_special_attributes_frame
//--------------------------------------------------
set_special_attributes_frame:= {
set vertex vertex_border_left 0;
set vertex vertex_border_left 1 where vertex_family == 2;
set vertex vertex_border_right 0;
set vertex vertex_border_right 1 where vertex_family == 3;
set vertex vertex_border_down 0;
set vertex vertex_border_down 1 where vertex_family == 4;
set vertex vertex_border_up 0;
set vertex vertex_border_up 1 where vertex_family == 5;

set edge edge_border_left 0;
set edge edge_border_left 1 where edge_family == 2;
set edge edge_border_right 0;
set edge edge_border_right 1 where edge_family == 3;
set edge edge_border_down 0;
set edge edge_border_down 1 where edge_family == 4;
set edge edge_border_up 0;
set edge edge_border_up 1 where edge_family == 5;

set vertex vertex_PG_boundary_left 0;
set vertex vertex_PG_boundary_right 0;
set vertex vertex_PG_boundary_down 0;
set vertex vertex_PG_boundary_up 0;

foreach edge ee do{ IF ee.edge_border_left == 1 THEN {tail_id:=ee.vertices[1].id;
vertex[tail_id].vertex_PG_boundary_left:= 1};
    IF ee.edge_border_right == 1 THEN {tail_id:=ee.vertices[1].id;
vertex[tail_id].vertex_PG_boundary_right:= 1};
             IF ee.edge_border_down == 1 THEN {tail_id:=ee.vertices[1].id;
vertex[tail_id].vertex_PG_boundary_down:= 1};
             IF ee.edge_border_up == 1 THEN {tail_id:=ee.vertices[1].id;
```

```
vertex[tail_id].vertex_PG_boundary_up:= 1};
};


}//________________________________________________________________


//________________________________________________________________
//  4.- SpringNetwork_PG
//---------------------------------------------------
SpringNetwork_PG:={

set edge sqcurve_string_mark 0;
set edge sqcurve_string_mark 1 where edge_family == 0;

set edge hooke_size length0;

// Note that ’ksp_glycan’ and ’ksp_probe’ will be devided by (length0^2) to make the spring energy scale
invariant.
// On the other hand ’ksp_peptide’ is multiplied by (length0^2) because the energy (WLC) is scale
invariant.

foreach edge ee do{ IF ee.edge_family == 0 THEN {ee.ksp_linear:=ksp_glycan/(length0^2);
ee.ksp_nonlinear:=0};
     IF ee.edge_family == 1 THEN {ee.ksp_linear:=0; ee.ksp_nonlinear:=(length0^2)*ksp_peptide};
    };
}//________________________________________________________________


//________________________________________________________________
//  5.- SpringNetwork_frame
//---------------------------------------------------
SpringNetwork_frame:={

set edge hooke_size length0;

// Note that ’ksp_probe’ will be divided by (length0^2) to make spring_energy_linear scale invariant.
// Note that ’bending_stiffness_probe’ will be divided by (length0^2) to make nematic_energy scale
invariant.

foreach edge ee do{ IF ee.edge_family == 2 THEN
{ee.nematic_stiffness:=bending_stiffness_probe/(length0^2);
      ee.ksp_linear:=ksp_probe/(length0^2); ee.ksp_nonlinear:=0};
     IF ee.edge_family == 3 THEN {ee.nematic_stiffness:=bending_stiffness_probe/(length0^2);
      ee.ksp_linear:=ksp_probe/(length0^2); ee.ksp_nonlinear:=0};
     IF ee.edge_family == 4 THEN {ee.nematic_stiffness:=bending_stiffness_probe/(length0^2);
      ee.ksp_linear:=ksp_probe/(length0^2); ee.ksp_nonlinear:=0};
     IF ee.edge_family == 5 THEN {ee.nematic_stiffness:=bending_stiffness_probe/(length0^2);
      ee.ksp_linear:=ksp_probe/(length0^2); ee.ksp_nonlinear:=0};
      };
}//________________________________________________________________


//________________________________________________________________
//  6.- reset_SpringNetwork_frame
```

```
//----------------------------------------------------
// It updates the properties of the frame according to the current configuration (frame deformed
geometry).
// The edge attribute "length0" is reset.
// The edge attribute "hooke_size" is reset.
// The edge attributes "rx0" and "ry0" are reset.
// The edge attribute "nematic_stiffness" is reset.
// The edge attribute "ksp_linear" is reset.

reset_SpringNetwork_frame:={
set edge length0 length where edge_family > 1;

set edge hooke_size length0;

set edge rx0 x;
set edge ry0 y;

// Note that 'ksp_probe' will be divided by (length0^2) to make spring_energy_linear scale invariant.
// Note that 'bending_stiffness_probe' will be divided by (length0^2) to make nematic_energy scale
invariant.

foreach edge ee do{ IF ee.edge_family == 2 THEN
{ee.nematic_stiffness:=bending_stiffness_probe/(length0^2);
      ee.ksp_linear:=ksp_probe/(length0^2); ee.ksp_nonlinear:=0};
     IF ee.edge_family == 3 THEN {ee.nematic_stiffness:=bending_stiffness_probe/(length0^2);
      ee.ksp_linear:=ksp_probe/(length0^2); ee.ksp_nonlinear:=0};
     IF ee.edge_family == 4 THEN {ee.nematic_stiffness:=bending_stiffness_probe/(length0^2);
      ee.ksp_linear:=ksp_probe/(length0^2); ee.ksp_nonlinear:=0};
     IF ee.edge_family == 5 THEN {ee.nematic_stiffness:=bending_stiffness_probe/(length0^2);
      ee.ksp_linear:=ksp_probe/(length0^2); ee.ksp_nonlinear:=0};
      };
}//___________________________________________________


//___________________________________________________
//  7.- ColorMaker
//----------------------------------------------------
ColorMaker:={
set edge color red where edge_family == 0;
set edge color black where edge_family == 1;
set edge color white where edge_family == 2;
set edge color white where edge_family == 3;
set edge color white where edge_family == 4;
set edge color white where edge_family == 5;
}//___________________________________________________
```