The Genetic Algorithm: Using Biology to Compute Liquid Crystal Director Configurations)

S. Yang
Department of Physics & Astronomy
Swarthmore College
Swarthmore, PA, U.S.A.

Peter J. Collings
Department of Physics & Astronomy
Swarthmore College
Swarthmore, PA, U.S.A.
and
Department of Physics and Astronomy
University of Pennsylvania
Philadelphia, PA, U.S.A.

Table of Contents

Table 1: Definition of Variables Used in the Genetic Algorithm Codes.

Variable	**Definition**
c	vector listing inequality constraints on the director components
c_eq	vector listing equality constraints on the director components
consfcn	name of function containing constraints on the director components
d	distance between the substrates in μm
delta_d	distance between calculation points in μm
deltachi	electric susceptibility anisotropy $\Delta\chi$
dn_x	partial derivative of n_x with respect to z (n_y, n_z also)
fval	value of objfcn after minimization
k_1, k_2, k_3	splay, twist, and bend elastic constants, respectively, in pN
epsilon0	permittivity of free space ϵ_0
factor	value of $\epsilon_0 \Delta\chi (V/d)^2$
lb	vector listing the lower bounds of the components of the director
n	components of the director (both x- and y-components)
nvars	number of calculation points of the director components
n_x	x-components of the director (y-, z-, and r-components also)
n_x_b	x-components of the director plus boundary points (y, z, and r also)
n_x_2	x-components of the director for the second half d (z also)
objfcn	name of function to be minimized
n_x_s	average value of n_x between calculation points (y, z, and r also)
r_s	average value of the radius between calculation points
ub	vector listing the upper bounds of the components of the director
V	voltage applied across the substrates in V
W0	planar anchoring strength in μJ/m^2
y	value of objfcn
y_s	value of the splay contribution of y (twist and bend also)

Hybrid Cell

```
%   Genetic Algorithm to Minimize Free Energy
global nvars delta_d k1 k3
nvars = 9; d = 10; delta_d = d/(nvars+1);
k1 = 6.4; k3 = 10;
objfcn = @energyHybridP;
lb = zeros([1 nvars]); ub = ones([1 nvars]);
consfcn = @constraintsHybridP;
options.FunctionTolerance = 1.0000e-07;
options.ConstraintTolerance = 1.0000e-07;
options.PopulationSize = 200;
options.CrossoverFraction = 0.1;
[n_x,fval] = ga(objfcn,nvars,[],[],[],[],lb,ub,consfcn,
   options);
fval
n_x_b = [1 n_x 0]; n_z_b = sqrt(1 - n_x_b.^2);


%   Free Energy Function to be Minimized
function y = energyHybridP(n_x)
global nvars delta_d k1 k3
n_x_b = [1 n_x 0];
dn_x = diff(n_x_b)/delta_d;
n_z_b = sqrt(1 - n_x_b.^2);
dn_z = diff(n_z_b)/delta_d;
y = delta_d*sum(k3*dn_x.^2 + k1*dn_z.^2)/2;


%   Constraints During the Minimization
function [c,c_eq] = constraintsHybridP(n_x)
c = [];
c_eq = [];
end
```

Frederiks Transition

```
%   Free Energy Function to be Minimized
function y = energyFrederiksP(n_x)
global nvars d delta_d k1 k3 factor
n_x_b = [1 n_x 1];
n_x_s = 0.5*diff(n_x_b) + n_x_b(1:nvars+1);
dn_x = diff(n_x_b)/delta_d;
n_z_s = sqrt(1 - n_x_s.^2);
n_z_b = sqrt(1 - n_x_b.^2);
dn_z = diff(n_z_b)/delta_d;
y = delta_d*sum((k1*dn_z.^2 + k3*dn_x.^2)/2 - (factor*
    n_z_s.^2)/2);


%   Genetic Algorithm to Minimize Half Free Energy
global nvars d delta_d k1 k3 factor
nvars = 10; d = 2; delta_d = d/(nvars+1);
k1 = 6.4; k3 = 10;
epsilon0 = 8.854; deltachi = 11; V = 1.5;
factor = epsilon0*deltachi*V*V/2;
objfcn = @energyFrederiksHalfP;
lb = zeros([1 nvars]); ub = ones([1 nvars]);
consfcn = @constraintsFrederiksHalfP;
options.FunctionTolerance = 1.0000e-07;
options.ConstraintTolerance = 1.0000e-07;
options.PopulationSize = 200;
options.CrossoverFraction = 0.8;
[n_x,fval] = ga(objfcn,nvars,[],[],[],[],lb,ub,consfcn,
    options);
n_x_2 = fliplr(n_x(1:nvars-1));
n_x_b = [1 n_x n_x_2 1]; n_z_b = sqrt(1 - n_x_b.^2);
```

Twist Cell

```
%   Free Energy Function to be Minimized
function y = energyTwistCellP(n_x)
global nvars delta_d
W0 = 2; k2 = 4; n_y = sqrt(1-n_x.^2);
n_x_s = 0.5*diff(n_x) + n_x(1:nvars-1);
dn_x = diff(n_x)/delta_d;
n_y_s = sqrt(1-n_x_s.^2);
dn_y = diff(n_y)/delta_d;
y_V = (delta_d/2)*k2*sum((n_y_s.*dn_x-n_x_s.*dn_y).^2);
y_S = (W0/2)*(n_y(1)^2 + n_x(nvars)^2);
y = y_V + y_S;
```

Escaped Radial Cylinder

```
%   Free Energy Function to be Minimized
function y = energyEscRadialP(n_r)
global nvars delta_r k1 k3
n_r_b = [0 n_r 1];
r_s = ([0:nvars] + 0.5)*delta_r;
n_r_s = 0.5*diff(n_r_b) + n_r_b(1:nvars+1);
dn_r= diff(n_r_b)/delta_r;
n_z_s = sqrt(1 - n_r_s.^2);
n_z_b = sqrt(1 - n_r_b.^2);
dn_z = diff(n_z_b)/delta_r;
y = delta_r*sum(r_s.*(k1*((n_r_s./r_s)+dn_r).^2 + k3*
    dn_z.^2))/2;
```

Twisted Nematic Display

```
%   Free Energy Function to be Minimized
function y = energyTNhalfP(n)
global npts nvars d delta_d k1 k2 k3 factor
n_x = n(1:2:nvars-1); n_y = n(2:2:nvars);
n_x_b = [1 n_x]; n_y_b = [0 n_y];
n_x_s = 0.5*diff(n_x_b) + n_x_b(1:npts); dnx = diff(
    n_x_b)/delta_d;
n_y_s = 0.5*diff(n_y_b) + n_y_b(1:npts); dny = diff(
    n_y_b)/delta_d;
n_z_b = sqrt(1 - n_x_b.^2 - n_y_b.^2);
n_z_s = 0.5*diff(n_z_b) + n_z_b(1:npts); dnz = diff(
    n_z_b)/delta_d;
y_s = k1*sum(dnz.^2);
y_t = k2*sum((-n_x_s.*dny + n_y_s.*dnx).^2);
y_b = k3*sum((-n_z_s.*dnx).^2+(-n_z_s.*dny).^2+(n_x_s.*
    dnx+n_y_s.*dny).^2);
y = delta_d*((y_s+y_t+y_b)/2 - sum(factor*n_z_s.^2));


%   Constraints During the Minimization
function [c,c_eq] = constraintsTNhalfP(n)
nvars = length(n);
c1 = zeros(nvars/2:1); c2 = zeros(nvars-2:1); c3 =
    zeros(nvars/2-1:1);
for i = 1:2:nvars-1; c1 = [c1;n(i)^2+n(i+1)^2-1]; end
for i = 1:2:nvars-3; c2 = [c2;n(i+2)-n(i);n(i+1)-n(i+3)
    ]; end
for i = 1:2:nvars-3; c3 = [c3; n(i+3)^2+n(i+2)^2-n(i+1)
    ^2-n(i)^2]; end
c = [c1; c2; c3]
c_eq = [n(nvars-1)-n(nvars)];
```