I wish to deeply acknowledge the Air Force Office of Scientific Research and particularly Dr. Nachman for supporting our work. During two years and nine months of supported research we performed the comprehensive investigation of the optical polariton modes bound to one-dimensional arrays of dielectric particles. The lowest frequency resonant modes in one-dimensional arrays of dielectric spherical particles were studied. We investigated whether they can exist, calculated their lifetimes (quality factors), characterized their propagation and interference. The investigation was performed using the multisphere Mie scattering formalism. We found this problem interesting practically because these systems can be used to manipulate optical energy in the sub-wavelength scale, guide, transfer, filter, store and covert optical energy. High quality factor modes can also be used in microlasers.

The results of our research are summarized in seven published articles [1-7] and one submitted paper [8] available through the Physics Preprint Archive. They were reported as invited talks in International Conferences.[9-11]

According to the guidance criterion bound modes can exist if the half of a resonant mode wavelength exceeds the interparticle distance. If this criterion is satisfied the optical energy cannot leave the particle chain because of the conflict with the energy and momentum conservation laws for photon emission (light cone constraints). We investigated this criterion numerically and demonstrated that propagating modes exist if the refractive index of dielectric particles exceeds *2*.[1-6]

To investigate the efficiency of light binding to the particle chains of various shapes we calculated quality factors of most bound mode depending on the array shape (circular[1,2] or linear chains[3-6]). We demonstrated both analytically and numerically that the quality factor of the most bound mode in circular array depends exponentially on the number of particles in the chain, while in linear chain this quality factor increases as the third power of the number of particles. Disordering crucially affects the mode quality factor in circular arrays while in linear particle chains its effect is less significant.

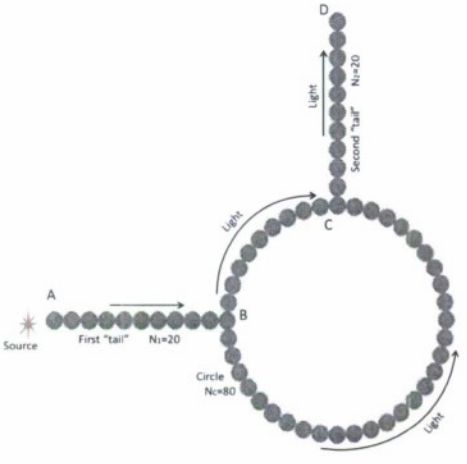


Fig. 1. Traffic circle model to study polariton interference

We demonstrated that modes possessing the highest quality factor represent interesting example of so called slow light modes.[6] These modes have a vanishing group velocity in the limit of the infinite length of the particle chain because they are located in the top edge of a guiding band. Guiding modes form energy bands in periodic chains of particles. The modes in the upper band edge have a zero group velocity so they behave like slow light modes.

In addition to quality factor we investigated the propagation of guiding modes. We demonstrated that if the mode is excited by the point source located near the edge of

# REPORT DOCUMENTATION PAGE

AFRL-SR-AR-TR-09-0023



| 1. REPORT DATE *(07-12-2009)* | 2. REPORT TYPE Final | 3. DATES COVERED *(03/2006-11/2009)* |
|---|---|---|

**4. TITLE AND SUBTITLE** Optical excitations and energy transfer in nanoparticle waveguides

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER FA9550-06-1-0110**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S) Alexander L. Burin**

**5d. PROJECT NUMBER**

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Tulane University
6823 ST CHARLES AVENUE
NEW ORLEANS LA 70118-5698

**8. PERFORMING ORGANIZATION REPORT NUMBER**

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

AFOSR
875 N RANDOLPH ST
ARLINGTON, VA 22203

Dr. Arje Nachman/NE

**10. SPONSOR/MONITOR'S ACRONYM(S)**

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION / AVAILABILITY STATEMENT**

Distribution A: Approved for Public Release

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

The main objective of the present proposal was the comprehensive investigation of the high-quality optical modes bound to one-dimensional arrays of dielectric particles.
Three particular goals were completed during the three years of the project:
1. We investigated the formation of high quality optical modes in linear chains of nanoparticles fabricated from different optical materials optical possessing a relatively high refraction index. We found that when the refractive index exceeds 2 bound modes are always formed and their quality factor increases unlimitedly with increasing the number of particles.
2. We calculated quality factors for most bound modes in linear chain and circular array of particles and demonstrated that it increases exponentially with the number of particles in circular array and as the third power of the number of particles in linear chains.
3. We investigated interference of propagating guiding modes and demonstrated that these modes behave similarly to propagating electrons.
All these goals were attained using multisphere Mie scattering formalism. We developed new codes, which are available online in the group website. The research has led to seven publications and one submitted paper, three invited and two contributed presentations in international conferences.

**15. SUBJECT TERMS**

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Alexander L. Burin |
| | | | | | 19b. TELEPHONE NUMBER *(include area code)* (504) 862-3574 |

the particle chain, then practically all emitted energy is absorbed by this mode that can propagate along the chain without losses if the source frequency belongs to the guiding band.[7]

Finally we investigated whether guiding modes can interfere with each other similarly to electronic waves. We investigated interference in the traffic circle arrays[8] (see Fig. 1) and demonstrated that it behaves similarly to the interference of electronic currents in two wires. It is thus possible to realize the Aharonov Bohm like effects in particle waveguides.

To complete all goals described above we used the multisphere Mie scattering formalism, which permitted us to efficiently solve frequency domain Maxwell equations for the system containing the large number of dielectric spheres. All calculations were performed using our own codes given in the Appendix section. The calculations were performed using Scilab programming package.[12]

The research has been performed with the help of two PI's graduate students Olga Samoylova and Gail Blaustein partially supported by this project. Olga Samoylova is planning to defend her PhD in May 2009 based on her achievements, while Gail Blaustein is currently support by the SMART Fellowship Program and works on the other project associated with DNA optical properties. Some work was made with the help of PI's collaborators Dr. Il'ya Polishchuk (Max Planck Institute for Physics of Complex Systems, Dresden, Germany) and Michael Gozman (Russian Research Center "Kurchatov Institute," Moscow, RUssia) under partial support of this project and the Tulane University Research and Enhancement Fund.

**References.**

1. A. L. Burin, Bound whispering gallery modes in circular arrays of dielectric spherical particles , Phys. Rev. E 73, 066614, 2006.
2. A. L. Burin, G. S. Blaustein, O. M. Samoylova, Bound Whispering Gallery modes in Circular Arrays of Dielectric Spherical Particles, Proceedings of SPIE 6452, Invited Paper, pp. 64520H-1 – 64520H-15 (2007).
3. G. S. Blaustein, A. L. Burin, Optical modes in linear arrays of dielectric spherical particles: A numerical investigation, Proc. of SPIE 6452, pp. 645212-1 – 645212-9 (2007).
4. G. S. Blaustein, M. I. Gozman, I. Y. Polishchuk, A. L. Burin, Optical Modes in Linear Arrays of Dielectric Spherical Particles: A Numerical Investigation, Transparent Optical Network. 2007. ICTON '07, 9th International Conference 4, 136-139 (2007).
5. A. L. Burin, G. S. Blaustein, O. M. Samoylova, Bound Whispering Gallery modes in Arrays of Dielectric Spherical Particles, Invited Paper, Transparent Optical Network. 2007. ICTON '07, 9th International Conference 3, 71-74 (2007).
6. G. S. Blaustein, M. I. Gozman, O. Samoylova, I. Y. Polishchuk, and A. L. Burin, "Guiding optical modes in chains of dielectric particles," Optical Express 15, pp. 17380-17391 (2007).
7. M. I. Gozman, I. Ya. Polishchuk, A. L. Burin, Light propagation in linear arrays of spherical particles, Physics Letters A 372, 5250–5253 (2008).
8. I. Ya. Polishchuk, M. I. Gozman, G. S. Blaustein, A. L. Burin, Interference of guiding polariton mode in "traffic" circle waveguides composed of dielectric

spherical particles, arXiv:0810.4169, http://lanl.arxiv.org/abs/0810.4169, submitted to Physical Review E.
9. 9th International Conference on Transparent Optical Networks, Rome (July 2007, invited Speaker)
10. International Conference on Physics of Microresonators, Charlotte, North Carolina (June 2007, invited Speaker)
11. Photonic West International Symposium of the International Society for Optical Engineering, San Jose, California (January 2007, invited Speaker)
12. Scilab is a free software compatible to the famous Matlab package. It can be found at their webpage http://www.scilab.org/.

**Appendix.** Codes in the Scilab[12] programming language used for our calculations. They can be easily converted to Matlab.

```
function y=Diag_int_A(r, thet, fi, m, n, mu, nu)
//This code calculates vector translation coefficients A as function of polar coordinates for translation
// vector.  m, n; mu, nu – projections of angular momentum for new and old spherical vector functions
m=-m;
//Calculation of prefactor
x1 = ((-1)^m)*exp((%i)*(mu+m)*fi)*(2*nu+1)/(2*n*(n+1));
x2 = gammaln(n-m+1)+gammaln(nu-mu+1)-gammaln(n+m+1)-gammaln(nu+mu+1);
xx = x1*exp(x2);
//disp(xx);
//Calculation of sum
qmax=min(n, nu, (n+nu-abs(m+mu))/2);
//Getting the final result
Summ=0;
//disp(qmax);
for q=0:qmax
p=n+nu-2*q;
uu=ab_GW(m, n, mu, nu, p);
y1=uu(1);
y3=LegendreXu(p, m+mu, cos(thet));
//y2=legendre(p, abs(m+mu), cos(thet));
//cc=m+mu;
//if cc < 0
//y2=y2*((-1)^cc)*exp(gammaln(p+cc+1)-gammaln(p-cc+1));
//end
y22=y3*besselh(p+1/2, r)*y1*sqrt(%pi/(2*r));
//disp(y2);
Summ=Summ+y22;//*(-1)^cc;
end
y=xx*Summ;
endfunction



function y=Diag_int_A_Four(k, q, a, m, n, mu, nu, Nmax)
//This code calculate Fourier transform of A for linear chain of period a

Summ=0;
for ii=1:Nmax
```

```
 Summ=Summ+exp(%i*k*a*ii)*Diag_int_A(a*ii*q, 0, 0, m, n, mu, nu)+exp(-
%i*k*a*ii)*Diag_int_A(a*ii*q, %pi, 0, m, n, mu, nu);
end
y=Summ;
endfunction

function y=Diag_int_A_Four_xy_sph(mm, q, a, m, n, mu, nu, N)
//This code calculate Fourier transform of A for the spherical array in x-y plane

Summ=0;
R=a*N/(2*%pi);
for ii=1:N-1
//  Summ=Summ+exp(%i*mm*ii*2*%pi/N)*Diag_int_A(2*q*R*sin(ii*%pi/N), %pi/2, ii*%pi/N, m, n,
mu, nu);
   Summ=Summ+exp(%i*mm*ii*2*%pi/N)*Diag_int_A(2*q*R*sin(ii*%pi/N), %pi/2, 0, m, n, mu, nu);
//  disp(Summ);
end
y=Summ;
endfunction



function y=ab_GW(m, n, mu, nu, p)
// Function used in evaluation of diagonal vector translation coefficients
// Calculation of diagonal a-part
// and off-diagonal b-part
z=(-1)^(m+mu)*(2*p+1)*Wigner3j(n, nu, p, 0, 0, 0)*Wigner3j(n, nu, p, m, mu, -m-mu);
zz=gammaln(n+m+1)+gammaln(nu+mu+1)+gammaln(p-m-mu+1)-gammaln(n-m+1)-gammaln(nu-mu+1)-
gammaln(p+m+mu+1);
zz=zz/2;
yy=z*exp(zz);

//Calculation of off-diagonal b-part
z1=(-1)^(m+mu)*(2*p+3)*Wigner3j(n, nu, p, 0, 0, 0)*Wigner3j(n, nu, p+1, m, mu, -m-mu);
zz1=gammaln(n+m+1)+gammaln(nu+mu+1)+gammaln(p-m-mu+2)-gammaln(n-m+1)-gammaln(nu-
mu+1)-gammaln(p+m+mu+2);
zz1=zz1/2;
yy1=z1*exp(zz1);
 y(1)=yy*((%i)^p)*(n*(n+1)+nu*(nu+1)-p*(p+1));
y(2)=yy1*((%i)^(p+1))*sqrt(((p+1)^2-(n-nu)^2)*((n+nu+1)^2-(p+1)^2));

endfunction

function y=Wigner3j(j1, j2, j3, m1, m2, m3)
// evaluation of WIgner 3j-symbols used in vector translation coefficients

if (m1+m2+m3 ~=0)
z=0;
else
kmax = min(j1+j2-j3, j1-m1, j2+m2);
kmin = max(0, j2-j3-m1, j1-j3+m2)
z1=(-1)^(j1+j2+m3);
z2 = gammaln(j1-m1+1)+gammaln(j1+m1+1)+gammaln(j2-m2+1)+gammaln(j2+m2+1)+ gammaln(j3-
m3+1)+gammaln(j3+m3+1);
z2=z2-gammaln(j1+j2-j3+1)-gammaln(j1-j2+j3+1)-gammaln(-j1+j2+j3+1)-gammaln(j1+j2+j3+2);
z2=z2/2;
```

```
Summ=0;
for k=kmin:kmax
z3 = gammaln(j1+j2-j3+1)+gammaln(j1-j2+j3+1)+gammaln(-j1+j2+j3+1)-gammaln(k+1)-gammaln(j1+j2-
j3-k+1);
z3=z3-gammaln(j1-m1-k+1)-gammaln(-j2+j3+m1+k+1) - gammaln(j2+m2-k+1)-gammaln(-j1+j3-
m2+k+1);
tt=((-1)^k)*exp(z3+z2);
Summ=Summ+tt;
end
z=Summ*z1;
end
y=z
endfunction


function y=LegendreXu(l, m, x)
//Redefinition of Legendre polynomials in accordance with the multisphere Mie scattering formalism
if abs(x)~=1
if (m>=0)
z=((-1)^m)*legendre(l, m, x);
else
m=-m;
z=((-1)^m)*legendre(l, m, x)*((-1)^m)*exp(gammaln(l-m+1)-gammaln(l+m+1));
end
else
if m~=0
z=0;
else
if x==1
z=1;
else
z=(-1)^l;
end
end
end
y=z;
endfunction

function y=LegendreXuDer(l, m, x)
//Evaluation of of Legendre polynomial derivatives
if abs(x)~=1
 if l==0
  z1=0;
 else
  if abs(m)<l
   z1=(l*x*LegendreXu(l, m, x)-(l+abs(m))*LegendreXu(l-1, m, x))/sqrt(1-x^2);
  else
   z1=(l*x*LegendreXu(l, m, x))/sqrt(1-x^2);
  end
 end
else
 if abs(m)~=1
 z1=0;
 else
 z1 = l*(l+1)/2;
 z1 = z1*LegendreXu(l, m, 1/2)/LegendreXu(l, abs(m), 1/2)*(x)^(l+1);
```

```
 end
end
y=z1;
endfunction

//Alternative approaches to vector translation coefficients A

function y=Diag_int_A1(r, thet, fi, m, n, mu, nu)
z1=((-1)^(n+m))*(2*n+1)/(2*n*(n+1))*exp(%i*(mu-m)*fi);
qmax=min(n, nu, (n+nu-abs(m-mu))/2);
Summ=0;
for q=0:qmax
p=n+nu-2*q;
Summ=Summ+Gaunt1(-m, n, mu, nu, n+nu-2*q)*((-1)^q)*(n*(n+1)+nu*(nu+1)-
p*(p+1))*sqrt(%pi/(2*r))*besselh(p+1/2, r)*legendre(p, mu-m, cos(thet));
end
y=Summ*z1;
endfunction

function y=Diag_int_A2(r, thet, fi, m, n, mu, nu)
z1=((-1)^(n+m))*exp(%i*(mu-m)*fi)*sqrt((2*n+1)*(2*nu+1)/(n*(n+1)*nu*(nu+1)));
z2 = (gammaln(n+m+1)+gammaln(nu-mu+1)-gammaln(n-m+1)-gammaln(nu+mu+1))/2;
z1=z1*exp(z2);
qmax=min(n, nu, (n+nu-abs(m-mu))/2);
Summ=0;
for q=0:qmax
p=n+nu-2*q;
Summ=Summ+Gaunt1(-m, n, mu, nu, n+nu-2*q)*((%i)^p)*(n*(n+1)+nu*(nu+1)-
p*(p+1))*sqrt(%pi/(2*r))*besselh(p+1/2, r)*legendre(p, mu-m, cos(thet));
end
y=Summ*z1;
endfunction

function y=OffDiag_int_B(r, thet, fi, m, n, mu, nu)
//This code calculates vector translation coefficients B as function of polar coordinates for translation
// vector.  m, n; mu, nu – projections of angular momentum for new and old spherical vector functions
m=-m;
//Calculation of prefactor
x1 = ((-1)^(m+1))*exp((%i)*(mu+m)*fi)*(2*nu+1)/(2*n*(n+1));
x2 = gammaln(n-m+1)+gammaln(nu-mu+1)-gammaln(n+m+1)-gammaln(nu+mu+1);
xx = x1*exp(x2);
//disp(xx);
//Calculation of sum
qmax=min(n, nu, (n+nu+1-abs(m+mu))/2);
//Getting the final result
Summ=0;
//disp(qmax);
for q=1:qmax
p=n+nu-2*q;
//disp(p);
uu=ab_GW(m, n, mu, nu, p);
y1=uu(2);
//disp(y1);
y3=LegendreXu(p+1, m+mu, cos(thet));
//y2=legendre(p+1, abs(m+mu), cos(thet));
//cc=m+mu;
```

```
//if cc < 0
//y2=y2*((-1)^cc)*exp(gammaln(p+1+cc+1)-gammaln(p+1-cc+1));
//disp(y2);
//end
y22=y3*besselh(p+3/2, r)*y1*sqrt(%pi/(2*r));
//disp(y2);
Summ=Summ+y22;//*(-1)^cc;
end
y=xx*Summ;
endfunction

function y=OffDiag_int_B_Four_xy_sph(mm, q, a, m, n, mu, nu, N)
//This code calculate Fourier transform of B for the spherical array in x-y plane

Summ=0;
R=a*N/(2*%pi);
for ii=1:N-1
   Summ=Summ+exp(%i*mm*ii*2*%pi/N)*OffDiag_int_B(2*q*R*sin(ii*%pi/N), %pi/2, 0, m, n, mu,
nu);
end
y=Summ;
endfunction

function y=Mie1(n, z, n_r)
//Mie scattering coefficient for electric type scattering
num=(ricatti_besselj(n, z)*ricatti_besselj_der(n, n_r*z)-n_r*ricatti_besselj_der(n, z)*ricatti_besselj(n,
n_r*z));
den=(ricatti_besselh(n, z)*ricatti_besselj_der(n, n_r*z)-n_r*ricatti_besselh_der(n, z)*ricatti_besselj(n,
n_r*z));
a_n=num/den;
y=a_n
endfunction

function y=Mie2(n, z, n_r)
//Mie scattering coefficient for magnetic type scattering
num=(ricatti_besselj(n, z)*ricatti_besselj_der(n, n_r*z)*n_r-ricatti_besselj_der(n, z)*ricatti_besselj(n,
n_r*z));
den=(ricatti_besselh(n, z)*ricatti_besselj_der(n, n_r*z)*n_r-ricatti_besselh_der(n, z)*ricatti_besselj(n,
n_r*z));
b_n=num/den;
y=b_n
endfunction

//Functions below use the generalized Newton-Raphson algorithm to calculate quality factor for various
//modes

function y=SolveMie2DipSph(n_r, n, m, N, mm, q_0)
// Solver for dipolar approach and lowest Mie resonance for closely packed circular array
// n_r -refractive index 2.7 - TiO_2, 3.5 - GaAs, 2 - ZnO
// n=1 - dipoles, m=0 - t1, m=1 - t2, m=-1 - l
// N - number of spheres
// mm=0, 1, ...N angular momentum of mode mm=N/2 - most interesting ease
// q_0 - initial valuc for iteration procedure
// Assuming the distance between sphere is 2, targets are the quality factor and the decay rate for //a=200nm
h=0.0000000001;
q=q_0;
```

```
if m==0
 disp("1st transverse mode t1");
 delt = 1;
 k=0;
 while abs(delt/q) > h^(3/2)
  k=k+1;
  zz = 1/Mie2(1, q ,n_r) + Diag_int_A_Four_xy_sph(mm, q, 2, 0, n, 0, n, N);
//  disp(1/Mie2(1, q, n_r));
//  disp(mm);
//  disp(n);
//  disp(N);
//  disp(Diag_int_A_Four_xy_sph(mm, q, 2, 0, n, 0, n, N));
//  disp(Diag_int_A_Four_xy_sph(50, q, 2, 0, 1, 0, 1, 100));
//  disp(q);
//  disp(zz);
  zz_h=1/Mie2(1, q+h ,n_r) + Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, n, 0, n, N);
  zz_der = (zz_h-zz)/h;
  delt=-zz/zz_der;
  q=q+delt;
  if (k>20)
   delt=0;
  end
 end
 disp(k);
end
if m==1
 disp("2nd transverse mode t2");
 delt = 1;
 k=0;
 while abs(delt/q) > h^(3/2)
  k=k+1;
  Av= (Diag_int_A_Four_xy_sph(mm+1, q, 2, 1, n, 1, n, N)+Diag_int_A_Four_xy_sph(mm-1, q, 2, 1, n,
1, n, N))/2;
  Diff=(Diag_int_A_Four_xy_sph(mm+1, q, 2, 1, n, 1, n, N)-Diag_int_A_Four_xy_sph(mm-1, q, 2, 1, n,
1, n, N))/2;
  Repuls = Diag_int_A_Four_xy_sph(mm, q, 2, 1, n, -1, n, N)*Diag_int_A_Four_xy_sph(mm, q, 2, -1, n,
1, n, N);
  zz = 1/Mie2(1, q, n_r) + Av+sqrt(Diff^2+Repuls);
//  zz = 1/Mie2(1, q, n_r)+Av + Diag_int_A_Four_xy_sph(mm, q, 2, 1, n, 1, n)/2;
  Av_h= (Diag_int_A_Four_xy_sph(mm+1, q+h, 2, 1, n, 1, n, N)+Diag_int_A_Four_xy_sph(mm-1, q+h,
2, 1, n, 1, n, N))/2;
  Diff_h=(Diag_int_A_Four_xy_sph(mm+1, q+h, 2, 1, n, 1, n, N)-Diag_int_A_Four_xy_sph(mm-1, q+h,
2, 1, n, 1, n, N))/2;
  Repuls_h = Diag_int_A_Four_xy_sph(mm, q+h, 2, 1, n, -1, n, N)*Diag_int_A_Four_xy_sph(mm, q+h,
2, -1, n, 1, n, N);
  zz_h = 1/Mie2(1, q+h, n_r) + Av_h+sqrt(Diff_h^2+Repuls_h);
//  zz_h = 1/Mie2(1, q+h, n_r)+Av_h + Diag_int_A_Four_xy_sph(mm, q+h, 2, 1, n, 1, n)/2;
//  zz = 1/Mie2(1, q ,n_r) + Diag_int_A_Four_xy_sph(mm, q, 2, 1, n, 1, n,
N)+sqrt(Diag_int_A_Four_xy_sph(mm, q, 2, 1, n, -1, n, N)*Diag_int_A_Four_xy_sph(mm, q, 2, -1, n, 1,
n, N));
//  zz_h=1/Mie2(1, q+h ,n_r) + Diag_int_A_Four_xy_sph(mm, q+h, 2, 1, n, 1, n,
N)+sqrt(Diag_int_A_Four_xy_sph(mm, q+h, 2, 1, n, -1, n, N)*Diag_int_A_Four_xy_sph(mm, q+h, 2, -1,
n, 1, n, N));
//  zz = 1/Mie2(1, q ,n_r) + Diag_int_A_Four_xy_sph(mm, q, 2, 1, n, 1, n, N)+
Diag_int_A_Four_xy_sph(mm, q, 2, 1, n, -1, n, N)/2;
```

```
//   zz_h=1/Mie2(1, q+h ,n_r) + Diag_int_A_Four_xy_sph(mm, q+h, 2, 1, n, 1, n,
N)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 1, n, -1, n, N)/2;

   zz_der = (zz_h-zz)/h;
   delt=-zz/zz_der;
   q=q+delt;
   if k>=20
    delt=0;
   end
//   disp(q);
//   disp(zz);
//   disp(delt);
 end
 disp(k);
end

if m==-1
 disp("longitudinal mode l");
 delt = 1;
 k=0;
 while abs(delt/q) > h^(3/2)
  k=k+1;
  Av= (Diag_int_A_Four_xy_sph(mm+1, q, 2, 1, n, 1, n, N)+Diag_int_A_Four_xy_sph(mm-1, q, 2, 1, n,
1, n, N))/2;
  Diff=(Diag_int_A_Four_xy_sph(mm+1, q, 2, 1, n, 1, n, N)-Diag_int_A_Four_xy_sph(mm-1, q, 2, 1, n,
1, n, N))/2;
  Repuls = Diag_int_A_Four_xy_sph(mm, q, 2, 1, n, -1, n, N)*Diag_int_A_Four_xy_sph(mm, q, 2, -1, n,
1, n, N);
  zz = 1/Mie2(1, q, n_r) + Av-sqrt(Diff^2+Repuls);
  Av_h= (Diag_int_A_Four_xy_sph(mm+1, q+h, 2, 1, n, 1, n, N)+Diag_int_A_Four_xy_sph(mm-1, q+h,
2, 1, n, 1, n, N))/2;
  Diff_h=(Diag_int_A_Four_xy_sph(mm+1, q+h, 2, 1, n, 1, n, N)-Diag_int_A_Four_xy_sph(mm-1, q+h,
2, 1, n, 1, n, N))/2;
  Repuls_h = Diag_int_A_Four_xy_sph(mm, q+h, 2, 1, n, -1, n, N)*Diag_int_A_Four_xy_sph(mm, q+h,
2, -1, n, 1, n, N);
  zz_h = 1/Mie2(1, q+h, n_r) + Av_h-sqrt(Diff_h^2+Repuls_h);
  zz_der = (zz_h-zz)/h;
  delt=-zz/zz_der;
  q=q+delt;
  if k>20
   delt=0;
  end
 end
 disp(k);
end
if imag(q)~=0
 disp(-real(q)/(2*imag(q)));
end
y=q;
endfunction

function y=SolveMie2abSph(n_r, N, q_0)
// Solver for dipolar approach + off-diagonal interaction and lowest Mie resonance for closely packed
// circular array taking into account the off-diagonal interaction
// n_r -refractive index 2.7 - TiO_2, 3.5 - GaAs, 2 - ZnO
// n=1 - dipoles, m=0 - t1, m=1 - t2, m=-1 - l
```

```
// N - number of spheres
// mm=0, 1, ...N angular momentum of mode mm=N/2 - most interesting case
// q_0 - initial value for iteration procedure
// Assuming the distance between sphere is 2, targets are the quality factor and the decay rate for
// a=200nm
h=0.0000000001;
q=q_0;
n=1;
mm=N/2;
 disp("1st transverse mode t1");
 delt = 1;
 k=0;
 while abs(delt/q) > h^(3/2)
  k=k+1;
  denomin=1/Mie1(1, q, n_r)+Diag_int_A_Four_xy_sph(mm-1, q, 2, 1, n, 1, n, N)-
2*Diag_int_A_Four_xy_sph(mm, q, 2, -1, n, 1, n, N);
  zz = 1/Mie2(1, q ,n_r) + Diag_int_A_Four_xy_sph(mm, q, 2, 0, n, 0, n, N)-
4*OffDiag_int_B_Four_xy_sph(mm-1/2, q, 2, 0, 1, 1, 1, N)^2/denomin ;
  denominh=1/Mie1(1, q+h, n_r)+Diag_int_A_Four_xy_sph(mm-1, q+h, 2, 1, n, 1, n, N)-
2*Diag_int_A_Four_xy_sph(mm, q+h, 2, -1, n, 1, n, N);
  zz_h=1/Mie2(1, q+h ,n_r) + Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, n, 0, n, N)-
4*OffDiag_int_B_Four_xy_sph(mm-1/2, q+h, 2, 0, 1, 1, 1, N)^2/denominh;
  zz_der = (zz_h-zz)/h;
  delt=-zz/zz_der;
  q=q+delt;
  if (k>20)
   delt=0;
  end
 end
 disp(k);
disp(-real(q)/(2*imag(q)));
y=q;
endfunction



function y=SolveMie2StrangeSph(n_r, N, q_0)
// Solver for dipolar approach + off-diagonal interaction and lowest Mie resonance for closely packed
//circular array taking into account the off-diagonal interaction
// n_r -refractive index 2.7 - TiO_2, 3.5 - GaAs, 2 - ZnO
// n=1 - dipoles, m=0 - t1, m=1 - t2, m=-1 - l
// N - number of spheres
// mm=0, 1, ...N angular momentum of mode mm=N/2 - most interesting case
// q_0 - initial value for iteration procedure
// Assuming the distance between sphere is 2, targets are the quality factor and the decay rate for //a=200nm
h=0.0000000001;
q=q_0;
n=1;
mm=N/2;
 disp("1st transverse mode t1");
 delt = 1;
 k=0;
 while abs(delt/q) > h^(3/2)
  k=k+1;
   denomin=1/Mie1(1, q, n_r)+Diag_int_A_Four_xy_sph(mm-1, q, 2, 1, n, 1, n, N)-
2*Diag_int_A_Four_xy_sph(mm, q, 2, -1, n, 1, n, N);
```

```
   zz = 1/Mie2(1, q ,n_r) + Diag_int_A_Four_xy_sph(mm, q, 2, 0, n, 0, n, N)-
4*OffDiag_int_B_Four_xy_sph(mm-1/2, q, 2, 0, 1, 1, 1, N)^2/denomin ;
   //b1-a2 00
   denomin1=1/Mie1(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 0, 2, 0, 2, N);
   zz=zz-OffDiag_int_B_Four_xy_sph(mm, q, 2, 0, 1, 0, 2, N)*OffDiag_int_B_Four_xy_sph(mm, q, 2, 0,
2, 0, 1, N)/denomin1;
   //b1-b3 00
//   denomin2=1/Mie2(3, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 0, 3, 0, 3, N);
//   zz=zz-Diag_int_A_Four_xy_sph(mm, q, 2, 0, 1, 0, 3, N)*Diag_int_A_Four_xy_sph(mm, q, 2, 0, 3, 0,
1, N)/denomin2;
   //b1-a2 02
   denomin3=1/Mie1(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 2, 2, 2, 2, N);
   zz=zz-OffDiag_int_B_Four_xy_sph(mm, q, 2, 0, 1, 2, 2, N)*OffDiag_int_B_Four_xy_sph(mm, q, 2, 2,
2, 0, 1, N)/denomin3;
   //b1-a2 0-2
   denomin4=1/Mie1(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, -2, 2, -2, 2, N);
   zz=zz-OffDiag_int_B_Four_xy_sph(mm, q, 2, 0, 1, -2, 2, N)*OffDiag_int_B_Four_xy_sph(mm, q, 2, -2,
2, 0, 1, N)/denomin4;
   //b1-b3 02
//   denomin5=1/Mie2(3, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 2, 3, 2, 3, N);
//   zz=zz-Diag_int_A_Four_xy_sph(mm, q, 2, 0, 1, 2, 3, N)*Diag_int_A_Four_xy_sph(mm, q, 2, 2, 3, 0,
1, N)/denomin5;
   //b1-b3 0-2
//   denomin6=1/Mie2(3, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, -2, 3, -2, 3, N);
//   zz=zz-Diag_int_A_Four_xy_sph(mm, q, 2, 0, 1, -2, 3, N)*Diag_int_A_Four_xy_sph(mm, q, 2, -2, 3, 0,
1, N)/denomin6;
   //b1-b2 01
  denomin7=1/Mie2(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 1, 2, 1, 2, N);
   zz=zz-Diag_int_A_Four_xy_sph(mm+1/2, q, 2, 0, 1, 1, 2, N)*Diag_int_A_Four_xy_sph(mm-1/2, q, 2,
1, 2, 0, 1, N)/denomin7;
   //b1-b2 0-1
   denomin8=1/Mie2(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, -1, 2, -1, 2, N);
   zz=zz-Diag_int_A_Four_xy_sph(mm-1/2, q, 2, 0, 1, -1, 2, N)*Diag_int_A_Four_xy_sph(mm+1/2, q, 2, -
1, 2, 0, 1, N)/denomin8;

   denominh=1/Mie1(1, q+h, n_r)+Diag_int_A_Four_xy_sph(mm-1, q+h, 2, 1, n, 1, n, N)-
2*Diag_int_A_Four_xy_sph(mm, q+h, 2, -1, n, 1, n, N);
   zz_h=1/Mie2(1, q+h ,n_r) + Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, n, 0, n, N)-
4*OffDiag_int_B_Four_xy_sph(mm-1/2, q+h, 2, 0, 1, 1, 1, N)^2/denominh;
   denomin1h=1/Mie1(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 2, 0, 2, N);
   zz_h=zz_h-OffDiag_int_B_Four_xy_sph(mm, q+h, 2, 0, 1, 0, 2, N)*OffDiag_int_B_Four_xy_sph(mm,
q+h, 2, 0, 2, 0, 1, N)/denomin1h;
//   denomin2h=1/Mie2(3, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 3, 0, 3, N);
//   zz_h=zz_h-Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 1, 0, 3, N)*Diag_int_A_Four_xy_sph(mm, q+h,
2, 0, 3, 0, 1, N)/denomin2h;
   denomin3h=1/Mie1(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 2, 2, 2, 2, N);
   zz_h=zz_h-OffDiag_int_B_Four_xy_sph(mm, q+h, 2, 0, 1, 2, 2, N)*OffDiag_int_B_Four_xy_sph(mm,
q+h, 2, 2, 2, 0, 1, N)/denomin3h;
   denomin4h=1/Mie1(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, -2, 2, -2, 2, N);
   zz_h=zz_h-OffDiag_int_B_Four_xy_sph(mm, q+h, 2, 0, 1, -2, 2, N)*OffDiag_int_B_Four_xy_sph(mm,
q+h, 2, -2, 2, 0, 1, N)/denomin4h;
//   denomin5h=1/Mie2(3, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 2, 3, 2, 3, N);
//   zz_h=zz_h-Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 1, 2, 3, N)*Diag_int_A_Four_xy_sph(mm, q+h,
2, 2, 3, 0, 1, N)/denomin5h;
//   denomin6h=1/Mie2(3, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, -2, 3, -2, 3, N);
```

```
//   zz_h=zz_h-Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 1, -2, 3, N)*Diag_int_A_Four_xy_sph(mm, q+h,
2, -2, 3, 0, 1, N)/denomin6h;
   //b1-b2 01
   denomin7h=1/Mie2(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 1, 2, 1, 2, N);
   zz_h=zz_h-Diag_int_A_Four_xy_sph(mm+1/2, q+h, 2, 0, 1, 1, 2, N)*Diag_int_A_Four_xy_sph(mm-
1/2, q+h, 2, 1, 2, 0, 1, N)/denomin7h;
   //b1-b2 0-1
   denomin8h=1/Mie2(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, -1, 2, -1, 2, N);
   zz_h=zz_h-Diag_int_A_Four_xy_sph(mm-1/2, q+h, 2, 0, 1, -1, 2,
N)*Diag_int_A_Four_xy_sph(mm+1/2, q+h, 2, -1, 2, 0, 1, N)/denomin8h;
   zz_der = (zz_h-zz)/h;
   delt=-zz/zz_der;
   q=q+delt;
   if (k>20)
    delt=0;
   end
 end
 disp(k);
disp(-real(q)/(2*imag(q)));
y=q;
endfunction

function y=VeryStrange(n_r, N, q_0)
// Solver for dipolar approach + off-diagonal interaction and lowest Mie resonance for closely packed
//circular array taking into account the off-diagonal interaction
// n_r -refractive index 2.7 - TiO_2, 3.5 - GaAs, 2 - ZnO
// n=1 - dipoles, m=0 - t1, m=1 - t2, m=-1 - l
// N - number of spheres
// mm=0, 1, ...N angular momentum of mode mm=N/2 - most interesting case
// q_0 - initial value for iteration procedure
// Assuming the distance between sphere is 2, targets are the quality factor and the decay rate for
// a=200nm
h=0.0000000001;
q=q_0;
n=1;
mm=N/2;
 disp("1st transverse mode t1");
 delt = 1;
 k=0;
 while abs(delt/q) > h^(3/2)
  k=k+1;
  // b1-a1 0 - 1-1
  denomin=1/Mie1(1, q, n_r)+Diag_int_A_Four_xy_sph(mm-1, q, 2, 1, n, 1, n, N)-
2*Diag_int_A_Four_xy_sph(mm, q, 2, -1, n, 1, n, N);
  zz = 1/Mie2(1, q ,n_r) + Diag_int_A_Four_xy_sph(mm, q, 2, 0, n, 0, n, N)-
4*OffDiag_int_B_Four_xy_sph(mm-1/2, q, 2, 0, 1, 1, 1, N)^2/denomin ;
  //b1-a2 00
  denomin1=1/Mie1(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 0, 2, 0, 2, N);
  zz=zz-OffDiag_int_B_Four_xy_sph(mm, q, 2, 0, 1, 0, 2, N)*OffDiag_int_B_Four_xy_sph(mm, q, 2, 0,
2, 0, 1, N)/denomin1;
  //b1-b3 00
  denomin2=1/Mie2(3, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 0, 3, 0, 3, N);
  zz=zz-Diag_int_A_Four_xy_sph(mm, q, 2, 0, 1, 0, 3, N)*Diag_int_A_Four_xy_sph(mm, q, 2, 0, 3, 0, 1,
N)/denomin2;
  //b1-a2 02
  denomin3=1/Mie1(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 2, 2, 2, 2, N);
```

```
  zz=zz-OffDiag_int_B_Four_xy_sph(mm, q, 2, 0, 1, 2, 2, N)*OffDiag_int_B_Four_xy_sph(mm, q, 2, 2,
2, 0, 1, N)/denomin3;
  //b1-a2 0-2
  denomin4=1/Mie1(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, -2, 2, -2, 2, N);
  zz=zz-OffDiag_int_B_Four_xy_sph(mm, q, 2, 0, 1, -2, 2, N)*OffDiag_int_B_Four_xy_sph(mm, q, 2, -2,
2, 0, 1, N)/denomin4;
  //b1-b3 02
  denomin5=1/Mie2(3, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 2, 3, 2, 3, N);
  zz=zz-Diag_int_A_Four_xy_sph(mm, q, 2, 0, 1, 2, 3, N)*Diag_int_A_Four_xy_sph(mm, q, 2, 2, 3, 0, 1,
N)/denomin5;
  //b1-b3 0-2
  denomin6=1/Mie2(3, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, -2, 3, -2, 3, N);
  zz=zz-Diag_int_A_Four_xy_sph(mm, q, 2, 0, 1, -2, 3, N)*Diag_int_A_Four_xy_sph(mm, q, 2, -2, 3, 0,
1, N)/denomin6;
  //b1-b2 01
 denomin7=1/Mie2(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, 1, 2, 1, 2, N);
  zz=zz-Diag_int_A_Four_xy_sph(mm+1/2, q, 2, 0, 1, 1, 2, N)*Diag_int_A_Four_xy_sph(mm-1/2, q, 2,
1, 2, 0, 1, N)/denomin7;
  //b1-b2 0-1
  denomin8=1/Mie2(2, q, n_r)+Diag_int_A_Four_xy_sph(mm, q, 2, -1, 2, -1, 2, N);
  zz=zz-Diag_int_A_Four_xy_sph(mm-1/2, q, 2, 0, 1, -1, 2, N)*Diag_int_A_Four_xy_sph(mm+1/2, q, 2, -
1, 2, 0, 1, N)/denomin8;

  denominh=1/Mie1(1, q+h, n_r)+Diag_int_A_Four_xy_sph(mm-1, q+h, 2, 1, n, 1, n, N)-
2*Diag_int_A_Four_xy_sph(mm, q+h, 2, -1, n, 1, n, N);
  zz_h=1/Mie2(1, q+h ,n_r) + Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, n, 0, n, N)-
4*OffDiag_int_B_Four_xy_sph(mm-1/2, q+h, 2, 0, 1, 1, 1, N)^2/denominh;
  denomin1h=1/Mie1(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 2, 0, 2, N);
  zz_h=zz_h-OffDiag_int_B_Four_xy_sph(mm, q+h, 2, 0, 1, 0, 2, N)*OffDiag_int_B_Four_xy_sph(mm,
q+h, 2, 0, 2, 0, 1, N)/denomin1h;
  denomin2h=1/Mie2(3, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 3, 0, 3, N);
  zz_h=zz_h-Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 1, 0, 3, N)*Diag_int_A_Four_xy_sph(mm, q+h, 2,
0, 3, 0, 1, N)/denomin2h;
  denomin3h=1/Mie1(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 2, 2, 2, 2, N);
  zz_h=zz_h-OffDiag_int_B_Four_xy_sph(mm, q+h, 2, 0, 1, 2, 2, N)*OffDiag_int_B_Four_xy_sph(mm,
q+h, 2, 2, 2, 0, 1, N)/denomin3h;
  denomin4h=1/Mie1(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, -2, 2, -2, 2, N);
  zz_h=zz_h-OffDiag_int_B_Four_xy_sph(mm, q+h, 2, 0, 1, -2, 2, N)*OffDiag_int_B_Four_xy_sph(mm,
q+h, 2, -2, 2, 0, 1, N)/denomin4h;
  denomin5h=1/Mie2(3, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 2, 3, 2, 3, N);
  zz_h=zz_h-Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 1, 2, 3, N)*Diag_int_A_Four_xy_sph(mm, q+h, 2,
2, 3, 0, 1, N)/denomin5h;
  denomin6h=1/Mie2(3, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, -2, 3, -2, 3, N);
  zz_h=zz_h-Diag_int_A_Four_xy_sph(mm, q+h, 2, 0, 1, -2, 3, N)*Diag_int_A_Four_xy_sph(mm, q+h,
2, -2, 3, 0, 1, N)/denomin6h;
  //b1-b2 01
  denomin7h=1/Mie2(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, 1, 2, 1, 2, N);
  zz_h=zz_h-Diag_int_A_Four_xy_sph(mm+1/2, q+h, 2, 0, 1, 1, 2, N)*Diag_int_A_Four_xy_sph(mm-
1/2, q+h, 2, 1, 2, 0, 1, N)/denomin7h;
  //b1-b2 0-1
  denomin8h=1/Mie2(2, q+h, n_r)+Diag_int_A_Four_xy_sph(mm, q+h, 2, -1, 2, -1, 2, N);
  zz_h=zz_h-Diag_int_A_Four_xy_sph(mm-1/2, q+h, 2, 0, 1, -1, 2,
N)*Diag_int_A_Four_xy_sph(mm+1/2, q+h, 2, -1, 2, 0, 1, N)/denomin8h;
  zz_der = (zz_h-zz)/h;
  delt=-zz/zz_der;
  q=q+delt;
```

```
//   disp(zz);
   if (k>20)
     delt=0;
   end
  end
  disp(k);
disp(-real(q)/(2*imag(q)));
y=q;
endfunction


function y=round_besselh(n, z)
// Calculates Round Bessel function (diverging wave)
if (n~=2)&(n~=1)&(n~=0)
  x1=besselh(n+1/2, z)/sqrt(%pi*z);
else
  if n==0
    x1 = -%i*exp(%i*z)/z;
  else
    if n==1
      x1 = exp(%i*z)*(-1/z-%i/z^2);
    else
      x1 = exp(%i*z)*(%i/z-3/z^2-3*%i/z^3);
    end
  end
end
y=x1;
endfunction

function y=round_besselh_sp_der(n, z)
// Calculates Round Bessel function derivative (diverging wave)
y=-round_besselh(n+1, z) + round_besselh(n, z)*(n+1)/z;
endfunction

function y=round_besselj(n, z)
// Calculates Round Bessel function (standing wave)
if (n~=2)&(n~=2)&(n~=0)
  x1=besselj(n+1/2, z)/sqrt(%pi*z);
else
  if n==0
    x1 = sin(z)/z;
  else
    if n==1
      x1=sin(z)/z^2 -cos(z)/z;
    else
      x1 = (-sin(z)/z-3*cos(z)/z^2+3*sin(z)/z^3);
    end
  end
end
y=x1;
endfunction


function y=round_besselj_sp_der(n, z)
// Calculates Round Bessel function derivatives (standing wave)
```

```
y=-round_besselj(n+1, z) + round_besselj(n, z)*(n+1)/z;
endfunction

function y=ricatti_besselh(n, z)
// Calculates Ricatti Bessel function (Hankel)
x1=z*besselh(n+1/2, z)/sqrt(%pi*z);
y=x1;
endfunction


function y=ricatti_besselh_der(n, z)
// Calculates Ricatti Bessel function derivative (Hankel)
y=-ricatti_besselh(n+1, z) + ricatti_besselh(n, z)*(n+1)/z;
endfunction

function y=ricatti_besselj(n, z)
// Calculates Ricatti Bessel function (Bessel)
x1=z*besselj(n+1/2, z)/sqrt(%pi*z);
y=x1;
endfunction


function y=ricatti_besselj_der(n, z)
// Calculates Ricatti Bessel function derivative (Bessel)
y=-ricatti_besselj(n+1, z) + ricatti_besselj(n, z)*(n+1)/z;
endfunction
```