Circular dichroism in assemblies of plasmonic nanoparticles — modelling in the dipole approximation

Baptiste Auguié

24th July 2011

The system consists in a three-dimensional arrangement of small ellipsoidal particles in arbitrary orientations, as shown in figure 1. Here the position and orientation of the particles was chosen to follow an helix.

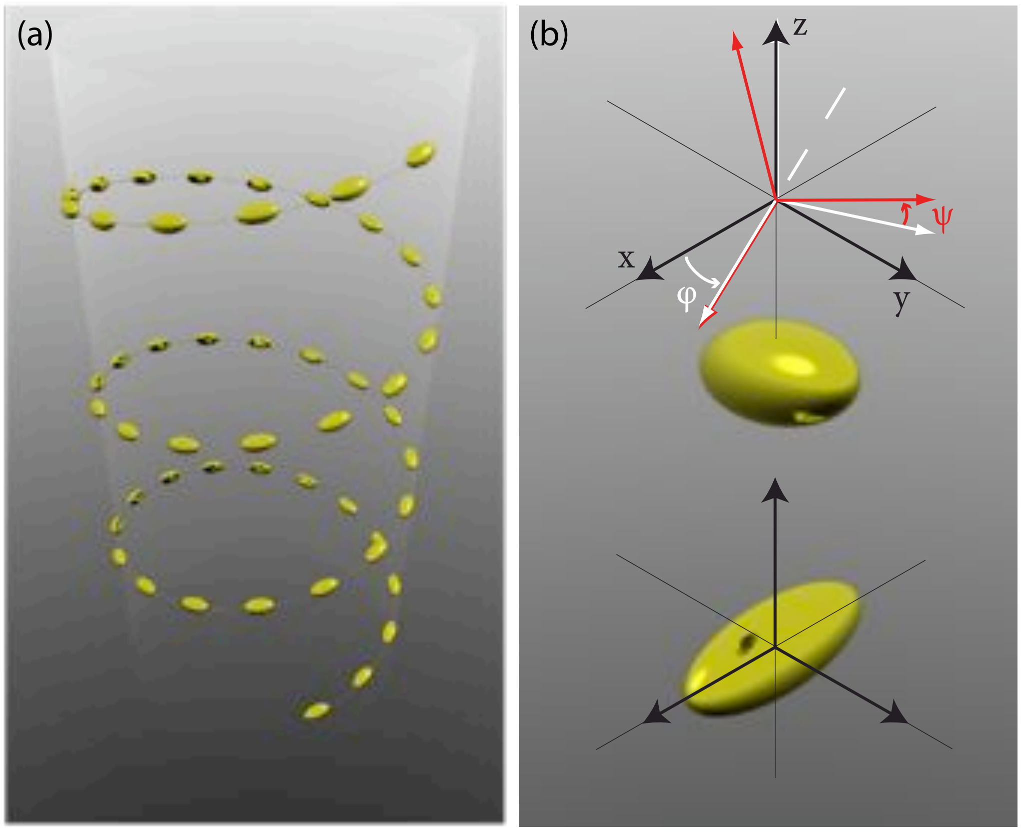

Figure 1: (a) Schematic of the 3D arrangement of particles. (b) Orientation of prolate particles in space using two Euler angles.

1 Coupled dipole model

Each dipole scatters light in proportion to the local field it experiences,

$$\mathbf{p}_{\text{dip}} = \alpha \mathbf{E}_{\text{loc}}, \tag{1}$$

where α is the polarizability tensor describing the individual nanoparticle. The prescription of Kuwata *et al.* [**?**] was chosen, which provides an accurate approximation for particles below ~100 nm in size. Each dipole may be placed in arbitrary orientation, and to this end the polarizability tensor must be transformed to a rotated frame.

Let us describe each nanoparticle by a diagonal polarizability tensor in the reference frame of its principal axes. The rotation of this reference frame to the actual postition of the particle can be described by three Euler angles ϕ, θ, ψ and a rotation matrix [1]

$$\mathrm{R} = \begin{bmatrix} \cos\psi\cos\phi - \cos\theta\sin\phi\sin\psi & \cos\psi\sin\phi + \cos\theta\cos\phi\sin\psi & \sin\psi\sin\theta \\ -\sin\psi\cos\phi - \cos\theta\sin\phi\cos\psi & -\sin\psi\sin\phi + \cos\theta\cos\phi\cos\psi & \cos\psi\sin\theta \\ \sin\phi\sin\theta & -\cos\phi\sin\theta & \cos\theta \end{bmatrix}. \tag{2}$$

A dipole in orientation ϕ, θ, ψ will be described by a polarizability $\mathrm{R}^{-1}\alpha\mathrm{R}$ in the global reference frame.

The local field

$$\mathbf{E}_{\text{loc}} = \mathbf{E}_{\text{inc}} + \sum_{\text{dipoles}\setminus\text{itself}} \mathbf{E}_{\text{d}}, \tag{3}$$

is the sum of the incident field plus the contribution of the dipolar field associated with the other dipoles in the system. The field radiated by a dipole reads,

$$\mathbf{E}_{\text{d}} = \frac{e^{i\omega r/c}}{4\pi\varepsilon_0}\left\{\frac{\omega^2}{c^2 r}\hat{\mathbf{r}}\times\mathbf{p}\times\hat{\mathbf{r}} + \left(\frac{1}{r^3} - \frac{i\omega}{cr^2}\right)[3(\hat{\mathbf{r}}\cdot\mathbf{p})\hat{\mathbf{r}} - \mathbf{p}]\right\}. \tag{4}$$

By grouping together the dipole moments we can cast equation 3 in matrix form,

$$A\mathbf{P} = \mathrm{E}_{\text{inc}}, \tag{5}$$

where A is the interaction matrix that describes the electromagnetic coupling between the dipoles in the non-diagonal blocks,

$$A_{ij} = \frac{e^{(ikr_{ij})}}{r_{ij}}\left\{k^2(\hat{\mathbf{r}}\otimes\hat{\mathbf{r}} - \mathbb{I}) + \frac{ikr_{ij} - 1}{r_{ij}^2}(3\hat{\mathbf{r}}\otimes\hat{\mathbf{r}} - \mathbb{I})\right\}, \tag{6}$$

and the block diagonal $A_{ii} = \alpha^{-1}$ is formed with the inverse polarizability of the individual dipoles, in the global (x, y, z) reference frame.

When the dipole moments are known by inversion of equation 5, the extinction cross-section can be obtained *for a given incident field* following,

$$\sigma_{\text{ext}} = \frac{4\pi k}{|\mathrm{E}_{\text{inc}}|^2}\Im(\mathbf{E}_{\text{inc}}{}^* \cdot \mathbf{P}). \tag{7}$$

1.1 Computation of circular dichroism

Circular dichroism can be calculated from the difference in extinction for left-handed and right-handed circularly polarised light, averaged over the full solid angle of incident light,

$$\sigma_{\text{CD}} = \langle\sigma_L\rangle_\Omega - \langle\sigma_R\rangle_\Omega\,. \tag{8}$$

The incident field incident along x is written as,

$$\mathbf{E}_{\text{inc}} = \frac{\exp i(\omega t - k_x x)}{\sqrt{2}}\begin{pmatrix}0\\ i\\ 1\end{pmatrix} \quad (\textit{right-handed}) \tag{9}$$

$$\mathbf{E}_{\text{inc}} = \frac{\exp i(\omega t - k_x x)}{\sqrt{2}}\begin{pmatrix}0\\ 1\\ i\end{pmatrix} \quad (\textit{left-handed}). \tag{10}$$

More generally, the incident beam will be characterised by a wave-vector $\mathbf{k}$ and an electric vector $\mathbf{E}_{\text{inc}}$ describing the light polarisation, and both of these vectors can be rotated using the rotation matrix R as $\mathrm{R}^{-1}\mathbf{k}$ and $\mathrm{R}^{-1}\mathbf{E}_{\text{inc}}$.

[1] using the same conventions as `http://mathworld.wolfram.com/EulerAngles.html`

The interaction matrix A does not depend on the direction of the incident field, it is therefore advantageous to compute A^{-1} (at each wavelength) and perform the matrix-vector products for all the required incident fields. The CD spectra obtained experimentally are averaged over all orientations of the incident beam, it is therefore necessary to use incident wave-vectors that span the full range of $\phi \in [0, 2\pi], \theta = \pi/2, \psi \in [-\pi/2, \pi/2]$.

Averaging the extinction cross-section over all incident field directions is performed by numerical integration,

$$\langle\sigma\rangle_\Omega = \frac{1}{4\pi}\int_0^{2\pi}\int_{-\pi/2}^{\pi/2} \sigma(\phi,\psi)\cos\psi d\psi d\phi. \tag{11}$$

A Gauss-Legendre quadrature scheme is used to perform the integration, so that the evaluation of the extinction cross-section is performed for a relatively low number of angles (20×20 seems sufficient).

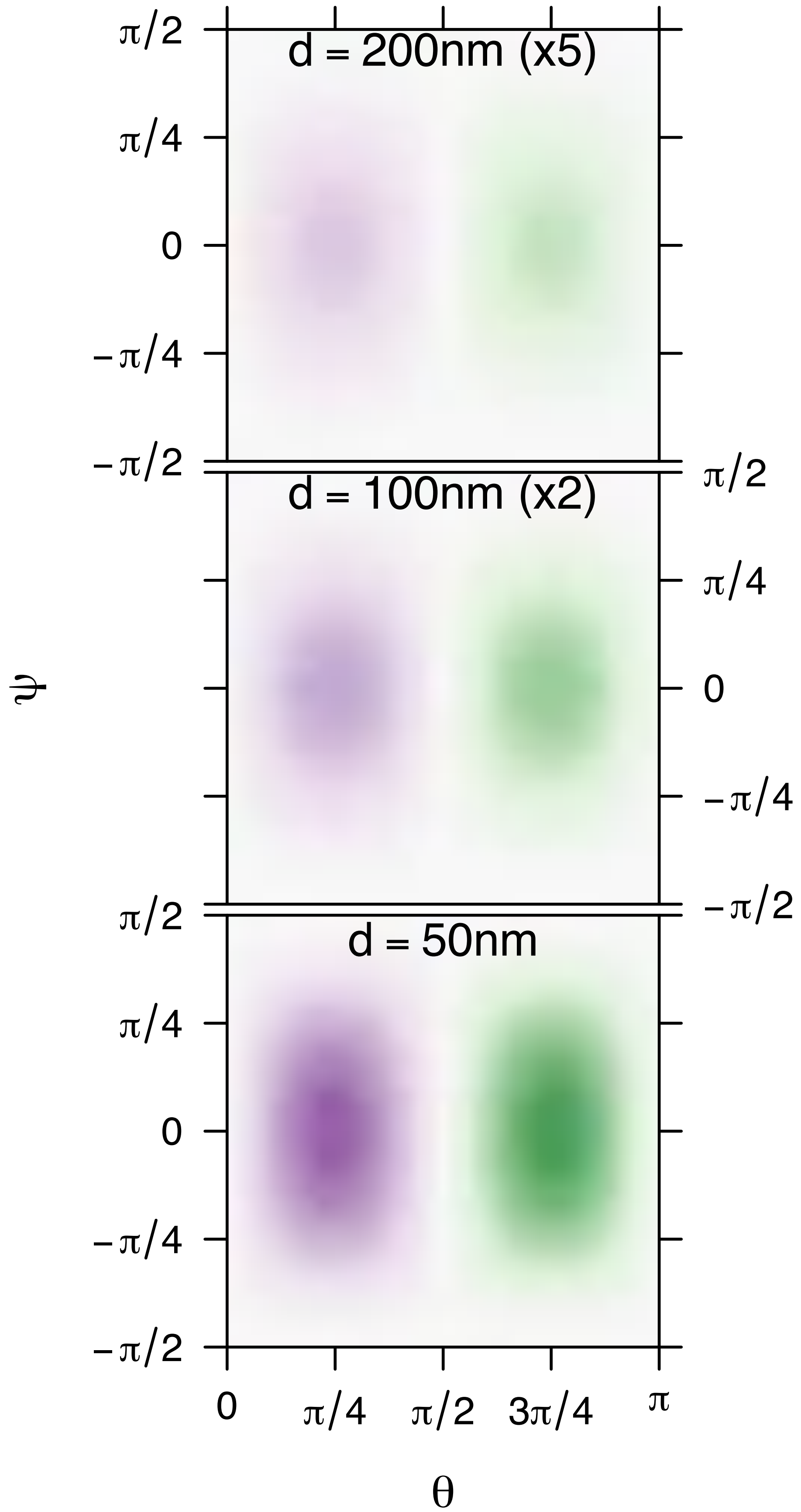

Figure 2: Peak CD value as a function of the dimer angles (θ,ψ) for three different inter-dipole distances. The peak value may be negative (purple) or positive (green). In all cases the strongest CD response is obtained for ($\theta = \pm\pi/4, \psi = 0$) (the result is identical if the average)

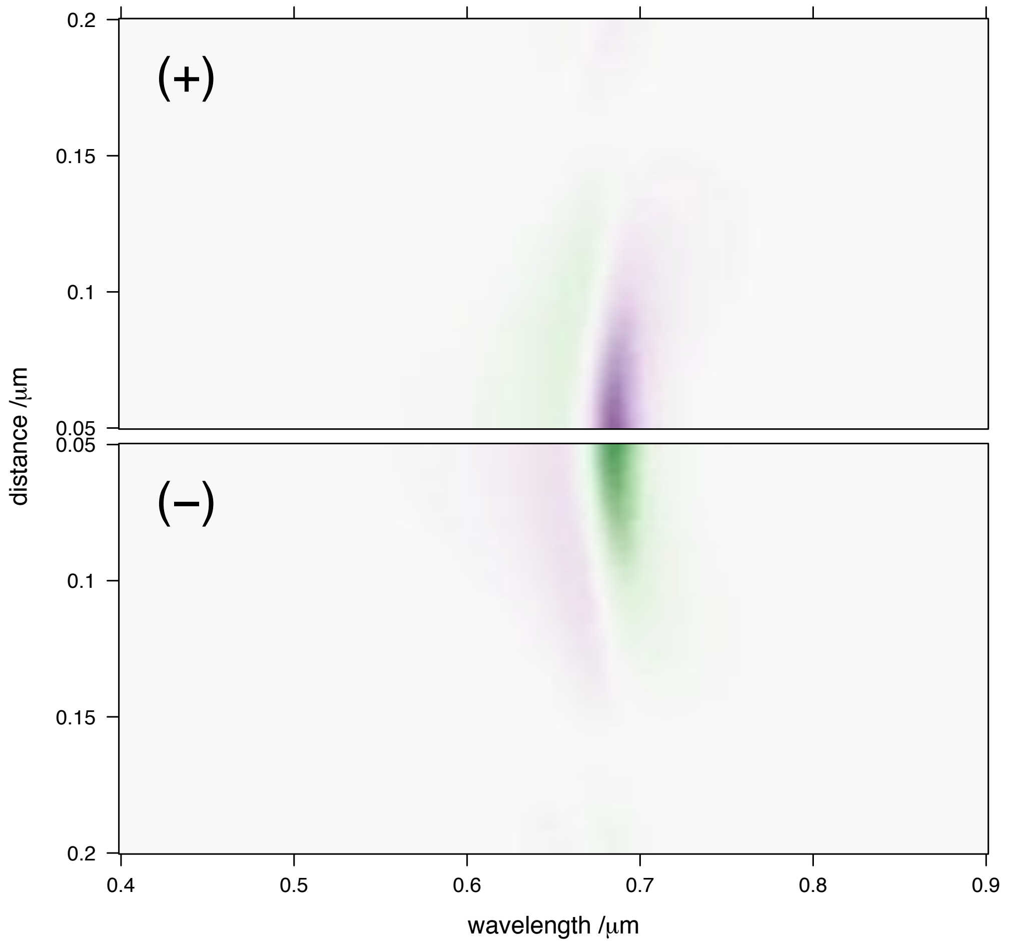

Figure 3: CD spectra as a function of inter-particle distance for a dimer with $(\theta = \pm\pi/4, \psi = 0)$.

C++ source code

This code uses the Armadillo library for complex linear algebra, and provides wrapper functions to use with R. The full code and precompiled binaries are available at R-forge.

Listing 1: Euler rotation matrix

```cpp
arma::mat euler(const double phi, const double theta, const double psi)
 {
   arma::mat Rot(3,3);
   const double cosphi = cos(phi), cospsi = cos(psi), costheta = cos(theta);
   const double sinphi = sin(phi), sinpsi = sin(psi), sintheta = sin(theta);
   Rot(0,0) = cospsi*cosphi - costheta*sinphi*sinpsi;
   Rot(0,1) = cospsi*sinphi + costheta*cosphi*sinpsi;
   Rot(0,2) = sinpsi*sintheta;

   Rot(1,0) = -sinpsi*cosphi - costheta*sinphi*cospsi;
   Rot(1,1) = -sinpsi*sinphi + costheta*cosphi*cospsi;
   Rot(1,2) = cospsi*sintheta;

   Rot(2,0) = sinphi*sintheta;
   Rot(2,1) = -cosphi*sintheta;
   Rot(2,2) = costheta;
   return (Rot);
 }
```

Listing 2: Diagonal polarizability blocks

```cpp
arma::cx_mat diagonal_polarisability(const arma::cx_mat& Alpha, const arma::mat& Euler) {

  const int N = Euler.n_rows;
  const arma::colvec phi = Euler.col(0), theta = Euler.col(1), psi = Euler.col(2);

  arma::mat Rot(3,3);
  arma::cx_mat polar(3*N,3*N);

  int ii=0;
  for(ii=0; ii<N; ii++){

    Rot = euler(phi[ii], theta[ii], psi[ii]);
    polar.submat(ii*3,ii*3,ii*3+2,ii*3+2) = inv(Rot) * diagmat(Alpha.col(ii)) * Rot;

  } // polar is done

  return polar;
}
```

Listing 3: Interaction matrix

```cpp
arma::cx_mat interaction_matrix(const arma::mat& R, const double kn, \
        const arma::cx_mat& invAlpha, const arma::mat& Euler,
        const int full) {

  const int N = R.n_rows;
  // temporary vars
  arma::mat Rot(3,3);
  const arma::cx_double i = arma::cx_double(0,1);
  arma::cx_mat A = arma::zeros<arma::cx_mat>( 3*N, 3*N );
  const arma::cx_mat I3 = arma::eye<arma::cx_mat>( 3, 3 );

  int jj=0, kk=0;
  arma::mat rk_to_rj = arma::mat(1,3), rjkhat = arma::mat(1,3) , rjkrjk = arma::mat(3,3);
  double rjk;
  arma::cx_mat Ajk = arma::cx_mat(3,3);

  // nested for loop over dipole locations
  for(jj=0; jj<N; jj++)
    {
      for(kk=0; kk<N; kk++)
  {
    if(jj!=kk){

            rk_to_rj = R.row(jj) − R.row(kk) ;
            rjk = norm(rk_to_rj,2);
            rjkhat = rk_to_rj / rjk;
            rjkrjk = trans(rjkhat) ∗ rjkhat;
      if(full == 1) {
        Ajk = exp(i∗kn∗rjk) / rjk ∗ (kn∗kn∗(rjkrjk − I3) + (i∗kn∗rjk − arma::cx_double(1,0)) / (rjk∗rjk↩
            ) ∗ (3∗rjkrjk − I3)) ;
      } else {
        Ajk = (I3 − 3∗rjkrjk)/ (rjk∗rjk∗rjk) ;
      }
      // assign block
      A.submat(jj∗3,kk∗3,jj∗3+2,kk∗3+2) = Ajk;
    }
  }
    } // end loops

  // diagonal blocks
  arma::cx_mat polar = diagonal_polarisability(invAlpha, Euler);

  A = A + polar;
  // return inv(A);
  return(A);
}
```

Listing 4: Extinction and absorption cross-sections

```
double extinction(const double kn, const arma::cx_colvec& P, const arma::cx_colvec& Eincident)
{
  const double c = 4∗arma::math::pi()∗kn ∗ imag(cdot(Eincident, P));
  return c;
}

double absorption(const double kn, const arma::cx_colvec& P, const arma::cx_mat& invpolar)
{
  const double c = 4∗arma::math::pi()∗kn∗(imag(cdot(invpolar ∗ P, P)) − \
              kn∗kn∗kn∗ 2/3 ∗ real(cdot(P, P)));
  return c;
}
```

Listing 5: Angular averaging

```

arma::colvec averaging(const arma::mat& R, const arma::cx_mat& A, const arma::cx_mat& invalpha, \
      const double kn, const arma::mat& QuadPhi, const arma::mat& QuadPsi)
  {
```

```
  const int N = R.n_rows, NqPhi = QuadPhi.n_rows, NqPsi = QuadPsi.n_rows;
  const arma::colvec nodes1 = QuadPhi.col(0), weights1 = QuadPhi.col(1), \
    nodes2 = QuadPsi.col(0), weights2 = QuadPsi.col(1);;
  //constants
  const arma::cx_double i = arma::cx_double(0,1);
  const double pi = arma::math::pi();
  arma::mat Rot(3,3);

  // incident field
  const arma::cx_colvec RCP="(0,0) (0,1) (1,0);", LCP="(0,0) (1,0) (0,1);";
  arma::cx_colvec ERCP(3), ELCP(3), Eincident(3*N), P(3*N);
  const arma::colvec khat="1 0 0;", kvec = kn*khat;
  arma::mat kr;
  arma::cx_mat expikr;
  double left=0, right=0, left2=0, right2=0,phi=0,psi=0;

  // begin quadrature
  const double b2=pi/2, a2=2*pi, b1=-pi/2, a1=0; // a is for phi, b for psi
  const double C2 = (b2 - b1) / 2, D2 = (b2 + b1) / 2, \
    C1 = (a2 - a1) / 2, D1 = (a2 + a1) / 2;

  const arma::colvec y = nodes2*C2 + D2; // for psi
  const arma::colvec x = nodes1*C1 + D1; // for phi

  double tmpleft=0, tmpright=0, tmpleft2=0, tmpright2=0, factor, factor2;
  int ll=0,mm=0; // Nq quadrature points

  arma::cx_mat B = pinv(A); /* inverting the interaction matrix
       to solve AP=Eincident multiple times */

  for(ll=0; ll<NqPhi; ll++){ // loop over phi
    phi = x[ll];
    tmpleft=0; tmpright=0; tmpleft2=0; tmpright2=0;
for(mm=0; mm<NqPsi; mm++){ // loop over psi
  psi = y[mm];
  Rot = euler(phi, pi/2, psi); // theta doesn't vary
  ELCP = sqrt(2)/2 * trans(Rot) * LCP ;
  ERCP = sqrt(2)/2 * trans(Rot) * RCP ;
  kr = R * trans(Rot) * kvec;
  expikr = exp(i*kr);

  factor = C1 * weights2[mm] * cos(psi) ;
  // left polarisation
Eincident = reshape(expikr * strans(ELCP), 3*N, 1, 1);
P = B * Eincident;
// P = solve(A, Eincident);// too slow, invert A before loop
tmpleft += factor * extinction(kn, P, Eincident);
tmpleft2 += factor * absorption(kn, P, invalpha);

// right polarisation
Eincident = reshape(expikr * strans(ERCP), 3*N, 1, 1);
P = B * Eincident;
// P = solve(A, Eincident); // too slow, invert A before loop
tmpright += factor * extinction(kn, P, Eincident);
tmpright2 += factor * absorption(kn, P, invalpha);

}
factor2 = C2 * weights1[ll];
left += factor2 * tmpleft;
right += factor2 * tmpright;
left2 += factor2 * tmpleft2;
right2 += factor2 * tmpright2;
    }

  arma::colvec res(4) ;

  res(0) = left / (4*pi); //ext L
  res(1) = right / (4*pi);//ext R
  res(2) = left2 / (4*pi); //abs L
  res(3) = right2 / (4*pi); //abs R
  return res ;
}
```