Supplementary Material: McStas Routine for Neutron Lenses

The aim was an efficient and simple routine for neutron lenses with the essential corrections for geometric aberrations. The formulas for the spherical and parabolic aberration were derived in the last section in equations (A6) and (A12), respectively. For a single lens the dimensions of the lens geometry (typically few cm) are much smaller than the focal width (few m) or the detector distance. So the real refractive incidents at the lens surface can be reduced to a single one. By simulations it was practically shown that even the refraction of N lenses can be reduced to a single refractive incidence. In the limit of small lenses and weakly refractive materials (1 >> $N\xi$ >> h^2/R^2) the two corrections become identical. More realistically, the corrections of off-central beams ($h^2/R^2 \approx$ 0.1-1) are more important than the refractive corrections ($N\xi \approx 10^{-2}$-10^{-3}), and thus the spherical lens is far less favorable than the parabolic lens.

The algorithm shown below lets the neutron propagate to the central plain. There the effective refraction is calculated in the following way: The neutron is virtually traced back to 0.7 times the focal length. From there an image is calculated, and the neutron at the central plain is directed to the image point. This particular solution nonetheless holds generally for all possible cases. The single refractive incidence allows switching on gravitational effects between the entrance aperture and the detector. Possible gravitational corrections inside the lens volume can safely be neglected (since angular changes inside the lens due to gravity are small compared to typical angles of refractive incidents).

McStas-Code of Lens

```
/***********************************************************************
* Component: Lens
* %P
* INPUT PARAMETERS
* rho:      Scattering length density [m-2]
* Rc:       Radius (concave: Rc>0) of biconcave lens     [m]
* Nl:       Number of single lenses [1]
* parab:    Switch (not 0 -> parabolic, for 0 spherical)
* radius:   Radius of slit in the z=0 plane, centered at origin [m]
* SigmaAL:  Absorption cross section per lambda (wavelength) [1/(m*AA)]
* d0:       Minimum material thickness in the centre of the lens [m]
* %E
***********************************************************************/
DEFINE COMPONENT Lens
DEFINITION PARAMETERS ()
SETTING PARAMETERS (rho=5.16e14, Rc=0.02, Nl=7.0, radius=0.05,
                    parab=1e0, SigmaAL=0.141, d0=0.002)
STATE PARAMETERS (x,y,z,vx,vy,vz,t,s1,s2,p)
```

```
INITIALIZE
%{   %}

TRACE
%{  double vvv = vx*vx + vy*vy + vz*vz;
    double Xi  = rho/PI *(4e-20*PI*PI)/(V2Q*V2Q*vvv);
    double foc = Rc/Xi/Nl;

    if (z>0e0 || vvv==0e0) ABSORB;
    PROP_Z0;

    double ss = x*x + y*y;
    if (ss > radius*radius)
      ABSORB;
    else
      SCATTER;
    if (parab==0e0 && ss >= Rc*Rc*(1e0-N1*Xi)) ABSORB;
    if (parab!=0e0 && ss >= Rc*Rc*(1e0+0.5/(N1*Xi)) ABSORB;

    if (parab!=0e0)                     //parab.//
      foc*= 0.5 - 0.5*N1*Xi*ss/(Rc*Rc)
                 + sqrt(0.25-0.5*N1*Xi*(ss-Rc*Rc)/(Rc*Rc));
    else                                //spher.//
      foc*= -N1*Xi + 0.5*sqrt(1e0-ss/(Rc*Rc))
                 + 0.5*sqrt(1e0-2*N1*Xi-ss/(Rc*Rc));

    double tt2 = (-0.7*abs(foc))/vz;  //move virtually back//
    double xx2 = x + vx*tt2;
    double yy2 = y + vy*tt2;
    double zz2 = -0.7*abs(foc);
    double ll1 = -zz2;                //image of this point??//
    double ll2 = 1.0/(1.0/foc-1.0/ll1);
    double llr = -ll2/ll1;            //ratio of images//
    xx2*= llr;                        //image of virtual point//
    yy2*= llr;
    zz2*= llr;
    xx2 = xx2 - x;                    //relative pos to actual pos//
    yy2 = yy2 - y;
    double zdir = zz2/abs(zz2);       //ensure forward movement//
    double xyzlen = xx2*xx2 + yy2*yy2 + zz2*zz2;
    vx = xx2*zdir*sqrt(vvv/xyzlen);   //velocities towards image//
    vy = yy2*zdir*sqrt(vvv/xyzlen);
    vz = zz2*zdir*sqrt(vvv/xyzlen);
```

```
      double thck;
      if (parab!=0e0)
        thck = ss/Rc + d0;
      else
        thck = 2.0*(Rc-sqrt(Rc*Rc-ss)) + d0;

      p*=exp(-Nl*SigmaAL*(2.0*PI/(V2Q*sqrt(vvv)))*thck); // transmission //

%}
```