for 45-50 cm

Group # 1 average flux: 0.002354...

Group # 2 average flux: 0.001151...

```
import java.util.Scanner;
class Problem4
{
  public static void main(String[] args)
  {
    try
    {
//************************************************************************
//                                                                       *
//       1D SN calculation of a slab:                                    *
//                                                                       *
//       |      |                  g                    |                *
//       |      |         -----------------             |                *
//       |      |         Grp 1       Grp 2             |                *
//       |      | Total    0.1        0.2               |                *
//       |      | G1->g    0.05       0.04              |                *
//       |S1 .4 | G2->g    0          0.1               |                *
//       |S2 .6 |                                       |                *
//       |      |                                       |                *
//       |      |                                       |                *
//       |      |                                       |                *
//       |      |                                       |                *
//       | 5 cm |              45 cm                    |                *
//       |<---->|<------------------------------------->|                *
//       |      |                                       |                *
//                                                                       *
//************************************************************************
//                                                                       *
//     Variable definitions:                                             *
//       totxs(ig)   Total cross section in group ig, 1/cm               *
//       scat(ig,jg) Scattering cross section from group ig to jg, 1/cm *
//       totscat(ig) Total scattering in group ig (i.e., sum of scat   *
//                      for all other groups jg                          *
//       sour(ig)    Source in group ig, #/cm3/sec                       *
//       bin(ib)     Bin values                                          *
//                   ib = 1 Left leakage for group 1                     *
//                      = 2 Left leakage for group 2                     *
//                      = 3 Right leakage for group 1                    *
//                      = 4 Right leakage for group 2                    *
//                      = 5 Flux for group 1                             *
//                      = 6 Flux for group 2                             *
//       ig          Current energy group of the particle               *
//       mu          Current direction cosine of the particle, (-1,1)  *
//       x           Current position of particle                       *
//       dd          Distance to next collision                         *
//       dx          x dimension distance to next collision = dd*mu    *
//       mfp         Mean free paths to next collision                  *
//                                                                       *
//************************************************************************
//                                                                       *
//       Set the source and cross sections                               *
```

```
//                                                                      *
//**********************************************************************
      double[] totxs={0.1,0.2};
      double[][] scat=new double[2][2];
      scat[0][0]=.05;
      scat[0][1]=0.04;
      scat[1][0]=0.;
      scat[1][1]=0.1;
      double[] sour={.4,.6};
      sour[0]/=5.;
      sour[1]/=5.;
//**********************************************************************
//                                                                      *
//       Find total scattering cross section for each group             *
//                                                                      *
//**********************************************************************
      int ng=2;
      System.out.println("Setting 1 division per 5cm");
      //Scanner sc=new Scanner(System.in);
      int ns=1; //sc.nextInt();
      int nx=ns*10;
      double dx=50./nx;
      double[][] scalar=new double[nx][ng];
      for(int ix=0;ix<nx;ix++)
      {
        for(int ig=0;ig<ng;ig++)
        {
          scalar[ix][ig]=0.;
        }
      }
      double[] sext=new double[nx];
      double[] sourin=new double[nx];
//**********************************************************************
//                                                                      *
//    For each group find the group-to-group tranfer (with the argument*
//    being how far away the cell is from this one                      *
//                                                                      *
//**********************************************************************
      double[][] transfer=new double[ng][nx];
      for(int ig=0;ig<ng;ig++)
      {
        double dtau=totxs[ig]*dx;
//**********************************************************************
//                                                                      *
//      Transfer to the same cell                                       *
//                                                                      *
//**********************************************************************
        transfer[ig][0]=1 - (1. / 2. / dtau) * (1. - 2. * E(3, dtau));
        //System.out.println(" Transfer ig "+ig+" 0 = "+transfer[ig][0]);
//**********************************************************************
//                                                                      *
```

```
//      Transfer to other cells                                          *
//                                                                       *
//***********************************************************************
        for(int ix=1;ix<nx;ix++)
        {
          double tau=(ix-1)*dtau;
          transfer[ig][ix]=(1./(2.*dtau))*(E(3,tau)-2.*E(3,dtau+tau)
           +E(3,tau+2.*dtau));
          //System.out.println(" Transfer ig "+ig+" "+ix+" = "+transfer[ig]
           [ix]);
        }
      }
//***********************************************************************
//                                                                       *
// Outer iterations: Loop over each group                               *
//                                                                       *
//***********************************************************************
      for(int ig=0;ig<ng;ig++)
      {
//***********************************************************************
//                                                                       *
//     Find the external and scattering source from other groups         *
//                                                                       *
//***********************************************************************
        for(int ix=0;ix<nx;ix++)
        {
          sext[ix]=0.;
          if(ix < ns)sext[ix]+=sour[ig];
          for(int igp=0;igp<ng;igp++)
          {
            if(ig != igp)sext[ix]+=scalar[ix][igp]*scat[igp][ig];
          }
        }
//***********************************************************************
//                                                                       *
//  Inner iterations                                                     *
//                                                                       *
//***********************************************************************
        int inner=0;
        double eps=0.00001;
        double conv=10000.;
        double[] scalarOld=new double[nx];
        while(Math.abs(conv)>eps)
        {
          inner++;
//***********************************************************************
//                                                                       *
//    Add within-group scattering source                                 *
//                                                                       *
//***********************************************************************
          for(int ix=0;ix<nx;ix++)
```

```
          {
            sourin[ix]=sext[ix]+scalar[ix][ig]*scat[ig][ig];
            scalarOld[ix]=scalar[ix][ig];
            scalar[ix][ig]=0.;
          }
//**************************************************************************
//                                                                         *
//        Loop over source cells                                           *
//                                                                         *
//**************************************************************************
          for(int ix0=0;ix0<nx;ix0++)
          {
//**************************************************************************
//                                                                         *
//        Loop over destination cells                                      *
//                                                                         *
//**************************************************************************
            for(int ix=0;ix<nx;ix++)
            {
              int dif=Math.abs(ix0-ix);
              double dcoll=transfer[ig][dif]*dx*(sourin[ix0]);
              scalar[ix][ig]+=dcoll/dx/totxs[ig];
            }
          }
//**************************************************************************
//                                                                         *
//    Check inner convergence                                              *
//                                                                         *
//**************************************************************************
          conv=0.;
          for(int ix=0;ix<nx;ix++)
          {
            double etry=(scalar[ix][ig]-scalarOld[ix])/scalar[ix][ig];
            if(Math.abs(etry)>conv)conv=Math.abs(etry);
          }
        }
      }
//**************************************************************************
//                                                                         *
//    Print results                                                        *
//                                                                         *
//**************************************************************************
      if(true)
      {
        for(int ig=0;ig<ng;ig++)
        {
          System.out.println("FOR GROUP "+(ig+1));
          for(int ix=0;ix<nx;ix++)
          {
            //System.out.println("  Flux pt "+(ix+1)+" = "+scalar[ix][ig]);
          }
```

```
        System.out.println(" average of 45-50");
        double avephi=0;
        for(int ix=nx-ns;ix<nx;ix++)
        {
          avephi+=scalar[ix][ig]/ns;
        }
        System.out.println("  Ave "+avephi);
      }
    }
  }
  catch(Exception e)
  {
    e.printStackTrace(System.out);
  }
}

static double E(int order,double x)
{
  double ret=0.;
  if(x == 0. && order == 3)
  {
    ret=0.5;
  }
  else if(order == 1)
  {
    if(x<1)
    {
      double a0=-.57721566;
      double a1=.9999193;
      double a2=-.24991055;
      double a3=.05519968;
      double a4=-.00976004;
      double a5=.00107857;
      ret=-Math.log(x)+a0+a1*x+a2*x*x+a3*x*x*x+a4*x*x*x*x+a5*x*x*x*x*x;
    }
    else
    {
      double a1=2.334733;
      double a2=.250621;
      double b1=3.330657;
      double b2=1.681534;
      ret=(x*x+a1*x+a2)/(x*x+b1*x+b2)/x/Math.exp(x);
    }
  }
  else
  {
    ret=(Math.exp(-x)-x*E(order-1,x))/(order-1.);
  }
  //System.out.println(" E  "+order+","+x+" = "+ret);
  return ret;
}
```

```
}
```