New Utilities in LISE++ (v.7.8.87 beta)

* Twinsol (solenoid) utility
  ray trace and matrix solutions
* the Kinematics calculator: Mott scattering and (A*, A+gamma) mode
* Wedge-wedge optimization
* "MOTER" ray trace code for MS Windows
  (FORTRAN and C++ versions)
* ISOL catcher utility

Version 7.8.87 beta from 06/06/06
available through LISE sites

Produce secondary beams in the Cyclotron Institute TA&MU (College Station, TX)

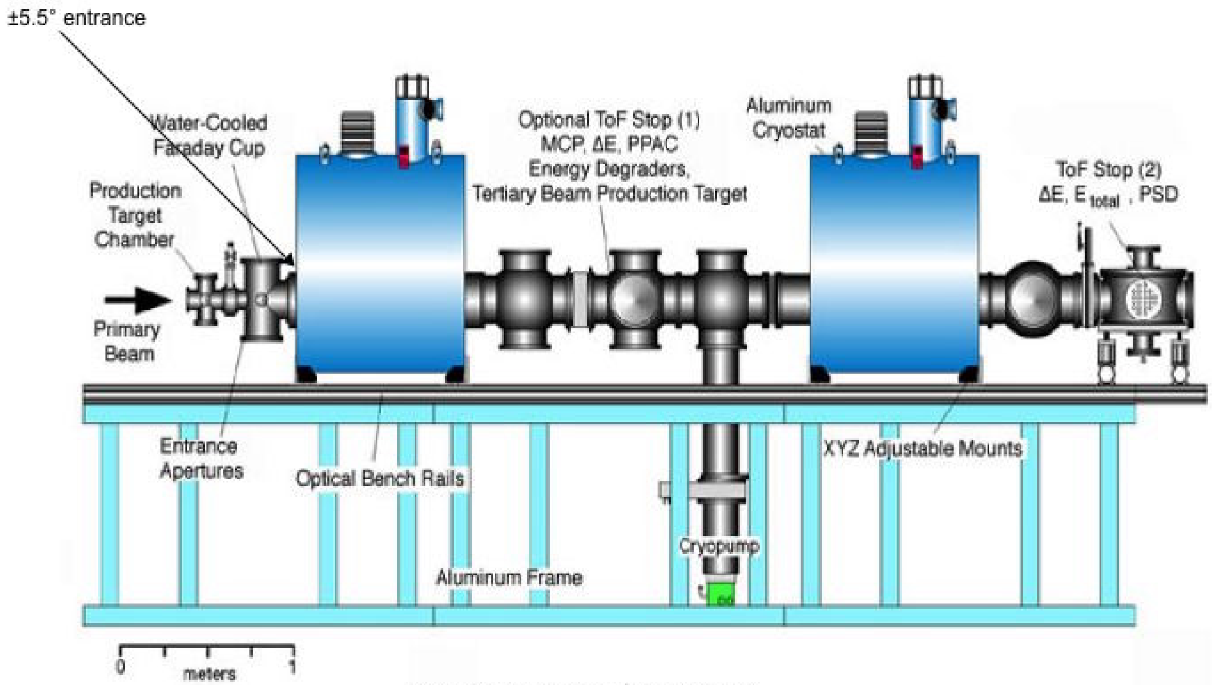
Find analytical solution to include the SOLENOID block in the LISE++ transmission calculations

I. TwinSol in LISE++

Thanks for the help (ideas, Fortran source, recommendations, etc) in developing the TwinSol utility in the LISE++ program:

Dr.G.Chubarian (TAMU)
Dr.Th.Materna (TAMU)
Prof. F.Becchetti (UMich)
Mr. H.Jiang (UMich)
Dr.V.Shchepunov (ORNL)

TwinSOL: RNB production facility



6 meter long= flight path

LISE++

Development stages

1. ***TwinSol utility dialog* (1st stage - DONE)**

a. RayTrace source transformation into C++
b. TwinSol-solenoid class library development
c. Relativistic solution for the RayTrace model
d. TwinSol matrix solution : main part (Transport)
e. TwinSol matrix solution : soft-edge corrections
f. TwinSol dialog construction
g. Solenoid dialog construction
h. Calculation results plot
i. Scratch file for multidisplay

2. ***TwinSol utility dialog* (2nd stage)**

a. Electrode dialog
b. Midplane absorber dialog
c. Gas filled solenoid

3. **TwinSol optimization utilities in the *TwinSol dialog* (3rd stage)**

4. **Research of TwinSol properties using the TwinSol utilities to develop a fragment transmission model through solenoids.**

5. **New blocks: round slits and beam stopper??**

6. **Solenoid block development to be used in LISE++ block frame**

a. Solenoid block optimization in LISE++
b. Transmission
c. Solenoid block setting for the fragment of interest
d. Optimization utility for Solenoid block
e. LISE transmission plots with Solenoid block

RayTrace source transformation into C++

```
class o_solenoid_electrode
{
..
};

//-----------------------------------------
class o_solenoid
{
….
o_solenoid_electrode electrode;
double CalculateOM(Cproj *beam, int Direct, bool
EdgeCorrection, OPTICAL_MATRIX *om, double z);
OPTICAL_MATRIX *OM;
};

//-----------------------------------------
class o_twinsol
{
o_solenoid s1, s2;
….
};

//-----------------------------------------
class G_twinsol : public o_twinsol
{

OPTICAL_MATRIX* CalculateOM(int k);
OPTICAL_MATRIX* CalculateGOM(double z);
…..
int RayTrace(distribution **d);

Cproj *Proj;
OPTICAL_MATRIX *GOM;
BEAM  *BeamSigma;
BEAM  *BeamRay;
};
```

Fortran Source:

```
 GX=V1*DT+.5*AX*DTS
 GY=V2*DT+.5*AY*DTS
 X=X+GX
 Y=Y+GY
 Z=DZ*FLOAT(STEP)
 T0 = T0 +  (SQRT(GX**2+GY**2+DZ**2))/(1.0E-09*V0)
 F7=AX*DT
 F8=AY*DT
 F9=AZ*DT
 HF7=.5*F7
 HF8=.5*F8
 HF9=.5*F9
 VX=VX+F7
 VY=VY+F8
 VZ=VZ+F9
```

C++ equivalent using the "Vector3" class

```
v3_shift = v3_VelocityWork * deltaT +
           v3_Acceleration * (0.5 * deltaT*deltaT);

v3_position += v3_shift;
v3_position.p3_z = this->Step * double(II);

Vector3 shift_step(   v3_shift.x(),
                      v3_shift.y(),
                      this->Step);

TOF +=  shift_step.r_xyz() /(1.0E-09*V0);

v3_Velocity += v3_Acceleration * deltaT;
```

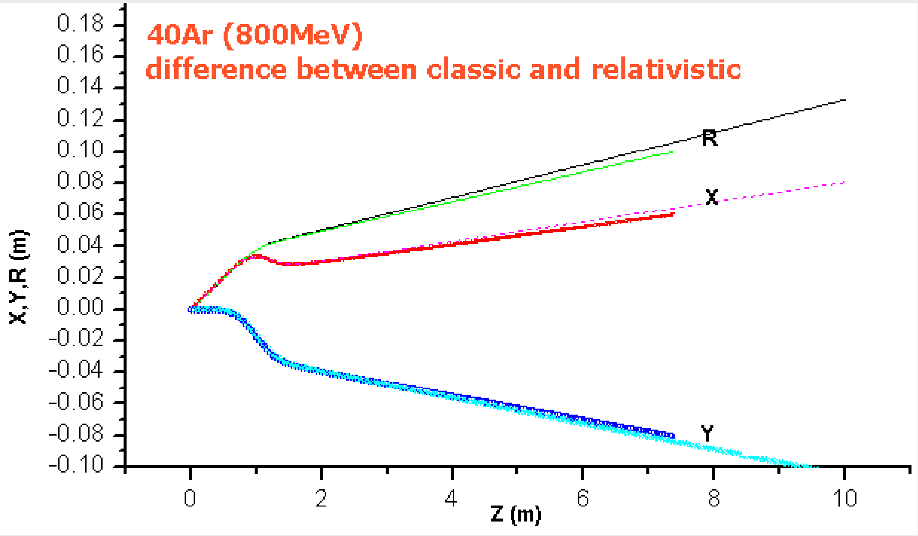
Relativistic solution for the RayTrace model.
Substitution by LISE++ library functions

Substitution expressions like that

V0=1.389E+07*SQRT(E/M)

by LISE++ relativistic functions

V0 = vc_m_s * *E_to_Beta* (energy);




```
C*****************************************************************
C calculate universal nuclear stopping power: SN in MeV/micron   *
C*****************************************************************
            ZABS=ABSORBER
       EPSIL=32.53*M2*E*1000/(N*ABSORBER*(M1+M2)*(N**.23+ZABS**.23))
            IF (EPSIL .GE. 30.) THEN
              SN = LOG(EPSIL)/(2*EPSIL)
            ELSE
              TEMP = (.01321*EPSIL**.21226) + (.19593*EPSIL**.5)
              SN = .5*LOG(1+1.1383*EPSIL)/(EPSIL+TEMP)
            ENDIF
          SN = SN*N*ABSORBER*M1*8.462/((M1+M2)*(N**.23+ZABS**.23))
          SN = SN * ATRHO * .001
       IF (ION .EQ. 1)CALL
PSTOP(ION,M1,ABSORBER,M2,VELSQR,PCOEF,SE)
       IF (ION .EQ. 2)CALL
HESTOP(ION,M1,ABSORBER,M2,VELSQR,PCOEF,SE)
       IF (ION .GT. 2)CALL
HISTOP(ION,M1,ABSORBER,M2,VELSQR,E,VFERMI,
   &          LFCTR,PCOEF,SE)
            SE = SE * ATRHO * .001
            E = E - SE - SN
            IF (E .LT. .000002) E = .000002
          END DO
          TEMP = E/EO(I)
          TEMP = SQRT(TEMP)
          .....
        ENDIF
        ENDIF
```

Using LISE++ library the substitution was done:

```
double  StoppingPower ( Celement *p,
                        Compound *cp,
                        double Energy,
                        int option);
```

TwinSol matrix solution : main part (Transport)

http://people.web.psi.ch/rohrer_u/trantext.htm#Solen

Urs C. Rohrer, PSI (SIN), CH-5232 Villigen-PSI, Switzerland

SOLENOID: Type code 19.0

Inside the solenoid, particles possessing a transverse velocity will describe an orbit which is helical in space. In order to study these movements, the beam centroid may be shifted and traced through the solenoid.
For B * L > Brho , the solenoid has to be divided into a sufficient amount of smaller elements in order to get an accurate image of the particle rays. But the R-matrix used in transport includes the fringe field effects at the entrance and exit of the solenoid. Therefore the slopes (x'/y') computed at the different segments are incorrect. Cubic spline interpolation used for the graphic display of the particle rays or the
envelopes need the correct slopes for a decent picture. Therefore provision has been made to do correct slope computation inside a solenoid by separating it into three regions: a) entrance face, b) homogenous region and c) exit face.

```
19.0 0. B. /ENTR/ ;
19.0 L1 B. /HOM1 / ;
 . ( L1  to  Ln  unequal 0. )
 .
19.0 Ln B. /HOMn/ ;
19.0 0. B. /EXIT/ ;
```

If the code encounters for the first time a 19. type code with zero length, it takes the R-matrix for the entrance fringe field. For all subsequent 19. type codes with Lx unequal 0 the R-matrix for the homogeneous longitudinal field is taken. If a 19. type code with zero length is encountered for a second time the R-matrix for the exit fringe field is selected. Other data entries between the sequence of 19. type codes are allowed (e. g. I/0 control or RMS addition to beam), but no checks are made if these entries make sense or not. In order to divide the solenoid into segments of equal length, the automatic segmenting feature of the space charge card may be used. In first order fitting, the lengths and the magnetic field strengths of the different segments may be adjusted through coupled vary codes.
Note: The original default is still available. If no type-code 19. card with zero length is present, then for all solenoid cards the matrix with entrance and exit fringe fields included will be taken.

First-order matrices for the solenoid:

1) Entrance face :

$$Ri = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & K & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ -K & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \qquad 2 * K = B / Brho$$

2) Exit face :

$$Ro = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -K & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ K & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

3) Homogeneous field:

$$Rh = \begin{pmatrix} 1 & S*C/K & 0 & S*S/K & 0 & 0 \\ 0 & 2*C*C-1 & 0 & 2*S*C & 0 & 0 \\ 0 & -S*S/K & 1 & S*C/K & 0 & 0 \\ 0 & -2*S*C & 0 & 2*C*C-1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

C = cos(K*L)
S = sin(K*L)

TwinSol matrix solution : soft-edge corrections

- Non-zero aperture - correction due to the finite length of the edge :

  - It decreases the solenoid total focusing – via the effective length of:

$$L = \frac{1}{B_0}\int_{-\infty}^{\infty} B_z(s)\, ds$$

  - It introduces axially symmetric edge focusing at each solenoid end:

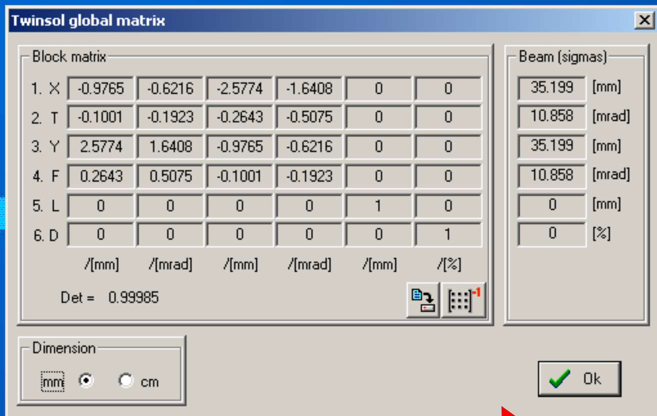
$$\Phi_{edge} = \frac{1}{2}\left(\int_{-\infty}^{\infty} B_z^2(s)\, ds - B_0^2\, L\right) = -\frac{k^2 a}{8} \qquad k = eB_0/pc$$

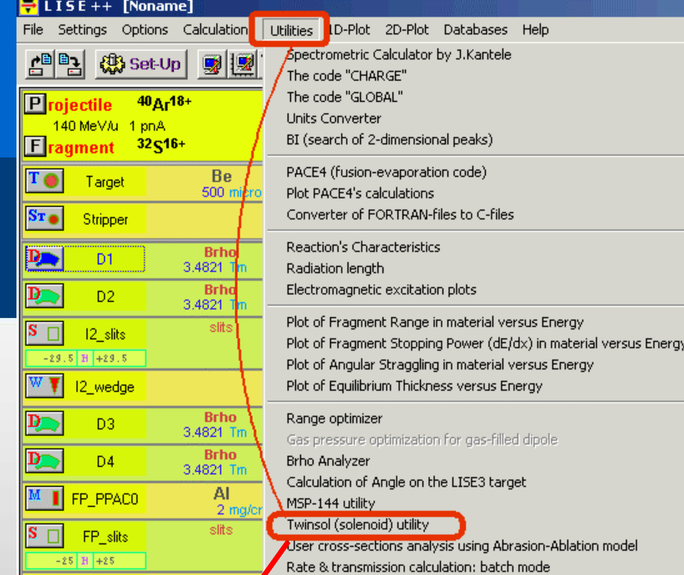
  - axially symmetric quadrupole

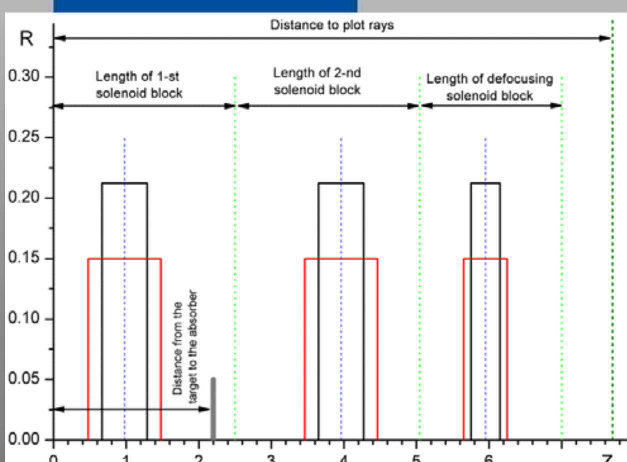
  - $\mathbf{M}_{soft\ sol} = \mathbf{M}_{edge}\,\mathbf{M}_{sol}\,\mathbf{M}_{edge}$

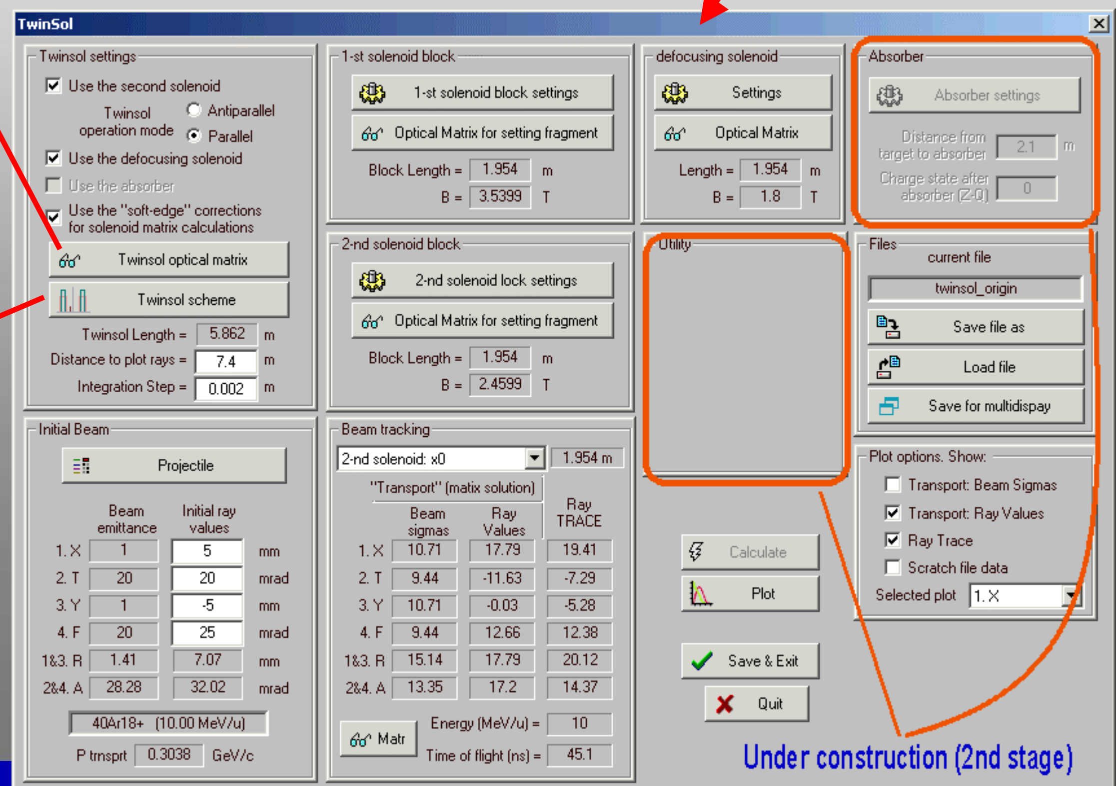
$$\mathbf{M}_{edge} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -\Phi_{edge} & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -\Phi_{edge} & 1 \end{bmatrix}$$

The TwinSol utility dialog in LISE++



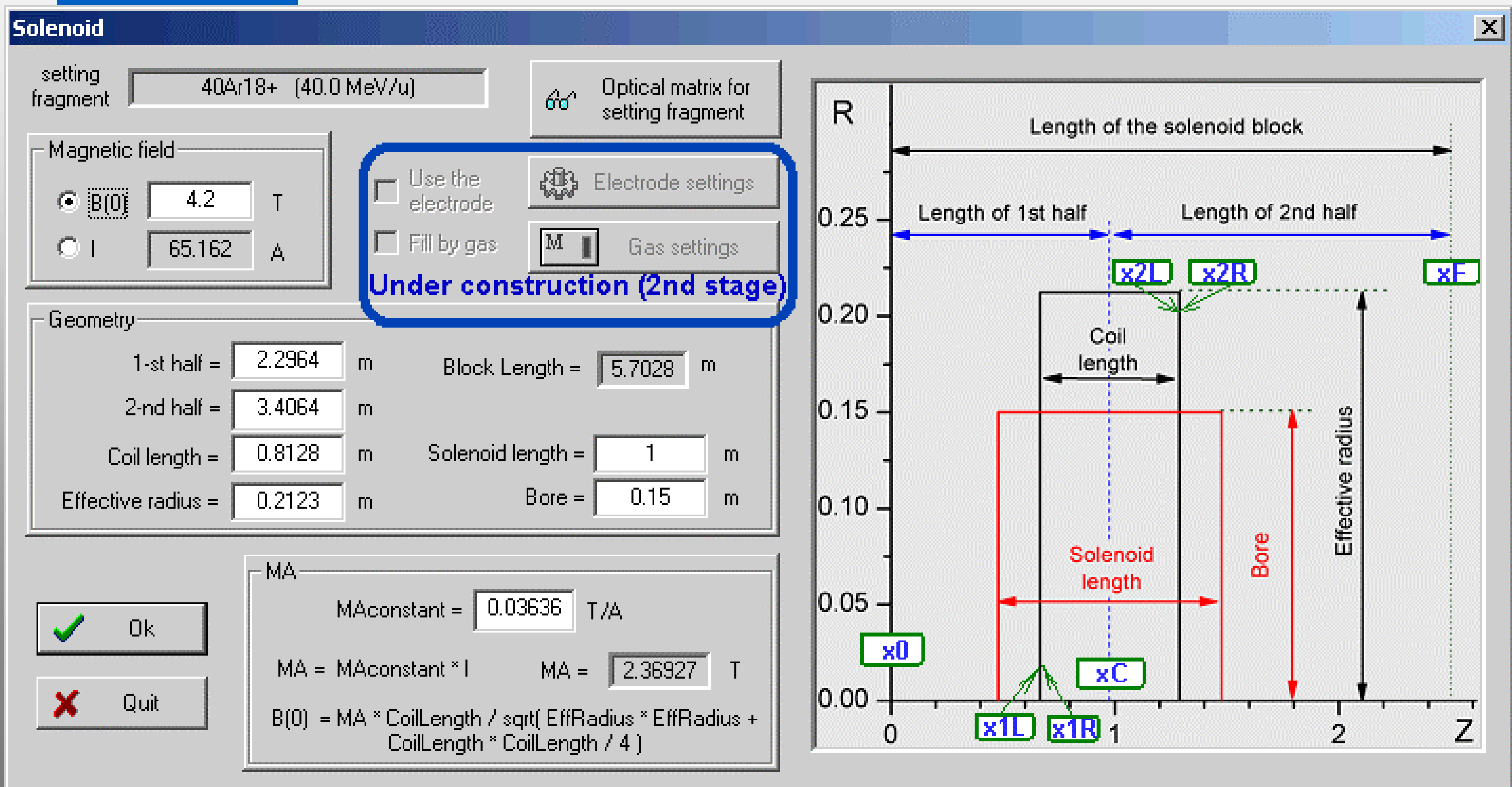




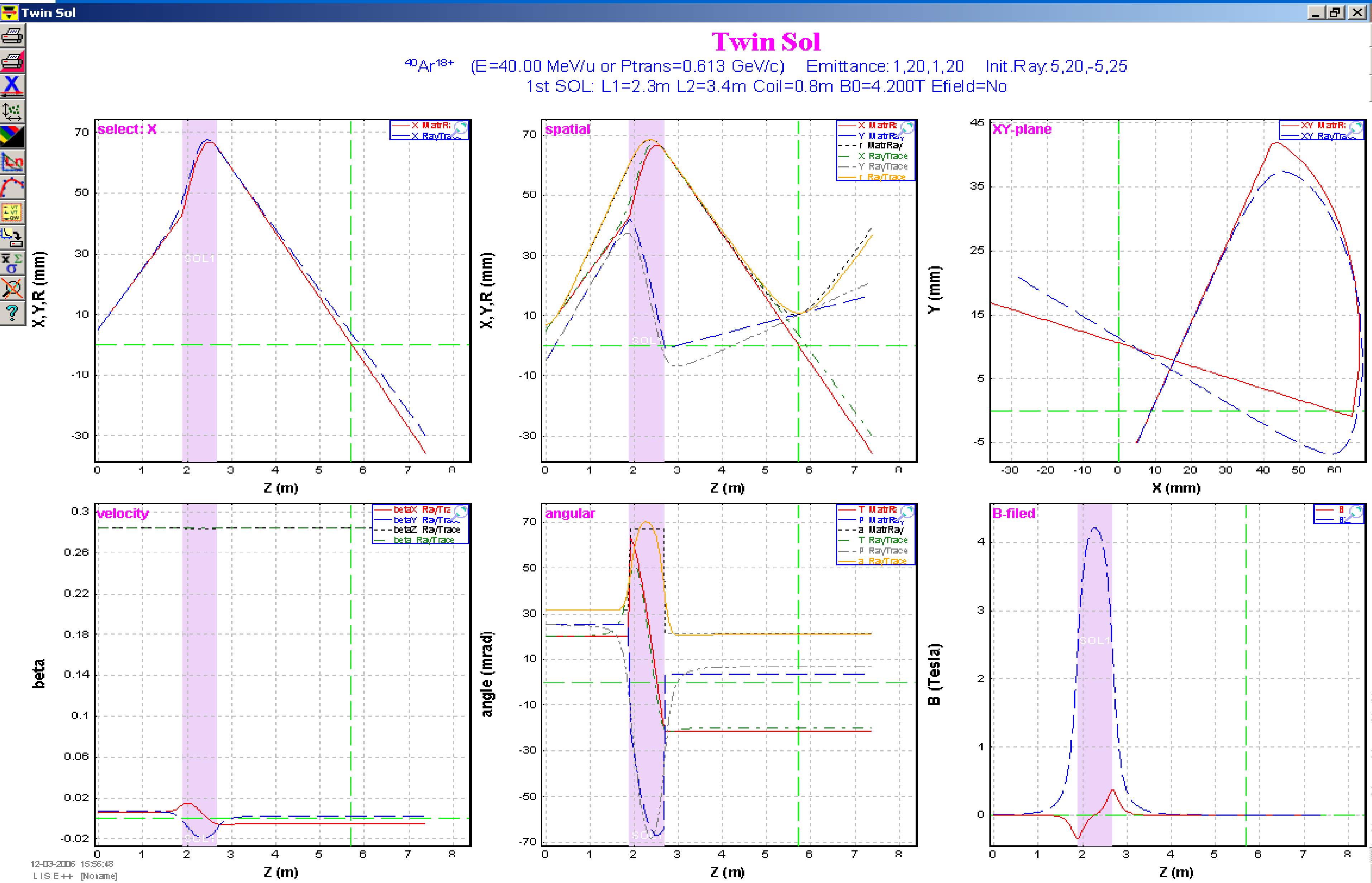


Under construction (2nd stage)

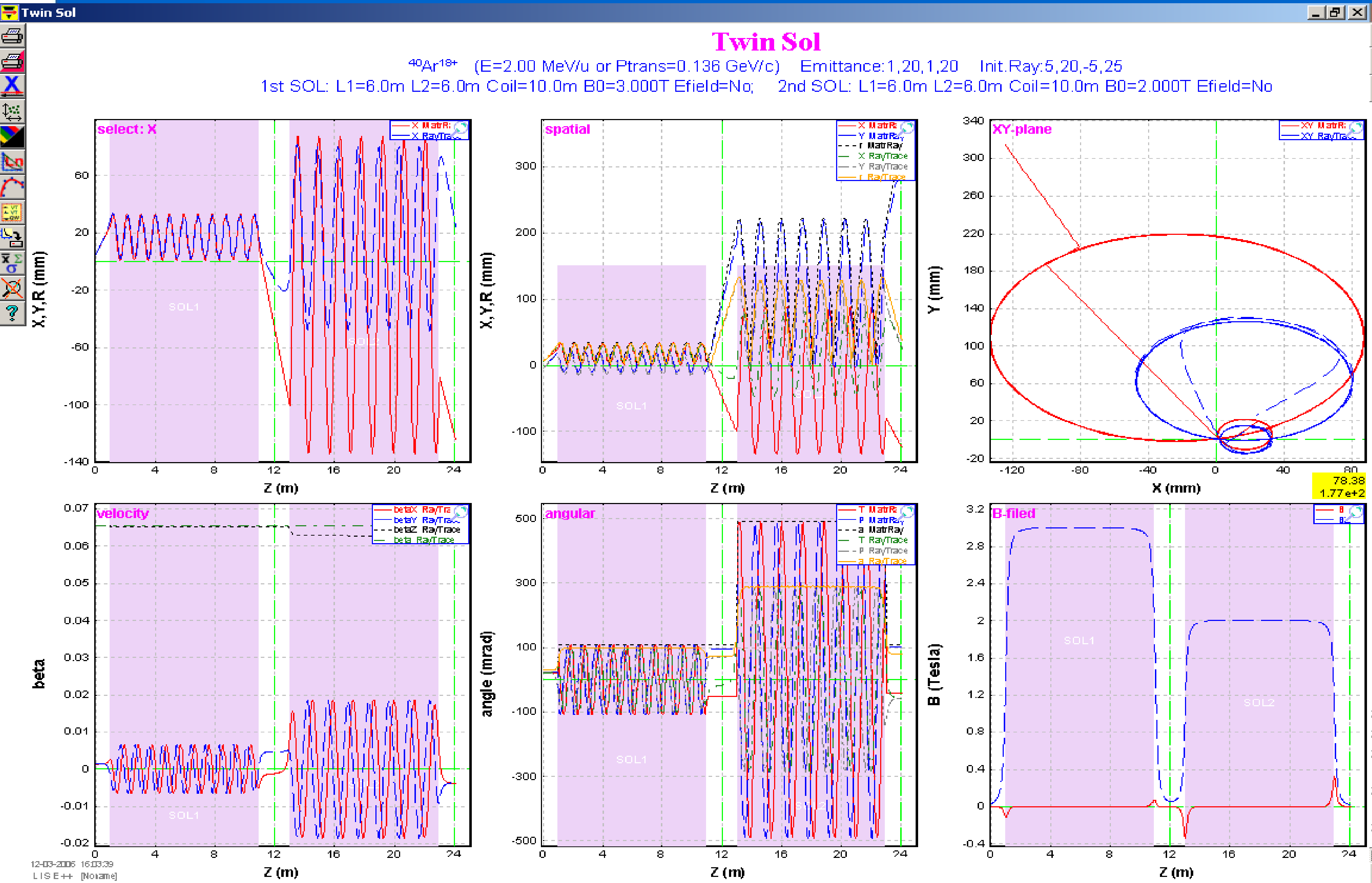
The Solenoid block dialog



Calculation results plot: 1 solenoid, L*B(0)/Br < 1



Calculation results plot: 2 solenoids, L*B(0)/Br >> 1



II. The Kinematics calculator: Mott scattering (1)

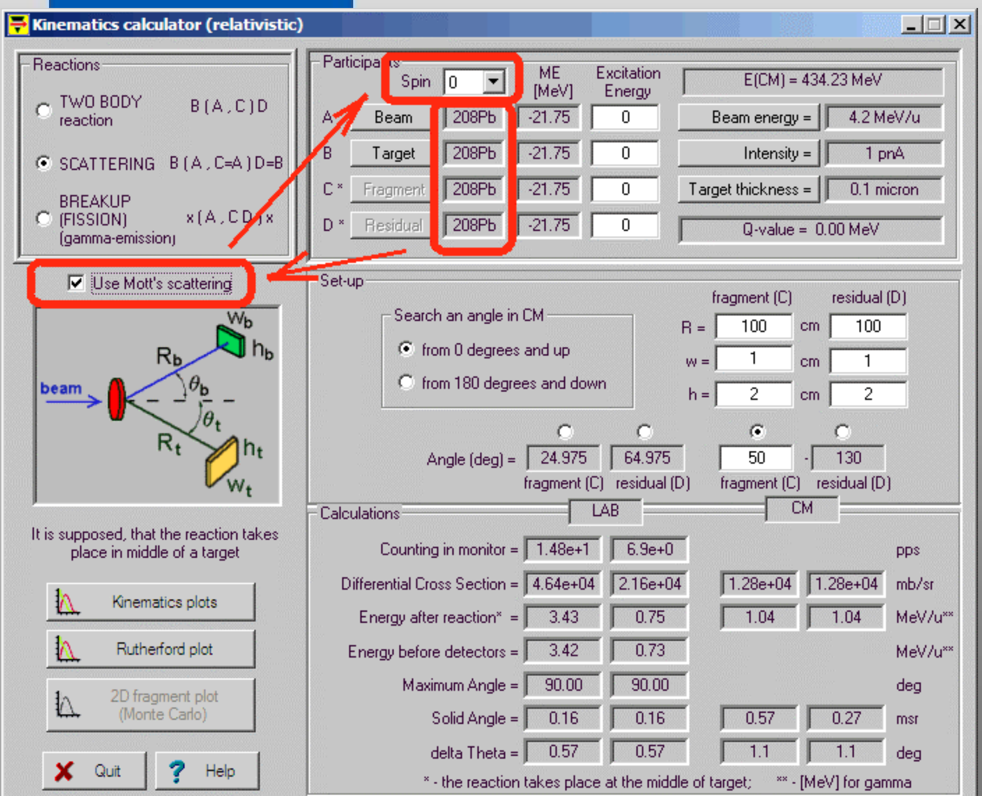
Thanks for the ideas to

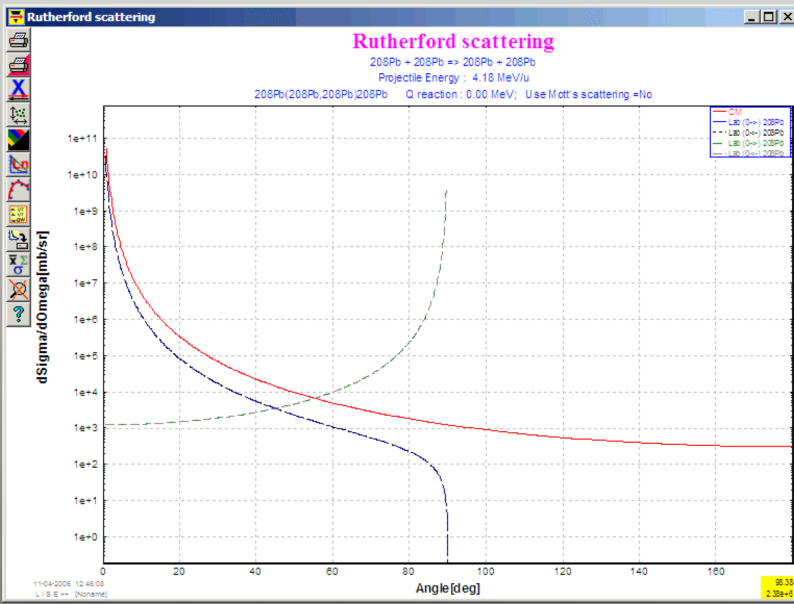
Dr. A. Villari (GANIL)

Quantum-mechanical scattering. Mott's approaches have been realized in LISE++ to:

a. **Relativistic case**: $\xi(\vartheta)d\omega = \xi_0(\vartheta)\left[1-\beta^2\cdot\sin^2(\vartheta/2)+....\right]d\omega$, where $\xi_0(\vartheta)$ is the classical differential cross section.

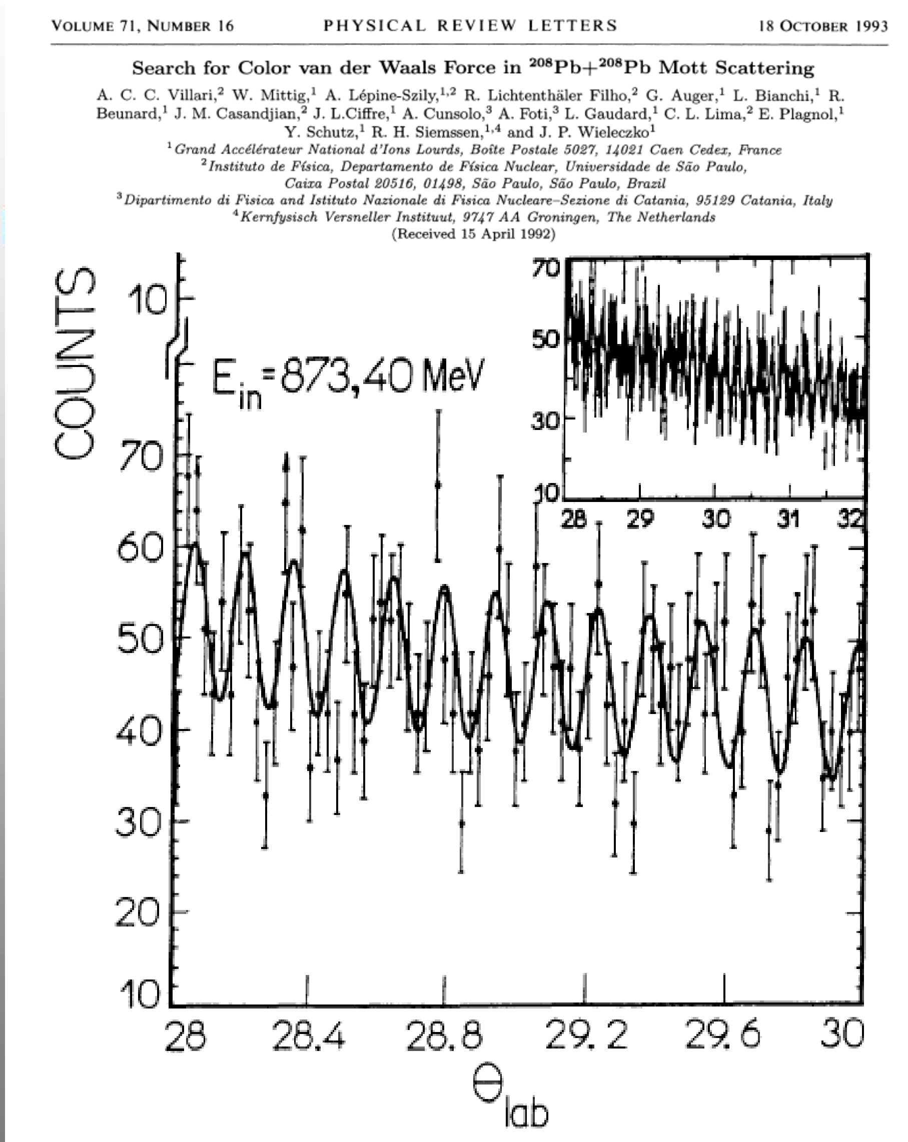
b. **Identical particles**: $d\sigma(\theta)=\left(\frac{q_i q_j}{4\cdot E}\right)^2\left[\frac{1}{\sin^4(\theta/2)}+\frac{\delta_{ij}}{\cos^4(\theta/2)}+\delta_{ij}\frac{(-1)^{2S}}{(2S+1)}\frac{2}{\cos^2(\theta/2)\,\sin^2(\theta/2)}\cos\left[\eta\log\tan^2(\theta/2)\right]\right]d\omega$





Coulomb scattering ^{208}Pb (873MeV)+^{208}Pb (classical case)

The Kinematics calculator: Mott scattering (2)



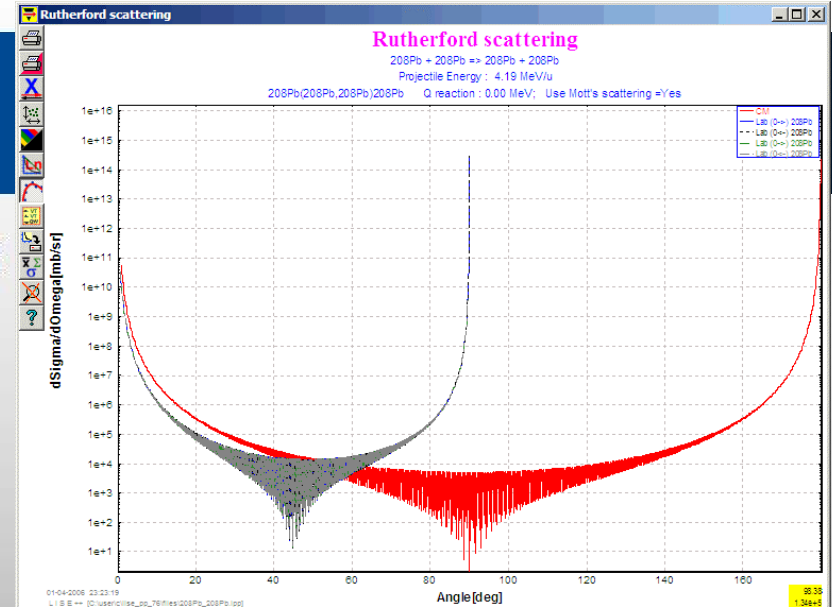
VOLUME 71, NUMBER 16 PHYSICAL REVIEW LETTERS 18 OCTOBER 1993

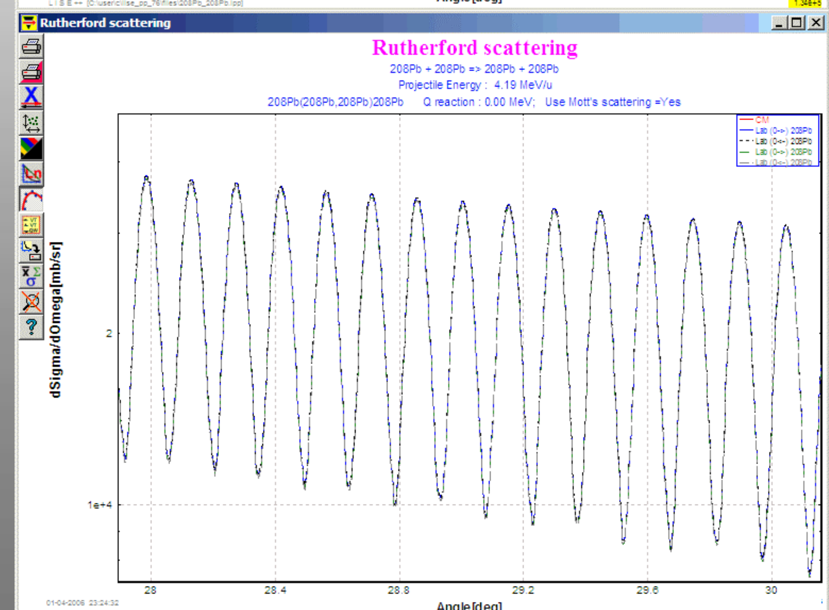
Search for Color van der Waals Force in ^{208}Pb+^{208}Pb Mott Scattering

A. C. C. Villari,[2] W. Mittig,[1] A. Lépine-Szily,[1,2] R. Lichtenthäler Filho,[2] G. Auger,[1] L. Bianchi,[1] R. Beunard,[1] J. M. Casandjian,[2] J. L.Ciffre,[1] A. Cunsolo,[3] A. Foti,[3] L. Gaudard,[1] C. L. Lima,[2] E. Plagnol,[1] Y. Schutz,[1] R. H. Siemssen,[1,4] and J. P. Wieleczko[1]
[1]*Grand Accélérateur National d'Ions Lourds, Boîte Postale 5027, 14021 Caen Cedex, France*
[2]*Instituto de Física, Departamento de Física Nuclear, Universidade de São Paulo, Caixa Postal 20516, 01498, São Paulo, São Paulo, Brazil*
[3]*Dipartimento di Fisica and Istituto Nazionale di Fisica Nucleare-Sezione di Catania, 95129 Catania, Italy*
[4]*Kernfysisch Versneller Instituut, 9747 AA Groningen, The Netherlands*

(Received 15 April 1992)

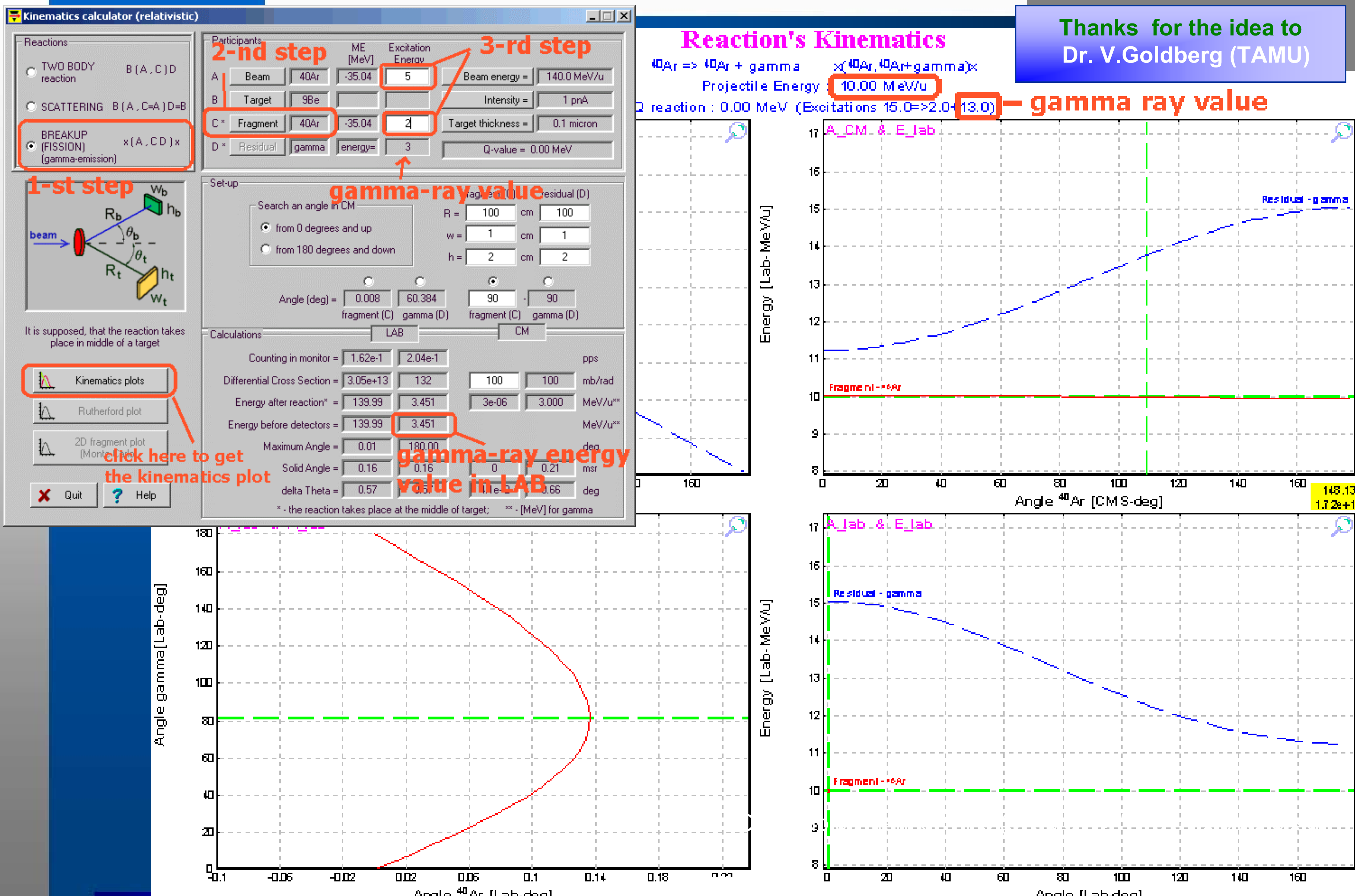
Experiment: Mott scattering of identical particles ^{208}Pb (873MeV)+^{208}Pb.





LISE++ calculation: Mott scattering of identical particles ^{208}Pb (873MeV)+^{208}Pb.

The Kinematics calculator: (A*, A+gamma)



III. Wedge-Wedge optimization

Wedge shape degrader in the Target-Wedge optimization utility

Before it was assumed that only curved profile degraders can be used for optimization. Now you can assign the wedge operation mode (Fig.1) in order to use in the Target-Wedge optimization dialog (Fig.2). Wedge angle of a wedge shape degrader will be recalculated according to chosen mode during optimization calculation.

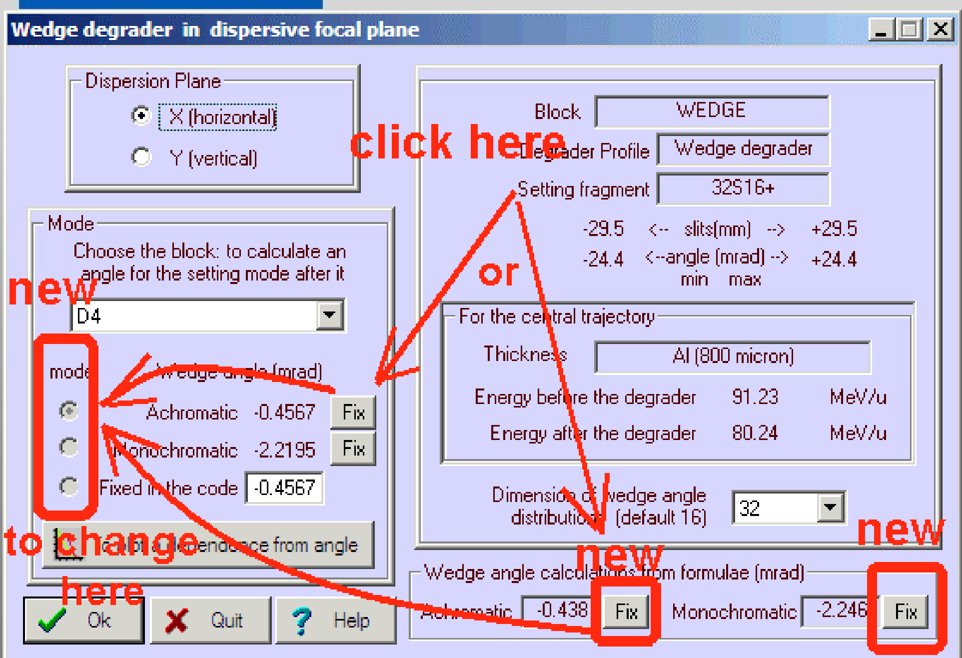


Fig.1

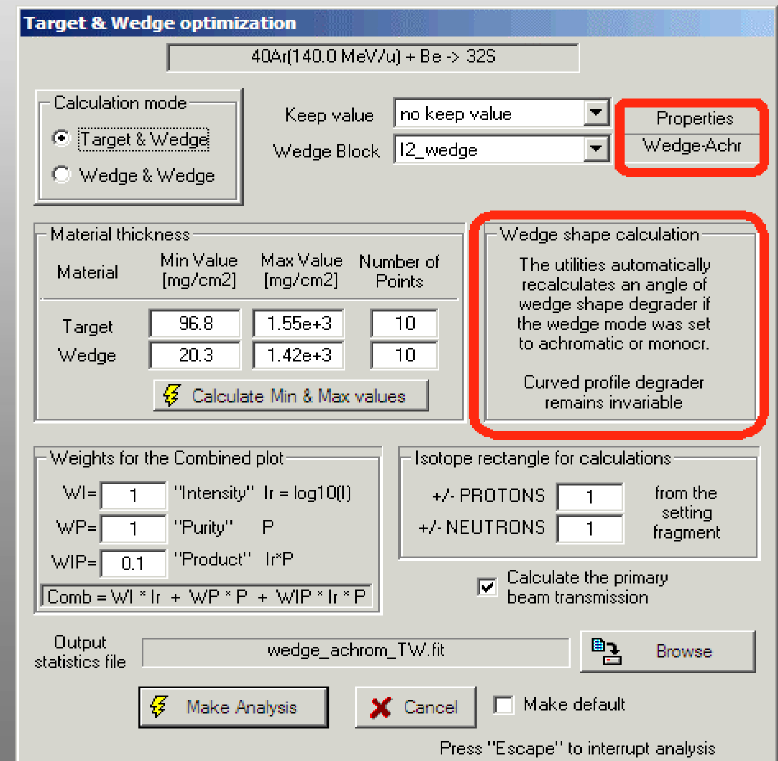
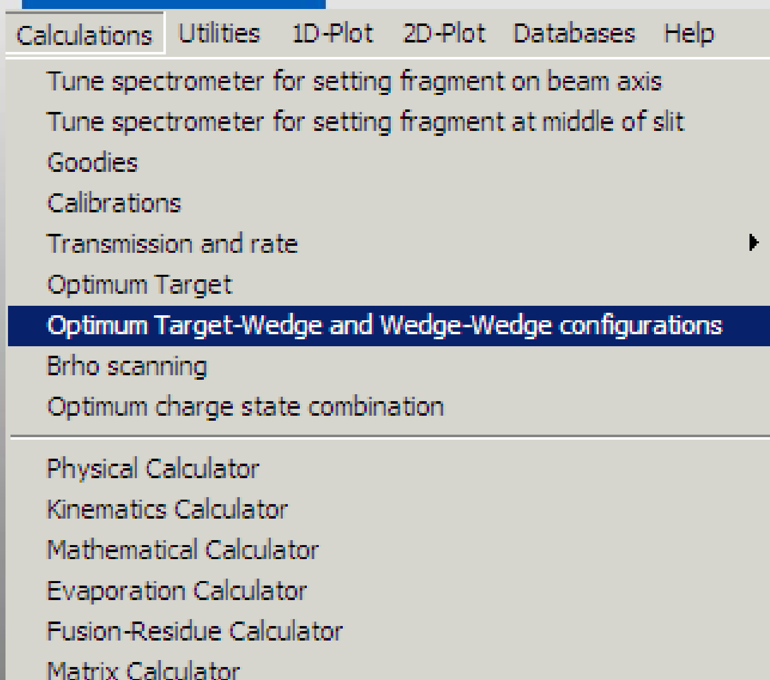
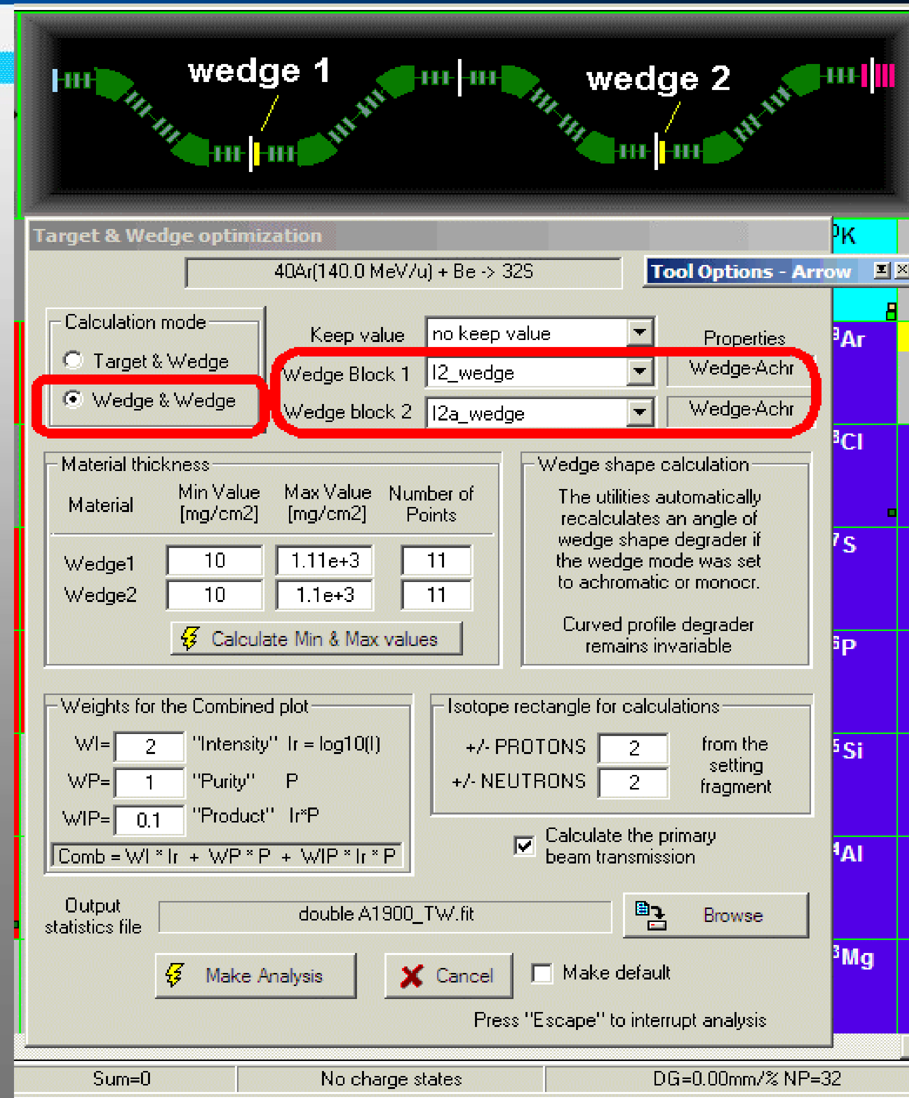


Fig.2

III. Wedge-Wedge optimization (2)

Wedge-Wedge optimization mode in the Target-Wedge optimization utility

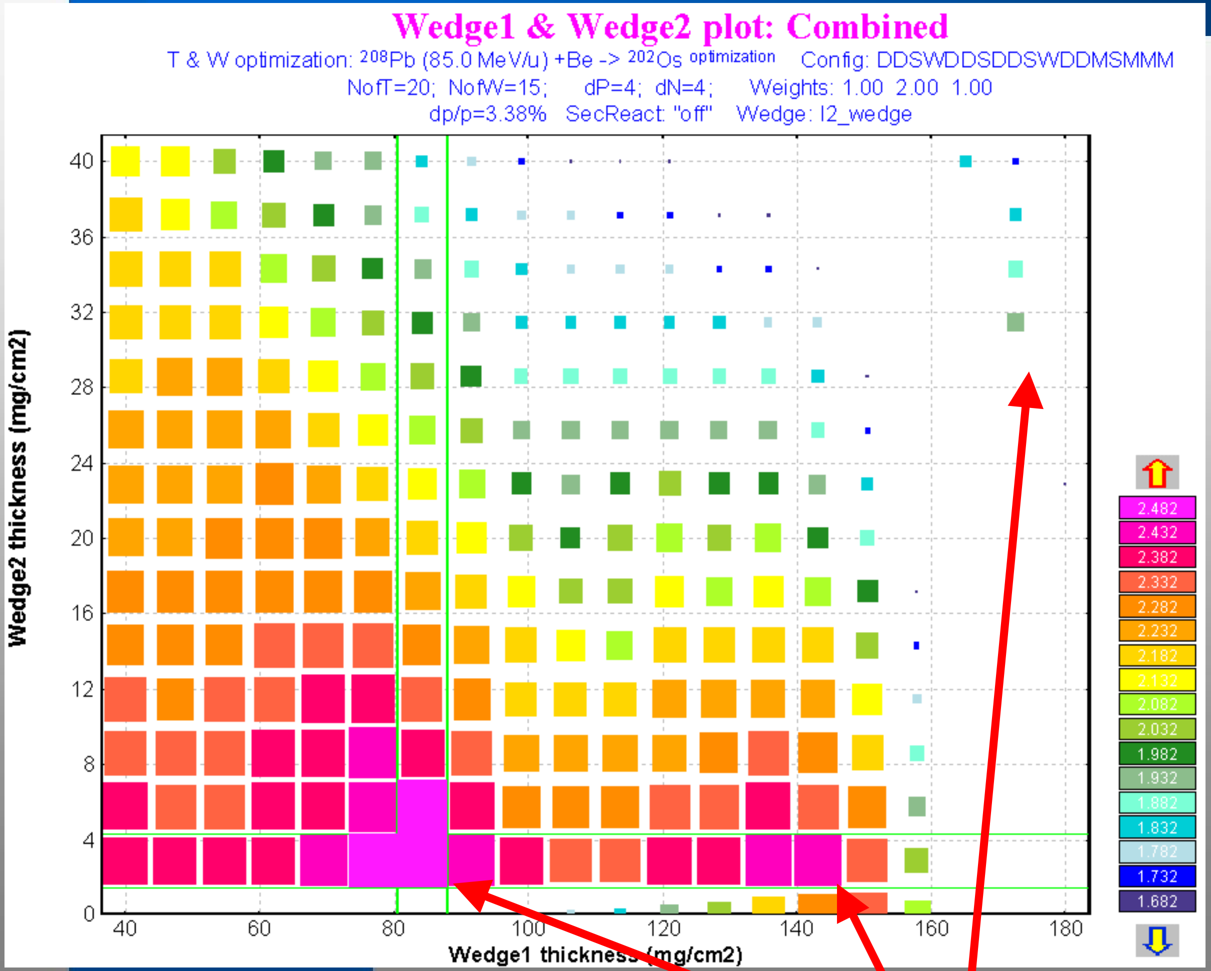




The wedge angle of wedge-shape degrader is taken from the formula (not from scan)

III. Wedge-Wedge optimization (3)

Wedge-Wedge optimization + optimum charge state combination



But this optimization now can be used with two wedges only for charge states analysis, and it is not so effective without secondary reactions in wedge.

The next important step in LISE development: secondary reactions in wedge!!!

Different charge state combinations

IV. "MOTER" ray trace code for MS Windows

(FORTRAN and C++ versions)

Prof.B.Sherrill
Prof.D.Morrissey

Transformation to C++
S.Lobastov (Dubna)

Using MOTER To Design PILAC*

H. S. Butler, Z. Li† and H. A. Thiessen
Los Alamos National Laboratory
P. O. Box 1663, Mail Stop H847
Los Alamos, New Mexico 87545

PHYSICAL REVIEW C VOLUME 47, NUMBER 2 FEBRUARY 1993

Reconstructive correction of aberrations in nuclear particle spectrographs

M. Berz, K. Joh,* J. A. Nolen,* B. M. Sherrill, and A. F. Zeller
Department of Physics and Astronomy and National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, Michigan 48824
(Received 24 August 1992)

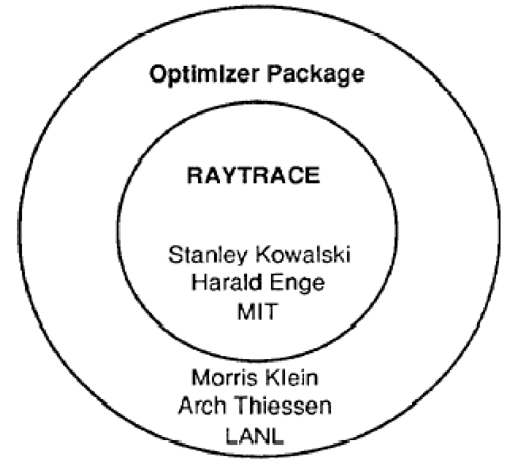


Figure 1. Structure of MOTER -- an optimizer package built around the RAYTRACE kernel.

	FORTRAN	C++
data file assignment is the next:	fort.12	K8ts.mag
	fort.14	K8ts.dem
	fort.19	For019.dat
	fort.6	K8ts.out
	fort.8	K8ts.opt

Next Steps

1. C++ classes and source optimization, search for Bugs
2. Substitution by functions from LISE++ library (for example energy loss, straggling)
3. Documentation, Manual
4. Shell construction
5. Graphical output of calculation results

Data and executable files are in the directory
moter_root = \\ projects \ proj4 \ temp \ Tarasov \ Moter

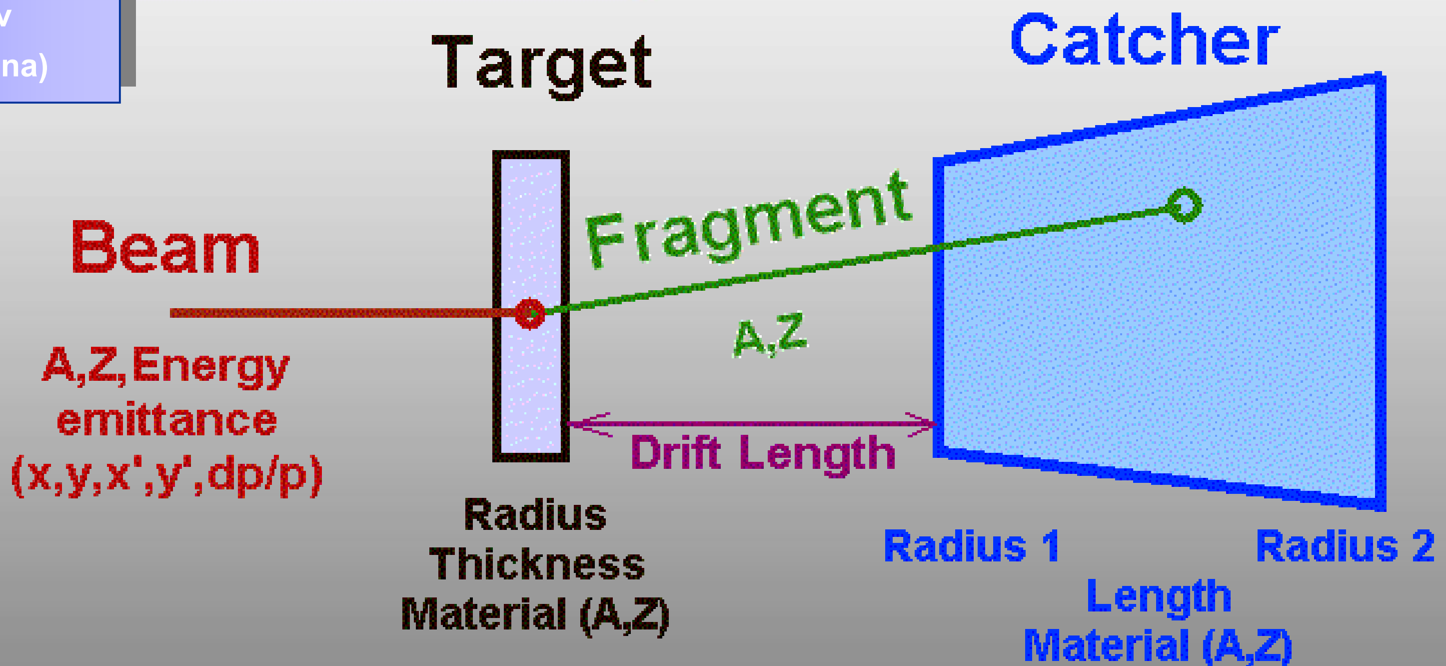
C++ version	moter_root \ CCMoter
executable file	moter_root \ CCMoter\Moter.exe
data files in the directory	moter_root \ CCMoter\data

FORTRAN version	moter_root \ Fmoter
executable file	moter_root \ Fmoter \ Debug \ FMoter.exe
data files in the directory	moter_root \ Fmoter \ debug

Thanks for the collaboration :

Prof.Yu.Peninozhkevich
Dr.G.Gulbekian
Mr.S.Mitrofanov
(FLNR, JINR, Dubna)

V. ISOL catcher in LISE++

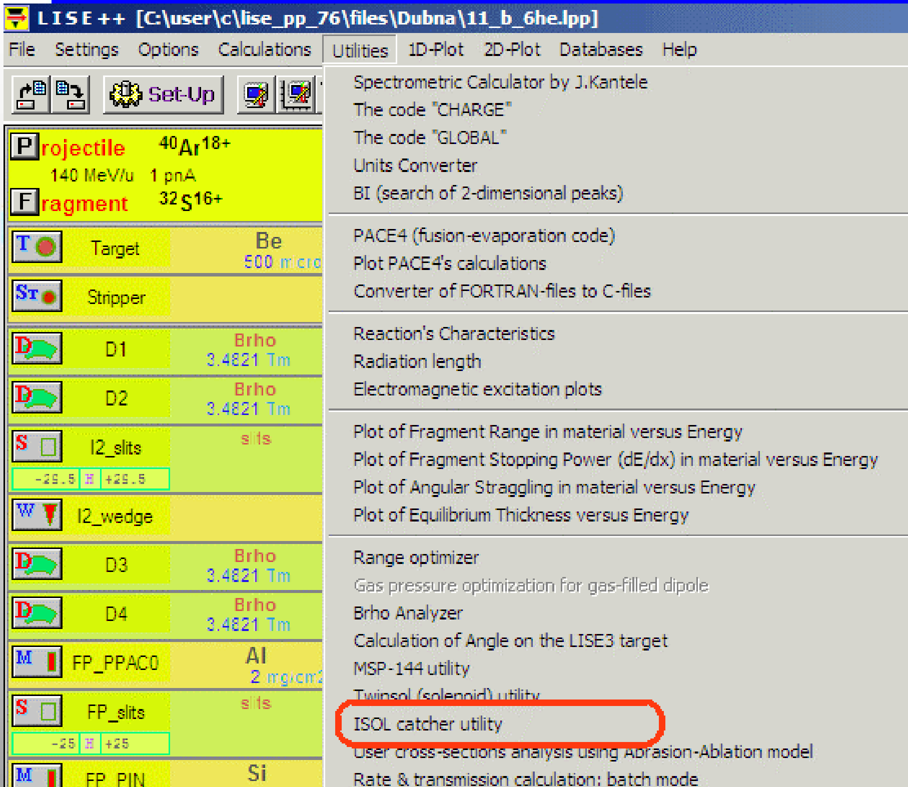


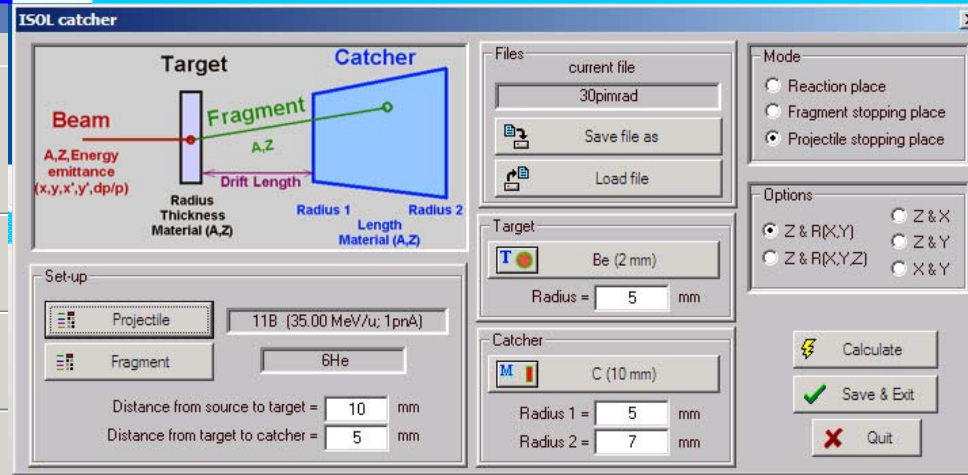
Aim: define catch efficiency of projectile fragmentation products for different geometrical and material configurations.

Monte Carlo solution

Access to the ISOL catcher utility

ISOL catcher dialog



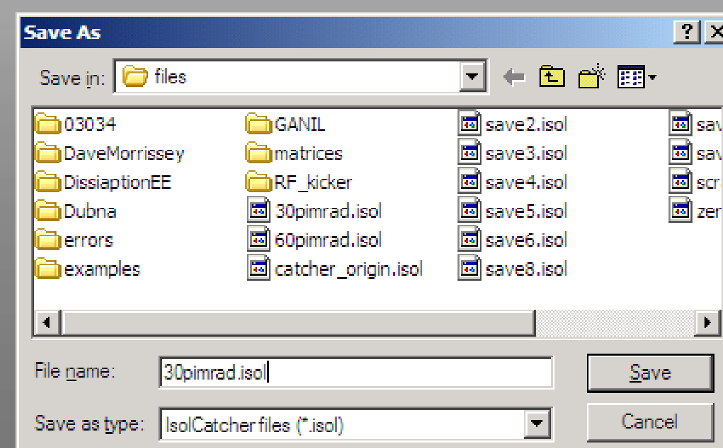


ISOL files

Last saved file is loaded at the beginning

Default directory "LISE/files"

Extension "*.isol"



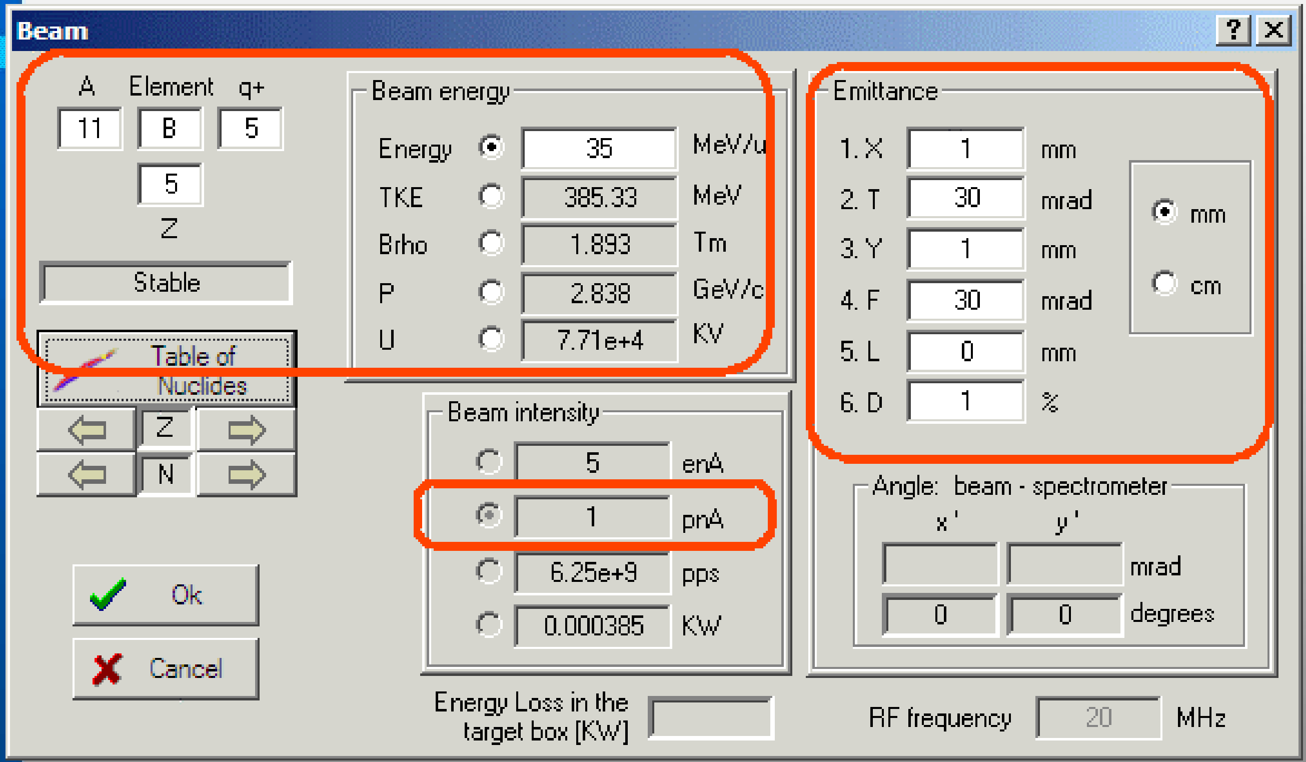
ISOL catcher calculations are performed for the "Projectile fragmentation" reaction mechanism set to the EPAX2 cross section mode.

Other options (Energy Loss mode, Velocity and momentum distribution width) are taken from LISE current settings.

It is recommended to load a LISE file with your settings before to use LISE ISOL catcher utility.

Do not use the Convolution model: it takes a lot of time for Monte Carlo calculations.

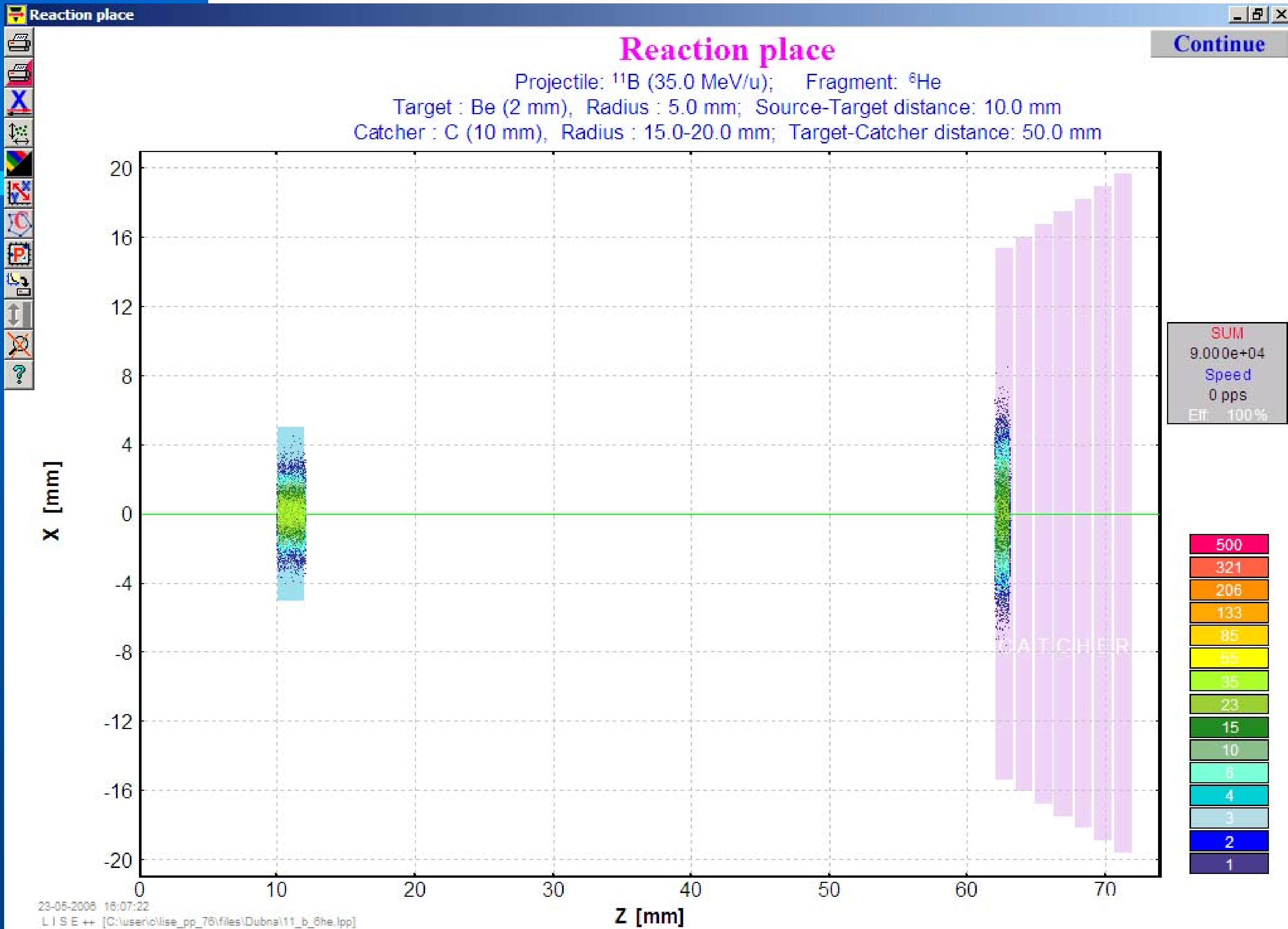
The primary beam dialog



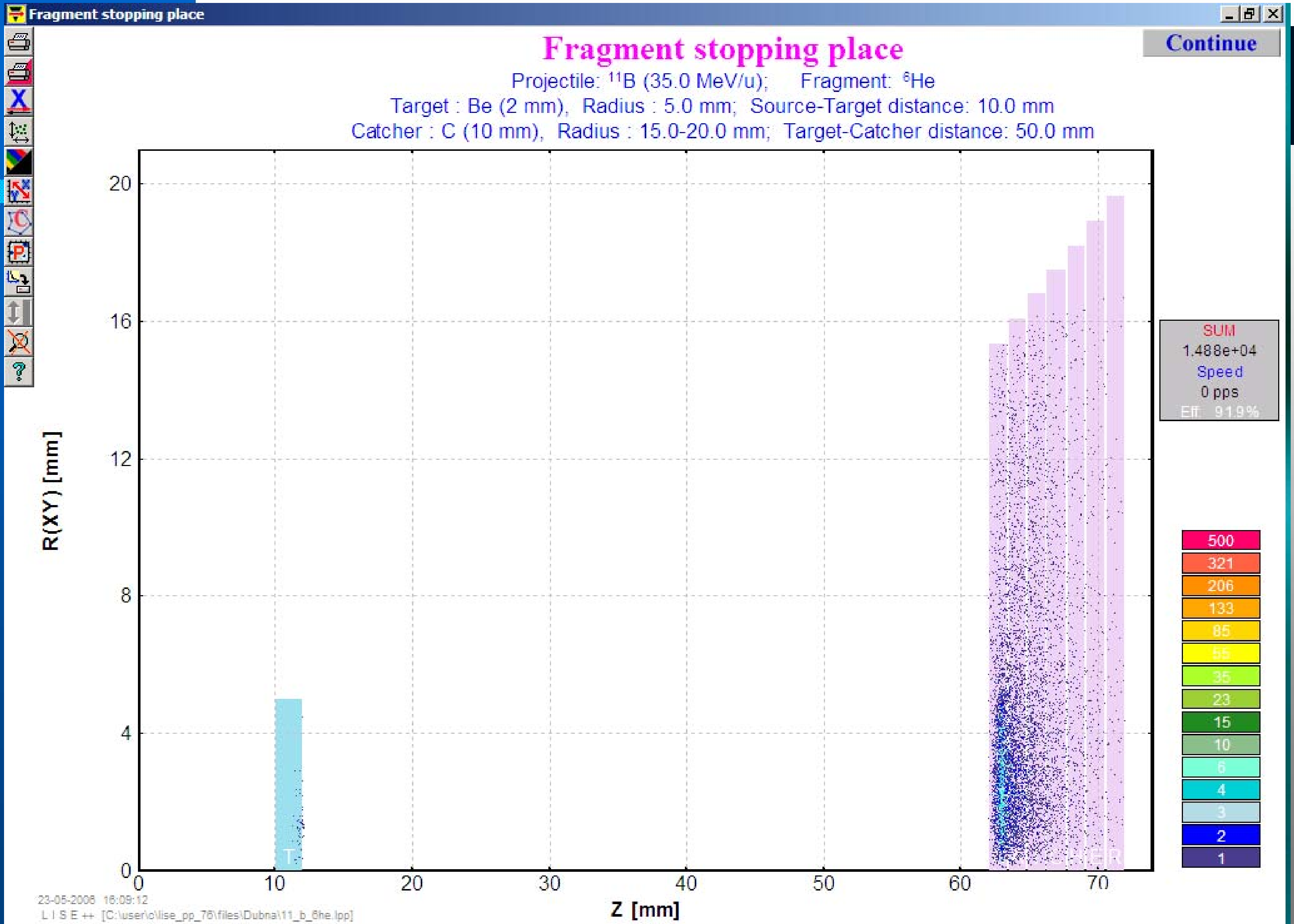
First simulation step is determination of initial coordinates (x, x', y, y', E) using the beam emittance

The primary beam intensity is always equal to 1 pnA

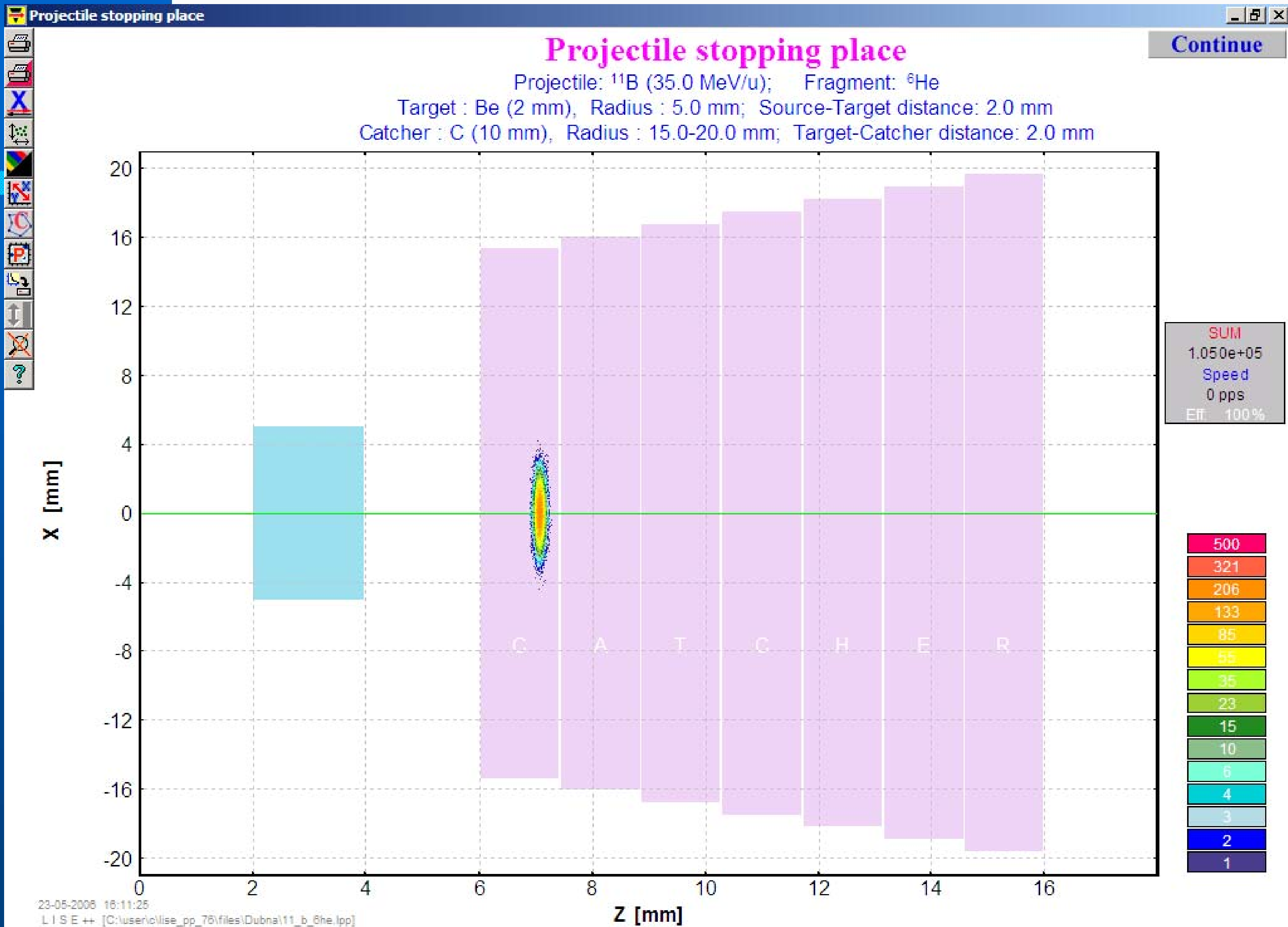
Reaction place



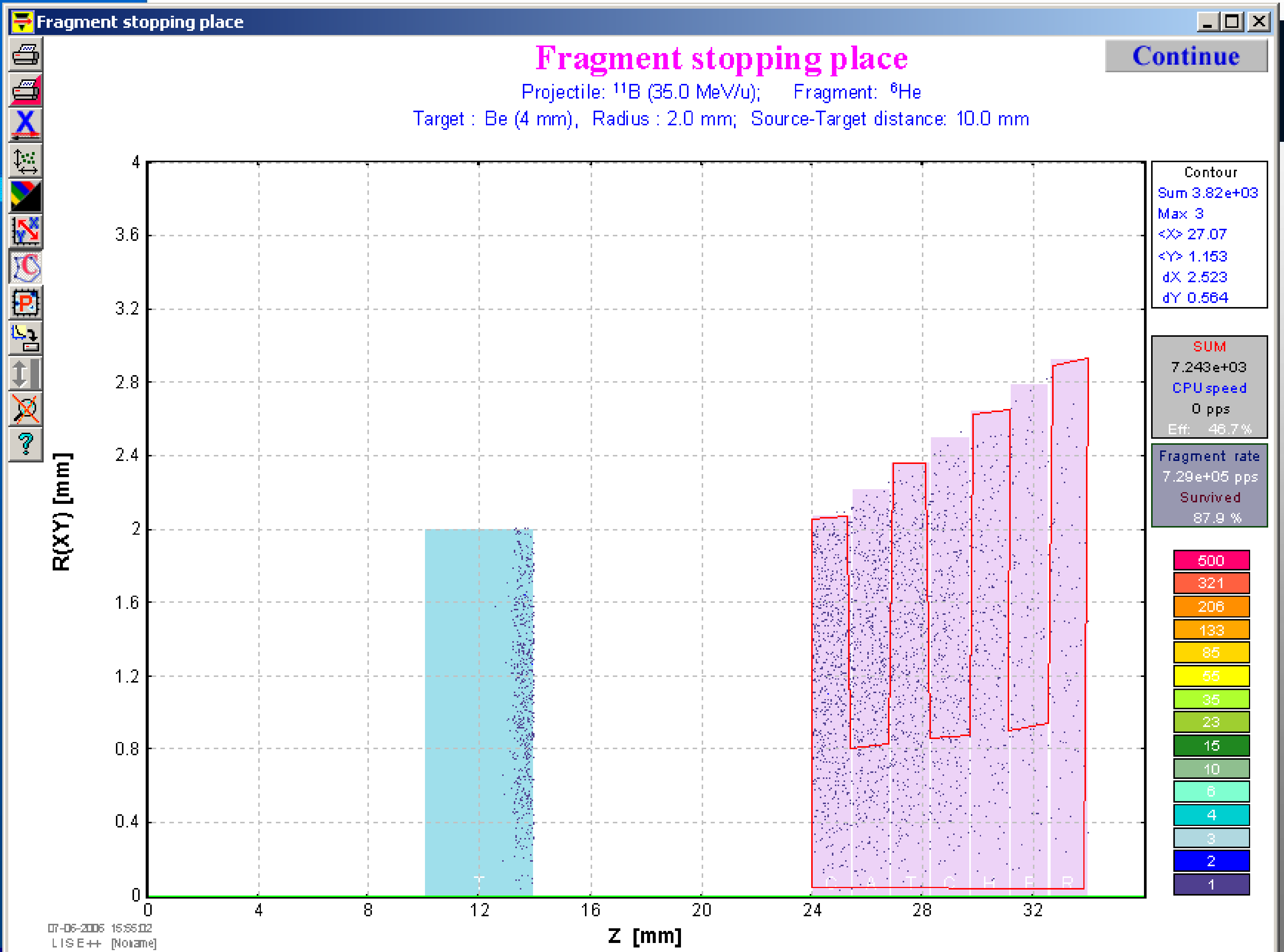
Fragment stopping place



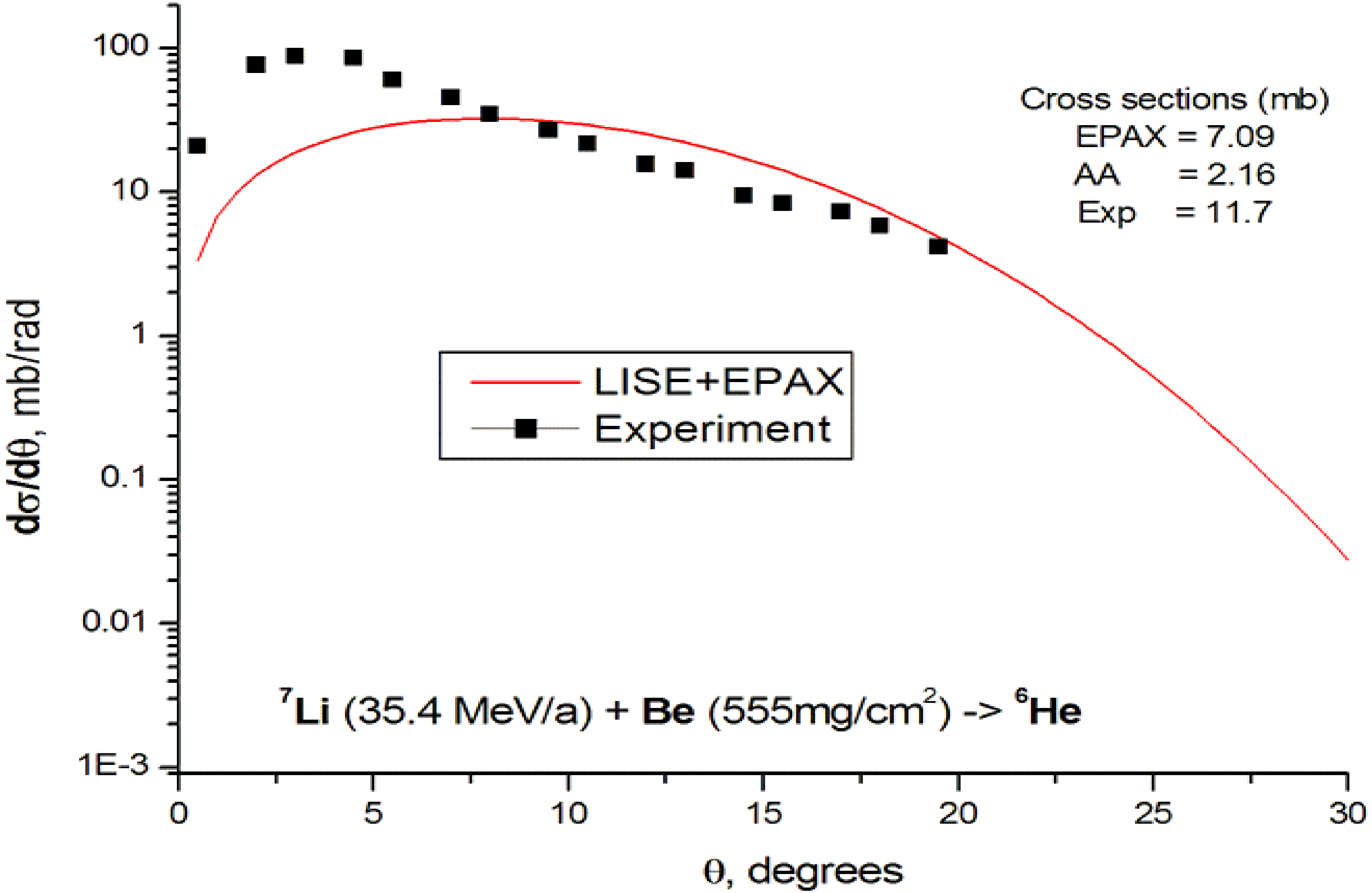
Projectile stopping place



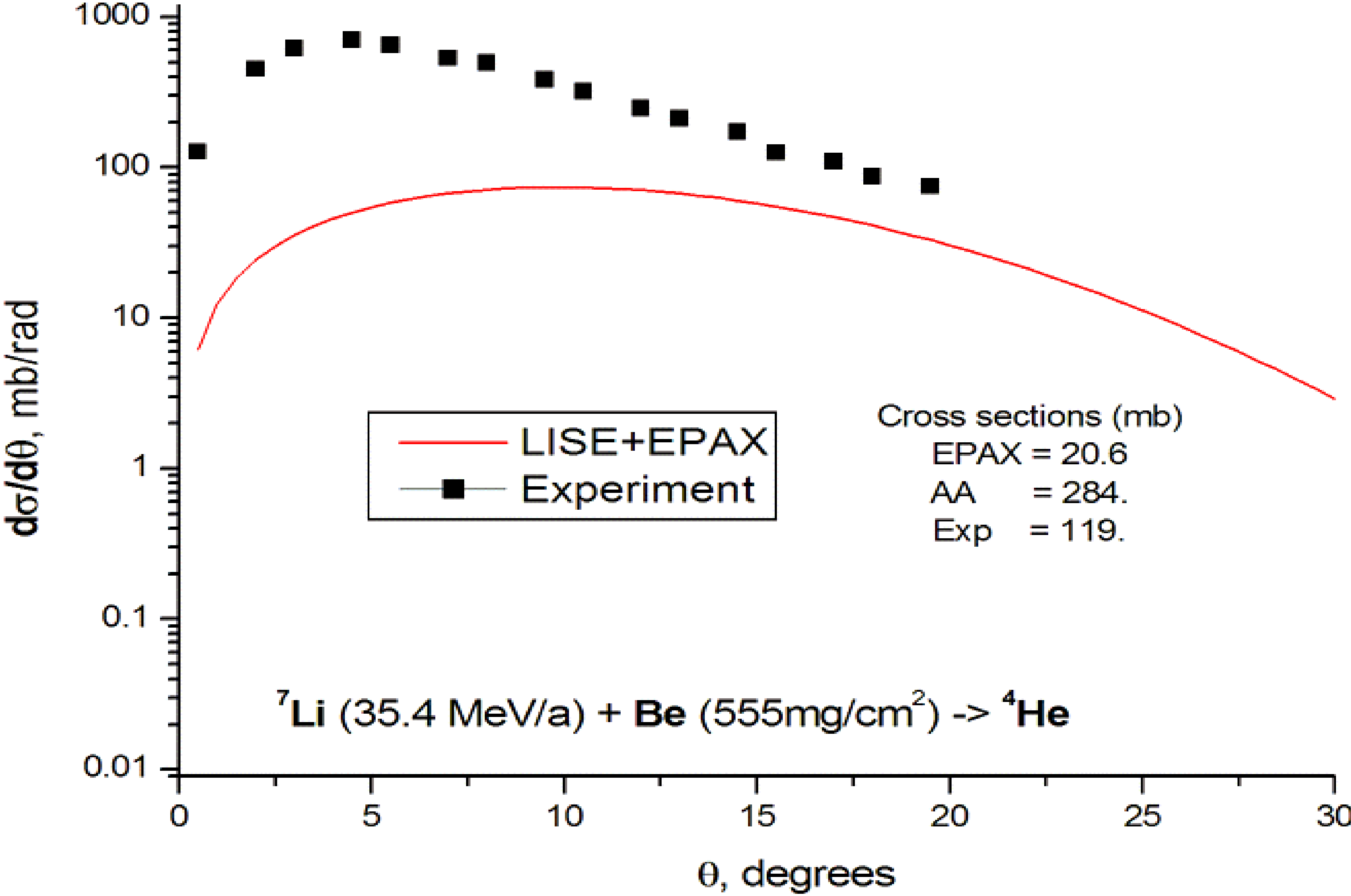
Fragment stopping place + contours



^{7}Li (35MeV/u) + Be => ^{6}He (A.Rodin et al. Acculina, Dubna)



^{7}Li (35MeV/u) + Be => ^{4}He (A.Rodin et al. Acculina, Dubna)



LISE++

NSCL

ISOL Catcher development

Next steps

- *Rate analysis*
- *Intensity loss due to reactions*
- Cross sections: Emin=Vc
- Cross section: user file
- Re-direction: modification
- Angular straggling
- Angle: beam-target
- Intensity variation
- Calculation speed optimization
- Secondary reactions
- Other reactions?

(for example: fission => SPIRAL2)

New modes

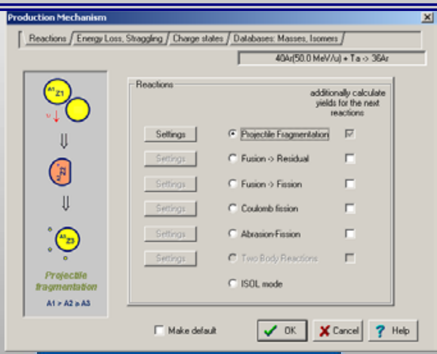
- Energy loss plot

After catcher

- Angular distribution plot
- Momentum, velocity and energy distributions
- Simulate dipole cuttings (angular and momentum acceptances)

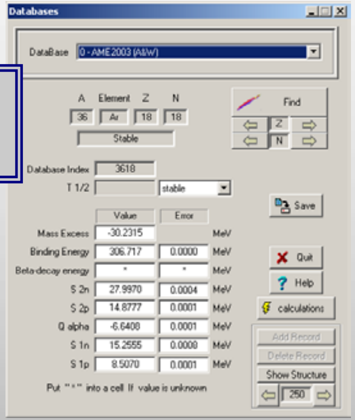
New code: LISE++ ⊗ MOTER = LISE_RAY??

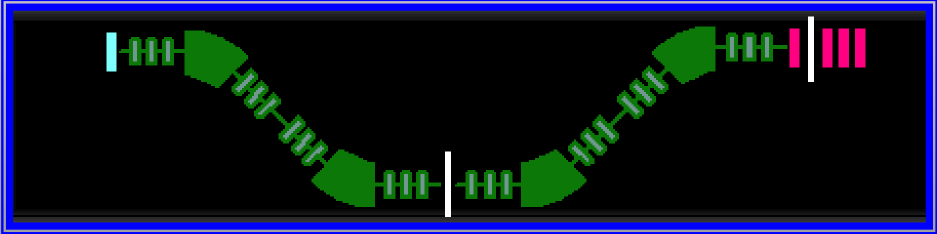
LISE physics: Production mechanisms, energy loss etc



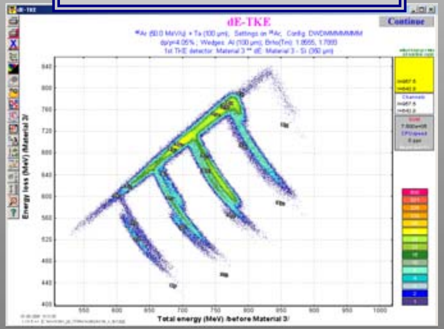
A la "LISE" shell to construct a spectrometer

LISE databases

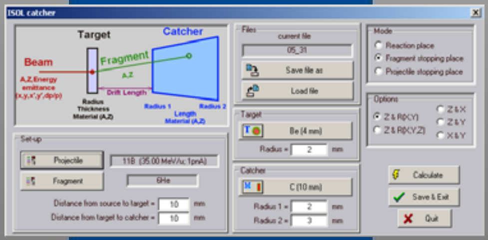




LISE graphics



LISE ISOL catcher



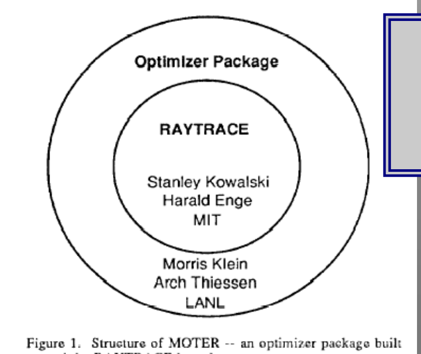


Figure 1. Structure of MOTER -- an optimizer package built around the RAYTRACE kernel.

MOTER: optics and optimization

LISE_RAY file can be adapted by LISE++, but in other side?