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
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)

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

TwinSOL: RNB production facility

6 meter long= flight path
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
TwinSol-solenoid class library development

```cpp
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:

```fortran
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

```cpp
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
\[ V_0 = 1.389 \times 10^7 \times \sqrt{E/M} \]
by LISE++ relativistic functions
\[ V_0 = v_c \cdot m_s \times E_{\text{to}\_\beta} (\text{energy}); \]

Using LISE++ library the substitution was done:

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

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 = (.04321*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,FERMI, 
& 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)
```
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 $B \times L > B_{rho}$, 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.

1) Entrance face:

$$
\begin{align*}
R_i &= \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & K & 0 & 0 & 0 \\
-K & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 
\end{pmatrix} \\
2 \cdot K &= \frac{B}{B_{rho}}
\end{align*}
$$

2) Exit face:

$$
\begin{align*}
R_o &= \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -K & 0 & 0 & 0 \\
K & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 
\end{pmatrix} \\
2 \cdot K &= \frac{B}{B_{rho}}
\end{align*}
$$

3) Homogeneous field:

$$
\begin{align*}
R_h &= \begin{pmatrix}
1 & S*C/K & 0 & S*S/K & 0 & 0 \\
0 & 2*C*C-1 & 0 & 2*S*C & 0 & 0 \\
0 & -S*S/K & 1 & S*C/K & 0 & 0 \\
0 & -2*S*C & 0 & 2*C*C-1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 
\end{pmatrix} \\
C &= \cos(K*L) \\
S &= \sin(K*L)
\end{align*}
$$

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:
TwinSol matrix solution: soft-edge corrections

Alex Bogacz, Workshop on Muon Collider Simulations, Miami Beach, FL December 15, 2004

- 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:

    \[ \Phi_{\text{edge}} = \frac{1}{2} \left( \int_{-\infty}^{\infty} B_z^2(s) \, ds - B_0^2 L \right) = -\frac{k^2 a}{\delta} \quad k = eB_0/pc \]

- Axially symmetric quadrupole

\[ M_{\text{soft sol}} = M_{\text{edge}} M_{\text{sol}} M_{\text{edge}} \quad M_{\text{edge}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -\Phi_{\text{edge}} & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\Phi_{\text{edge}} \end{bmatrix} \]
The TwinSol utility dialog in LISE++
The Solenoid block dialog

Solenoid

setting fragment
40Ar18+ (40.0 MeV/u)

Magnetic field

Use the electrode
Electrode settings

Fill by gas
Gas settings

Under construction (2nd stage)

Geometry

1-st half = 2.2964 m
2-nd half = 3.4064 m

Coil length = 0.8128 m

Effective radius = 0.2123 m

Block Length = 5.7028 m

Solenoid length = 1 m

Bore = 0.15 m

Length of the solenoid block

Length of 1st half

Length of 2nd half

Coil length

Effective radius

Solenoid length

Bore

June 9, 2006, NSCL/MSU
Calculation results plot: 1 solenoid, $L^*B(0)/B_r < 1$

Twin Sol

$^{}^{36}$Ar$^{+}$  (E=40 MeV/u or $p_{trans}=0.813$ GeV/c)  Emittance 1.20,1.20  Int Ray 5.20,-5.25
1st SOL: L1=2.3m L2=3.4m Coil=0.8m $B_0=4.20$T Effield=No
Calculation results plot: 2 solenoids, $L^*B(0)/B_r >> 1$

**Twin Sol**

- $E=2.00$ MeV/u or $P_{trans}=0.136$ GeV/c
- Emittance: 1.20, 1.20
- Int. Ray: 20, 5.25
- 1st SOL: $L_1=8.0$ m, $L_2=8.0$ m, Coil=10.0 m, $B_0=3.000$ T, Efield=No
- 2nd SOL: $L_1=6.0$ m, $L_2=6.0$ m, Coil=10.0 m, $B_0=2.000$ T, Efield=No
II. The Kinematics calculator: Mott scattering (1)

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

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

b. **Identical particles**: 
   \[
   d\sigma(\theta) = \left( \frac{q_1 q_2}{4 E} \right)^2 \left[ \frac{1}{\sin^4(\theta/2)} + \frac{\delta_q}{\cos^4(\theta/2)} + \delta_q \frac{(-1)^{2s}}{(2s+1) \cos^2(\theta/2) \sin^2(\theta/2)} \cos[\eta \log \tan^2(\theta/2)] \right] d\omega
   \]

Coulomb scattering \(^{208}\text{Pb} (873\text{MeV}) + ^{208}\text{Pb} \) (classical case)
The Kinematics calculator: Mott scattering (2)

LISE++ calculation: Mott scattering of identical particles

Experiment: Mott scattering of identical particles

\(^{208}\text{Pb} (873\text{MeV}) + ^{208}\text{Pb}\).

June 9, 2006, NSCL/MSU
The Kinematics calculator: \((A^*, A+\gamma)\)

Thanks for the idea to Dr. V. Goldberg (TAMU)
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.
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!!!
Morris’s Optimized Tracing of Enge’s Rays

IV. “MOTER” ray trace code for MS Windows
(FORTRAN and C++ versions)

Prof. B. Sherrill
Prof. D. Morrissey
Transformation to C++
S. Lobastov (Dubna)

FORTRAN file assignment is the next:

<table>
<thead>
<tr>
<th>FORTRAN</th>
<th>C++</th>
</tr>
</thead>
<tbody>
<tr>
<td>fort.12</td>
<td>K8ts.mag</td>
</tr>
<tr>
<td>fort.14</td>
<td>K8ts.dem</td>
</tr>
<tr>
<td>fort.19</td>
<td>For019.dat</td>
</tr>
<tr>
<td>fort.6</td>
<td>K8ts.out</td>
</tr>
<tr>
<td>fort.8</td>
<td>K8ts.opt</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>C++ version</th>
<th>moter_root \ CCMoter</th>
</tr>
</thead>
<tbody>
<tr>
<td>executable file</td>
<td>moter_root \ CCMoter\Moter.exe</td>
</tr>
<tr>
<td>data files in the directory</td>
<td>moter_root \ CCMoter\data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FORTRAN version</th>
<th>moter_root \ FMoter</th>
</tr>
</thead>
<tbody>
<tr>
<td>executable file</td>
<td>moter_root \ FMoter \ Debug \ FMoter.exe</td>
</tr>
<tr>
<td>data files in the directory</td>
<td>moter_root \ FMoter\debug</td>
</tr>
</tbody>
</table>
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 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 intensity is always equal to 1 pnA
Reaction place

Projectile: $^{11}\text{B}$ (35.0 MeV/u); Fragment: $^6\text{He}$
Target: Be (2 mm), Radius: 5.0 mm; Source-Target distance: 10.0 mm
Catcher: C (10 mm), Radius: 15.0-20.0 mm; Target-Catcher distance: 50.0 mm
Fragment stopping place

Projectile: **^11_B** (35.0 MeV/u); Fragment: **^6_He**
Target: Be (2 mm), Radius: 5.0 mm; Source-Target distance: 10.0 mm
Catcher: C (10 mm), Radius: 15.0-20.0 mm; Target-Catcher distance: 50.0 mm
Projectile stopping place

Projectile: $^{11}$B (35.0 MeV/u); Fragment: $^{8}$He
Target: Be (2 mm); Radius: 5.0 mm; Source-Target distance: 2.0 mm
Catcher: C (10 mm); Radius: 15.0-20.0 mm; Target-Catcher distance: 2.0 mm
Fragment stopping place + contours

Fragment stopping place

Projectile: $^1$B (95.0 MeV/u); Fragment: $^6$He

Target: Be (4 mm), Radius: 2.0 mm; Source-Target distance: 10.0 mm

Contour
Sum: 3.02e+03
Max: 3
XX: 27.07
YY: 1.159
dX: 2.523
dY: 0.564

SUM
7.243e+03
CPU speed
0 pos
Eth.: <0.7%

Fragment rate
7.29e+05 pps
Survived: 87.9 %
$^7$Li (35MeV/u) + Be $\Rightarrow$ $^6$He (A.Rodin et al. Acculina, Dubna)
$^7$Li (35MeV/u) + Be $\rightarrow$ $^4$He (A.Rodin et al. Acculina, Dubna)

Cross sections (mb)

- EPAX = 20.6
- AA = 284.
- Exp = 119.

$^7$Li (35.4 MeV/a) + Be (555mg/cm$^2$) $\rightarrow$ $^4$He
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
- 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 ISOL catcher

LISE graphics

MOTER:
optics and optimization

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