User Programming in the FLUKA environment

FLUKA Advanced Course

Why user routines

- Fluka offers a rich choice of **built-in options** for scoring most quantities and for applying variance reduction techniques, without requiring the users to write a single line of code
- However there are special cases where "ad-hoc" routines are unavoidable, because the needed information cannot be obtained through standard options

What is available for the users

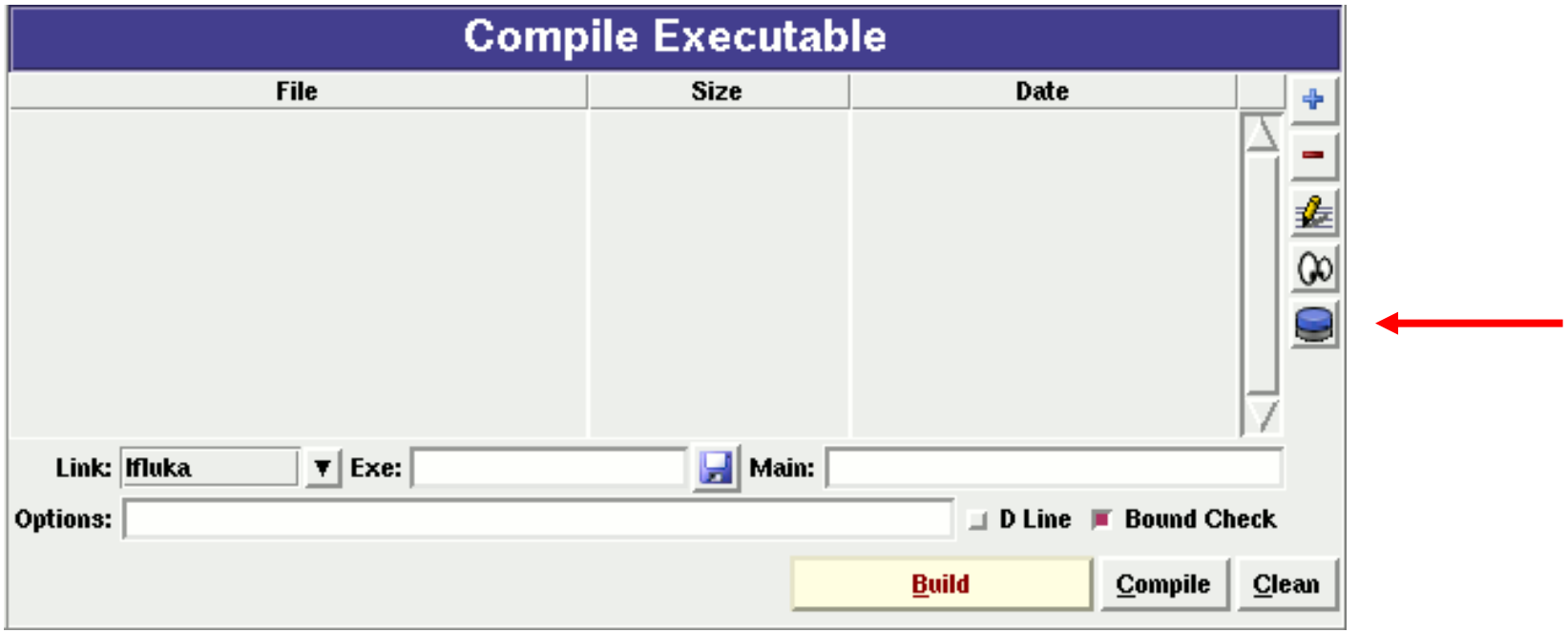
- A number of **user routine** templates are available in the $FLUPRO/usermvax directory and can be modified/activated by the user in order to fulfill non-standard tasks
- The **INCLUDE** files containing the COMMON blocks are in the $FLUPRO/flukapro directory
- An extended **mathematical library** can in principle be exploited by properly calling its members from inside an user routine
- The **compiling and linking scripts** are in the directory $FLUPRO/flutil

Flair can be used to edit, compile and link user routines in order to build a user-specific FLUKA executable

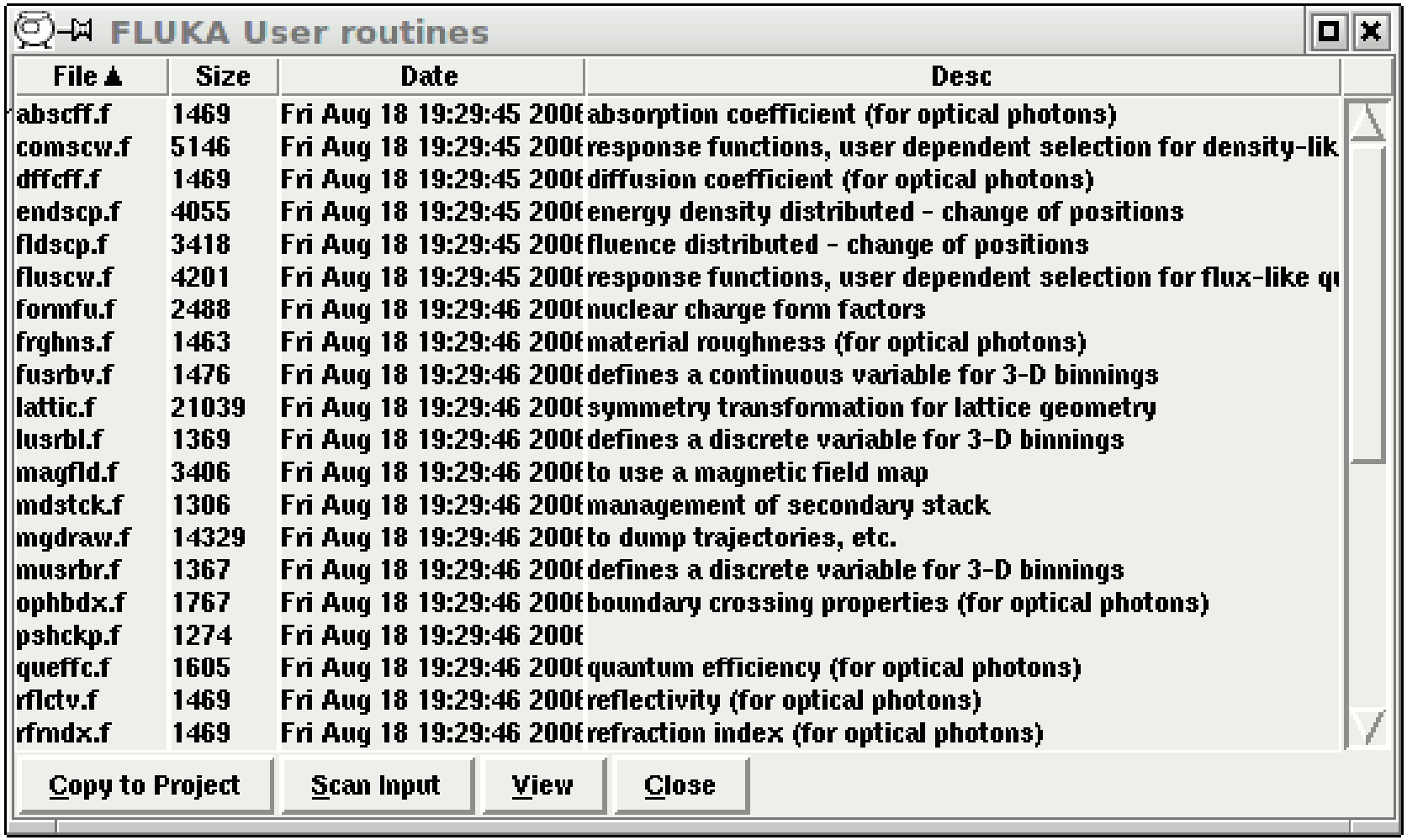
Flair interface (I)

Flair has a button in the Compile frame which scans the input file for possible cards that require an user routine

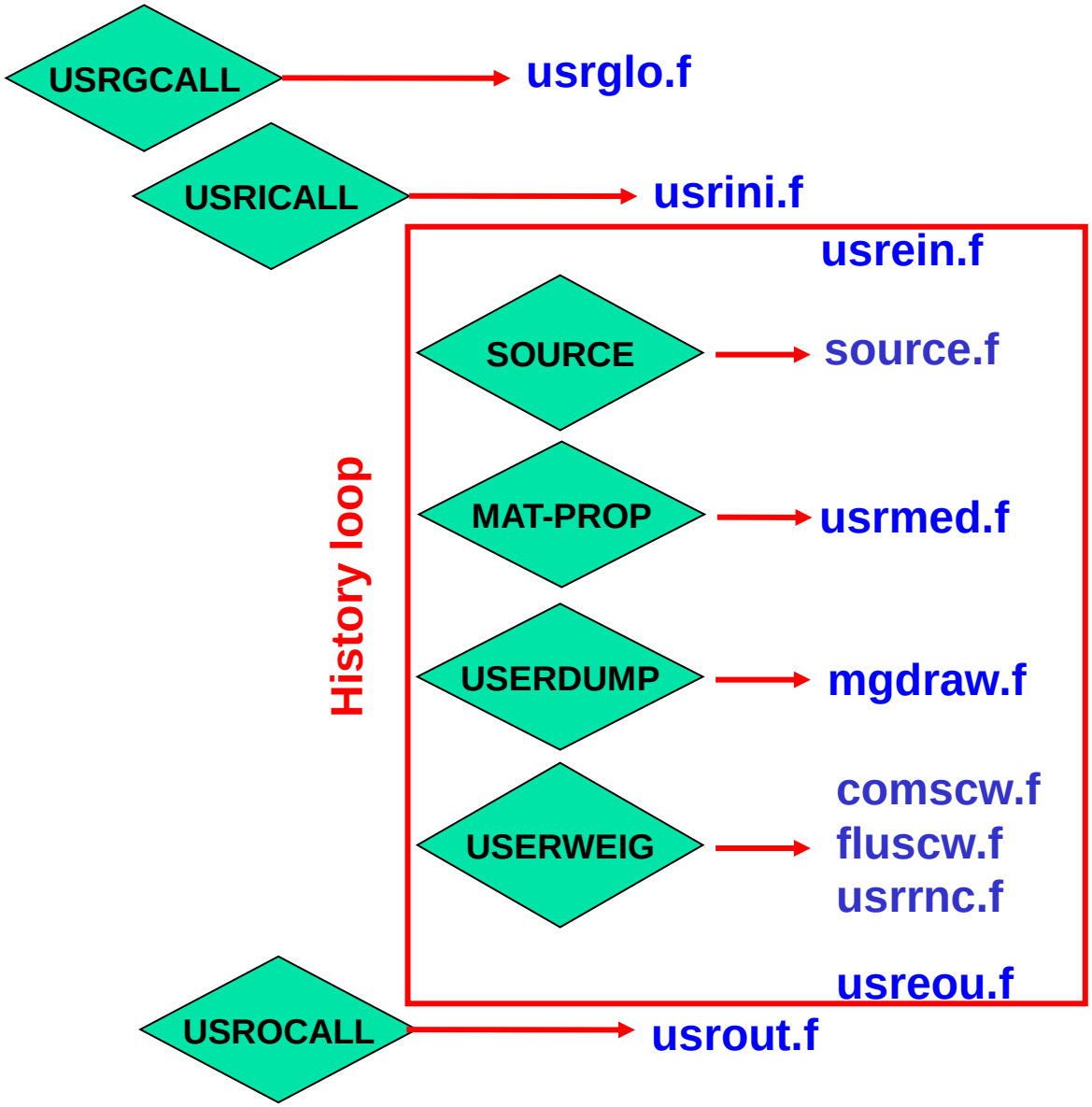
It allows to copy the template routine from $FLUPRO/usermvax to the project directory



Flair interface (II)



Card – user routine correspondence



User routine scope (I)

SCORING

- comscw.f
- fluscw.f
- endscp.f
- fldscp.f
- musrbr.f
- lusrbl.f
- fusrbv.f
- usrrnc.f

BIASING

- usbset.f
- usimbs.f
- udcdrl.f

LATTICE GEOMETRY

- lattic.f

SOURCE GENERATION

- source.f
- (soevsv.f)

MAGNETIC FIELD

- magfld.f

OPTICAL PHOTONS

- abscff.f
- dffcff.f
- frghns.f
- ophbdx.f
- queffc.f
- rflctv.f
- rfrndx.f

INITIALIZATION

- usrglo.f
- usrini.f
- usrein.f

OUTPUT

- usreou.f
- usrout.f

User routine scope (II)

accessing
particle stack

- mdstck.f
- stupre.f
- stuprf.f

accessing
(almost) everything

- mgdraw.f

multipurpose

- usrmed.f

Compiling and linking

- A FLUKA executable with user routines is in general application specific. It must be named and kept separately from the standard FLUKA
- Everything is managed today by FLAIR, however it is important to know the following details (managed automatically inside FLAIR):
- $FLUPRO/flutil/fff is the compiling script with the proper path to the INCLUDE subdirectory and the required compiler (g77) options

  Example: $FLUPRO/flutil/fff usrini.f generates usrini.o

  then $FLUPRO/flutil/lfluka –m fluka –o flukamy usrini.o will perform the proper linking generating the executable here called flukamy
- Tip: $FLUPRO/flutil/lfluka –m fluka –o flukamy usrini.f will automatically call $FLUPRO/flutil/fff

FLUKA programming rules

- Language is Fortran 77 (C routines can be linked)
- Double Precision everywhere, except for integer variables beginning with a letter in the range [i-n]
- Common blocks are in $FLUPRO/flukapro files and are loaded by the INCLUDE statement
- Each routine must start with the following includes/common blocks:

```
INCLUDE '(DBLPRC)'
INCLUDE '(DIMPAR)'
INCLUDE '(IOUNIT)'
```

Note the parentheses which are an integral part of the Fluka INCLUDE file names

- Users may add other FLUKA commons as well as their own commons which may reside in different places

Some COMMON blocks in short

BEAMCM:	beam particle properties (from BEAM and BEAMPOS)
SOURCM:	user variables and information for a user-written source
SOUEVT:	recording of the source event
CASLIM:	number of primary particles followed
FLKSTK:	main particle stack of FLUKA
EMFSTK:	particle stack for electrons and photons
GENSTK:	properties of secondaries created in a hadronic event
FHEAVY:	special stack for nuclear fragments
FLKMAT:	material properties
LTCLCM:	LaTtice CeLl CoMmon for lattice cell identification
TRACKR:	properties of the particle currently transported
PAPROP:	intrinsic particle properties (mass, charge, half live…)
SCOHLP:	variables concerning the current estimator type

(DBLPRC) (I)

DouBLe PReCision common
Included in all routines of Fluka, contains the declaration

IMPLICIT DOUBLE PRECISION (A-H,O-Z)

and sets many mathematical and physical constants.
Users are strongly encouraged to adhere to "Fluka style" by

- using systematically double precision (except for very good reasons such as calling external single precision scoring packages)
- and to use constants defined in this file for maximum accuracy.

(DBLPRC) (II)

```
*========== M A T H E M A T I C A L C O N S T A N T S ==========*

* -------- Numerical constants (double precision): --------*
* Zerzer = 0 *
PARAMETER ( ZERZER = 0.D+00 )
* Oneone = 1 *
PARAMETER ( ONEONE = 1.D+00 )
* Twotwo = 2 *
PARAMETER ( TWOTWO = 2.D+00 )
* Pipipi = Circumference / diameter *
PARAMETER ( PIPIPI = 3.14159265358979323846264338327 9D+00 )
* Twopip = 2 x Pipipi *
PARAMETER ( TWOPIP = 6.28318530717958647692528676655 9D+00 )
* Eneper = "e", base of natural logarithm *
PARAMETER ( ENEPER = 2.71828182845904523536028747135 3D+00 )
* Sqrtwo = square root of 2 *
PARAMETER ( SQRTWO = 1.41421356237309504880168872421 0D+00 )
```

(DBLPRC) (III)

```
*========== P H Y S I C A L C O N S T A N T S ==========*
* -------- Primary constants: -------- *
* Clight = speed of light in cm s-1 *
PARAMETER ( CLIGHT = 2.99792458 D+10 )
* Boltzm = k Boltzmann constant (J K-1) *
PARAMETER ( BOLTZM = 1.380658 D-23 )
* Amelgr = electron mass (g) *
PARAMETER ( AMELGR = 9.1093897 D-28 )
* Plckbr = reduced Planck constant (erg s) *
PARAMETER ( PLCKBR = 1.05457266 D-27 )

* -------- Derived constants: -------- *
*Alamb0 = Compton wavelength = 2 pi r0 / fsc , being r0 the classical electron radius *
* and fsc the fine structure constant *
PARAMETER ( ALAMB0 = TWOTWO * PIPIPI * RCLSEL / ALPFSC )

* -------- Astronomical constants: -------- *
* Rearth = Earth equatorial radius (cm) *
PARAMETER ( REARTH = 6.378140 D+08 )

* -------- Conversion constants: -------- *
* GeVMeV = from GeV to MeV *
PARAMETER ( GEVMEV = 1.0 D+03 )
```

(IOUNIT)

Logical input and output unit numbers

The logical units up to 19 (included) are reserved for FLUKA

```
* lunin = standard input unit *
PARAMETER ( LUNIN = 5 )
* lunout = standard output unit *
PARAMETER ( LUNOUT = 11 )
* lunerr = standard error unit *
PARAMETER ( LUNERR = 15 )
...
```

Use the pre-defined output units when you need messages from your user routines:

```
WRITE ( LUNOUT, *)   ' My initialization is active'
WRITE (LUNERR, *)    '  MySource : warning, energy is 0'
```

(CASLIM)

Keeps preset number of histories and current number of histories

```
*     /caslim/ is needed to decide when to stop the run                *
*       Trnlim = if cpu-time-left<tlim the run will be ended           *
*       Tpmean = is the average time needed for the following of one beam particle *
*       Tprmax = is the maximum time needed for the following of one beam particle *
*       Trntot = the cumulative time needed to follow the beam particles *
*       Ncases = maximum number of beam particles to be followed       *
*              modulo 1,000,000,000)                                   *
*       Mcases = maximum number of beam particles to be followed       *
*              in excess of 1,000,000,000, divided by 1,000,000,000 *
*       Ncase  = current number of beam particles followed (modulo     *
*              1,000,000,000)                                          *
*       Mcase  = current number of beam particles followed in excess   *
*              of 1,000,000,000, divided by 1,000,000,000              *
```

Useful to be included whenever the current event number is needed

(FLKSTK)

```
*    /Flkstk/ stack for the primaries                                    *
*       Wtflk  = particle statistical weight                              *
*       Pmoflk = particle (laboratory) momentum (GeV/c)                     *
*       Tkeflk = particle (laboratory) kinetic energy (GeV)                *
*       Xflk   = particle position  x-coordinate                          *
*       Yflk   = particle position  y-coordinate                          *
*       Zflk   = particle position  z-coordinate                          *
*       Txflk  = particle direction x-coordinate                           *
*       Tyflk  = particle direction y-coordinate                          *
*       Tzflk  = particle direction z-coordinate                          *
*       Txpol  = x direction cosine of the particle polarization          *
*       Typol  = y direction cosine of the particle polarization           *
*       Tzpol  = z direction cosine of the particle polarization           *
*       Dfnear = distance to the nearest boundary                          *
*       Agestk = age of the particle (seconds)                            *
*       Cmpath = cumulative path travelled by the particle since it was produced (cm) *
*       Iloflk = particle identity (Paprop numbering)                       *
*       Igroup = energy group for low energy neutrons                      *
*       Loflk  = particle generation                                      *
*       Louse  = user flag                                              *
*       Nrgflk = particle region number                                   *
*       Nlattc = particle lattice cell number                             *
```

(PAPROP)

intrinsic PArticle PROPerties

```
*       am    (i) = i_th particle mass (GeV)                              *
*       ichrge(i) = electric charge of the i_th particle                  *
*       ibarch(i) = baryonic charge of the i_th particle                  *
*       ijdisc(i) = flag for discarding the i_th particle type            *
*       tmnlf (i) = mean (not half!) life of the i_th particle (s)        *
*       biasdc(i) = decay biasing factor for the i_th particle            *
*       biasin(i) = inelastic interaction biasing factor for the i_th particle *
*       lhadro(i) = True if the i_th particle type is a hadron            *
*       jspinp(i) = i_th particle spin (in units of 1/2)                  *
*       iparty(i) = i_th particle parity (when meaningful)                *
```

(FLKMAT)

FLuKa MATerials

```
*      Amss(i) = Atomic weight (g/mole) of the i_th material          *
*       Rho(i) = Density of the i_th material                         *
*      Ztar(i) = Atomic number of the i_th material                   *
*    Ainlng(i) = Inelastic scattering length of the i_th material    *
*                for beam particles at the average beam energy in cm *
*    Aellng(i) = Elastic scattering length of the i_th material for  *
*                beam particles at average beam energy in cm         *
*     X0rad(i) = Radiation length of the i_th material in cm         *
*    Dmgene(i) = Damage energy of the i_th material (GeV)            *
*    Ainnth(i) = Inelastic scattering length of the i_th material    *
*                for neutrons at threshold energy in cm              *
*    Medium(k) = Material number of the k_th region                  *
*    Mssnum(i) = Mass number of the target nucleus for the i_th material *
*                if =< 0 it means that it is in the natural isotopic composition *
*    Libsnm(i) = flag whether inelastic interaction biasing must be done for this medium *
*    Matnam(i) = Alphabetical name of the i_th material number       *
*    Aocmbm(i) = Atomic density of the i_th material in barn^-1 cm^-1 *
*                (Atoms Over Cm times Barn for Materials)            *
*    Eocmbm(i) = Electron density of the i_th material in barn^-1cm^-1*
*                (Atoms Over Cm times Barn for Materials)            *
```

(FHEAVY)

```
*      npheav     = number of secondaries                          *
*      kheavy(ip) = type of the secondary ip                        *
*                ( 3 = deuteron, 4 = 3-H, 5 = 3-He, 6 = 4-He,      *
*                  7-12 = "Heavy" fragment specified by Ibheav and Icheav ) *
*      cxheav(ip) = direction cosine of the secondary ip with respect to x-axis *
*      cyheav(ip) = direction cosine of the secondary ip with respect to y-axis *
*      czheav(ip) = direction cosine of the secondary ip with respect to z-axis *
*      tkheav(ip) = kinetic energy of secondary ip                  *
*      pheavy(ip) = momentum of the secondary ip                     *
*      wheavy(ip) = weight of the secondary ip                      *
*      agheav(ip) = "age" of the secondary ip with respect to the interaction time *

*      amheav(kp) = atomic masses of the twelve types of evaporated  *
*                   or fragmented or fissioned particles           *
*      amnhea(kp) = nuclear masses of the twelve types of evaporated *
*                   or fragmented or fissioned particles           *
*      anheav(kp) = name of the kp-type heavy particle              *
*      icheav(kp) = charge of the kp-type heavy particle            *
*      ibheav(kp) = mass number of the kp-type heavy particle       *
```

Note that kp = kheavy(ip) !!!

(TRACKR)

TRACK Recording

Ntrack = number of track segments
Mtrack = number of energy deposition events along the track
0 < i < Ntrack
Xtrack = end x-point of the ith track segment
Ytrack = end y-point of the ith track segment
Ztrack = end z-point of the ith track segment
1 < i < Ntrack
Ttrack = length of the ith track segment
1 < j < Mtrack
Dtrack = energy deposition of the jth deposition event
Dptrck = momentum loss of the jth deposition event

Ntrack > 0, Mtrack > 0 : energy loss distributed along the track
Ntrack > 0, Mtrack = 0 : no energy loss along the track
Ntrack = 0, Mtrack = 0 : local energy deposition (the value and the point are not recorded in Trackr)

```
 COMMON / TRACKR /  XTRACK ( 0:MXTRCK ), YTRACK ( 0:MXTRCK ),
&                   ZTRACK ( 0:MXTRCK ), TTRACK  ( MXTRCK ),
&                   DTRACK  ( MXTRCK ), DPTRCK ( 3,MXTRCK ),
```

(TRACKR) : 2nd part

Jtrack = identity number of the particle: for recoils or
kerma deposition it can be outside the allowed
particle id range, assuming values like:
208: "heavy" recoil
211: EM below threshold
308: low energy neutron kerma
in those cases the id of the particle originating
the interaction is saved inside J0trck (which othe-
rwise is zero)

J0trck = see above

Etrack = total energy of the particle

Ptrack = momentum of the particle (not always defined, if
< 0 must be obtained from Etrack)

Cx,y,ztrck = direction cosines of the current particle

Cx,y,ztrpl = polarization cosines of the current particle

Wtrack = weight of the particle

Wscrng = scoring weight: it can differ from Wtrack if some
biasing techniques are used (for example inelastic
interaction length biasing)

Ctrack = total curved path

Cmtrck = cumulative curved path since particle birth

(TRACKR) : 3rd part

Zfftrk = <Z_eff> of the particle
Zfrttk = actual Z_eff of the particle
Atrack = age of the particle
Wninou = neutron algebraic balance of interactions (both for "high" energy particles and "low" energy neutrons)
Wcinou = charge algebraic balance of interactions (for all interactions)
Spausr = user defined spare variables for the current particle
Ktrack = if > 0 neutron group of the particle (neutron)

Lt1trk = initial lattice cell of the current track (or lattice cell for a point energy deposition)
Lt2trk = final lattice cell of the current track
Iprodc = flag for prompt(=1)/radioactive products(=2)
Ltrack = flag recording the generation number
Llouse = user defined flag for the current particle
Ispusr = user defined spare flags for the current particle

```
...
     &          SPAUSR(MKBMX1), STTRCK, SATRCK, TKNIEL, TKEDPA,
     &          WCINOU,
...
     &          IPRODC, ISPUSR(MKBMX2), LFSSSC, LPKILL
```

(EVTFLG)

EVenT FLaGs:

Flags indicating the event interaction type:

LELEVT = Elastic interaction
LINEVT = Inelastic interaction
LDECAY = Particle decay
LDLTRY = Delta ray production (Moller and Bhabha included)
LPAIRP = Pair production
LBRMSP = Bremsstrahlung
LANNRS = Annihilation at rest
LANNFL = Annihilation in flight
LPHOEL = Photoelectric effect
LCMPTN = Compton effect
LCOHSC = Rayleigh scattering
LLENSC = Low energy neutron scattering
LOPPSC = Optical photon scattering
LELDIS = Electromagnetic dissociation
LRDCAY = Radioactive decay

All **LOGICAL** variables!!!

stuprf.f and stupre.f (I)

SeT User PRoperties for Fluka <Emf> particles

These two functions are used to assign a value to one or more stack user variables when the corresponding particle is loaded onto one of the stacks (FLKSTK for hadrons/muons, and EMFSTK for electrons/positrons/photons).
In each of these stacks the user has access to one integer variable, one integer array and one double precision array.
Each of them is copied to a correspondent variable or array in COMMON TRACKR at the beginning of transport:

Correspondence	FLKSTK	EMFSTK		TRACKR
integer variable:	LOUSE	LOUEMF	$\longrightarrow$	LLOUSE
integer array:	ISPARK	IESPAK	$\longrightarrow$	ISPUSR
double precision array:	SPAREK	ESPARK	$\longrightarrow$	SPAUSR

In this way, user variables can be PROPAGATED and KEPT in memory across tracking and interactions !

stuprf.f and stupre.f (II)

The user can access and modify user variables in TRACKR via subroutine MGDRAW and its entries ENDRAW, SODRAW and especially USDRAW.
STUPRF and STUPRE can be used to copy TRACKR user variables to those of the relevant stack.
Note that a stack OPPHST exists also for optical photons, containing similar user variables and arrays LOUOPP, ISPORK and SPAROK. They can be used in user routines, but they are not handled by STUPRE.

STUPRF is called before loading into stack hadrons, muons, neutrinos and low-energy neutrons. The default version copies to stack the user flags of the parent.

STUPRE is called before loading into stack electrons, positrons and photons.
The default version does nothing (the user variables of the parent particle are already set equal to the original projectile by the various electromagnetic interaction routines). Also the region/position etc. are already set inside the stack arrays.

By default , the last place of the ISPARK array keeps the TRACK NUMBER of the current particle

Typical use of STUPRF/ STUPRE is to keep in memory the "history " of a particle

Stuprf: the default

IJ = ID of interacting particle
MREG,XX,YY,ZZ : region and posi tion of the interaction

```
SUBROUTINE STUPRF ( IJ, MREG, XX, YY, ZZ, NPSECN, NPPRMR )

      INCLUDE '(DBLPRC)'
      INCLUDE '(DIMPAR)'
      INCLUDE '(IOUNIT)'
      INCLUDE '(EVTFLG)'
      INCLUDE '(FLKSTK)'
      INCLUDE '(TRACKR)'

      LOUSE   (NPFLKA)  = LLOUSE
      DO 100 ISPR = 1, MKBMX1
         SPAREK (ISPR,NPFLKA) = SPAUSR (ISPR)
  100 CONTINUE
      DO 200 ISPR = 1, MKBMX2
         ISPARK (ISPR,NPFLKA) = ISPUSR (ISPR)
  200 CONTINUE
*     Increment the track number and put it into the last flag:
      IF ( NPSECN .GT. NPPRMR ) THEN
         IF ( NTRCKS .EQ. 2000000000 ) NTRCKS = -2000000000
         NTRCKS = NTRCKS + 1
         ISPARK (MKBMX2,NPFLKA) = NTRCKS
      END IF
      RETURN
```

Suprf is called once for each particle in the stack of secondaries . NPSECN is the index of the current secondary , NPPRMR is the number of particles still flagged as "primary" (i.e. after elastic interaction

This is the default: copy the TRACKR user variables to the stack.

By default: if this Particle is new, consider it as a new track.

Stuprf:an example

The user need: keep the history of neutrino production from a proton beam
The reaction scheme is :
Proton on thick target-> mesons -> decay into leptons and neutrinos.
reinteractions and multiple decay ($\pi \rightarrow \nu+\mu \rightarrow \nu+\nu+e$)
Want to know: which particle decayed, and where ,
where was produced the meson that decayed and its initial mom.

```
* if decay : store father identity,  energy , r,z
   IF ( LDECAY ) THEN
      SPAREK (1,NPFLKA) = ETRACK
      SPAREK (2,NPFLKA) = SQRT ( XX**2 + YY**2 )
      SPAREK (3,NPFLKA) = ZZ
      ISPARK (1,NPFLKA) = IJ
* If inelastic interaction
   ELSE IF ( LINEVT ) THEN
       ISPARK (2,NPFLKA) = KPART (NPSECN)
       ISPARK (3,NPFLKA) = MREG
       SPAREK (4,NPFLKA) = XX
       SPAREK (5,NPFLKA) = YY
       SPAREK (6,NPFLKA) = ZZ
       SPAREK (7,NPFLKA) = PLR(NPSECN) * CXR (NPSECN)
       SPAREK (8,NPFLKA) = PLR(NPSECN) * CYR (NPSECN)
       SPAREK (9,NPFLKA) = PLR(NPSECN) * CZR (NPSECN)
   END IF
```

LDECAY, LINEVT : from common EVTFLG

Store in the first users variables the energy and identity of the decaying particle , and the position

Use more variables to store the id and momentum of each particle from inelastic interaction

Stuprf:an example

In between interactions/decays, the user variables are copied WITHOUT CHANGES to the trackr common, and back to the stack . They are propagated to i.e. decay secondaries (neutrinos) by the default lines in the stuprf.f routine.
They are accessible from the TRACKR common at every moment.
The user can dump them on disk from , for instance, the mgdraw.f routine (see lecture on scoring for details)

```
 * if I am in detector and it is a neutrino
IF ( NEWREG = MYDETREG)   THEN
IF ( JTRACK .EQ. 5 .OR. JTRACK .EQ. 6 .
  &          OR .JTRACK .EQ. 27 .OR. JTRACK .EQ. 28) THEN
       WRITE (MYDUMP , *) 'dump event ', NCASE
  &   JTRACK,   SNGL(ETRACK),
  &   (ISPUSR(I),I=1,3),
  &  (SNGL(SPAUSR(I)),I=1,9),
  &   SNGL(WEE)
ENDIF
ENDIF
```

Particle indeces of neutrinos

Recover the infos in the TRACKR user arrays and dump them

Written : event number, neutrino ID, neutrino energy, ID of decaying particle, ID of ancestor from last inelastic int., Position of decay, position of last inel.int., momentum of the ancestor from the last inel. Int.

mdstck.f

MDSTCK is called after a nuclear interaction in which at least one secondary particle has been produced, before any biasing is applied, to decide which secondary will be loaded in the main stack for further transport.

The properties of the secondaries are stored in the secondary stack (COMMON GENSTK). With MDSTCK, users can analyse those secondaries, write them to a file, or even modify the content of GENSTK (for instance applying their own biasing). In the latter case, however, it is their responsibility to make sure that energy is conserved, the various physical quantities are still consistent, etc.

usrmed.f (I)

USeR MEDium dependent directives

Argument list

IJ	:	particle type
EKSCO	:	particle kinetic energy (GeV)
PLA	:	particle momentum (GeV/c)
WEE	:	particle weight
MREG	:	previous region number
NEWREG	:	current region number
XX, YY, ZZ	:	particle position
TXX, TYY, TZZ	:	particle direction

Subroutine USRMED is activated by option MAT-PROP with SDUM = USERDIRE, for one or more materials indicated by the user. It is called every time a particle is going to be transported in one of the user-tagged materials.

usrmed.f (II)

Two cases are possible

1) MREG = NEWREG: the particle is going to move from a point inside the medium. The user is normally allowed to change only the particle weight.

simulating *attenuation* of optical photons in an absorbing medium by reducing the photon weight

2) MREG ≠ NEWREG: the particle is going to move from a point on a boundary between two regions. The user may change any of the following: particle weight, current region number, direction cosines.

simulating *refraction*, by changing the direction cosines so that the particle is still inside the new region. To do this, one generally needs the direction cosines of the normal to the surface: TXNOR(NPFLKA), TYNOR(NPFLKA), TZNOR(NPFLKA) (COMMON FLKSTK must be included)

simulating *reflection* (albedo) at a boundary. The direction cosines must be modified according to some reflection law or albedo angular distribution, and NEWREG must be set = MREG

In both cases the weight can also be reduced to account for surface reflectivity

But ... one can also kill the particle by putting WEE=ZERZER
(note that its energy will be lost and not deposited)

and particle **coordinates and energy** can be altered as well !!

a big power implies a big responsibility

Mathematical library

FLUKA contains many mathematical routines of general utility, so in general it should not be necessary to call external mathematical libraries (many taken from SLATEC):

flgaus:	Gaussian adaptative integration
erffun:	Error function
expin1:	E1 exponential function
besi0d:	Bessel function I0 (also I1, J0, J1, K0, K1)
dawsni:	Dawson function
gamfun:	Gamma function
radcub:	Real solutions of 3rd order algebraic equation
flgndr:	Legendre polynomials
yinter, d..intp:	interpolation routines
rordin, rordde:	Sorting of vector values

.....

Also: expansion in Laguerre and Chebyshev polynomials, Bezier fit, and many others...

For users who access the FLUKA source: they are in mathmvax directory

At some time it will be possible to have a short-writeup for their use.

A few examples (I)

```
 EXTERNAL FINTEG
 DOUBLE PRECISION FUNCTION FLGAUS ( FINTEG, XA, XB, EPSEPS, IOPT,
&                                   NXEXP )
* Adaptive Gaussian quadrature routine
```

It gives the integral over the (XA,XB) interval of the product between X**NXEXP and the FINTEG function, to be coded by the user as a separate

```
 DOUBLE PRECISION FUNCTION FINTEG (X)
```

```
SUBROUTINE RADCUB ( AA0, AA1, AA2, AA3, X, X0, NRAD )
* Real solutions of 3rd order algebric equation
```

It computes real solutions of the equation:

A0*X^3++A1*X^2+A2*X+A3=0

The solutions are put in the array X; if there is only one real solution it is put into X(1), while X(2) and X(3) are set to 1.d32. If A0=0 the routine computes standard solutions of a second or first degree equation. If it doesn't exist any real solution the whole array X is set to 1.d32. It is possible to compute solutions with a scale factor X0, to avoid loss of significancy with very large or very small numbers. The flag NRAD records the number of real solutions found.

A few examples (II)

DOUBLE PRECISION FUNCTION GAMFUN (X)

It calculates the double precision complete **Gamma function** for double precision argument X

SUBROUTINE RORDIN (RVECT, ICORR, LEN)

It rearranges a real array in increasing order

SUBROUTINE RORDDE (RVECT, ICORR, LEN)

It rearranges a real array in decreasing order

DOUBLE PRECISION FUNCTION FLGNDR (X, LMAX, PLGNDR)

* Function for **LeGeNDRe polynomials**

It computes P_{lmax} (x) and stores all values P_i (x) for i=0,lmax into the PLGNDR array