FLUKA

Source routines

An introduction to a new approach to source routines

Why user routines?

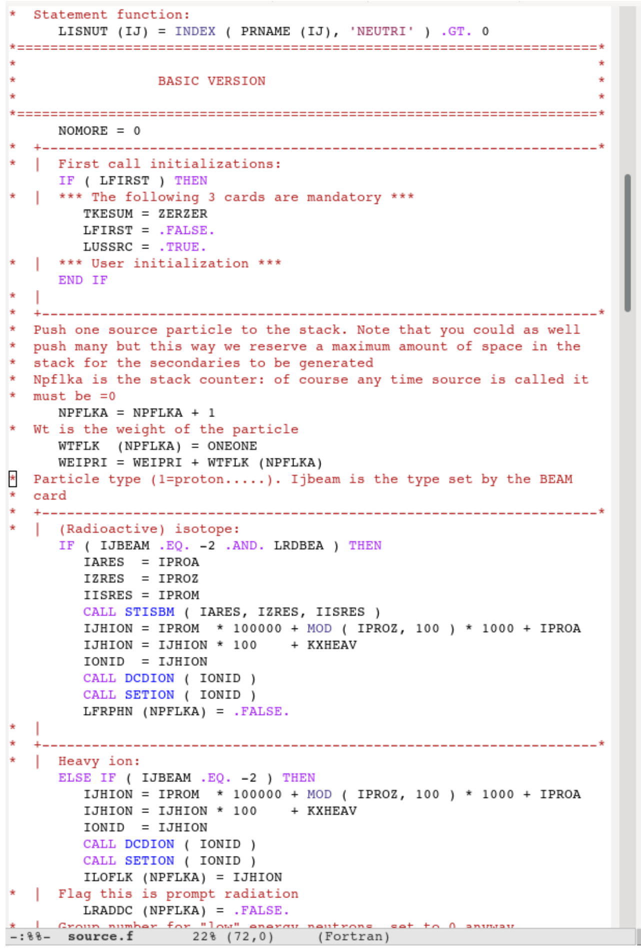
- Fluka offers plenty of built-in tools to define primary beams and estimate quantities
- Sometime these are not enough
- There is the need to write some dedicated code: a “User Routine”
- UR are beyond the scope of this course because of intrinsic difficulties
- Nevertheless, we have a started an effort to make URs more user-friendly
- We want to introduce here the first effort in this direction:

  a new format for the **source routine**
- Why the source routine first? Built-in options allow to sample from a limited number

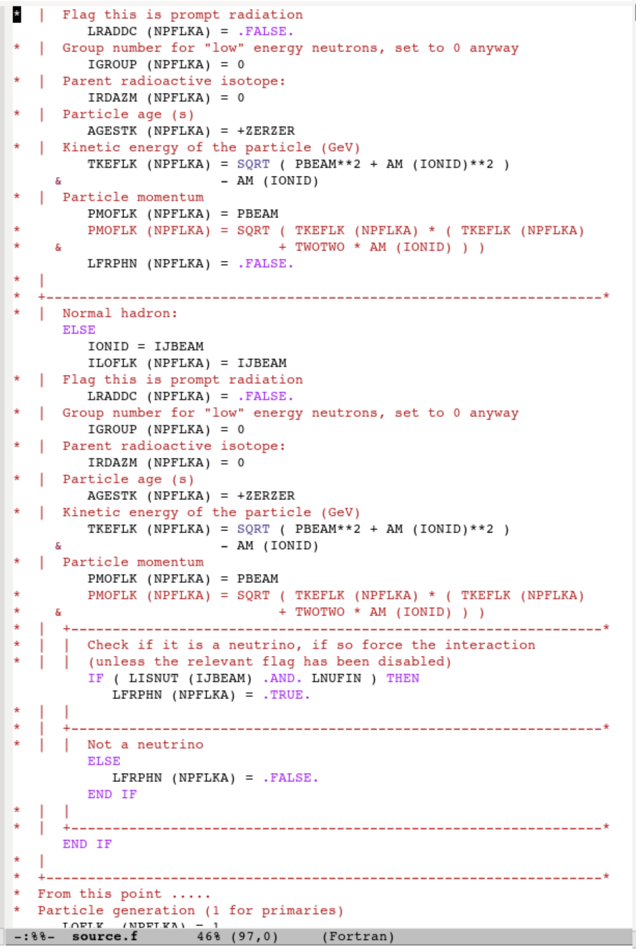
  of distribution and not from histograms. This is an effort to overcome this limitation

The “old” source routine

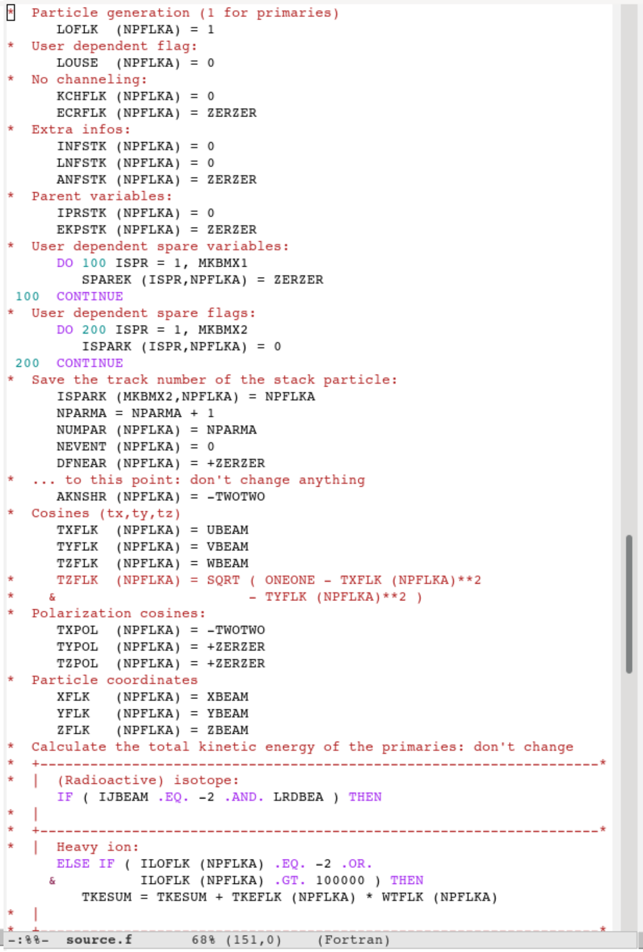
- Scary for beginners, limited documentation
- Use of **IMPLICIT** and **FORTRAN77** naming convention (see later)



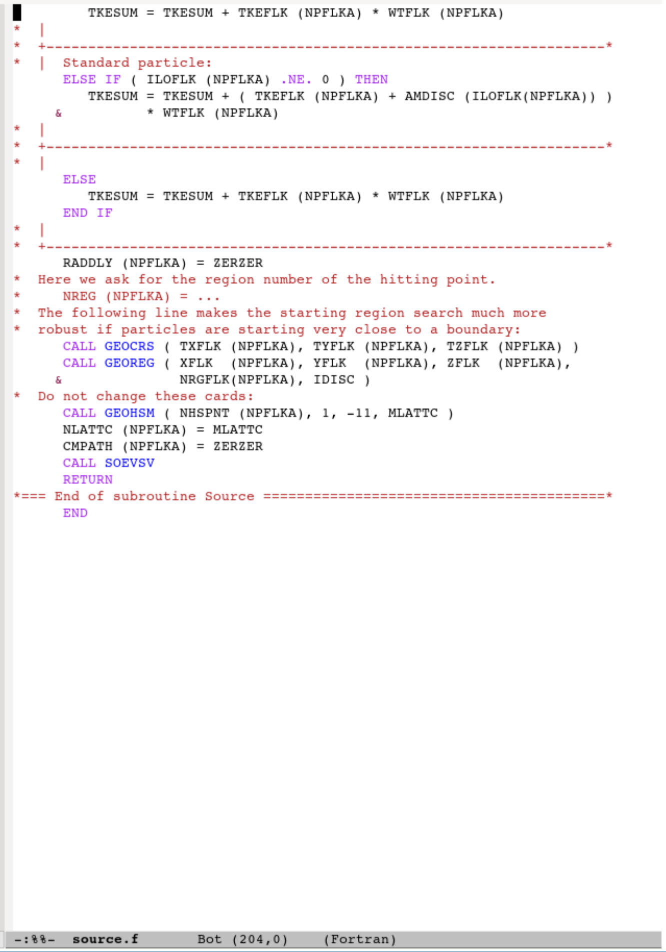
```
*  Statement function:
      LISNUT (IJ) = INDEX ( PRNAME (IJ), 'NEUTRI' ) .GT. 0
*======================================================================*
*                                                                      *
*                 BASIC VERSION                                        *
*                                                                      *
*======================================================================*
      NOMORE = 0
*  +-------------------------------------------------------------------*
*  |  First call initializations:
      IF ( LFIRST ) THEN
*  |  *** The following 3 cards are mandatory ***
         TKESUM = ZERZER
         LFIRST = .FALSE.
         LUSSRC = .TRUE.
*  |  *** User initialization ***
      END IF
*  |
*  +-------------------------------------------------------------------*
*  Push one source particle to the stack. Note that you could as well
*  push many but this way we reserve a maximum amount of space in the
*  stack for the secondaries to be generated
*  Npflka is the stack counter: of course any time source is called it
*  must be =0
      NPFLKA = NPFLKA + 1
*  Wt is the weight of the particle
      WTFLK  (NPFLKA) = ONEONE
      WEIPRI = WEIPRI + WTFLK (NPFLKA)
*  Particle type (1=proton.....). Ijbeam is the type set by the BEAM
*  card
*  +-------------------------------------------------------------------*
*  |  (Radioactive) isotope:
      IF ( IJBEAM .EQ. -2 .AND. LRDBEA ) THEN
         IARES  = IPROA
         IZRES  = IPROZ
         IISRES = IPROM
         CALL STISBM ( IARES, IZRES, IISRES )
         IJHION = IPROM  * 100000 + MOD ( IPROZ, 100 ) * 1000 + IPROA
         IJHION = IJHION * 100    + KXHEAV
         IONID  = IJHION
         CALL DCDION ( IONID )
         CALL SETION ( IONID )
         LFRPHN (NPFLKA) = .FALSE.
*  |
*  +-------------------------------------------------------------------*
*  |  Heavy ion:
      ELSE IF ( IJBEAM .EQ. -2 ) THEN
         IJHION = IPROM  * 100000 + MOD ( IPROZ, 100 ) * 1000 + IPROA
         IJHION = IJHION * 100    + KXHEAV
         IONID  = IJHION
         CALL DCDION ( IONID )
         CALL SETION ( IONID )
         ILOFLK (NPFLKA) = IJHION
*  |  Flag this is prompt radiation
         LRADDC (NPFLKA) = .FALSE.
*  |  Group number for "low" energy neutrons, set to 0 anyway
-:%%-  source.f       22% (72,0)      (Fortran)
```



```
*  |  Flag this is prompt radiation
         LRADDC (NPFLKA) = .FALSE.
*  |  Group number for "low" energy neutrons, set to 0 anyway
         IGROUP (NPFLKA) = 0
*  |  Parent radioactive isotope:
         IRDAZM (NPFLKA) = 0
*  |  Particle age (s)
         AGESTK (NPFLKA) = +ZERZER
*  |  Kinetic energy of the particle (GeV)
         TKEFLK (NPFLKA) = SQRT ( PBEAM**2 + AM (IONID)**2 )
     &                   - AM (IONID)
*  |  Particle momentum
         PMOFLK (NPFLKA) = PBEAM
*        PMOFLK (NPFLKA) = SQRT ( TKEFLK (NPFLKA) * ( TKEFLK (NPFLKA)
*    &                          + TWOTWO * AM (IONID) ) )
         LFRPHN (NPFLKA) = .FALSE.
*  |
*  +-------------------------------------------------------------------*
*  |  Normal hadron:
      ELSE
         IONID = IJBEAM
         ILOFLK (NPFLKA) = IJBEAM
*  |  Flag this is prompt radiation
         LRADDC (NPFLKA) = .FALSE.
*  |  Group number for "low" energy neutrons, set to 0 anyway
         IGROUP (NPFLKA) = 0
*  |  Parent radioactive isotope:
         IRDAZM (NPFLKA) = 0
*  |  Particle age (s)
         AGESTK (NPFLKA) = +ZERZER
*  |  Kinetic energy of the particle (GeV)
         TKEFLK (NPFLKA) = SQRT ( PBEAM**2 + AM (IONID)**2 )
     &                   - AM (IONID)
*  |  Particle momentum
         PMOFLK (NPFLKA) = PBEAM
*        PMOFLK (NPFLKA) = SQRT ( TKEFLK (NPFLKA) * ( TKEFLK (NPFLKA)
*    &                          + TWOTWO * AM (IONID) ) )
*  |  +----------------------------------------------------------------*
*  |  |  Check if it is a neutrino, if so force the interaction
*  |  |  (unless the relevant flag has been disabled)
         IF ( LISNUT (IJBEAM) .AND. LNUFIN ) THEN
            LFRPHN (NPFLKA) = .TRUE.
*  |  |
*  |  +----------------------------------------------------------------*
*  |  |  Not a neutrino
         ELSE
            LFRPHN (NPFLKA) = .FALSE.
         END IF
*  |  |
*  |  +----------------------------------------------------------------*
      END IF
*  |
*  +-------------------------------------------------------------------*
*  From this point .....
*  Particle generation (1 for primaries)
      LOFLK  (NPFLKA) = 1
-:%%-  source.f       46% (97,0)      (Fortran)
```



```
*  Particle generation (1 for primaries)
      LOFLK  (NPFLKA) = 1
*  User dependent flag:
      LOUSE  (NPFLKA) = 0
*  No channeling:
      KCHFLK (NPFLKA) = 0
      ECRFLK (NPFLKA) = ZERZER
*  Extra infos:
      INFSTK (NPFLKA) = 0
      LNFSTK (NPFLKA) = 0
      ANFSTK (NPFLKA) = ZERZER
*  Parent variables:
      IPRSTK (NPFLKA) = 0
      EKPSTK (NPFLKA) = ZERZER
*  User dependent spare variables:
      DO 100 ISPR = 1, MKBMX1
         SPAREK (ISPR,NPFLKA) = ZERZER
 100  CONTINUE
*  User dependent spare flags:
      DO 200 ISPR = 1, MKBMX2
         ISPARK (ISPR,NPFLKA) = 0
 200  CONTINUE
*  Save the track number of the stack particle:
      ISPARK (MKBMX2,NPFLKA) = NPFLKA
      NPARMA = NPARMA + 1
      NUMPAR (NPFLKA) = NPARMA
      NEVENT (NPFLKA) = 0
      DFNEAR (NPFLKA) = +ZERZER
*  ... to this point: don't change anything
      AKNSHR (NPFLKA) = -TWOTWO
*  Cosines (tx,ty,tz)
      TXFLK  (NPFLKA) = UBEAM
      TYFLK  (NPFLKA) = VBEAM
      TZFLK  (NPFLKA) = WBEAM
*     TZFLK  (NPFLKA) = SQRT ( ONEONE - TXFLK (NPFLKA)**2
*    &                       - TYFLK (NPFLKA)**2 )
*  Polarization cosines:
      TXPOL  (NPFLKA) = -TWOTWO
      TYPOL  (NPFLKA) = +ZERZER
      TZPOL  (NPFLKA) = +ZERZER
*  Particle coordinates
      XFLK   (NPFLKA) = XBEAM
      YFLK   (NPFLKA) = YBEAM
      ZFLK   (NPFLKA) = ZBEAM
*  Calculate the total kinetic energy of the primaries: don't change
*  +-------------------------------------------------------------------*
*  |  (Radioactive) isotope:
      IF ( IJBEAM .EQ. -2 .AND. LRDBEA ) THEN
*  |
*  +-------------------------------------------------------------------*
*  |  Heavy ion:
      ELSE IF ( ILOFLK (NPFLKA) .EQ. -2 .OR.
     &          ILOFLK (NPFLKA) .GT. 100000 ) THEN
         TKESUM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)
*  |
-:%%-  source.f       68% (151,0)     (Fortran)
```

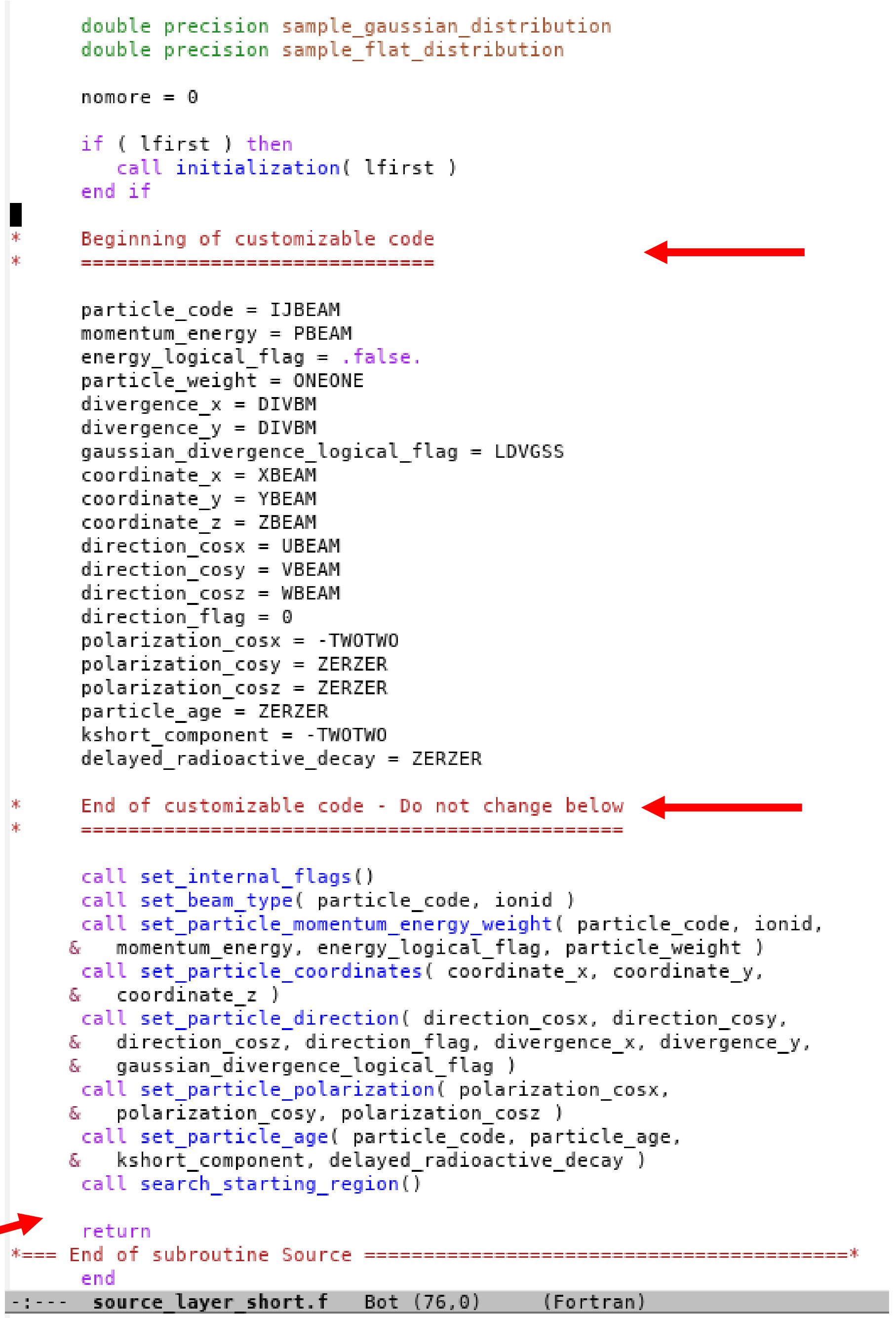


```
         TKESUM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)
*  |
*  +-------------------------------------------------------------------*
*  |  Standard particle:
      ELSE IF ( ILOFLK (NPFLKA) .NE. 0 ) THEN
         TKESUM = TKESUM + ( TKEFLK (NPFLKA) + AMDISC (ILOFLK(NPFLKA)) )
     &          * WTFLK (NPFLKA)
*  |
*  +-------------------------------------------------------------------*
*  |
      ELSE
         TKESUM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)
      END IF
*  |
*  +-------------------------------------------------------------------*
      RADDLY (NPFLKA) = ZERZER
*  Here we ask for the region number of the hitting point.
*     NREG (NPFLKA) = ...
*  The following line makes the starting region search much more
*  robust if particles are starting very close to a boundary:
      CALL GEOCRS ( TXFLK (NPFLKA), TYFLK (NPFLKA), TZFLK (NPFLKA) )
      CALL GEOREG ( XFLK  (NPFLKA), YFLK  (NPFLKA), ZFLK  (NPFLKA),
     &              NRGFLK(NPFLKA), IDISC )
*  Do not change these cards:
      CALL GEOHSM ( NHSPNT (NPFLKA), 1, -11, MLATTC )
      NLATTC (NPFLKA) = MLATTC
      CMPATH (NPFLKA) = ZERZER
      CALL SOEVSV
      RETURN
*=== End of subroutine Source =========================================*
      END
-:%%-  source.f       Bot (204,0)     (Fortran)
```

The “new” source routine

- To be distributed in the next release
- Simplified appearance
- Long & meaningful names for variables and routines
- Use of **`implicit none`** (see later)
- Abundant comments (removed in the snapshot)
- Variables for user’s usage clearly indicated
- Lines not to be edited are “hidden” in routines

  in the **`source_library.inc`** library file
- **Old source routines can still be used**

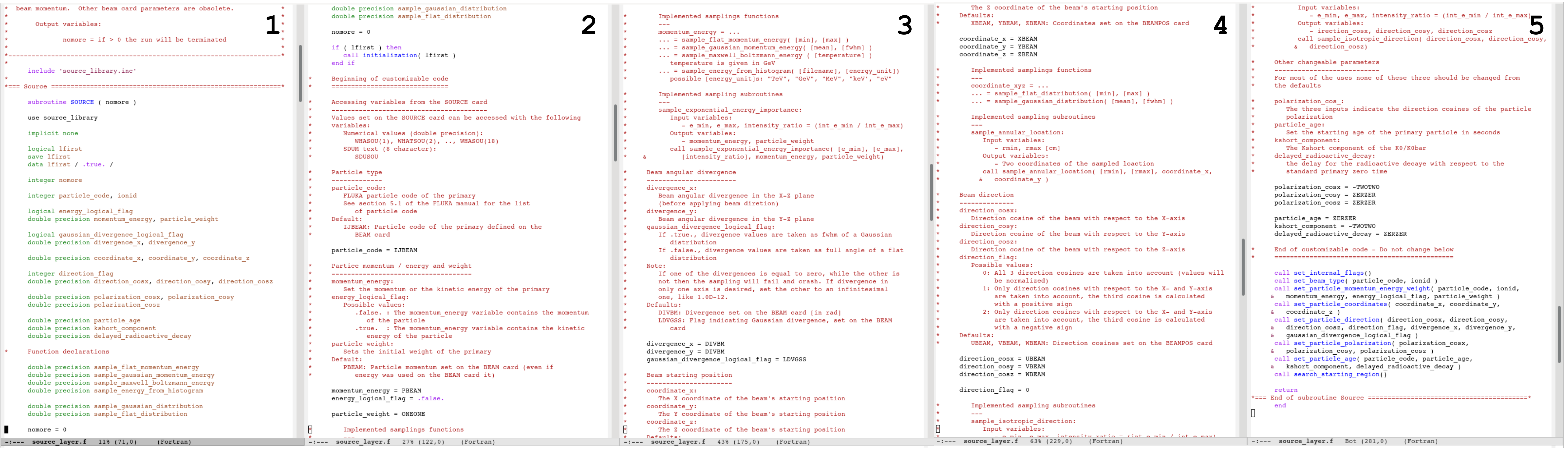


```
      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0

      if ( lfirst ) then
         call initialization( lfirst )
      end if
*
*     Beginning of customizable code
*     ==============================

      particle_code = IJBEAM
      momentum_energy = PBEAM
      energy_logical_flag = .false.
      particle_weight = ONEONE
      divergence_x = DIVBM
      divergence_y = DIVBM
      gaussian_divergence_logical_flag = LDVGSS
      coordinate_x = XBEAM
      coordinate_y = YBEAM
      coordinate_z = ZBEAM
      direction_cosx = UBEAM
      direction_cosy = VBEAM
      direction_cosz = WBEAM
      direction_flag = 0
      polarization_cosx = -TWOTWO
      polarization_cosy = ZERZER
      polarization_cosz = ZERZER
      particle_age = ZERZER
      kshort_component = -TWOTWO
      delayed_radioactive_decay = ZERZER

*     End of customizable code - Do not change below
*     ==============================================

      call set_internal_flags()
      call set_beam_type( particle_code, ionid )
      call set_particle_momentum_energy_weight( particle_code, ionid,
     &   momentum_energy, energy_logical_flag, particle_weight )
      call set_particle_coordinates( coordinate_x, coordinate_y,
     &   coordinate_z )
      call set_particle_direction( direction_cosx, direction_cosy,
     &   direction_cosz, direction_flag, divergence_x, divergence_y,
     &   gaussian_divergence_logical_flag )
      call set_particle_polarization( polarization_cosx,
     &   polarization_cosy, polarization_cosz )
      call set_particle_age( particle_code, particle_age,
     &   kshort_component, delayed_radioactive_decay )
      call search_starting_region()

      return
*=== End of subroutine Source =========================================*
      end
-:---  source_layer_short.f   Bot (76,0)      (Fortran)
```

Comments removed for clarity in this snapshot

The “new” source routine

- Without removing comments (notice the ratio code_lines / comment_lines)
- Note: the snapshot is not meant to be read
- A step by step look will follow



1

```
*  beam momentum.  Other beam card parameters are obsolete.                 *
*                                                                           *
*       Output variables:                                                   *
*                                                                           *
*              nomore = if > 0 the run will be terminated                   *
*                                                                           *
*---------------------------------------------------------------------------*
*
      include 'source_library.inc'
*
*=== Source ================================================================*

      subroutine SOURCE ( nomore )

      use source_library

      implicit none

      logical lfirst
      save lfirst
      data lfirst / .true. /

      integer nomore

      integer particle_code, ionid

      logical energy_logical_flag
      double precision momentum_energy, particle_weight

      logical gaussian_divergence_logical_flag
      double precision divergence_x, divergence_y

      double precision coordinate_x, coordinate_y, coordinate_z

      integer direction_flag
      double precision direction_cosx, direction_cosy, direction_cosz

      double precision polarization_cosx, polarization_cosy
      double precision polarization_cosz

      double precision particle_age
      double precision kshort_component
      double precision delayed_radioactive_decay

*     Function declarations

      double precision sample_flat_momentum_energy
      double precision sample_gaussian_momentum_energy
      double precision sample_maxwell_boltzmann_energy
      double precision sample_energy_from_histogram

      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0
-:---  source_layer.f   11% (71,0)      (Fortran)
```

2

```
      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0

      if ( lfirst ) then
         call initialization( lfirst )
      end if

*     Beginning of customizable code
*     ==============================

*     Accessing variables from the SOURCE card
*     ----------------------------------------
*     Values set on the SOURCE card can be accessed with the following
*     variables:
*        Numerical values (double precision):
*           WHASOU(1), WHATSOU(2), .., WHASOU(18)
*        SDUM text (8 character):
*           SDUSOU

*     Particle type
*     -------------
*     particle_code:
*        FLUKA particle code of the primary
*        See section 5.1 of the FLUKA manual for the list
*           of particle code
*     Default:
*        IJBEAM: Particle code of the primary defined on the
*           BEAM card

      particle_code = IJBEAM

*     Partice momentum / energy and weight
*     ------------------------------------
*     momentum_energy:
*        Set the momentum or the kinetic energy of the primary
*     energy_logical_flag:
*        Possible values:
*           .false. : The momentum_energy variable contains the momentum
*              of the particle
*           .true.  : The momentum_energy variable contains the kinetic
*              energy of the particle
*     particle weight:
*        Sets the initial weight of the primary
*     Default:
*        PBEAM: Particle momentum set on the BEAM card (even if
*           energy was used on the BEAM card it)

      momentum_energy = PBEAM
      energy_logical_flag = .false.

      particle_weight = ONEONE

*        Implemented samplings functions
-:---  source_layer.f   27% (122,0)     (Fortran)
```

3

```
*        Implemented samplings functions
*        ---
*        momentum_energy = ...
*        ... = sample_flat_momentum_energy( [min], [max] )
*        ... = sample_gaussian_momentum_energy( [mean], [fwhm] )
*        ... = sample_maxwell_boltzmann_energy ( [temperature] )
*           temperature is given in GeV
*        ... = sample_energy_from_histogram( [filename], [energy_unit])
*           possible [energy_unit]s: "TeV", "GeV", "MeV", "keV', "eV"

*        Implemented sampling subroutines
*        ---
*        sample_exponential_energy_importance:
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - momentum_energy, particle_weight
*           call sample_exponential_energy_importance( [e_min], [e_max],
*    &         [intensity_ratio], momentum_energy, particle_weight)

*     Beam angular divergence
*     -----------------------
*     divergence_x:
*        Beam angular divergence in the X-Z plane
*        (before applying beam diretion)
*     divergence_y:
*        Beam angular divergence in the Y-Z plane
*     gaussian_divergence_logical_flag:
*        If .true., divergence values are taken as fwhm of a Gaussian
*           distribution
*        If .false., divergence values are taken as full angle of a flat
*           distribution
*     Note:
*        If one of the divergences is equal to zero, while the other is
*        not then the sampling will fail and crash. If divergence in
*        only one axis is desired, set the other to an infinitesimal
*        one, like 1.0D-12.
*     Defaults:
*        DIVBM: Divergence set on the BEAM card [in rad]
*        LDVGSS: Flag indicating Gaussian divergence, set on the BEAM
*           card

      divergence_x = DIVBM
      divergence_y = DIVBM
      gaussian_divergence_logical_flag = LDVGSS

*     Beam starting position
*     ----------------------
*     coordinate_x:
*        The X coordinate of the beam's starting position
*     coordinate_y:
*        The Y coordinate of the beam's starting position
*     coordinate_z:
*        The Z coordinate of the beam's starting position
-:---  source_layer.f   43% (175,0)     (Fortran)
```

4

```
*        The Z coordinate of the beam's starting position
*     Defaults:
*        XBEAM, YBEAM, ZBEAM: Coordinates set on the BEAMPOS card

      coordinate_x = XBEAM
      coordinate_y = YBEAM
      coordinate_z = ZBEAM

*        Implemented samplings functions
*        ---
*        coordinate_xyz = ...
*        ... = sample_flat_distribution( [min], [max] )
*        ... = sample_gaussian_distribution( [mean], [fwhm] )

*        Implemented sampling subroutines
*        ---
*        sample_annular_location:
*           Input variables:
*              - rmin, rmax [cm]
*           Output variables:
*              - Two coordinates of the sampled loaction
*           call sample_annular_location( [rmin], [rmax], coordinate_x,
*          &   coordinate_y )

*     Beam direction
*     --------------
*     direction_cosx:
*        Direction cosine of the beam with respect to the X-axis
*     direction_cosy:
*        Direction cosine of the beam with respect to the Y-axis
*     direction_cosz:
*        Direction cosine of the beam with respect to the Z-axis
*     direction_flag:
*        Possible values:
*           0: All 3 direction cosines are taken into account (values will
*              be normalized)
*           1: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a positive sign
*           2: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a negative sign
*     Defaults:
*        UBEAM, VBEAM, WBEAM: Direction cosines set on the BEAMPOS card

      direction_cosx = UBEAM
      direction_cosy = VBEAM
      direction_cosz = WBEAM

      direction_flag = 0

*        Implemented sampling subroutines
*        ---
*        sample_isotropic_direction:
*           Input variables:
-:---  source_layer.f   63% (229,0)     (Fortran)
```

5

```
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - irection_cosx, direction_cosy, direction_cosz
*           call sample_isotropic_direction( direction_cosx, direction_cosy,
*          &   direction_cosz)

*     Other changeable parameters
*     ---------------------------
*     For most of the uses none of these three should be changed from
*     the defaults

*     polarization_cos_:
*        The three inputs indicate the direction cosines of the particle
*        polarization
*     particle_age:
*        Set the starting age of the primary particle in seconds
*     kshort_component:
*        The Kshort component of the K0/K0bar
*     delayed_radioactive_decay:
*        the delay for the radioactive decaye with respect to the
*        standard primary zero time

      polarization_cosx = -TWOTWO
      polarization_cosy = ZERZER
      polarization_cosz = ZERZER

      particle_age = ZERZER
      kshort_component = -TWOTWO
      delayed_radioactive_decay = ZERZER

*     End of customizable code - Do not change below
*     ==============================================

      call set_internal_flags()
      call set_beam_type( particle_code, ionid )
      call set_particle_momentum_energy_weight( particle_code, ionid,
     &   momentum_energy, energy_logical_flag, particle_weight )
      call set_particle_coordinates( coordinate_x, coordinate_y,
     &   coordinate_z )
      call set_particle_direction( direction_cosx, direction_cosy,
     &   direction_cosz, direction_flag, divergence_x, divergence_y,
     &   gaussian_divergence_logical_flag )
      call set_particle_polarization( polarization_cosx,
     &   polarization_cosy, polarization_cosz )
      call set_particle_age( particle_code, particle_age,
     &   kshort_component, delayed_radioactive_decay )
      call search_starting_region()

      return
*=== End of subroutine Source ==========================================*
      end

-:---  source_layer.f   Bot (281,0)     (Fortran)
```

History of Fortran

- Fortran born in the early 1950s, and the first compiler was released in 1957

Standards:

- Fortran 66 – The first standard
- Fortran 77 – Extension on Fortran 66
- Fortran 90 – Dynamic memory allocation / introduction of the *Free* format
- Fortran 95 – High performance Fortran specification
- Fortran 2003 – Object oriented programming
- Fortran 2008 / 2018 – Extensions of Fortran 2003

FLUKA is still mostly (if not fully) compatible Fortran 77
This doesn't mean that we can't use newer things in our user routines

(Unexpected) Features and limitations of Fortran (77)

- Source file format
  - Fixed
  - Free
- Naming convention
- Subprograms
  - Functions
  - Subroutines
- Variable declaration
  - Implicit
  - Explicit

Source file format

- Fortran 77 uses the *Fixed* file format (extensions: **.f** or **.for**):
  - Maximum 78 characters in one line
  - First 6 are reserved for special function:
    - If the first character is ‘c’ or ‘*’, then the line is a comment
    - If the 6th position is not empty, then the line is treated as a continuation of the previous one (Often the ‘&’ character is used)
  - With the gfortran compiler it is possible to increase the maximum line length
    - In FLUKA 4 it is set to 132

- Fortran 90 introduced the *Free* format (extensions: **.f90**, [**.f95**, etc.]):
  - Code can start at the 1st position

- *Note*: It is not possible to mix both in the same source file
  Gfortran compiler expects the “correct” format based on the file extension.

Naming convention

- Fortran 77 variable and (subprogram) names:
  - Limited to 6 alphanumerical characters
  - Have to start with a letter
  - Case insensitive

- Starting with Fortran 90 the variable names $\longrightarrow$ Feature exploited in the new source routine
  - Can be up to 31 character long
  - Can contain letters, numbers and underscore (‘_’)
  - Have to start with a letter
  - Case insensitive

- *Note*: Try to use descriptive names, to make code readable

Subprograms

- Two types:
  - Function
    - Has a return value
    - Used in assignment: `variable = function(input_variable_1, …)`
  - Subroutine
    - Doesn't have a return value
    - Accessible with the CALL statement: `call subroutine(input_variable_1, …)`

- Passing variables
  - In Fortran you pass the variable, not the value of the variable. (Like passing a pointer in C)
  - This means the subprograms may irreversibly modify the value of the input variables
    - Desired behavior if you want to return multiple variables
    - Can lead to side effects

Variable declaration

- Fortran by default uses *implicit declaration*, which means the type of the variable (integer, real, etc.) is determined by a preset rule.
- The default rule is:
  - If the variable starts with the letter I, J, K, L, M, or N it is an integer,
  - Otherwise it is a real (single precision float)
- In FLUKA however:
  - Variables with the 1st letter I, J, K, L, M, and N are still integers
  - But the others are double precision (floats)

- It is possible (and necessary) to overwrite this with *explicit declaration*, where you manually specify the type of the variable, like:

```
double precision my_intensity
logical my_flag
```

Variable declaration

- Biggest issue is that typos remain hidden:

  If you have a typo in a variable name, the compiler won't raise an error

  It is a different, but valid variable without a value

  Using it in calculations will lead to unexpected results

- Other issue is the unexpected type conversion:

  For example: Information is lost if you want to assign a double precision number to INTEGER

- Solution in the "new" source routine: `implicit none`

  This statement disables the implicit declaration and every variable has to me manually declared

  Exception: FLUKAs built in variables don't need to be declared in the source routine (they will remain implicitly declared)

- Convention in the "new" source routine:
  - Variables with uppercase names: FLUKA variables
  - Variables with lowercase names: explicitly declared variables

Numbers and Constants in User routines

- To keep the high accuracy of the calculation
  - Every variable containing a floating point number should have the type *double precision*
  - The assigned numbers should also be double precision:

    For example: **`radius = 2.0D0`**

    The 'D' character indicated, that this is number should be treated as double precision.

    If it is 'E' or missing, then the number will be single precision

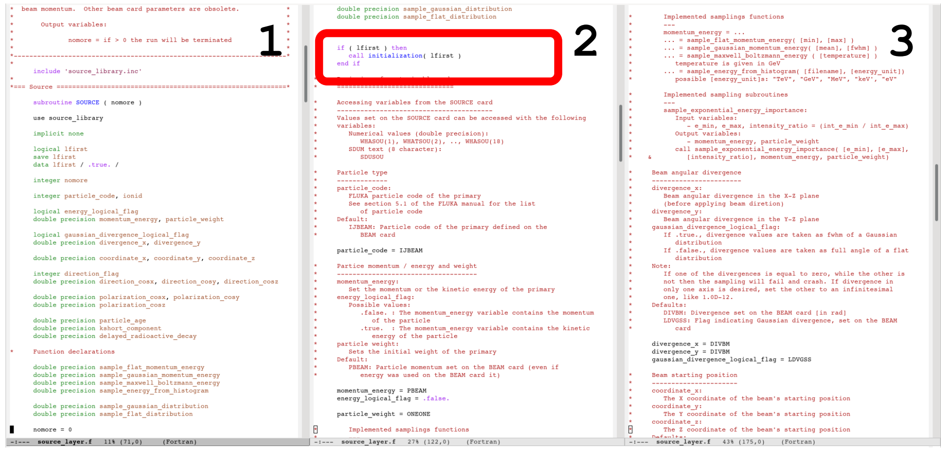
- To simplify writing numbers FLUKA already defined many numbers as variables:
  - **`ONEONE = 1.0D0`**
  - **`TWOTWO = 2.0D0`**
  - **`HLFHLF = 0.5D0`**
  - **`PIPIPI`** = π = **`3.141592`**…
  - **`TWOPIP`** = 2π = **`6.283185`**…

  Full list available in the **`dblprc.inc`** include file

Source routine – initialization

```
if ( lfirst ) then
    call initialization( lfirst )
end if
```

- Initialization of internal variables
- Only performed the first time the routine is called



```
*  beam momentum.  Other beam card parameters are obsolete.             *
*                                                                       *
*       Output variables:                                               *
*                                                                       *
*              nomore = if > 0 the run will be terminated               *
*                                                                       *
*-----------------------------------------------------------------------*
*
      include 'source_library.inc'
*
*=== Source ============================================================*

      subroutine SOURCE ( nomore )

      use source_library

      implicit none

      logical lfirst
      save lfirst
      data lfirst / .true. /

      integer nomore

      integer particle_code, ionid

      logical energy_logical_flag
      double precision momentum_energy, particle_weight

      logical gaussian_divergence_logical_flag
      double precision divergence_x, divergence_y

      double precision coordinate_x, coordinate_y, coordinate_z

      integer direction_flag
      double precision direction_cosx, direction_cosy, direction_cosz

      double precision polarization_cosx, polarization_cosy
      double precision polarization_cosz

      double precision particle_age
      double precision kshort_component
      double precision delayed_radioactive_decay

*     Function declarations

      double precision sample_flat_momentum_energy
      double precision sample_gaussian_momentum_energy
      double precision sample_maxwell_boltzmann_energy
      double precision sample_energy_from_histogram

      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0
-:---  source_layer.f   11% (71,0)     (Fortran)
```

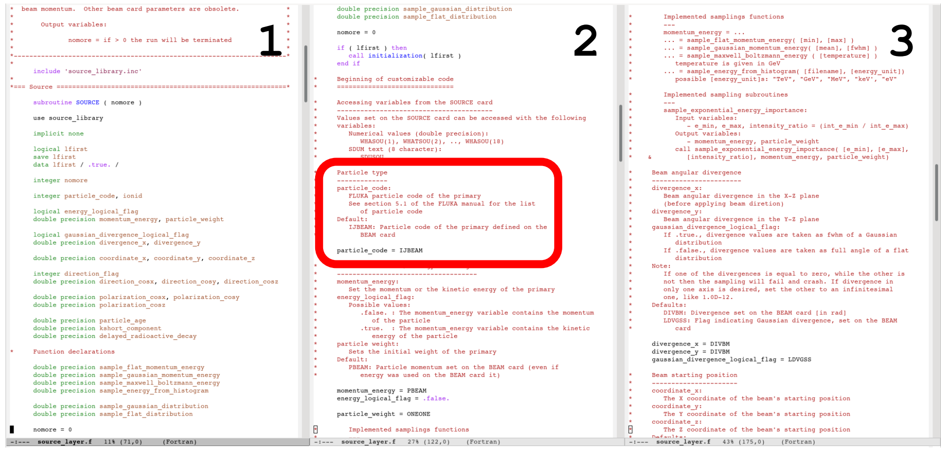
```
      double precision sample_gaussian_distribution
      double precision sample_flat_distribution


      if ( lfirst ) then
         call initialization( lfirst )
      end if

*
*     ==============================

*     Accessing variables from the SOURCE card
*     ----------------------------------------
*     Values set on the SOURCE card can be accessed with the following
*     variables:
*        Numerical values (double precision):
*           WHASOU(1), WHATSOU(2), .., WHASOU(18)
*        SDUM text (8 character):
*           SDUSOU

*     Particle type
*     -------------
*     particle_code:
*        FLUKA particle code of the primary
*        See section 5.1 of the FLUKA manual for the list
*           of particle code
*     Default:
*        IJBEAM: Particle code of the primary defined on the
*           BEAM card

      particle_code = IJBEAM

*     Partice momentum / energy and weight
*     ------------------------------------
*     momentum_energy:
*        Set the momentum or the kinetic energy of the primary
*     energy_logical_flag:
*        Possible values:
*           .false. : The momentum_energy variable contains the momentum
*              of the particle
*           .true.  : The momentum_energy variable contains the kinetic
*              energy of the particle
*     particle weight:
*        Sets the initial weight of the primary
*     Default:
*        PBEAM: Particle momentum set on the BEAM card (even if
*           energy was used on the BEAM card it)

      momentum_energy = PBEAM
      energy_logical_flag = .false.

      particle_weight = ONEONE

*        Implemented samplings functions
-:---  source_layer.f   27% (122,0)    (Fortran)
```

```
*        Implemented samplings functions
*        ---
*        momentum_energy = ...
*        ... = sample_flat_momentum_energy( [min], [max] )
*        ... = sample_gaussian_momentum_energy( [mean], [fwhm] )
*        ... = sample_maxwell_boltzmann_energy ( [temperature] )
*           temperature is given in GeV
*        ... = sample_energy_from_histogram( [filename], [energy_unit])
*           possible [energy_unit]s: "TeV", "GeV", "MeV", "keV', "eV"

*        Implemented sampling subroutines
*        ---
*        sample_exponential_energy_importance:
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - momentum_energy, particle_weight
*           call sample_exponential_energy_importance( [e_min], [e_max],
*    &         [intensity_ratio], momentum_energy, particle_weight)

*     Beam angular divergence
*     -----------------------
*     divergence_x:
*        Beam angular divergence in the X-Z plane
*        (before applying beam diretion)
*     divergence_y:
*        Beam angular divergence in the Y-Z plane
*     gaussian_divergence_logical_flag:
*        If .true., divergence values are taken as fwhm of a Gaussian
*           distribution
*        If .false., divergence values are taken as full angle of a flat
*           distribution
*     Note:
*        If one of the divergences is equal to zero, while the other is
*        not then the sampling will fail and crash. If divergence in
*        only one axis is desired, set the other to an infinitesimal
*        one, like 1.0D-12.
*     Defaults:
*        DIVBM: Divergence set on the BEAM card [in rad]
*        LDVGSS: Flag indicating Gaussian divergence, set on the BEAM
*           card

      divergence_x = DIVBM
      divergence_y = DIVBM
      gaussian_divergence_logical_flag = LDVGSS

*     Beam starting position
*     ----------------------
*     coordinate_x:
*        The X coordinate of the beam's starting position
*     coordinate_y:
*        The Y coordinate of the beam's starting position
*     coordinate_z:
*        The Z coordinate of the beam's starting position
-:---  source_layer.f   43% (175,0)    (Fortran)
```

Source routine – particle type

```
particle_code = IJBEAM
```

- By default the particle type given in the BEAM card is taken (`IJBEAM` variable)
- The particle type can be overridden in the source routine
- Possible application: beam made of more than one type particles
- Particle codes explained in Fluka manual section 5.1



```
*  beam momentum.  Other beam card parameters are obsolete.            *
*                                                                      *
*       Output variables:                                              *
*                                                                      *
*              nomore = if > 0 the run will be terminated              *
*                                                                      *
*----------------------------------------------------------------------*
*
      include 'source_library.inc'
*
*=== Source ===========================================================*

      subroutine SOURCE ( nomore )

      use source_library

      implicit none

      logical lfirst
      save lfirst
      data lfirst / .true. /

      integer nomore

      integer particle_code, ionid

      logical energy_logical_flag
      double precision momentum_energy, particle_weight

      logical gaussian_divergence_logical_flag
      double precision divergence_x, divergence_y

      double precision coordinate_x, coordinate_y, coordinate_z

      integer direction_flag
      double precision direction_cosx, direction_cosy, direction_cosz

      double precision polarization_cosx, polarization_cosy
      double precision polarization_cosz

      double precision particle_age
      double precision kshort_component
      double precision delayed_radioactive_decay

*     Function declarations

      double precision sample_flat_momentum_energy
      double precision sample_gaussian_momentum_energy
      double precision sample_maxwell_boltzmann_energy
      double precision sample_energy_from_histogram

      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0
-:---  source_layer.f   11% (71,0)     (Fortran)
```

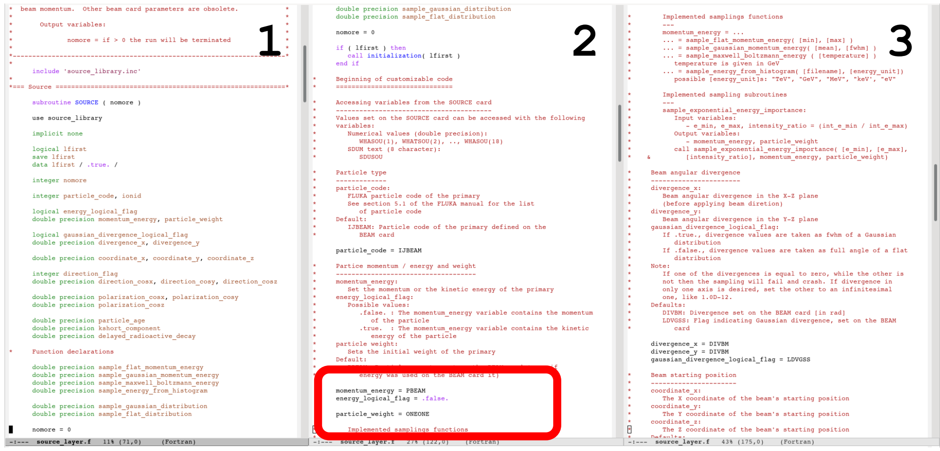
```
      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0

      if ( lfirst ) then
         call initialization( lfirst )
      end if

*     Beginning of customizable code
*     ==============================

*     Accessing variables from the SOURCE card
*     ----------------------------------------
*     Values set on the SOURCE card can be accessed with the following
*     variables:
*        Numerical values (double precision):
*           WHASOU(1), WHATSOU(2), .., WHASOU(18)
*        SDUM text (8 character):
*           SDUSOU

*     Particle type
*     -------------
*     particle_code:
*        FLUKA particle code of the primary
*        See section 5.1 of the FLUKA manual for the list
*           of particle code
*     Default:
*        IJBEAM: Particle code of the primary defined on the
*           BEAM card

      particle_code = IJBEAM

*     ----------------------------------
*     momentum_energy:
*        Set the momentum or the kinetic energy of the primary
*     energy_logical_flag:
*        Possible values:
*           .false. : The momentum_energy variable contains the momentum
*              of the particle
*           .true.  : The momentum_energy variable contains the kinetic
*              energy of the particle
*     particle weight:
*        Sets the initial weight of the primary
*     Default:
*        PBEAM: Particle momentum set on the BEAM card (even if
*           energy was used on the BEAM card it)

      momentum_energy = PBEAM
      energy_logical_flag = .false.

      particle_weight = ONEONE

*        Implemented samplings functions
-:---  source_layer.f   27% (122,0)     (Fortran)
```

```
*        Implemented samplings functions
*        ---
*        momentum_energy = ...
*        ... = sample_flat_momentum_energy( [min], [max] )
*        ... = sample_gaussian_momentum_energy( [mean], [fwhm] )
*        ... = sample_maxwell_boltzmann_energy ( [temperature] )
*           temperature is given in GeV
*        ... = sample_energy_from_histogram( [filename], [energy_unit])
*           possible [energy_unit]s: "TeV", "GeV", "MeV", "keV', "eV"

*        Implemented sampling subroutines
*        ---
*        sample_exponential_energy_importance:
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - momentum_energy, particle_weight
*           call sample_exponential_energy_importance( [e_min], [e_max],
*    &            [intensity_ratio], momentum_energy, particle_weight)

*     Beam angular divergence
*     -----------------------
*     divergence_x:
*        Beam angular divergence in the X-Z plane
*        (before applying beam diretion)
*     divergence_y:
*        Beam angular divergence in the Y-Z plane
*     gaussian_divergence_logical_flag:
*        If .true., divergence values are taken as fwhm of a Gaussian
*           distribution
*        If .false., divergence values are taken as full angle of a flat
*           distribution
*     Note:
*        If one of the divergences is equal to zero, while the other is
*        not then the sampling will fail and crash. If divergence in
*        only one axis is desired, set the other to an infinitesimal
*        one, like 1.0D-12.
*     Defaults:
*        DIVBM: Divergence set on the BEAM card [in rad]
*        LDVGSS: Flag indicating Gaussian divergence, set on the BEAM
*           card

      divergence_x = DIVBM
      divergence_y = DIVBM
      gaussian_divergence_logical_flag = LDVGSS

*     Beam starting position
*     ----------------------
*     coordinate_x:
*        The X coordinate of the beam's starting position
*     coordinate_y:
*        The Y coordinate of the beam's starting position
*     coordinate_z:
*        The Z coordinate of the beam's starting position
-:---  source_layer.f   43% (175,0)     (Fortran)
```

Source routine – particle momentum/energy

```
momentum_energy = PBEAM

energy_logical_flag = .false.
```

- By default the particle momentum given in the **BEAM** card is taken (**PBEAM** variable)
- **PBEAM** is calculated internally by Fluka
- **PBEAM** is always the momentum even if energy was provided in the **BEAM** card
- Energy can be given in the source routine by setting the logical flag as true



```
*  beam momentum.  Other beam card parameters are obsolete.               *
*                                                                         *
*       Output variables:                                                 *
*                                                                         *
*              nomore = if > 0 the run will be terminated                 *
*                                                                         *
*-------------------------------------------------------------------------*
*
      include 'source_library.inc'
*
*=== Source ==============================================================*

      subroutine SOURCE ( nomore )

      use source_library

      implicit none

      logical lfirst
      save lfirst
      data lfirst / .true. /

      integer nomore

      integer particle_code, ionid

      logical energy_logical_flag
      double precision momentum_energy, particle_weight

      logical gaussian_divergence_logical_flag
      double precision divergence_x, divergence_y

      double precision coordinate_x, coordinate_y, coordinate_z

      integer direction_flag
      double precision direction_cosx, direction_cosy, direction_cosz

      double precision polarization_cosx, polarization_cosy
      double precision polarization_cosz

      double precision particle_age
      double precision kshort_component
      double precision delayed_radioactive_decay

*     Function declarations

      double precision sample_flat_momentum_energy
      double precision sample_gaussian_momentum_energy
      double precision sample_maxwell_boltzmann_energy
      double precision sample_energy_from_histogram

      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0
-:---  source_layer.f   11% (71,0)     (Fortran)
```

```
      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0

      if ( lfirst ) then
         call initialization( lfirst )
      end if

*     Beginning of customizable code
*     ==============================

*     Accessing variables from the SOURCE card
*     ----------------------------------------
*     Values set on the SOURCE card can be accessed with the following
*     variables:
*        Numerical values (double precision):
*           WHASOU(1), WHATSOU(2), .., WHASOU(18)
*        SDUM text (8 character):
*           SDUSOU

*     Particle type
*     -------------
*     particle_code:
*        FLUKA particle code of the primary
*        See section 5.1 of the FLUKA manual for the list
*           of particle code
*     Default:
*        IJBEAM: Particle code of the primary defined on the
*           BEAM card

      particle_code = IJBEAM

*     Partice momentum / energy and weight
*     ----------------------------------
*     momentum_energy:
*        Set the momentum or the kinetic energy of the primary
*     energy_logical_flag:
*        Possible values:
*           .false. : The momentum_energy variable contains the momentum
*              of the particle
*           .true.  : The momentum_energy variable contains the kinetic
*              energy of the particle
*     particle weight:
*        Sets the initial weight of the primary
*     Default:
*        [illegible]
*           energy was used on the BEAM card it)

      momentum_energy = PBEAM
      energy_logical_flag = .false.

      particle_weight = ONEONE

*        Implemented samplings functions
-:---  source_layer.f   27% (122,0)     (Fortran)
```

```
*        Implemented samplings functions
*        ---
*        momentum_energy = ...
*        ... = sample_flat_momentum_energy( [min], [max] )
*        ... = sample_gaussian_momentum_energy( [mean], [fwhm] )
*        ... = sample_maxwell_boltzmann_energy ( [temperature] )
*           temperature is given in GeV
*        ... = sample_energy_from_histogram( [filename], [energy_unit])
*           possible [energy_unit]s: "TeV", "GeV", "MeV", "keV', "eV"

*        Implemented sampling subroutines
*        ---
*        sample_exponential_energy_importance:
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - momentum_energy, particle_weight
*           call sample_exponential_energy_importance( [e_min], [e_max],
*    &         [intensity_ratio], momentum_energy, particle_weight)

*     Beam angular divergence
*     -----------------------
*     divergence_x:
*        Beam angular divergence in the X-Z plane
*        (before applying beam diretion)
*     divergence_y:
*        Beam angular divergence in the Y-Z plane
*     gaussian_divergence_logical_flag:
*        If .true., divergence values are taken as fwhm of a Gaussian
*           distribution
*        If .false., divergence values are taken as full angle of a flat
*           distribution
*     Note:
*        If one of the divergences is equal to zero, while the other is
*        not then the sampling will fail and crash. If divergence in
*        only one axis is desired, set the other to an infinitesimal
*        one, like 1.0D-12.
*     Defaults:
*        DIVBM: Divergence set on the BEAM card [in rad]
*        LDVGSS: Flag indicating Gaussian divergence, set on the BEAM
*           card

      divergence_x = DIVBM
      divergence_y = DIVBM
      gaussian_divergence_logical_flag = LDVGSS

*     Beam starting position
*     ----------------------
*     coordinate_x:
*        The X coordinate of the beam's starting position
*     coordinate_y:
*        The Y coordinate of the beam's starting position
*     coordinate_z:
*        The Z coordinate of the beam's starting position
-:---  source_layer.f   43% (175,0)     (Fortran)
```

Source routine – particle momentum/energy

- Some predefined routines (4 functions and 1 subroutine) are already available

Flat spectrum

Gaussian spectrum

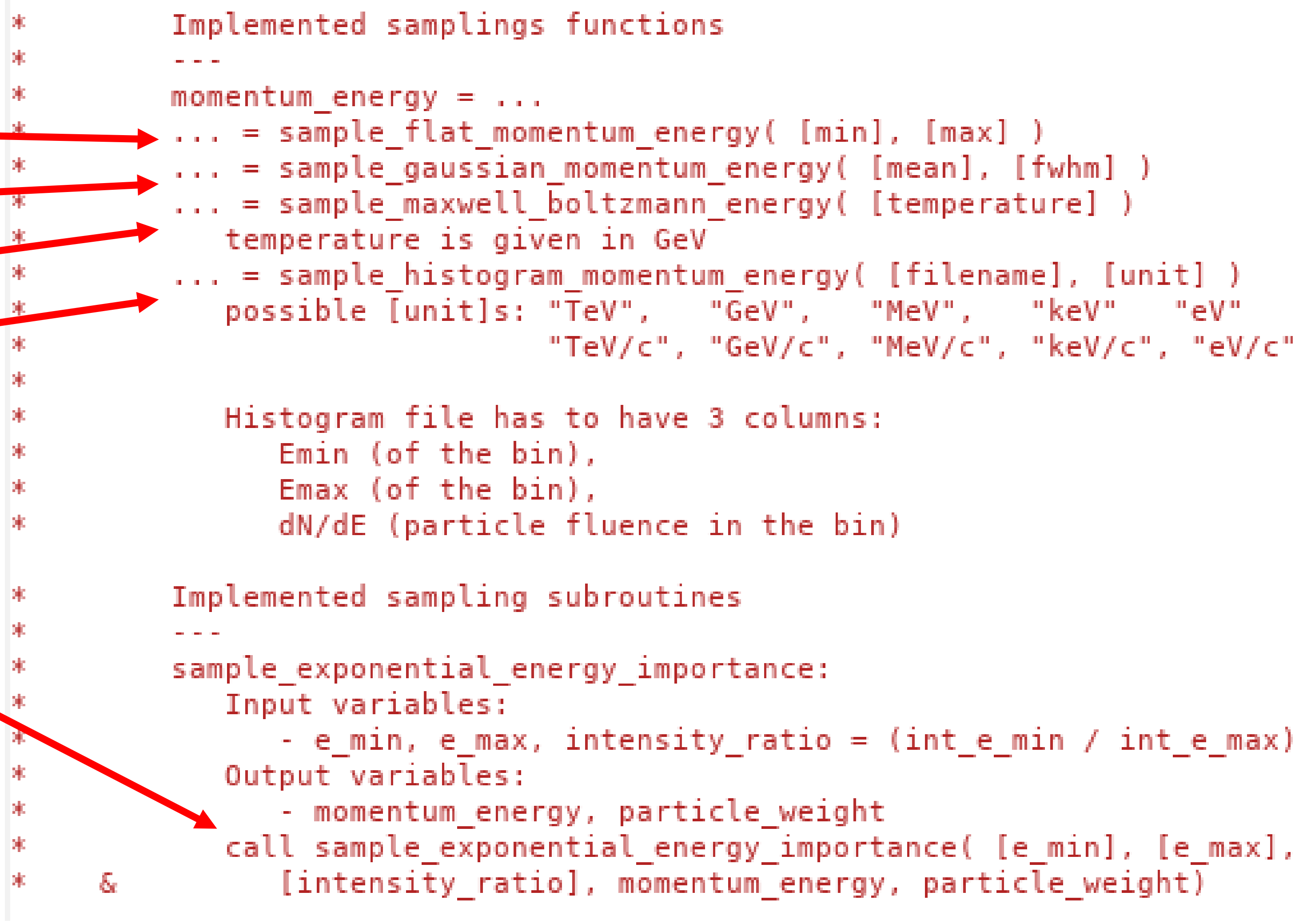
Maxwell-Boltzmann spectrum

Spectrum from histogram

Exponential spectrum

(via the change of

particle's weight)

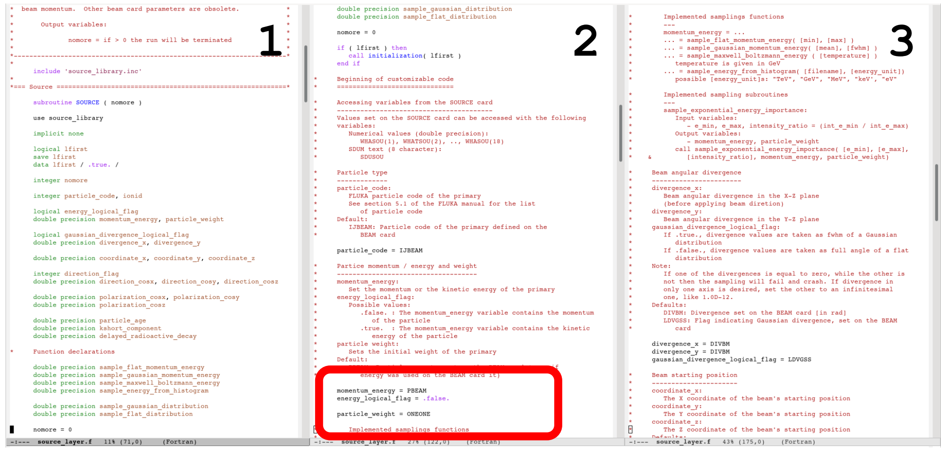


```
*        Implemented samplings functions
*        ---
*        momentum_energy = ...
*        ... = sample_flat_momentum_energy( [min], [max] )
*        ... = sample_gaussian_momentum_energy( [mean], [fwhm] )
*        ... = sample_maxwell_boltzmann_energy( [temperature] )
*           temperature is given in GeV
*        ... = sample_histogram_momentum_energy( [filename], [unit] )
*           possible [unit]s: "TeV",   "GeV",   "MeV",   "keV"   "eV"
*                             "TeV/c", "GeV/c", "MeV/c", "keV/c", "eV/c"
*
*           Histogram file has to have 3 columns:
*              Emin (of the bin),
*              Emax (of the bin),
*              dN/dE (particle fluence in the bin)

*        Implemented sampling subroutines
*        ---
*        sample_exponential_energy_importance:
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - momentum_energy, particle_weight
*           call sample_exponential_energy_importance( [e_min], [e_max],
*    &         [intensity_ratio], momentum_energy, particle_weight)
```

Source routine – particle weight

```
particle_weight = ONEONE
```

- Allows to set the weight of the primary particles
- 99% of the times weight=1 is ok
- Can be changed to distort the distribution of primaries (e.g. exponential distribution)
- Can be useful if dealing with more than one single type of primaries
- Not for a beginners’ use, mentioned here for completeness



```
*  beam momentum.  Other beam card parameters are obsolete.          *
*                                                                    *
*       Output variables:                                            *
*                                                                    *
*              nomore = if > 0 the run will be terminated            *
*                                                                    *
*--------------------------------------------------------------------*
*
      include 'source_library.inc'
*
*=== Source =========================================================*

      subroutine SOURCE ( nomore )

      use source_library

      implicit none

      logical lfirst
      save lfirst
      data lfirst / .true. /

      integer nomore

      integer particle_code, ionid

      logical energy_logical_flag
      double precision momentum_energy, particle_weight

      logical gaussian_divergence_logical_flag
      double precision divergence_x, divergence_y

      double precision coordinate_x, coordinate_y, coordinate_z

      integer direction_flag
      double precision direction_cosx, direction_cosy, direction_cosz

      double precision polarization_cosx, polarization_cosy
      double precision polarization_cosz

      double precision particle_age
      double precision kshort_component
      double precision delayed_radioactive_decay

*     Function declarations

      double precision sample_flat_momentum_energy
      double precision sample_gaussian_momentum_energy
      double precision sample_maxwell_boltzmann_energy
      double precision sample_energy_from_histogram

      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0
-:---  source_layer.f   11% (71,0)     (Fortran)
```

```
      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0

      if ( lfirst ) then
         call initialization( lfirst )
      end if

*     Beginning of customizable code
*     ==============================

*     Accessing variables from the SOURCE card
*     ----------------------------------------
*     Values set on the SOURCE card can be accessed with the following
*     variables:
*        Numerical values (double precision):
*           WHASOU(1), WHATSOU(2), .., WHASOU(18)
*        SDUM text (8 character):
*           SDUSOU

*     Particle type
*     -------------
*     particle_code:
*        FLUKA particle code of the primary
*        See section 5.1 of the FLUKA manual for the list
*           of particle code
*     Default:
*        IJBEAM: Particle code of the primary defined on the
*           BEAM card

      particle_code = IJBEAM

*     Partice momentum / energy and weight
*     ----------------------------------
*     momentum_energy:
*        Set the momentum or the kinetic energy of the primary
*     energy_logical_flag:
*        Possible values:
*           .false. : The momentum_energy variable contains the momentum
*              of the particle
*           .true.  : The momentum_energy variable contains the kinetic
*              energy of the particle
*     particle weight:
*        Sets the initial weight of the primary
*     Default:
*        [illegible]
*           energy was used on the BEAM card it)

      momentum_energy = PBEAM
      energy_logical_flag = .false.

      particle_weight = ONEONE

*        Implemented samplings functions
-:---  source_layer.f   27% (122,0)     (Fortran)
```

```
*         Implemented samplings functions
*         ---
*         momentum_energy = ...
*         ... = sample_flat_momentum_energy( [min], [max] )
*         ... = sample_gaussian_momentum_energy( [mean], [fwhm] )
*         ... = sample_maxwell_boltzmann_energy ( [temperature] )
*            temperature is given in GeV
*         ... = sample_energy_from_histogram( [filename], [energy_unit])
*            possible [energy_unit]s: "TeV", "GeV", "MeV", "keV', "eV"

*         Implemented sampling subroutines
*         ---
*         sample_exponential_energy_importance:
*            Input variables:
*               - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*            Output variables:
*               - momentum_energy, particle_weight
*            call sample_exponential_energy_importance( [e_min], [e_max],
*    &          [intensity_ratio], momentum_energy, particle_weight)

*     Beam angular divergence
*     -----------------------
*     divergence_x:
*        Beam angular divergence in the X-Z plane
*        (before applying beam diretion)
*     divergence_y:
*        Beam angular divergence in the Y-Z plane
*     gaussian_divergence_logical_flag:
*        If .true., divergence values are taken as fwhm of a Gaussian
*           distribution
*        If .false., divergence values are taken as full angle of a flat
*           distribution
*     Note:
*        If one of the divergences is equal to zero, while the other is
*        not then the sampling will fail and crash. If divergence in
*        only one axis is desired, set the other to an infinitesimal
*        one, like 1.0D-12.
*     Defaults:
*        DIVBM: Divergence set on the BEAM card [in rad]
*        LDVGSS: Flag indicating Gaussian divergence, set on the BEAM
*           card

      divergence_x = DIVBM
      divergence_y = DIVBM
      gaussian_divergence_logical_flag = LDVGSS

*     Beam starting position
*     ----------------------
*     coordinate_x:
*        The X coordinate of the beam's starting position
*     coordinate_y:
*        The Y coordinate of the beam's starting position
*     coordinate_z:
*        The Z coordinate of the beam's starting position
*     Defaults:
-:---  source_layer.f   43% (175,0)     (Fortran)
```

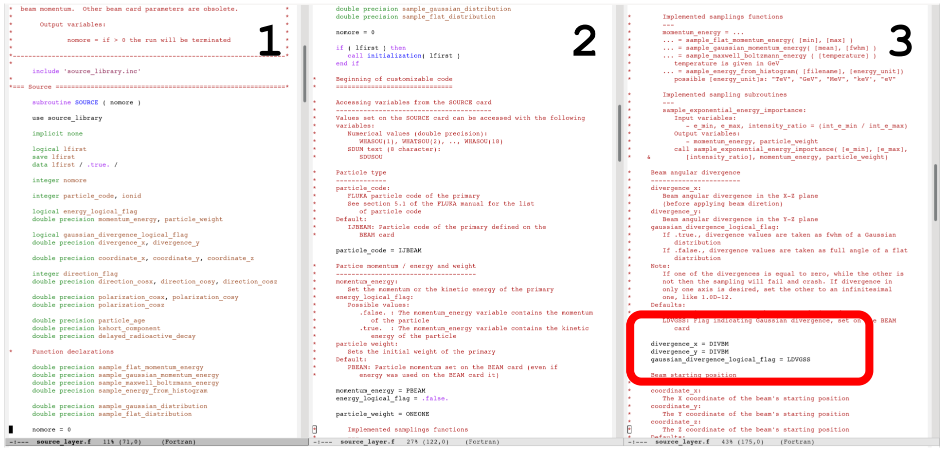
Source routine – beam divergence

```
divergence_x = DIVBM
divergence_y = DIVBM
gaussian_divergence_logical_flag = LDVGSS
```

$$\xleftarrow{\text{X-Z plane}}$$

$$\xleftarrow{\text{Y-Z plane}}$$

- By default, values are taken from the **BEAM** card
- Divergence values are taken
  - As Gaussian FWHM, if flag set `.true.`
  - As flat distribution full angle, if flag set `.false.`



1

```
*  beam momentum.  Other beam card parameters are obsolete.           *
*                                                                     *
*       Output variables:                                             *
*                                                                     *
*              nomore = if > 0 the run will be terminated             *
*                                                                     *
*---------------------------------------------------------------------*
*
      include 'source_library.inc'
*
*=== Source ==========================================================*

      subroutine SOURCE ( nomore )

      use source_library

      implicit none

      logical lfirst
      save lfirst
      data lfirst / .true. /

      integer nomore

      integer particle_code, ionid

      logical energy_logical_flag
      double precision momentum_energy, particle_weight

      logical gaussian_divergence_logical_flag
      double precision divergence_x, divergence_y

      double precision coordinate_x, coordinate_y, coordinate_z

      integer direction_flag
      double precision direction_cosx, direction_cosy, direction_cosz

      double precision polarization_cosx, polarization_cosy
      double precision polarization_cosz

      double precision particle_age
      double precision kshort_component
      double precision delayed_radioactive_decay

*     Function declarations

      double precision sample_flat_momentum_energy
      double precision sample_gaussian_momentum_energy
      double precision sample_maxwell_boltzmann_energy
      double precision sample_energy_from_histogram

      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0
-:---  source_layer.f   11% (71,0)     (Fortran)
```

2

```
      double precision sample_gaussian_distribution
      double precision sample_flat_distribution

      nomore = 0

      if ( lfirst ) then
         call initialization( lfirst )
      end if

*     Beginning of customizable code
*     ==============================

*     Accessing variables from the SOURCE card
*     ----------------------------------------
*     Values set on the SOURCE card can be accessed with the following
*     variables:
*        Numerical values (double precision):
*           WHASOU(1), WHATSOU(2), .., WHASOU(18)
*        SDUM text (8 character):
*           SDUSOU

*     Particle type
*     -------------
*     particle_code:
*        FLUKA particle code of the primary
*        See section 5.1 of the FLUKA manual for the list
*           of particle code
*     Default:
*        IJBEAM: Particle code of the primary defined on the
*           BEAM card

      particle_code = IJBEAM

*     Partice momentum / energy and weight
*     ----------------------------------
*     momentum_energy:
*        Set the momentum or the kinetic energy of the primary
*     energy_logical_flag:
*        Possible values:
*           .false. : The momentum_energy variable contains the momentum
*              of the particle
*           .true.  : The momentum_energy variable contains the kinetic
*              energy of the particle
*     particle weight:
*        Sets the initial weight of the primary
*     Default:
*        PBEAM: Particle momentum set on the BEAM card (even if
*           energy was used on the BEAM card it)

      momentum_energy = PBEAM
      energy_logical_flag = .false.

      particle_weight = ONEONE

*        Implemented samplings functions
-:---  source_layer.f   27% (122,0)     (Fortran)
```

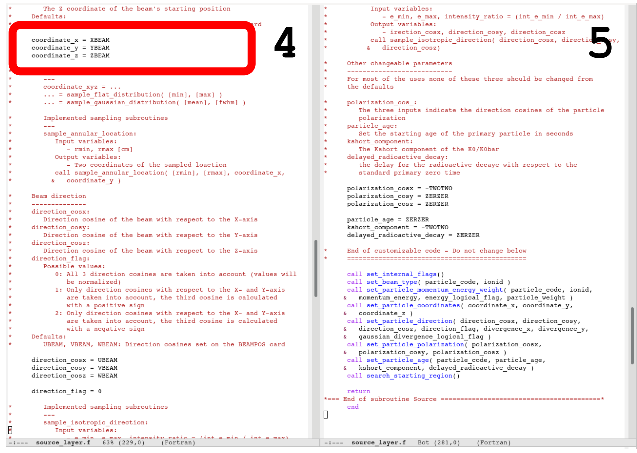
3

```
*        Implemented samplings functions
*        ---
*        momentum_energy = ...
*        ... = sample_flat_momentum_energy( [min], [max] )
*        ... = sample_gaussian_momentum_energy( [mean], [fwhm] )
*        ... = sample_maxwell_boltzmann_energy ( [temperature] )
*           temperature is given in GeV
*        ... = sample_energy_from_histogram( [filename], [energy_unit])
*           possible [energy_unit]s: "TeV", "GeV", "MeV", "keV', "eV"

*        Implemented sampling subroutines
*        ---
*        sample_exponential_energy_importance:
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - momentum_energy, particle_weight
*           call sample_exponential_energy_importance( [e_min], [e_max],
*    &         [intensity_ratio], momentum_energy, particle_weight)

*     Beam angular divergence
*     -----------------------
*     divergence_x:
*        Beam angular divergence in the X-Z plane
*        (before applying beam diretion)
*     divergence_y:
*        Beam angular divergence in the Y-Z plane
*     gaussian_divergence_logical_flag:
*        If .true., divergence values are taken as fwhm of a Gaussian
*           distribution
*        If .false., divergence values are taken as full angle of a flat
*           distribution
*     Note:
*        If one of the divergences is equal to zero, while the other is
*        not then the sampling will fail and crash. If divergence in
*        only one axis is desired, set the other to an infinitesimal
*        one, like 1.0D-12.
*     Defaults:
*
*        LDVGSS: Flag indicating Gaussian divergence, set on the BEAM
*           card

      divergence_x = DIVBM
      divergence_y = DIVBM
      gaussian_divergence_logical_flag = LDVGSS

*     Beam starting position
*
*     coordinate_x:
*        The X coordinate of the beam's starting position
*     coordinate_y:
*        The Y coordinate of the beam's starting position
*     coordinate_z:
*        The Z coordinate of the beam's starting position
-:---  source_layer.f   43% (175,0)     (Fortran)
```

Source routine – beam starting position

```
coordinate_x = XBEAM
coordinate_y = YBEAM
coordinate_z = ZBEAM
```

- By default, values are taken from the **BEAMPOS** card
- Extended sources can be defined using different starting positions



4

```
*        The Z coordinate of the beam's starting position
*     Defaults:

      coordinate_x = XBEAM
      coordinate_y = YBEAM
      coordinate_z = ZBEAM

*        ---
*        coordinate_xyz = ...
*        ... = sample_flat_distribution( [min], [max] )
*        ... = sample_gaussian_distribution( [mean], [fwhm] )

*        Implemented sampling subroutines
*        ---
*        sample_annular_location:
*           Input variables:
*              - rmin, rmax [cm]
*           Output variables:
*              - Two coordinates of the sampled loaction
*           call sample_annular_location( [rmin], [rmax], coordinate_x,
*          &   coordinate_y )

*     Beam direction
*     --------------
*     direction_cosx:
*        Direction cosine of the beam with respect to the X-axis
*     direction_cosy:
*        Direction cosine of the beam with respect to the Y-axis
*     direction_cosz:
*        Direction cosine of the beam with respect to the Z-axis
*     direction_flag:
*        Possible values:
*           0: All 3 direction cosines are taken into account (values will
*              be normalized)
*           1: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a positive sign
*           2: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a negative sign
*     Defaults:
*        UBEAM, VBEAM, WBEAM: Direction cosines set on the BEAMPOS card

      direction_cosx = UBEAM
      direction_cosy = VBEAM
      direction_cosz = WBEAM

      direction_flag = 0

*        Implemented sampling subroutines
*        ---
*        sample_isotropic_direction:
*           Input variables:
-:---  source_layer.f   63% (229,0)    (Fortran)
```

5

```
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - irection_cosx, direction_cosy, direction_cosz
*           call sample_isotropic_direction( direction_cosx, direction_cosy,
*          &   direction_cosz)

*     Other changeable parameters
*     ---------------------------
*     For most of the uses none of these three should be changed from
*     the defaults

*     polarization_cos_:
*        The three inputs indicate the direction cosines of the particle
*        polarization
*     particle_age:
*        Set the starting age of the primary particle in seconds
*     kshort_component:
*        The Kshort component of the K0/K0bar
*     delayed_radioactive_decay:
*        the delay for the radioactive decaye with respect to the
*        standard primary zero time

      polarization_cosx = -TWOTWO
      polarization_cosy = ZERZER
      polarization_cosz = ZERZER

      particle_age = ZERZER
      kshort_component = -TWOTWO
      delayed_radioactive_decay = ZERZER

*     End of customizable code - Do not change below
*     ==============================================

      call set_internal_flags()
      call set_beam_type( particle_code, ionid )
      call set_particle_momentum_energy_weight( particle_code, ionid,
     &   momentum_energy, energy_logical_flag, particle_weight )
      call set_particle_coordinates( coordinate_x, coordinate_y,
     &   coordinate_z )
      call set_particle_direction( direction_cosx, direction_cosy,
     &   direction_cosz, direction_flag, divergence_x, divergence_y,
     &   gaussian_divergence_logical_flag )
      call set_particle_polarization( polarization_cosx,
     &   polarization_cosy, polarization_cosz )
      call set_particle_age( particle_code, particle_age,
     &   kshort_component, delayed_radioactive_decay )
      call search_starting_region()

      return
*=== End of subroutine Source ==========================================*
      end

-:---  source_layer.f   Bot (281,0)    (Fortran)
```

Source routine – beam starting position

- Some predefined routines (2 functions and 1 subroutine) are already available

```
Implemented samplings functions
---
coordinate_xyz = ...
... = sample_flat_distribution( [min], [max] )
... = sample_gaussian_distribution( [mean], [fwhm] )

Implemented sampling subroutines
---
sample_annular_distribution:
   Input variables:
      - rmin, rmax [cm]
      - Coordinates of the center of the annular distribution
   Output variables:
      - Modified coordinates of the sampled loaction
           (input values have been overwritten)
   call sample_annular_distribution( [rmin], [rmax],
  &   coordinate_x, coordinate_y )
```

Flat/Gaussian spatial distribution

for the chosen coordinate

(x,y) coordinates of an annular distribution

centered on the provided location

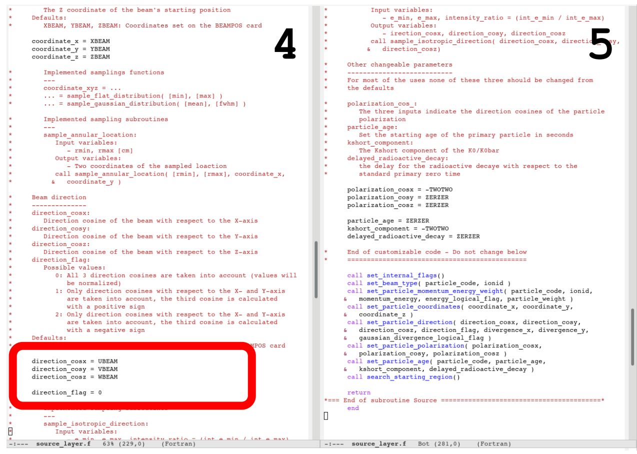
Source routine – beam direction

```
direction_cosx = UBEAM
direction_cosy = VBEAM
direction_cosz = WBEAM
direction_flag = 0
```

- By default, values are taken from the **`BEAM`** card
- If **`direction_flag = 0`** :

  all 3 director cosines are considered

  (normalization is performed in a subroutine)


4

```
*        The Z coordinate of the beam's starting position
*     Defaults:
*        XBEAM, YBEAM, ZBEAM: Coordinates set on the BEAMPOS card

      coordinate_x = XBEAM
      coordinate_y = YBEAM
      coordinate_z = ZBEAM

*        Implemented samplings functions
*        ---
*        coordinate_xyz = ...
*        ... = sample_flat_distribution( [min], [max] )
*        ... = sample_gaussian_distribution( [mean], [fwhm] )

*        Implemented sampling subroutines
*        ---
*        sample_annular_location:
*           Input variables:
*              - rmin, rmax [cm]
*           Output variables:
*              - Two coordinates of the sampled loaction
*           call sample_annular_location( [rmin], [rmax], coordinate_x,
*          &   coordinate_y )

*     Beam direction
*     --------------
*     direction_cosx:
*        Direction cosine of the beam with respect to the X-axis
*     direction_cosy:
*        Direction cosine of the beam with respect to the Y-axis
*     direction_cosz:
*        Direction cosine of the beam with respect to the Z-axis
*     direction_flag:
*        Possible values:
*           0: All 3 direction cosines are taken into account (values will
*              be normalized)
*           1: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a positive sign
*           2: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a negative sign
*     Defaults:
*                                                              MPOS card

      direction_cosx = UBEAM
      direction_cosy = VBEAM
      direction_cosz = WBEAM

      direction_flag = 0

*
*        ---
*        sample_isotropic_direction:
*           Input variables:
-:---  source_layer.f   63% (229,0)    (Fortran)
```

5

```
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - irection_cosx, direction_cosy, direction_cosz
*           call sample_isotropic_direction( direction_cosx, directio  sy,
*          &   direction_cosz)

*     Other changeable parameters
*     -------------------------
*     For most of the uses none of these three should be changed from
*     the defaults

*     polarization_cos_:
*        The three inputs indicate the direction cosines of the particle
*        polarization
*     particle_age:
*        Set the starting age of the primary particle in seconds
*     kshort_component:
*        The Kshort component of the K0/K0bar
*     delayed_radioactive_decay:
*        the delay for the radioactive decaye with respect to the
*        standard primary zero time

      polarization_cosx = -TWOTWO
      polarization_cosy = ZERZER
      polarization_cosz = ZERZER

      particle_age = ZERZER
      kshort_component = -TWOTWO
      delayed_radioactive_decay = ZERZER

*     End of customizable code - Do not change below
*     ==============================================

      call set_internal_flags()
      call set_beam_type( particle_code, ionid )
      call set_particle_momentum_energy_weight( particle_code, ionid,
     &   momentum_energy, energy_logical_flag, particle_weight )
      call set_particle_coordinates( coordinate_x, coordinate_y,
     &   coordinate_z )
      call set_particle_direction( direction_cosx, direction_cosy,
     &   direction_cosz, direction_flag, divergence_x, divergence_y,
     &   gaussian_divergence_logical_flag )
      call set_particle_polarization( polarization_cosx,
     &   polarization_cosy, polarization_cosz )
      call set_particle_age( particle_code, particle_age,
     &   kshort_component, delayed_radioactive_decay )
      call search_starting_region()

      return
*=== End of subroutine Source ==========================================*
      end

-:---  source_layer.f   Bot (281,0)    (Fortran)
```

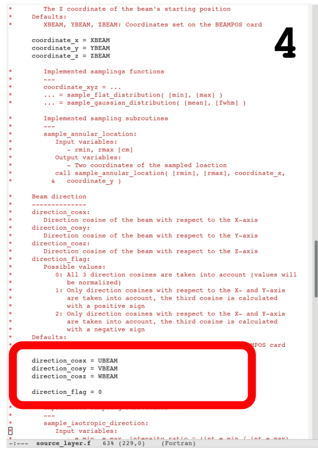
Source routine – beam direction

```
direction_cosx = UBEAM
direction_cosy = VBEAM
direction_cosz = WBEAM
direction_flag = 0
```

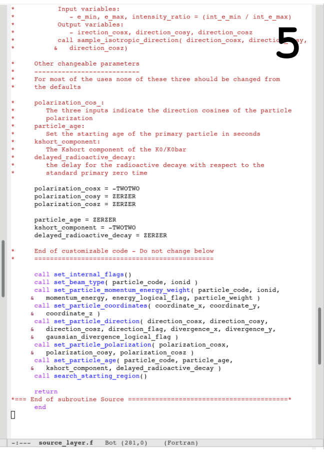
- By default, values are taken from the **`BEAM`** card
- If **`direction_flag = 1`** :

  `direction_cosz` is calculated from the other 2

  and assumed positive



```
*        The Z coordinate of the beam's starting position
*     Defaults:
*        XBEAM, YBEAM, ZBEAM: Coordinates set on the BEAMPOS card

      coordinate_x = XBEAM
      coordinate_y = YBEAM
      coordinate_z = ZBEAM

*        Implemented samplings functions
*        ---
*        coordinate_xyz = ...
*        ... = sample_flat_distribution( [min], [max] )
*        ... = sample_gaussian_distribution( [mean], [fwhm] )

*        Implemented sampling subroutines
*        ---
*        sample_annular_location:
*           Input variables:
*              - rmin, rmax [cm]
*           Output variables:
*              - Two coordinates of the sampled loaction
*           call sample_annular_location( [rmin], [rmax], coordinate_x,
*          &   coordinate_y )

*     Beam direction
*     --------------
*     direction_cosx:
*        Direction cosine of the beam with respect to the X-axis
*     direction_cosy:
*        Direction cosine of the beam with respect to the Y-axis
*     direction_cosz:
*        Direction cosine of the beam with respect to the Z-axis
*     direction_flag:
*        Possible values:
*           0: All 3 direction cosines are taken into account (values will
*              be normalized)
*           1: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a positive sign
*           2: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a negative sign
*     Defaults:
*                                                           MPOS card

      direction_cosx = UBEAM
      direction_cosy = VBEAM
      direction_cosz = WBEAM

      direction_flag = 0

*
*        ---
*        sample_isotropic_direction:
*           Input variables:
-:---  source_layer.f   63% (229,0)    (Fortran)
```



```
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - irection_cosx, direction_cosy, direction_cosz
*           call sample_isotropic_direction( direction_cosx, direction_cosy,
*          &   direction_cosz)

*     Other changeable parameters
*     ---------------------------
*     For most of the uses none of these three should be changed from
*     the defaults

*     polarization_cos_:
*        The three inputs indicate the direction cosines of the particle
*        polarization
*     particle_age:
*        Set the starting age of the primary particle in seconds
*     kshort_component:
*        The Kshort component of the K0/K0bar
*     delayed_radioactive_decay:
*        the delay for the radioactive decaye with respect to the
*        standard primary zero time

      polarization_cosx = -TWOTWO
      polarization_cosy = ZERZER
      polarization_cosz = ZERZER

      particle_age = ZERZER
      kshort_component = -TWOTWO
      delayed_radioactive_decay = ZERZER

*     End of customizable code - Do not change below
*     ==============================================

      call set_internal_flags()
      call set_beam_type( particle_code, ionid )
      call set_particle_momentum_energy_weight( particle_code, ionid,
     &   momentum_energy, energy_logical_flag, particle_weight )
      call set_particle_coordinates( coordinate_x, coordinate_y,
     &   coordinate_z )
      call set_particle_direction( direction_cosx, direction_cosy,
     &   direction_cosz, direction_flag, divergence_x, divergence_y,
     &   gaussian_divergence_logical_flag )
      call set_particle_polarization( polarization_cosx,
     &   polarization_cosy, polarization_cosz )
      call set_particle_age( particle_code, particle_age,
     &   kshort_component, delayed_radioactive_decay )
      call search_starting_region()

      return
*=== End of subroutine Source ==========================================*
      end

-:---  source_layer.f   Bot (281,0)    (Fortran)
```

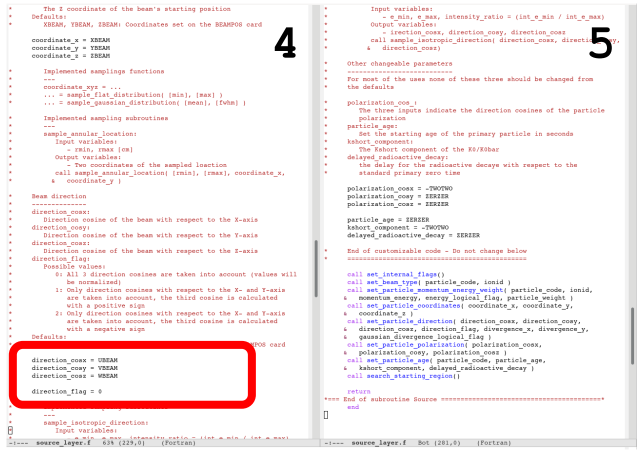
Source routine – beam direction

```
direction_cosx = UBEAM
direction_cosy = VBEAM
direction_cosz = WBEAM
direction_flag = 0
```

- By default, values are taken from the **`BEAM`** card
- If **`direction_flag = 2`** :

  `direction_cosz` is calculated from the other 2

  and assumed negative



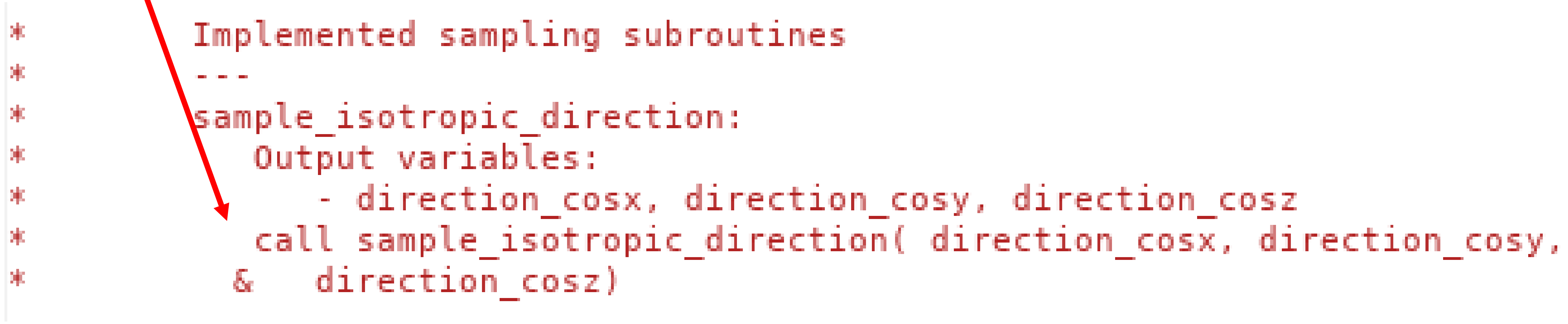
```
*          The Z coordinate of the beam's starting position
*       Defaults:
*          XBEAM, YBEAM, ZBEAM: Coordinates set on the BEAMPOS card

      coordinate_x = XBEAM
      coordinate_y = YBEAM
      coordinate_z = ZBEAM

*          Implemented samplings functions
*          ---
*          coordinate_xyz = ...
*          ... = sample_flat_distribution( [min], [max] )
*          ... = sample_gaussian_distribution( [mean], [fwhm] )

*          Implemented sampling subroutines
*          ---
*          sample_annular_location:
*             Input variables:
*                - rmin, rmax [cm]
*             Output variables:
*                - Two coordinates of the sampled loaction
*             call sample_annular_location( [rmin], [rmax], coordinate_x,
*            &   coordinate_y )

*       Beam direction
*       --------------
*       direction_cosx:
*          Direction cosine of the beam with respect to the X-axis
*       direction_cosy:
*          Direction cosine of the beam with respect to the Y-axis
*       direction_cosz:
*          Direction cosine of the beam with respect to the Z-axis
*       direction_flag:
*          Possible values:
*             0: All 3 direction cosines are taken into account (values will
*                be normalized)
*             1: Only direction cosines with respect to the X- and Y-axis
*                are taken into account, the third cosine is calculated
*                with a positive sign
*             2: Only direction cosines with respect to the X- and Y-axis
*                are taken into account, the third cosine is calculated
*                with a negative sign
*       Defaults:
*                                                        MPOS card

      direction_cosx = UBEAM
      direction_cosy = VBEAM
      direction_cosz = WBEAM

      direction_flag = 0

*          ---
*          sample_isotropic_direction:
*             Input variables:
-:---  source_layer.f   63% (229,0)    (Fortran)
```

```
*             Input variables:
*                - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*             Output variables:
*                - irection_cosx, direction_cosy, direction_cosz
*             call sample_isotropic_direction( direction_cosx, directio_  sy,
*            &   direction_cosz)

*       Other changeable parameters
*       ---------------------------
*       For most of the uses none of these three should be changed from
*       the defaults

*       polarization_cos_:
*          The three inputs indicate the direction cosines of the particle
*          polarization
*       particle_age:
*          Set the starting age of the primary particle in seconds
*       kshort_component:
*          The Kshort component of the K0/K0bar
*       delayed_radioactive_decay:
*          the delay for the radioactive decaye with respect to the
*          standard primary zero time

      polarization_cosx = -TWOTWO
      polarization_cosy = ZERZER
      polarization_cosz = ZERZER

      particle_age = ZERZER
      kshort_component = -TWOTWO
      delayed_radioactive_decay = ZERZER

*       End of customizable code - Do not change below
*       ==============================================

      call set_internal_flags()
      call set_beam_type( particle_code, ionid )
      call set_particle_momentum_energy_weight( particle_code, ionid,
     &   momentum_energy, energy_logical_flag, particle_weight )
      call set_particle_coordinates( coordinate_x, coordinate_y,
     &   coordinate_z )
      call set_particle_direction( direction_cosx, direction_cosy,
     &   direction_cosz, direction_flag, divergence_x, divergence_y,
     &   gaussian_divergence_logical_flag )
      call set_particle_polarization( polarization_cosx,
     &   polarization_cosy, polarization_cosz )
      call set_particle_age( particle_code, particle_age,
     &   kshort_component, delayed_radioactive_decay )
      call search_starting_region()

      return
*=== End of subroutine Source =========================================*
      end

-:---  source_layer.f   Bot (281,0)    (Fortran)
```

Source routine – beam direction

- A predefined subroutine is are already available

Isotropic direction

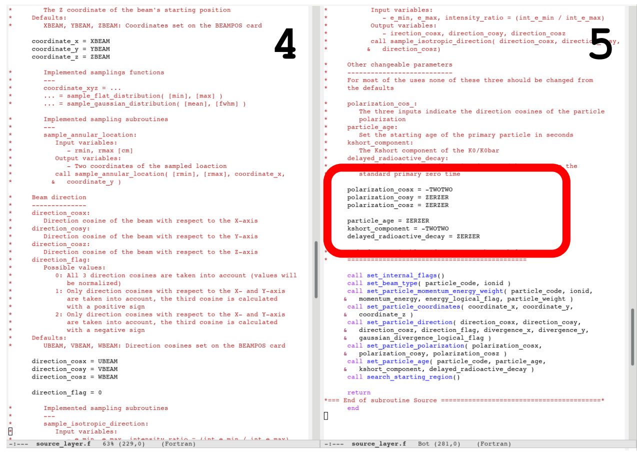


```
*         Implemented sampling subroutines
*         ---
*         sample_isotropic_direction:
*            Output variables:
*                - direction_cosx, direction_cosy, direction_cosz
*            call sample_isotropic_direction( direction_cosx, direction_cosy,
*           &    direction_cosz)
```

Source routine – other parameters

```
polarization_cosx = -TWOTWO
polarization_cosy = ZERZER
polarization_cosz = ZERZER
particle_age = ZERZER
kshort_component = -TWOTWO
delayed_radioactive_decay = ZERZER
```

- Variable names are pretty self-explanatory
- Not for beginners' use



4

```
*        The Z coordinate of the beam's starting position
*     Defaults:
*        XBEAM, YBEAM, ZBEAM: Coordinates set on the BEAMPOS card

      coordinate_x = XBEAM
      coordinate_y = YBEAM
      coordinate_z = ZBEAM

*        Implemented samplings functions
*        ---
*        coordinate_xyz = ...
*        ... = sample_flat_distribution( [min], [max] )
*        ... = sample_gaussian_distribution( [mean], [fwhm] )

*        Implemented sampling subroutines
*        ---
*        sample_annular_location:
*           Input variables:
*              - rmin, rmax [cm]
*           Output variables:
*              - Two coordinates of the sampled loaction
*           call sample_annular_location( [rmin], [rmax], coordinate_x,
*          &   coordinate_y )

*     Beam direction
*     --------------
*     direction_cosx:
*        Direction cosine of the beam with respect to the X-axis
*     direction_cosy:
*        Direction cosine of the beam with respect to the Y-axis
*     direction_cosz:
*        Direction cosine of the beam with respect to the Z-axis
*     direction_flag:
*        Possible values:
*           0: All 3 direction cosines are taken into account (values will
*              be normalized)
*           1: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a positive sign
*           2: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a negative sign
*     Defaults:
*        UBEAM, VBEAM, WBEAM: Direction cosines set on the BEAMPOS card

      direction_cosx = UBEAM
      direction_cosy = VBEAM
      direction_cosz = WBEAM

      direction_flag = 0

*        Implemented sampling subroutines
*        ---
*        sample_isotropic_direction:
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
-:---  source_layer.f   63% (229,0)    (Fortran)
```

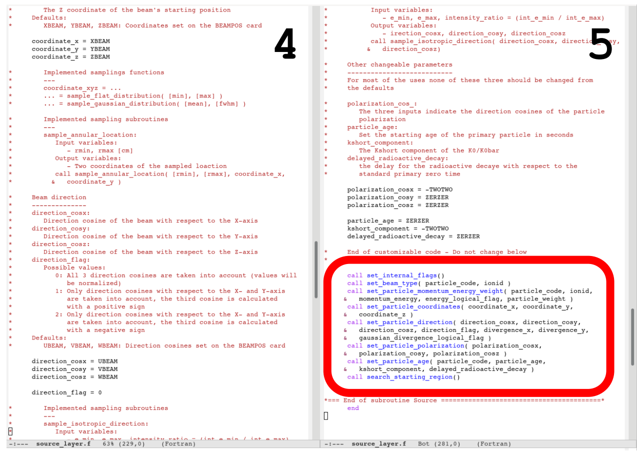
5

```
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - irection_cosx, direction_cosy, direction_cosz
*           call sample_isotropic_direction( direction_cosx, directio[illegible]sy,
*          &   direction_cosz)

*     Other changeable parameters
*     -------------------------
*     For most of the uses none of these three should be changed from
*     the defaults

*     polarization_cos_:
*        The three inputs indicate the direction cosines of the particle
*        polarization
*     particle_age:
*        Set the starting age of the primary particle in seconds
*     kshort_component:
*        The Kshort component of the K0/K0bar
*     delayed_radioactive_decay:
*        [illegible] the
*        standard primary zero time

      polarization_cosx = -TWOTWO
      polarization_cosy = ZERZER
      polarization_cosz = ZERZER

      particle_age = ZERZER
      kshort_component = -TWOTWO
      delayed_radioactive_decay = ZERZER

*     ==============================================

      call set_internal_flags()
      call set_beam_type( particle_code, ionid )
      call set_particle_momentum_energy_weight( particle_code, ionid,
     &   momentum_energy, energy_logical_flag, particle_weight )
      call set_particle_coordinates( coordinate_x, coordinate_y,
     &   coordinate_z )
      call set_particle_direction( direction_cosx, direction_cosy,
     &   direction_cosz, direction_flag, divergence_x, divergence_y,
     &   gaussian_divergence_logical_flag )
      call set_particle_polarization( polarization_cosx,
     &   polarization_cosy, polarization_cosz )
      call set_particle_age( particle_code, particle_age,
     &   kshort_component, delayed_radioactive_decay )
      call search_starting_region()

      return
*=== End of subroutine Source ==========================================*
      end

-:---  source_layer.f   Bot (281,0)    (Fortran)
```

Source routine – lines not to be touched

```
call set_internal_flags()
call set_beam_type(…)
call set_particle_momentum_energy_weight(…)
call set_particle_coordinates(…)
call set_particle_direction(…)
call set_particle_polarization(…)
call set_particle_age(…)
call search_starting_region()
```

- These calls pass the provided inputs to Fluka
- Not to be touched for any reason



4

```
*        The Z coordinate of the beam's starting position
*     Defaults:
*        XBEAM, YBEAM, ZBEAM: Coordinates set on the BEAMPOS card

      coordinate_x = XBEAM
      coordinate_y = YBEAM
      coordinate_z = ZBEAM

*        Implemented samplings functions
*        ---
*        coordinate_xyz = ...
*        ... = sample_flat_distribution( [min], [max] )
*        ... = sample_gaussian_distribution( [mean], [fwhm] )

*        Implemented sampling subroutines
*        ---
*        sample_annular_location:
*           Input variables:
*              - rmin, rmax [cm]
*           Output variables:
*              - Two coordinates of the sampled loaction
*           call sample_annular_location( [rmin], [rmax], coordinate_x,
*          &   coordinate_y )

*     Beam direction
*     --------------
*     direction_cosx:
*        Direction cosine of the beam with respect to the X-axis
*     direction_cosy:
*        Direction cosine of the beam with respect to the Y-axis
*     direction_cosz:
*        Direction cosine of the beam with respect to the Z-axis
*     direction_flag:
*        Possible values:
*           0: All 3 direction cosines are taken into account (values will
*              be normalized)
*           1: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a positive sign
*           2: Only direction cosines with respect to the X- and Y-axis
*              are taken into account, the third cosine is calculated
*              with a negative sign
*     Defaults:
*        UBEAM, VBEAM, WBEAM: Direction cosines set on the BEAMPOS card

      direction_cosx = UBEAM
      direction_cosy = VBEAM
      direction_cosz = WBEAM

      direction_flag = 0

*        Implemented sampling subroutines
*        ---
*        sample_isotropic_direction:
*           Input variables:
-:---  source_layer.f   63% (229,0)    (Fortran)
```

5

```
*           Input variables:
*              - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)
*           Output variables:
*              - irection_cosx, direction_cosy, direction_cosz
*           call sample_isotropic_direction( direction_cosx, directio_cosy,
*          &   direction_cosz)

*     Other changeable parameters
*     ---------------------------
*     For most of the uses none of these three should be changed from
*     the defaults

*     polarization_cos_:
*        The three inputs indicate the direction cosines of the particle
*        polarization
*     particle_age:
*        Set the starting age of the primary particle in seconds
*     kshort_component:
*        The Kshort component of the K0/K0bar
*     delayed_radioactive_decay:
*        the delay for the radioactive decaye with respect to the
*        standard primary zero time

      polarization_cosx = -TWOTWO
      polarization_cosy = ZERZER
      polarization_cosz = ZERZER

      particle_age = ZERZER
      kshort_component = -TWOTWO
      delayed_radioactive_decay = ZERZER

*     End of customizable code - Do not change below
*

      call set_internal_flags()
      call set_beam_type( particle_code, ionid )
      call set_particle_momentum_energy_weight( particle_code, ionid,
     &   momentum_energy, energy_logical_flag, particle_weight )
      call set_particle_coordinates( coordinate_x, coordinate_y,
     &   coordinate_z )
      call set_particle_direction( direction_cosx, direction_cosy,
     &   direction_cosz, direction_flag, divergence_x, divergence_y,
     &   gaussian_divergence_logical_flag )
      call set_particle_polarization( polarization_cosx,
     &   polarization_cosy, polarization_cosz )
      call set_particle_age( particle_code, particle_age,
     &   kshort_component, delayed_radioactive_decay )
      call search_starting_region()

*=== End of subroutine Source ==========================================*
      end

-:---  source_layer.f   Bot (281,0)    (Fortran)
```

Some predefined FLUKA random sampling routines

- Fluka offers some predefined routines for random sampling

- **`my_variable = FLRNDM(XDUMMY)`**

  Assigns a 64-bit random number in [0,1)

- **`call FLNRRN(gauss1)`**

  Returns a Gaussian distributed random number

- **`call FLNRR2(gauss1,gauss2)`**

  Returns two uncorrelated Gaussian distributed random numbers

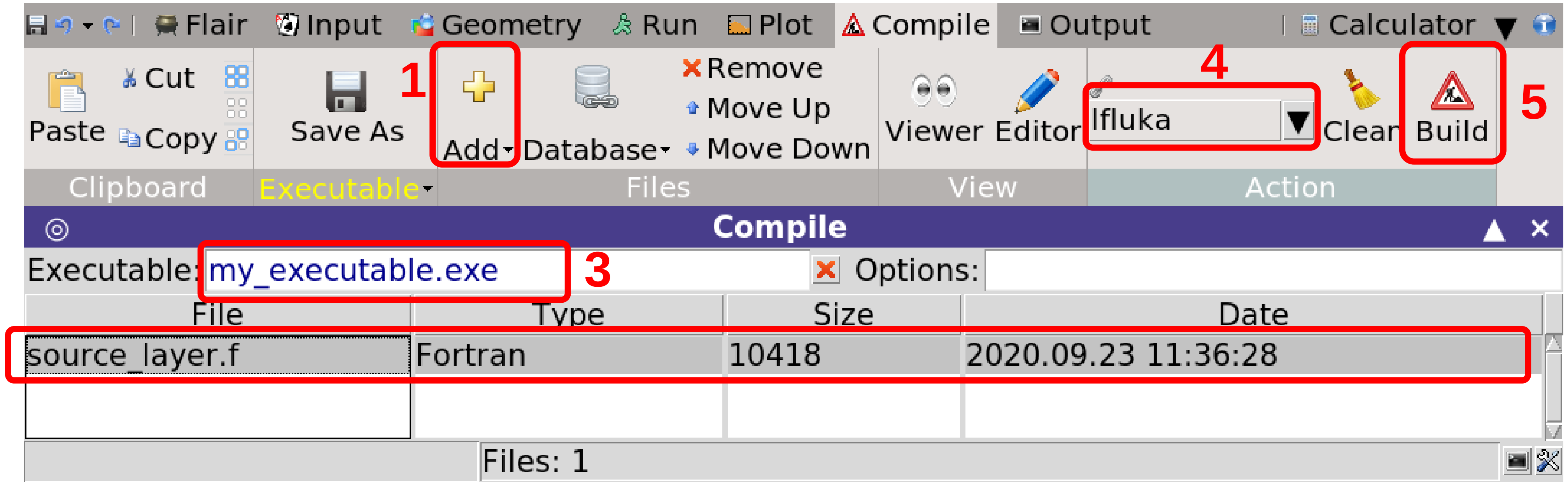
- **`call SFECFE(sint,cost)`**

  Returns sine and cosine of a random azimuthal angle

Compile

1. Add the routine
2. Verify that it appears
3. Insert the name of your executable
4. Select the compiler
5. Build your executable

- Warning: the library file (**`source_library.inc`**) must be in the same directory of the source file (**`source_layer.f`**)



Compile

Executable: my_executable.exe Options:

File	Type	Size	Date
source_layer.f	Fortran	10418	2020.09.23 11:36:28

lfluka

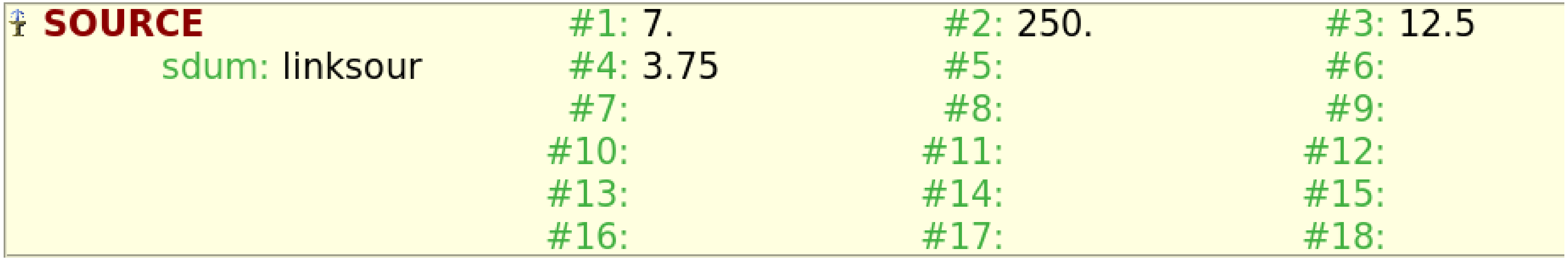
Files: 1

SOURCE card and passing parameters

- To invoke a source routine it is necessary to add a **SOURCE** card
- A **SOURCE** card can be empty or can be used to pass parameters to the routine
- Max. 18 numerical values (**WHASOU(ii)**) and 1 string can be passed (**SDUSOU**)
  SOURCE card and **BEAM** card can coexist
- Good practice advice:

  even if the beam energy/momentum is defined in the source routine,

  specify it in the **BEAM** card as it is used internally as default for some scoring



```
SOURCE                      #1: 7.              #2: 250.            #3: 12.5
        sdum: linksour      #4: 3.75            #5:                 #6:
                            #7:                 #8:                 #9:
                           #10:                #11:                #12:
                           #13:                #14:                #15:
                           #16:                #17:                #18:
```

Time to do some hands-on practice!

- We will now see together a small example of “new” source routine

FLUKA