



# Source routine

An introduction to the source routine

# Why do we need a source routine

- The source routine is used to define complex sources, when the options provided in the **BEAM**, **BEAMPOS** and **BEAMAXES** cards are not enough.
- Most common use cases:
  - Mixed field
  - Beam with an energy spectrum
  - Complex beam shape
  - Second step of a two-step simulation

# The “old” source routine

- Scary for beginners, limited documentation
- Written according to the FORTRAN 77 standard, encouraged errors

```
1 *
2 *==== source =====
3 *
4 SUBROUTINE SOURCE ( NOMORE )
5
6 INCLUDE 'dblprc.inc'
7 INCLUDE 'dimpar.inc'
8 INCLUDE 'iounit.inc'
9
10 -----
11 *
12 * Copyright (C) 2003-2019: CERN & INFN
13 * All Rights Reserved.
14 *
15 * New source for FLUKA9x-FLUKA20xy:
16 *
17 * Created on 07 January 1990 by Alfredo Ferrari & Paola Sala
18 *                               Infn - Milan
19 *
20 * This is just an example of a possible user written source routine.
21 * note that the beam card still has some meaning - in the scoring the
22 * maximum momentum used in deciding the binning is taken from the
23 * beam momentum. Other beam card parameters are obsolete.
24 *
25 * Output variables:
26 *
27 *     Nomore = if > 0 the run will be terminated
28 *
29 -----
30 *
31 INCLUDE 'beamcm.inc'
32 INCLUDE 'fheavy.inc'
33 INCLUDE 'fkstk.inc'
34 INCLUDE 'ioiocm.inc'
35 INCLUDE 'ltclcm.inc'
36 INCLUDE 'paprop.inc'
37 INCLUDE 'sourcm.inc'
38 INCLUDE 'sumcou.inc'
39 *
40 LOGICAL LFIRST, LISNUT
41 *
42 SAVE LFIRST
43 DATA LFIRST / .TRUE. /
44 * Statement function:
45 LISNUT (I) = INDEX ( PRNAME (I), 'NEUTRI' ) .GT. 0
46 -----
47 *
48 * BASIC VERSION
49 *
50 -----
51 *
52 * NOMORE = 0
53 *
54 * First call initializations:
55 * IF ( LFIRST ) THEN
56 * ** The following 3 cards are mandatory **
57 TKESUM = ZERZER
58 LFIRST = .FALSE.
59 LUSSRC = .TRUE.
60 * ** User initialization **
61 END IF
62 *
63 * Push one source particle to the stack. Note that you could as well
64 * push many but this way we reserve a maximum amount of space in the
65 * stack for the secondaries to be generated
66 * Npflka is the stack counter: of course any time source is called it
67 * must be = 0
68 *
69 * NPFPLKA = NPFPLKA + 1
70 * Wt is the weight of the particle
71 * WTFLK (NPFPLKA) = ONEONE
72 * WEIPRI = WEIPRI + WTFLK (NPFPLKA)
73 * Particle type (=proton.....). Ijbeam is the type set by the BEAM
74 * card
75 *
76 * (Radioactive) isotope:
77 IF ( IJBEAM .EQ. -2 .AND. LRDBEA ) THEN
78 IARES = IPROA
79 IZRES = IPROZ
80 IISRES = IPRON
81 CALL STISBM ( IARES, IZRES, IISRES )
82 IJHION = IPRON * 100000 + MOD ( IPROZ, 100 ) * 1000 + IPROA
83 IJHION = IJHION * 100 + KXHEAV
84 IONID = IJHION
85 CALL DCION ( IONID )
86 CALL SETION ( IONID )
87 LFRPHN (NPFPLKA) = .FALSE.
88 *
89 * Heavy ion:
90 ELSE IF ( IJBEAM .EQ. -2 ) THEN
91 IJHION = IPRON * 100000 + MOD ( IPROZ, 100 ) * 1000 + IPROA
92 IJHION = IJHION * 100 + KXHEAV
93 IONID = IJHION
94 CALL DCION ( IONID )
95 CALL SETION ( IONID )
96 ILOFLK (NPFPLKA) = IJHION
97 * Flag this is prompt radiation
98 LRADDC (NPFPLKA) = .FALSE.
99 * Group number for "low" energy neutrons, set to 0 anyway
100 IGROUP (NPFPLKA) = 0
101 * Parent radioactive isotope:
102 IRDAZM (NPFPLKA) = 0
103 * Particle age (s)
104 AGESTK (NPFPLKA) = +ZERZER
105 * Kinetic energy of the particle (GeV)
106 TKEFLK (NPFPLKA) = SQRT ( PBEAM**2 + AM (IONID)**2 )
107 *
108 * Particle momentum
109 PMOFLK (NPFPLKA) = PBEAM
110 *
111 *
112 *
113 *
114 *
115 * Normal hadron:
116 ELSE
117 IONID = IJBEAM
118 ILOFLK (NPFPLKA) = IJBEAM
119 * Flag this is prompt radiation
120 LRADDC (NPFPLKA) = .FALSE.
121 * Group number for "low" energy neutrons, set to 0 anyway
122 IGROUP (NPFPLKA) = 0
123 * Parent radioactive isotope:
124 IRDAZM (NPFPLKA) = 0
125 * Particle age (s)
126
127 *
128 * Kinetic energy of the particle (GeV)
129 TKEFLK (NPFPLKA) = SQRT ( PBEAM**2 + AM (IONID)**2 )
130 *
131 * Particle momentum
132 PMOFLK (NPFPLKA) = PBEAM
133 *
134 *
135 *
136 *
137 * Check if it is a neutrino, if so force the interaction
138 * (unless the relevant flag has been disabled)
139 IF ( LISNUT (IJBEAM) .AND. LNUFIN ) THEN
140 LFRPHN (NPFPLKA) = .TRUE.
141 *
142 *
143 *
144 *
145 *
146 *
147 *
148 *
149 *
150 * From this point .....
151 * Particle generation (1 for primaries)
152 LOFLK (NPFPLKA) = 1
153 * User dependent flag:
154 LOUSE (NPFPLKA) = 0
155 * No channeling:
156 KCHFLK (NPFPLKA) = 0
157 ECRFLK (NPFPLKA) = ZERZER
158 * Extra infos:
159 INFSTK (NPFPLKA) = 0
160 LNFSTK (NPFPLKA) = 0
161 ANFSTK (NPFPLKA) = ZERZER
162 * Parent variables:
163 IPRSTK (NPFPLKA) = 0
164 EKPSK (NPFPLKA) = ZERZER
165 * User dependent spare variables:
166 DO 100 ISPR = 1, MKBM1
167 SPAREK (ISPR,NPFPLKA) = ZERZER
168 100 CONTINUE
169 * User dependent spare flags:
170 DO 200 ISPR = 1, MKBM2
171 ISPAK (ISPR,NPFPLKA) = 0
172 200 CONTINUE
173 * Save the track number of the stack particle:
174 ISPAK (MKBM2,NPFPLKA) = NPFPLKA
175 NPARMA = NPARMA + 1
176 NUNPAR (NPFPLKA) = NPARMA
177 NEVENT (NPFPLKA) = 0
178 DFNEAR (NPFPLKA) = +ZERZER
179 * ... to this point: don't change anything
180 AKNSHR (NPFPLKA) = -TWOTWO
181 * Cosines (tx,ty,tz)
182 TXFLK (NPFPLKA) = UBEAM
183 TYFLK (NPFPLKA) = VBEAM
184 TZFLK (NPFPLKA) = WBEAM
185 *
186 *
187 *
188 * Polarization cosines:
189 TXPOL (NPFPLKA) = -TWOTWO
189
190 *
191 * Particle coordinates
192 XFLK (NPFPLKA) = XBEAM
193 YFLK (NPFPLKA) = YBEAM
194 ZFLK (NPFPLKA) = ZBEAM
195 * Calculate the total kinetic energy of the primaries: don't change
196 *
197 *
198 * (Radioactive) isotope:
199 IF ( IJBEAM .EQ. -2 .AND. LRDBEA ) THEN
200 *
201 *
202 * Heavy ion:
203 ELSE IF ( ILOFLK (NPFPLKA) .EQ. -2 .OR.
204 ILOFLK (NPFPLKA) .GT. 100000 ) THEN
205 *
206 *
207 *
208 *
209 *
210 *
211 *
212 *
213 *
214 *
215 *
216 *
217 *
218 *
219 *
220 * Here we ask for the region number of the hitting point.
221 * NREG (NPFPLKA) = ...
222 * The following line makes the starting region search much more
223 * robust if particles are starting very close to a boundary:
224 CALL GEORCS ( TXFLK (NPFPLKA), TYFLK (NPFPLKA), TZFLK (NPFPLKA) )
225 CALL GEOREG ( XFLK (NPFPLKA), YFLK (NPFPLKA), ZFLK (NPFPLKA),
226 * NREGFLK(NPFPLKA), IDISC )
227 * Do not change these cards:
228 CALL GEOSH ( NHSPNT (NPFPLKA), 1, -11, MLATTC )
229 NLATTC (NPFPLKA) = MLATTC
230 CMPATH (NPFPLKA) = ZERZER
231 CALL SOEVSV
232 RETURN
233 *==== End of subroutine Source =====
234 END
235
236
```

# The “new” source routine

- Simplified appearance
- Long & meaningful names for variables and routines
- Forced declaration of variables – Use of **implicit none**
- Documented by comments and in the manual
- 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**

# The “new” source routine

```
1 *
2 *****
3 * Copyright (C) 2000: CERN
4 * All rights reserved.
5 *
6 * Source routine or FLUKA 4i
7 *
8 * Created on 24 September 2020 by David Hervath & Roberto Versaci
9 * FLK quantities
10 *
11 * Modified on 17 November 2020 by David Hervath & Roberto Versaci
12 * ILLI Beamlines
13 *
14 * This is a simplified user-written source routine utilizing a
15 * separate source routine library.
16 *
17 * It is intended as an alternative user-friendly version of the
18 * source routine. Existing FLUKA + source routines remain
19 * compatible.
20 *
21 * Note that the beam card still has some meaning - in the scoring the
22 * maximum momentum used in deciding the binning is taken from the
23 * beam momentum. Other beam card parameters are obsolete.
24 *
25 * Output variables:
26 *
27 * none - if > 0 the run will be terminated
28 *
29 *
30 *
31 *
32 *
33 * Quick start guide:
34 *
35 * This user source routine template aims to modernize the legacy routine
36 * by implementing modern Fortran conventions and to provide built in
37 *
38 * sampling functions.
39 *
40 * The users only need to change / add code between the BEGINNING and END
41 * marks, one section for declaration of user variables, and one for
42 * assigning values to the beam parameters.
43 *
44 * By default there is no user variable defined, and all code lines for
45 * parameter assignment are commented out. These comments start with the
46 * symbol: " ! " To enable user, the " ! " needs to be deleted.
47 *
48 * (Note: In Fortran each code line should start in column 7 or further in.)
49 *
50 * Every beam parameter has a default value based on the FLUKA input file.
51 * A parameter assignment should only be used if the default value has to be
52 * modified.
53 *
54 * There are three ways to assign a value to a parameter:
55 *
56 * 1. Direct assignment: A parameter is equal to a value. For example:
57 *
58 * momentum_energy = 0.100
59 *
60 * If the parameter defined as a double precision, then the assigned
61 * value should be represented as double precision as well so as to
62 * not lose numerical precision. To do this a "D" exponential mark
63 * must be used.
64 *
65 * 2. Using a sampling function: A parameter is assigned to a value,
66 * which is calculated by a separate function. For example:
67 *
68 * coordinate_x = sample_flat_distribution([min], [max])
69 *
70 * The parameters between the '[' and ']' brackets need to be replaced
71 * with numbers, or user variables containing the desired values.
72 *
73 * 3. Using a sampling subroutine: They are similar to function, but they
74 * are not returning values directly, instead they modify the variables
75 * in their argument list. For example:
76 *
77 * call sample_angular_distribution([min], [max], coordinate_x, coordinate_y)
78 *
79 * The example above has two input parameters between brackets, and two
80 * output parameters (without brackets). The input parameters have to be
81 * provided, but the output parameters names usually don't
82 * need to be changed, but there are cases where a subset of possible
83 * output parameters has to be selected.
84 *
85 * For further details see the FLUKA manual.
86 *
87 *
88 *
89 *
90 *
91 *
92 *
93 *
94 *
95 *
96 *
97 *
98 *
99 *
100 *
101 *
102 *
103 *
104 *
105 *
106 *
107 *
108 *
109 *
110 *
111 *
112 *
113 *
114 *
115 *
116 *
117 *****
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

• Note: the snapshot is not meant to be read – Detailed view will follow

# User declaration

```
! =====  
! BEGINNING of user declared variables  
! =====
```

```
! =====  
! END of user declared variables  
! =====
```

- Dedicated space for the declaration of additional variables (and functions)

# User declaration (example)

```
! =====  
! BEGINNING of user declared variables  
! =====
```

```
integer :: counter  
double precision :: energy  
logical :: flag
```

```
! =====  
! END of user declared variables  
! =====
```

# Initialisation

- Initialisation of internal variables  
Runs every time, resetting their values to the defaults

```
call initialization()
```

```
if ( first_run ) then
```

```
! =====  
! BEGINNING of custom initialization  
! =====
```

```
! =====  
! END of custom initialization  
! =====
```

```
first_run = .false.
```

```
end if
```

- Custom initialisation block

Runs only the first time the source routine is used

# Main section

- For setting the internal variables directly, or using one of the sampling routines
- To enable a line, remove the `\*`  
(The command should start on the 7<sup>th</sup> column)
- The variables with `[` `]` brackets and ... are placeholders, they need to be replaced with values or user variables  
(Brackets should be deleted)
- Always use double precision format for floating point numbers (1.0d0)

```
! =====  
! BEGINNING of customizable code  
! =====  
  
* particle_code = ...  
  
* heavyion_atomic_number = ...  
* heavyion_mass_number = ...  
* heavyion_isomer = ...  
  
* momentum_energy = ...  
  
* energy_logical_flag = .true.  
  
* particle_weight = ...  
  
* momentum_energy = sample_flat_momentum_energy( [min], [max] )  
* momentum_energy = sample_gaussian_momentum_energy( [mean], [fwhm] )  
* momentum_energy = sample_maxwell_boltzmann_energy( [temperature] )  
* momentum_energy = sample_histogram_momentum_energy( [filename], [unit] )  
* momentum_energy = sample_spectrum_momentum_energy( [filename], [unit] )  
  
* call sample_exponential_energy_weight( [e_min], [e_max], [intensity_ratio], momentum_energy, particle_weight )  
  
* divergence_x = ...  
* divergence_y = ...  
  
* gaussian_divergence_logical_flag = .true.  
  
* coordinate_x = ...  
* coordinate_y = ...  
* coordinate_z = ...  
  
* coordinate_[a] = sample_flat_distribution( [min], [max] )  
* coordinate_[a] = sample_gaussian_distribution( [mean], [fwhm] )  
* call sample_annular_distribution( [rmin], [rmax], coordinate_[a], coordinate_[b] )  
  
* direction_cosx = ...  
* direction_cosy = ...  
* direction_cosz = ...  
  
* direction_flag = ...  
  
* call sample_isotropic_direction( direction_cosx, direction_cosy, direction_cosz )  
  
* polarization_cosx = ...  
* polarization_cosy = ...  
* polarization_cosz = ...  
  
* particle_age = ...  
* kshort_component = ...  
* delayed_radioactive_decay = ...  
  
* call read_phase_space_file( [filename], [energy_unit], [length_unit], phase_space_entry, [sequential_logical_flag], nomore )  
  
* particle_code = phase_space_entry%pc  
* momentum_energy = phase_space_entry%e  
  
* energy_logical_flag = .true.  
  
* coordinate_x = phase_space_entry%x  
* coordinate_y = phase_space_entry%y  
* coordinate_z = phase_space_entry%z  
  
* direction_cosx = phase_space_entry%u  
* direction_cosy = phase_space_entry%v  
* direction_cosz = phase_space_entry%w  
  
* particle_weight = phase_space_entry%wei  
  
* debug_logical_flag = .true.  
* debug_lines = ...
```

# Primary particle

```
particle_code = ...
```

- By default, the particle type given in the **BEAM** card is taken
- Particle codes are explained in FLUKA manual section 5.1
- Possible application: beam made of more than one type of particles

```
heavyion_atomic_number = ...  
heavyion_mass_number = ...  
heavyion_isomer = ...
```

- Only used if primary particle is set to HEAVYION or ISOTOPE on the **BEAM** card
- Default values are set on the **HI-PROPE** card, or for  $^{12}\text{C}$  if the card is missing

# Energy / momentum

```
momentum_energy = ...
```

- By default, the particle momentum is expected
- The default value is based on the **BEAM** card  
(Automatically converted into momentum if energy is given in the **BEAM** card)
- If energy is specified in the source routine, the following logical value must be set to **.true.**

```
energy_logical_flag = ...
```

# Energy / momentum

- The momentum divergence set on the **BEAM** card is not retained
- It is necessary to specify it in the source routine
- It is easy with the supplied functions / subroutine

Sampling functions:

## **Analytical**

- Flat (Uniform)
- Gaussian
- Maxwell-Boltzmann
- Exponential

## **From an external file**

- Histogram
- Continuous spectrum
- Discrete spectrum

# Energy / Momentum – Analytical samplings

- Flat / uniform:

- Samples uniformly between two momentum [GeV/c] or, energy [GeV] values

```
momentum_energy = sample_flat_momentum_energy( [min], [max] )
```

- Gaussian:

- Samples from a Gaussian distribution with a given mean and FWHM value ([GeV/c] or [GeV])

```
momentum_energy = sample_gaussian_momentum_energy( [mean], [fwhm] )
```

- Maxwell-Boltzmann:

- Samples from a Maxwell-Boltzmann energy distribution with a given temperature [GeV]
- It is only meaningful if the **energy\_logical\_flag** is set to **.true.**

```
momentum_energy = sample_maxwell_boltzmann_energy( [temperature] )
```

# Energy / Momentum – Analytical samplings

- Exponential:

- Samples according the exponential distribution between two energies [GeV], with a given intensity ratio at the specified energies

```
call sample_exponential_energy_weight(  
    [e_min], [e_max], [intensity_ratio],  
    momentum_energy, particle_weight  
)
```

- It is only meaningful if the **energy\_logical\_flag** is set to **.true.**
- Note 1: Different syntax used (function vs. subroutine)
- Note 2: The returned values **momentum\_energy** and **particle\_weight** are among the arguments
- Note 3: **This is a biased sampling!** It is not suitable for cases where fully analogue simulation is required (E.g. scoring with **DETECT** card)

# Energy / Momentum – Sampling from file

- Histogram:

- Samples from a histogram specified in an external file

```
momentum_energy = sample_histogram_momentum_energy(  
    [filename], [unit]  
)
```

- The external file should have 3 columns:
  - Lower energy boundary of the histogram's bins
  - Upper energy boundary of the histogram's bins
  - Intensity per energy unit (dN/dp or dN/dE)
- The particle momentum / energy is sampled uniformly within a bin
- **[unit]** variable is the momentum / energy unit used in the external file. Possible values:
  - TeV/c, GeV/c, MeV/c, keV/c, J
  - TeV, GeV, MeV, keV, eV, J

# Energy / Momentum – Sampling from file

- Continuous spectrum:

- Samples from a continuous spectrum specified in an external file

```
momentum_energy = sample_spectrum_momentum_energy(  
    [filename], [unit]  
)
```

- Samples from a discrete (line) spectrum specified in an external file

```
momentum_energy = sample_discrete_momentum_energy(  
    [filename], [unit]  
)
```

- The external file should have 2 columns:

- Energy
- Intensity at the specified energy

- With the continuous spectrum, the intensity is linearly interpolated

# Energy / Momentum – Examples

- Setting energy flag to true:

```
energy_logical_flag = .true.
```

- Monoenergetic beam:

```
momentum_energy = 1.0d-1
```

- Gaussian beam:

```
momentum_energy = sample_gaussian_momentum_energy(1.0d-1, 1.0d-2)
```

# Energy / Momentum – Examples

- Sampling from an exponential distribution:

```
call sample_exponential_energy_weight(1.0d-6, 1.0d-3, 0.01d0,  
& momentum_energy, particle_weight)
```

- Sampling from an external spectrum:

```
momentum_energy = sample_spectrum_momentum_energy(  
& 'spectrum.txt', 'MeV')
```

Note: The & character is for line continuation, it should always be in column 6.

# Source routine – Particle weight

```
particle_weight = ...
```

- To create biased sources (see Monte Carlo and Biasing lectures)
- Usually needed only for complex source definitions
- Note: The exponential spectrum sampling subroutine uses variable particle weight, but in that case, it is automatically set

# Source routine – Beam divergence

```
divergence_x = ...  
divergence_y = ...
```

- By default:
  - values are taken from the **BEAM** card
  - It is assumed to be a flat angular distribution
- For Gaussian divergence the following logical value must be set *.true.*

```
gaussian_divergence_logical_flag = .true.
```

# Source routine – Beam starting position

```
coordinate_x = ...  
coordinate_y = ...  
coordinate_z = ...
```

- By default, values are taken from the **BEAMPOS** card
- Beam shape set in the **BEAM** card are not implemented
- Extended sources specified in additional **BEAMPOS** cards are not implemented

# Source routine – Beam starting position

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

Flat distribution:

```
coordinate_[a] = sample_flat_distribution( [min], [max] )
```

Gaussian distribution:

```
coordinate_[a] = sample_gaussian_distribution( [mean], [fwhm] )
```

Annular distribution:

```
call sample_annular_distribution( [rmin], [rmax],  
                                coordinate_[a], coordinate_[b] )
```

Remember the values must be in double precision (**1.0d0**).

*Note:* If annular sampling is used, the coordinates have to be selected as well.

# Source routine – Beam direction




```
direction_cosx = ...  
direction_cosy = ...  
direction_cosz = ...
```

- By default, values are taken from the **BEAMPOS** card
- If the **direction\_flag** is set to: `direction_flag = ...`
  - 0 : All three values are considered and they are normalised automatically (Default)
  - 1 : The manually set value of the z direction is disregarded. Instead, it is calculated from the x and y direction cosines with a positive sign.
  - 2 : As with option 1, but negative sign is used.
- A predefined subroutine is already available for isotropic direction sampling

```
call sample_isotropic_direction ( direction_cosx,  
                                direction_cosy, direction_cosz )
```

# Source routine – Unused values

- It is important to remember, not all values used in the FLUKA input are used in the source routine:
  - The beam momentum distribution
  - The shape of the extended beam / volumetric sources
  - The separate coordinate system set up for the beam

|   |                    |                                    |                    |
|---|--------------------|------------------------------------|--------------------|
|  <b>BEAM</b>     | Beam: Momentum ▼   | p:                                 | Part: ▼            |
| <del>Δp: Flat ▼</del>   | <del>Δp:</del>     | <del>Δφ: Flat ▼</del>              | <del>Δφ:</del>     |
| <del>Shape(X): Rectangular ▼</del>  | <del>Δx:</del>     | <del>Shape(Y): Rectangular ▼</del> | <del>Δy:</del>     |
|  <b>BEAMPOS</b>  | x:                 | y:                                 | z:                 |
|   | cosX:              | cosy:                              | Type: POSITIVE ▼   |
|  <b>BEAMAXES</b> | cosBxx:            | cosBxy:                            | cosBxz:            |
|   | <del>cosBzx:</del> | <del>cosBzy:</del>                 | <del>cosBzz:</del> |

- If one of these features is required, it needs to be programmed in the source routine as well by using the available sampling procedures or by custom code.

# Source routine – Phase-space sampling

- Used for the second step in a two-step simulation

- It reads a file containing information on individual particles:

- Particle code
- Momentum / energy
- Starting coordinate
- Starting direction
- Weight

- Can replay the particles sequentially, or select from them randomly

```
*      call read_phase_space_file( [filename], [en
*      particle_code   = phase_space_entry%pc
*      momentum_energy = phase_space_entry%m_e
*
*      energy_logical_flag = .true.
*
*      coordinate_x = phase_space_entry%x
*      coordinate_y = phase_space_entry%y
*      coordinate_z = phase_space_entry%z
*
*      direction_cosx = phase_space_entry%u
*      direction_cosy = phase_space_entry%v
*      direction_cosz = phase_space_entry%w
*
*      particle_weight = phase_space_entry%wei
```

# Source routine – Debugging

- To help debug the source routine, the major particle parameters can be printed
- To enable this feature, set `debug_logical_flag = .true.`

- The printed parameters:

- Particle type
- Energy / momentum
- Coordinates
- Direction
- Weight

```
source_newgen.f - Debug output
-----
Particle -E [GeV]      X [cm]      Y [cm]      Z [cm]      Cosx      Cosy      Cosz      Weight
3 -4.6846462E-02      0.0000000E+00 0.0000000E+00 0.0000000E+00 -1.2028663E-03 -4.0994198E-04 9.9999919E-01 1.0000000E+00
3 -2.2732349E-03      0.0000000E+00 0.0000000E+00 0.0000000E+00 -8.1620103E-04 5.0567202E-04 9.9999954E-01 1.0000000E+00
3 -3.0508784E-04      0.0000000E+00 0.0000000E+00 0.0000000E+00 4.6318357E-04 6.8805110E-04 9.9999966E-01 1.0000000E+00
3 -6.0162525E-04      0.0000000E+00 0.0000000E+00 0.0000000E+00 -8.4087805E-04 8.2140424E-04 9.9999931E-01 1.0000000E+00
3 -1.1779098E-03      0.0000000E+00 0.0000000E+00 0.0000000E+00 8.7684219E-04 1.5051092E-03 9.9999848E-01 1.0000000E+00
3 -2.1253372E-03      0.0000000E+00 0.0000000E+00 0.0000000E+00 4.8552257E-04 -1.3290855E-04 9.999987E-01 1.0000000E+00
3 -6.2072769E-04      0.0000000E+00 0.0000000E+00 0.0000000E+00 3.0720252E-04 8.0856146E-05 9.9999925E-01 1.0000000E+00
```

- The number of primaries printed can be set with:

```
debug_lines = 100
```

# 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 (max. 8 characters) (**SDUSOU**)

```
# 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:
```

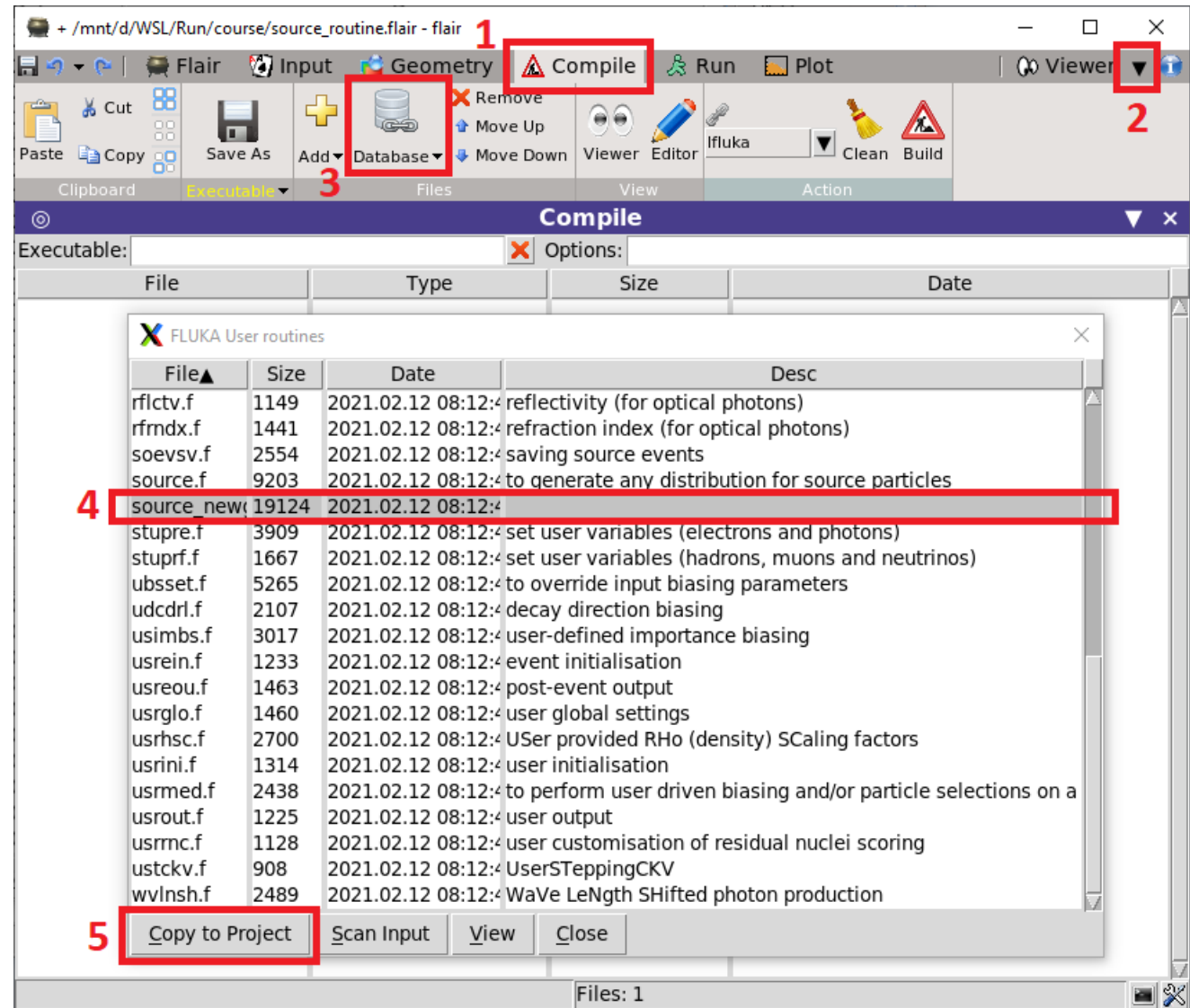
- Good practice:

Even if the beam energy / momentum is defined in the source routine, specify it in the **BEAM** card as it is used for internal initialisation. Set a momentum value higher than the maximum possible one.

# Adding the user routine to the project folder

1. Open the [Compile] tab
2. It is maybe hidden in the dropdown menu
3. Click the [Database] button (Use [Add] for an existing file)
4. Select the user routine you want to use
5. Click [Copy to Project]

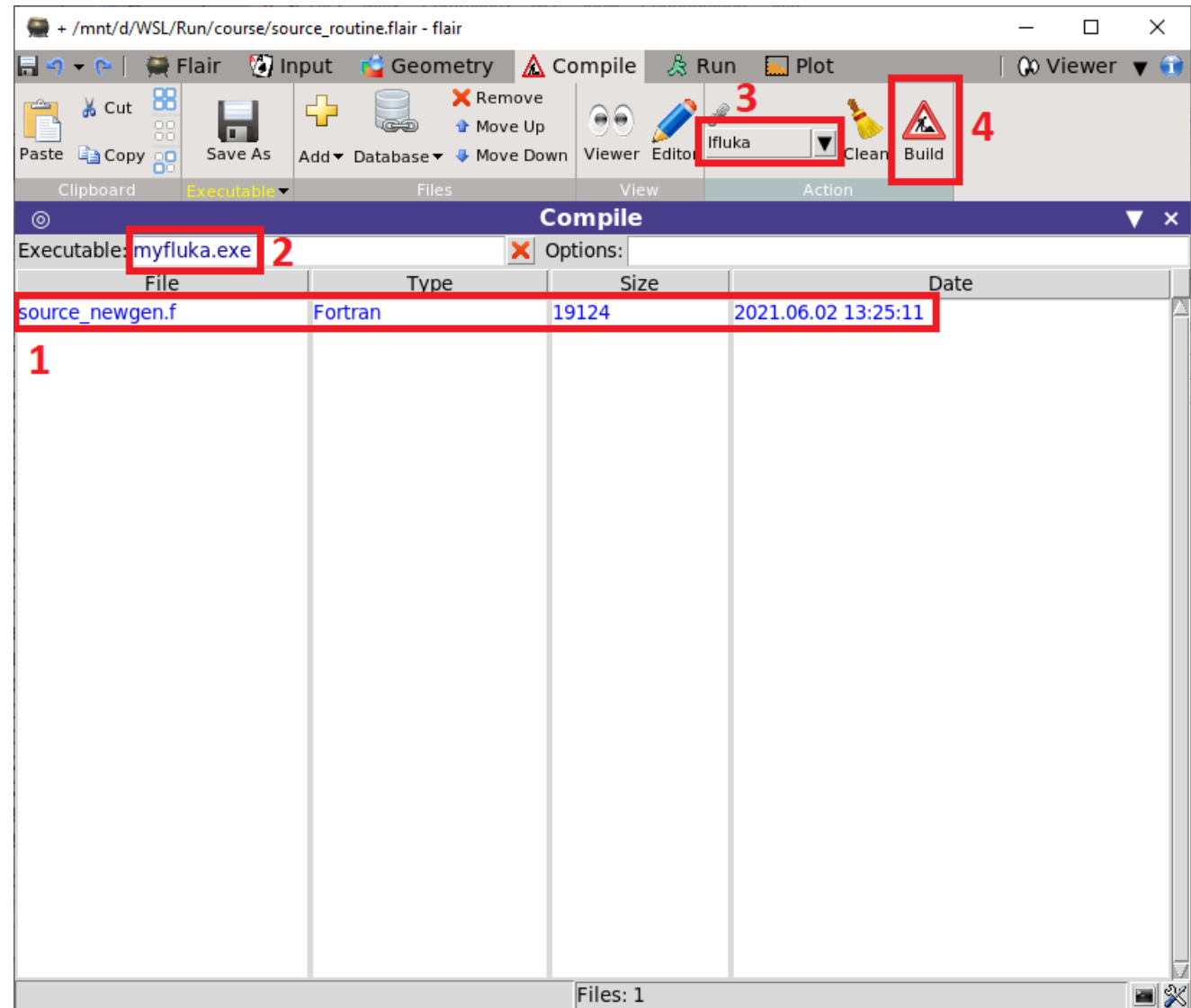
The copied user routine will be in the Flair project directory



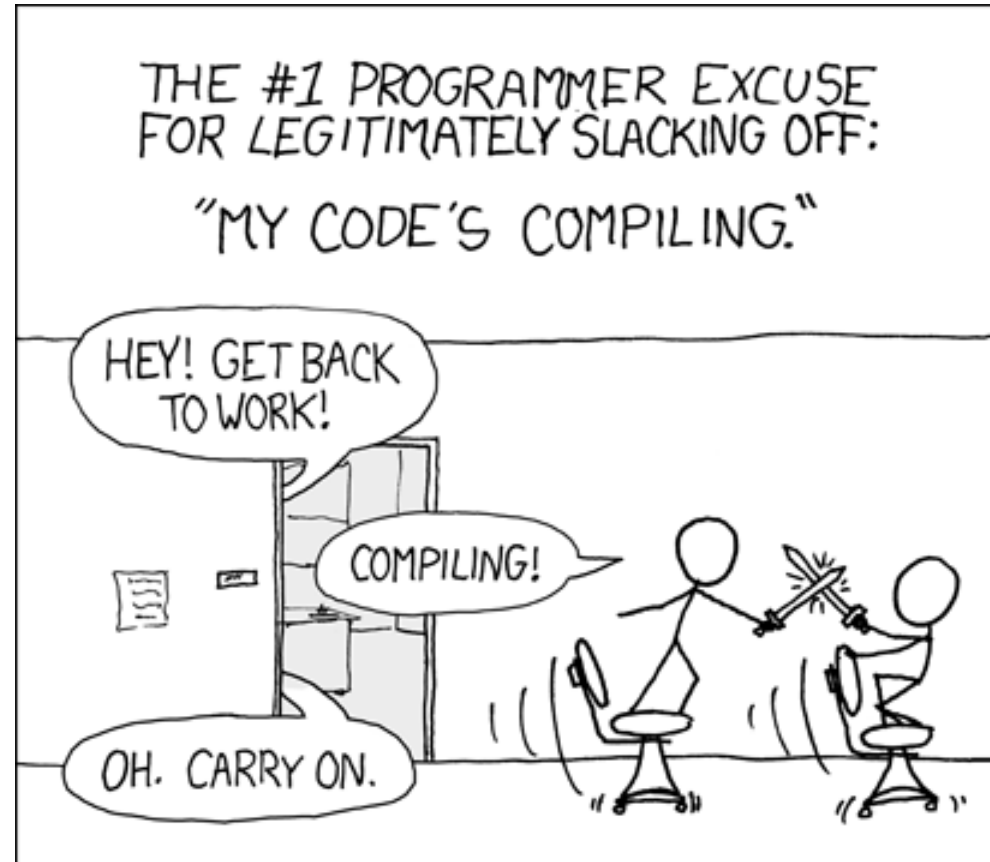
# Compiling a custom FLUKA executable

1. Verify that the user routine is in the list
2. Name your custom executable
3. Select the appropriate linker:
  - a. Use *lfluka* by default
  - b. Use *ldpmqmd* if DPMJET or RQMD models are needed
4. Compile the executable

Users must check that the custom executable is selected in the [Run] tab



# Time for an exercise!



[xkcd.com/303](http://xkcd.com/303)

