以下のサンプルコードを説明する

(1) udm_int_sample1.f90

(2) udm_int_sample2.f90

(3) udm_part_sample1.f90

(4) udm_part_sample2.f90

(5) udm_Manager.f90

説明の箇所を変更することで、独自のユーザーコードを作成できる

ファイル名に関するルール

相互作用を定義するファイル名は以下のようにする

`udm_int_<Your File Name>.f90`

粒子を定義するファイル名は以下のようにする

`udm_part_<Your File Name>.f90`

[Note]: makefile の `udm_int_*.f90`, `udm_part_*.f90`
の箇所を変更することで、これ以外のファイル名をつけられる

相互作用ファイル（*udm_int_*.f90*）に関して

ユーザーが主に変更する関数/サブルーチンは以下の２つ

`function Xsec_per_atom(...)`
相互作用の全断面積を定義する。

`subroutine generate_final_state`
相互作用の分布に基づき、相互作用の終状態をサンプリングする。

粒子ファイル（*udm_part_*.f90*）に関して

ユーザーが主に変更する関数/サブルーチンは以下の３つ

`function mass()`
粒子の質量を定義する。

`function lifetime()`
粒子の寿命を定義する。

`subroutine decay`
粒子の崩壊を定義する。

(1) udm_int_sample1.f90

電子または陽電子の入射によりユーザー定義粒子 (X) を放出させる

e + atom (Z,A) → e + X + ...

```
1   ! Interaction Template Version = 1.0
2
3   !=======================================================================
4   module udm_int_sample_1
5   !=======================================================================
6
7   use udm_Parameter
8   use udm_Utility
9   implicit none
10  private ! Functions and variables are set to private by default.
11  public :: caller ! The 'caller' subroutine should be
12
13  !-----------------------------------------------------------------------
14  ! Default variables
15  !-----------------------------------------------------------------------
16  character(len=99), parameter :: Name = "my_interaction_1" ! This
17  double precision, allocatable, save :: Parameters(:) ! Para
18  integer, parameter :: num_initial = 2 ! The number of incident pa
19  integer, save :: kf_initial(num_initial) = (/ 11, -11 /) !
20  !-----------------------------------------------------------------------
21  ! User variables
22  !-----------------------------------------------------------------------
23  ! integer i,j
24  ! double precision x,y
25  integer kf_X
26  contains
```

```
329
330  !=======================================================================
331  end module udm_int_sample_1
332  !=======================================================================
333
334
```

任意のモジュール名を定義する。
後述の udm_Manager.f90 内で使用する。

この相互作用に対する Name を定義する。これは、インプットファイル内で使用する。

この相互作用を引き起こす入射粒子の数。

この相互作用を引き起こす入射粒子の kf-code.
この場合は、電子 (11) と陽電子 (-11)

(1) udm_int_sample1.f90

```
39  !=======================================================================
40  subroutine initialize
41  ! This subroutine is called only once at the beginning of the calculation.
42  !=======================================================================
43  kf_X=Parameters(1) ! kf-codes (particle ID) of X (outgoing particle).
44  end subroutine initialize
```

インプットファイルの [user defined interaction] セクションに入力したパラメータは
ソースコードの中では、配列 `Parameters(i)` によって利用できる。

（インプットファイルの入力例）

```
[ user defined interaction ]
  n_int = 2

$ Name                  Bias     Parameters
  my_interaction_1      1        900000
  my_interaction_3      100      900000   1.1   2.2
```

この値は、`Name` が `my_interaction_1` のソースコード内
において、`Parameters(1)` として利用可能

`Parameters(1)`

`Parameters(2)`

`Parameters(3)`

（`my_interaction_3` 内において）

(1) udm_int_sample1.f90

関数 Xsec_per_atom に 1 原子あたりの全断面積を定義する。単位は barn.

入射粒子の kf-code は udm_kf_incident に格納されている

```
48  !=======================================================================
49  double precision function Xsec_per_atom(Kin,Z,A)
50  ! Integrated cross section per an atom.
51  ! Unit: barn (10^-24 cm2)
52  implicit none
53  double precision Kin ! Kinetic energy of incident particle [MeV]
54                   integer Z ! Atomic number of target atom
55                   integer A ! Mass   number of target atom
56  !-----------------------------------------------------------------------
57  ! [Variables available in this function]
58  ! udm_kf_incident : The kf-codes (particle IDs) of the incident particles.
59  !=======================================================================
60
61  if(Kin < 100.0) then
62    Xsec_per_atom=0.0
63    return
64  endif
65
66  if     (udm_kf_incident ==  11) then
67    Xsec_per_atom=1e-6*Z
68  else if(udm_kf_incident == -11) then
69    Xsec_per_atom=2e-6*Z
70  else
71    print*,"error"
72    stop
73  endif
74
75  return
76  end
```

この場合、入射粒子運動エネルギー (= Kin) が 100 MeV 以下の場合は 0 barn.

入射粒子が電子の場合は、10^{-6} * Z barn, 陽電子の場合は、2 * 10^{-6} * Z barn.

(1) udm_int_sample1.f90

前半

```
!==========================================
subroutine generate_final_state
!==========================================
! Subroutine to determine final state infor
! In this example, the X and e+- momenta of
! [Variables available in this subroutine]
! udm_Kin : Kinetic Energy of the i
!-----------------------------------------
integer Z_A_hit(2), Z, A
double precision Kout, Kout_min, Kout_max
double precision P, R
double precision m_X, E_X, p_X, theta_X
double precision m_e, E_e, p_e, theta_e
!-----------------------------------------
! You may use Z and A for final state varia
! Z and A of a target material are automati
Z_A_hit=get_hit_nuclide_Z_A(udm_Kin)
Z=Z_A_hit(1)
A=Z_A_hit(2)
!-----------------------------------------

! ==================================
! = Your original sampling section =
! ==================================
! Range of sampling variable
Kout_min=0.0d0
Kout_max=udm_Kin-get_mass(kf_X)
!
! Start sampling
n_sampling_max = 100000

                        in the range.
  Kout=get_random(Kout_min, Kout_max)
  !-----------------------------------
  ! Rejection sampling method.
  P=distribution(udm_Kin, Z, A, Kout) ! The
  R=get_random_0to1(
  ! If P > R, this K
  if(P > R) then
    !
    ! Accepted!!!
    ! Initialization required before fi
    call initialize_udm_event_info
    !
    !
    ! -----------------------------------
    ! Set number of final state
    set_final_state_number = 2
    !
```

generate_final_state で終状態をサンプリングする。ここでは *Rejection sampling method* が用いられている。

必要に応じて相互作用した原子情報 (Z, A) を取得する

粒子Xのエネルギー (Kout) の分布は別途定義されている

この Kout が採用された

【必須】終状態の情報をセットする前に配列を初期化する

【必須】セットしたい終状態の粒子数の数を指定する

後半

```
    !
    ! -----------------------------------
    ! [Important Notice]
    ! Here, you set the final state momenta to the following "set_*" array,
    ! assuming the direction of the momentum of the incident particle to be
    ! positive along the "Z-axis". The final state momenta are automatically
    ! rotated based on the actual direction outside of this subroutine.
    ! -----------------------------------

    m_X=get_mass(kf_X)
    E_X=Kout+m_X
    p_X=sqrt(E_X**2-m_X**2)
    theta_X=get_random(0.0d0,0.1d0)
    !
    ! Set 4-momentum of X
    set_kf                  (1) = kf_X
    set_Total_Energy_in_MeV(1) = E_X
    set_Px_in_MeV           (1) = p_X*sin(theta_X)
    set_Py_in_MeV           (1) = 0.0d0
    set_Pz_in_MeV           (1) = p_X*cos(theta_X)
    !
    m_e=get_mass(udm_kf_incident)
    E_e=udm_Kin+m_e-E_X
    p_e=sqrt(E_e**2-m_e**2)
    theta_e=get_random(0.0d0,0.1d0)
    !
    ! Set 4-momentum of e+-
    set_kf                  (2) = udm_kf_incident
    set_Total_Energy_in_MeV(2) = E_e
    set_Px_in_MeV           (2) = p_e*sin(theta_e)
    set_Py_in_MeV           (2) = 0.0d0
    set_Pz_in_MeV           (2) = p_e*cos(theta_e)
    !

    ! [Additional Quantities]
    ! set_isomer_level            (?) = ?? ! (Default=0)    0: Not nuclear isome
    ! set_excitation_energy_in_MeV(?) = ?? ! (Default=0.0) Excitation energy [M

    ! -----------------------------------
    ! The final states are recorded.
    call fill_final_state2
    ! -----------------------------------
    return
```

入射粒子の向きがZ軸と仮定し終状態の運動量を計算する。

【必須】1つ目の終状態（粒子X）の kf-code、エネルギー、運動量を用意された配列にセットする

【必須】2つ目の終状態（e+/e--）の kf-code、エネルギー、運動量をセットする

【必須】セットした後に call する

(2) udm_int_sample2.f90

udm_int_sample1.f90 と同様のコード。ただし、入射粒子がミューオンである。

μ + atom (Z,A) → μ + X + ...

```
16   character(len=99), parameter :: Name = "my_interaction_2"
17   double precision, allocatable, save :: Parameters(:) ! Par
18   integer, parameter :: num_initial = 2 ! The number of inci
19   integer, save :: kf_initial(num_initial) = (/ 13, -13 /) !
```

【変更箇所】
入射粒子が μ- (13) と μ+ (-13).

(3) udm_part_sample1.f90

"(1) udm_int_sample1.f90" 等で生成される 粒子 X を定義する。

```
1   ! Particle Template Version = 1.0
2
3   !=============================================================
4   module udm_part_sample_1
5   !=============================================================
6
7   use udm_Parameter
8   use udm_Utility
9   implicit none
10  private ! Functions and variables are set to p
11  public :: caller ! The 'caller' subroutine sh
12
13  !-------------------------------------------------------------
14  ! Default variables
15  !-------------------------------------------------------------
16  character(len=99), parameter :: Name = "my_particle_1"
```

任意のモジュール名を定義する。
後述の udm_Manager.f90 内で使用する。

この粒子に対する Name を定義する。
これは、インプットファイル内で使用する。

```
145 !=============================================================
146 end module udm_part_sample_1
147 !=============================================================
```

(3) udm_part_sample1.f90

```
46  !=======================================================================
47  double precision function mass() ! Unit: MeV
48  !=======================================================================
49  mass=50.0 ! 50 MeV
50  return
51  end
52
53  !=======================================================================
54  double precision function lifetime() ! Unit: Second
55  ! The value should be greater than 0.
56  !=======================================================================
57  lifetime=0.1e-9 ! 0.1 nano second
58  return
59  end
60
61  !=======================================================================
62  subroutine decay
63  !=======================================================================
64  ! Variables available in this subroutine:
65  ! udm_Kin          : Kinetic Energy of the incident particle [MeV]
66  !-----------------------------------------------------------------------
67  if(0.5 > get_random_0to1()) then
68    call two_body_decay_uniform(12,12)       ! X -> 2 neutrinos (50%)
69  else
70    call three_body_decay_uniform(12,12,12) ! X -> 3 neutrinos (50%)
71  endif
72  end subroutine decay
```

質量を 50 MeV に定義

寿命を 0.1 x 10^{-9} 秒に定義

分岐比 50%

0~1の範囲で乱数を取得

decay サブルーチンで崩壊パターンを定義

two_body_decay_uniform (kf1, kf2) ： kf-code が kf1とkf2 の 2 体に崩壊させるサブルーチン

three_body_decay_uniform (kf1, kf2, kf3) ： kf-code が kf1, kf2, kf3 の 3 体に崩壊させるサブルーチン

（ただし、それぞれ崩壊粒子のスピンの向きは考慮されない）

この場合は、50% が２つの電子ニュートリノ (kf=12) に、50% が３つの電子ニュートリノに崩壊する（非物理的だがわかりやすさのため）

(4) udm_part_sample2.f90

udm_part_sample1.f90 と同様のコード

ただし、質量と寿命は [user defined particle] セクションで入力された値を用いる

```
 4   module udm_part_sample_2
                ⋮
16   character(len=99), parameter :: Name = "my_particle_2"
                ⋮
46   !=============================================================
47   double precision function mass() ! Unit: MeV
48   !=============================================================
49   mass=Parameters(1) ! MeV
50   return
51   end
52
53   !=============================================================
54   double precision function lifetime() ! Unit: Second
55   ! The value should be greater than 0.
56   !=============================================================
57   lifetime=Parameters(2) ! second
58   return
59   end
```

（インプットファイルの入力例）

```
[ user defined particle ]
  n_part = 1

$ Name             kfcode    Parameters
  my_particle_2    900000    100   1.0e-9
```

Parameters(1)　Parameters(2)

(5) udm_Manager.f90

利用したいユーザーコードの情報を udm_Manager.f90 に記入する

利用したい全てのモジュールを並べる

```
module udm_Manager
use udm_Parameter

!=============================================================
! [udm_int]
use udm_int_sample_1,   caller_udm_int_sample_1  => caller
use udm_int_sample_2,   caller_udm_int_sample_2  => caller
! [udm_part]
use udm_part_sample_1,  caller_udm_part_sample_1 => caller
use udm_part_sample_2,  caller_udm_part_sample_2 => caller
!=============================================================
```

use <モジュール名>, caller_<モジュール名> => caller
のように記入する

利用したい相互作用を並べる

```
subroutine user_defined_interaction(action,index)
integer action,index
!=============================================================
call caller_udm_int_sample_1(action,index)
call caller_udm_int_sample_2(action,index)
!=============================================================
end subroutine user_defined_interaction
```

call caller_<モジュール名>(action,index)
のように記入する

利用したい粒子を並べる

```
subroutine user_defined_particle(action)
integer action,index
do index=1,udm_part_nMax
!=============================================================
call caller_udm_part_sample_1(action,index)
call caller_udm_part_sample_2(action,index)
!=============================================================
```

call caller_<モジュール名>(action,index)
のように記入する