Internal PSI draft

MANUAL OF MUSRSIM

KAMIL SEDLÁK AND TONI SHIROKA PSI

The program "musrSim" is a simulation program based on Geant4 optimised for the μSR instruments. This document describes some of the commands used in the user macros of the simulation program for the μSR instruments based on the GEANT4. The root output variables are also described.

1 Initial muon parameters

/gun/vertex *x0 y0 z0 unit*
(default: /gun/vertex 0 0 -100 mm)
Set mean values of the x, y and z coordinates of the generated particles (muons). The smearing around these mean values instead is set by /gun/vertexsigma and restricted by /gun/vertexboundary (see below).
(Applicable also to TURTLE input).

/gun/vertexsigma *xSigma ySigma zSigma unit*
(default: /gun/vertexsigma 0 0 0 mm)
If $xSigma > 0$ ... set σ, i.e. the standard deviation (RMS), of the x coordinate of the generated particles (muons) to $\sigma = xSigma$. The x coordinate of the initial muon is then generated according to the Gaussian distribution with the mean value of $x0$ and the standard deviation of $xSigma$.
If $xSigma < 0$ ... the x coordinate of the initial muon is generated **uniformly** in the interval of ($x0-$ $xSigma$, $x0+$ $xSigma$).
If $xSigma = 0$ ... no smearing on the x coordinate is applied.
Similar is true for $ySigma$ and $zSigma$.
This variables are ignored when TURTLE input is requested.

/gun/vertexboundary *R_max z_min z_max unit*
Set maximum allowed radius, and minimum and maximum z coordinate of the generated particles (muons). This command might be useful especially if the user wants to restrict the area, in which initial particles are created, e.g. to force the initial muons to be created inside the beam pipe.

If the particles (muons) are read in from an external TURTLE file, only the restriction on the maximum radius R_max is applied on the initial particles, while z_min and z_max are ignored.

/gun/kenergy *kineticEnergy unit*
Set the mean kinetic energy of the initial particles (muons).

/gun/momentum *momentum unit*
Set the mean momentum of the initial particles (muons).

/gun/momentumsmearing *momentumSigma unit*
Set σ, i.e. the standard deviation (RMS), of the momentum spread, which is aplied randomly to each generated initial particle (muon). It is the magnitude of the momentum, which is smeared.
(Ignored by the TURTLE input. However, a similar command "/gun/turtleMomentumBite" can be used for the TURTLE input file.)

/gun/momentumboundary *p_min p_max dummy unit*
Set a boundary for the minimum and maximum momentum of the initial particles (muons). The third argument *dummy* is ignored.
(Presently ignored by the TURTLE input).

/gun/tilt *xangle0 yangle0 dummy unit*
The "beam tilt" is understood as a constant angle tilt that is applied to all initial particles (muons) regardless on their distance from the centre of the beam.
(Applicable also to TURTLE input).

/gun/tiltsigma *xangleSigma yangleSigma dummy unit*
Gaussian smearing of the tilt angle.
(Presently ignored by the TURTLE input).

/gun/pitch *pitch unit*
The "beam pitch" is understood as a variable angle applied to the initial particles (muons), which is directly proportional to the distance from the beam axis. The particles closer to the beam axis become smaller pitch than particles further away from the beam axis. The angle given as *pitch* will be applied to a particle generated 1 mm away from the beam centre, i.e. the particle generated 7 mm away from the beam axis will be assigned the angle of $7 \cdot pitch$. The pitch allows the user to focus or defocuse the initial particle beam. Particles will be focused for positive pitch and defocused for the negative pitch.
(Applicable also to TURTLE input).

/gun/muonPolarizVector *xpolaris ypolaris zpolaris*
Set polarisation of the initial particle (muon) in a form of a vector $P = (xpolaris, ypolaris, zpolaris)$. The polarisation vector does not have to be normalised, the normalisation will be done internally by the program. However note that if the magnitude of P is set to less than 1e-8, the user can achieve an unpolarised muon beam. See the source code of musrPrimaryGeneratorAction.cc if you need to use unpolarised beam by this parameter, because there is some trick based on the magnitude of P.
(Applicable also to TURTLE input).

/gun/muonPolarizFraction *polarisFraction*
Set the fraction of the muon polarisation. The variable *polarisFraction* has to be set in the range of -1 to 1.
If *polarisFraction* is set to 1, all muons are polarised in the direction of polarisation vector defined by "/gun/muonPolarizVector".
If *polarisFraction* is set to 0, half of the muons are polarised in the direction of polarisation vector, the second half is polarised in the oposite dirrection, so in the end the muon beam should act as unpolarised.
If *polarisFraction* is set to -1, all muons are polarised in the direction oposite to the polarisation vector.
If *polarisFraction* is set to 0.9, then 95% of the muons is polarised in in the direction of polarisation vector, and 5% of them is polarised in the oposite dirrection!.
This command is ignored if magnitude of polarisation vector defined by "/gun/muonPolarizVector" is smaller than 1e-8!
(Applicable also to TURTLE input).

/gun/decaytimelimits *muDecayTimeMin muDecayTimeMax muMeanLife unit*
(default: /gun/decaytimelimits −1 −1 2197.03 ns)
If *muDecayTimeMax* is less or equal to zero, this command is ignored, and the muon decay time is set internally by GEANT4. Otherwise the muon will be forced to decay within a time interval given by *muDecayTimeMin* and *muDecayTimeMax*, and the mean muon lifetime will be set to *muMeanLife*. In such case *muDecayTimeMin* has to be equal or larger than 0 and *muDecayTimeMax* has to be larger **or equal** to *muDecayTimeMin*.
(Applicable also to TURTLE input).

/gun/turtlefilename *turtleFileName*
Set the filename of the TURTLE input file. If this varialble is set, TURTLE file will be used to initiate muons. Otherwise the mouns would be generated randomly.

/gun/turtleZ0position *z0_InitialTurtle unit*
Set the z-position which has been used to generate the TURTLE file.
If this value differes from the $z0$ value of the "/gun/vertex" command, than the particle initial position is extrapolated from $z0\_InitialTurtle$ to the point corresponding to $z0$, using the direction of its momenta.
MORE DETAILS:
When running TURTLE (e.g. when generating the TURTLE file using the TURTLE program), the user has to specify the z position, at which the TURTLE particles (muons) would be exported. Sometimes this z position does not correspond to the point of origin of the musrSim geometry. In

such case, the variable *z0_InitialTurtle* should be set to the value, that in musrSim coordinate system corresponds to the point, at which the TURTLE file was exported. For example – if the TURTLE file was exported just after the last quadrupole of a beam-pipe, and in the simulation the edge of the last quadrupole corresponds to 100 cm, than the *z0_InitialTurtle* should be also set to 100 cm.

/gun/turtleMomentumBite *turtleMomentumP0 turtleSmearingFactor dummy*
Modify the smearing of the momentum bite specified in the TURTLE input file. Normally the muon momentum is defined already in the turtle input file. This command allows the user to modify the momentum smearing (momentum bite) of the muon beam. The variable *turtleMomentumP0* will be taken as the mean momentum (in MeV/c), around which the momentum will be increased/decreased. It does not have to be the real mean value of the initial muon momentum distribution. The variable *turtleSmearingFactor* is the smearing factor in per cent, by which the momentum bite will be increased/decreased around the *turtleMomentumP0*. The following equation is used to change the muon momentum: p^{new} = *turtleMomentumP0* - (*turtleMomentumP0*-p^{TURTLE})·0.01·*turtleSmearingFactor*.
This means that:
turtleSmearingFactor = 100 ... the muon beam momentum will not be modified.
turtleSmearingFactor = 0 ... the muon beam momentum will be set to the constant value of *turtleMomentumP0*.
turtleSmearingFactor = 200 ... the muon beam will have two times broader distribution compared to the original TURTLE file.

/gun/turtleFirstEventNr *lineNumberOfTurtleFile*

2 Detector construction

/musr/command rotation *matrixName* α β γ
/musr/command rotation *matrixName vx vy vz angle*
These commands define a rotation matrix of the name "matrixName" that can be used later on during the definition of the detector geometry (see command "/musr/command construct"). It can be defined either by the Euler angles (if there are three float parameters behind the *matrixName*) or by the vector *(vx,vy,vz)* and an *angle* of rotation around this vector (if the fourth float parameter behind the *matrixName* is non-zero). All angles are specified in degrees.

/musr/command construct *solid*=string *name*=string *dimensions*=float ... *material*=string *x*=float *y*=float *z*=float *motherVolume*=string *rotationMatrix*=string *sensitiveClass*=string *idNumber*=int
This command defines a volume in GEANT4 (It comprises three steps of GEANT4: defines a solid, logical volume and physical volume. More details have to be found in GEANT4 manual).

- *solid* can be one of the G4VSolid.cc particular types, presently "tubs", "box", "sphere", or it can be one of the specifically implemented solids by our program as "uprofile" (an U-profiled bar), "alcSupportPlate" (shape specific to ALC support plate), "tubsbox" (a tube with a rectangular hole along its axis) and "tubsboxsegm" (a volume that looks like an intersection of tube and box). Not all G4VSolids are presently supported, but it is relatively easy to implement a new kind of solids in the musrDetectorConstruction.cc class.
- *name* stands for the name of the volume. As the "/musr/command construct" construct three kinds of classes (volumes) – the solid, logical volume and physical volume – there are three names of the concrete volume used internally inside the program: sol_*name*, log_*name* and phys_*name*. The main volume, inside which all other volumes are positioned, has to be called "World".
- *dimensions* define the size of the required solid. They are kept equivalent to the dimensions of solids as used in GEANT4. For example the "box" is defined by its halfwidths along x, y and z coordinates. Note that the number of *dimensions* varies for each type of solid.
- *material* one of the materials defined in GEANT4, namely in the file $G4INSTALL/source/materials/src/G4NistMaterialBuilder.cc (e.g. "G4_Galactic" for vacuum, "G4_Cu" for copper, "G4_AIR" for air and "G4_PLASTIC_SC_VINYLTOLUENE"

for a scintillator). One can also define a new material inside the function musrDetectorConstruction::DefineMaterials(). Presently "Mylar", "Brass" and "Steel" are defined there.

- *x, y, z* – coordinates of the volume, used to position the volume within its mother volume (as used by the G4PVPlacement).
- *motherVolume* – name of the mother volume, in which the given volume should be positioned. Note that the mother volume has to be defined first (before its daughter), and that the name of mother starts with a string **log_***name*, following the naming convention defined above. When the "World" volume is defined, its *motherVolume* should be set to "no_logical_volume".
- *rotationMatrix* – name of the rotation matrix that will be used to position the volume inside its mother volume (as used in member function G4PVPlacement()). Use string "norot" if no rotation is required for the given volume. Otherwise the rotation matrix has to be defined by the command line "/musr/command rotation" **before** the given is defined.
- *sensitiveClass* – specifies whether the volume is sensitive or not. Use the string "dead" for the non-senstive volume (i.e. for the dead material), and the string "musr/ScintSD" for a scintillator (a sensitive volume, i.e. a volume where hits are observed). No other detector type (other than "dead" and "musr/ScintSD") is supported at the moment, but the program might be extended in the future (e.g. to properly include also the semiconductor tracking detectors, etc.).
- *idNumber* – idNumber serves as a unique identifier of the volume. It is primarily used in the output Root tree to identify the volume: 1) in which muons stop (tree variable "muDecayDetID"), 2) in which hits were deposited in case of sensitive volume (the variable "det_ID[det_n]").

/musr/command logicalVolumeToBeReweighted mu *logicalVolume*=string *weight*=int
(default: not defined; no reweighting is done unless explicitly requested by this command.)
Events can be reweighted by this command. If muon **stops and decays** in the volume *logicalVolume*, the event will be reweighted using the requested *weight*. Namely, only each n^{th} event will be stored (n =*weight*) with the parameter "weight" in the Root output tree set to *weight*, while other (non-n^{th}) events will be aborted. (The decision which event is to be stored and which to be aborted is done at random). This reweighting might be usefull in the cases when the user wants to speed-up the simulation (respectively to reduce the number of fully simulated events), while keeping the high number of events interesting for the analysis. For example, one can set the reweighting of events in which muons stop in the collimator. One should then use the *weight* stored in the Root tree when filling histograms. Compared to the simulation with no weighting applied, the histograms with weighted events will have larger errors, but the distribution should not differ more then within the statistical errors.
Note that the *weight* parameter is integer, and "mu" stands for "muons" (at the moment reweighting based on electrons or positrons is not supported).

3 Visualisation

/musr/command visattributes *volumeName color*
/musr/command visattributes *materialName color*
In case of visualisation, one can set the color of a logical volume *volumeName* or of all volumes made of the material with the name *materialName*. The distinction between the two options is by the first four letters of the *volumeName* – if it contains the string "log_", it is considered as *volumeName*, otherwise it is considered to be a material with *materialName*.

Presently the following colors are predefined: "invisible", "white", "black", "red", "green", "blue", "lightblue", "yellow", "gray", "cyan" and "magenta". New colours can be easily added, if needed, in the member function "musrDetectorConstruction::SetColourOfLogicalVolume".

4 Physics processes

/musr/command process addDiscreteProcess *particle*=string *process*=string
/musr/command process addProcess *particle*=string *process*=string *ordAtRestDoIt*=int *ordAlongSteptDoIt*=int *ordPostStepDoIt*=int
Adds processes for particles. See GEANT4 manual for more details. Look in the file musrPhysicsList.cc for the list of defined processes (e.g. G4MultipleScattering, G4eIonisation, ...)

There is one special process, combined from G4MultipleScattering and G4CoulombScattering, defined by the following command:
/musr/command process addProcess *particle*=string MultipleAndCoulombScattering *ordAtRestDoIt*=int *ordAlongSteptDoIt*=int *ordPostStepDoIt*=int *G4Region1*=string [*G4Region2*=string] [*G4Region3*=string]
The G4MultipleScattering (rough but very fast approximation of scattering) will be applied elswhere in the detector, except for the *G4Region1* (and eventually *G4Region2* and *G4Region3*), where more precise but very slow process G4CoulombScattering will be applied instead of G4MultipleScattering. Note that up to three G4Regions are supported at the moment, but this limitation is not intrinsic to GEANT4 and it can be therefore changed in musrPhysicsList.cc, if needed. The G4Regions have to be defined in the detector construction phase by the command "/musr/command region define ...".

5 Output root tree variables

The value of -999 or -1000 indicates that the given variable could not be filled (was undefined in a given event). For example if the variable "muTargetTime" is set to -1000 it means that the initial muon missed the sample, and therefore no time can be assigned to the sample hit.

runID (Int_t) – run ID number.

eventID (Int_t) – event ID number.

weight (Double_t) – event weight.

BFieldAtDecay_Bx, BFieldAtDecay_By, BFieldAtDecay_Bz, BFieldAtDecay_B3, BFieldAtDecay_B4, BFieldAtDecay_B5 (Double_t) – value of the 6 coordinates of the electromagnetic field at the position and time where and when the muon decayed. The first three coordinates correspond to the magnetic field, the last three to the electric field.

muIniPosX, muIniPosY, muIniPosZ (Double_t) – initial position where muon was generated (in mm).

muIniMomX, muIniMomY, muIniMomZ (Double_t) – initial momentum of the muon when it was generated (in MeV/c).

muIniPolX, muIniPolY, muIniPolZ (Double_t) – initial polarisation of the muon when it was generated.

muDecayDetID (Int_t) – ID number of the detector in which the muon stopped and decayed.

muDecayPosX, muDecayPosY, muDecayPosZ (Double_t) – the position where the muon stopped and decayed (in mm).

muDecayTime (Double_t) – the time at which the muon stopped and decayed (in μs).

muDecayPolX, muDecayPolY, muDecayPolZ (Double_t) – polarisation of the muon when it stopped and decayed.

muTargetTime (Double_t) – time at which the muon entered the volume whose name starts by "target" – usually the sample (in μs).

muTargetPolX, muTargetPolY, muTargetPolZ (Double_t) – polarisation of the muon when it entered the volume whose name starts with "target" – usually the sample.

muM0Time (Double_t) – time at which the muon entered the detector called "M0" or "m0" (in μs).

muM0PolX, muM0PolY, muM0PolZ (Double_t) – polarisation of the muon when it entered the detector called "M0" or "m0".

muM1Time (Double_t) – time at which the muon entered the detector called "M1" or "m1" (in μs).

muM1PolX, muM1PolY, muM1PolZ (Double_t) – polarisation of the muon when it entered the detector called "M1" or "m1".

muM2Time (Double_t) – time at which the muon entered the detector called "M2" or "m2" (in μs).

muM2PolX, muM2PolY, muM2PolZ (Double_t) – polarisation of the muon when it entered the detector called "M2" or "m2".

posIniMomX, posIniMomY, posIniMomY (Double_t) – Initial momentum of the decay positron (in MeV/c).

nFieldNomVal (Int_t) – number of the elementary fields that make together the global field.

fieldNomVal[nFieldNomVal] (array of Double_t) – nominal values of all elementary fields. (They are usually constant, but sometimes they may vary from event to event).

BxIntegral, ByIntegral, BzIntegral, BzIntegral1, BzIntegral2, BzIntegral3 (Double_t) – calculates the field integrals along the muon path and path lengths defined as

$$\text{BxIntegral} = \int_{\mu \text{ path}} B_x(s)\, ds \tag{1}$$

$$\text{ByIntegral} = \int_{\mu \text{ path}} B_y(s)\, ds \tag{2}$$

$$\text{BzIntegral} = \int_{\mu \text{ path}} B_z(s)\, ds \tag{3}$$

$$\text{BzIntegral1} = \int_{Z_0}^{Z_{decay}} B_z(z)\, dz \tag{4}$$

$$\text{BzIntegral2} = \int_{\mu \text{ path}} ds \tag{5}$$

$$\text{BzIntegral3} = \int_{Z_0}^{Z_{decay}} dz \tag{6}$$

The units are tesla·m (for the first four variables) and mm (for the last two variables). To calculate the integrals properly, the user must force Geant to use very small step size (e.g. by using something like "/musr/command globalfield setparameter SetLargestAcceptableStep 2"), and probably also to generate the muons well outside the magnetic field and put target such that muons stop at $z = 0$.

Note that these variables are by default not calculated (and not stored) and therefore the user has to switch the calculation on by "/musr/command rootOutput fieldIntegralBx on" in the macro file.

det_n (Int_t) – number of "detector hits" in this event. Note that more then 1 detector might be hit, and even the same detector might be hit more than once. The hit might be induced by just one particle, by more then one particle originating from the same particle initially hitting the detector, or from more "independent" particles. For example, the decay positron can emit an Bremsstrahlung photon in the sample and then both the Bremsstrahlung photon and positron hit the same positron counter at approximately the same time.

det_ID[det_n] (array of Int_t) – ID number of the detector where the given hit occurred.

det_edep[det_n] (array of Double_t) – energy deposited in the given hit (in MeV).

det_edep_el[det_n] (array of Double_t) – energy deposited in the given hit due to electron-based interactions (in MeV).

det_edep_pos[det_n] (array of Double_t) – energy deposited in the given hit due to positron-based interactions (in MeV).

det_edep_gam[det_n] (array of Double_t) – energy deposited in the given hit due to photon-based interactions (in MeV).

det_edep_mup[det_n] (array of Double_t) – energy deposited in the given hit due to muon-based interactions (in MeV).

det_nsteps[det_n] (array of Int_t) – number of "steps" (in GEANT4 terminology) that were integrated together in the given hit. (The det_edep[] energy is the sum of the energy deposits during all these steps).

det_length[det_n] (array of Double_t) – the length of the trajectory of the particle (particles) that contributed to the given hit (in mm).

det_time_start[det_n], det_time_end[det_n] (array of Double_t) – the initial and final time belonging of the hit. It should be the "global time" of the track when the first and last hit occurred (in μs).

det_x[det_n], det_y[det_n], det_z[det_n] (array of Double_t) – the coordinates of the first step of the given hit.

det_Vrtx***[det_n]** – All the variables starting with "det_Vrtx" refer to the particle with the first (in time) energy deposit belonging to the given hit. (Note that the hit might be induceed by more than one particle.) The vertex, at which the particle was created, may or may not be positioned within the sensitive volume, in which the hit is observed.

det_VrtxKine[det_n] (array of Double_t) – the kinetic energy of the first (in time) particle belonging to the hit.

det_VrtxX[det_n], det_VrtxY[det_n], det_VrtxZ[det_n] (array of Double_t) – the position of the vertex of the first particle that belongs to the given hit (in mm).

det_VrtxVolID[det_n] (array of Int_t) – ID of the detector in which the vertex (see above) was created.

det_VrtxProcID[det_n] (array of Int_t) – ID of the physics process in which the vertex (see above) was created.

det_VrtxTrackID[det_n] (array of Int_t) – track ID of the first particle that belongs to the given hit. If the track ID is negative, there were more than just one track contributing to this hit. The absolute value of det_VrtxTrackID[det_n] corresponds to the first (in time) track.

det_VrtxParticleID[det_n] (array of Int_t) – particle ID of the first particle that belongs to the given hit.

det_Vvv***[det_n]** – similar to the variables det_Vrtx*****[det_n] above, but if the first particle belonging to the hit was created inside of the logical volume where the hit occurs, then it's track is followed to its mother track (even several times) until the track (particle) is found that has been created outside the given volume. This way one can better investigate which (hopefully) single particle coused the hit. Even though even in this case it is not guarranteed that only a single particle gave origin to the hit, it is quite likely, though, that it was in fact just a single particle. If the

save_n (Int_t) – number of special kind of "save" volume that were hit in this event. The "save volume" is any volume whose name starts with letters "save". Their purpose in the simulation is usually to check positions and momenta of particles at some position of the detector, even if the particle does not deposit any energy in the given "save" volume. Save volumes can therefore be made of vacuum.

save_detID[save_n] (array of Int_t) – ID number of the save volume.

save_particleID[save_n] (array of Int_t) – particle ID of the particle that entered the save volume.

save_x[save_n], save_y[save_n], save_z[save_n] (array of Double_t) – position of the particle where it entered the save volume ("GetPreStepPoint()") (in mm).

save_px[save_n], save_py[save_n], save_pz[save_n] (array of Double_t) – momentum of the particle when it entered the save volume (in GeV).

6 Conclusions

The ... in [1].

7 Appendix A: Steering file for the simulation

```
# Macro file for seg06.cc
# set detector parameters
# This line fills some space
# This line fills some space
/run/beamOn 2
```

References

1. A. Aktas *et al.* [H1 Collaboration], Submitted to Eur. Phys. J. **C**, [hep-ex/0401010].