SpdRoot: simulation and analysis package for Spin Physics Detector (SPD).

Version 0.0.1

October 2018, JINR, Dubna

Contents

1 Introduction

Measurements of asymmetries in the lepton pair (Drell-Yan) production in collisions of nonpolarized, longitudinally and transversally polarized protons and deuterons beams are suggested to be performed at the collider NICA of the JINR using the specialized Spin Physics Detector (SPD). These measurements can provide an access to all leading twist collinear and Transverse-Momentum Dependent ditsribution functions of quarks and antiquarks in nucleons. The measurements of asymmetries in production of J/Ψ and direct photons, which supply complimentary information on the nucleon structure, will be performed simultaneously with Drell-Yan data using dedicated triggers. The set of these measurements permits to tests the quark-parton model of nucleons at the QCD twist-2 level with minimal systematic errors. Physics motivations and possible layout of SPD will be presented as well as the plans for polarized beams at NICA.

2 How to install

Before SpdRoot installation, the following packages need to be installed

- FairSoft.
- FairRoot.

For more detailed information about these packages see: https://fairroot.gsi.de

2.1 FairSoft

The FairSoft package installs all necessary external (such as ROOT, Geant4, Pythia, CLHEP etc.) packages in the correct order and with the right compilation flags. FairSoft also contains a configuration scripts which check if all the needed system packages are installed. If some system packages are missing, the configuration script will stop with a detailed error message. In the end all additional software is installed in one directory.

Current version of the FairSoft package can be downloaded from:

git clone https://github.com/FairRootGroup/FairSoft.git FairSoft

Let's **/work/FairSoft** - full path to FairSoft package.

Then

1. System packages installation.
   The complete list of needed system packages can be found in the file **/work/FairSoft/DEPENDENCIES**.
   This file also contains complete command lines to install the necessary packages on the most common Linux systems.

2. Create installation directory, for example:
   mkdir /work/fairsoft_install

3. cd /work/FairSoft
   ./configure

   An example of the answer you can see in the file:
   /spdroot/doc/fairsoft_configure.txt

   Result of successful installation is the message:

   ***** End installation of external packages without errors *****

   As a result of the installation, the following folders should appear (minimally) in the fairsoft_install directory:

   ```
   /work/fairroot_install/
                          bin/
                          include/
                          lib/
                          share/
   ```

4. Make link to the FairSoft installation directory:
   cd /work
   ln -s fairsoft_install fairsoft

Full list of installed packages and its versions can be seen in **/work/FairSoft/Readme.md**.

2.2 FairRoot

The FairRoot framework is an object oriented simulation, reconstruction and data analysis framework. It includes core services for detector simulation and offline analysis of particle physics data. The basic idea of FairRoot is to provide a unified package with generic mechanisms to deal with most commonly used tasks in HEP. FairRoot allows physicists to concentrate on detector performance details, avoiding purely software engineering issues like storage, retrieval, code organization etc.

Current version of the FairRoot package can be downloaded from

git clone https://github.com/FairRootGroup/FairRoot.git FairRoot

Let's **/work/FairRoot** - full path to FairRoot package
and **/work/fairsoft_install** - full path to FairSoft installation directory.

Then

1. Create FairRoot installation directory, for example:
   /work/fairroot_install

2. Set environment variable SIMPATH:
   cd /work
   export SIMPATH=/work/fairsoft_install

3. cd /work/FairRoot
   mkdir build
   cd build
   cmake -DCMAKE_INSTALL_PREFIX= "/work/fairroot_install" ..
   make -j8
   make install

   As a result of the installation, the following folders should appear (minimally) in the fairroot_install directory:

   /work/fairroot_install/
   bin/
   include/
   lib/
   share/

4. Make link fairroot to the fairroot_install installation directory in /work:

   cd /work
   ln -s fairroot_install fairroot

2.3 SpdRoot

SpdRoot is a simulation and analysis software package that is being developed for a future SPD (Spin Physics Detector). At the moment, SpdRoot can be considered as an extended adaptation of the FairRoot package to the SPD detector, which, taking into account the installation features and is intended for solving similar task. SpdRoot has the same code structure and directly uses a significant amount of technical capabilities provided by the FairRoot package.

Current version of the SpdRoot package can be downloaded from:

git clone https://git.jinr.ru/Tkachenko/spdroot

Unpack the project in the folder spdroot/

cd spdroot/

1. Check the contents of the SetEnv.sh:
   cat /spdroot/SetEnv.sh
   If the paths in the file are incorrect, it is necessary to correct:

   ```
   #!/bin/bash
   export SIMPATH=/work/fairsoft_install
   export FAIRROOTPATH=/work/fairroot_install
   source build/config.sh
   ```

   (do not edit line "source build/config.sh")

2. cd spdroot/

   . SetEnv.sh

   (ignore warning "build/config.sh: no such file or directory")

   mkdir build

   cd build

   cmake ..

   make -jN (N is the number of computer processors)

2.4 Create HTML documentation

It is possible to create html documentation with list of SpdRoot classes.

Prerequirements: Doxygen package (debian, ubuntu: sudo apt-get install doxygen)

1. Switch ON "BUILD_DOXYGEN" option at the end of file spdroot/CMakeLists.txt :
   Option(BUILD_DOXYGEN
   "Whether to generate automatic documentation with Doxygen "
   OFF)
2. Remove (or clear) build/ directory:
   rm -rf build/
3. Build spdroot:
   mkdir build
   cd build/
   cmake ..
   make
4. Make html:
   make html-doc

Find html-documentation in spdroot/build/doc/html/classes.html .

3 General notes and package structure

In SpdRoot there are three setups (geometries):

- Solenoidal;
- Toroidal;
- Hybrid (it is similar to toroidal, but there are some differences in magnet and tracking system construction).

Each type of geometry has its own type of magnetic field.

General package structure is presented in the Table 1 :

1st level folders	2nd level folders	Short description
ts[s,t]/	ecps/ barrel/	Tracking system: endcaps and barrel (code)
ecal[s,t]/	ecps/ barrel/	Electromagnetic calorimeter: endcaps and barrel (code)
rs[s,t]/	ecps/ barrel/	Range system: endcaps and barrel (code)
geometry/		Media and geometry ascii-files
input/		Magnetic field maps
gconfig/		Configuration files and scrips for MC-application
spdgenerators/		Primary generators (code)
field/		Magnetic field (code)
spddata/		Output: data, parameters, geometry navigation (code)
common/		Common usage classes, geometry regions (code)
passive/		Passive geometry modules (code)
spddisplay/		Visualization, event viewers (code)
test/		Experimental (code)
doc/		Documentation
tools/		Utilities for development (shell scripts)
scripts/		Run scripts
build/		Project build directory

Table 1: General package structure.

4 How to run

Before you get started, you need to run a bash script which sets the necessary environment variables and is located in the spdroot/ folder:

```
cd spdroot/
. SetEnv.sh
```

Important note! Setting environment variables with SetEnv.sh is necessary each time before starting work.

Then in macro/ folder:

```
cd macro
root -l
```

The macro/ folder contains a number of root scripts with which you can simulate an event (or a series of events) in the detector. All simulation scripts have **Simu** in the name and can take the number of events as a parameter. For example:

```
root [0] .x SimuHyb.C          (to simulate a single event)
root [0] .x SimuHyb.C(5)       (to simulate five events)
```

The words **Hyb**, **Tor** or **Sol** in the script name show the choice for simulation one of the available geometric configurations setup: hybrid, toroidal or solenoidal, and its magnetic field.

Important note! In fact, **Simu** scripts are configuration files with which the installation geometry and a number of simulation parameters are selected. Some simulation parameters that can be specified in the **Simu** scripts are described in 5.

The result of the simulation script is two files: data file **run*.root** (tracks, hits, installation geometry, etc.) and file with parameters, **param*.root** (geometry, magnetic field, ...):

```
    run_tor.root    param_tor.root      (for hybrid and toroidal geometry)
or
    run_sol.root    param_sol.root      (for solenoidal geometry)
```

To view the content of the data file, you can use one of the scripts - the viewer. As a parameter, scripts can take the number of the event that should be displayed after the program starts. By default, the first event in the file will be shown (if the option of saving geometric tracks was enabled, if not, only the geometry will be shown). Event number starts from zero:

```
root [0] .x DisplayTorEvent.C       (the first event from run_tor.root)
root [0] .x DisplaySolEvent.C(3)    (the first event from file fifth run_sol.root)
```

For more detailed information about the programms from the macro/ directory see the section 7.

5 Input settings

The list of main settings for simulation which is available in the **Simu**-scripts (see macro/ directory):

- **Number of events to be generated**.

  The number of events is specified as an argument of the main function and can be set when macro is run. See, for example, the section 4 (by default, just one event is simulated).

- **Output files**.

  The result of the simulation of **Simu**-scripts is two output files: with 1) data and 2) simulation parameters. These files have "root" format. The names of these files can be set in the root-script:

  outFile = "run_tor.root"
  parFile = "params_tor.root"

  It should be noted, when analyzing scripts and visualization programs start from the macro/ directory (for example, Display*Event.C), the names of the input files also have to be corrected accordingly.

- **Media file**.

  According to the FairRoot rules, the file with the description of materials necessary for building the setup should be located in the /spdroot/geometry folder. The standard file is called **media.geo**. You can put a file with a description of materials into the program using the method:

  FairRunSim :: SetMaterials("media.geo")

  If the file does not contain the necessary material, it can be added by the user. More details can be found: https://fairroot.gsi.de/?q=node/34

- **Transport generator**.

  TGeant4 or TGeant3 can be selected as a transport generator. The default is TGeant4:

  FairRunSim :: SetName("TGeant4")

  For more information about generator settings (for example, choosing the physics list for Geant4), see 8.

- **Set of setup parts: beam pipe, magnet, setup frame, tracking system, electromagnetic calorimeter, range system etc**.

  SpdRoot allows to change the geometric configuration of the setup directly from the **Simu** script. The setup consists of a set of geometric modules (passive and active) that can be added (or removed by commenting out the corresponding line in the script) using

  FairRunSim :: AddModule(module) ,

  where the **module** is a pointer to the object that describes this module and which should be previously created and configured (if necessary).

  The **passive** modules of the setup are a magnet, a beam pipe, supporting structure elements and **CAVE** - the top level geometry module. The **active** modules (tracking system, electromagnetic calorimeter, range system etc), in addition to geometry, also

contains a description of physical information about the simulated events that will be saved in the output file.

More information about the modules is presented in 11, 12 и 13.

- **Global geometry settings**.

  A number of global geometry parameters are available through the methods of the static class **SpdCommonGeoMapper** (see section 11).

- **Primary generator**.

  The primary generator creates a list of particles (typically, spreading from the region of beam-beam interaction - "vertex") for their further transportation through the detector. Several generators can be set in the sumulation at the same time. You can add a generator using the method

  SpdPrimaryGenerator :: AddGenerator(generator) ,

  where **generator** is a pointer to the object that describes this generator and which should be previously created and configured.

  SpdPrimaryGenerator is a standard container class for a list of generators. In the process of forming the list of primary particles, all generators will be called in the same order in which they were added to the container.

  Currently, the main primary vertex generator in SpdRoot is **Pythia6**. A list of available generators, as well as their description, is presented in 9.

- **Magnetic field**.

  The magnetic field is set independently of the setup geometry, despite the fact that its configuration is closely related to the design of the magnet and the distribution of matter in the detector. The SpdRoot magnetic field can be set: a) analytically; b) using the table; c) using an arbitrary combination of the first two options. You can add a field using the method:

  FairRunSim :: AddField(field) ,

  where **field** is a pointer to the object that describes this generator and which should be previously created and configured.

  The field region can be defined in the **Simu** script, i.e. geometrical region, beyond which the value of the magnetic field is zero. The field region can by specified by:

  SpdField :: CreateFieldRegion("region_type") ,

  where **"region_type"** is one of the "box","tube" or "physical".

  For more information about magnetic field and regions, see 10.

- **Other settings**.

  Some other settings are available in the **Simu** script. For example, creating a file with the setup geometry or writing to the output data file the particle tracks (via TGeoTrack's) for the visualization of the event using Display*Event.C:

  FairRunSim :: SetStoreTraj(true)

  Important note! It is strongly recommended to save geotracks only during the running a **Simu** -script with a small number of events due to the large amount of the output data file.

6 Output data

7 Root macros for simulation, visualization, analysis and tests

The **macro**/ directory content:

- a number of macros for simulations and visualization (this section);
- geom/ - setup geometry (and it's parts) testings (sections 12, 13);
- primgen/ - primary vertex generators setting examples and tests (section 9);
- field/ - magnetic field manipulations (section 10);
- analysis/ - scripts for analysis (section 14);
- testmisc/ - miscellaneous programmes.

7.1 Simulation

A number of scripts for a different geometry and magnetic field setups (hybrid, toroidal, solenoidal) are available from the **macro**/ folder:

Simu[Hyb,Tor,Sol].C - standard macros for simulations:

- Usable vertex generator: Pythia6 (global option: 1, minimum bias: MSEL = 1, beam = p+p, energy = 26 GeV in CMS).

XSimu[Hyb,Tor,Sol].C - these programmes is mainly for geometry and magnetic field tests:

- Two primary vertex generators of type IsotropicGenerator are involved;
- Unset materials option is switched on;
- Save geometry into separate root file.

Simu[Hyb,Tor,Sol]ExtDecayer.C - examples of simulations with external decayer:

- Usable vertex generator: Pythia6 (global option: 0, minimum bias: MSEL = 0, J/ψ will be generated in the primary vertex, beam = p+p, energy = 8 TeV in CMS).

7.2 Visualization

Output data visualization programmes (setup geometry and particle tracks) are available from the **macro**/ folder:

- DisplaySolEvents.C - for solenoidal geometry;
- DisplayTorEvents.C - for toroidal and hybrid geometry.

7.3 Tests

Some scripts for tests.

8 Transport generator and some MC application settings

A few classes for simulation with VMC are essentially required:

* TVirtualMC: usually means transport generator (TGeant4, TGeant3, etc).
* TVirtualMCStack: a container with transported particles - MC Stack (SpdStack).
* TVirtualMCApplication: MC task (FairRunSim).

Global settings for transport generator and MC stack are accessible in macros g4Config.C and g3Config.C for Geant4 and Geant3 respectively. The most of MC-application settings should be determined directly in Simu-scripts (see sections 5).

8.1 Geant4

1. Available settings from g4Config.C:

   - physics list selection;
   - physical cuts (by energy) for particles and processes selection in Geant3-like style through the SetCuts.C loading;
   - user settings for Geant4 transport generator by g4config.in loading;
   - MC stack settings.

2. User decayer settings (if needed): DecayConfig.C .

3. Special user decays definition (if needed): UserDecay.C .

8.2 Geant3

It is available but not tuned and tested yet.

9 Primary generators

9.1 Pythia6

9.2 Pythia8

9.3 IsotropicGenerator

10 Magnetic field

10.1 Kind of field

The magnetic field in SpdRoot can be set in one of the following ways:

a) **Analitically (function)**. A field at any point is calculated by a previously described function (or algorithm), executed as a class. The simplest case of an analytically description is a constant field (class **SpdConstField**).

b) **By a table (map)**. The field is set in the nodes of the cubic grid, previously recorded in a file. The field values at an arbitrary point are defined using an inter-node approximation (linear by default). A standard class is used to load the fields of this type **SpdFieldMap**.

c) **Fields set (mulifield)**. An arbitrary complex of the two types of fields described above (as well as the corresponding geometric regions). In this case, the value of the field at the point is defined as the sum of the values of all fields in the set. To load a set of fields of different types, use the class **SpdMultiField**.

10.2 Field regions

For fields of types a) and b), area of action can be defined, i.e., a geometric area beyond which the field value is zero. At the moment there are three possible choices for the field region:

- **Box**. The region has the form of a parallelepiped. For determination, it is necessary to set six parameters - the boundaries of the region along the coordinate axes.
- **Tube**. The region has the form of a cylinder, the axis of which coincides with the coordinate axis Z. For determination, it is necessary to set four parameters - the minimum and maximum radius of the cylinder, and the boundary of the region along the Z axis.
- **Physical**. Special case - the region is defined either by the specified name (more exactly, by volume geopath) of physical volume, or by material.

10.3 Tutorials

Magnetic field tutorial scripts is placed in **spdroot/macro/field** .

11 Setup modules and global geometry settings

The setup configuration is determined by sets of modules. In the narrow sense, the setup module is simply program code containing a description of the geometry of a part of an setup that can be physically and functionally considered as a whole. This understanding is valid for modules defined as **Passive**. In a broad sense, the setup module is a program code consisting of a set of classes that often designed into a separate library in the process of compiling. These classes in addition to describing the geometry, also carry out collection and some primary processing of useful physical information obtained during the simulation format suitable for saving to a file. Modules with such wide functionality are defined as **Active**. Active module may also contain tools for secondary data processing - working with files obtained as a result of simulation. Setup module (both active and passive) can form a list of its parameters during the simulation, which will be added to the output file.

Each module in SpdRoot has its own unique number (integer), which takes positive values ($>$ or $= 0$) for active modules and negative for passive ones.

Detailed description of passive and active modules, which are available for simulation and testings are presented in sections 12 and 13 respectively.

12 Passive modules

Full list of passive modules - unique module id, relevant class to add in **Simu**-script and applicable geometry are presented in the table:

13 Active modules: geometry and getting data

Full list of pactive modules unique module id, main relevant class to add in **Simu**-script and applicable geometry are presented in the table:

14 Analysis

Direct data access for analysis (example):

macro/analysis/

- CheckOutputData.C - searching for simulation parameters, event header, tracks and branches with hits in the output files.