# polematrix

**a spin tracking code for electron accelerators using elegant**

Jan Felix Schmidt <janschmidt@mailbox.org>

documentation for version 1.0

*polematrix* is a spin tracking code. It calculates the motion of the spin of ultra relativistic electrons in particle accelerators. It implements a simple matrix tracking as multi-thread C++ application. The input can be any lattice file from *elegant* or *MAD-X*. All required particle trajectories are also imported from the established particle tracking codes *elegant* or *MAD-X*.

*polematrix* includes synchrotron radiation via import of particle trajectories from *elegant* or via an own implementation of longitudinal phasespace. A short description of *polematrix* is given in the IPAC'16 contribution [Sch+16] and in my phd thesis [Sch17], which will be available soon (in German).

Do not hesitate to contact me if you have any questions and please report bugs.

## Contents

## 1. *polematrix* can

- perform element by element electron spin tracking
- read a lattice from common *elegant* or *MAD-X* files (see appendix B for supported elements)
- import closed orbit, optics parameters and particle trajectories from *elegant* or *MAD-X*
- configure and execute *elegant* (or *MAD-X*) automatically
- track multiple spins in parallel (any number of threads)
- consider synchrotron radiation for beam dynamics:
  - longitudinal and transversal phase space via import from *elegant* (or *MAD-X*)
  - longitudinal phase space via fast own implementation
- simulate linear energy ramps
- configure any magnet's field to oscillate harmonically (including linear frequency sweep)
- simulate depolarizing resonances including synchrotron sidebands
- estimate resonance strengths of depolarizing resonances
- be used and modified for free under terms of the GNU General Public License

A detailed description and discussion of *polematrix* as well as simulation examples can be found in [Sch17].

## 2. *polematrix* cannot

- simulate radiative polarization build-up (Sokolov-Ternov effect)
- track spins of other particles than electrons
- consider electric fields for spin tracking
- perform transversal particle tracking independent of *elegant* and *MAD-X*
- simulate nonlinear energy ramps
- consider a field strength which varies longitudinally within a magnet

Some of these features could be added with little effort. Please contact me if you have any questions.

## 3. Installation

*polematrix* has been tested only on Ubuntu/Debian Linux with the GCC compiler so far.

### 3.1. Dependencies

*polematrix* requires the following programs and libraries:

- GCC compiler
- CMake
- Gnu Scientific Library (GSL) [GNU]
- Boost program options
- Boost filesystem
- Boost property tree
- Boost random
- Armadillo library [SC16]
- *palattice* library [Sch]

*palattice* is available on github. The source code can be downloaded at [Sch] and compiled and installed as described in the enclosed README file. All other dependencies may be in the package repositories of your Linux distribution. On Ubuntu you can install all required packages with the following command:

```
sudo apt-get install g++ cmake libgsl-dev libboost-dev libboost-program-options-dev
 ↪  libboost-filesystem-dev libarmadillo-dev
```

Additionally, *polematrix* relies on the established particle tracking program *elegant* (or *MAD-X*), whose lattice files and tracking results are used for spin tracking. *elegant* (and *MAD-X*) can be configured and execute automatically by *polematrix* using *palattice*. The installation of *elegant* and *MAD-X* is described in section 3.3.

## 3.2. Build and Install *polematrix*

*polematrix* is compiled and installed using *CMake* with the following commands:

```
cd polematrix/build
cmake ../
make
sudo make install
```

The default install path is `/usr/local/bin`. It can be changed by setting the `CMAKE_INSTALL_PREFIX` variable in `polematrix/CMakeLists.txt`. To uninstall *polematrix* run

```
cd polematrix/build
sudo make uninstall
```

## 3.3. Underlying Particle Tracking Programs

I recommend to use *polematrix* with *elegant* since much more tracking parameters can be set automatically – including number of particles (starting with a Gaussian profile in radiation equilibrium), tracking duration and linear energy ramps. If you have only a lattice file in *MAD-X* format, you can convert it automatically to an *elegant* lattice using the program *convertlattice* which is included in *palattice* [Sch].

### 3.3.1. *elegant*

The particle tracking program *elegant* [Bor00] is developed at the Advanced Photon Source at Argonne National Laboratory. Packages for many operating systems can be downloaded at [Arg]. Also install the *SDDSToolKit*, which is needed to access the binary output files of *elegant*. If there is no package for your system, you can download the `Build-AOP-RPMs` script instead which automatically builds all desired programs from source (see the guide in appendix A). There is a parallel version of *elegant*, called *Pelegant*, which speeds up the tracking [WB06] (installation hints in appendix A). The *elegant* manual can be found at [Bor17].

### 3.3.2. *MAD-X*

The particle tracking program *MAD-X* is developed at CERN and can be downloaded from the website [CER], which also provides the *MAD-X* manual.

# 4. Usage

## 4.1. Execution

*polematrix* is called via

```
polematrix [options] [CONFIGURATION FILE]
```

where `[options]` are the command line options shown in table 1. All parameters of the spin tracking can be set up in an *xml* configuration file (`[CONFIGURATION FILE]`) which is described in section 7. The lattice of the accelerator is given as any conventional *elegant* or *MAD-X* lattice file and is set in the configuration file (see section 7). A template configuration file with default values can be generated anytime by executing `polematrix --template`.

| | |
|---|---|
| `-h [ --help ]` | display this help message |
| `-V [ --version ]` | display version |
| `-T [ --template ]` | create config file template (`template.pole`) and quit |
| `-R [ --resonance-strengths ]` | estimate resonance strengths instead of spin tracking |
| `-t [ --threads ] arg (=all)` | set number of threads used for tracking |
| `-o [ --output-path ] arg (=.)` | path for output files |
| `-v [ --verbose ]` | activate additional status output |
| `-n [ --no-progressbar ]` | do not show progress bar during tracking (e.g. if output is redirected to a log file) |
| `-a [ --all ]` | write additional output files (e.g. lattice und orbit) |
| `-s [ --spintune ] arg` | in resonance strengths mode (`-R`): calculate for given spin tune only |

Table 1: Command line options of *polematrix*. They consist of special program modes in the upper part and configuration options in the lower part.

## 4.2. Output

The components of each spin vector $\vec{S}_i(t)$ is written to the text file `spins/spin_i.dat` in the output path during tracking. The output step width can be set in the configuration file (see section 7.1). When tracking is completed for all spins, the polarization vector $\vec{P}(t)$ is calculated for each output step as average over all successfully tracked spins. It is saved as `polarization.dat` in the output path.

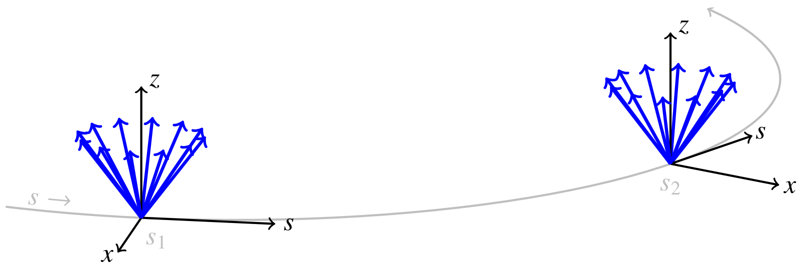

Figure 1: Accelerator coordinate system and precessing spin vectors. [Sch17]

## 5. Coordinate System and Polarization

*polematrix* uses the typical accelerator coordinate system sketched in fig. 1, which is moving along the reference orbit. The position along the reference orbit is called $s$. Since *polematrix* assumes ultra relativistic electrons, the velocity along $s$ is assumed to be constant $v \approx c$ and the position is alternatively parametrized by the time $t = s/c$. $t$ is used as parameter for multiple turns in a circular accelerator and to define the duration and output step width of the spin tracking. The output can be fixed to the position of a certain element in the accelerator using the entry `<outElement>` in the configuration file (see section 7).

At each position $s$ the $x$ axis points in horizontal direction and the $z$ axis points in vertical direction. $z > 0$ is above the reference orbit. In a circular electron accelerator $x > 0$ corresponds to the outer side of the ring (to larger radii of the bending magnets). The longitudinal axis is also referred to as $s$ axis.

Accordingly, the tracked spin vector $\vec{S}_i(t)$ of electron $i$ is parametrized by a horizontal component $S_x$, vertical component $S_z$ and longitudinal component $S_s$. Since the spin naturally "moves" with the corresponding electron, $\vec{S}_i(t)$ only includes the spin orientation at the current spacial position of the electron. The absolute value is always $|\vec{S}_i| = 1$. The same coordinate system is used for the resulting polarization vector

$$\vec{P}(t) = \frac{1}{N} \sum_{i=0}^{N-1} \vec{S}_i(t) \quad .$$

Its absolute value $0 \leq |\vec{P}| \leq 1$ is the polarization degree. The projections $P_x$, $P_z$ and $P_s$ can be interpreted as probabilities to measure spins oriented in parallel to the corresponding axis. The $N$ tracked spins are labeled with $i \in \{0, 1, \dots, N-1\}$.

## 6. Brief Introduction to the Concept of *polematrix*

### 6.1. Spin Tracking and Magnetic Fields

Spin precession of a relativistic electron in a particle accelerator can be described in the laboratory frame by the THOMAS-BMT equation [Tho27; BMT59]

$$\begin{aligned} \frac{\mathrm{d}\vec{S}}{\mathrm{d}t} &= \vec{\Omega}_{\mathrm{TBMT}} \times \vec{S} \\ \text{with} \quad \vec{\Omega}_{\mathrm{TBMT}} &:= -\frac{e}{\gamma m} \left[ (1+\gamma a)\vec{B}_\perp + (1+a)\vec{B}_\parallel - \left( \frac{\gamma}{1+\gamma} + \gamma a \right) \vec{\beta} \times \frac{\vec{E}}{c} \right] \quad . \end{aligned} \tag{1}$$

Here, $e$, $m$ and $a = (g_s - 2)/2$ are charge, rest mass and gyromagnetic anomaly of an electron. $\gamma$ is the LORENTZ factor and $\vec{B}$ and $\vec{E}$ are the magnetic and electric field at the position of the electron, where the components of $\vec{B}$ have to be distinguished with respect to the direction of motion of the electron. Neglecting transversal electric fields and normalizing the magnetic fields to the magnetic rigidity as

$$\vec{\mathcal{B}} := \frac{e}{p_0}\vec{B} \quad , \tag{2}$$

the THOMAS-BMT equation can be simplified to

$$\frac{\mathrm{d}\vec{S}}{\mathrm{d}t} \approx -c \cdot \left[(1+\gamma a)\vec{\mathcal{B}}_\perp + (1+a)\vec{\mathcal{B}}_\parallel\right] \times \vec{S} \quad . \tag{3}$$

This equation describes a precession of $\vec{S}$ around any magnetic field vector. The precession frequency around a vertical field $B_z^{\mathrm{dip}}$ of a bending magnet is

$$\Omega_{\mathrm{TBMT},z} = -\frac{e}{\gamma m}(1+\gamma a)B_z^{\mathrm{dip}} = (1+\gamma a)\omega_{\mathrm{rev}} \tag{4}$$

with the revolution frequency in a circular accelerator

$$\omega_{\mathrm{rev}} = -\frac{e}{\gamma m}B_z^{\mathrm{dip}} \quad .$$

Now the revolution frequency $\omega_{\mathrm{rev}}$ is subtracted from the precession frequency to transform eq. (3) from the laboratory frame to the co-moving accelerator coordinate system introduced in section 5, which rotates in the lab frame once per revolution (as the particle's momentum). This yields the equation

$$\frac{\mathrm{d}\vec{S}(t)}{\mathrm{d}t} \approx -c \cdot \left[\gamma(t)a\vec{\mathcal{B}}_\perp(t) + a\vec{\mathcal{B}}_\parallel(t)\right] \times \vec{S}(t) \quad , \tag{5}$$

where all time dependencies are now given explicitly. Both $\gamma(t)$ and $\vec{B}(t)$ depend on the time $t$:

- In case of the magnetic field $\vec{B}(t)$, $t$ parameterizes two dependencies. First, the field depends on the position $s$ in the lattice; second, for some element types it depends on the transversal position $(x, z)$ of the electron relative to the reference orbit. The element positions and parameters like bending radius $R$, quadrupole strength $k_1$ and sextupole strength $k_2$ are imported from the *elegant* or *MAD-X* lattice file. The formula for field calculation is given in section 6.2. The transversal trajectories $x(t)$ and $z(t)$ can be considered by different models described in section 6.3.

- The LORENTZ factor $\gamma(t)$ is proportional to the particle's energy and therefore depends on its longitudinal dynamics, which can be considered by different models described in section 6.4.

Equation (5) describes a three dimensional rotation of $\vec{S}$ around $\vec{\mathcal{B}}$ with variable speed. *polematrix* models this spin motion by three dimensional rotation matrices using the following assumptions/approximations:

- The spin does not precess in-between the magnets because of $\vec{\mathcal{B}} = 0$ in the drift spaces.

- The magnetic field of an element does not change along the particles path within the element.

- The particle's angle to the reference orbit can be neglected and $\vec{B}_\parallel$ can be set to $\vec{B}_s$.

Hence, the spin rotation can be tracked element by element. This means, the spin precession from a position $s_1$ in front of an element with reference position $s_e$ to a position $s_2$ behind the element is calculated as

$$\vec{S}(s_2) = \mathbf{R}_{\hat{r}}(\vartheta) \cdot \vec{S}(s_1) \tag{6}$$

with the general three dimensional rotation matrix

$$\mathbf{R}_{\hat{r}}(\vartheta) = \begin{pmatrix} \hat{r}_x^2\,(1-\cos\vartheta)+\cos\vartheta & \hat{r}_x\hat{r}_s\,(1-\cos\vartheta)-\hat{r}_z\sin\vartheta & \hat{r}_x\hat{r}_z\,(1-\cos\vartheta)+\hat{r}_s\sin\vartheta \\ \hat{r}_s\hat{r}_x\,(1-\cos\vartheta)+\hat{r}_z\sin\vartheta & \hat{r}_s^2\,(1-\cos\vartheta)+\cos\vartheta & \hat{r}_s\hat{r}_z\,(1-\cos\vartheta)-\hat{r}_x\sin\vartheta \\ \hat{r}_z\hat{r}_x\,(1-\cos\vartheta)-\hat{r}_s\sin\vartheta & \hat{r}_z\hat{r}_s\,(1-\cos\vartheta)+\hat{r}_x\sin\vartheta & \hat{r}_z^2\,(1-\cos\vartheta)+\cos\vartheta \end{pmatrix}$$

for a rotation with the angle $\vartheta$ around the axis $\hat{r}$. Rotation angle and axis are given by eq. (5) and the integrated field of the element with effective length $l_{\text{eff}}$:

$$\begin{aligned} \vec{\vartheta}_{x,z} &= \gamma(t)a\int_{s_1}^{s_2}\vec{\mathcal{B}}_{x,z}\mathrm{d}s \approx \gamma a\cdot\vec{\mathcal{B}}_{x,z}(x(t),z(t))\cdot l_{\text{eff}} \\ \text{and}\quad \vec{\vartheta}_s &= a\int_{s_1}^{s_2}\vec{\mathcal{B}}_s\mathrm{d}s \approx a\cdot\vec{\mathcal{B}}_s(x(t),z(t))\cdot l_{\text{eff}} \quad . \end{aligned}$$

The rotation angle is $\vartheta = |\vec{\vartheta}|$ and the rotation axis is $\hat{r} = \vec{\vartheta}/\vartheta$. The effective length $l_{\text{eff}}$ is taken from the *elegant* or *MAD-X* lattice. The LORENTZ factor $\gamma(t)$ and the trajectories $x(t)$ and $z(t)$ are inserted for the time $t$ corresponding to the current tracking step.

## 6.2. Magnetic Fields

The magnetic field $\vec{\mathcal{B}}(x,z)$ of an element is calculated using the library *palattice*. Currently, it supports elements of all types listed in appendix B. The field vector is

$$\begin{aligned} \mathcal{B}_x(x,z) &= k_{0,x} \;+\; k_1\cdot z \;+\; k_2\cdot x\cdot z \quad , \\ \mathcal{B}_z(x,z) &= k_{0,z} \;+\; k_1\cdot x \;+\; k_2\cdot\frac{x^2-z^2}{2} \quad , \\ \mathcal{B}_s(x,z) &= k_{0,s} \quad . \end{aligned}$$

All parameters are imported from the *elegant* or *MAD-X* lattice automatically by *palattice* using the *elegant*/*MAD-X* parameters listed in table 3.

**Misalignments** Currently, there are two kinds of magnet misalignments implemented in *palattice* and thus supported by *polematrix*:

- transversal displacements $\Delta x$ and $\Delta z$
- rotations around the beam axis $s$

Both types are imported automatically from *elegant* and *MAD-X* respecting the particular sign conventions of the particle tracking program. The corresponding *elegant*/*MAD-X* parameters are also listed in table 3.

## 6.3. Transversal Phase Space

Currently, there are three models of transversal trajectories $x(t)$ and $z(t)$ in *polematrix*, which can be selected with the configuration file entry `<trajectoryModel>` (see section 7.1):

- `closed orbit`: all particles move along the closed orbit imported from *elegant* or *MAD-X*
- `simtool`: individual trajectories imported from *elegant* or *MAD-X* particle tracking for each particle
- `oscillation`: simplified individual trajectories as oscillations

**closed orbit** Modeling transversal phase space simply by the closed orbit for all particles is sufficient when simulating integer depolarizing resonances, because betatron tunes are never integer and thus betatron motion does not contribute to integer resonances.

**simtool** Using realistic individual trajectories requires particle tracking over the full period requested for spin tracking. I recommend to use the parallel version of *elegant* and to compile *palattice* with *libSDDS1* support to allow *palattice* reading binary SDDS files without converting them to ascii. Nevertheless a lot of RAM is recommended since the trajectories are stored in memory completely by *libSDDS1*.

**oscillation** For simulations of intrinsic resonances without particle tracking, there is the simplified `<trajectoryModel>` `oscillation`. It calculates the betatron oscillation of particle $i$ as

$$x_i(t) = x_{\text{orbit}}(t) + \sqrt{\epsilon_{x,i}\beta_x} \cdot \cos\left(\omega_x t + \psi_{x,i}\right) \qquad \text{with} \quad \omega_x = \frac{2\pi Q_x}{T_{\text{rev}}} \tag{7}$$

with the imported closed orbit $x_{\text{orbit}}$, the single particle emittance $\epsilon_{x,i}$, the beta function $\beta_x$, the start phase $\psi_{x,i}$, the tune $Q_x$ and the revolution time $T_{\text{rev}}$. The same equation is used for the vertical trajectory. Beta functions, tunes and revolution time are imported automatically from the particle tracking program. The start phases are chosen randomly (uniform distribution $[0, 2\pi]$). The beam emittances $\epsilon_x$ and $\epsilon_z$ have to be set in the configuration file. Based on those, the single particle emittances are determined randomly (Gaussian distribution around zero with $\sigma_x = \epsilon_x$).

## 6.4. Longitudinal Phase Space

Currently, there are two realistic models of longitudinal phase space in *polematrix*, which can be selected with the configuration file entry `<gammaModel>` (see section 7.1):

- `radiation`: implementation of stochastic synchrotron radiation within *polematrix*
- `simtool`: $\gamma_i(t)$ imported from *elegant* or *MAD-X* particle tracking

Furthermore, there are some simplified models of $\gamma(t)$ – see section 7.1.

**radiation** This model is described in detail in [Sch17, section 4.2]. It calculates the energy loss by radiating photons in dipole magnets based on the correct statistical distributions of photon energy and photon number. The phase advance along the accelerator is not tracked via the time of flight (dispersion), but approximated using the momentum compaction factor (first and second order).

**simtool** The $\gamma_i(t)$ from *elegant* (or *MAD-X*) particle tracking may be used considering the same advises given above for the transversal trajectories.

### 6.4.1. Linear Energy Ramp

A linear energy ramp can be set up in the *polematrix* configuration file. When using `<gammaModel>` "simtool" with *elegant*, the ramp is automatically transmitted to *elegant* before starting the particle tracking. With *MAD-X* it has to be set up manually in advance. For all other models the linear ramp is added directly to the $\gamma_i(t)$.

# 7. Configuration File Reference

*polematrix* is configured with an *xml* file. The individual entries are arranged in several thematic groups like this:

```
<groupA>
  <entry1> value </entry1>
  <entry2> value </entry2>
</groupA>
<groupB>
  <entry42> value </entry42>
</groupB>
```

The order of groups and entries is arbitrary. Many entries have default values and must not be set. When *polematrix* is executed, a configuration file with the actual values of all entries is saved as `currentconfig.pole`. All entries are documented in the following.

## 7.1. spintracking

The group `<spintracking>` contains the setup of the spin tracking itself.

`<t_start>` Type: `double` Unit: s Default Value: 0.0

The start time $t_{\text{start}}$ of the spin tracking

`<t_stop>` Type: `double` Unit: s

The stop time $t_{\text{stop}}$ of the spin tracking

`<E0>` Type: `double` Unit: GeV

Beam energy at the time $t = 0\,\text{s}$

`<dE>` Type: `double` Unit: $\text{GeV}\,\text{s}^{-1}$

Speed of the linear energy ramp

`<Emax>` Type: `double` Unit: GeV Default Value: $10^{10}$

Maximum energy to confine the energy ramp. If the energy ramp reaches this energy, it is kept until the end of the tracking. Use a sufficiently large number do disable this feature (default).

`<s_start>` Group

Initial value of the spin vector $\vec{S}(t_{\text{start}})$. It is used for all spins. Thus, the tracking always starts with polarization $|\vec{P}|(t_{\text{start}}) = 1$. The three components of the vector are each set as a separate entry:

`<x>` Type: `double` Unit: Default Value: 0.0

Horizontal component $S_x$ of the initial value

`<z>` Type: `double` Unit: Default Value: 1.0

Vertical component $S_z$ of the initial value

`<s>` Type: `double` Unit: Default Value: 0.0

Longitudinal component $S_s$ of the initial value

`<numParticles>` Type: `unsigned int` Unit: Default Value: 1

Number of particles (spin vectors) for the spin tracking

`<dt_out>` Type: `double` Unit: s Default Value: $(t_{\text{stop}} - t_{\text{start}})/1000$

Step width of the output of $\vec{P}$ and $\vec{S}_i$

`<outElement>` Type: `string` Unit:

Name of a specific element in the lattice. If it is given, the output of $\vec{P}$ and $\vec{S}_i$ is printed only when passing this element. Thereby, the polarization can be observed at a specific position in a circular accelerator (e.g. a detector or extraction device). The output is written at the next passage after the time $t$ has increased by `<dt_out>`.

`<gammaModel>` Type: `string` Unit: Default Value: radiation

Model for longitudinal beam dynamics $\gamma_i(t)$ during spin tracking. The two realistic and three simple models are discussed in [Sch17, chapter 4].

`radiation` This model is implemented in *polematrix* and calculates stochastic emission of synchrotron radiation.

`simtool` $\gamma_i(t)$ is imported from the particle tracking program, which is selected as `<simTool>` in the group `<palattice>` (see below). I recommend to use this model with *elegant*.

`linear` Deactivation of longitudinal dynamics. All particles just follow the linear energy ramp: $\gamma_i(t) = \gamma_0$.

`simtool+linear` Sum of the two options above: The linear energy ramp is added to the imported $\gamma_i(t)$, which have been calculated for a constant energy.

`simtool_no_interpolation` Similar to `simtool`, but faster and less precise. The recorded trajectory at the closest quadrupole is used directly instead of interpolating it to the current element position.

`offset` Every particle gets an energy offset, which is constant over time: $\gamma_i(t) = \gamma_0 + \Delta\gamma_i$.

`oscillation` Approximation of longitudinal dynamics by harmonic oscillations: $\gamma_i(t) = \gamma_0 + \Delta\gamma_i \cos(\omega_s t + \psi_i)$

`<trajectoryModel>` Type: `string` Unit: Default Value: closed orbit

Model for transverse particle trajectories $x_i(t)$ and $z_i(t)$ during spin tracking. To date, three models are available:

`closed orbit` The closed orbit is used as trajectory for all particles. Thereby, all spins experience the same magnetic fields, which exclusively consist of revolution harmonic contributions (apart from explicitly time dependent fields). The closed orbit can be determined without particle tracking. This model is sufficient if the simulation of intrinsic resonances is not necessary.

`simtool` The individual trajectories are imported from the particle tracking program selected as `<simTool>` in the group `<palattice>` below. The particle tracking is executed automatically.

`oscillation` The individual trajectories are simplified as oscillations using tunes, beta functions and emittances as given in eq. (7). The emittances and tunes can be set in the group `<oscillation>` described below.

`<edgeFocussing>` Type: `bool` Unit: Default Value: false

Switch to enable horizontal magnetic fields at the edges of dipole magnets (edge focusing) during the spin tracking. Can be activated with `true` or `1` and deactivated with `false` or `0`.

## 7.2. palattice

The group `<palattice>` contains the configuration of lattice import and particle tracking.

`<simTool>` Type: `string` Unit:

Selection of the simulation tool for lattice import and particle tracking. Possible values: `elegant` or `madx`. I recommend using *elegant* since much more tracking parameters can be set automatically – including number of particles, tracking duration and linear energy ramps. A *MAD-X* lattice can be converted automatically to an *elegant* lattice using *palattice*.

`<mode>` Type: `string` Unit:

Selection of import mode:

`online` The simulation tool is configured and executed automatically. Typically, this mode should be used.

`offline` In this mode the simulation tool is not executed. Instead, existing output files are imported. This can be helpful to avoid repeated execution of a particle tracking or to use *polematrix* if *elegant* and *MAD-X* are not available.

`<file>` Type: `string` Unit:

Lattice file for spin tracking. The file format has to be suitable for the program selected as `<simTool>`. Depending on `<mode>` varying files must be given:

- In `online` mode `<file>` is the lattice file (`.lte` for *elegant*, usually `.madx` for *MAD-X*).
- In `offline` mode `<file>` is the `.param` output file (*elegant*) and the `.twiss` output file (*MAD-X*) respectively.

`<saveGamma>` Type: `string` Unit: Default Value:

This option is meaningful only with `<gammaModel>` `simtool`. It allows to export the $\gamma_i(t)$, which are imported from the particle tracking, to text files. The option contains a list of particle numbers $i$ for the export. The particle numbers are delimited by commas (e.g. `0,2,7`). Additionally, ranges of numbers can be given by hyphens (e.g. `0,3-6,8`). The text files are saved as `gammaSimTool_i.dat` (particle number $i$).

**Attention:** The *elegant* `particleIDs` start at 1. This particle is assigned to particle number 0 in *polematrix*.

`<simToolRamp>` Group

If an *elegant* tracking is executed, the energy ramp configured in the group `<spintracking>` can be set for the particle tracking automatically. This is not implemented for *MAD-X*.

`<set>` Type: `bool` Unit: Default Value: true

Switch for the energy ramp transfer to *elegant*. Can be activated with `true` or `1` and deactivated with `false` or `0`.

`<steps>` Type: `unsigned int` Unit: Default Value: 200

If the transfer is active, the ramp is calculated at discrete sampling points, which are then written to a *SDDS* file. Here, the number of sampling points can be changed. They are distributed equidistant from $t_{\text{start}}$ to $t_{\text{stop}}$.

`<rfMagnets>` GROUP

These options can be used to configure the magnetic field $B$ of any lattice element as a time dependent alternating field (rf field)

$$B(T) = B \cdot \cos\left(2\pi\left[Q_{\mathrm{rf},1}T + \frac{1}{2}\Delta Q_{\mathrm{rf}}T^2\right]\right)$$

as a function of the turn $T$. The setup only affects the spin tracking and not the particle tracking with *elegant* or *MAD-X*. In addition, the field has to be set up in *elegant*/*MAD-X*, if the influence of an rf field on beam dynamics should be included.

An arbitrary number of elements can be set up with an rf field since all following options allow for giving multiple values as a comma separated list. All options must have the same number of entries, which are then assigned to the corresponding elements in `<elements>`.

`<elements>` TYPE: `string` UNIT: DEFAULT VALUE:

Name of the element for rf field. [multiple elements as comma separated list]

`<Q1>` TYPE: `string` UNIT: DEFAULT VALUE:

Frequency of the rf field at the beginning of the spin tracking (turn $T = 1$) normalized to the revolution frequency (tune $Q_{\mathrm{rf},1}$). [for multiple elements as comma separated list]

`<dQ>` TYPE: `string` UNIT: DEFAULT VALUE:

Frequency change per turn (frequency sweep) – also as a tune $\Delta Q_{\mathrm{rf}}$. [for multiple elements as comma separated list]

`<period>` TYPE: `string` UNIT: DEFAULT VALUE:

Duration of the frequency sweep in turns. After this period the sweep starts again. [for multiple elements as comma separated list]

## 7.3. radiation

This group is for configuration of the `<gammaModel>` `radiation`, which is a model of longitudinal dynamics implemented in *polematrix* to allow for a stochastic synchrotron radiation model without import from a particle tracking program. If another `<gammaModel>` is selected, the whole group can be ignored.

`<seed>` TYPE: `int` UNIT: DEFAULT VALUE: *random*

The seed of the random number generator, which is used for the initial particle distribution in phase space and the stochastic photon emission.

`<savePhaseSpace>` GROUP

Here, particle numbers $i$ can be chosen for a plain text export of longitudinal phase space $(\phi, \gamma)$ – analog to `<saveGamma>` in group `<palattice>`.

`<list>` TYPE: `string` UNIT: DEFAULT VALUE:

A list of particle numbers $i$ for the export. The particle numbers are delimited by commas (e.g. `0,2,7`). Additionally, ranges of numbers can be given by hyphens (e.g. `0,3-6,8`). The text files are saved as `longPhaseSpace_i.dat` (particle number $i$).

`<elementName>` TYPE: `string` UNIT: DEFAULT VALUE:

The name of an element in the lattice. The phase space coordinates are recorded at the position of this element. This option must be set to use `<savePhaseSpace>`. Else no output is written.

`<startDistribution>` GROUP

*polematrix* calculates the initial phase space coordinates of all particles so that the $\phi_i$ and the $\gamma_i$ are Gaussian distributed and average and width of the distribution correspond to the radiation equilibrium. The following two options can be used to vary the width of these distributions.

`<sigmaPhaseFactor>` TYPE: `double` UNIT: DEFAULT VALUE: 1.0

Factor to scale the width of the Gaussian distribution of the phases $\phi_i$. A value of 1 corresponds to the radiation equilibrium.

`<sigmaGammaFactor>` TYPE: `double` UNIT: DEFAULT VALUE: 1.0

Factor to scale the width of the Gaussian distribution of the particle energies $\gamma_i$. A value of 1 corresponds to the radiation equilibrium.

The following options are physics parameters of the accelerator, which are required by the synchrotron radiation model. They are determined by the particle tracking program and imported automatically if the corresponding option is not set here (default value 0.0). Thus, these options can be used to enforce other values.

`<momentum_compaction_factor>` TYPE: `double` UNIT: DEFAULT VALUE: 0.0

the momentum compaction factor $\alpha_c$

`<momentum_compaction_factor_2>` TYPE: `double` UNIT: DEFAULT VALUE: 0.0

the second order momentum compaction factor $\alpha_c^{(2)}$

`<overvoltage_factor>` TYPE: `double` UNIT: DEFAULT VALUE: 0.0

the overvoltage factor $q$

`<harmonic_number>` TYPE: `unsigned int` UNIT: DEFAULT VALUE: 0

the harmonic number $h$

`<bending_radius>` TYPE: `double` UNIT: m DEFAULT VALUE: 0.0

the average bending radius $R$ of the dipole magnets

`<longitudinal_damping_partition_number>` TYPE: `double` UNIT: DEFAULT VALUE: 0.0

the longitudinal damping partition number $J_s$

## 7.4. oscillation

This group is for configuration of the `<trajectoryModel>` `oscillation`, which is described in section 6.3. If another `<trajectoryModel>` is selected, the whole group can be ignored.

`<emittance>` GROUP

The amplitude of the betatron oscillations depends on the beam emittance, which has to be set here. It is not imported from *elegant* or *MAD-X*.

`<x>` TYPE: `double` UNIT: m rad

horizontal emittance $\epsilon_x$

`<z>` TYPE: `double` UNIT: m rad

vertical emittance $\epsilon_z$

`<tune>` GROUP

The frequency of the betatron oscillations is given by the tunes. They are determined by the particle tracking program and imported automatically if the corresponding option is not set here (default value 0.0). Thus, these options can be used to enforce other values.

`<x>` TYPE: `double` UNIT: m rad DEFAULT VALUE: 0.0

horizontal tune $Q_x$

`<z>` TYPE: `double` UNIT: m rad DEFAULT VALUE: 0.0

vertical tune $Q_z$

## 7.5. resonancestrengths

This group is for configuration of the estimation of resonance strengths of depolarizing resonances. This special mode can be activated by the command line option `-R`. If *polematrix* is used for spin tracking, the whole group can be omitted.

`<spintune>` GROUP

The resonance strength is calculated for discrete values of the spin tune $\gamma a$ and written to the text file `resonance-strengths.dat`. Here, the output range and step width can be set.

`<min>` TYPE: `double` UNIT: DEFAULT VALUE: 0.0

minimum spin tune $\gamma a$

`<max>` TYPE: `double` UNIT: DEFAULT VALUE: 10.0

maximum spin tune $\gamma a$

`<step>` TYPE: `double` UNIT: DEFAULT VALUE: 1.0

step width $\Delta\gamma a$ for calculation and output of the spin tune

`<turns>` TYPE: `unsigned int` UNIT: DEFAULT VALUE: 0

If the resonance strengths are calculated for non integer $\gamma a$, multiple particles with their individual trajectories have to be taken into account and the resulting resonance strength is the average over all particles. For this purpose, the trajectories are imported from the particle tracking program as it is configured in the group `<palattice>`. The number of particles is taken from the option `<numParticles>` (Gruppe `<spintracking>`).

Here the number of turns $N_u$ for the trajectories can bes set. If the default value 0 is used, the number of turns is set automatically according to the minimum number required for the chosen frequency resolution `<step>`: $N_u = 1/\Delta\gamma a$.

# A. *elegant* Installation Guide

*elegant* packages for many operating systems can be downloaded at [Arg]. Additionally you have to download the `defns.rpn` file and store its path in the environment variable `RPN_DEFNS`. E.g. add the following line to your `~/.bashrc` file:

```
export RPN_DEFNS = '/path/to/defns.rpn'
```

If there is no *elegant* package for your system, you can download the *Build-AOP-RPMs* script instead which automatically builds all desired programs from source. If you want to use *Pelegant*, first install *mpicc*. Under Ubuntu it is in the package `mpi-default-dev`. Now go to the folder with the *Build-AOP-RPMs* script and run

```
sudo ./Build-AOP_RPMs
```

Accept the programs you want to install by hitting `y` and decline all others with `n`. For *palattice* and *polematrix* you need

- SDDS (Not SDDSEpics)
- elegant
- (Pelegant recommended)

For *Pelegant* the script asks for the path of *mpicc*. You can determine it by calling `which mpicc`. The script expects the directory (without `mpicc` at the end). The scripts downloads the source code of all programs, builds them and creates rpm packages. This will take some time. Finally, these packages can be found in `~/rpmbuild/RPMs/` (additional subfolder depending on the architecture). Install each package. Under Ubuntu rpm package files can be installed via

```
sudo alien -i packagename.rpm
```

*Pelegant* is called via `mpirun -n 4 Pelegant`, where the argument of the option `-n` is the number of threads used for tracking. To use *Pelegant* for the automatic *elegant* execution by *polematrix*, you have to modify the *elegant* command in the `config.hpp` file of *palattice* and build and install *palattice* again (see *palattice* README for details).

## B. Supported *elegant*/*MAD-X* Element Types and Parameters

| *palattice* | *elegant* | *MAD-X* |
|---|---|---|
| Dipole | CSBEND | SBEND |
| Corrector | (H,V)KICK oder KICKER | (H,V)KICKER, |
| Solenoid | SOLE | SOLENOID |
| Quadrupole | KQUAD | QUADRUPOLE |
| Sextupole | KSEXT | SEXTUPOLE |
| Multipole | MULT | MULTIPOLE |
| Marker | MARK | MARKER |
| Monitor | MONI | MONITOR |
| Cavity | RFCA | RFCAVITY |
| Rcollimator | RCOL | RCOLLIMATOR |
| Drift | DRIF | DRIFT |

Table 2: Assignment of lattice element types in the *palattice* library to the types in *elegant* and *MAD-X*. Other types are ignored during lattice import in *polematrix*.

| *palattice* | *elegant* | *MAD-X* |
|---|---|---|
| name | ElementName | NAME |
| length | L | L |
| k0.x | sin([V]KICK)/L | sin(VKICK)/L |
| k0.z | ANGLE/L - sin([H]KICK)/L | ANGLE/L - sin(HKICK)/L |
| k0.s | KS | KSI/L |
| k1 | K1 | K1L/L |
| k2 | K2 | K2L/L |
| e1 | E1 | E1 |
| e2 | E2 | E2 |
| tilt | TILT | TILT + DPSI |
| displacement.x | DX | DX |
| displacement.z | DY | DY |
| halfWidth.x | X_MAX | APER_1 |
| halfWidth.z | Y_MAX | APER_2 |
| volt | VOLT | VOLT |
| freq | FREQ | FREQ |

Table 3: Assignment of element parameters in *palattice* to parameters in *elegant* and *MAD-X*. Special cases: For *elegant* k0.x and k0.z have to be distinguished between element types KICK (parameters HKICK und VKICK) and HKICK und VKICK (parameter KICK). Aperture is only imported from *MAD-X* if APERTYPE has the value RECTANGLE.

## References

[Arg] Argonne National Laboratory. *Software Download.* URL: www.aps.anl.gov/Accelerator_Systems_Division/Accelerator_Operations_Physics/software.shtml (cit. on pp. 4, 16).

[BMT59] V. Bargmann, L. Michel, and V.L. Telegdi. “Precession of the polarization of particles moving in a homogeneous electromagnetic field”. In: *Physical Review Letters* 2.10 (1959), pp. 435–436. DOI: 10.1103/physrevlett.2.435 (cit. on p. 6).

[Bor00] Michael Borland. *elegant: A Flexible SDDS-Compliant Code for Accelerator Simulation.* Report LS-287. Advanced Photon Source, Sept. 2000. DOI: 10.2172/761286 (cit. on p. 4).

[Bor17] Michael Borland. *User’s Manual for elegant.* Advanced Photon Source. Oct. 2017. URL: www.aps.anl.gov/Accelerator_Systems_Division/Accelerator_Operations_Physics/manuals/elegant_latest/elegant.html (cit. on p. 5).

[CER] CERN Accelerator Beam Physics Group. *MAD – Methodical Accelerator Design.* URL: madx.web.cern.ch (cit. on p. 5).

[GNU] GNU. *Scientific Library.* URL: www.gnu.org/software/gsl/ (cit. on p. 4).

[SC16] Conrad Sanderson and Ryan Curtin. “Armadillo: a template-based C++ library for linear algebra”. In: *Journal of Open Source Software* (2016). DOI: 10.21105/joss.00026 (cit. on p. 4).

[Sch] Schmidt, Jan Felix. *palattice C++ library.* URL: github.com/janfschmidt/palattice (cit. on p. 4).

[Sch+16] Jan Felix Schmidt et al. “Studies on Depolarization by Synchrotron Radiation using Elegant Particle Tracking”. In: *Proceedings IPAC’16.* (Busan). MOPOR046. 2016 (cit. on p. 1).

[Sch17] Jan Felix Schmidt. “Spindynamik in Elektronensynchrotronen”. PhD Thesis in preparation. Universität Bonn, 2017 (cit. on pp. 1, 3, 6, 9, 11).

[Tho27] L. H. Thomas. “The kinematics of an electron with an axis”. In: *Philosophical Magazine* 3.13 (1927), pp. 1–22. DOI: 10.1080/14786440108564170 (cit. on p. 6).

[WB06] Y. Wang and M. Borland. “Pelegant: A Parallel Accelerator Simulation Code for Electron Generation and Tracking”. In: *AIP Conf. Proc.* (Lake Geneva, WI, USA). Vol. 877. 2006, p. 241 (cit. on p. 5).