Profiles in SMILEI

F. Pérez

June 5, 2015

Several quantities require the input of a spatial and/or temporal profile:

- Species density: to define either charge or number density of a species at each point.
- Species average velocity: to define an offset of the velocity distribution at each point.
- Species temperature: to define the width of the velocity distribution at each point (valid only for distributions which require a temperature value).
- External field: to define an initial external field at each point.
- Laser: to define a temporal and/or spatial laser enveloppe.

Contents

1 Species profiles

For each species, it is possible to define spatial profiles that will determine the initial distribution of the particles. There are many ways to define a profile.

1.1 First technique: no profile

You may decide to have a constant profile: a constant value over the whole box. In this case, it is very easy: you do not need to define a profile.

For the density, you only need to define the keyword `density`. For example:

```
Species( ... ,
        density = 10.,
        ... )
```

defines a species with density $10\,n_c$ over the whole box (n_c is the critical density of light which wavelength is equal to the normalization length).

For the drift velocity, you only need to define the keyword `mean_velocity`. For example:

```
Species( ... ,
        mean_velocity = [0.05, 0., 0.],
        ... )
```

defines a species with drift velocity $v_x = 0.05\,c$ over the whole box (c is the speed of light).

For the temperature, you only need to define the keyword `temperature`. For example:

```
Species( ... ,
        initMomentum_type = "maxwell-juettner",
        temperature = [1e-5],
        ... )
```

defines a species with a Maxwell-Jüttner distribution of temperature $T = 10^{-5}\,m_e c^2$ over the whole box. Note that the temperature may be anisotropic: `temperature=[1e-5, 2e-5, 2e-5]`.

1.2 Second technique: *python* profile

Any *python* function can be a profile. You must have basic *python* knowledge to build these functions. Examples:

```
def f(x):
        if x<1.: return 0.
        if x>1.: return 1.
```

```
def f(x,y):    # two variables for 2D simulation
        import math
        twoPI = 2.* math.pi
        return math.cos(  twoPI * x/3.2 )
```

```
f = lambda x: x**2 - 1
```

Once the function is created, you have to include it in the species definition, using the keywords `dens_profile`, `mvel_x_profile`, `mvel_y_profile`, `mvel_z_profile`, `temp_x_profile`, `temp_y_profile` or `temp_z_profile`:

```
Species( ... ,
        density = 10.,
        dens_profile = f,
        ... )
```

```
Species( ... ,
        mean_velocity = [0.05, 0., 0.],
        mvel_x_profile = f,
        ... )
```

```
Species( ... ,
        temperature = [1e-5],
        temp_x_profile = f,
        temp_y_profile = f,
        temp_z_profile = f,
        ... )
```

and so on ...

Important note:
The `density`, `mean_velocity` and `temperature` are always required. They define a factor by which the profile is multiplied.

1.3 Third technique: built-in *python* functions

SMILEI provides some *python* functions to help you build your profiles.
The same keywords from the previous sections apply. We use the notation `****_profile`, where `****` is one of `dens`, `mvel_x`, `mvel_y`, `mvel_z`, `temp_x`, `temp_y` or `temp_z`, depending on what profile you want to define.
Do not forget the `density`, `mean_velocity` and `temperature` keywords in any case!

To define a constant profile:

```
****_profile = constant(value=1., xvacuum=0., yvacuum=0.)
```

The argument `value` can be used to change the magnitude, but typically `constant()` is used.
`xvacuum` and `yvacuum` are empty lengths before starting the profile.

To define a trapezoidal profile:

```
****_profile = trapezoidal(max=1.,
                           xvacuum=0., xplateau=None, xslope1=0., xslope2=0.,
                           yvacuum=0., yplateau=None, yslope1=0., yslope2=0. )
```

where `max` is the maximum value, `xvacuum` is the empty length before the ramp up, `xplateau` is the length of the plateau (default is `sim_length - xvacuum`), `xslope1` is the length of the ramp up and `xslope2` is the length of the ramp down. The other arguments are the same, but for a 2D simulation.
All arguments are optional.

To define a gaussian profile:

```
****_profile = gaussian(max=1.,
                        xvacuum=0., xlength=None, xfwhm=None, xcenter=None, xorder=2,
                        yvacuum=0., ylength=None, yfwhm=None, ycenter=None, yorder=2 )
```

where `max` is the maximum value, `xvacuum` is the empty length before starting the profile, `xlength` is the length of the profile (default is `sim_length - xvacuum`), `xfwhm` is the gaussian FWHM (default is `xlength/3.`), `xcenter` is the gaussian center position (default is in the middle of `xlength` and `xorder` is the order of the gaussian. The other arguments are the same, but for a 2D simulation. If `yorder==0`, then the profile is constant over y.
All arguments are optional.

To define a polygonal profile:

```
****_profile = polygonal( xpoints=[], xvalues=[] )
```

where `xpoints` is a list defining the position of points, and `xvalues` is a list defining the values of the profile at each point.
All arguments are optional.

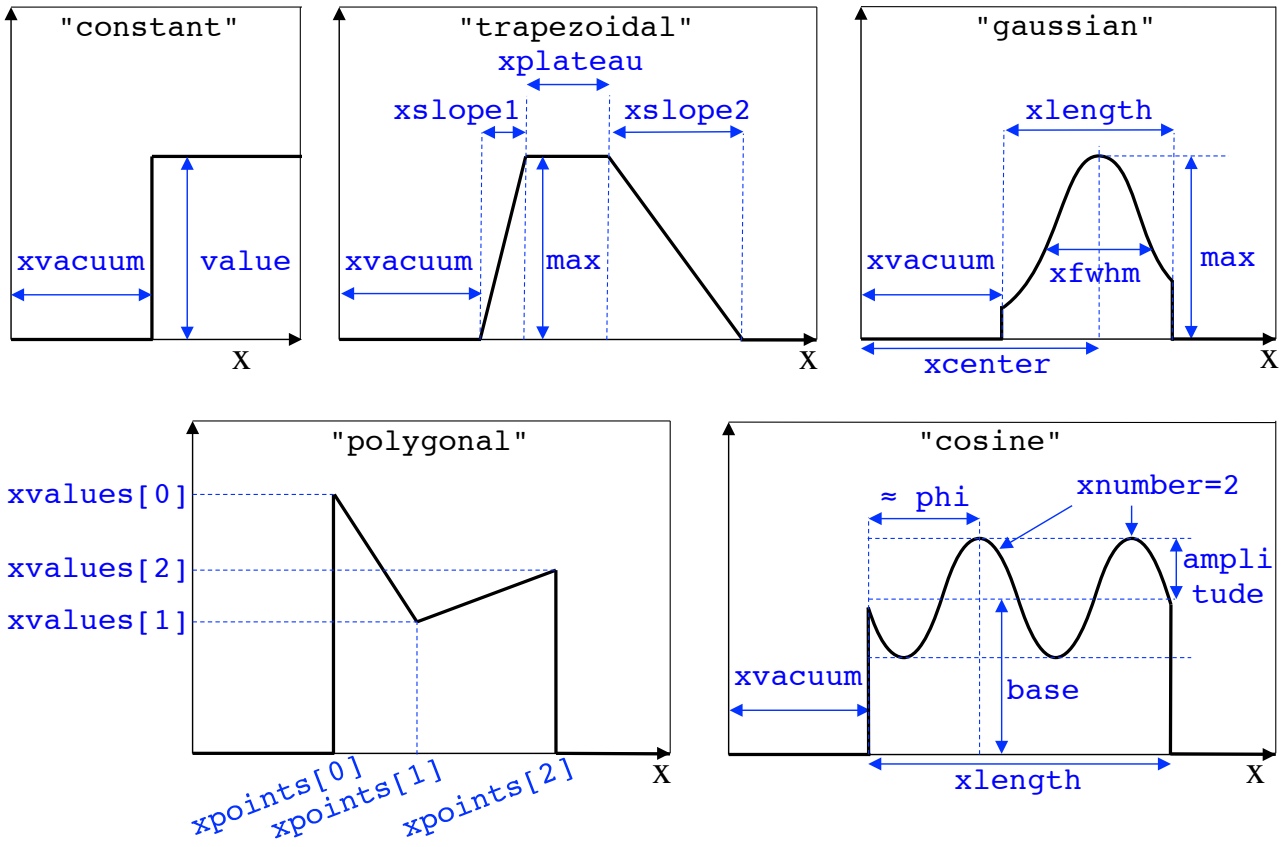
To define a cosine profile:

```
****_profile = cosine( base=1., amplitude=1.,
                       xvacuum=0., xlength=None, phi=0., xnumber=1 )
```

where `base` is an offset of the profile value, `amplitude` is the amplitude of the cosine, `xvacuum` is the empty length before starting the profile, `xlength` is the length of the profile (default is `sim_length - xvacuum`), `phi` is the phase offset and `xnumber` is the number of periods within `xlength`.
All arguments are optional.

Example:

```
Species( ... ,
        density = 10.,
        dens_profile = gaussian(xfwhm=0.3, xcenter=0.8),
        ... )
```



1.4 Fourth technique: old-style built-in c++ profiles (not recommended)

SMILEI still has hard-coded profiles that will be deprecated. They are mostly used for backwards-compatibility.

In this situation, several keywords may be needed: `****_profile`, `vacuum_length`, `****_length_x`, `****_length_y`, `****_length_z`, `****_dbl_params` and `****_int_params`.

The use of these keywords depends on the type of profile.

```
Species( ... ,
        ****_profile = "constant",
        vacuum_length = [x, y],
        ... )
```

```
Species( ... ,
        ****_profile = "trapezoidal",
        vacuum_length = [x, y],
        ****_length_x = [xplateau, xslope1, xslope2],
        ****_length_y = [yplateau, yslope1, yslope2],
        ... )
```

```
Species( ... ,
        ****_profile = "gaussian",
        vacuum_length = [x, y],
        ****_length_x = [xlength, xfwhm, xcenter],
        ****_length_y = [ylength, yfwhm, ycenter],
        ****_int_params = [xorder, yorder],
        ... )
```

```
Species( ... ,
        ****_profile = "polygonal",
        vacuum_length = [x, y],
        ****_length_x = [xpoints],
        ****_dbl_params = [xvalues],
        ... )
```

```
Species( ... ,
        ****_profile = "cosine",
        vacuum_length = [x, y],
        ****_length_x = [xlength],
        ****_dbl_params = [amplitude, xnumber],
        ... )
```

2 External field profiles

External field profiles work almost the same way as species profiles.

2.1 First technique: no profile

```
ExtField(
        field = "any_field",
        magnitude = magnitude
)
```

The `magnitude` keyword determines a constant value of the chosen external field.

2.2 Second technique: *python* function or *python* built-in profile

```
ExtField(
        field = "any_field",
        magnitude = magnitude,
        profile = myProfile
)
```

where *myProfile* is a *python* function or a built-in *python* profile.
All the built-in profiles from section 1.3 are available here.

2.3 Last technique: old-style built-in c++ profiles (not recommended)

The built-in profiles are the same as the species profiles (see section 1.4). However, the keywords are different: `profile`, `length_params_x`, `length_params_y`, `length_params_z`, `double_params` and `int_params`.

3 Laser profiles

Anybody ?