

creating channel coefficients

$p = l.\text{init_builder}$

\rightarrow layout initialized

$p \rightarrow$ new channel builder object

~~def~~ $p.\text{gen_ssf_parameters} \rightarrow$ small scale fading req for channel coefficients

$s.\text{use_spherical_waves} = 1$

$s \rightarrow$ simulation object

$\text{spherical} \rightarrow$ enables drifting

* $c = \text{get_channels}(p)$

get_channels a method of channel builder object.

c is now a g d-channel object

~~c is a 1×6 g d-channel, I think this is~~

~~because there are 3 objects (~~2 Rx~~, 2 Rx, 1 Tx)~~

$cn = \text{merge}(c)$

$\rightarrow cn$ is a 1×2 g d-channel

\hookrightarrow breaks up into the linear and circular path denoted in layout

"merge" combines channel segments into a continuous time evolution channel

* d is the non-drifting channel coefficients

$h = \text{cn}(1,1) . \text{fr}(100\text{e}6, 512)$

takes the first g d-channel from the merged c channel coefficients and
transforms the channel into frequency domain.
Returns frequency response

Input parameters are a bandwidth of $100\text{e}6$
and 512 equally spaced carriers

$h = \text{squeeze}(h)$ reduces parallel dimensionality

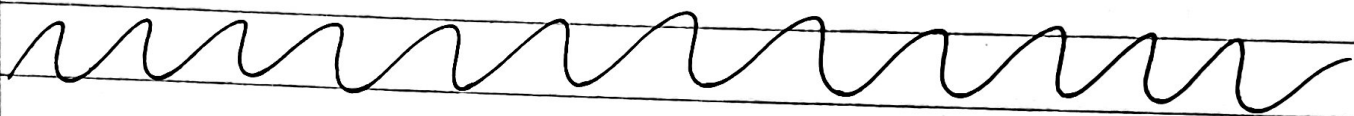
pdp \rightarrow power delay profile

gd-channel objects hold the data for the channel coefficients

gd-builder generates the parameters needed to extract channel coefficients

↳ get-channels generates the actual coefficients

get-channels generates random clusters around receiver



gd-channel objects are outputs

Channel coefficients provided in time domain

↳ list of delays

↳ list of complex-valued amplitudes

coefficients are under the `coeff` property

with indices of `[no_rxant, no_txant, no_path, no_snapshot]`

If delays are different they are 2-D

`rxant` → receive antenna

`txant` → transmit antenna

`path` → path

`snap` → snapshot

`no` → number of