creating channel coefficients

p = l.init_builder   l → layout initialized
p → new channel builder object

~~pssf~~ p.gen_ssf_parameters → small scale fading rng for channel coefficients

s.use_spherical_waves = 1   s → simulation object
spherical → enables drifting

c = get_channels(p)   get_channels a method of channel builder object.
c is now a qd_channel object

~~c is a 1×6 qd_channel, I think this is because there are 3 objects (~~2Tx~~, 2Rx, 1Tx)~~

the ×6 is b/c segments

cn = merge(c) → cn is a 1×2 qd_channel
↳ 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= cn(1,1).fr(100e6 ,512)

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

Input parameters are a bandwidth of 100e6 and 512 equally spaced carriers

h= squeeze(h) reduces parallel dimensionality

pdp → power delay profile

qd_channel objects hold the data for the channel coefficients

qd_builder generates the parameters needed to extract channel coefficients
↳ get_channels generates the actual coefficients

get_channels generates random <u>clusters</u> around receiver

qd_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_trant, no_path, no_snapshot]
If delays are different, they 4, else 2-D
rxant → receive antenna
trant → transmit antenna
path → path
snap → snapshot
no → number of