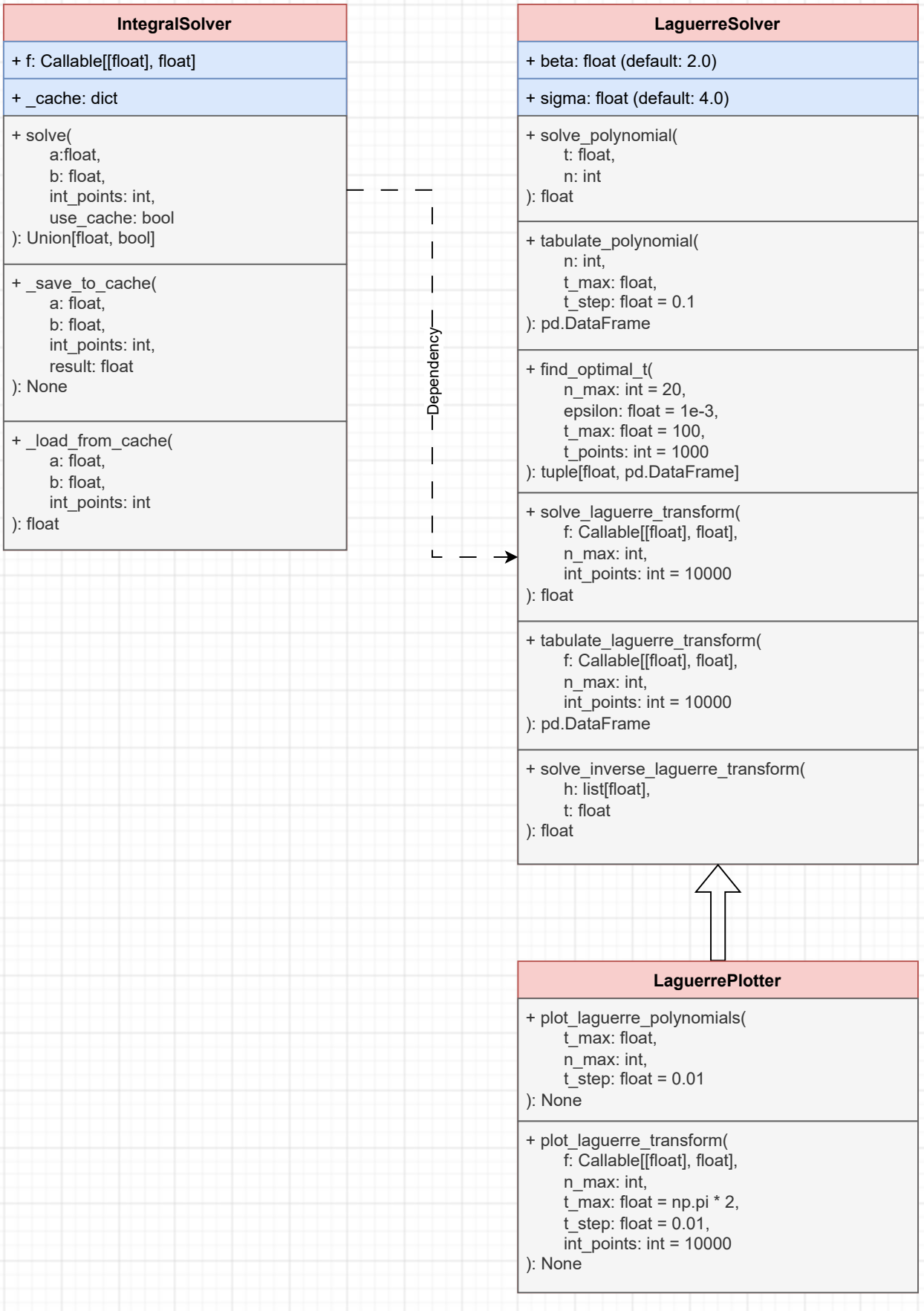
IntegralSolver
+ f: Callable[[float], float]
+ _cache: dict
+ solve(
a:float,
b: float,
int_points: int,
use_cache: bool
): Union[float, bool]
+ _save_to_cache(
a: float,
b: float,
int_points: int,
result: float
): None
+ _load_from_cache(
a: float,
b: float,
int_points: int
): float
Dependency
LaguerreSolver
+ beta: float (default: 2.0)
+ sigma: float (default: 4.0)
+ solve_polynomial(
t: float,
n: int
): float
+ tabulate_polynomial(
n: int,
t_max: float,
t_step: float = 0.1
): pd.DataFrame
+ find_optimal_t(
n_max: int = 20,
epsilon: float = 1e-3,
t_max: float = 100,
t_points: int = 1000
): tuple[float, pd.DataFrame]
+ solve_laguerre_transform(
f: Callable[[float], float],
n_max: int,
int_points: int = 10000
): float
+ tabulate_laguerre_transform(
f: Callable[[float], float],
n_max: int,
int_points: int = 10000
): pd.DataFrame
+ solve_inverse_laguerre_transform(
h: list[float],
t: float
): float
LaguerrePlotter
+ plot_laguerre_polynomials(
t_max: float,
n_max: int,
t_step: float = 0.01
): None
+ plot_laguerre_transform(
f: Callable[[float], float],
n_max: int,
t_max: float = np.pi * 2,
t_step: float = 0.01,
int_points: int = 10000
): None