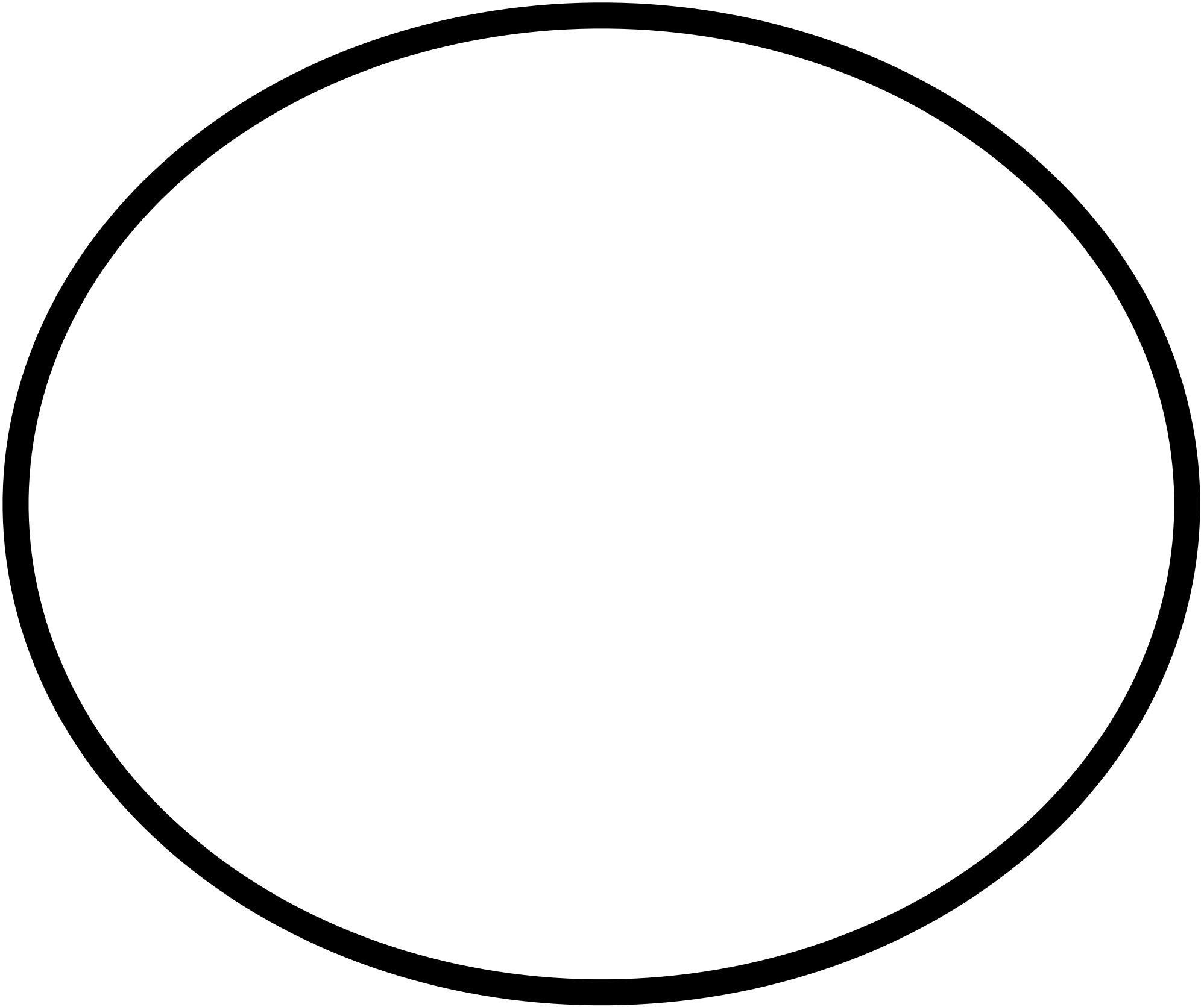
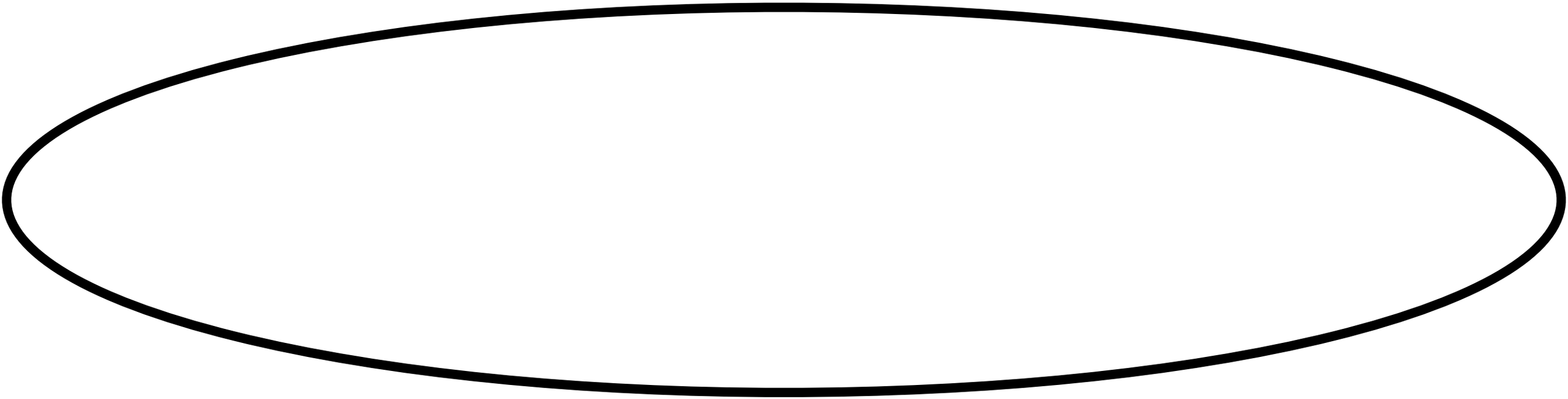
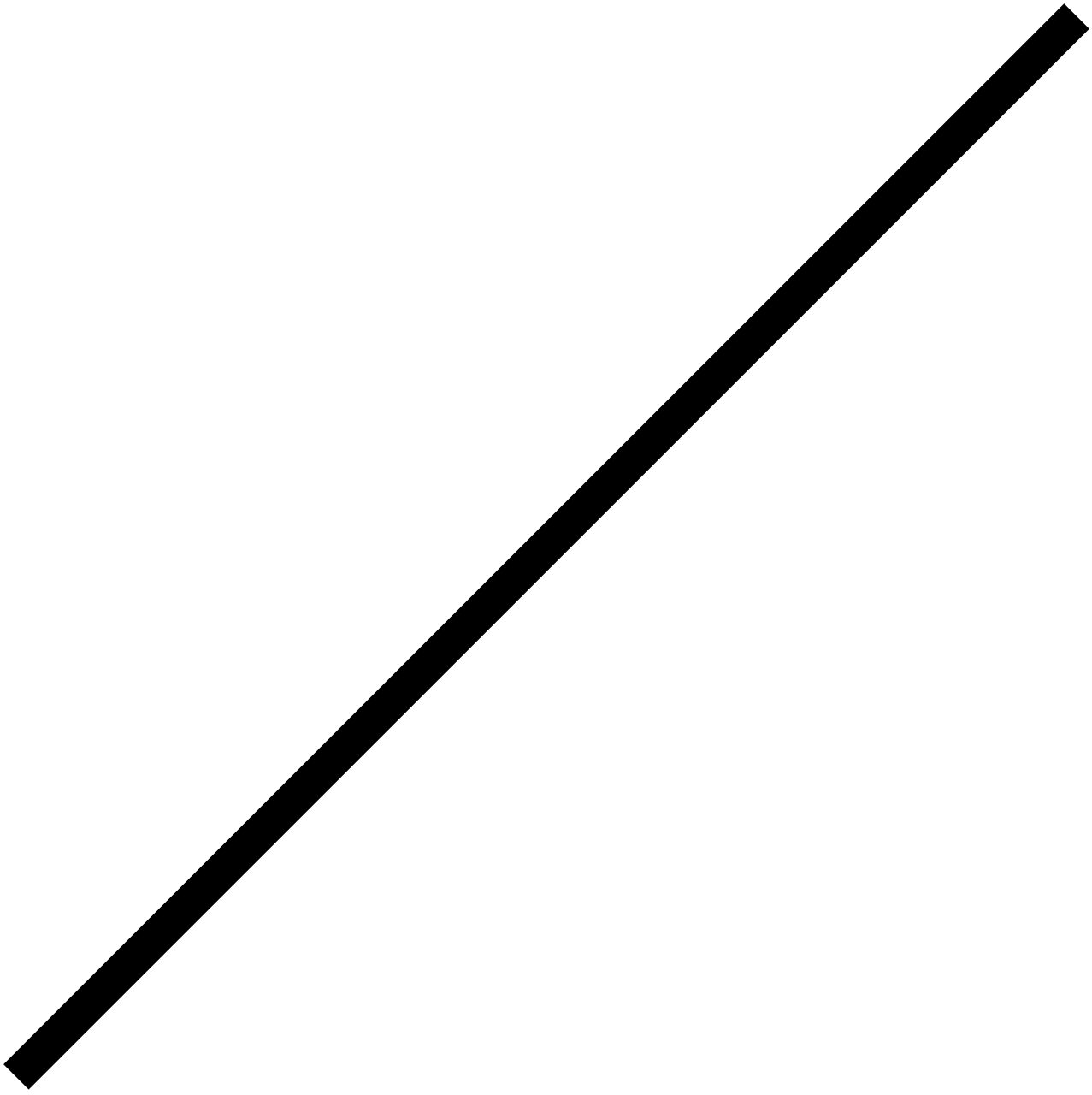
CLI Plugins: Reusable Parameters

```
@click.command(
    "pathsampling",
    short_help="Run any path sampling simulation, including TIS variants",
)
@INPUT_FILE.clicked(required=True)
@OUTPUT_FILE.clicked(required=True)
@SCHEME.clicked(required=False)
@INIT_CONDS.clicked(required=False)
@N_STEPS_MC
def pathsampling(input_file, output_file, scheme, init_conds, nsteps):
    """General path sampling, using setup in INPUT_FILE"""
    storage = INPUT_FILE.get(input_file)
    pathsampling_main(output_storage=OUTPUT_FILE.get(output_file),
                      scheme=SCHEME.get(storage, scheme),
                      init_conds=INIT_CONDS.get(storage, init_conds),
                      n_steps=nsteps)
```

Reusable CLI parameter decorators ensure consistency







CLI Plugins: Reusable Parameters

```
@click.command(
    "pathsampling",
    short_help="Run any path sampling simulation, including TIS variants",
)
@INPUT_FILE.clicked(required=True)
@OUTPUT_FILE.clicked(required=True)
@SCHEME.clicked(required=False)
@INIT_CONDS.clicked(required=False)
@N_STEPS_MC
def pathsampling(input_file, output_file, scheme, init_conds, nsteps):
    """General path sampling, using setup in INPUT_FILE"""
    storage = INPUT_FILE.get(input_file)
    pathsampling_main(output_storage=OUTPUT_FILE.get(output_file),
                      scheme=SCHEME.get(storage, scheme),
                      init_conds=INIT_CONDS.get(storage, init_conds),
                      n_steps=nsteps)
```

Reusable CLI parameter decorators ensure consistency

CLI Plugins: Main function

```
def pathsampling_main(output_storage, scheme, init_conds, n_steps):
    import openpathsampling as paths
    init_conds = scheme.initial_conditions_from_trajectories(init_conds)
    simulation = paths.PathSampling(
        storage=output_storage,
        move_scheme=scheme,
        sample_set=init_conds
    )
    simulation.run(n_steps)
    if output_storage:
        output_storage.tags['final_conditions'] = simulation.sample_set
    return simulation.sample_set, simulation
```