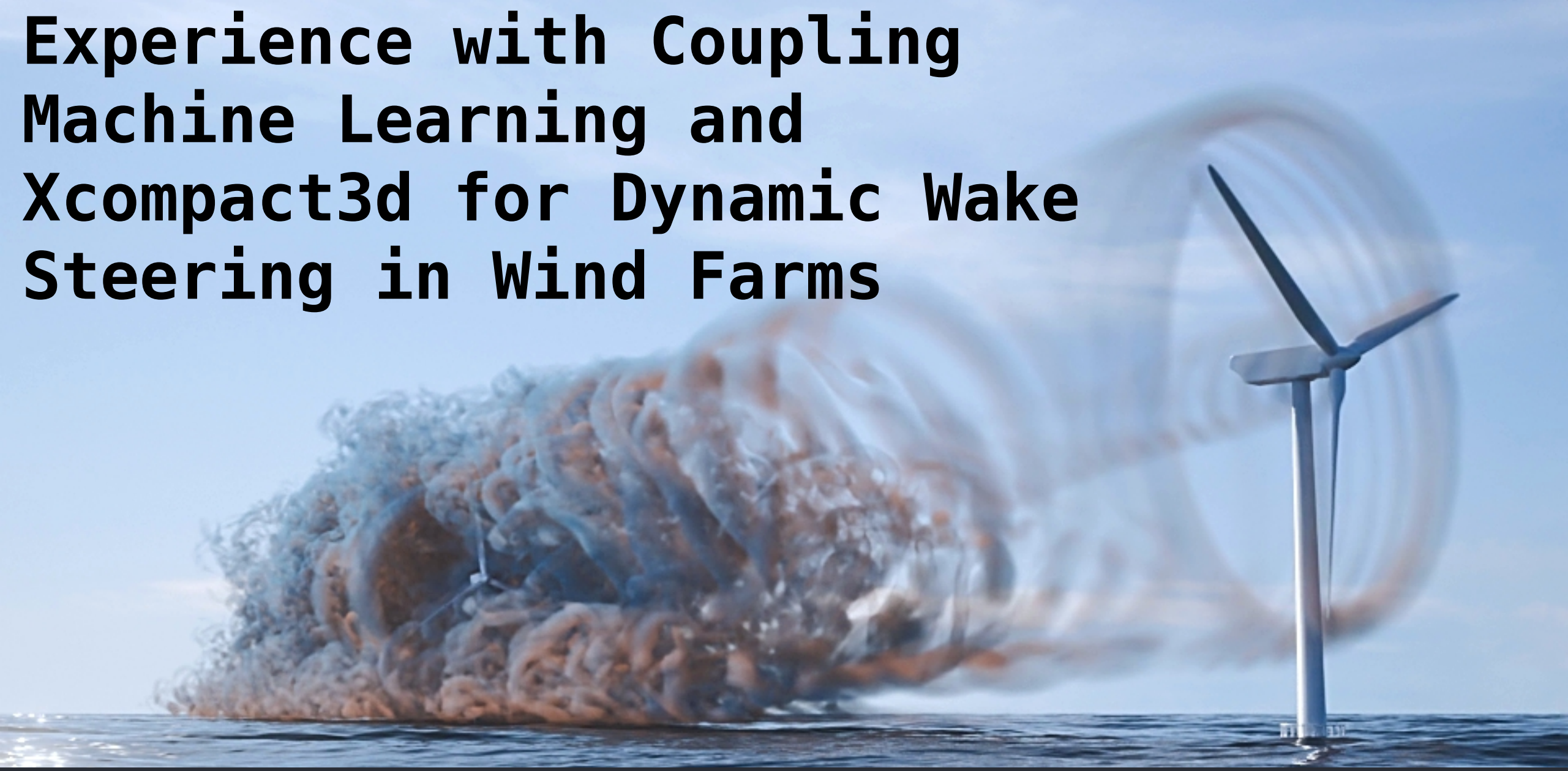
Experience with Coupling Machine Learning and Xcompact3d for Dynamic Wake Steering in Wind Farms



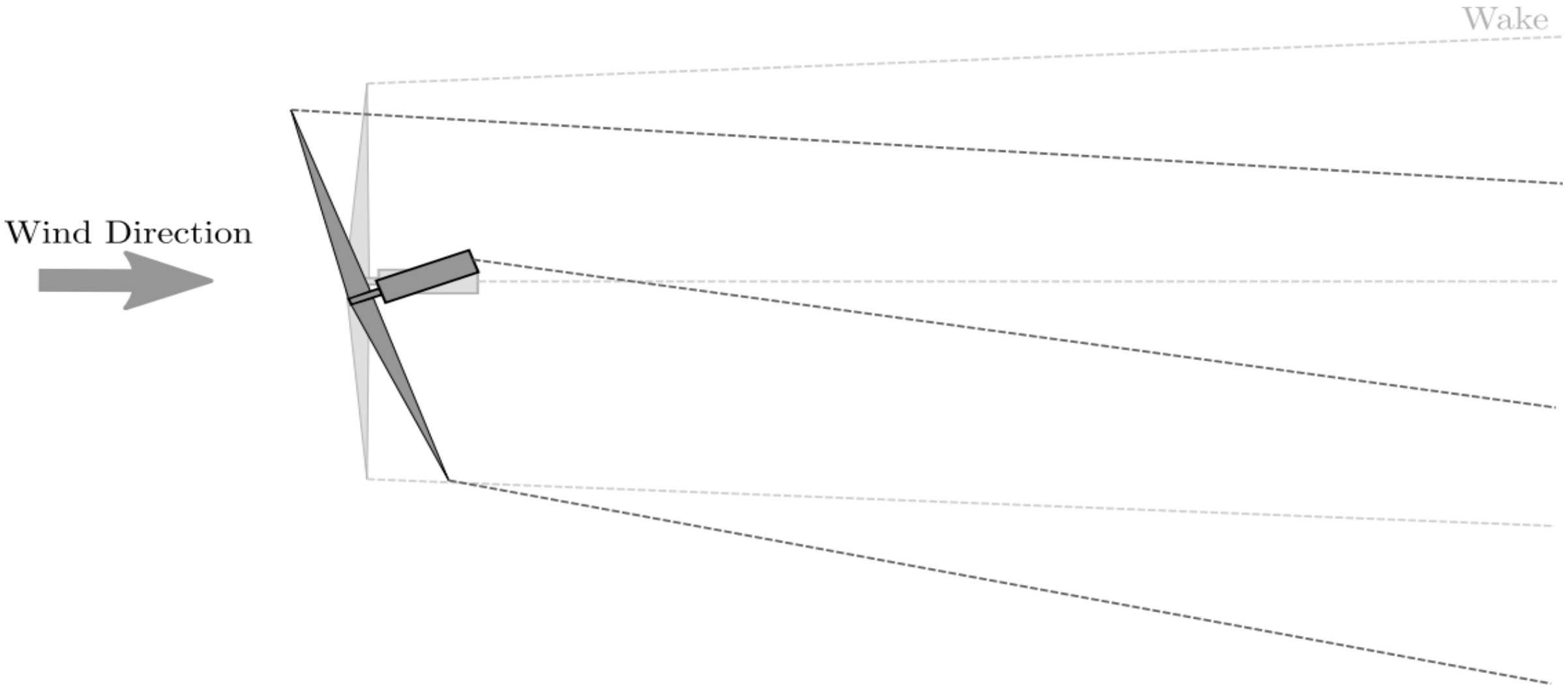
IMPERIAL

Dr Andrew Mole

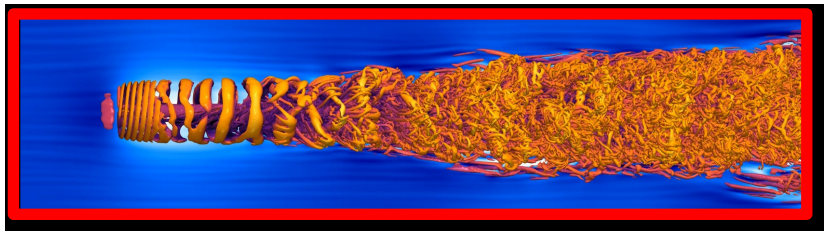
Wind Farms

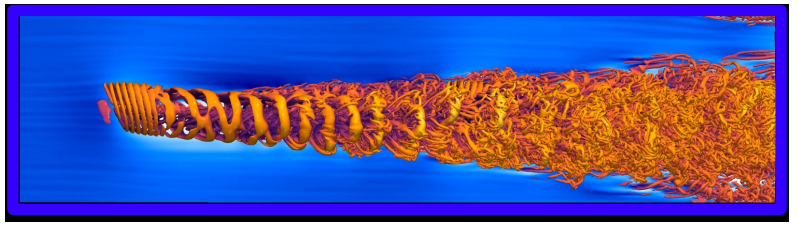
Wind Farm Yaw Control

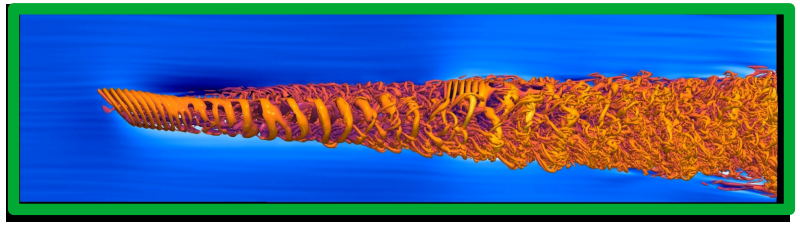
- Control the wakes by yawing the turbines.

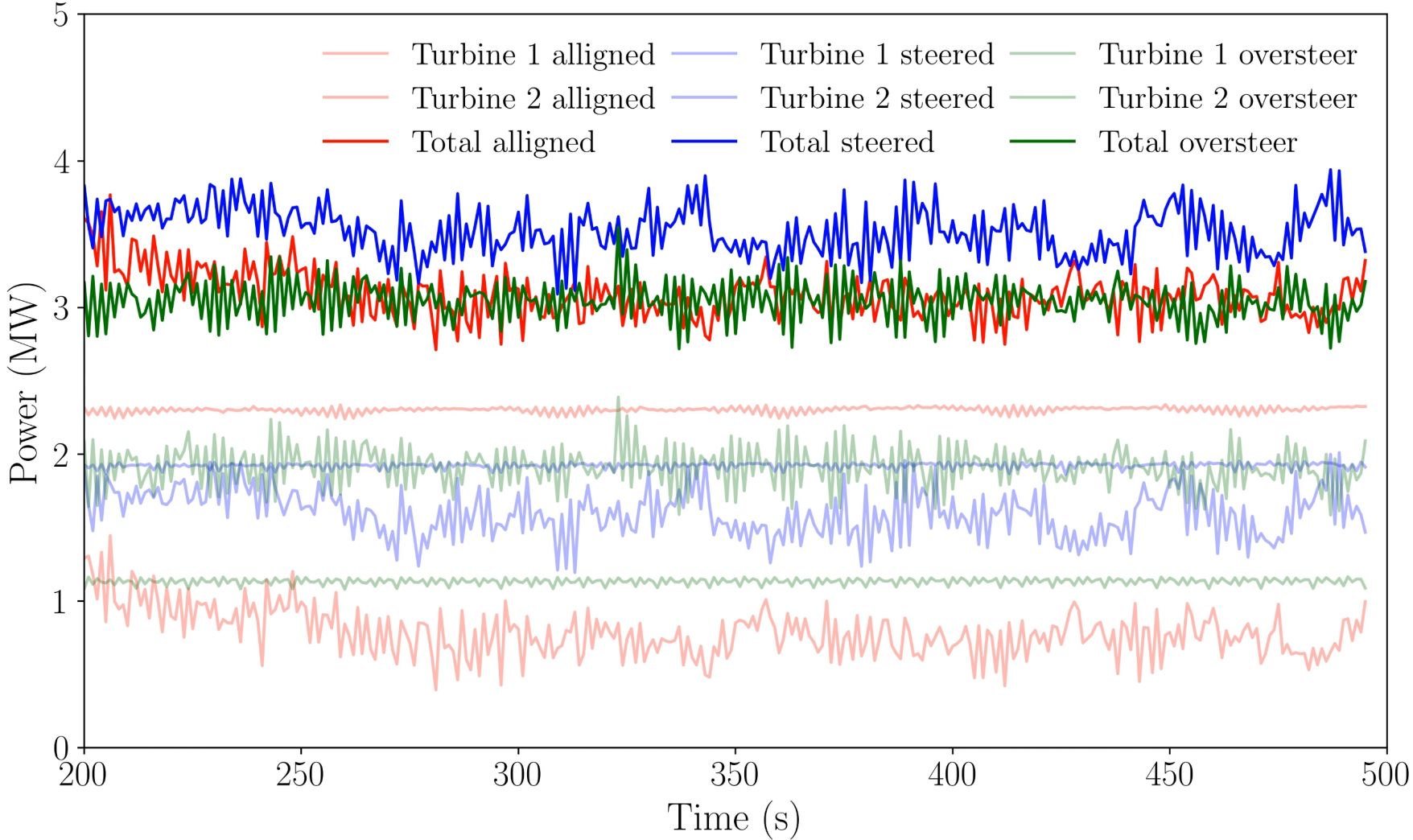


Wind Farm Yaw Control

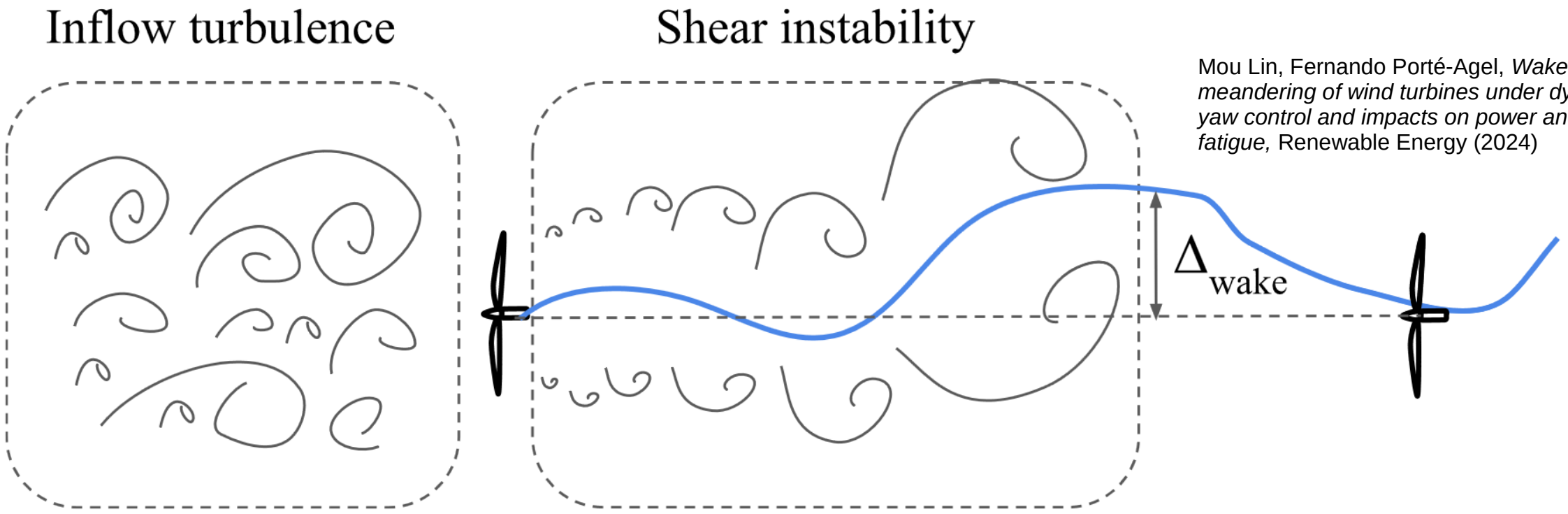




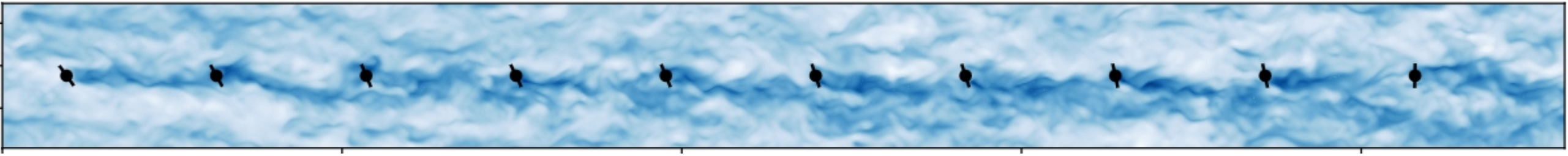


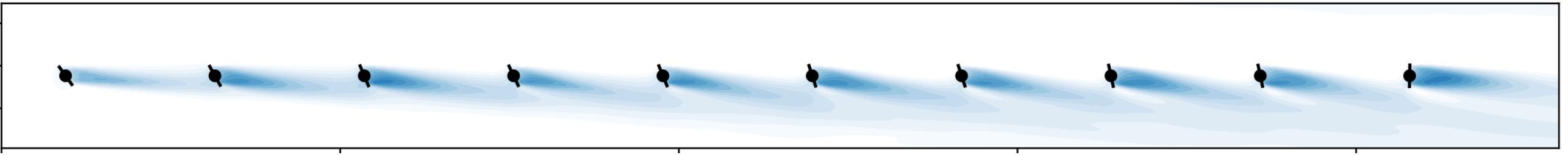


Wake Meandering



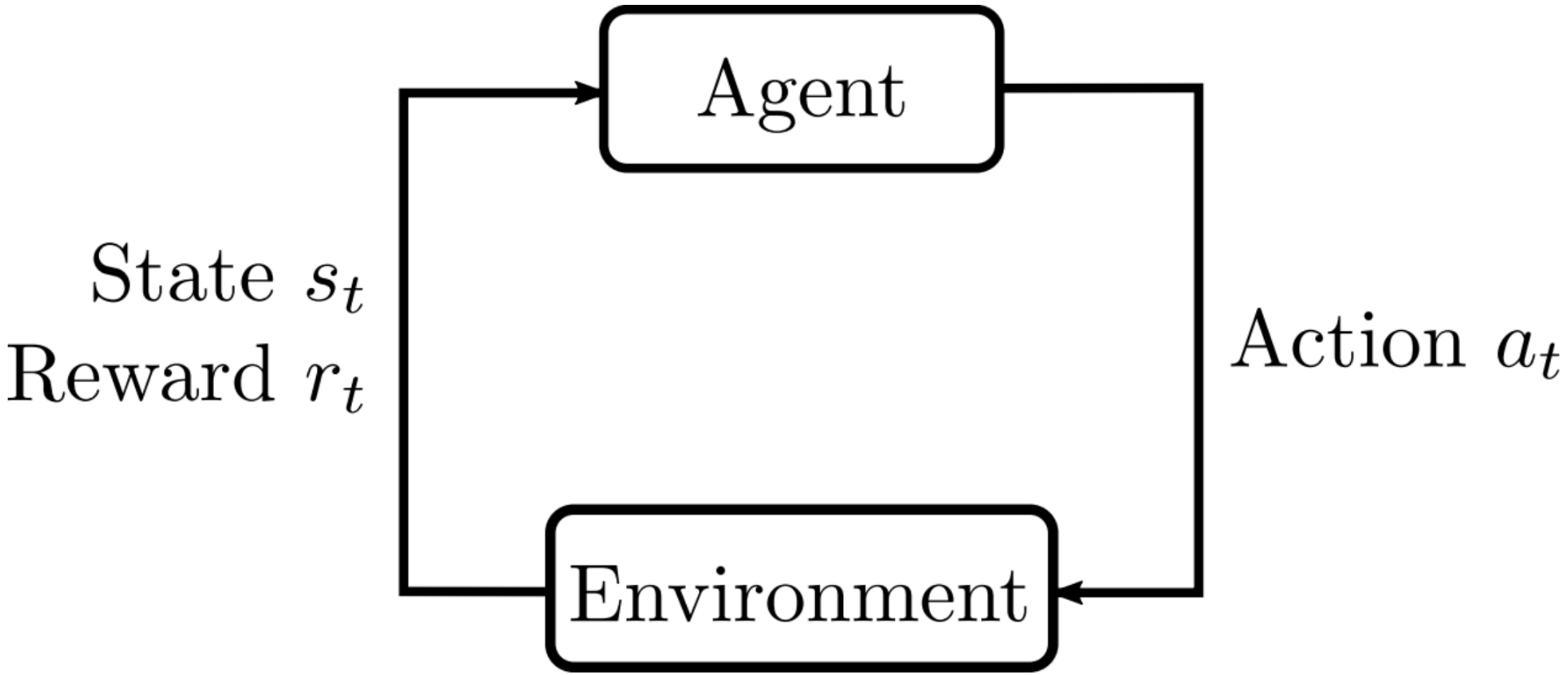
Mou Lin, Fernando Porté-Agel, *Wake meandering of wind turbines under dynamic yaw control and impacts on power and fatigue,* Renewable Energy (2024)





Reinforcement Learning

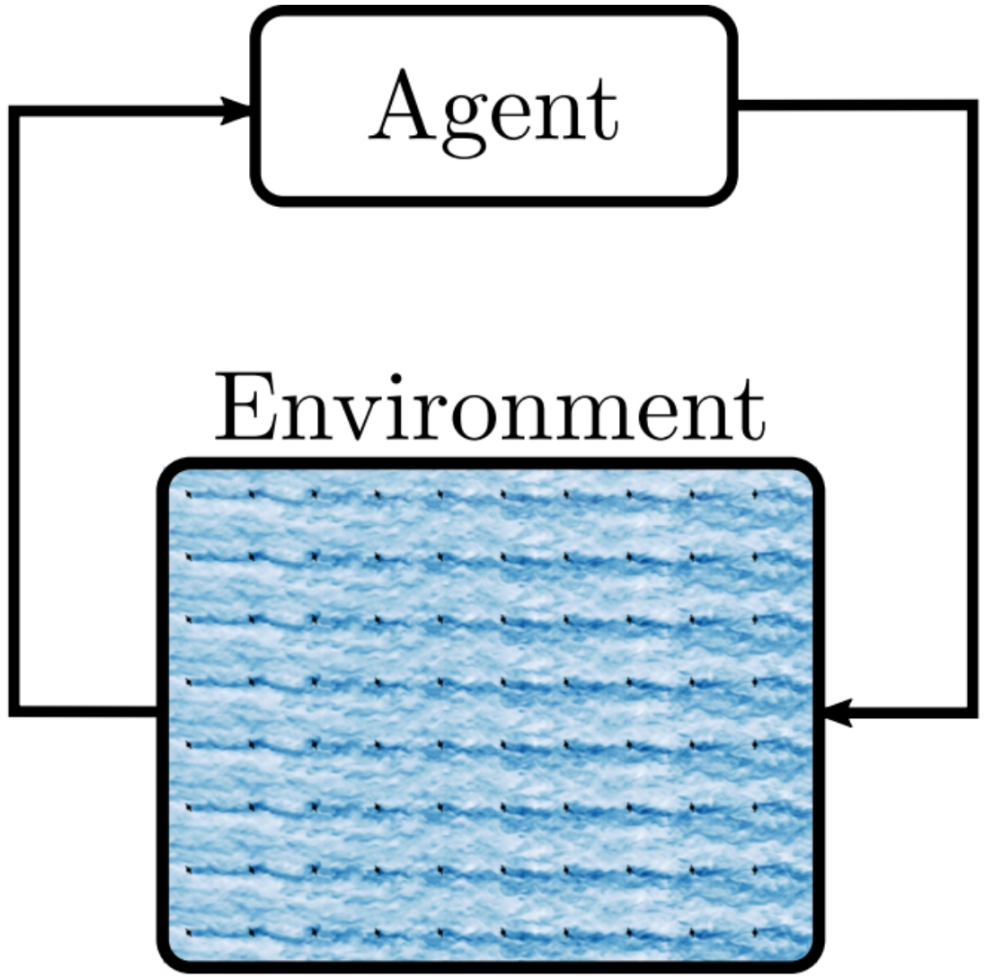
Wind Farm Reinforcement Learning

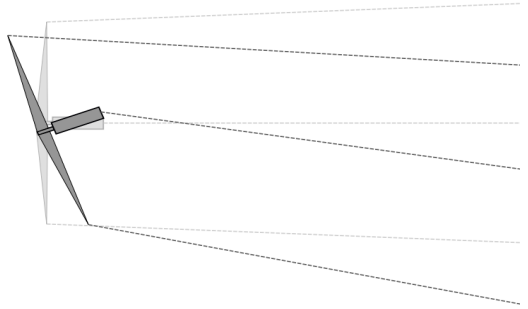


Wind Farm Reinforcement Learning

$$s_t = \boldsymbol{u}_i$$

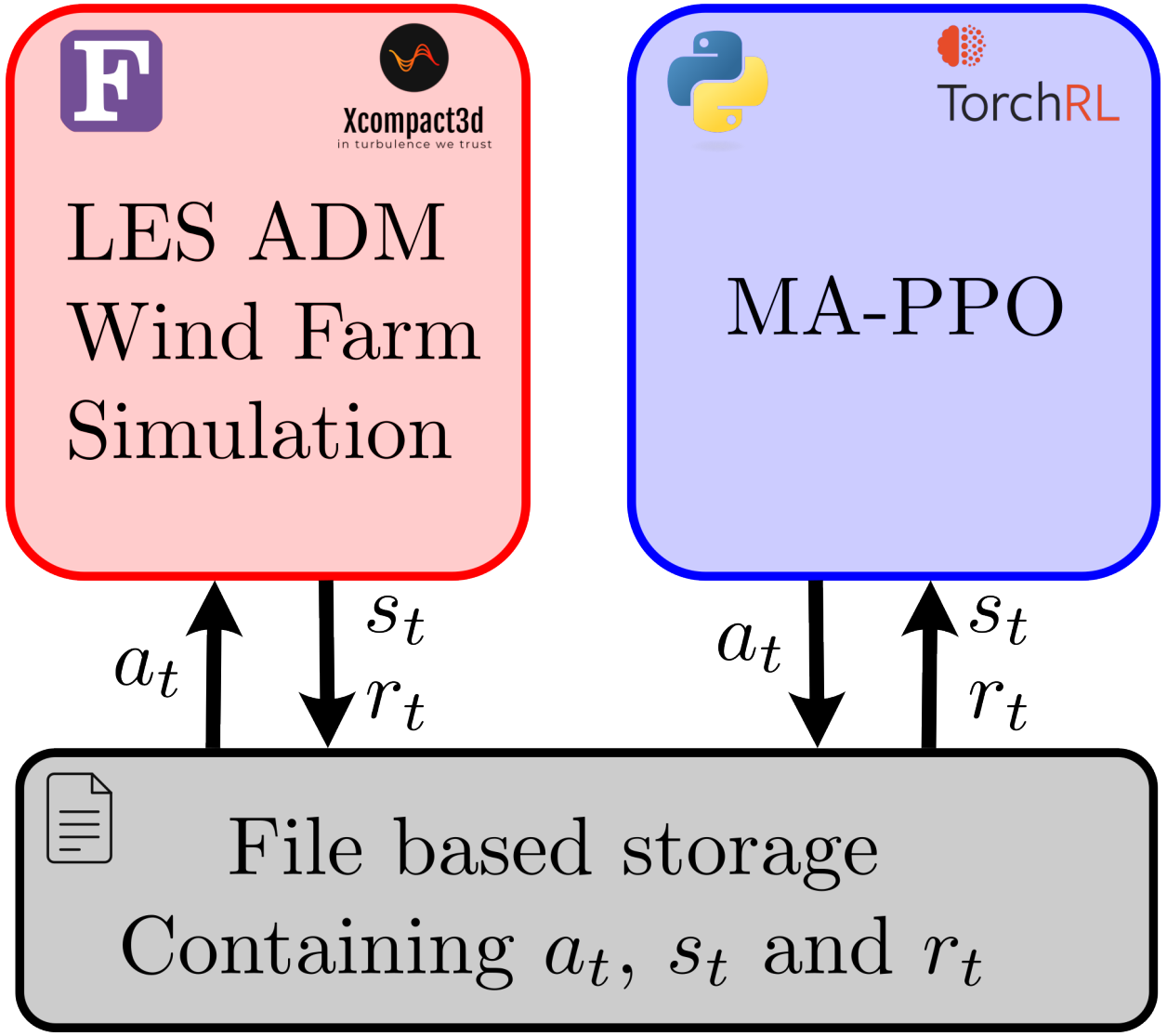
$$r_t = \sum_{n=1}^{N} P_n(\boldsymbol{\theta})$$



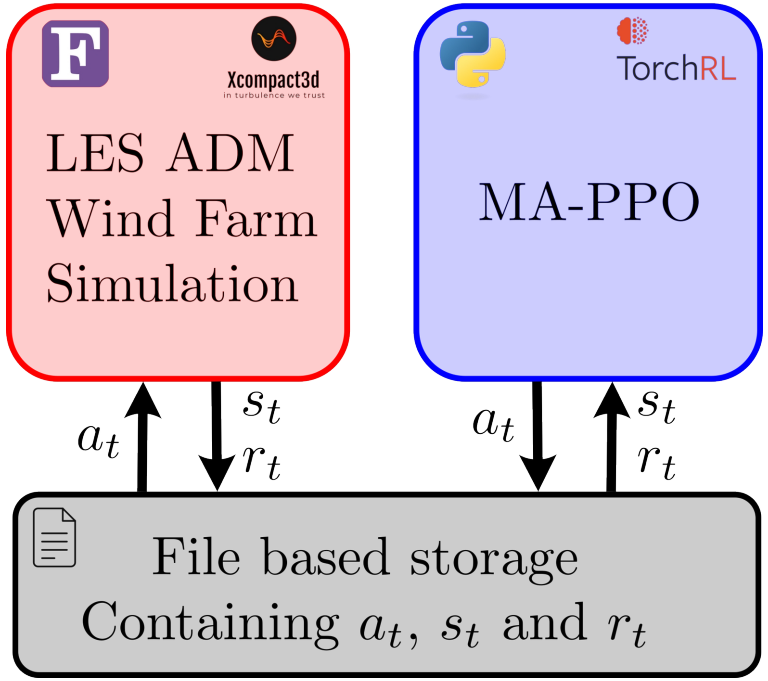


$$a_t = \frac{d\boldsymbol{\theta}}{dt}$$

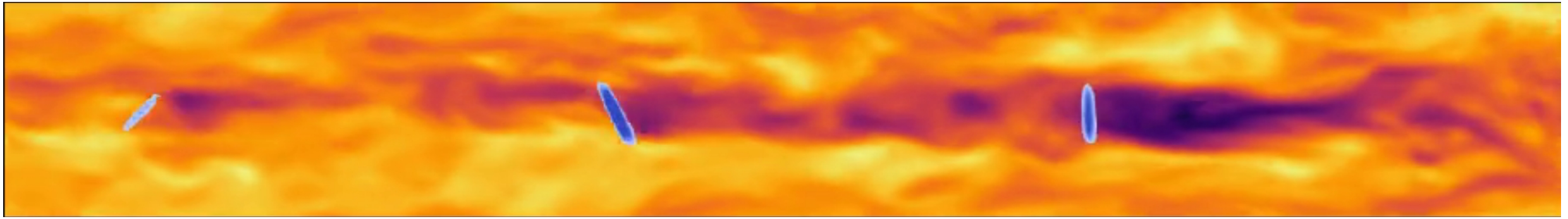
File Based Coupling



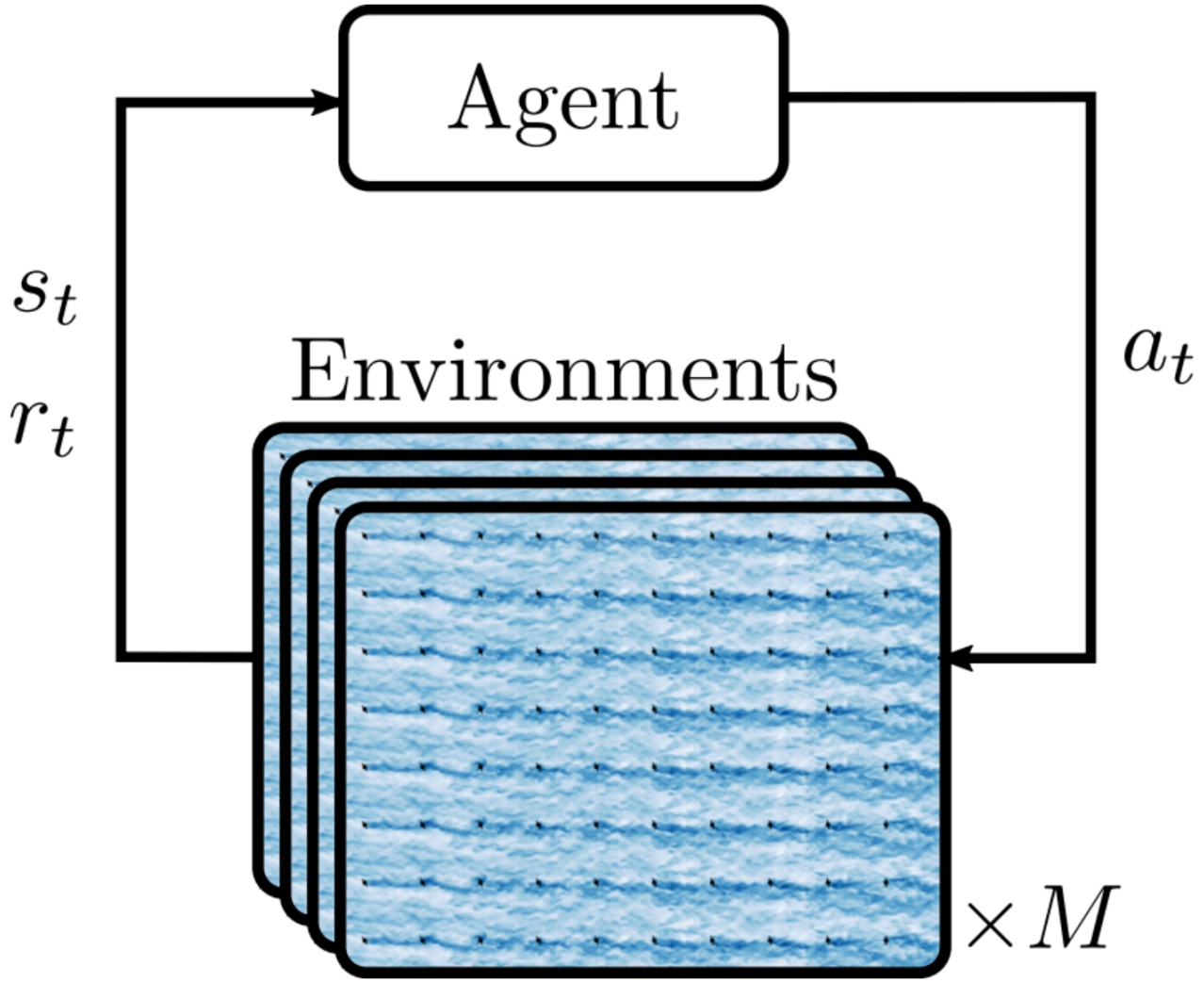
File Based Coupling



- File based I/O slows down interaction
- Lots of observations leads to large files
- Stopping and starting simulations costly
- Managing multiple environments difficult

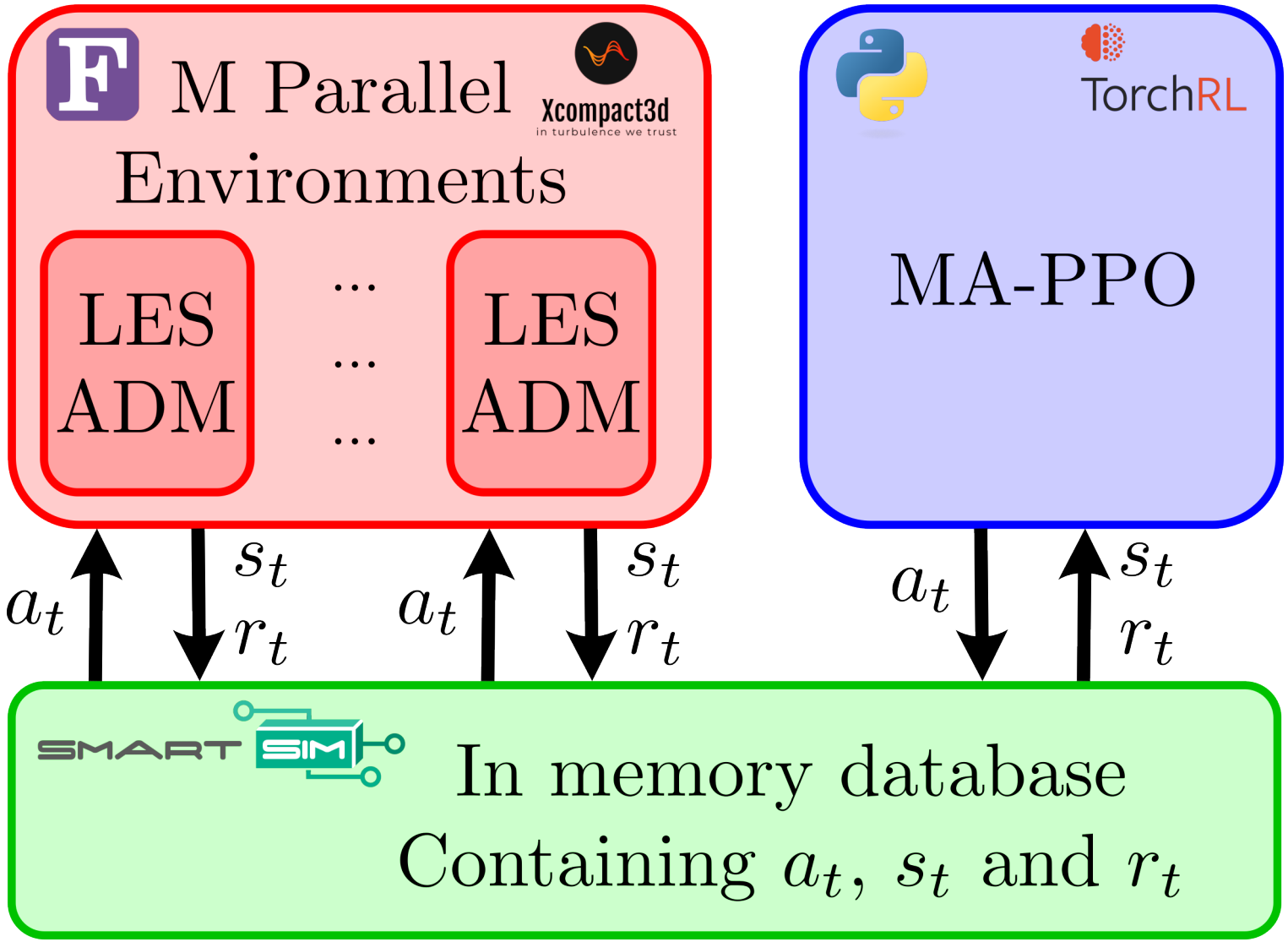


Parallel Environments



SmartSim

SmartSim Coupling



Xcompact3d Modifications

```
!*******************************************************************************
!
subroutine actuator_disc_model_smartredis_output()
!
!*******************************************************************************

  use param, only: itime, initstat, dt
  use MPI

  implicit none
  integer :: idisc,result,ierr
  real(kind=c_double), dimension(Nad) :: AllPowers
  real(kind=c_double), dimension(1) :: simulation_done, controller_done

  if (Nad>0) then
      AllPowers = [(ActuatorDisc(idisc)%Power_ave, idisc=1,Nad)]
      if (nrank==0) then
          result = client%put_tensor(trim(name_prefix)//'_turbine_powers', AllPowers, shape(AllPowers))
          if (result /= 0) then
              write(*,*) 'SmartRedis write failed'
              call MPI_ABORT(MPI_COMM_WORLD, result, ierr)
          endif
      endif
  endif

  controller_done(1) = 0
  if (nrank==0) then
      write(*,*) 'setting smartredis controller_done = 0'
      result = client%put_tensor(trim(name_prefix)//'_yaws_done', controller_done, shape(controller_done))
      if (result /= 0) then
          write(*,*) 'SmartRedis write failed'
          call MPI_ABORT(MPI_COMM_WORLD, result, ierr)
      endif
  end if

  simulation_done(1) = 1
  if (nrank==0) then
      write (*,*) 'setting smartredis simulation_done = 1'
      result = client%put_tensor(trim(name_prefix)//'_sim_done', simulation_done, shape(simulation_done))
      if (result /= 0) then
          write(*,*) 'SmartRedis write failed'
          call MPI_ABORT(MPI_COMM_WORLD, result, ierr)
      endif
  endif

  return

end subroutine actuator_disc_model_smartredis_output
```

```
result = client%put_tensor(trim
 (name_prefix)//'_turbine_powers',
 AllPowers, shape(AllPowers))
```

Python RL Modifications

```python
def _communicate(self, new_alpha):
    ######### Communication with SmartRedis server ############
    # Send yaws to X3D
    self.client.put_tensor(f"{self.instance}_yaws", new_alpha.detach().cpu().numpy().squeeze().astype(np.float64))
    self.client.put_tensor(f"{self.instance}_yaws_done", np.ones(1))

    # Poll whether X3D simulation is done
    while not self.client.get_tensor(f"{self.instance}_sim_done")[0]:
        continue

    # Receive power and observations from X3D
    turbine_powers = self.client.get_tensor(f"{self.instance}_turbine_powers")
    turbine_obs = np.zeros((self.total_probes, self.obs_per_probe))
    for i in range(self.total_probes):
        probe = self.client.get_tensor(f"{self.instance}_probe_{i+1}")
        turbine_obs[i] = self._normalise_probe_data(probe)
    turbine_obs = turbine_obs.reshape(self.n_turbs, self.probes_per_turbine, self.obs_per_probe)
    turbine_obs = turbine_obs.reshape(self.n_turbs, self.probes_per_turbine * self.obs_per_probe)

    # Process the outputs from the solver
    turbine_powers /= 1e06
    farm_power = turbine_powers.mean(axis=-1)
    farm_power = np.broadcast_to(farm_power, shape: (self.n_turbs,))

    # Convert to Torch tensors
    power = torch.tensor(farm_power, dtype=torch.float32).to(self.device)
    observation = torch.tensor(turbine_obs, dtype=torch.float32).to(self.device)

    return power, observation
```

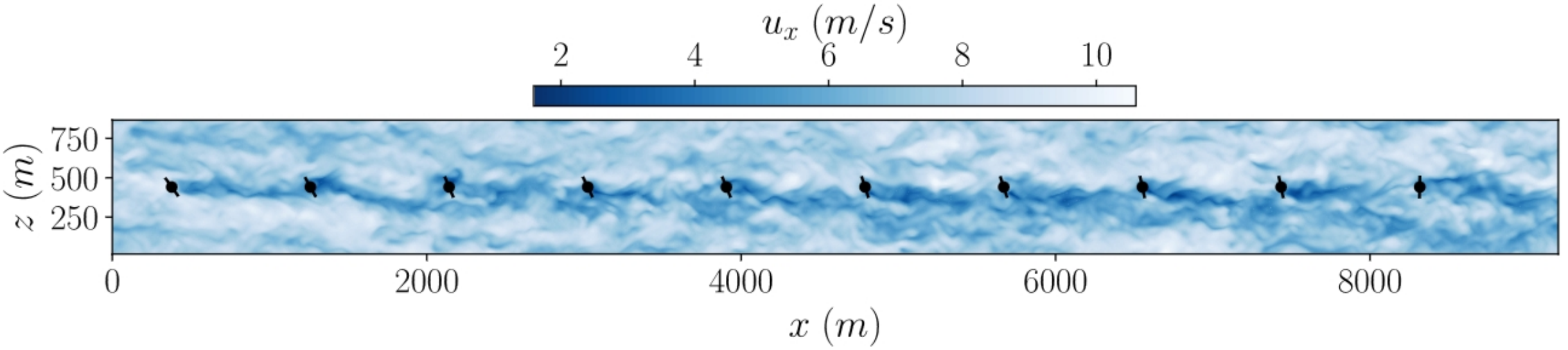
```
turbine_powers =
  self.client.get_tensor(f"
  {self.instance}_turbine_powers")
```

SmartSim Launch

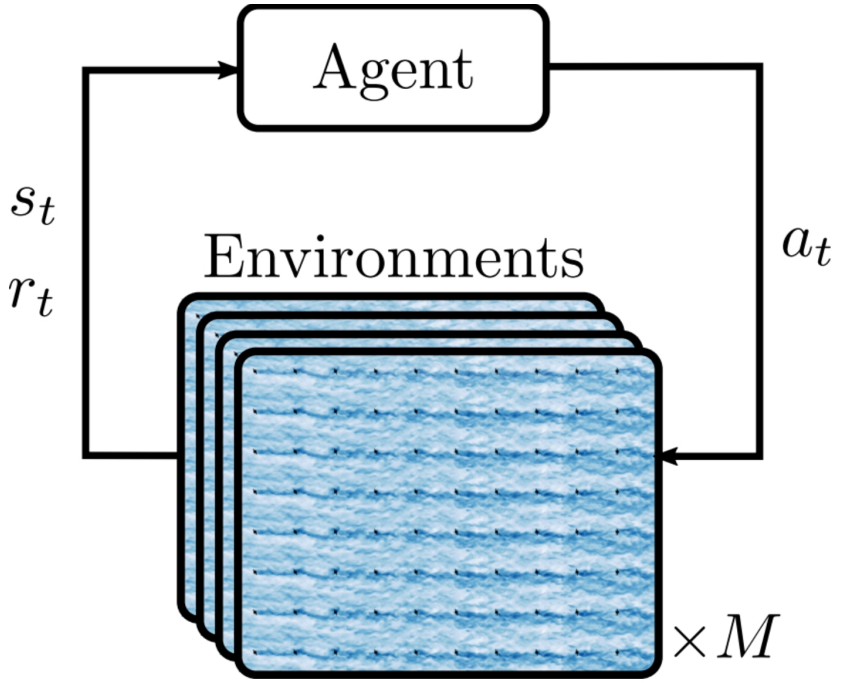
```python
if __name__ == '__main__':
    # Read PPO config
    initialize(config_path="./ppo/", version_base="1.2")
    cfg = compose(config_name="config_ppo.yaml")
    # Runtime parameters
    n_environments = cfg.env.n_parallel

    # Set up experiment
    exp = Experiment("training_ppo", launcher="auto")
    # Setup database
    db_port = 6783
    db = launch_database(exp, db_port)
    # Setup RL
    rl_app = launch_ppo(exp, cfg)
    # Setup simulations
    simulations = []
    for i in range(1, n_environments+1):
        simulation = launch_solver(exp, instance=i, cfg=cfg)
        simulations.append(simulation)

    # Start everything
    everything = simulations + [rl_app, db]
    exp.start(rl_app, block=False, summary=False)
    time.sleep(60)
    exp.start(*simulations, block=False, summary=False)

    while True:
        statuses = exp.get_status(*everything)
        ended = [(s == SmartSimStatus.STATUS_COMPLETED or s == SmartSim
Status.STATUS_FAILED) for s in statuses]
        if any(ended):
            print('Something finished/crashed so stopping everything')
            break
        time.sleep(30)

    exp.stop(*everything)
    print(exp.summary())
```

	Name	Entity-Type	JobID	RunID	Time	Status	Returncode
0	ppo	Model	7689439.1	0	21697.3175	SmartSimStatus.STATUS_COMPLETED	0
1	WindFarm_1	Model	7689439.2	0	21663.3177	SmartSimStatus.STATUS_CANCELLED	0
2	WindFarm_2	Model	7689439.3	0	21660.6035	SmartSimStatus.STATUS_CANCELLED	0
3	WindFarm_3	Model	7689439.4	0	21657.8969	SmartSimStatus.STATUS_CANCELLED	0
4	WindFarm_4	Model	7689439.5	0	21655.1793	SmartSimStatus.STATUS_CANCELLED	0
5	WindFarm_5	Model	7689439.6	0	21652.5100	SmartSimStatus.STATUS_CANCELLED	0
6	WindFarm_6	Model	7689439.7	0	21649.8602	SmartSimStatus.STATUS_CANCELLED	0
7	WindFarm_7	Model	7689439.8	0	21647.1894	SmartSimStatus.STATUS_CANCELLED	0
8	WindFarm_8	Model	7689439.9	0	21644.5002	SmartSimStatus.STATUS_CANCELLED	0
9	WindFarm_9	Model	7689439.10	0	21641.7967	SmartSimStatus.STATUS_CANCELLED	0
10	WindFarm_10	Model	7689439.11	0	21639.0930	SmartSimStatus.STATUS_CANCELLED	0
11	WindFarm_11	Model	7689439.12	0	21636.3064	SmartSimStatus.STATUS_CANCELLED	0
12	WindFarm_12	Model	7689439.13	0	21633.6314	SmartSimStatus.STATUS_CANCELLED	0
13	WindFarm_13	Model	7689439.14	0	21630.9200	SmartSimStatus.STATUS_CANCELLED	0
14	WindFarm_14	Model	7689439.15	0	21628.2001	SmartSimStatus.STATUS_CANCELLED	0
15	WindFarm_15	Model	7689439.16	0	21625.5273	SmartSimStatus.STATUS_CANCELLED	0
16	WindFarm_16	Model	7689439.17	0	21622.8169	SmartSimStatus.STATUS_CANCELLED	0
17	WindFarm_17	Model	7689439.18	0	21620.1195	SmartSimStatus.STATUS_CANCELLED	0
18	WindFarm_18	Model	7689439.19	0	21617.4098	SmartSimStatus.STATUS_CANCELLED	0
19	WindFarm_19	Model	7689439.20	0	21614.7084	SmartSimStatus.STATUS_CANCELLED	0
20	WindFarm_20	Model	7689439.21	0	21612.0054	SmartSimStatus.STATUS_CANCELLED	0
21	WindFarm_21	Model	7689439.22	0	21609.3044	SmartSimStatus.STATUS_CANCELLED	0
22	WindFarm_22	Model	7689439.23	0	21606.6511	SmartSimStatus.STATUS_CANCELLED	0
23	WindFarm_23	Model	7689439.24	0	21603.9819	SmartSimStatus.STATUS_CANCELLED	0
24	WindFarm_24	Model	7689439.25	0	21601.2724	SmartSimStatus.STATUS_CANCELLED	0
25	WindFarm_25	Model	7689439.26	0	21598.5844	SmartSimStatus.STATUS_CANCELLED	0
26	WindFarm_26	Model	7689439.27	0	21595.8873	SmartSimStatus.STATUS_CANCELLED	0
27	WindFarm_27	Model	7689439.28	0	21593.2880	SmartSimStatus.STATUS_CANCELLED	0
28	WindFarm_28	Model	7689439.29	0	21590.6699	SmartSimStatus.STATUS_CANCELLED	0
29	WindFarm_29	Model	7689439.30	0	21587.9985	SmartSimStatus.STATUS_CANCELLED	0
30	WindFarm_30	Model	7689439.31	0	21585.3287	SmartSimStatus.STATUS_CANCELLED	0
31	WindFarm_31	Model	7689439.32	0	21582.6322	SmartSimStatus.STATUS_CANCELLED	0
32	WindFarm_32	Model	7689439.33	0	21579.9227	SmartSimStatus.STATUS_CANCELLED	0
33	orchestrator_0	DBNode	7689439.0	0	21770.3947	SmartSimStatus.STATUS_CANCELLED	0

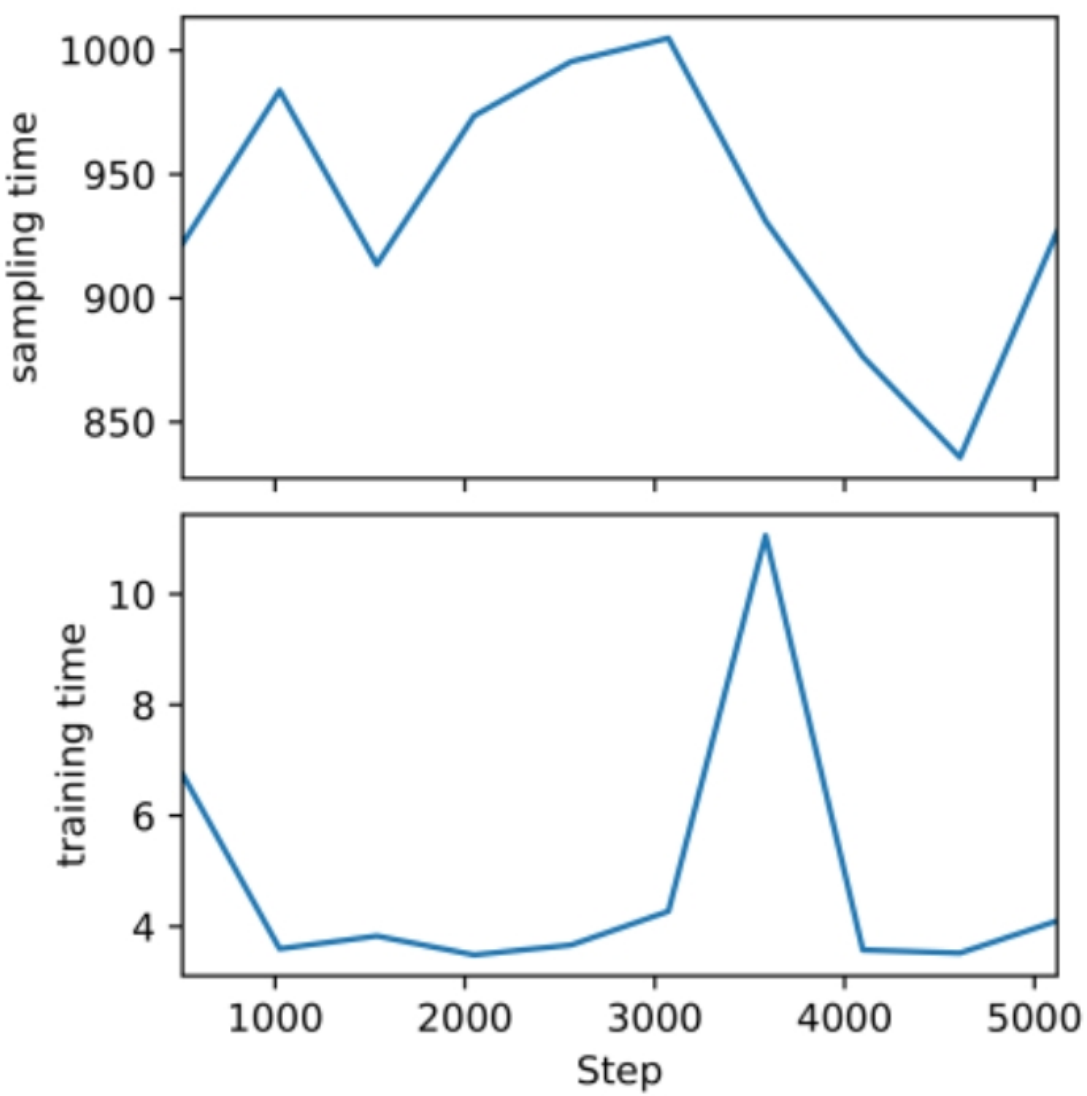
Set-up

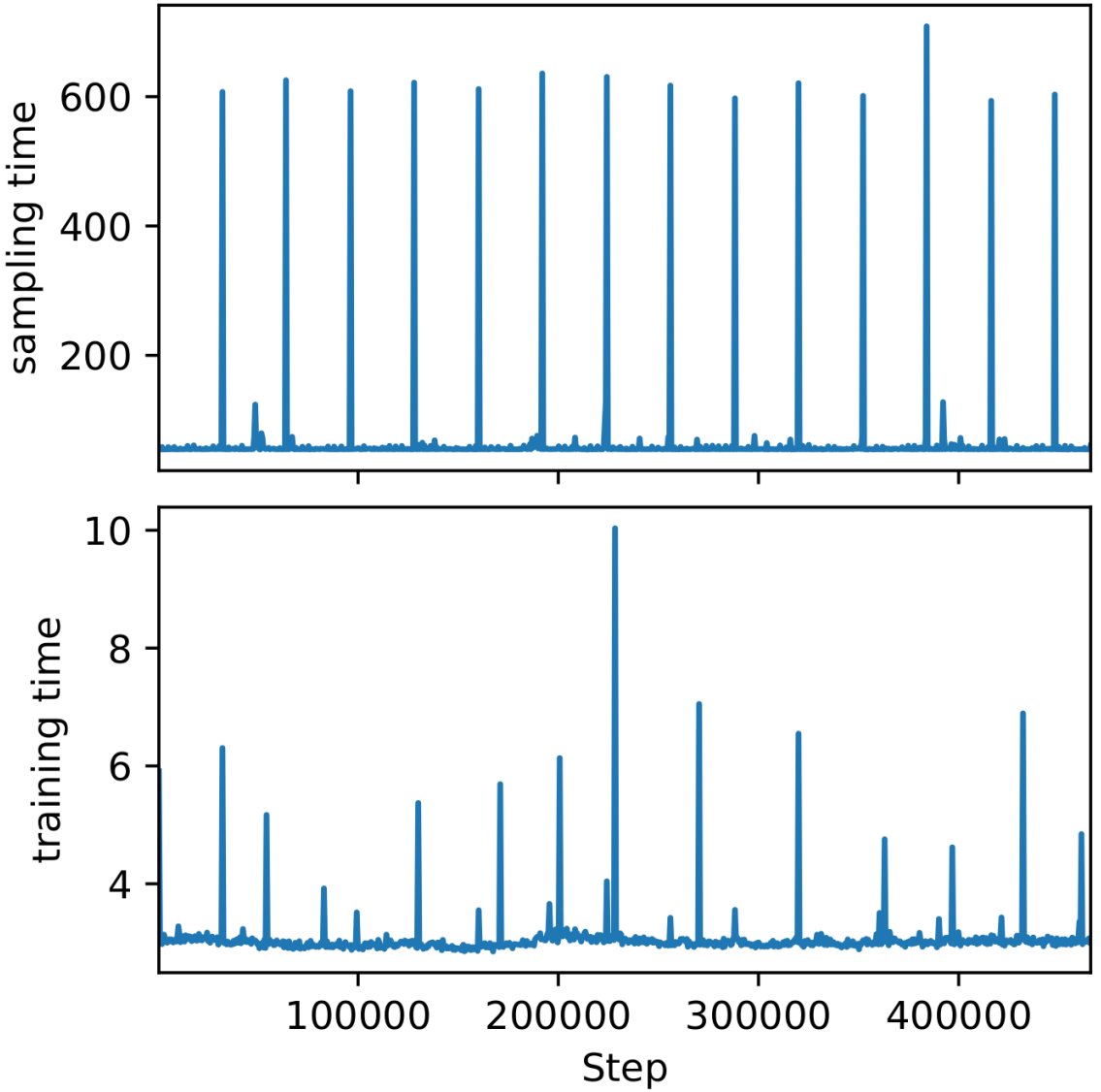


Number of Turbines: 10
Parallel Environments: 32
Cores per Environment: 128
Simulation steps per frame: 50
Frames per batch: 512
Max episode length: 1,000
Total frames: 1,000,000

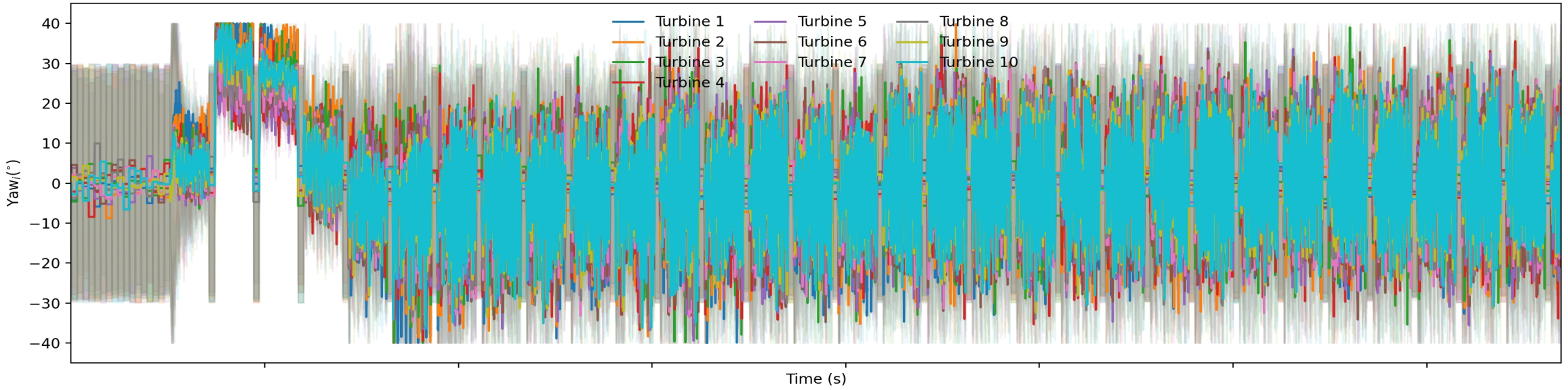


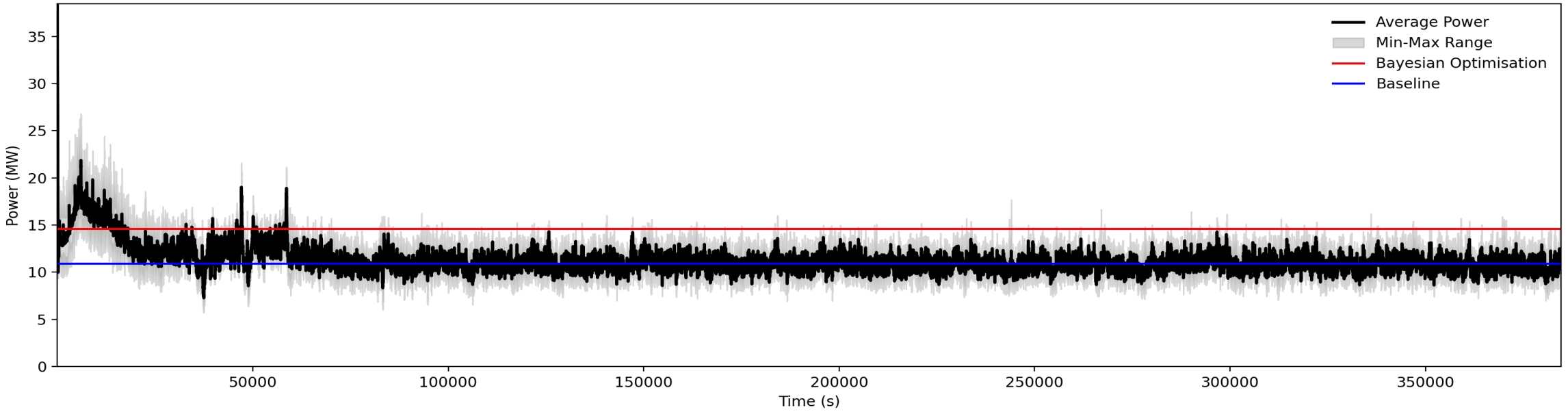
Speed


sampling time
1000
950
900
850
training time
10
8
6
4
1000
2000
3000
4000
5000
Step

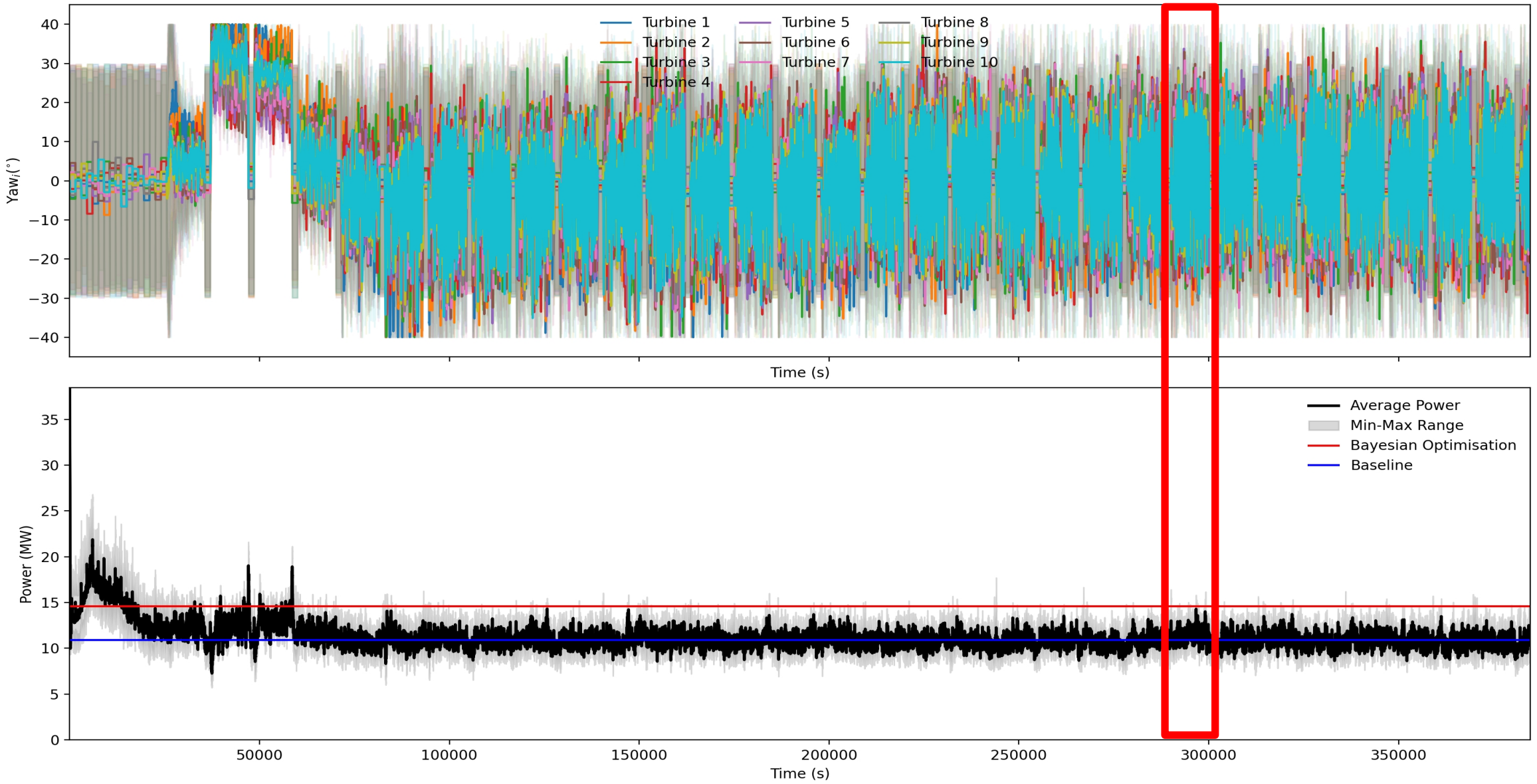

sampling time
600
400
200
training time
10
8
6
4
100000
200000
300000
400000
Step

Initial Results

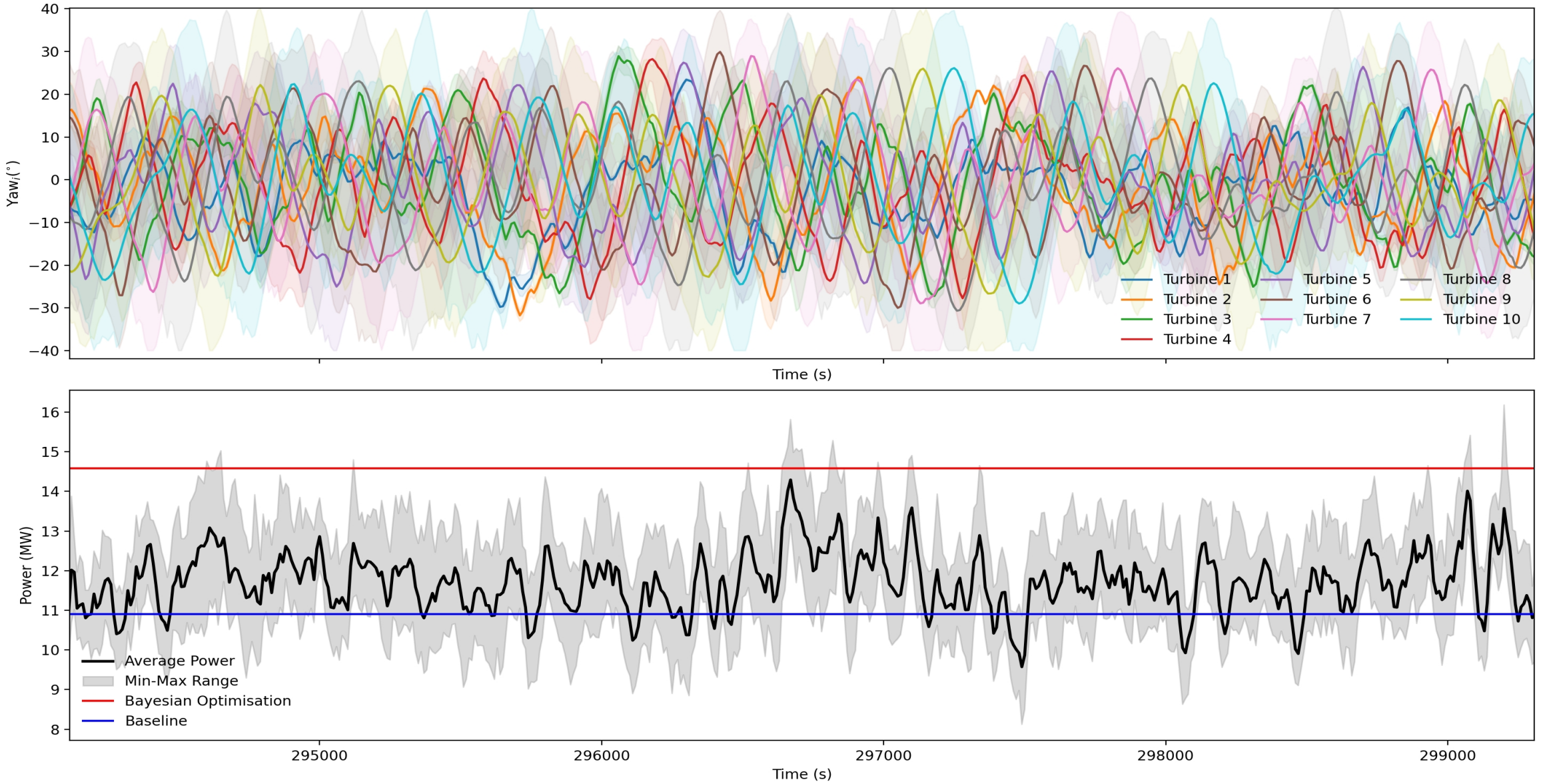




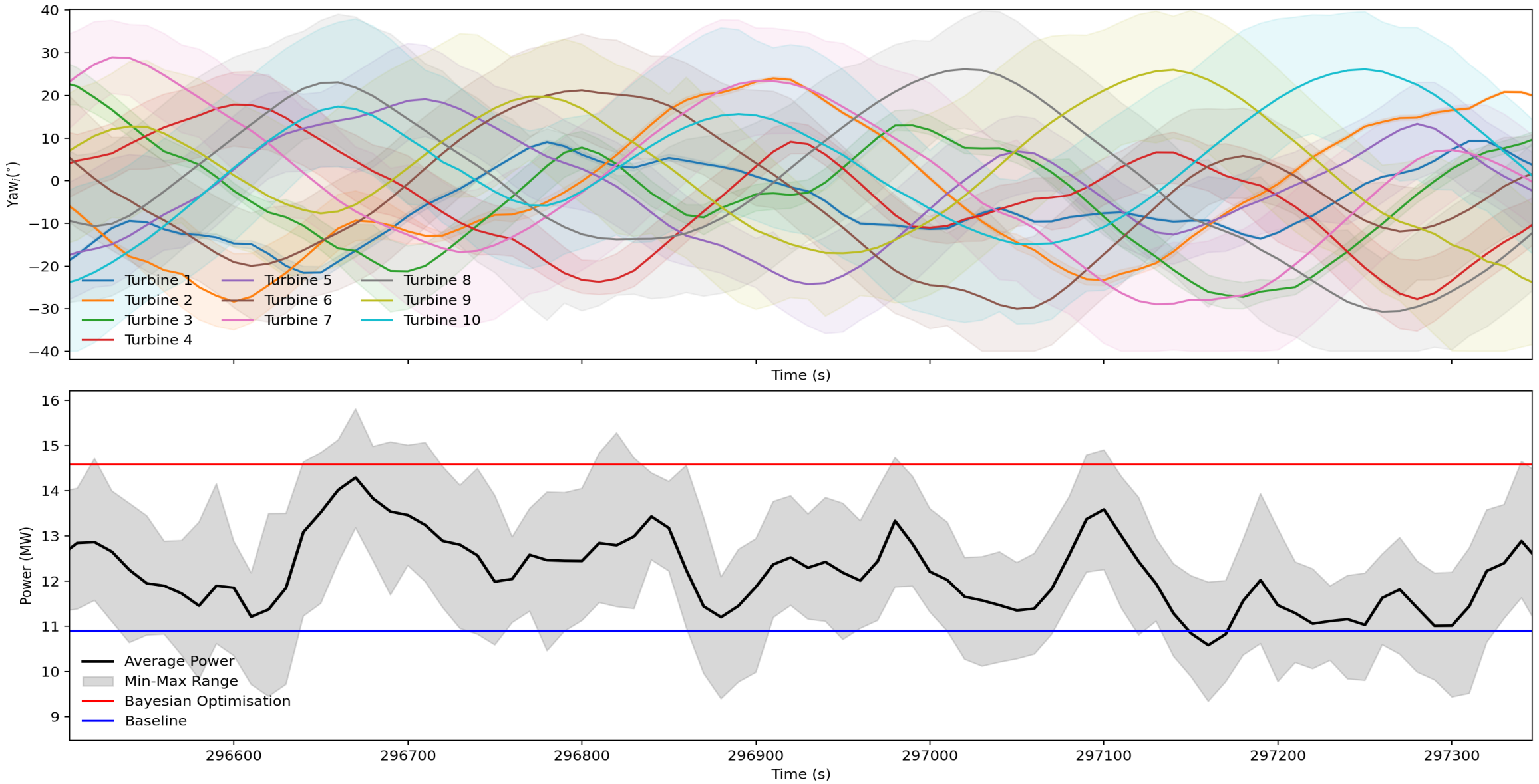
Initial Results



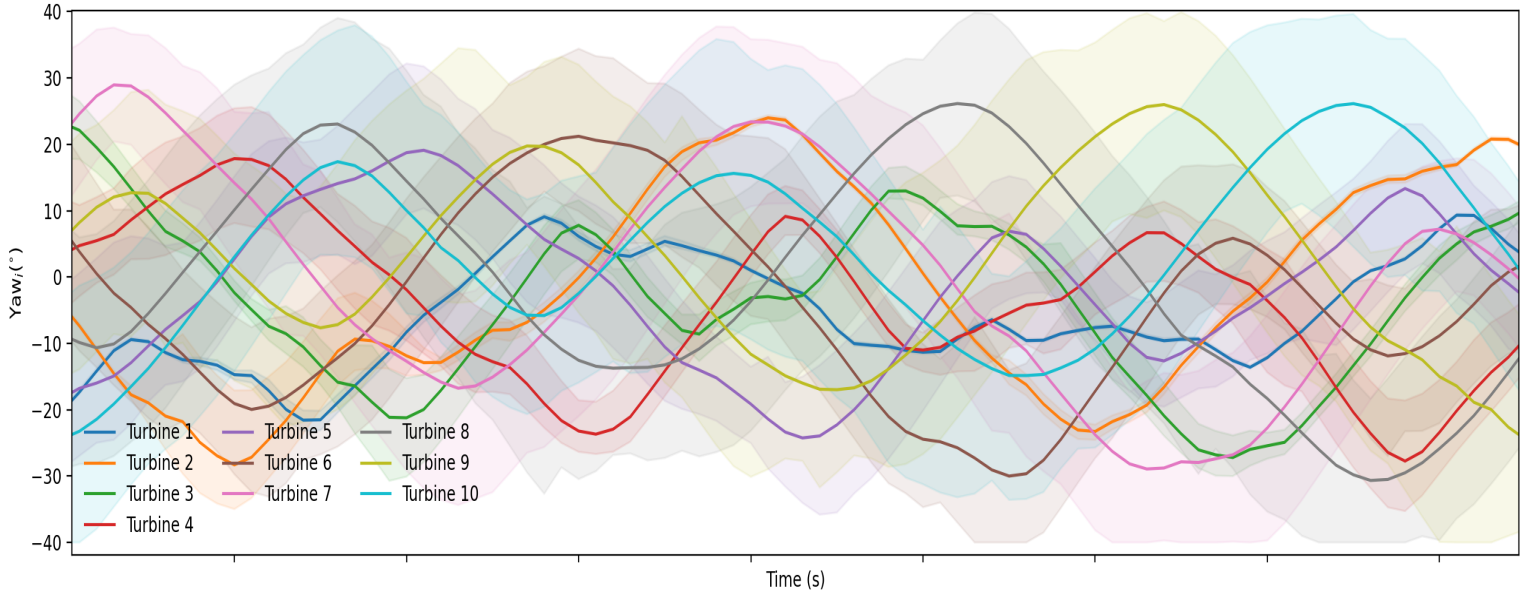
Initial Results

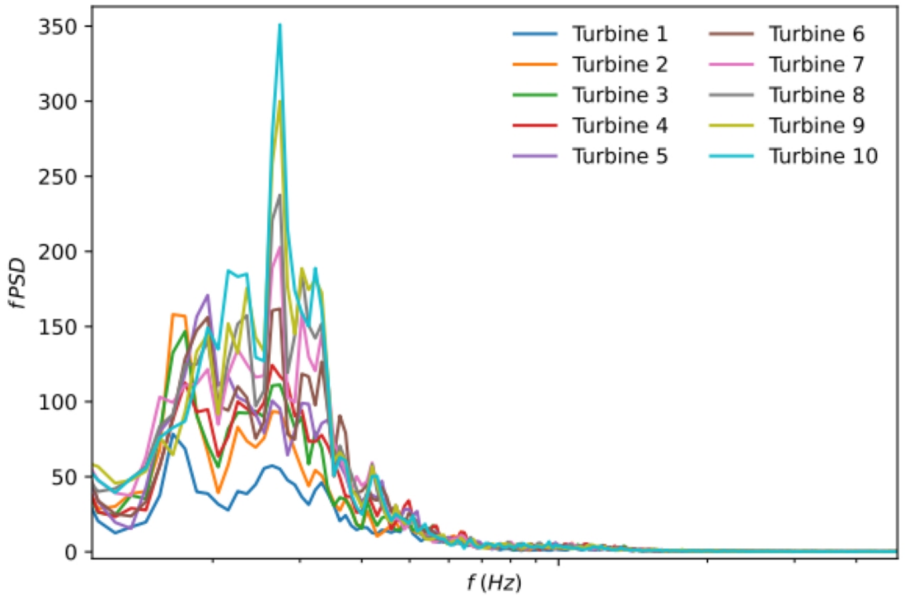


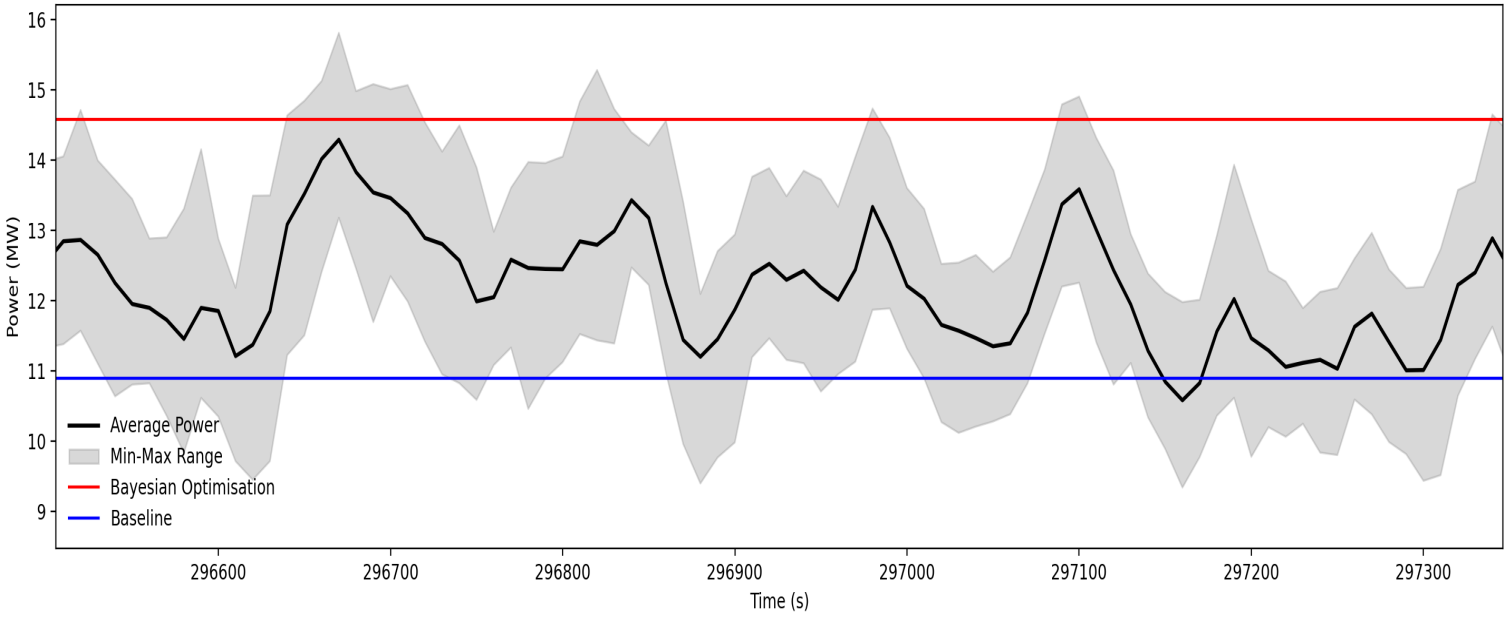
Initial Results

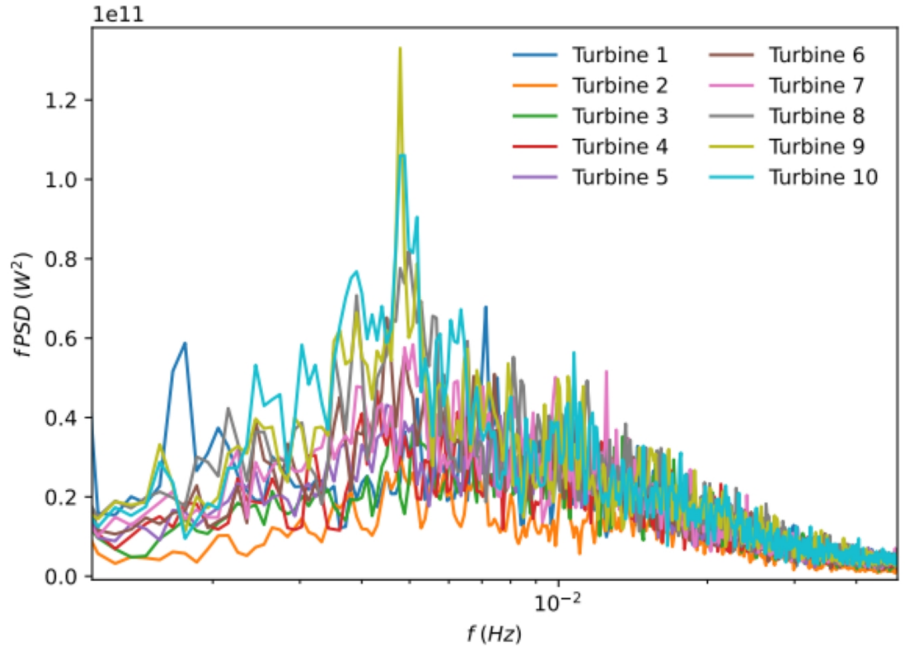


Initial Results









Summary

- **Reinforcement Learning** being tested for exploring dynamic and active control.
- Including the **physics provided from LES** in the RL environment.
- **SmartSim** allows for efficient coupling of high-fidelity environment with RL on HPC.