Parallelised Optimization with BioDynaMo

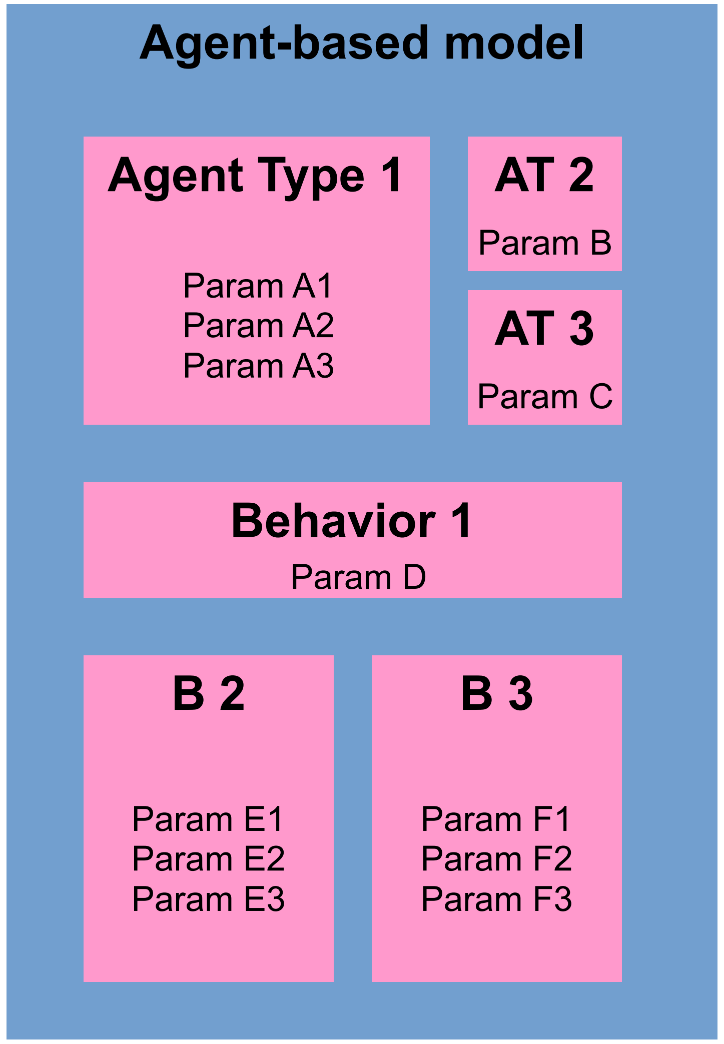
Ahmad Hesam

19-11-2021

BioDynaMo Workshop

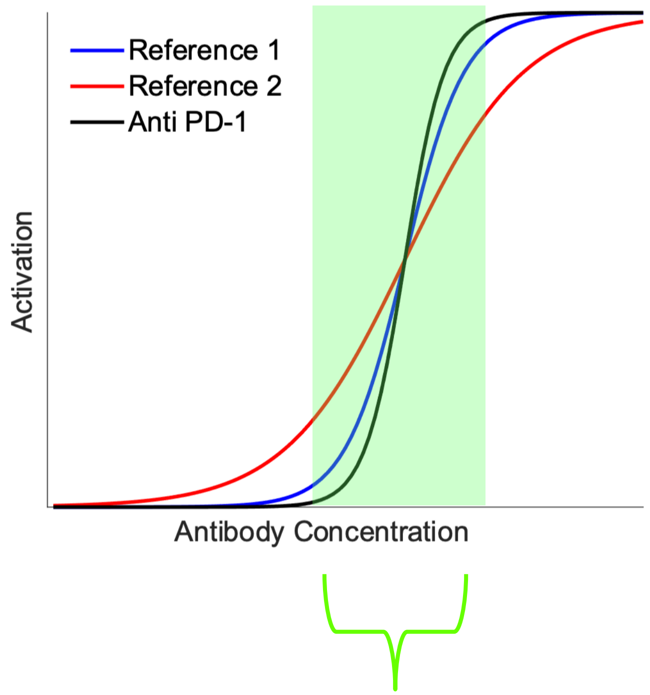
Challenge in Agent-Based Modelling

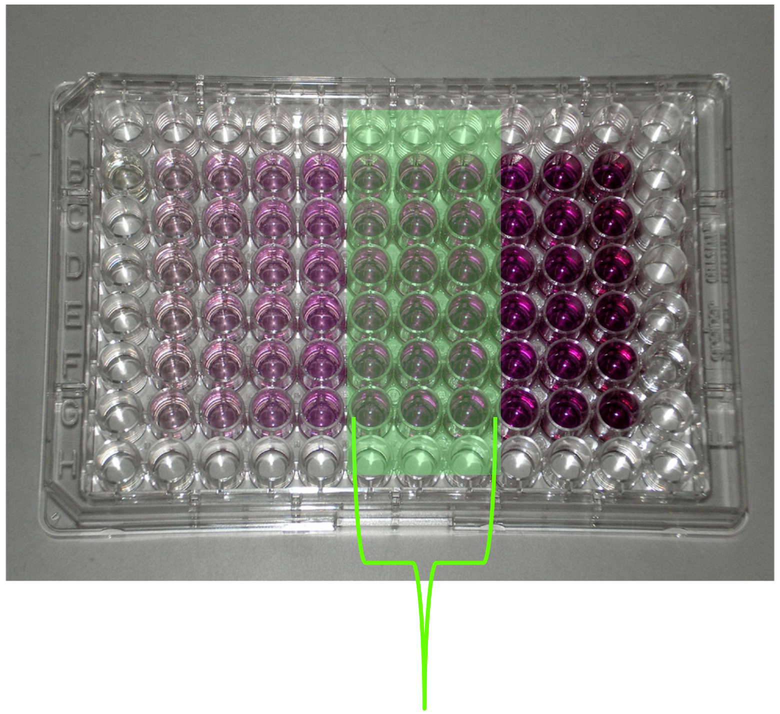
- My model has a large number of parameters: what should their values be?
- I want to evaluate my model for a certain parameter space: how do I do so efficiently?
- How can I train my model to follow a real-life dataset?



Real-Life Example: Immunobrain Checkpoint

- Immunobrain Checkpoint Ltd. (Tel-Aviv, Israel)
- Develops immunotherapy treatments for combatting Alzheimer's

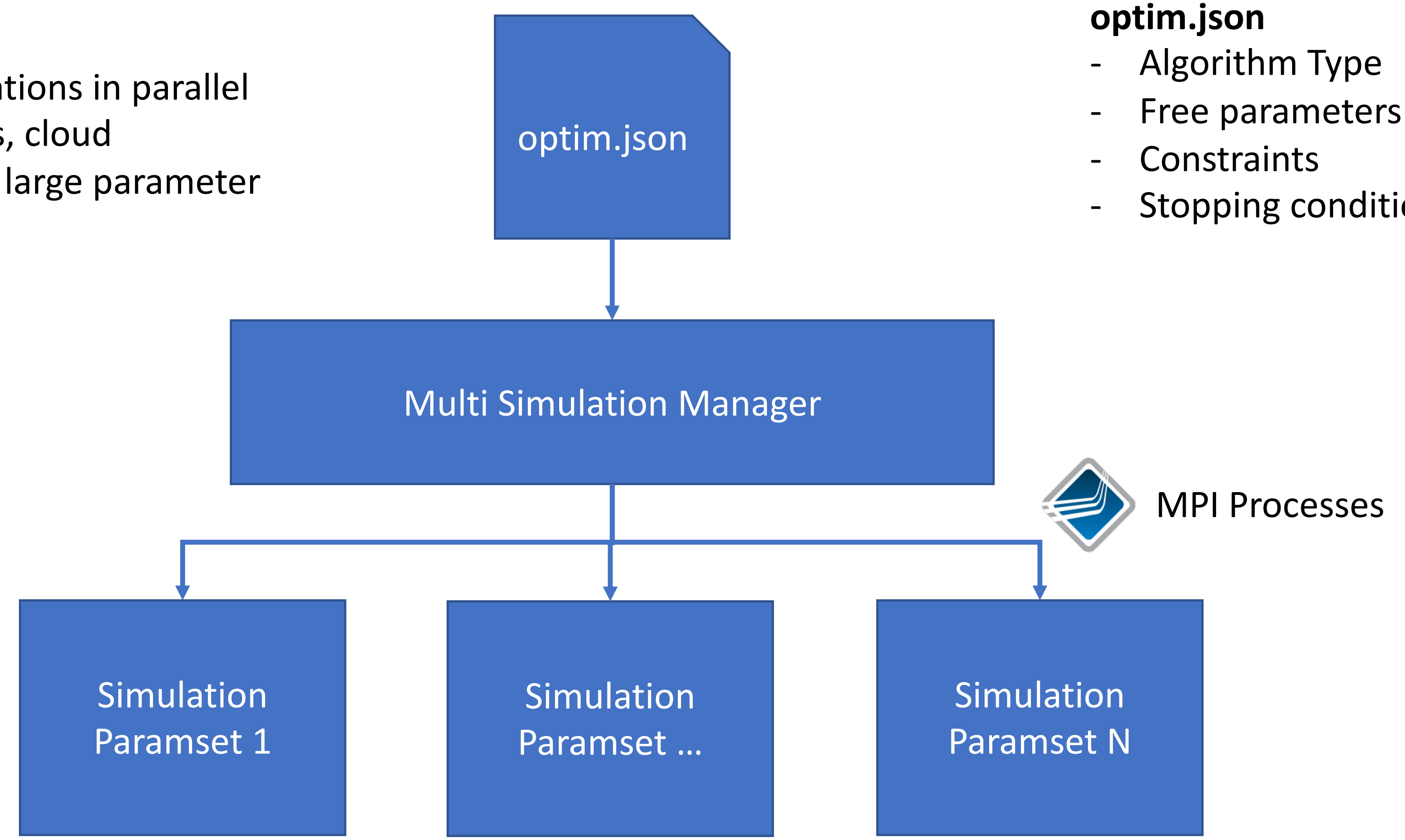




Simulation with BioDynaMo should tell lab researchers what the interesting samples are

Multi-Simulation Runtime

- Run multiple simulations in parallel
- Scale out to clusters, cloud
- Rapid evaluation of large parameter spaces



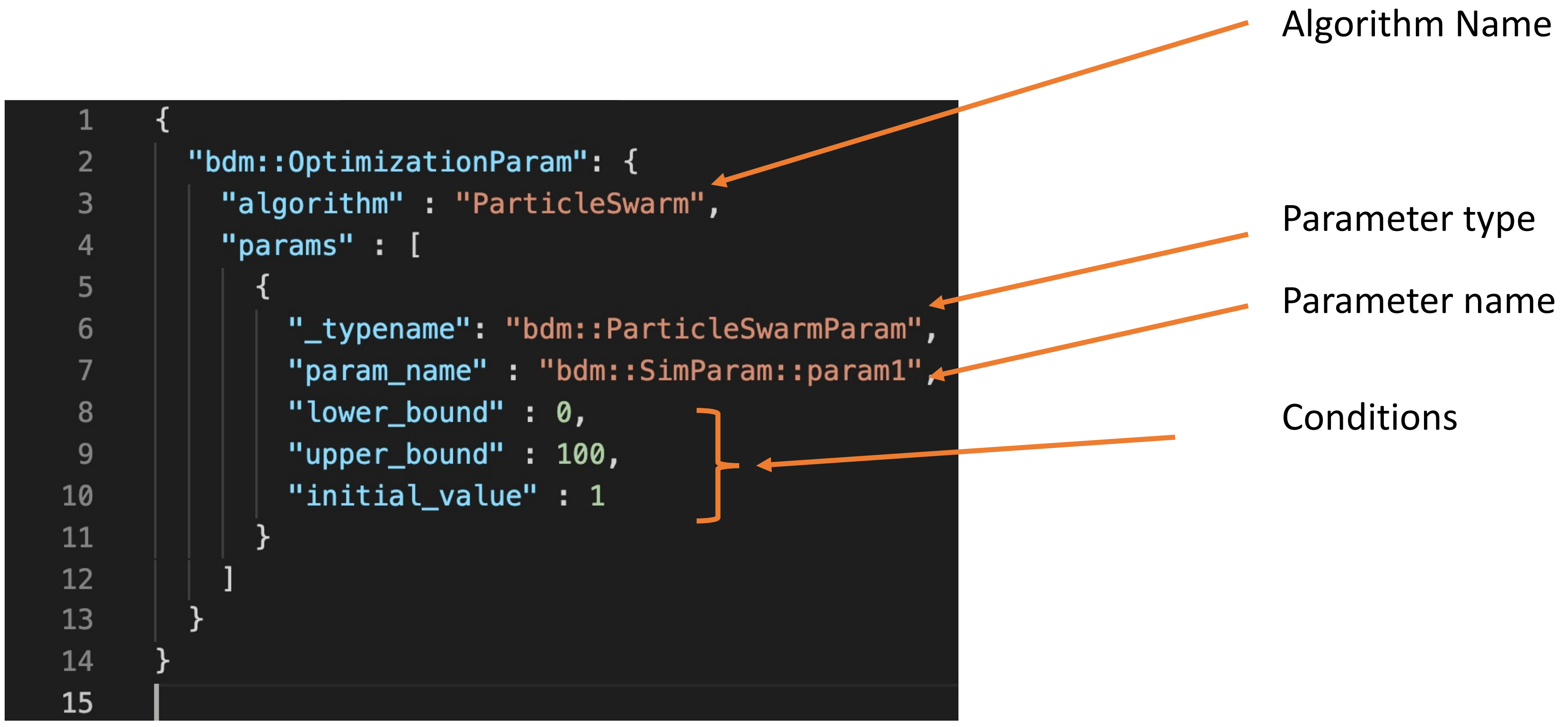
optim.json
- Algorithm Type
- Free parameters
- Constraints
- Stopping condition

How to use

(Full description: https://biodynamo.org/docs/userguide/multi_simulation/)

1. Select an algorithm, and choose your free parameters
2. Depending on the algorithm, load your real-life data
3. Adjust your simulation code (slightly) to run in multi-simulation mode
4. Execute your binary with `mpirun`

Walkthrough

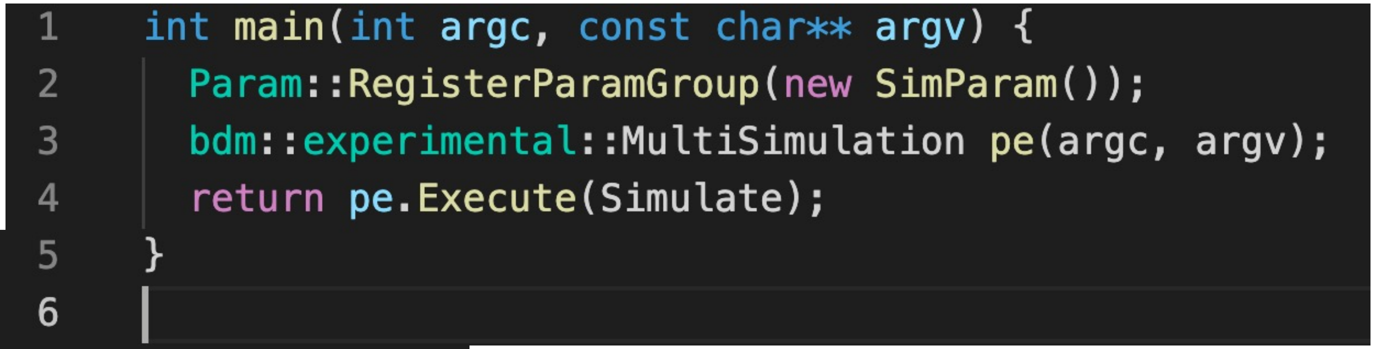


```
{
  "bdm::OptimizationParam": {
    "algorithm" : "ParticleSwarm",
    "params" : [
      {
        "_typename": "bdm::ParticleSwarmParam",
        "param_name" : "bdm::SimParam::param1",
        "lower_bound" : 0,
        "upper_bound" : 100,
        "initial_value" : 1
      }
    ]
  }
}

```

optim.json

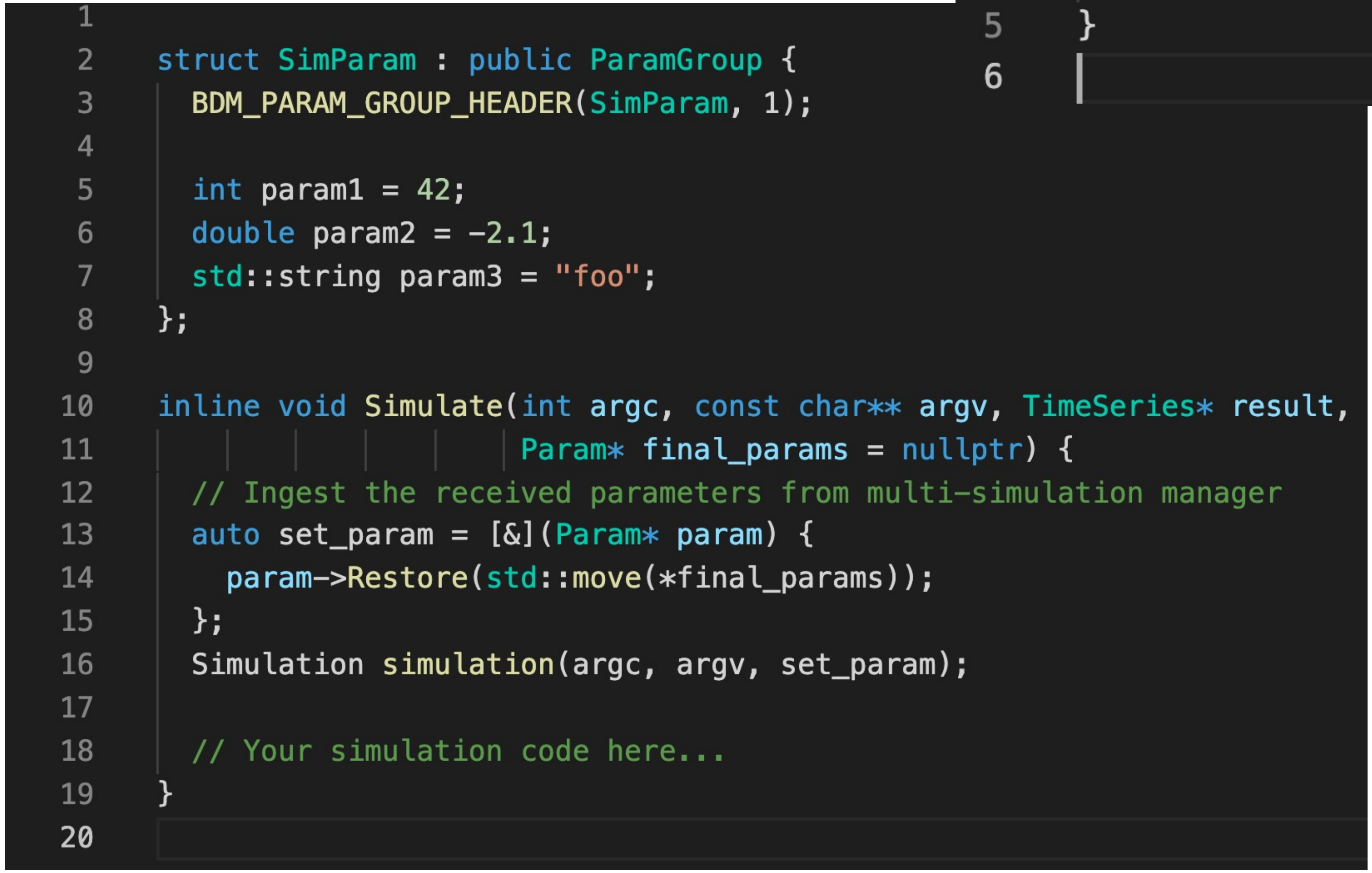
Walkthrough



```cpp
int main(int argc, const char** argv) {
  Param::RegisterParamGroup(new SimParam());
  bdm::experimental::MultiSimulation pe(argc, argv);
  return pe.Execute(Simulate);
}

```

my_sim.cc



```cpp

struct SimParam : public ParamGroup {
  BDM_PARAM_GROUP_HEADER(SimParam, 1);

  int param1 = 42;
  double param2 = -2.1;
  std::string param3 = "foo";
};

inline void Simulate(int argc, const char** argv, TimeSeries* result,
                     Param* final_params = nullptr) {
  // Ingest the received parameters from multi-simulation manager
  auto set_param = [&](Param* param) {
    param->Restore(std::move(*final_params));
  };
  Simulation simulation(argc, argv, set_param);

  // Your simulation code here...
}

```

my_sim.h

Walkthrough

Command line execution

```
$ mpirun -np <num_procs> -H <hostfile> ./my_sim --config=optim.json
```

# of processes

List of servers

Simulation binary

Optimization parameter file

Hostfile

```
host1.example.com
host2.example.com
host3.example.com
```

With MPI you are capable of finetuning the number of threads / cores per host

Adding an Optimization Algorithm

<u>Currently available</u>

- **ParameterSweep**: performs an exhaustive sweep of predefined ranges and sets of parameters
- **ParticleSwarm**: performs a particle swarm optimization (backend: optimlib) with user-defined error matrices

Add your own optimization algorithm by inheriting from `bdm::experimental::Algorithm`

and adding it to the AlgorithmRegistry (see multi_simulation/algorithm_test.cc for example)

QUESTIONS?