SimParm

Simple and flexible C++ configuration framework

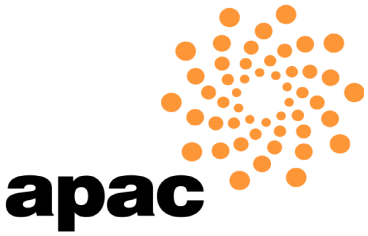
Kevin Pulo
kevin.pulo@anu.edu.au
http://www.kev.pulo.com.au/simparm/

APAC National Facility, Australian National University
Canberra, ACT, Australia

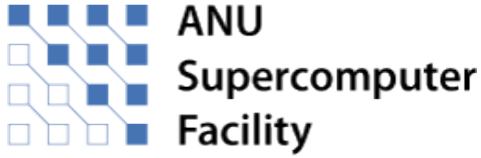
ANU Supercomputer Facility

THE AUSTRALIAN NATIONAL UNIVERSITY

apac



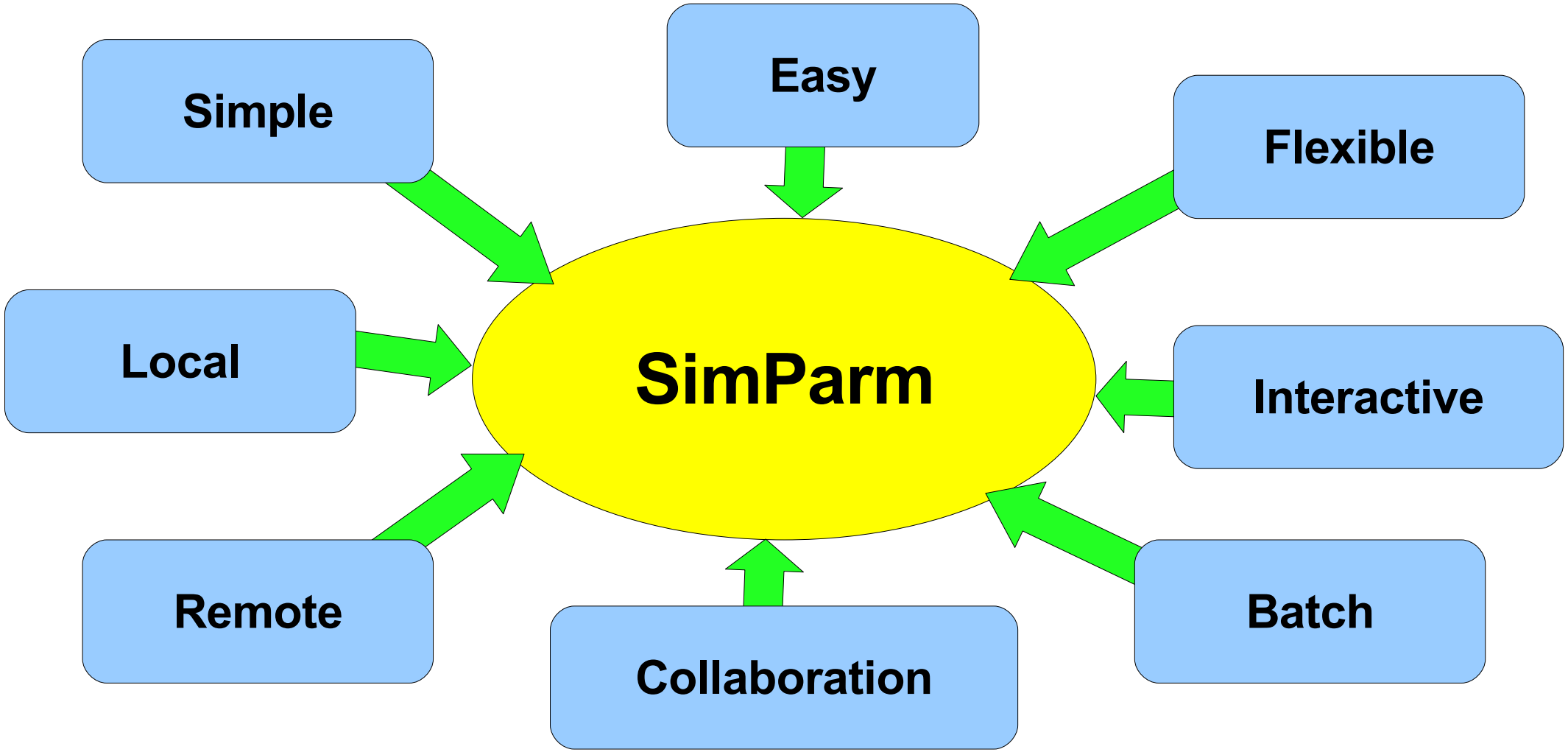
Motivation

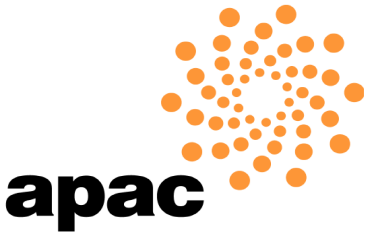


- **Problem:** Runtime configuration of simulation parameters is often overlooked
- **Results:**
  - Worst-case: Hard-coded parameters
  - Development time spent on configuration system, not simulation code
  - Configuration systems are often adhoc, poorly-defined, difficult to modify, clumsy to use, etc
- **Example:** Virtually every major computational package has a different input file format

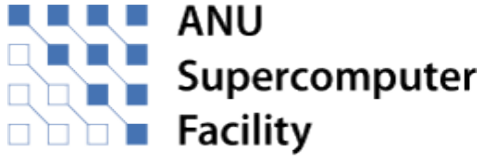
Solution

SimParm addresses these problems

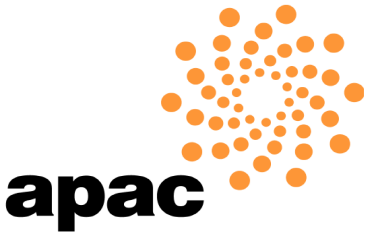




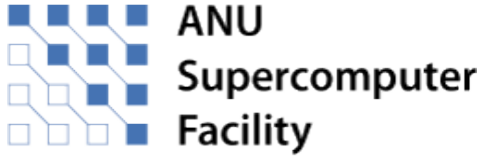
Features



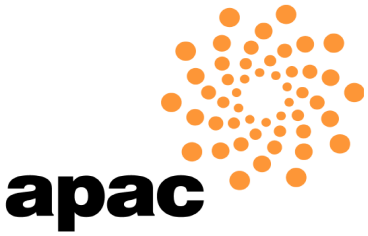
- Portable C++ library
- Values are easily used in simulation code
- Parameter definitions are easy and simple
- Simple ASCII configuration file format
- Parameters are statically typed and dynamically accessible
- Parameter types are extensible (object-oriented)
- Interactive real-time parameter adjustment
  - Including simulations running remotely
- Remote collaboration of multiple researchers



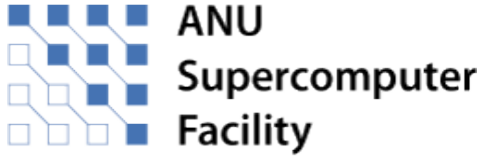
Definition



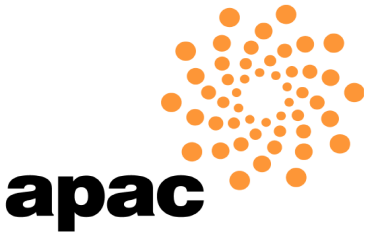

```
#include "ConfigSet.hh"

class ConfigSetRelax : public ConfigSet {
    public:
        ConfigEntryDouble timestep;
        ConfigEntryUnsignedLong max_t;
        ConfigEntryDouble epsilon;
        ConfigSetRelax();
};
```



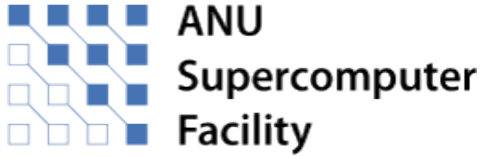
Constructor



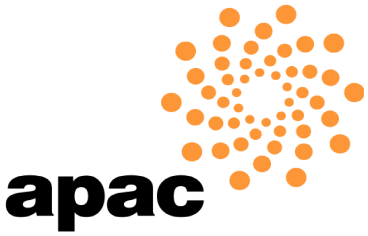

```
ConfigSetRelax::ConfigSetRelax() : ConfigSet() {
    timestep.setName("timestep");
    timestep.setDesc("Relax timestep");
    timestep = 1.0;
    register_entry(&timestep);

    max_t.setName("max_t");
    max_t.setDesc("Relax max timesteps");
    max_t = 1000000;
    register_entry(&max_t);

    epsilon.setName("epsilon");
    epsilon.setDesc("Relax epsilon");
    epsilon = 0.1;
    register_entry(&epsilon);
}
```



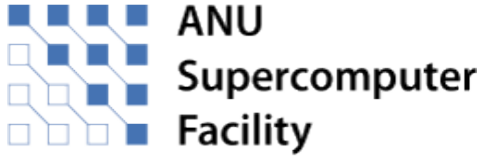
Inheritance



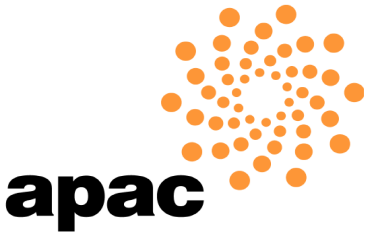

```
class ConfigSetRunControl : public ConfigSet {
    public:
        ConfigEntryBool running;
        ConfigEntryBool finished;
        ConfigEntryBool autorestart;
        ConfigSetRunControl();
};

class ConfigSetApplication : public ConfigSetRelax,
                             public ConfigSetRunControl {
    public:
        ConfigEntryString input_filename;
        ConfigEntryString output_filename;
        ConfigSetApplication();
};
```



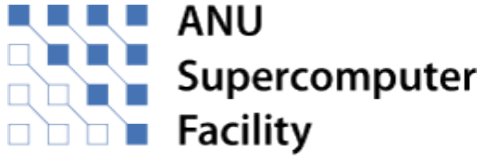
Usage



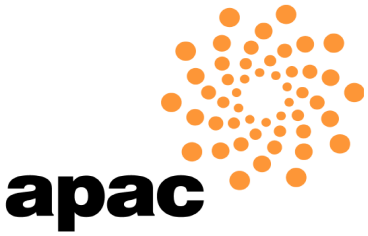

```
// Declare as usual:
ConfigSetApplication config;

// Access value with operator():
double e = config.epsilon();

// For example:
for (long t = 0; t < config.max_t(); t++) {
    computeForces();
    actionForces(config.timestep());
    if (errorsum < config.epsilon())
        break;
    if (errorHasWorsened())
        config.timestep *= 0.9;
}
```



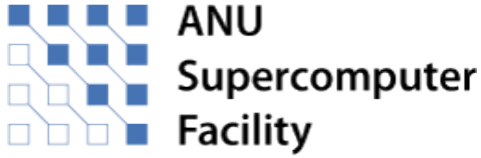
Parameter Types



- Floating-point and integer types
  - `ConfigEntryFloat`, `ConfigEntryDouble`
  - `ConfigEntryInt`, `ConfigEntryUnsignedInt`
  - `ConfigEntryLong`, `ConfigEntryUnsignedLong`
  - etc
- `ConfigEntryBool` for boolean choices
- `ConfigEntryString` for string values
- `ConfigEntryChoice` for enumerated types (ie. Choose from a set of possible options)

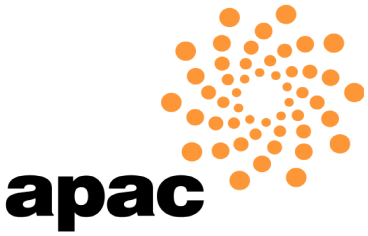


Choice Example

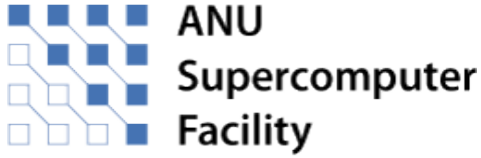



```
namespace Convergence { enum {
    nothing, exit, reduce
}; }
ConfigEntryChoice convergence;
```

```
convergence.setName("convergence");
convergence.setDesc("Convergence action");
convergence.addChoice(Convergence::nothing,
                      "nothing", "Nothing");
convergence.addChoice(Convergence::exit,
                     "exit", "Exit program");
convergence.addChoice(Convergence::reduce,
                   "reduce", "Reduce timestep");
convergence = Convergence::nothing;
register_entry(&convergence);
```

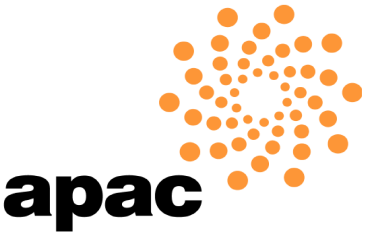


Choice Example

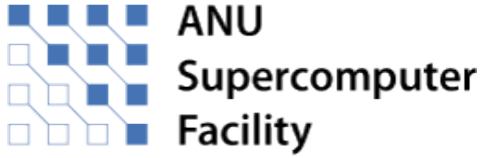


- Now the convergence test becomes:

```
if (errorsum < config.epsilon()) {
    switch(config.convergence()) {
        case Convergence::nothing:
            break;
        case Convergence::exit_program:
            exit(0);
        case Convergence::reduce_timestep:
            config.timestep *= 0.9;
            break;
    }
}
```



Config File Format



- Simple format: plain text, line-based

```
<name> = <value>
```

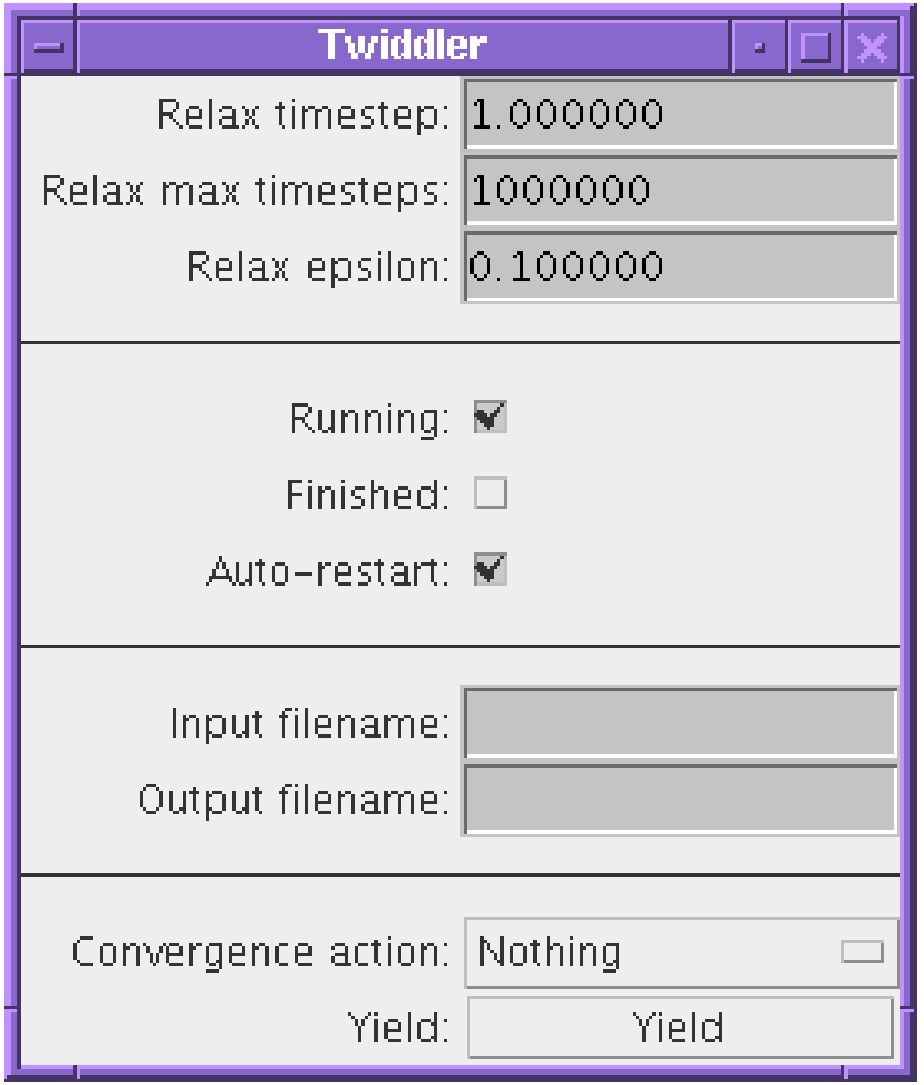
- For example:

```
timestep = 1.5
max_t = 5000
epsilon = 0.001
running = true
finished = false
autorestart = true
input_filename = input.dat
output_filename = output.dat
convergence = exit_program
```

Interactivity

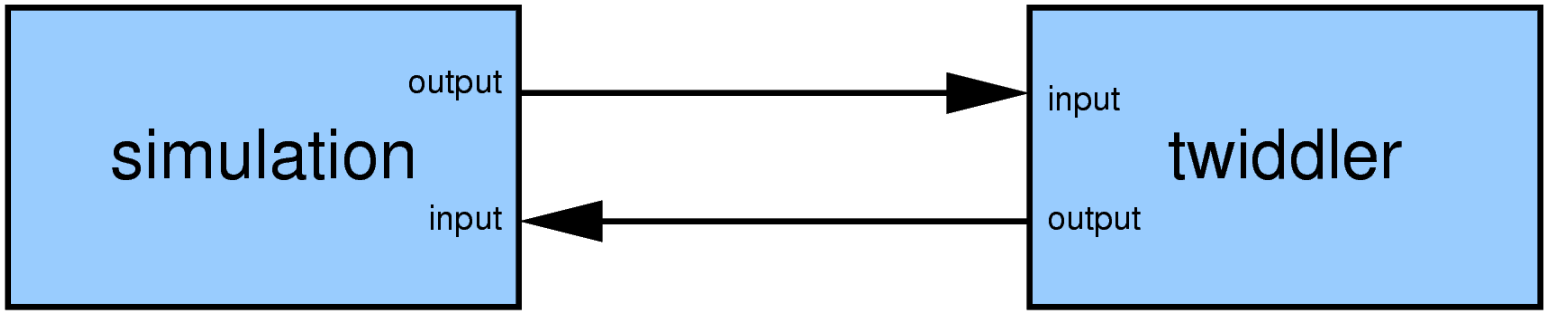
Twiddler: Java-based parameter GUI

- `ConfigEntryTrigger` allows button actions to trigger simulation events
- `ConfigEntryDivider` allows grouping of parameters

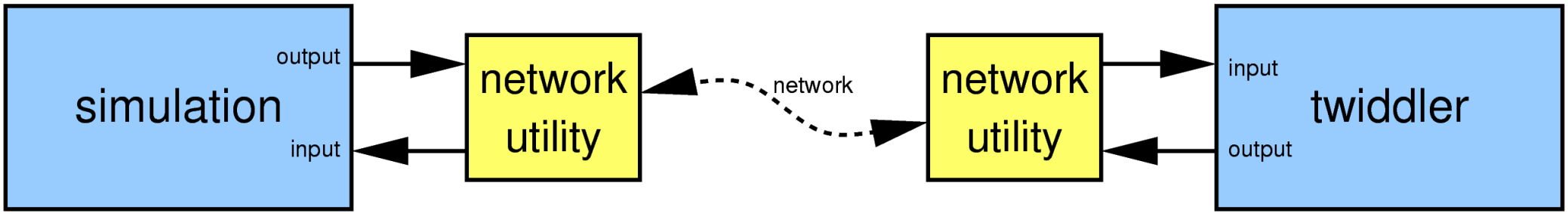


Remote Interactivity

- Twiddler is connected to simulation via ASCII communication over pipes

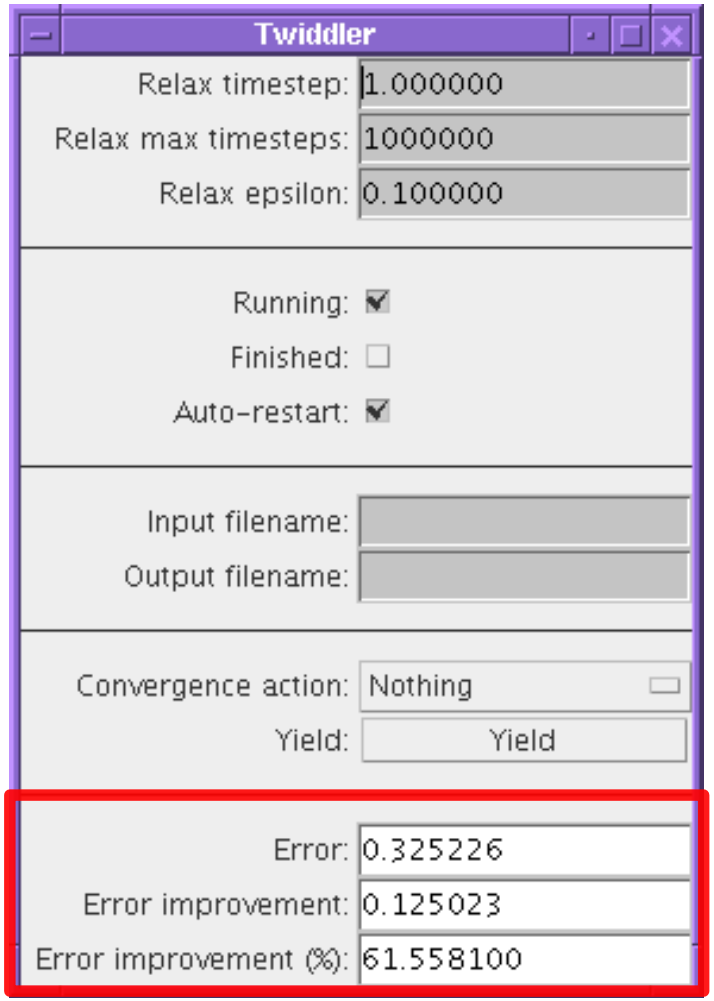


- Allows remote interactivity over network
  - For example, using ssh, netcat, etc

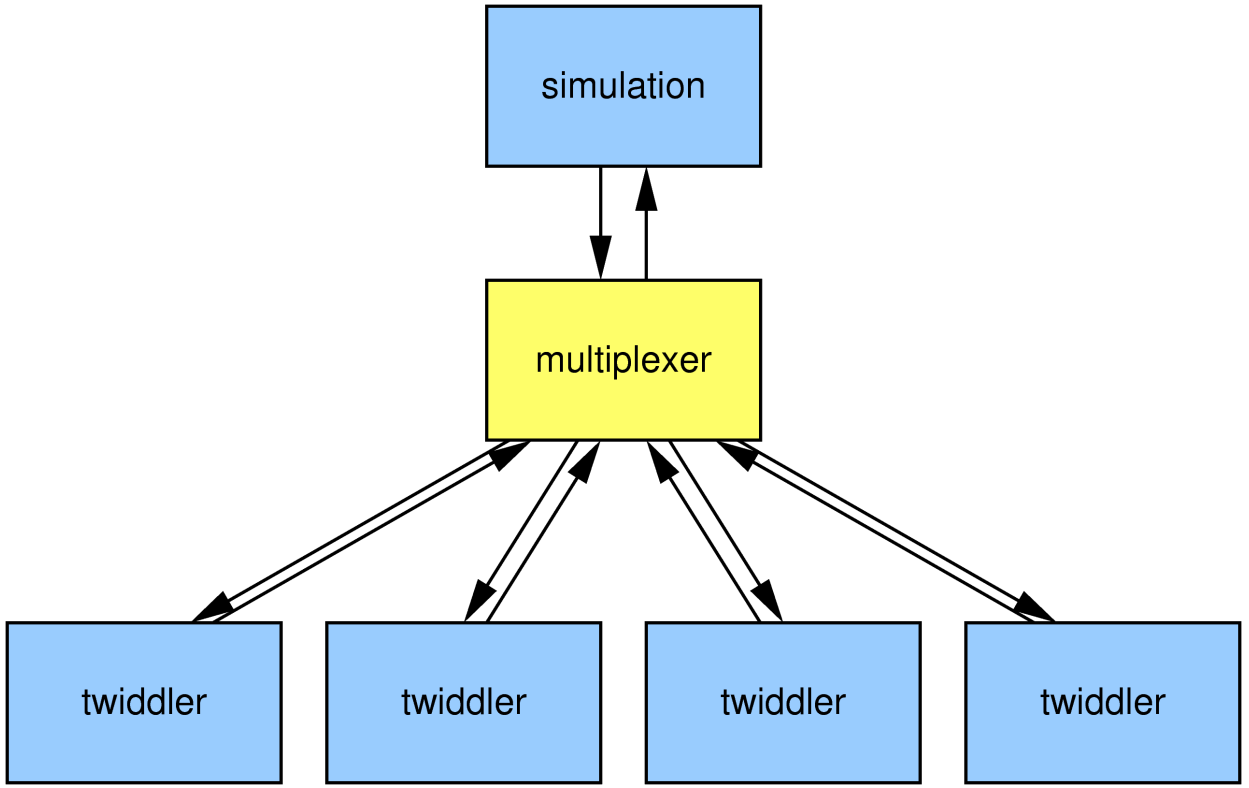


Collaboration

- Read-only parameters to monitor simulation



- Multiple twiddlers connected to simulation via multiplexer (in Python) and network



Sample Application geomslab

Interactive demonstration