Unified diff: `step3_trivial_mc` *vs.* `step4_pi`

```
Files step3_trivial_mc/CMakeLists.txt and step4_pi/CMakeLists.txt are identical

diff -N -u -r -b -s step3_trivial_mc/main.cpp step4_pi/main.cpp
--- step3_trivial_mc/main.cpp   2016-06-15 14:36:21.094853744 -0400
+++ step4_pi/main.cpp   2016-06-15 14:36:39.478852887 -0400
@@ -28,10 +28,34 @@
     mysim.run(alps::stop_callback(std::size_t(p["timelimit"])));

     std::cout << "Simulation finished"
+              << std::endl
+              << "Collecting results..."
               << std::endl;
+
+    alps::accumulators::result_set results=mysim.collect_results();
+
+    // Print all results:
+    std::cout << "All results:\n" << results << std::endl;
+
+    // Access individual results:
+    const alps::accumulators::result_wrapper& obj=results["objective"];
     std::cout << "Simulation ran for "
-              << mysim.count()
+              << obj.count()
               << " steps." << std::endl;

+    // should get $\pi$:
+    const alps::accumulators::result_wrapper& pi_res=obj*4.;
+
+    std::cout << "Mean: " << pi_res.mean<double>() << std::endl;
+    std::cout << "Error: " << pi_res.error<double>() << std::endl;
+    std::cout << "Range: "
+              << pi_res.mean<double>()-pi_res.error<double>()
+              << " ... "
+              << pi_res.mean<double>()+pi_res.error<double>()
+              << std::endl;
+    std::cout << "Autocorrelation length: "
+              << pi_res.autocorrelation<double>()
+              << std::endl;
+
     return 0;
 }


diff -N -u -r -b -s step3_trivial_mc/simulation.cpp step4_pi/simulation.cpp
--- step3_trivial_mc/simulation.cpp     2016-06-15 12:00:13.967290368 -0400
+++ step4_pi/simulation.cpp     2016-06-15 12:31:08.191203939 -0400
Files differ
```

```
diff -N -u -r -b -s step3_trivial_mc/simulation.hpp step4_pi/simulation.hpp
--- step3_trivial_mc/simulation.hpp     2016-06-15 12:07:16.203270687 -0400
+++ step4_pi/simulation.hpp     2016-06-15 12:27:23.359214419 -0400
@@ -1,14 +1,25 @@
 #pragma once

 #include <alps/mc/mcbase.hpp>
+#include <alps/accumulators.hpp>

 class MySimulation : public alps::mcbase {
   private:
-    int istep_;
-    int maxcount_;
-    bool verbose_;
+    long burnin_;
+    long maxcount_;
+    double stepsize_;
+
+    long istep_;
+    double x_,y_;

   public:
+    // These we need for the simulation.
+    static bool is_inside_area(double x, double y);
+    static double objective_function(double x, double y);
+
+    // Accumulator type to collect observables.
+    typedef alps::accumulators::FullBinningAccumulator<double> my_accumulator_type;
+
     MySimulation(const parameters_type& params, std::size_t seed_offset=0);

     void update();
@@ -16,6 +27,4 @@
     double fraction_completed() const;

     static parameters_type& define_parameters(parameters_type&);
-
-    int count() { return istep_; }
 };
```