Quick start with imcmc

Youhua Xu[1, 2, *]
[1] *School of Physics, Nanjing University, 22 Hankou Road, Nanjing, Jiangsu 210093, China*
[2] *Key Laboratory of Space Astronomy and Technology,*
National Astronomical Observatories, CAS, Beijing 100012, China
(Dated: January 14, 2017)

This document demonstrates how to use the MCMC library – **imcmc**, with a few simple examples.

I. WHY BUILD A LIBRARY?

It's a nightmare when if you're managing a project with lots of source files.

II. BASICS: BAYESIAN ANALYSIS & MCMC

Before getting start to generate MC samples with imcmc, I will brieflly review the basics of Bayesian analysis and MCMC (use Metropolis-Hastings algorithm as the example)

III. EXAMPLES

A. Mutil-Gaussian

,label="../examples/gaussian.cpp",style=Style1

```
#include "ensemble.hpp"
using namespace std;
using namespace imcmc;

struct Gaussian{    //  test model
    imcmc_double p;

    void AddParam( imcmc_vector_string& param ){
        imcmc_vector_string_iterator it = param.begin();
        while( it != param.end() ){
            p[*it] = 0;
            ++it;
        }
    }

    void Update( imcmc_double full_param ){
        imcmc_double_iterator it = p.begin();
        while( it != p.end() ){
            p[it->first] = full_param[it->first];
            ++it;
        }
    }

    double GD(){
        double chisq = 0;
```

[*] Electronic address: yhxu@nao.cas.cn

```
        imcmc_double_iterator it = p.begin();
        double i=1;
        while( it != p.end() ){
            chisq += p[it->first]*p[it->first]/(i*i);
            i += 1.;
            ++it;
        }
        return chisq;
    }
};

double TestLike( imcmc_double&  full_param,
                 double&        lndet,
                 double&        chisq,
                 void*          model,
                 void*          data,
                 istate&        state ){

    lndet = chisq = 0;

    Gaussian *g = static_cast<Gaussian *>(model);

    full_param[”x+y”] = full_param[”x”] + full_param[”y”];  //
new added for test derived parameters

    // state.this_like_is_ok = true;
    // state.store_mesg(”nothing happened!”);

    //  how to pass error information to imcmc::ensemble_workspace
    if( full_param[”x”] < -5.0 || full_param[”x”] > 5.0 ){

        state.this_like_is_ok = false;
        state.store_mesg(”   fabs(x) is larger than 5, this should not happen!”);

        chisq = _IMCMC_CHISQ_MAX_;
    }
    else{
        g->Update(full_param);  //  now the model is workable

        chisq = g->GD();
    }

    return -0.5*chisq;
}

int main( int argc, char *argv[] )
{
    MPI::Init(argc, argv);

    ensemble_workspace ew;

    imcmc_vector_string param;
    param.push_back(”x”);
    param.push_back(”y”);
    param.push_back(”z”);

    imcmc_vector_string dparam;
    dparam.push_back(”x+y”);
    Gaussian g;
    g.AddParam(param);

//    ew.add_likelihood( TestLike, param, &g, NULL );
    ew.add_likelihood( TestLike, param, dparam, &g, NULL );
    ew.init("gaussian.ini");
    ew.do_sampling();

    MPI::Finalize();

}
```

B. CMB

Appendix A: Parser

Usually one will use may parameters in her/his codes, which might be model parameters, names/paths of data and even precision controlling parameters, thus a well-designed parser is very helpful.