HMC example

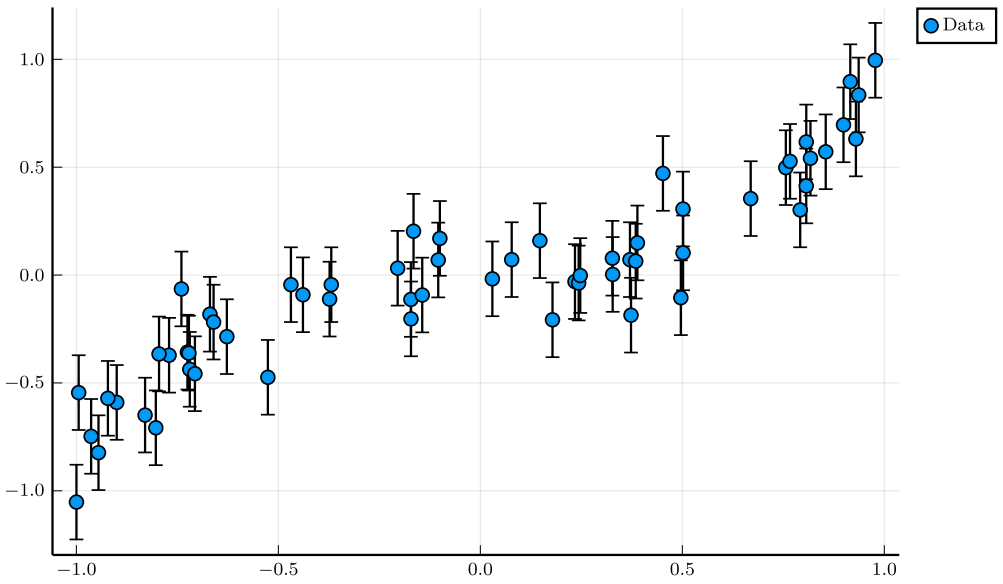
Alberto ramos <alberto.ramos@maths.tcd.ie>
Wed Apr 6 19:28:17 2022

Contents

Code (Julia)
```julia
using Random, Plots, Printf, DelimitedFiles, Statistics
pgfplotsx();
```

First read the data, and plot it

Code (Julia)
```julia
x = readdlm("data_x.dat")
y = readdlm("data_y.dat")
dy = [sqrt(0.03) for i in 1:length(x)]

pl = plot(x, y, yerr=dy, seriestype=:scatter, label="Data")
display(pl)
```



Now we code the HMC

Code (Julia)
```julia
mlogp(x, y, dy, dpr, th) = 0.5*sum( (y .- (th[1] .+ th[2] .*x .+ th[3] .*x.^2 .+ th[4] .* x .^3)).^
    0.5*sum(th.^2 ./ dpr)

println("Computation of -log(p): ", mlogp(x, y, dy, 1.4, [0.0,0.0,0.0,0.0]))

function force!(frc, x, y, dy, dpr, th)

```

```
    for i in 1:length(frc)
        frc[i] = -sum( (y .- (th[1] .+ th[2] .*x .+ th[3] .*x.^2 .+ th[4] .* x .^3)) .* x.^(i-
    end

    return nothing
end

println("Test force")
h = 1.0E-5
th = randn(4)
pp = deepcopy(th)
pm = deepcopy(th)
frc = deepcopy(th)
for i in 1:length(th)
    pp[i] = th[i] + h
    pm[i] = th[i] - h

    mlp = mlogp(x, y, dy, 1.4, pp)
    mlm = mlogp(x, y, dy, 1.4, pm)
    println("Coordinate ", i, "  ", (mlp-mlm)/(2*h))
    pp[i] = th[i]
    pm[i] = th[i]
end
force!(frc, x, y, dy, 1.4, th)
println(frc)
```

```
Computation of -log(p): 191.6059813166831
Test force
Coordinate 1  -2520.08601498801
Coordinate 2  -1056.1571816879223
Coordinate 3  -897.6243061169952
Coordinate 4  -711.1499071470461
[-2520.086014994195, -1056.1571816047624, -897.6243061174312, -711.1499071628045]
```

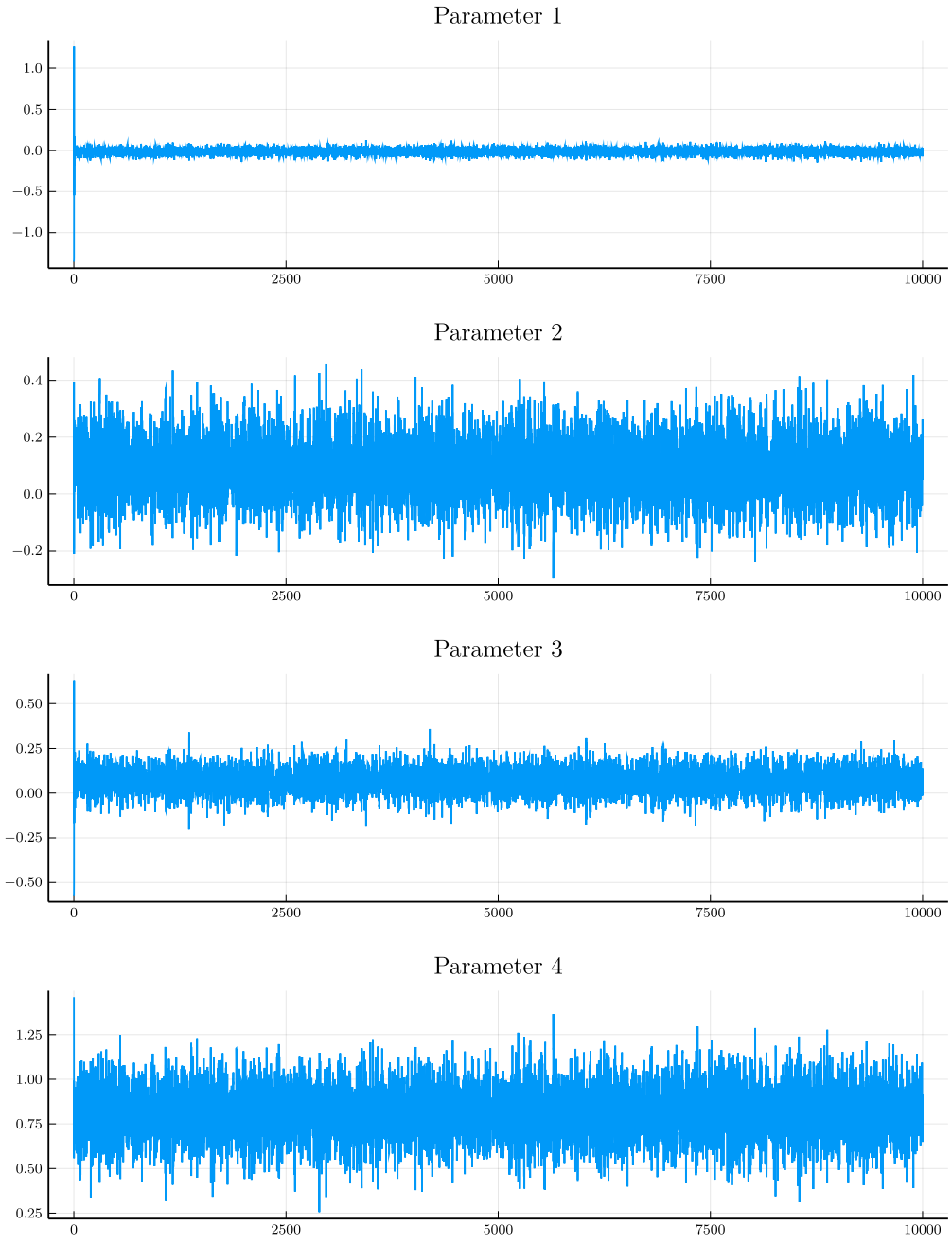
Now we need to integrate equations of motion

Code (Julia)

```
function MD!(p, th, x,y,dy,dpr, ns, ep)

    frc = similar(p)
    for i in 1:ns
        force!(frc, x, y, dy, dpr, th)
        p .= p - (ep/2) .* frc
        th .= th + ep .* p
        force!(frc, x, y, dy, dpr, th)
        p .= p - (ep/2) .* frc
    end

    return nothing
end

```

```julia
hamiltonian(p, th, x,y,dy,dpr) = sum(p.^2)/2 + mlogp(x, y, dy, dpr, th)

function HMC!(th, x,y,dy,dpr,ns,ep)

    thcp = deepcopy(th)
    p = randn(4)
    hini = hamiltonian(p, th, x,y,dy,dpr)
    MD!(p,th,x,y,dy,dpr,ns,ep)

    # Now Metropolis accept-reject
    dh   = hamiltonian(p, th, x,y,dy,dpr) - hini
    pacc = exp(-dh)

    acc = true
    if pacc < 1.0
        r = rand()
        if pacc < r
            th  .= thcp
            acc = false
        end
    end

    return dh, acc
end

nt = 10000
samp = Array{Float64, 2}(undef, nt, 4)
th = randn(4)
ns = 100
ep = 0.01
let nacc = 0
    for i in 1:nt
        nas = rand(1:ns)
        dh, acc = HMC!(th, x,y,dy,1.4,nas,ep)
        for k in 1:4
            samp[i,k] = th[k]
        end
        if acc
            nacc = nacc + 1
        end
    end
    println("\nAcceptance: ", 100*nacc/nt)
end

pl = Vector{Any}()
for k in 1:4
    push!(pl, plot(samp[:,k], title="Parameter $k", label=""))
end
plt = plot(pl..., layout=(4,1), size=(800,1000))
```

```
Acceptance: 98.66
```



Print final values of parameters. We discard 100 measurements for thermalization. Compute average and standard deviation for each parameter.

Code (Julia)

```julia
for k in 1:4
    avg = mean(samp[100:end, k])
    err = std(samp[100:end, k])
    println("# Parameter $k: ", avg, " +/- ", err)
end
```

```
# Parameter 1: -0.013688787530637393 +/- 0.03618949840492048
```

```
# Parameter 2: 0.09154458086374177 +/- 0.09982304230135046
# Parameter 3: 0.06681354389357536 +/- 0.06975960933845657
# Parameter 4: 0.8079708959144705 +/- 0.14259248400961697
```

It seems that all fit parameters are compatible with zero, except the cubic term...