Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. doi: 10.18637/jss.v000.i00

Piecewise Deterministic Markov Processes in Julia: PDMP.jl

Romain Veltz
INRIA

Simon Frost
University of Cambridge

Abstract

Simulation of stochastic processes is an important part of modeling and inference in many biological fields, including neuroscience, epidemiology population genetics, and systems biology. We present **PDMP.jl**, a package written in Julia for the efficient simulation of stochastic processes that comprise of both a discrete and a continuous component.

Keywords: Markov processes, simulation, Julia.

1. Introduction

The True Jump Method (TJM, Veltz (2015)) is an algorithm for the simulation of PDMPs.

2. The PDMP.jl package

The **PDMP.jl** package is written in the Julia programming language.

The True Jump Method involves solving stiff ordinary differential equations. We use the CVODE routine available in the **Sundials.jl** package; in preliminary analyses, CVODE was approximately twice as fast as the ode23s routine, implemented in native Julia code in the **ODE.jl** package.

3. Examples

The interface to the simulations closely resembles that used in the R package **GillespieSSA** (Pineda-Krch 2008).

3.1. The Morris-Lecar model

```
using PDMP, JSON, GR

const p0  = convert(Dict{AbstractString,Float64}, JSON.parsefile("ml.json")["type II"])
const p1  = ( JSON.parsefile("ml.json"))
include("morris_lecar_variables.jl")
const p_ml = ml(p0)

function F_ml(xcdot::Vector{Float64}, xc::Vector{Float64},xd::Array{Int64},t::Float64, par
  # vector field used for the continuous variable
  #compute the current, v = xc[1]
  xcdot[1] = xd[2] / p_ml.N * (p_ml.g_Na * (p_ml.v_Na - xc[1])) +
    xd[4] / p_ml.M * (p_ml.g_K  * (p_ml.v_K  - xc[1]))  +
    (p_ml.g_L  * (p_ml.v_L  - xc[1])) + p_ml.I_app
  nothing
end

function R_ml(xc::Vector{Float64},xd::Array{Int64},t::Float64, parms::Vector, sum_rate::Bo
  if sum_rate==false
    return vec([p_ml.beta_na * exp(4.0 * p_ml.gamma_na * xc[1] + 4.0 * p_ml.k_na) * xd[1],
                p_ml.beta_na * xd[2],
                p_ml.beta_k * exp(p_ml.gamma_k * xc[1] + p_ml.k_k) * xd[3],
                p_ml.beta_k * exp(-p_ml.gamma_k * xc[1]  -p_ml.k_k) * xd[4]])
  else
    return (p_ml.beta_na * exp(4.0 * p_ml.gamma_na * xc[1] + 4.0 * p_ml.k_na) * xd[1] +
              p_ml.beta_na * xd[2] +
              p_ml.beta_k * exp( p_ml.gamma_k * xc[1] + p_ml.k_k) * xd[3] +
              p_ml.beta_k * exp(-p_ml.gamma_k * xc[1] - p_ml.k_k) * xd[4])
  end
end

function Delta_ml(xc::Array{Float64},xd::Array{Int64},t::Float64,parms::Vector,ind_reactio
  # this function return the jump in the continuous component
  return true
end

immutable F_type; end
call(::Type{F_type},xcd, xc, xd, t, parms) = F_ml(xcd, xc, xd, t, parms)

immutable R_type; end
call(::Type{R_type},xc, xd, t, parms, sr) = R_ml(xc, xd, t, parms, sr)

immutable DX_type; end
call(::Type{DX_type},xc, xd, t, parms, ind_reaction) = Delta_ml(xc, xd, t, parms, ind_reac

xc0 = vec([p1["v(0)"]])
```

```
xd0 = vec([Int(p0["N"]),    #Na closed
           0,               #Na opened
           Int(p0["M"]),    #K closed
           0])              #K opened

nu = [[-1 1 0 0];[1 -1 0 1];[0 0 -1 1];[0 0 1 -1]]
parms = vec([0.])
tf = p1["t_end"]

dummy_t = chv(6,xc0,xd0, F_ml, R_ml,(x,y,t,pr,id)->vec([0.]), nu , parms,0.0,0.01,false)
srand(123)
dummy_t = @time chv(4500,xc0,xd0, F_ml, R_ml,(x,y,t,pr,id)->vec([0.]), nu , parms,0.0,tf,f
result =  PDMP.chv_optim(2,xc0,xd0,F_type,R_type,DX_type,nu,parms,0.0,tf,false)
srand(123)
result =  @time PDMP.chv_optim(4500,xc0,xd0,F_type,R_type,DX_type,nu,parms,0.0,tf,false) #
println("#jumps = ", length(dummy_t.time)," ", length(result.time))
try
  println(norm(dummy_t.time-result.time))
  println("--> xc_f-xc_t = ",norm(dummy_t.xc-result.xc))
  println("--> xd_f-xd_t = ",norm(dummy_t.xd-result.xd))
end
GR.plot(result.time,result.xc[1,:],"y",result.time, 0*result.xd[3,:],title = string("#Jump
```

4. Future directions

5. Acknowledgements

References

Pineda-Krch M (2008). "GillespieSSA: Implementing the stochastic simulation algorithm in R." *Journal of Statistical Software*, **25**(12), 1–18.

Veltz R (2015). "A new twist for the simulation of hybrid systems using the true jump method." 1504.06873.

Affiliation:

Romain Veltz
MathNeuro
Institut National de Recherche en Informatique et en Automatique
E-mail: romain.veltz@inria.fr
URL: https://romainveltz.pythonanywhere.com/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd