Diffusion Model

Samuel Isaacson, Chris Rackauckas

March 5, 2019

```julia
using DifferentialEquations, Plots, Statistics, DiffEqProblemLibrary.JumpProblemLibrary
gr()
fmt = :svg
JumpProblemLibrary.importjumpproblems()
```

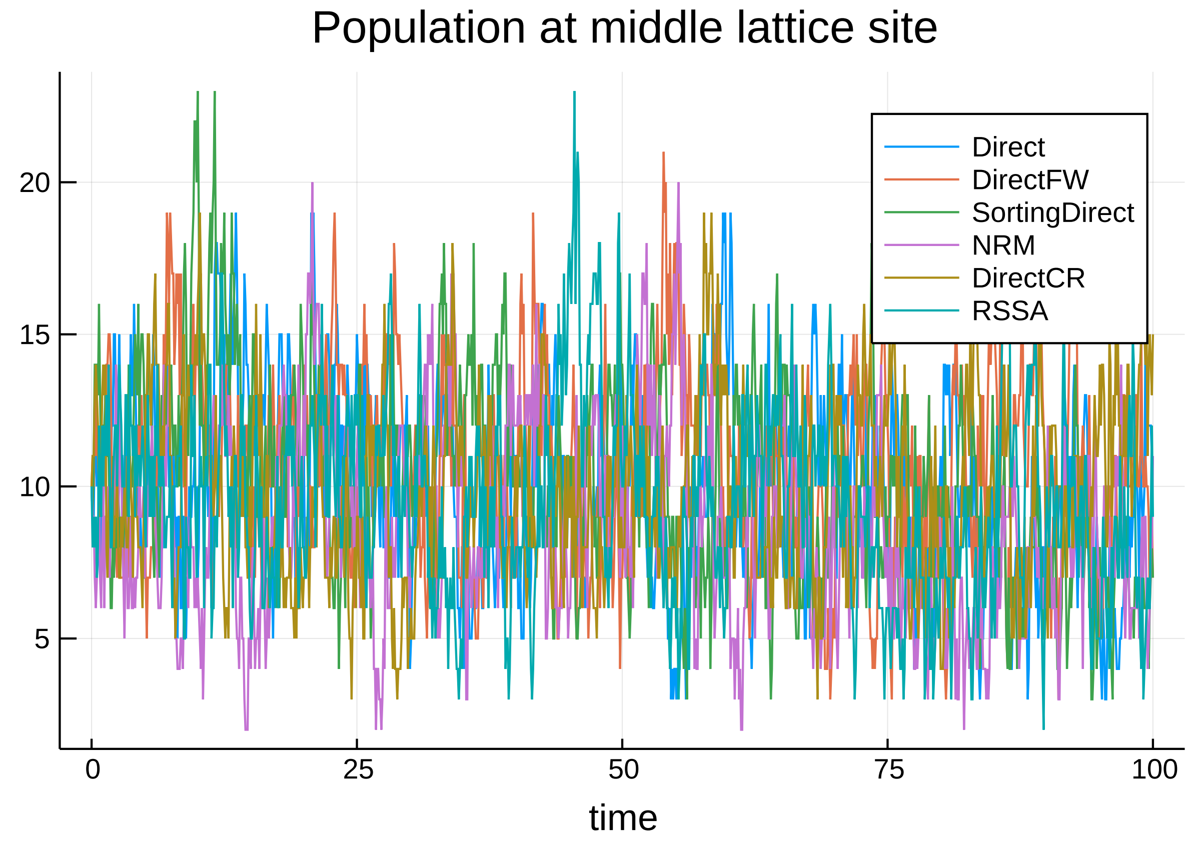
1 Model and example solutions

Here we implement a 1D continuous time random walk approximation of diffusion for N lattice sites, with reflecting boundary conditions

```julia
N = 256
jprob = prob_jump_diffnetwork
rn = jprob.network(N)
rnpar = jprob.rates
u0 = jprob.u0(N)
tf = 100. #jprob.tstop
```

```
100.0
```

```julia
methods = (Direct(),DirectFW(),SortingDirect(),NRM(),DirectCR(),RSSA())
legs    = [typeof(method) for method in methods]
shortlabels = [string(leg)[12:end] for leg in legs]
prob    = prob = DiscreteProblem(u0, (0.0, tf), rnpar)
ploth   = plot(reuse=false)
for (i,method) in enumerate(methods)
    jump_prob = JumpProblem(prob, method, rn, save_positions=(false,false))
    sol = solve(jump_prob, SSAStepper(), saveat=tf/1000.)
    plot!(ploth,sol.t,sol[Int(N//2),:],label=shortlabels[i], format=fmt)
end
plot!(ploth, title="Population at middle lattice site", xlabel="time",format=fmt)
```



2 Benchmarking performance of the methods

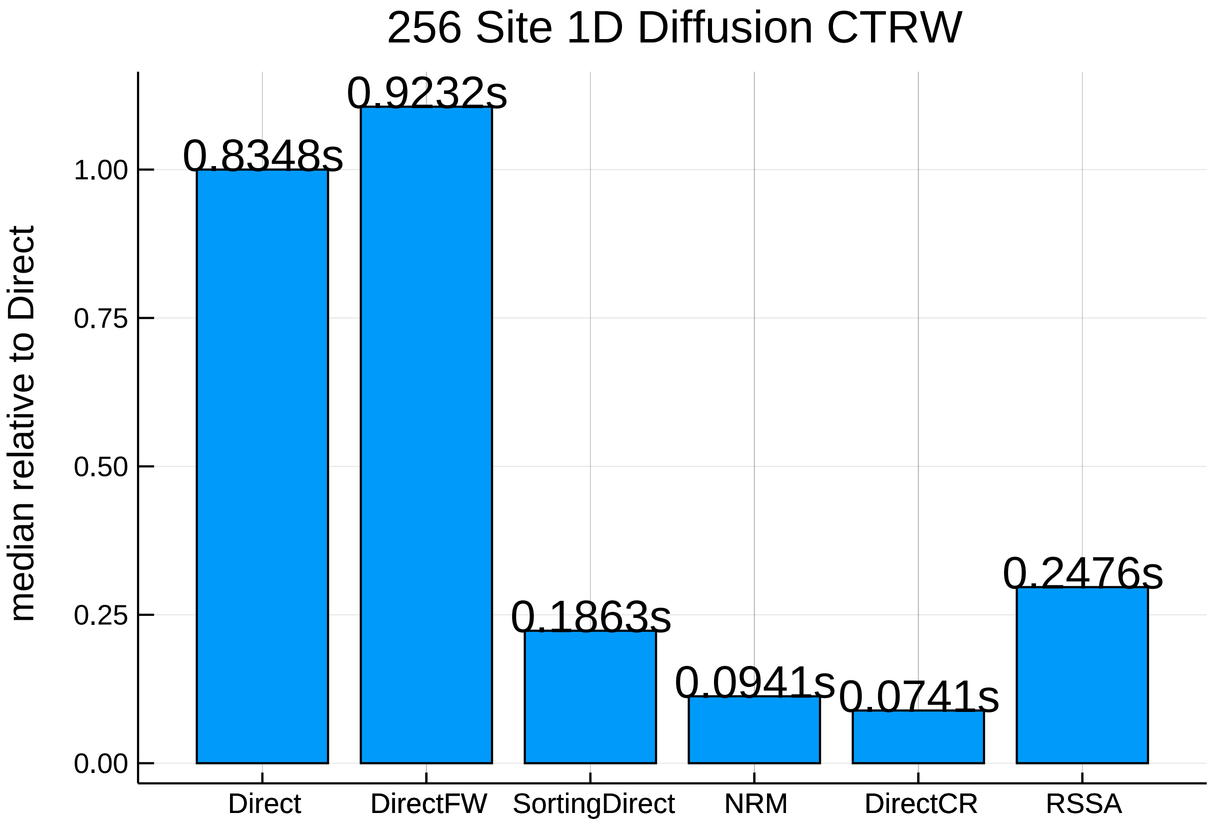
```
function run_benchmark!(t, jump_prob, stepper)
    sol = solve(jump_prob, stepper)
    @inbounds for i in 1:length(t)
        t[i] = @elapsed (sol = solve(jump_prob, stepper))
    end
end

run_benchmark! (generic function with 1 method)

nsims = 50
benchmarks = Vector{Vector{Float64}}()
for method in methods
    jump_prob = JumpProblem(prob, method, rn, save_positions=(false,false))
    stepper = SSAStepper()
    t = Vector{Float64}(undef,nsims)
    run_benchmark!(t, jump_prob, stepper)
    push!(benchmarks, t)
end

medtimes = Vector{Float64}(undef,length(methods))
stdtimes = Vector{Float64}(undef,length(methods))
avgtimes = Vector{Float64}(undef,length(methods))
for i in 1:length(methods)
    medtimes[i] = median(benchmarks[i])
    avgtimes[i] = mean(benchmarks[i])
    stdtimes[i] = std(benchmarks[i])
end
using DataFrames
```

```
df = DataFrame(names=shortlabels,medtimes=medtimes,relmedtimes=(medtimes/medtimes[1]),
                avgtimes=avgtimes, std=stdtimes, cv=stdtimes./avgtimes)
```

	names	medtimes	relmedtimes	avgtimes	std	cv
	String	Float64	Float64	Float64	Float64	Float64
1	Direct	0.834761	1.0	0.839253	0.0226454	0.0269828
2	DirectFW	0.923186	1.10593	0.926974	0.0198834	0.0214498
3	SortingDirect	0.186279	0.223152	0.187372	0.00440098	0.0234879
4	NRM	0.0940938	0.112719	0.0946399	0.00284365	0.0300471
5	DirectCR	0.0740931	0.0887597	0.0747331	0.00340332	0.0455397
6	RSSA	0.247621	0.296637	0.248613	0.00385581	0.0155093

3 Plotting

```
sa = [string(round(mt,digits=4),"s") for mt in df.medtimes]
bar(df.names,df.relmedtimes,legend=:false, fmt=fmt)
scatter!(df.names, .025 .+df.relmedtimes, markeralpha=0, series_annotations=sa, fmt=fmt)
ylabel!("median relative to Direct")
title!("256 Site 1D Diffusion CTRW")
```



```
using DiffEqBenchmarks
DiffEqBenchmarks.bench_footer(WEAVE_ARGS[:folder],WEAVE_ARGS[:file])
```

3.1 Appendix

These benchmarks are a part of the DiffEqBenchmarks.jl repository, found at: https://github.com/JuliaDi

To locally run this tutorial, do the following commands:

```
using DiffEqBenchmarks
DiffEqBenchmarks.weave_file("Jumps","Diffusion_CTRW.jmd")
```

Computer Information:

```
Julia Version 1.1.0
Commit 80516ca202 (2019-01-21 21:24 UTC)
Platform Info:
  OS: macOS (x86_64-apple-darwin14.5.0)
  CPU: Intel(R) Core(TM) i7-6920HQ CPU @ 2.90GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-6.0.1 (ORCJIT, skylake)
```

Package Information:

```
Status: `/Users/isaacsas/Dropbox/github_public_checkout/DiffEqBenchmarks.jl/Project.tom
[f3b72e0c-5b89-59e1-b016-84e28bfd966d] DiffEqDevTools 2.7.0
[7073ff75-c697-5162-941a-fcdaad2a7d2a] IJulia 1.17.0
[7f56f5a3-f504-529b-bc02-0b1fe5e64312] LSODA 0.4.0
[c030b06c-0b6d-57c2-b091-7029874bd033] ODE 2.4.0
[09606e27-ecf5-54fc-bb29-004bd9f985bf] ODEInterfaceDiffEq 3.0.0
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.3.0
[65888b18-ceab-5e60-b2b9-181511a3b968] ParameterizedFunctions 4.1.1
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 0.23.1
[c3572dad-4567-51f8-b174-8c6c989267f4] Sundials 3.1.0
[44d3d7a6-8a23-5bf8-98c5-b353f8df5ec9] Weave 0.7.2
[b77e0a4c-d291-57a0-90e8-8db25a27a240] InteractiveUtils
[d6f4376e-aef5-505a-96c1-9c027394607a] Markdown
[44cfe95a-1eb2-52ea-b672-e2afdf69b78f] Pkg
[9a3f8284-a2c9-5f02-9a11-845980a1fd5c] Random
```