Mendes Multistate Model

Samuel Isaacson, Chris Rackauckas

March 5, 2019

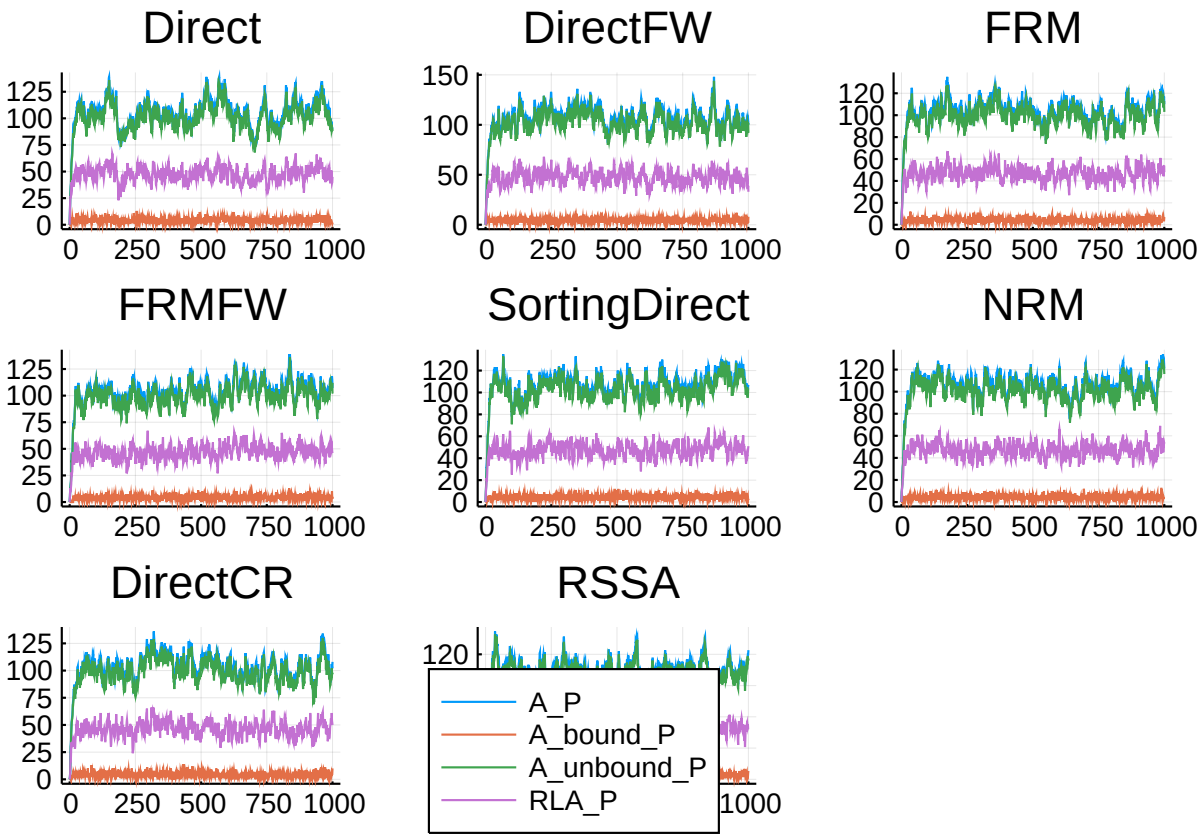
Taken from Gupta and Mendes, *An Overview of Network-Based and -Free Approaches for Stochastic Simulation of Biochemical Systems*, Computation, 6 (9), 2018.

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

1 Plot solutions by each method

```
methods = (Direct(),DirectFW(),FRM(),FRMFW(),SortingDirect(),NRM(),DirectCR(),RSSA())
legs    = [typeof(method) for method in methods]
#shortlabels = [string(leg)[12:end] for leg in legs]
shortlabels = [string(leg) for leg in legs]
jprob   = prob_jump_multistate
tf      = 10.0*jprob.tstop
rn      = jprob.network
#prob = jprob.discrete_prob
prob = DiscreteProblem(jprob.u0, (0.0,tf), jprob.rates)
varlegs = ["A_P", "A_bound_P", "A_unbound_P", "RLA_P"]
varsyms = [
    [:S7,:S8,:S9],
    [:S9],
    [:S7,:S8],
    [:S7]
]
varidxs = []
fmt = :png
for vars in varsyms
    push!(varidxs, [findfirst(x -> x==sym, rn.syms) for sym in vars])
end

p = []
for (i,method) in enumerate(methods)
    jump_prob = JumpProblem(prob, method, rn, save_positions=(false,false))
    sol = solve(jump_prob, SSAStepper(), saveat=tf/1000.)
    solv = zeros(1001,4)
    for (i,varidx) in enumerate(varidxs)
        solv[:,i] = sum(sol[varidx,:],dims=1)
    end
    if i < length(methods)
        push!(p, plot(sol.t,solv,title=shortlabels[i],legend=false,format=fmt))
```

```
    else
        push!(p,
    plot(sol.t,solv,title=shortlabels[i],legend=true,labels=varlegs,format=fmt))
    end
end
plot(p...,format=fmt)
```



2 Benchmarking performance of the methods

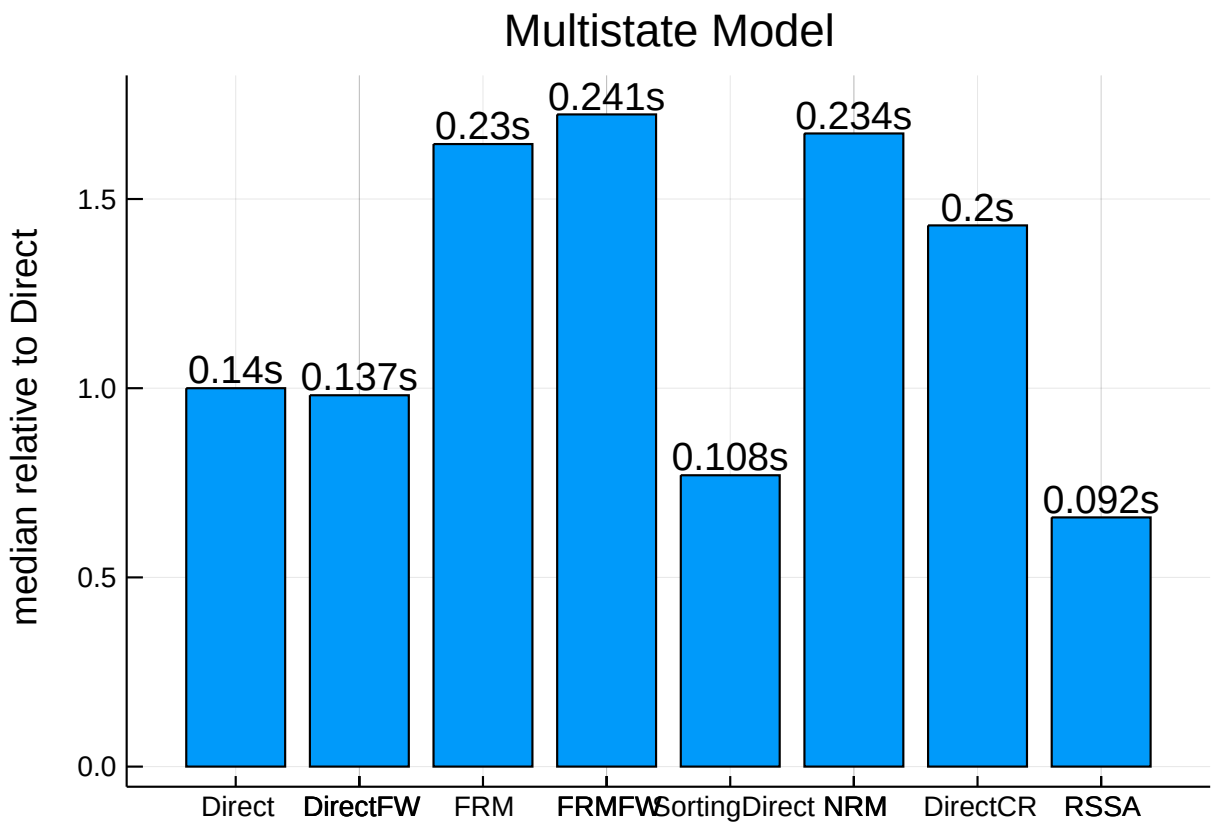
```
function run_benchmark!(t, jump_prob, stepper)
    sol = solve(jump_prob, stepper)
    @inbounds for i in 1:length(t)
        t[i] = @elapsed (sol = solve(jump_prob, stepper))
    end
end

run_benchmark! (generic function with 1 method)

nsims = 100
benchmarks = Vector{Vector{Float64}}()
for method in methods
    println("Benchmarking method:  ", method)
    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

Benchmarking method: Direct()
Benchmarking method: DirectFW()
```

```
Benchmarking method: FRM()
Benchmarking method: FRMFW()
Benchmarking method: SortingDirect()
Benchmarking method: NRM()
Benchmarking method: DirectCR()
Benchmarking method: RSSA()

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	0.139693	1.0	0.138995	0.00561666	0.040409
2	DirectFW	0.137088	0.981352	0.138433	0.00502957	0.0363321
3	FRM	0.229825	1.64521	0.231296	0.00563419	0.0243592
4	FRMFW	0.240764	1.72352	0.241687	0.00552775	0.0228716
5	SortingDirect	0.107551	0.769909	0.108765	0.00329881	0.0303296
6	NRM	0.233737	1.67322	0.234927	0.00675003	0.0287324
7	DirectCR	0.199796	1.43025	0.200791	0.0045094	0.0224581
8	RSSA	0.0919927	0.658534	0.0929124	0.00230664	0.024826

3 Plotting

```
sa = [text(string(round(mt,digits=3),"s"),:center,12) for mt in df.medtimes]
bar(df.names,df.relmedtimes,legend=:false, fmt=fmt)
scatter!(df.names, .05 .+ df.relmedtimes, markeralpha=0, series_annotations=sa, fmt=fmt)
ylabel!("median relative to Direct")
title!("Multistate Model")
```




```
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","Mendes_multistate_example.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
```