# Oval2 Timings

Chris Rackauckas

May 4, 2019

```julia
using Distributed
addprocs()

@everywhere begin
  using StochasticDiffEq, DiffEqProblemLibrary, ParallelDataTransfer, Random
  using DiffEqProblemLibrary.SDEProblemLibrary: importsdeproblems; importsdeproblems()
  prob =
    DiffEqProblemLibrary.SDEProblemLibrary.oval2ModelExample(largeFluctuations=true,useBigs=false)
  Random.seed!(99 + myid())
  prob = remake(prob,tspan=(0.0,1.0))
  println("Solve once to compile.")
  sol = solve(prob,EM(),dt=1/2^(18),adaptive=false,save_everystep=false)
  sol = solve(prob,RKMil(),dt=1/2^(18),adaptive=false,save_everystep=false)
  sol = solve(prob,SRIW1(),dt=1/2^(18),adaptive=false,save_everystep=false)
  sol = solve(prob,SRI(),dt=1/2^(18),adaptive=false,save_everystep=false)
  sol = solve(prob,SOSRI(),dt=1/2^(18),adaptive=false,save_everystep=false)
  sol = solve(prob,SOSRI2(),dt=1/2^(18),adaptive=false,save_everystep=false)
  Int(sol.u[1]!=NaN)
  println("Compilation complete.")
  js = 16:21
  dts = 1.0 ./ 2.0 .^ (js)
  fails = Array{Int}(undef,length(dts),3)
  times = Array{Float64}(undef,length(dts),3)
  numRuns = 10000
end
```

```
From worker 21: Solve once to compile.
      From worker 3:    Solve once to compile.
      From worker 18:   Solve once to compile.
      From worker 7:    Solve once to compile.
      From worker 17:   Solve once to compile.
      From worker 20:   Solve once to compile.
      From worker 15:   Solve once to compile.
      From worker 14:   Solve once to compile.
      From worker 19:   Solve once to compile.
      From worker 10:   Solve once to compile.
      From worker 6:    Solve once to compile.
      From worker 11:   Solve once to compile.
      From worker 9:    Solve once to compile.
      From worker 16:   Solve once to compile.
      From worker 5:    Solve once to compile.
      From worker 4:    Solve once to compile.
      From worker 13:   Solve once to compile.
      From worker 8:    Solve once to compile.
      From worker 2:    Solve once to compile.
      From worker 12:   Solve once to compile.
      From worker 12:   Compilation complete.
      From worker 2:    Compilation complete.
      From worker 15:   Compilation complete.
      From worker 4:    Compilation complete.
      From worker 7:    Compilation complete.
      From worker 5:    Compilation complete.
      From worker 19:   Compilation complete.
      From worker 21:   Compilation complete.
      From worker 10:   Compilation complete.
      From worker 18:   Compilation complete.
      From worker 3:    Compilation complete.
      From worker 6:    Compilation complete.
      From worker 16:   Compilation complete.
      From worker 9:    Compilation complete.
      From worker 13:   Compilation complete.
      From worker 14:   Compilation complete.
      From worker 11:   Compilation complete.
      From worker 20:   Compilation complete.
      From worker 8:    Compilation complete.
      From worker 17:   Compilation complete.
      From worker 25:   Solve once to compile.
      From worker 23:   Solve once to compile.
      From worker 28:   Solve once to compile.
      From worker 29:   Solve once to compile.
      From worker 30:   Solve once to compile.
      From worker 27:   Solve once to compile.
      From worker 35:   Solve once to compile.
      From worker 26:   Solve once to compile.
      From worker 33:   Solve once to compile.
      From worker 32:   Solve once to compile.
      From worker 24:   Solve once to compile.
      From worker 36:   Solve once to compile.
      From worker 31:   Solve once to compile.
      From worker 37:   Solve once to compile.
      From worker 22:   Solve once to compile.
      From worker 34:   Solve once to compile.
      From worker 25:   Compilation complete.
      From worker 28:   Compilation complete.
      From worker 30:   Compilation complete.
      From worker 26:   Compilation complete.
      From worker 35:   Compilation complete.
      From worker 29:   Compilation complete.
      From worker 36:   Compilation complete.
      From worker 33:   Compilation complete.
      From worker 23:   Compilation complete.
      From worker 22:   Compilation complete.
      From worker 31:   Compilation complete.
      From worker 34:   Compilation complete.
      From worker 37:   Compilation complete.
      From worker 32:   Compilation complete.
      From worker 24:   Compilation complete.
      From worker 27:   Compilation complete.
Error: UndefVarError: myid not defined
top-level scope at none:6
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```julia
println("Setup Complete")
```

```
Setup Complete
```

```julia
## Timing Runs

@everywhere function runAdaptiveSRIW1(i)
  sol =
    solve(prob,SRIW1(),abstol=2.0^(-13),reltol=2.0^(-7),maxIters=Int(1e11),qmax=1.125,save_everystep=f
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```julia
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIW1,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```julia
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```julia
## Timing Runs

@everywhere function runAdaptiveSRI(i)
    sol =
    solve(prob,SRI(error_terms=2),abstol=2.0^(-13),reltol=2.0^(-7),maxIters=Int(1e11),qmax=1.125,save_
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```julia
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRI,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```julia
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```julia
## Timing Runs

@everywhere function runAdaptiveSRI(i)
  sol =
    solve(prob,SRI(),abstol=2.0^(-14),reltol=2.0^(-18),maxIters=Int(1e11),qmax=1.125,save_everystep=fa
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRI,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```
## Timing Runs

@everywhere function runAdaptiveSRIOpt1(i)
  sol =
    solve(prob,SRI(tableau=StochasticDiffEq.constructSRIOpt1()),abstol=2.0^(-7),reltol=2.0^(-4),maxIte
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIOpt1,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```
## Timing Runs

@everywhere function runAdaptiveSRIOpt1(i)
  sol =
    solve(prob,SOSRI(),abstol=2.0^(-7),reltol=2.0^(-4),maxIters=Int(1e11),qmax=1.125,save_everystep=fa
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIOpt1,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```
## Timing Runs

@everywhere function runAdaptiveSRIOpt1(i)
  sol =
    solve(prob,SOSRI(),abstol=2.0^(-7),reltol=2.0^(-6),maxIters=Int(1e11),qmax=1.125,save_everystep=fa
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIOpt1,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```
## Timing Runs

@everywhere function runAdaptiveSRIOpt1(i)
  sol =
    solve(prob,SOSRI(),abstol=2.0^(-12),reltol=2.0^(-15),maxIters=Int(1e11),qmax=1.125,save_everystep=
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIOpt1,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```
## Timing Runs

@everywhere function runAdaptiveSRIOpt1(i)
  sol =
    solve(prob,SOSRI(),abstol=2.0^(-13),reltol=2.0^(-7),maxIters=Int(1e11),qmax=1.125,save_everystep=f
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIOpt1,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```julia
## Timing Runs

@everywhere function runAdaptiveSRIOpt1(i)
  sol =
    solve(prob,SOSRI(),abstol=2.0^(-12),reltol=2.0^(-15),maxIters=Int(1e11),qmax=1.125,save_everystep=
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```julia
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIOpt1,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```julia
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```julia
## Timing Runs

@everywhere function runAdaptiveSRIOpt2(i)
    sol =
    solve(prob,SOSRI2(),abstol=2.0^(-12),reltol=2.0^(-15),maxIters=Int(1e11),qmax=1.125,save_everystep
    Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```julia
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIOpt2,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```julia
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```julia
## Timing Runs

@everywhere function runAdaptiveSRIOpt2(i)
    sol =
    solve(prob,SOSRI2(),abstol=2.0^(-13),reltol=2.0^(-11),maxIters=Int(1e11),qmax=1.125,save_everystep
    Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIOpt2,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```
## Timing Runs

@everywhere function runAdaptiveSRIOpt2(i)
    sol =
    solve(prob,SOSRI2(),abstol=2.0^(-16),reltol=2.0^(-9),maxIters=Int(1e11),qmax=1.125,save_everystep=
    Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
@everywhere Random.seed!(99 + myid())
```

```
Error: UndefVarError: myid not defined
top-level scope at none:0
eval at ./boot.jl:328 [inlined]
(::getfield(Distributed, Symbol("##164#166")){Module,Expr})() at ./task.jl:
259
```

```
adaptiveTime = @elapsed numFails = sum(pmap(runAdaptiveSRIOpt2,1:numRuns))
```

```
Error: UndefVarError: numRuns not defined
```

```
println("The number of Adaptive Fails is $numFails.  Elapsed time was $adaptiveTime")
```

```
Error: UndefVarError: numFails not defined
```

```
@everywhere function runEM(i,j)
  sol =solve(prob,EM(),dt=dts[j],maxIters=Int(1e11),save_everystep=false,verbose=false)
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
for j in eachindex(js)
  println("j = $j")
  sendto(workers(), j=j)
  @everywhere Random.seed!(99 + myid())
  t1 = @elapsed numFails = sum(pmap((i)->runEM(i,j),1:numRuns))
  println("The number of Euler-Maruyama Fails is $numFails.  Elapsed time was $t1")
  fails[j,1] = numFails
  times[j,1] = t1
end
```

```
Error: UndefVarError: js not defined
```

```
@everywhere function runSRI(i,j)
  sol
    =solve(prob,SRIW1(),dt=dts[j],maxIters=Int(1e11),adaptive=false,save_everystep=false,verbose=fals
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
for j in 1:4
  println("j = $j")
  sendto(workers(), j=j)
  @everywhere Random.seed!(99 + myid())
  t2 = @elapsed numFails = sum(pmap((i)->runSRI(i,j),1:numRuns))
  println("The number of Rossler-SRI Fails is $numFails.  Elapsed time was $t2")
  fails[j,2] = numFails
  times[j,2] = t2
end
```

```
j = 1
Error: UndefVarError: sendto not defined
```

```
@everywhere js = 17:21
@everywhere dts = 1.0 ./2.0 .^ (js)
@everywhere function runIEM(i,j)
  sol
    =solve(prob,ImplicitEM(),dt=dts[j],maxIters=Int(1e11),save_everystep=false,verbose=false)
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
for j in 1:6
  println("j = $j")
  sendto(workers(), j=j)
  @everywhere Random.seed!(99 + myid())
  t2 = @elapsed numFails = sum(pmap((i)->runIEM(i,j),1:numRuns))
    println("The number of Implicit-EM Fails is $numFails.  Elapsed time was $t2")
  fails[j,2] = numFails
  times[j,2] = t2
end
```

```
j = 1
Error: UndefVarError: sendto not defined
```

```
@everywhere js = 17:21
@everywhere dts = 1.0 ./ 2.0 .^(js)
@everywhere function runIRM(i,j)
    sol
    =solve(prob,ImplicitRKMil(),dt=dts[j],maxIters=Int(1e11),save_everystep=false,verbose=false)
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
for j in 1:4
  println("j = $j")
  sendto(workers(), j=j)
  @everywhere Random.seed!(99 + myid())
  t2 = @elapsed numFails = sum(pmap((i)->runIRM(i,j),1:numRuns))
    println("The number of Implicit-RKMil Fails is $numFails.  Elapsed time was $t2")
  fails[j,2] = numFails
  times[j,2] = t2
end
```

```
j = 1
Error: UndefVarError: sendto not defined
```

```
@everywhere function runMil(i,j)
  sol
    =solve(prob,RKMil(),dt=dts[j],maxIters=Int(1e11),save_everystep=false,verbose=false)
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end
for j in eachindex(js)
  println("j = $j")
  sendto(workers(), j=j)
  @everywhere Random.seed!(99 + myid())
  t3 = @elapsed numFails = sum(pmap((i)->runMil(i,j),1:numRuns))
  println("The number of RK-Milstein Fails is $numFails.  Elapsed time was $t3")
  fails[j,3] = numFails
  times[j,3] = t3
end
```

```
Error: UndefVarError: js not defined
```

```
using Plots
lw = 3
p2 =
    plot(dts,times,xscale=:log2,yscale=:log2,guidefont=font(16),tickfont=font(14),yguide="Elapsed
    Time (s)",xguide=L"Chosen $\Delta
    t$",top_margin=50px,linewidth=lw,lab=["Euler-Maruyama" "RK-Mil"
    "RosslerSRI"],legendfont=font(14))
```

```
Error: LoadError: UndefVarError: @L_str not defined
in expression starting at none:1
```

```
plot!(dts,repmat([adaptiveTime],11),linewidth=lw,line=:dash,lab="ESRK+RSwM3",left_margin=75px)
```

```
Error: UndefVarError: adaptiveTime not defined
```

```
scatter!([2.0^(-20);2.0^(-20);2.0^(-18)],[times[5,1];times[5,2];times[3,3]],markersize=20,c=:red,lab="
```

```
Error: UndefVarError: times not defined
```

```
plot(p2,size=(800,800))
```

```
Error: UndefVarError: p2 not defined
```

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

## 0.1 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("AdaptiveSDE","Oval2Timings.jmd")
```

Computer Information:

```
Julia Version 1.1.0
Commit 80516ca202 (2019-01-21 21:24 UTC)
Platform Info:
  OS: Linux (x86_64-pc-linux-gnu)
  CPU: Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-6.0.1 (ORCJIT, haswell)
```

Package Information:

```
Status: `/home/crackauckas/.julia/environments/v1.1/Project.toml`
[c52e3926-4ff0-5f6e-af25-54175e0327b1] Atom 0.8.5
[bcd4f6db-9728-5f36-b5f7-82caef46ccdb] DelayDiffEq 5.2.0
```

```
[bb2cbb15-79fc-5d1e-9bf1-8ae49c7c1650] DiffEqBenchmarks 0.1.0
[459566f4-90b8-5000-8ac3-15dfb0a30def] DiffEqCallbacks 2.5.2
[f3b72e0c-5b89-59e1-b016-84e28bfd966d] DiffEqDevTools 2.8.0
[78ddff82-25fc-5f2b-89aa-309469cbf16f] DiffEqMonteCarlo 0.14.0
[77a26b50-5914-5dd7-bc55-306e6241c503] DiffEqNoiseProcess 3.1.0
[055956cb-9e8b-5191-98cc-73ae4a59e68a] DiffEqPhysics 3.1.0
[a077e3f3-b75c-5d7f-a0c6-6bc4c8ec64a9] DiffEqProblemLibrary 4.1.0
[41bf760c-e81c-5289-8e54-58b1f1f8abe2] DiffEqSensitivity 3.2.2
[0c46a032-eb83-5123-abaf-570d42b7fbaa] DifferentialEquations 6.3.0
[b305315f-e792-5b7a-8f41-49f472929428] Elliptic 0.5.0
[e5e0dc1b-0480-54bc-9374-aad01c23163d] Juno 0.7.0
[7f56f5a3-f504-529b-bc02-0b1fe5e64312] LSODA 0.4.0
[c030b06c-0b6d-57c2-b091-7029874bd033] ODE 2.4.0
[54ca160b-1b9f-5127-a996-1867f4bc2a2c] ODEInterface 0.4.5
[09606e27-ecf5-54fc-bb29-004bd9f985bf] ODEInterfaceDiffEq 3.1.0
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.6.0
[2dcacdae-9679-587a-88bb-8b444fb7085b] ParallelDataTransfer 0.5.0
[65888b18-ceab-5e60-b2b9-181511a3b968] ParameterizedFunctions 4.1.1
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 0.24.0
[d330b81b-6aea-500a-939a-2ce795aea3ee] PyPlot 2.8.1
[90137ffa-7385-5640-81b9-e52037218182] StaticArrays 0.10.3
[789caeaf-c7a9-5a7d-9973-96adeb23e2a0] StochasticDiffEq 6.1.1+
[c3572dad-4567-51f8-b174-8c6c989267f4] Sundials 3.4.1
[92b13dbe-c966-51a2-8445-caca9f8a7d42] TaylorIntegration 0.4.1
[44d3d7a6-8a23-5bf8-98c5-b353f8df5ec9] Weave 0.9.0
[e88e6eb3-aa80-5325-afca-941959d7151f] Zygote 0.3.0
```