qmax Determination

Chris Rackauckas

May 5, 2019

```
qs = 1.0 .+ 2.0.^(-5:2)
times = Array{Float64}(undef,length(qs),4)
means = Array{Float64}(undef,length(qs),4)

using StochasticDiffEq, DiffEqProblemLibrary, Random,
      Plots, ParallelDataTransfer, DiffEqMonteCarlo, Distributed
Random.seed!(99)

using DiffEqProblemLibrary.SDEProblemLibrary: importsdeproblems; importsdeproblems()
full_prob =
    DiffEqProblemLibrary.SDEProblemLibrary.oval2ModelExample(largeFluctuations=true,useBigs=false)
import DiffEqProblemLibrary.SDEProblemLibrary: prob_sde_additivesystem,
            prob_sde_additive, prob_sde_2Dlinear, prob_sde_linear, prob_sde_wave
prob = remake(full_prob,tspan=(0.0,1.0))

println("Solve once to compile.")
```

```
Solve once to compile.
```

```
sol = solve(prob,EM(),dt=1/2^(18))
Int(sol.u[end][1]!=NaN)
println("Compilation complete.")
```

```
Compilation complete.
```

```
num_runs = 10000

probs = Vector{SDEProblem}(undef,3)
p1 = Vector{Any}(undef,3)
p2 = Vector{Any}(undef,3)
p3 = Vector{Any}(undef,3)
## Problem 1
probs[1] = prob_sde_linear
## Problem 2
probs[2] = prob_sde_wave
## Problem 3
probs[3] = prob_sde_additive

println("Setup Complete")
```

```
Setup Complete
```

```
## Timing Runs

function runAdaptive(i,k)
```

```
  sol = solve(prob,SRIW1(),dt=1/2^(8),abstol=2.0^(-15),reltol=2.0^(-10),
              verbose=false,maxIters=Int(1e12),qmax=qs[k])
  Int(any(isnan,sol[end]) || sol.t[end] != 1)
end

#Compile
monte_prob = MonteCarloProblem(probs[1])
test_mc =
    solve(monte_prob,SRIW1(),dt=1/2^(4),adaptive=true,num_monte=1000,abstol=2.0^(-1),reltol=0)
DiffEqBase.calculate_monte_errors(test_mc);
```

0.1 qmax test on Oval2 Model

```
for k in eachindex(qs)
  global times
  Random.seed!(99)
  adaptiveTime = @elapsed numFails = sum(map((i)->runAdaptive(i,k),1:num_runs))
  println("k was $k.  The number of Adaptive Fails is $numFails.  Elapsed time was
    $adaptiveTime")
  times[k,4] = adaptiveTime
end
```

```
k was 1. The number of Adaptive Fails is 0. Elapsed time was 317.647723094
k was 2. The number of Adaptive Fails is 0. Elapsed time was 282.236955592
k was 3. The number of Adaptive Fails is 0. Elapsed time was 261.132643023
k was 4. The number of Adaptive Fails is 0. Elapsed time was 267.127418119
k was 5. The number of Adaptive Fails is 0. Elapsed time was 292.05685787
k was 6. The number of Adaptive Fails is 0. Elapsed time was 298.303537178
k was 7. The number of Adaptive Fails is 0. Elapsed time was 302.916861552
k was 8. The number of Adaptive Fails is 0. Elapsed time was 312.431735455
```

0.2 qmax test on other problems

```
for k in eachindex(probs)
  global probs, times, means, qs
  println("Problem $k")
  ## Setup
  prob = probs[k]

  for i in eachindex(qs)
    msim =
    solve(monte_prob,dt=1/2^(4),SRIW1(),adaptive=true,num_monte=num_runs,abstol=2.0^(-13),reltol=0,qma
    test_msim = DiffEqBase.calculate_monte_errors(msim)
    times[i,k] = test_msim.elapsedTime
    means[i,k] = test_msim.error_means[:final]
    println("for k=$kand i=$i, we get that the error was $(means[i,k])and it took
    $(times[i,k])seconds")
  end
end
```

```
Problem 1
for k=1 and i=1, we get that the error was 4.752435711479388e-6 and it took
 32.143574147 seconds
for k=1 and i=2, we get that the error was 3.322909161411088e-5 and it took
 31.521448137 seconds
for k=1 and i=3, we get that the error was 1.3730083619195504e-5 and it too
k 31.421578055 seconds
```

```
for k=1 and i=4, we get that the error was 4.403850872946114e-6 and it took
 31.744041866 seconds
for k=1 and i=5, we get that the error was 6.779258874328445e-6 and it took
 31.381334593 seconds
for k=1 and i=6, we get that the error was 1.0554260580802185e-5 and it too
k 31.076058815 seconds
for k=1 and i=7, we get that the error was 4.419128392544222e-6 and it took
 30.800909129 seconds
for k=1 and i=8, we get that the error was 6.255351353721408e-6 and it took
 31.615934798 seconds
Problem 2
for k=2 and i=1, we get that the error was 4.4063662698478e-6 and it took 3
1.187699644 seconds
for k=2 and i=2, we get that the error was 9.332424379550144e-6 and it took
 31.460754253 seconds
for k=2 and i=3, we get that the error was 1.1891533270357014e-5 and it too
k 30.971309683 seconds
for k=2 and i=4, we get that the error was 4.437600751222222e-6 and it took
 31.05549018 seconds
for k=2 and i=5, we get that the error was 4.475310048182175e-6 and it took
 30.715568677 seconds
for k=2 and i=6, we get that the error was 5.240470568789925e-6 and it took
 31.233117833 seconds
for k=2 and i=7, we get that the error was 4.475640710204674e-6 and it took
 30.990549261 seconds
for k=2 and i=8, we get that the error was 4.3839301040267514e-6 and it too
k 31.266823178 seconds
Problem 3
for k=3 and i=1, we get that the error was 4.70733966326788e-6 and it took
30.961395408 seconds
for k=3 and i=2, we get that the error was 5.9436074401966624e-6 and it too
k 30.747156944 seconds
for k=3 and i=3, we get that the error was 6.089280870308689e-6 and it took
 31.24136521 seconds
for k=3 and i=4, we get that the error was 4.672848482701407e-6 and it took
 30.921427472 seconds
for k=3 and i=5, we get that the error was 5.969913432008566e-6 and it took
 30.736554648 seconds
for k=3 and i=6, we get that the error was 5.0646882986115735e-6 and it too
k 30.799384576 seconds
for k=3 and i=7, we get that the error was 1.5374997720857875e-5 and it too
k 30.50048996 seconds
for k=3 and i=8, we get that the error was 4.628683468435381e-6 and it took
 31.009805606 seconds
```

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

0.3 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","qmaxDetermination.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 3.2.0
[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 3.2.0
[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
[295af30f-e4ad-537b-8983-00126c2a3abe] Revise 2.1.4
[90137ffa-7385-5640-81b9-e52037218182] StaticArrays 0.10.3
[789caeaf-c7a9-5a7d-9973-96adeb23e2a0] StochasticDiffEq 6.2.0
[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
```