BCR Work-Precision Diagrams

Chris Rackauckas

October 7, 2020

The following benchmark is of a 1122 ODEs with 24388 terms that describe a stiff chemical reaction network.

```
using ReactionNetworkImporters, OrdinaryDiffEq, DiffEqBiological,
      Sundials, Plots, DiffEqDevTools, ODEInterface, ODEInterfaceDiffEq,
      LSODA, TimerOutputs, LinearAlgebra

LinearAlgebra.BLAS.set_num_threads(4)
gr()
prnbng = loadrxnetwork(BNGNetwork(), "BNGRepressilator",

joinpath(dirname(pathof(ReactionNetworkImporters)),"..","data","bcr","bcr.net"))

rn = deepcopy(prnbng.rn)
addodes!(rn; build_jac=false, build_symfuncs=false, build_paramjac=false)
tf = 100000.0
oprob = ODEProblem(rn, prnbng.u_0, (0.,tf), prnbng.p);

densejac_rn = deepcopy(prnbng.rn)
# zeroout_jac=true is needed to keep the generated expressions from being too big for
the compiler
addodes!(densejac_rn; build_jac=true, zeroout_jac = true, sparse_jac = false,
build_symfuncs=false, build_paramjac=false)
densejacprob = ODEProblem(densejac_rn, prnbng.u_0, (0.,tf), prnbng.p);

sparsejac_rn = deepcopy(prnbng.rn)
addodes!(sparsejac_rn; build_jac=true, sparse_jac = true, build_symfuncs=false,
build_paramjac=false)
sparsejacprob = ODEProblem(sparsejac_rn, prnbng.u_0, (0.,tf), prnbng.p);
```

```
Error: ArgumentError: Package ODEInterface not found in current path:
- Run `import Pkg; Pkg.add("ODEInterface")` to install the ODEInterface pac
kage.
```

```
@show numspecies(rn) # Number of ODEs
@show numreactions(rn) # Apprx. number of terms in the ODE
@show numparams(rn) # Number of Parameters
```

```
Error: UndefVarError: rn not defined
```

0.1 Time ODE derivative function compilation

As compiling the ODE derivative functions has in the past taken longer than running a simulation, we first force compilation by evaluating these functions one time.

```
const to = TimerOutput()
u_0 = prnbng.u_0
u = copy(u_0);
du = similar(u);
p = prnbng.p
@timeit to "ODERHS Eval1" rn.f(du,u,p,0.)
@timeit to "ODERHS Eval2" rn.f(du,u,p,0.)
sparsejac_rn.f(du,u,p,0.)

J = zeros(length(u),length(u))
@timeit to "DenseJac Eval1" densejac_rn.jac(J,u,p,0.)
@timeit to "DenseJac Eval2" densejac_rn.jac(J,u,p,0.)

Js = similar(sparsejac_rn.odefun.jac_prototype)
@timeit to "SparseJac Eval1" sparsejac_rn.jac(Js,u,p,0.)
@timeit to "SparseJac Eval2" sparsejac_rn.jac(Js,u,p,0.)
show(to)
```

```
Error: UndefVarError: TimerOutput not defined
```

0.2 Picture of the solution

```
sol = solve(oprob, CVODE_BDF(), saveat=tf/1000., reltol=1e-5, abstol=1e-5)
plot(sol,legend=false, fmt=:png)
```

```
Error: UndefVarError: tf not defined
```

For these benchmarks we will be using the timeseries error with these saving points since the final time point is not well-indicative of the solution behavior (capturing the oscillation is the key!).

0.3 Generate Test Solution

```
@time sol = solve(oprob,CVODE_BDF(),abstol=1/10^12,reltol=1/10^12)
test_sol = TestSolution(sol)
```

```
Error: UndefVarError: oprob not defined
```

0.4 Setups

```
abstols = 1.0 ./ 10.0 .^ (5:8)
reltols = 1.0 ./ 10.0 .^ (5:8);
setups = [
          #Dict(:alg=>Rosenbrock23(autodiff=false)),
          Dict(:alg=>TRBDF2(autodiff=false)),
          Dict(:alg=>CVODE_BDF()),
          Dict(:alg=>CVODE_BDF(linear_solver=:LapackDense)),
          #Dict(:alg=>rodas()),
          #Dict(:alg=>radau()),
          #Dict(:alg=>Rodas4(autodiff=false)),
          #Dict(:alg=>Rodas5(autodiff=false)),
          Dict(:alg=>KenCarp4(autodiff=false)),
          #Dict(:alg=>RadauIIA5(autodiff=false)),
          #Dict(:alg=>lsoda()),
          ]
```

```
4-element Array{Dict{Symbol,V} where V,1}:
 Dict{Symbol,OrdinaryDiffEq.TRBDF2{0,false,DiffEqBase.DefaultLinSolve,DiffE
qBase.NLNewton{Rational{Int64},Rational{Int64},Rational{Int64}},DataType}}(
:alg => OrdinaryDiffEq.TRBDF2{0,false,DiffEqBase.DefaultLinSolve,DiffEqBase
.NLNewton{Rational{Int64},Rational{Int64},Rational{Int64}},DataType}(DiffEq
Base.DefaultLinSolve(nothing, nothing), DiffEqBase.NLNewton{Rational{Int64}
,Rational{Int64},Rational{Int64}}(1//100, 10, 1//5, 1//5), Val{:forward}, t
rue, :linear, :PI))
 Dict{Symbol,Sundials.CVODE_BDF{:Newton,:Dense,Nothing,Nothing}}(:alg => Su
ndials.CVODE_BDF{:Newton,:Dense,Nothing,Nothing}(0, 0, 0, false, 10, 5, 7,
3, 10, nothing, nothing, 0))
 Dict{Symbol,Sundials.CVODE_BDF{:Newton,:LapackDense,Nothing,Nothing}}(:alg
 => Sundials.CVODE_BDF{:Newton,:LapackDense,Nothing,Nothing}(0, 0, 0, false
, 10, 5, 7, 3, 10, nothing, nothing, 0))
 Dict{Symbol,OrdinaryDiffEq.KenCarp4{0,false,DiffEqBase.DefaultLinSolve,Dif
fEqBase.NLNewton{Rational{Int64},Rational{Int64},Rational{Int64}},DataType}
}(:alg => OrdinaryDiffEq.KenCarp4{0,false,DiffEqBase.DefaultLinSolve,DiffEq
Base.NLNewton{Rational{Int64},Rational{Int64},Rational{Int64}},DataType}(Di
ffEqBase.DefaultLinSolve(nothing, nothing), DiffEqBase.NLNewton{Rational{In
t64},Rational{Int64},Rational{Int64}}(1//100, 10, 1//5, 1//5), Val{:forward
}, true, :linear, :PI))
```

0.5 Automatic Jacobian Solves

Due to the computational cost of the problem, we are only going to focus on the methods which demonstrated computational efficiency on the smaller biochemical benchmark problems. This excludes the exponential integrator, stabilized explicit, and extrapolation classes of methods.

First we test using auto-generated Jacobians (finite difference)

```
wp = WorkPrecisionSet(oprob,abstols,reltols,setups;error_estimate=:l2,
                      saveat=tf/10000.,appxsol=test_sol,maxiters=Int(1e5),numruns=1)
plot(wp)
```

```
Error: UndefVarError: tf not defined
```

0.6 Analytical Jacobian

Now we test using the generated analytic Jacobian function.

```
wp = WorkPrecisionSet(densejacprob,abstols,reltols,setups;error_estimate=:l2,
                      saveat=tf/10000.,appxsol=test_sol,maxiters=Int(1e5),numruns=1)
plot(wp)
```

```
Error: UndefVarError: tf not defined
```

0.7 Sparse Jacobian

Finally we test using the generated sparse analytic Jacobian function.

```
setups = [
          #Dict(:alg=>Rosenbrock23(autodiff=false)),
          Dict(:alg=>TRBDF2(autodiff=false)),
          #Dict(:alg=>CVODE_BDF(linear_solver=:KLU)),
```

```
          #Dict(:alg=>rodas()),
          #Dict(:alg=>radau()),
          #Dict(:alg=>Rodas4(autodiff=false)),
          #Dict(:alg=>Rodas5(autodiff=false)),
          Dict(:alg=>KenCarp4(autodiff=false)),
          #Dict(:alg=>RadauIIA5(autodiff=false)),
          #Dict(:alg=>lsoda()),
          ]
wp = WorkPrecisionSet(sparsejacprob,abstols,reltols,setups;error_estimate=:l2,
                      saveat=tf/10000.,appxsol=test_sol,maxiters=Int(1e5),numruns=1)
plot(wp)
```

```
Error: UndefVarError: tf not defined
```