Burgers Pseudospectral Methods Work-Precision Diagrams

HAO HAO

December 20, 2019

```
using ApproxFun, OrdinaryDiffEq, Sundials
using DiffEqDevTools
using LinearAlgebra
using Plots; gr()
```

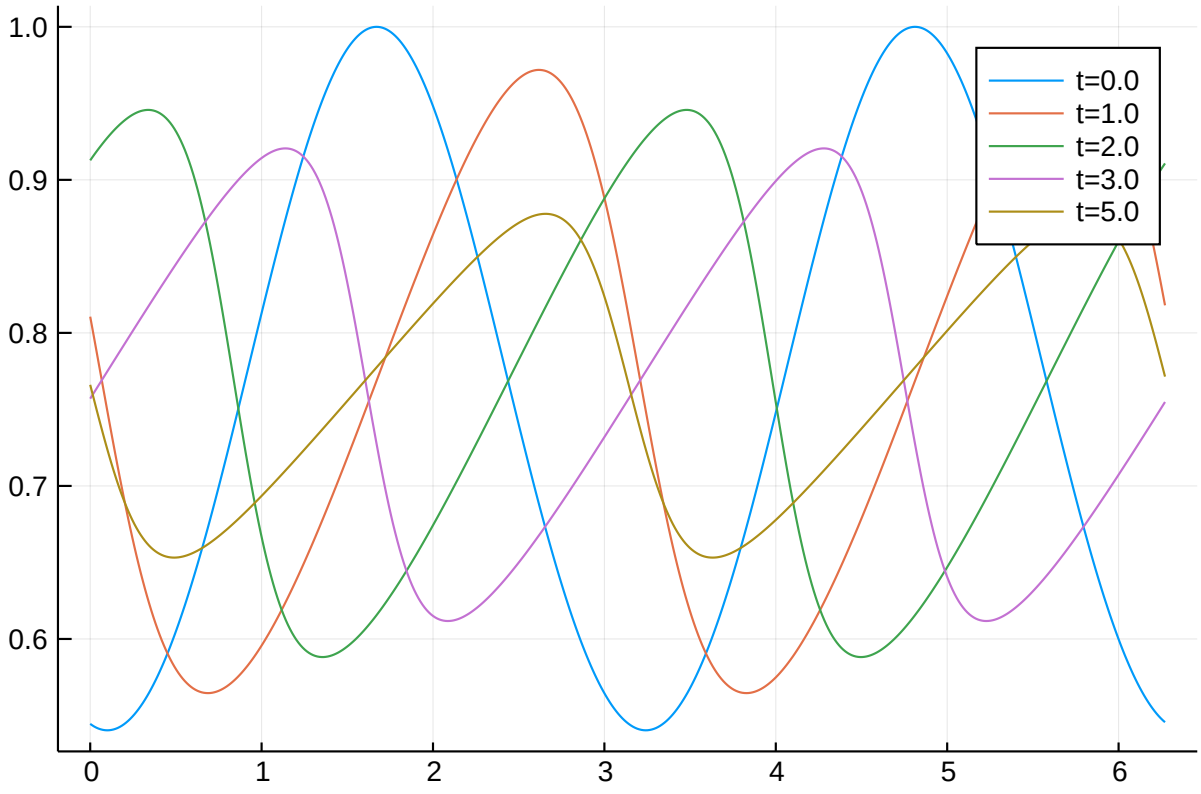
Here is the Burgers equation using Fourier spectral methods.

```
S = Fourier()
n = 512
x = points(S, n)
D2 = Derivative(S,2)[1:n,1:n]
D  = (Derivative(S) → S)[1:n,1:n]
T = ApproxFun.plan_transform(S, n)
Ti = ApproxFun.plan_itransform(S, n)

û_0 = T*cos.(cos.(x.-0.1))
A = 0.03*D2
tmp = similar(û_0)
p = (D,D2,T,Ti,tmp,similar(tmp))
function burgers_nl(dû,û,p,t)
    D,D2,T,Ti,u,tmp = p
    mul!(tmp, D, û)
    mul!(u, Ti, tmp)
    mul!(tmp, Ti, û)
    @. tmp = tmp*u
    mul!(u, T, tmp)
    @. dû = - u
end
```

```
burgers_nl (generic function with 1 method)
```

Reference solution using Rodas5 is below:

```
prob = SplitODEProblem(DiffEqArrayOperator(Diagonal(A)), burgers_nl, û_0, (0.0,5.0), p)
sol  = solve(prob, Rodas5(autodiff=false); reltol=1e-12,abstol=1e-12)
test_sol = TestSolution(sol)

tslices=[0.0 1.0 2.0 3.0 5.0]
ys=hcat((Ti*sol(t) for t in tslices)...)
labels=["t=$t" for t in tslices]
plot(x,ys,label=labels)
```



0.1 High tolerances

```
diag_linsolve=LinSolveFactorize(W->let tmp = tmp
    for i in 1:size(W, 1)
        tmp[i] = W[i, i]
    end
    Diagonal(tmp)
end)
```

```
DiffEqBase.LinSolveFactorize{Main.WeaveSandBox11.var"#5#6"}(Main.WeaveSandB
ox11.var"#5#6"(), nothing)
```

0.2 In-family comparisons

1.IMEX methods (diagonal linear solver)

```
abstols = 0.1 .^ (5:8)
reltols = 0.1 .^ (1:4)
multipliers =  0.5 .^ (0:3)
setups = [Dict(:alg => IMEXEuler(linsolve=diag_linsolve), :dts => 1e-3 * multipliers),
          Dict(:alg => CNAB2(linsolve=diag_linsolve), :dts => 5e-3 * multipliers),
          Dict(:alg => CNLF2(linsolve=diag_linsolve), :dts => 5e-3 * multipliers),
          Dict(:alg => SBDF2(linsolve=diag_linsolve), :dts => 1e-3 * multipliers)]
labels = ["IMEXEuler" "CNAB2" "CNLF2" "SBDF2"]
@time wp1 = WorkPrecisionSet(prob,abstols,reltols,setups;
                             print_names=true,names=labels,
                             numruns=5,seconds=5,
                             save_everystop=false,appxsol=test_sol,maxiters=Int(1e5));
```

```
IMEXEuler
Error: LinearAlgebra.SingularException(504)
```

```
plot(wp1,label=labels,markershape=:auto,title="IMEX methods, diagonal linsolve, low
order")
```

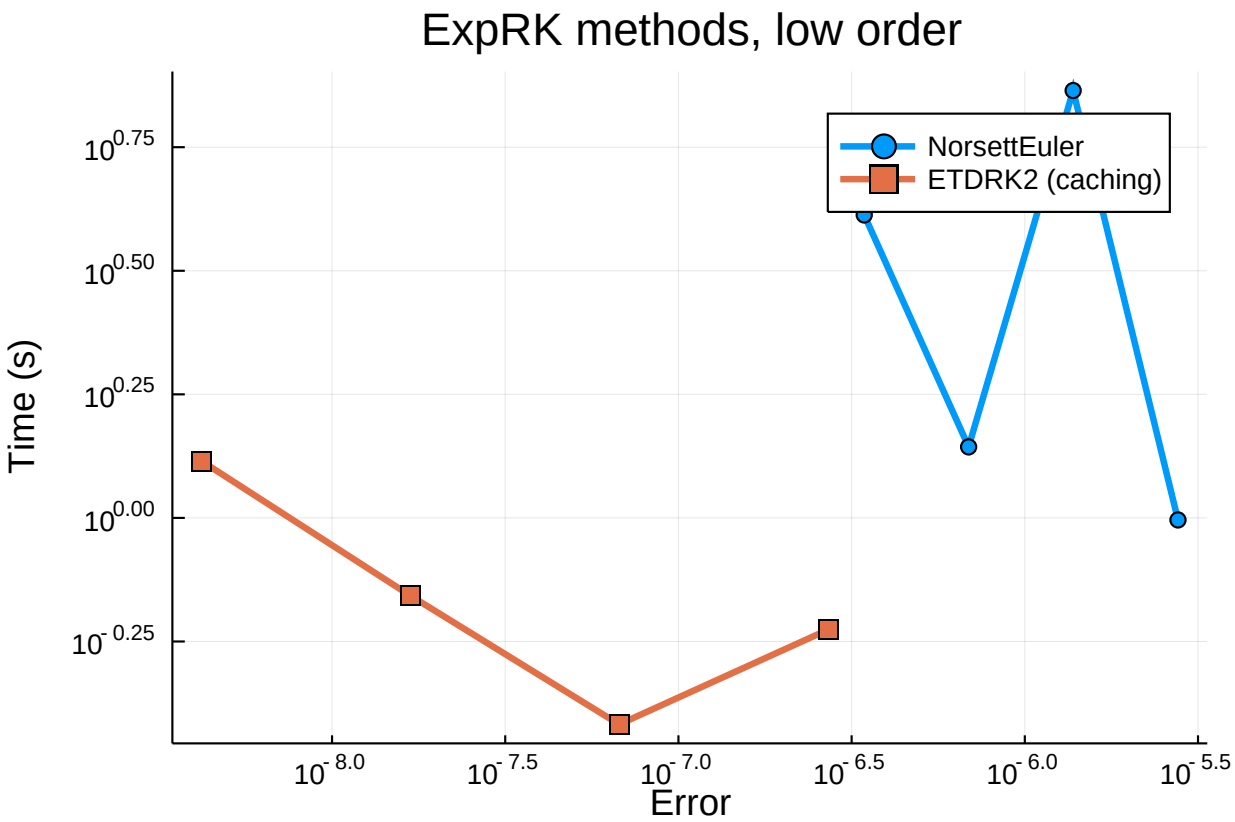
```
Error: UndefVarError: wp1 not defined
```

2. ExpRK methods

```
abstols = 0.1 .^ (5:8) # all fixed dt methods so these don't matter much
reltols = 0.1 .^ (1:4)
multipliers = 0.5 .^ (0:3)
setups = [Dict(:alg => NorsettEuler(), :dts => 1e-3 * multipliers),
          Dict(:alg => ETDRK2(), :dts => 1e-2 * multipliers)]
labels = hcat("NorsettEuler",
              "ETDRK2 (caching)")
@time wp2 = WorkPrecisionSet(prob,abstols,reltols,setups;
                             print_names=true, names=labels,
                             numruns=5,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));
```

```
NorsettEuler
ETDRK2 (caching)
117.053941 seconds (68.58 M allocations: 5.212 GiB, 0.92% gc time)
```

```
plot(wp2, label=labels, markershape=:auto, title="ExpRK methods, low order")
```



0.3 Between family comparisons

```
abstols = 0.1 .^ (5:8) # all fixed dt methods so these don't matter much
reltols = 0.1 .^ (1:4)
multipliers = 0.5 .^ (0:3)
setups = [Dict(:alg => CNAB2(linsolve=diag_linsolve), :dts => 5e-3 * multipliers),
          Dict(:alg => ETDRK2(), :dts => 1e-2 * multipliers)]
labels = ["CNAB2 (diagonal linsolve)" "ETDRK2"]
@time wp3 = WorkPrecisionSet(prob,abstols,reltols,setups;
                             print_names=true, names=labels,
                             numruns=5, error_estimate=:l2,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));
```

```
CNAB2 (diagonal linsolve)
Error: LinearAlgebra.SingularException(264)
```

```
plot(wp3, label=labels, markershape=:auto, title="Between family, low orders")
```

```
Error: UndefVarError: wp3 not defined
```

0.4 Low tolerances

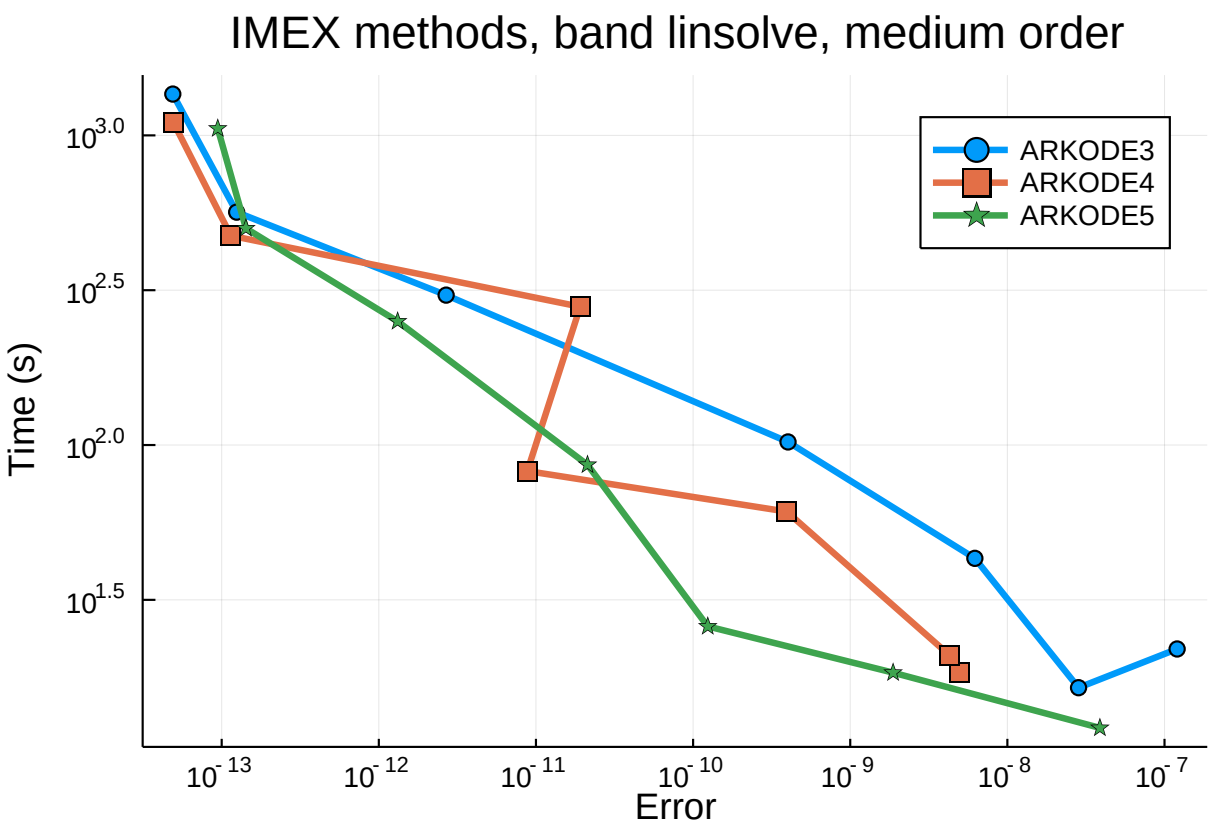
0.5 In-family comparisons

1.IMEX methods (band linear solver)

```
abstols = 0.1 .^ (7:13)
reltols = 0.1 .^ (4:10)
setups = [Dict(:alg => ARKODE(Sundials.Implicit(), order=3, linear_solver=:Band,
jac_upper=1, jac_lower=1)),
          Dict(:alg => ARKODE(Sundials.Implicit(), order=4, linear_solver=:Band,
jac_upper=1, jac_lower=1)),
          Dict(:alg => ARKODE(Sundials.Implicit(), order=5, linear_solver=:Band,
jac_upper=1, jac_lower=1))]
labels = hcat("ARKODE3", "ARKODE4", "ARKODE5")
@time wp4 = WorkPrecisionSet(prob,abstols,reltols,setups;
                             print_names=true, names=labels,
                             numruns=5, error_estimate=:l2,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));
```

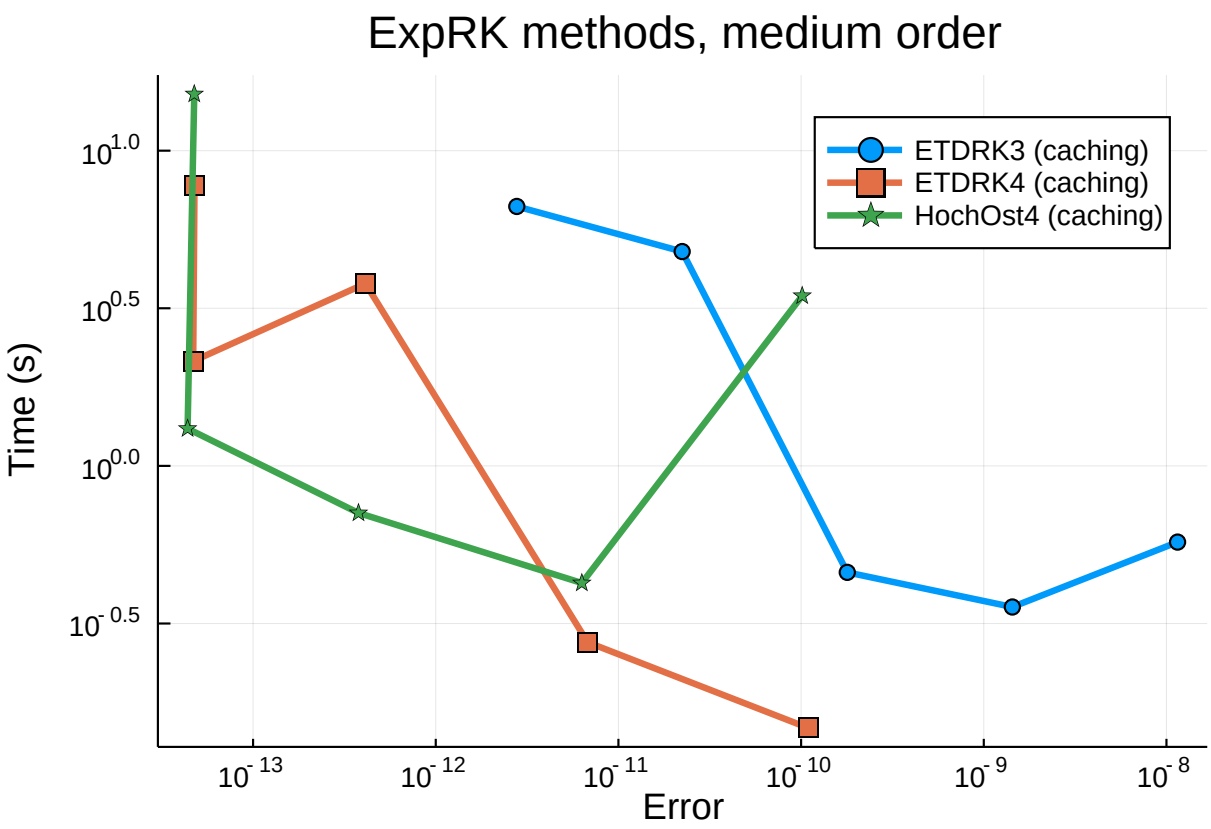
```
ARKODE3
ARKODE4
ARKODE5
19740.877027 seconds (7.42 G allocations: 592.938 GiB, 0.62% gc time)
```

```
plot(wp4, label=labels, markershape=:auto, title="IMEX methods, band linsolve, medium
order")
```



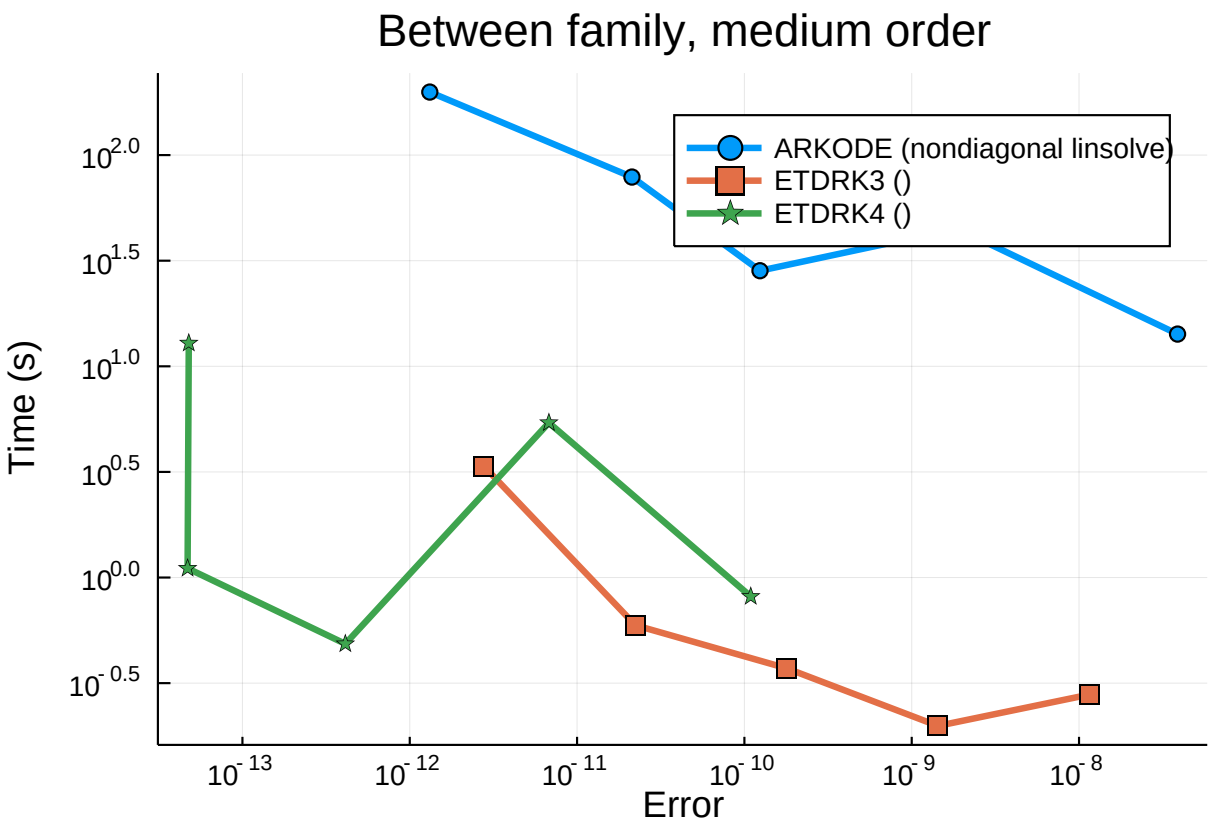
2.ExpRK methods

```
abstols = 0.1 .^ (7:11) # all fixed dt methods so these don't matter much
reltols = 0.1 .^ (4:8)
multipliers = 0.5 .^ (0:4)
setups = [Dict(:alg => ETDRK3(), :dts => 1e-2 * multipliers),
          Dict(:alg => ETDRK4(), :dts => 1e-2 * multipliers),
          Dict(:alg => HochOst4(), :dts => 1e-2 * multipliers)]
labels = hcat("ETDRK3 (caching)", "ETDRK4 (caching)",
              "HochOst4 (caching)")
@time wp5 = WorkPrecisionSet(prob,abstols,reltols,setups;
                             print_names=true, names=labels,
                             numruns=5, error_estimate=:l2,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));

ETDRK3 (caching)
ETDRK4 (caching)
HochOst4 (caching)
249.685117 seconds (108.76 M allocations: 8.310 GiB, 0.65% gc time)

plot(wp5, label=labels, markershape=:auto, title="ExpRK methods, medium order")
```



0.6 Between family comparisons

```
abstols = 0.1 .^ (7:11)
reltols = 0.1 .^ (4:8)
multipliers = 0.5 .^ (0:4)
setups = [Dict(:alg => ARKODE(Sundials.Implicit(), order=5, linear_solver=:Band,
jac_upper=1, jac_lower=1)),
          Dict(:alg => ETDRK3(), :dts => 1e-2 * multipliers),
          Dict(:alg => ETDRK4(), :dts => 1e-2 * multipliers)]
labels = hcat("ARKODE (nondiagonal linsolve)", "ETDRK3 ()", "ETDRK4 ()")
                        #"ARKODE (Krylov linsolve)")
@time wp6 = WorkPrecisionSet(prob,abstols,reltols,setups;
                             print_names=true, names=labels,
                             numruns=5, error_estimate=:l2,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));

ARKODE (nondiagonal linsolve)
ETDRK3 ()
ETDRK4 ()
1279.024139 seconds (513.10 M allocations: 41.074 GiB, 0.65% gc time)

plot(wp6, label=labels, markershape=:auto, title="Between family, medium order")
```




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

0.7 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("MOLPDE","burgers_spectral_wpd.jmd")
```

Computer Information:

```
Julia Version 1.3.0
Commit 46ce4d7933 (2019-11-26 06:09 UTC)
Platform Info:
  OS: Linux (x86_64-pc-linux-gnu)
  CPU: Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-6.0.1 (ORCJIT, skylake)
Environment:
  JULIA_NUM_THREADS = 8
```

Package Information:

```
Status: `/home/chrisrackauckas/.julia/dev/DiffEqBenchmarks/Project.toml`
```