Burgers Pseudospectral Methods Work-Precision Diagrams

HAO HAO

December 30, 2019

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

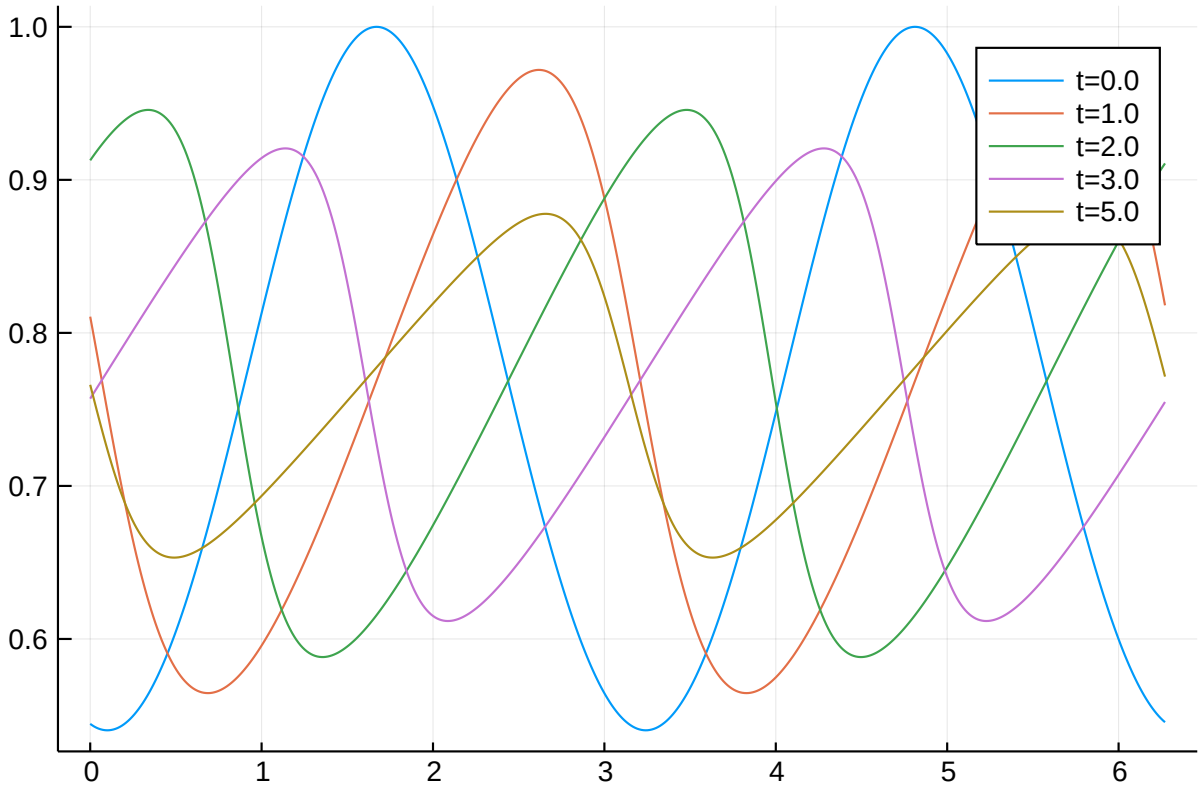
Here is the Burgers equation using Fourier spectral methods.

```
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.WeaveSandBox4.var"#5#6"}(Main.WeaveSandBo
x4.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")
```

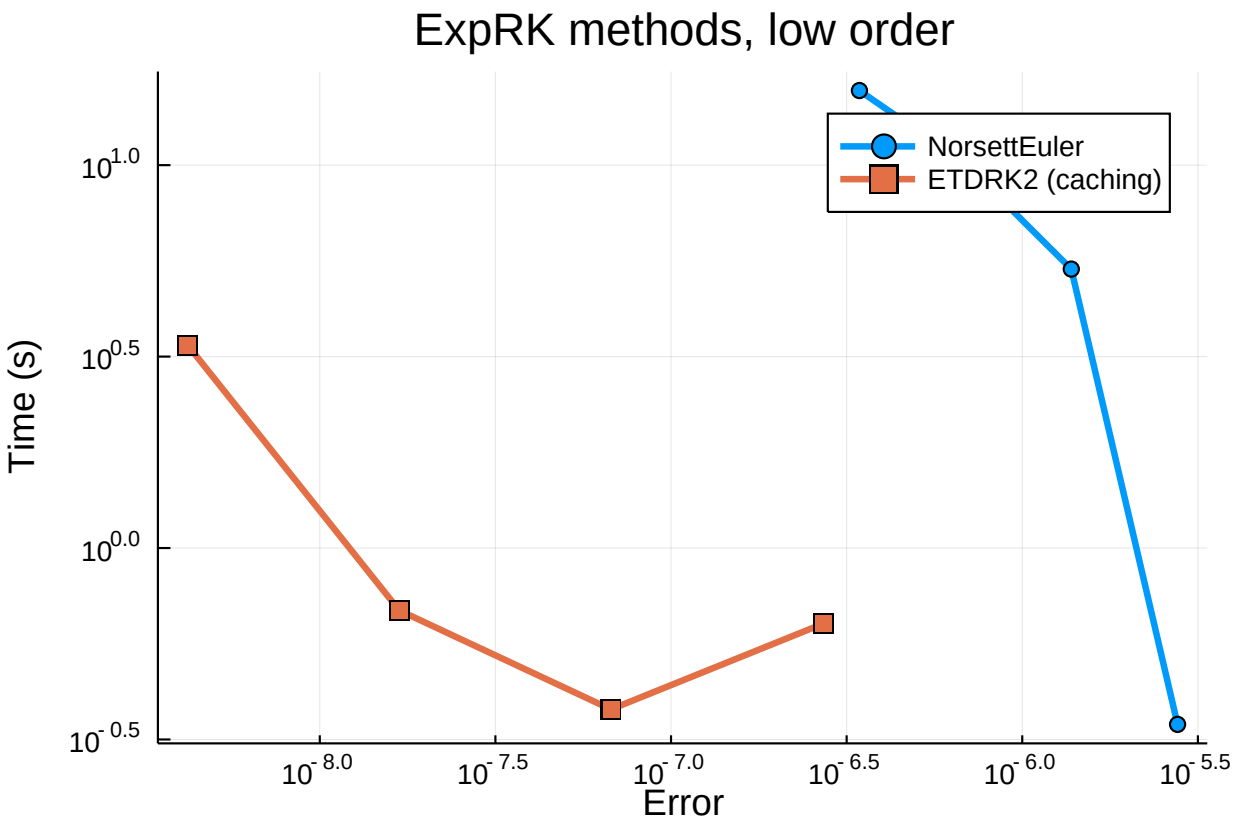
```
Error: UndefVarError: wp1 not defined
```

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)
133.994862 seconds (53.16 M allocations: 3.981 GiB, 0.65% 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

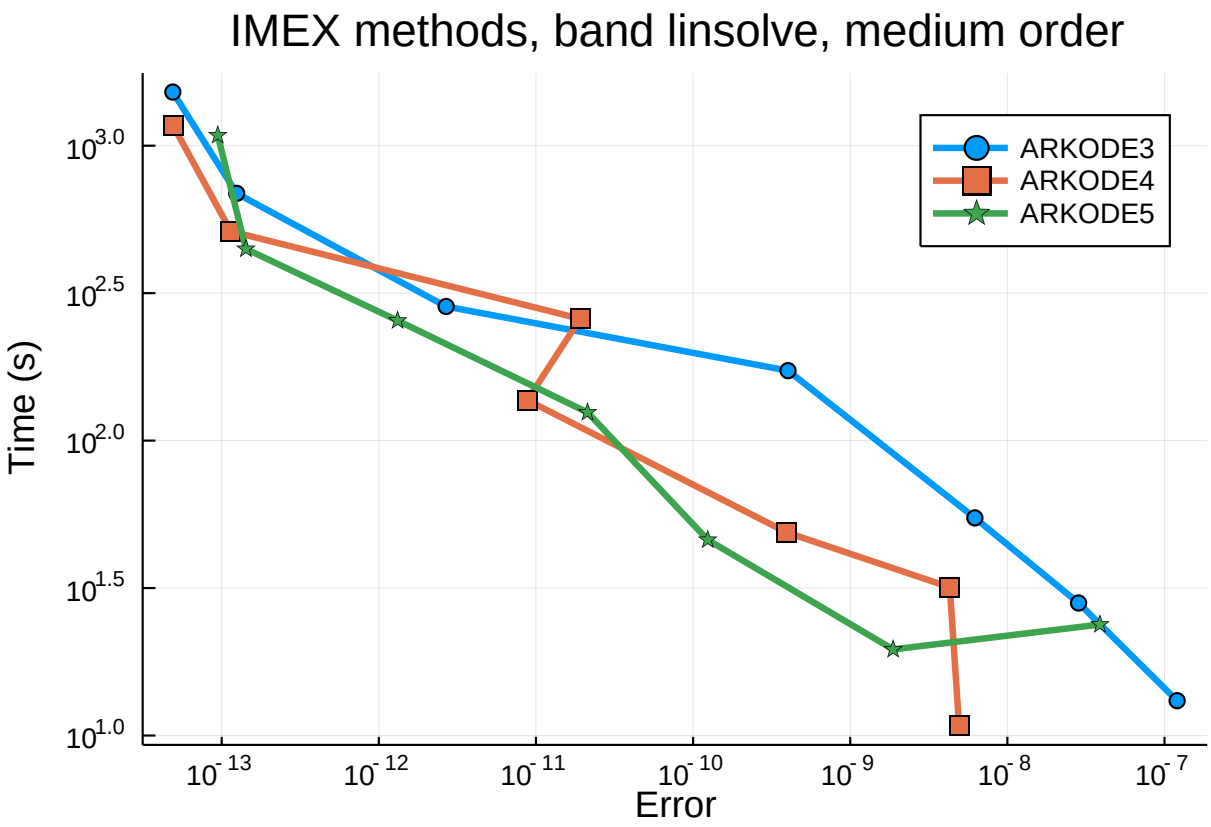
0.5 In-family comparisons

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));
```

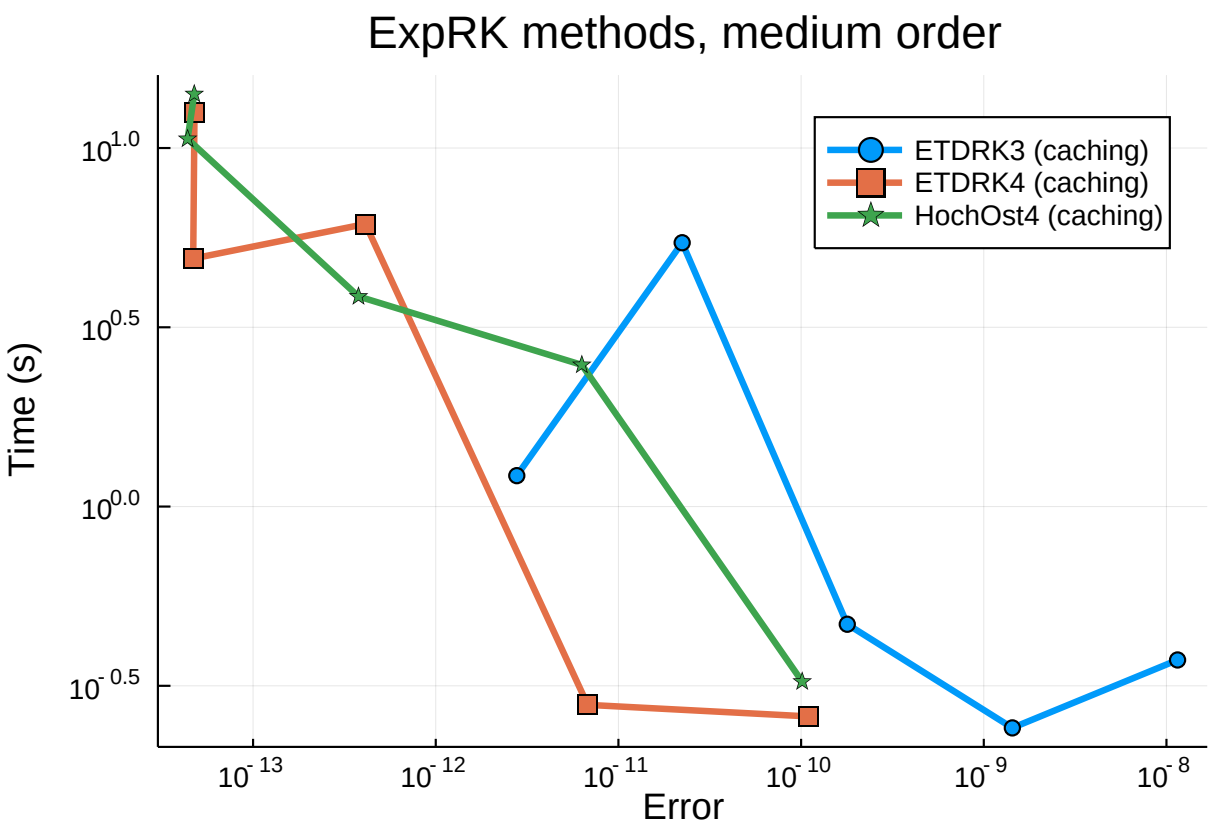
```
ARKODE3
ARKODE4
ARKODE5
20708.538189 seconds (7.42 G allocations: 592.850 GiB, 0.67% gc time)
```

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



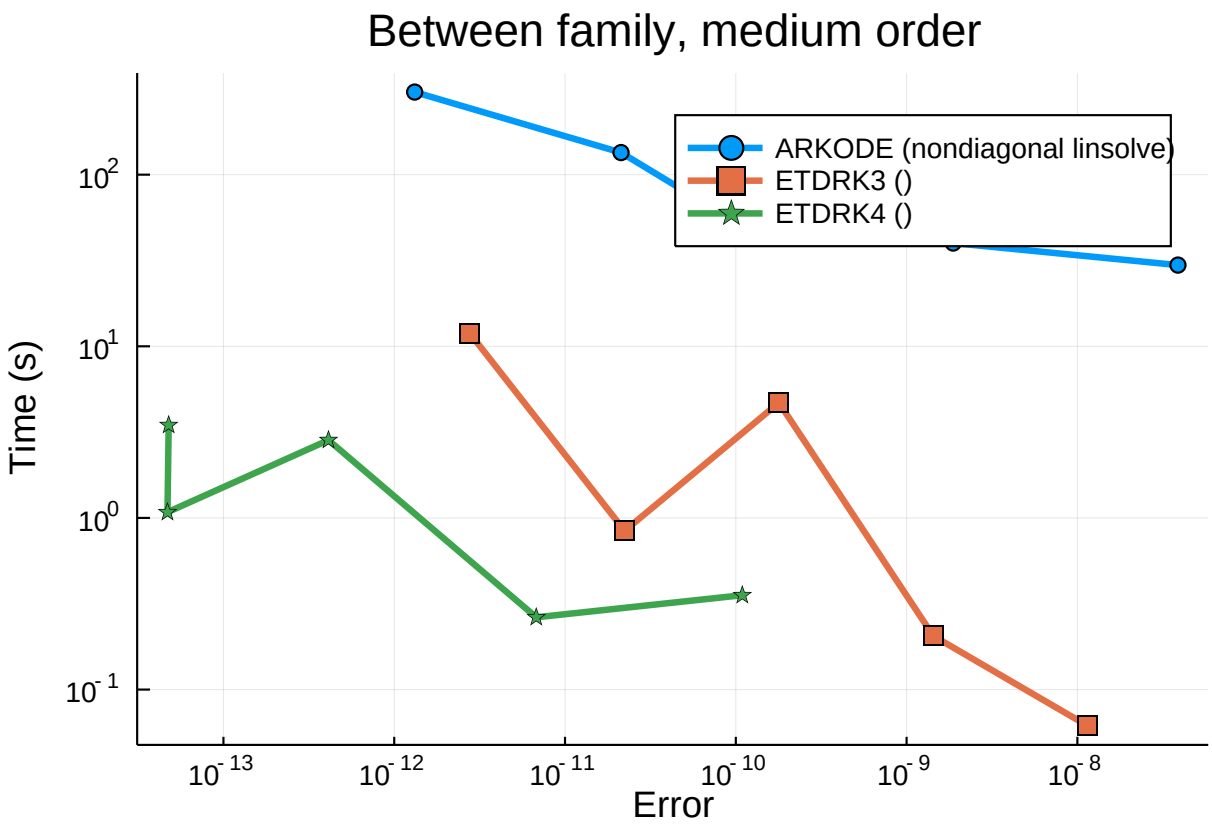
2.ExpRK methods

```
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)
241.950029 seconds (104.56 M allocations: 7.972 GiB, 0.71% 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 ()
1567.620429 seconds (507.71 M allocations: 40.677 GiB, 0.60% 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`
[28f2ccd6-bb30-5033-b560-165f7b14dc2f] ApproxFun 0.11.8
[a134a8b2-14d6-55f6-9291-3336d3ab0209] BlackBoxOptim 0.5.0
[a93c6f00-e57d-5684-b7b6-d8193f3e46c0] DataFrames 0.20.0
[2b5f629d-d688-5b77-993f-72d75c75574e] DiffEqBase 6.10.0
[eb300fae-53e8-50a0-950c-e21f52c2b7e0] DiffEqBiological 4.1.0
[f3b72e0c-5b89-59e1-b016-84e28bfd966d] DiffEqDevTools 2.16.1
[c894b116-72e5-5b58-be3c-e6d8d4ac2b12] DiffEqJump 6.4.0
[1130ab10-4a5a-5621-a13d-e4788d82bd4c] DiffEqParamEstim 1.10.0
[a077e3f3-b75c-5d7f-a0c6-6bc4c8ec64a9] DiffEqProblemLibrary 4.6.4
[ef61062a-5684-51dc-bb67-a0fcdec5c97d] DiffEqUncertainty 1.4.0
[0c46a032-eb83-5123-abaf-570d42b7fbaa] DifferentialEquations 6.9.0
[7073ff75-c697-5162-941a-fcdaad2a7d2a] IJulia 1.20.2
[7f56f5a3-f504-529b-bc02-0b1fe5e64312] LSODA 0.6.1
[76087f3c-5699-56af-9a33-bf431cd00edd] NLopt 0.5.1
[c030b06c-0b6d-57c2-b091-7029874bd033] ODE 2.6.0
[54ca160b-1b9f-5127-a996-1867f4bc2a2c] ODEInterface 0.4.6
[09606e27-ecf5-54fc-bb29-004bd9f985bf] ODEInterfaceDiffEq 3.5.0
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.26.7
[2dcacdae-9679-587a-88bb-8b444fb7085b] ParallelDataTransfer 0.5.0
[65888b18-ceab-5e60-b2b9-181511a3b968] ParameterizedFunctions 4.2.1
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 0.28.4
[b4db0fb7-de2a-5028-82bf-5021f5cfa881] ReactionNetworkImporters 0.1.5
[f2c3362d-daeb-58d1-803e-2bc74f2840b4] RecursiveFactorization 0.1.0
[9672c7b4-1e72-59bd-8a11-6ac3964bc41f] SteadyStateDiffEq 1.5.0
[c3572dad-4567-51f8-b174-8c6c989267f4] Sundials 3.8.1
[a759f4b9-e2f1-59dc-863e-4aeb61b1ea8f] TimerOutputs 0.5.3
[44d3d7a6-8a23-5bf8-98c5-b353f8df5ec9] Weave 0.9.1
[b77e0a4c-d291-57a0-90e8-8db25a27a240] InteractiveUtils
[d6f4376e-aef5-505a-96c1-9c027394607a] Markdown
[44cfe95a-1eb2-52ea-b672-e2afdf69b78f] Pkg
[9a3f8284-a2c9-5f02-9a11-845980a1fd5c] Random
```