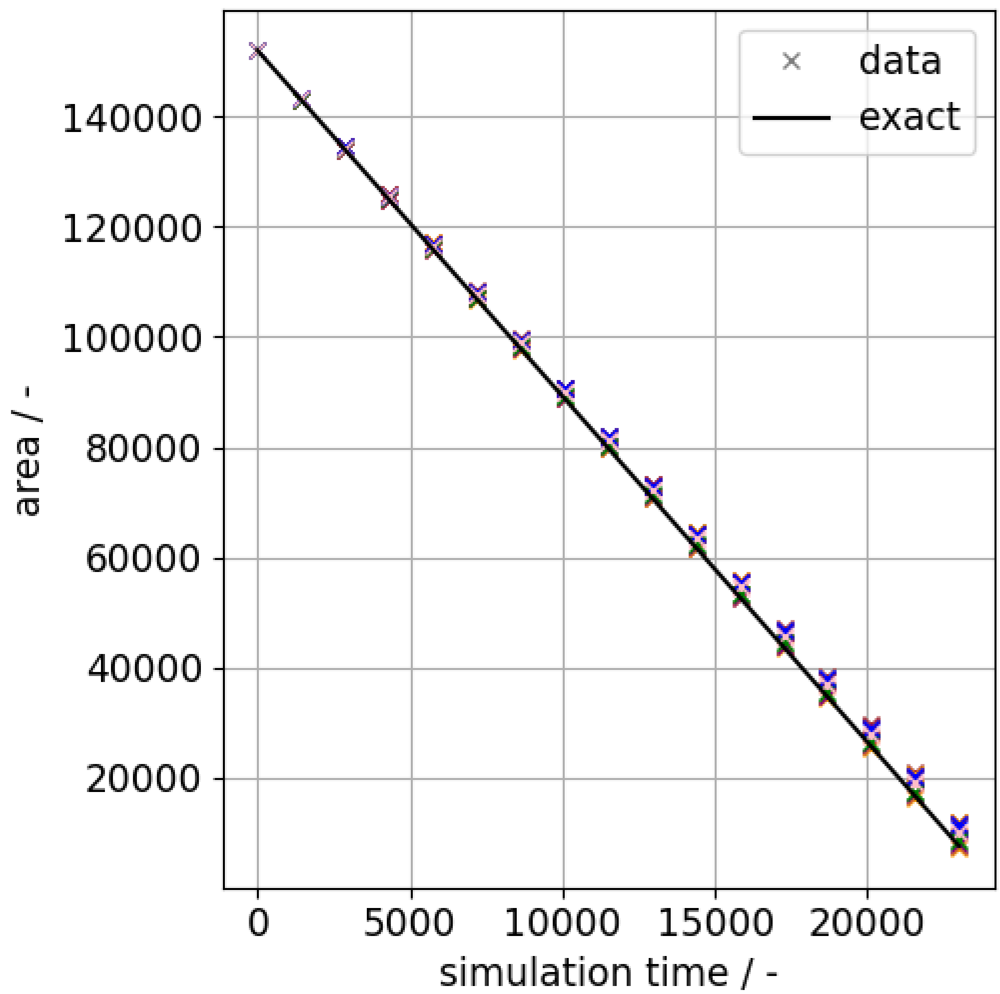
This notebook investigates the single grain growth case described in section 4.3 of the paper.

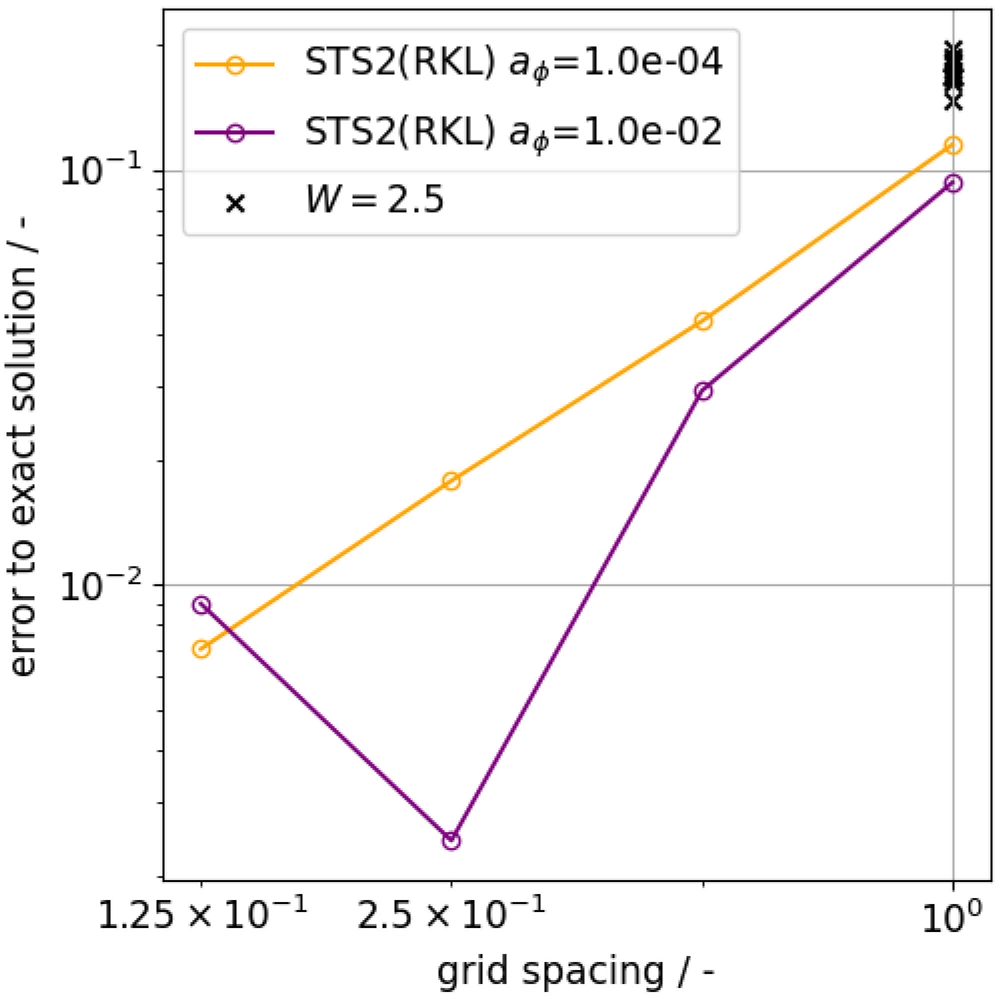
```
found  graingrain2d_MPF(SSP(10)4)_rtol1.00e-14_atol1.00e-14_Dbulk1.00e+00_W2.50e+00_d
x1.00e+00_nx512_cu_ as reference
```

From the following plot we can see that all results match qualitatively, though the quantitativeness depends on the details.

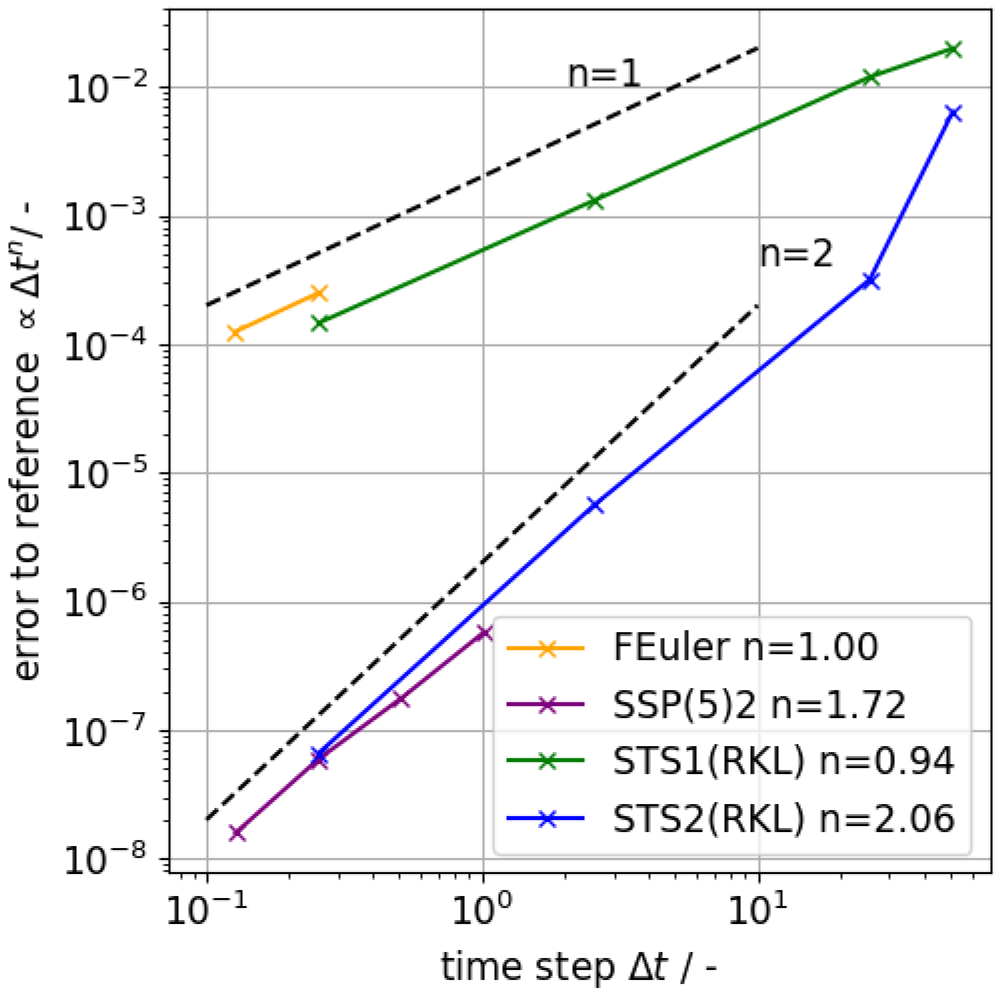



```
graingrain2d_MPF(STS1(RKL))_rtol1.00e-04_atol1.00e-02_Dbulk1.00e+00_W2.50e+00_dx1.00e
+00_nx512_cu_ has highest relative error with 0.03114701603179525
```

The convergence study shows that indeed we converge to the desired sharp interface limit, though the integration tolerance can limit this as seen by the uptick for the looser tolerance.

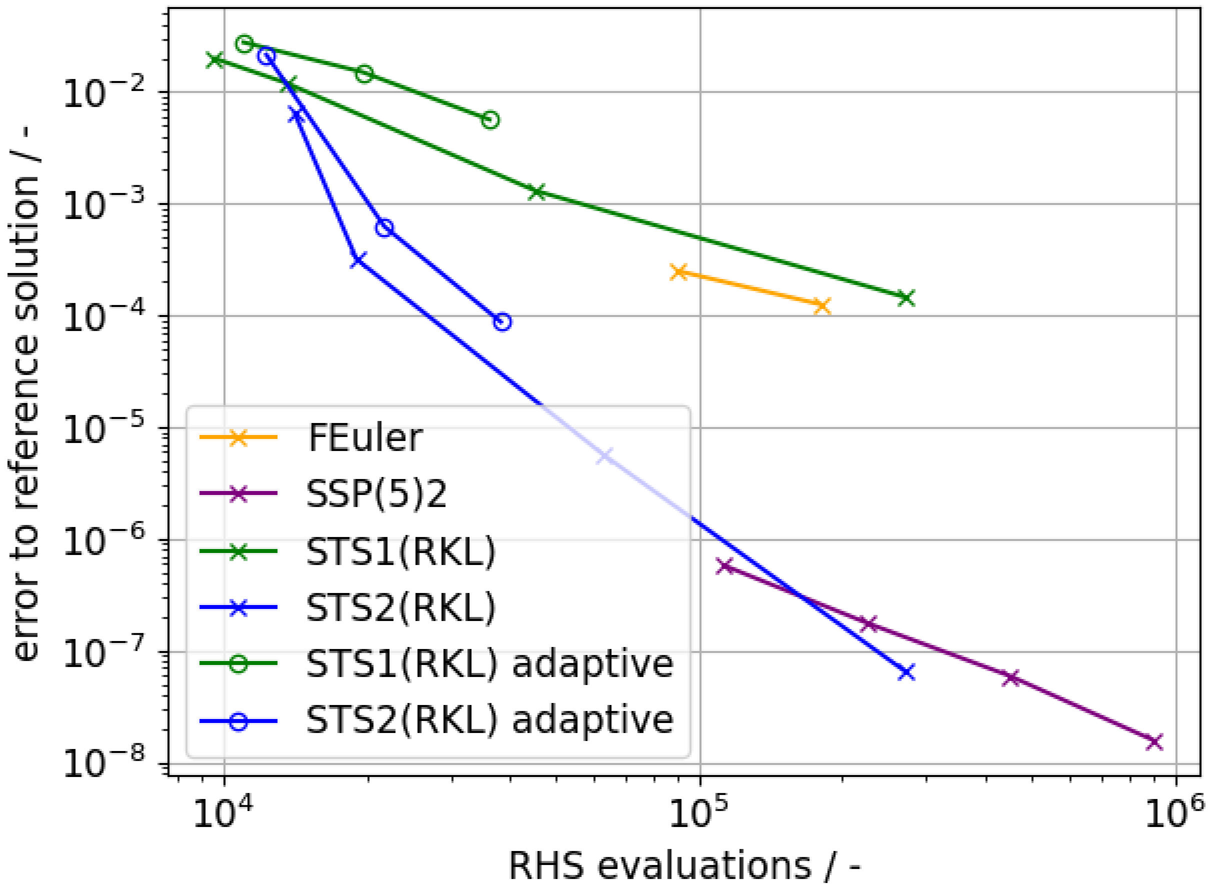


Now focusing on only the practical resolution results, we can check for the correct integration order based on the results by comparing against the reference integrator at tight tolerances. We find that the orders are generally recovered, though with some deviation at larger time steps.



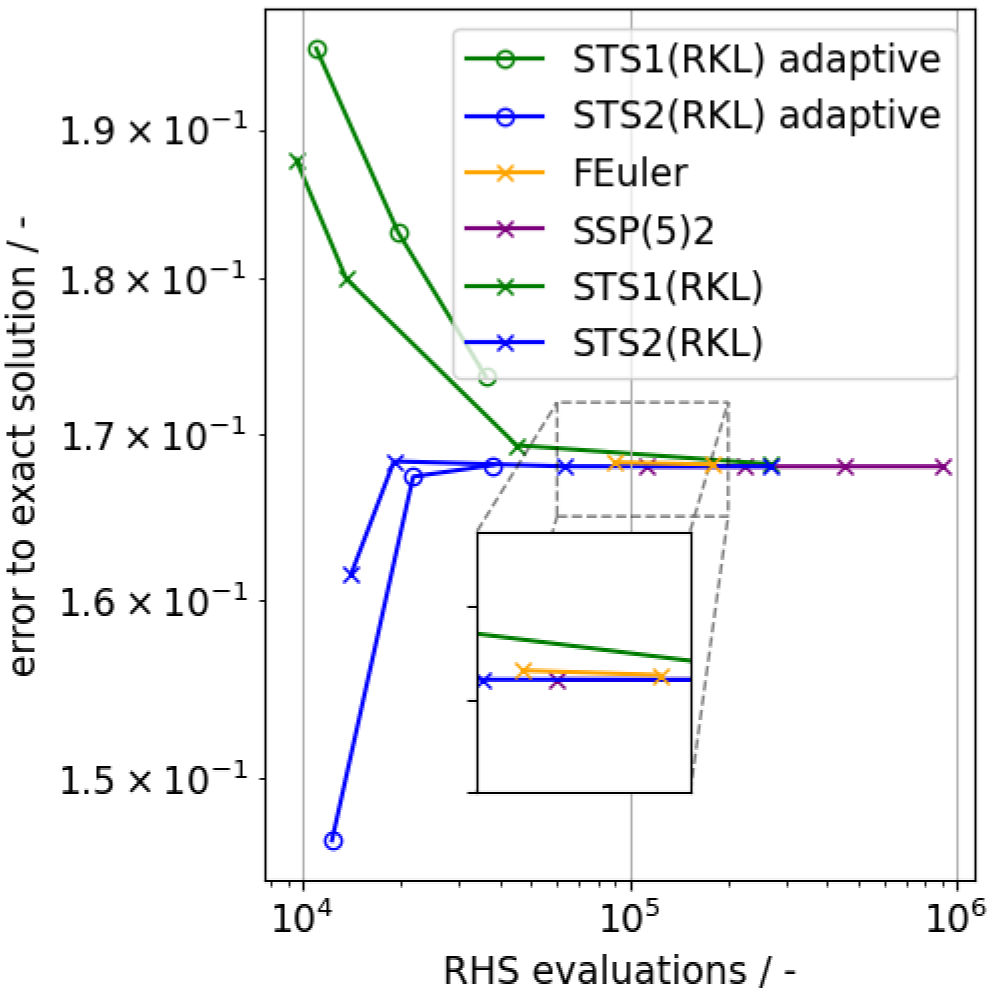

```
/tmp/ipykernel_31554/2130388879.py:6: OptimizeWarning: Covariance of the parameters c
ould not be estimated
  isol = optimize.curve_fit(linpow, np.log(iv["dt"]), np.log(iv["error"]), p0=(1.5,
0.1))
```

A standard work-precision chart by comparing to the reference solution is shown next, with more usual curves for these being reproduced. However, we are not interested in the zero time integration error case, but rather in the error w.r.t. the sharp interface problem, which is why we don't use this.



Comparing against the sharp interface solution shows that cleary the spatial and interface width related effects dominate over any time integration error; only at the largest time steps is there a relevant influence.

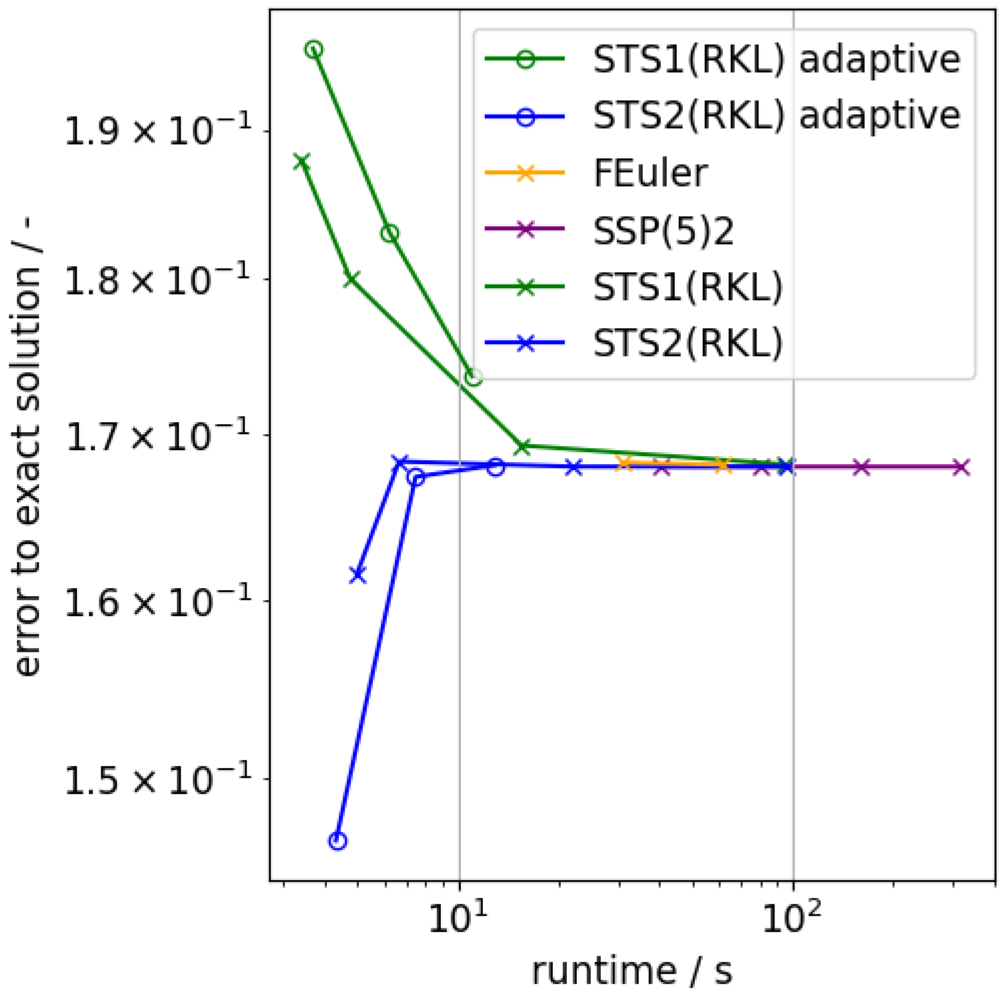
We observe with this, as well as the prior chart, that the STS integrators are generally more efficient than the forward Euler integrator.




```
adaptive STS1(RKL)(MPF::Obstacle) speedup 8.19095203488372 rejection ratio 0.27380952
380952384 0.2711693548387097 0.2967788635541079
adaptive STS2(RKL)(MPF::Obstacle) speedup 7.33295380611581 rejection ratio 0.17667844
522968199 0.179456906729634 0.19047619047619047
fixed FEuler(MPF::Obstacle) speedup 1.0
fixed SSP(5)2(MPF::Obstacle) speedup 0.7999112845990064
fixed STS1(RKL)(MPF::Obstacle) speedup 9.439489112227806
fixed STS2(RKL)(MPF::Obstacle) speedup 6.396566401816118
```

We also plot this over the runtime for consistency and to show that the RHS and runtime are well correlated, since the charts and reported speedups are quite similar. This is simply since the RHS evaluation is the most expensive step, with the additional vector FMAs from the STS integration schemes being relatively cheap.

Note that for small domain sizes / integration times, the I/O part may be significant which can cause the RHS count - runtime correlation for explicit integrators to break down.




```
adaptive STS1(RKL)(MPF::Obstacle) speedup 8.392076723166628 rejection ratio 0.2738095
2380952384 0.2711693548387097 0.2967788635541079
adaptive STS2(RKL)(MPF::Obstacle) speedup 7.166437859481785 rejection ratio 0.1766784
4522968199 0.179456906729634 0.19047619047619047
fixed FEuler(MPF::Obstacle) speedup 1.0
fixed SSP(5)2(MPF::Obstacle) speedup 0.768459581774886
fixed STS1(RKL)(MPF::Obstacle) speedup 9.104143837587644
fixed STS2(RKL)(MPF::Obstacle) speedup 6.198417347057809
```