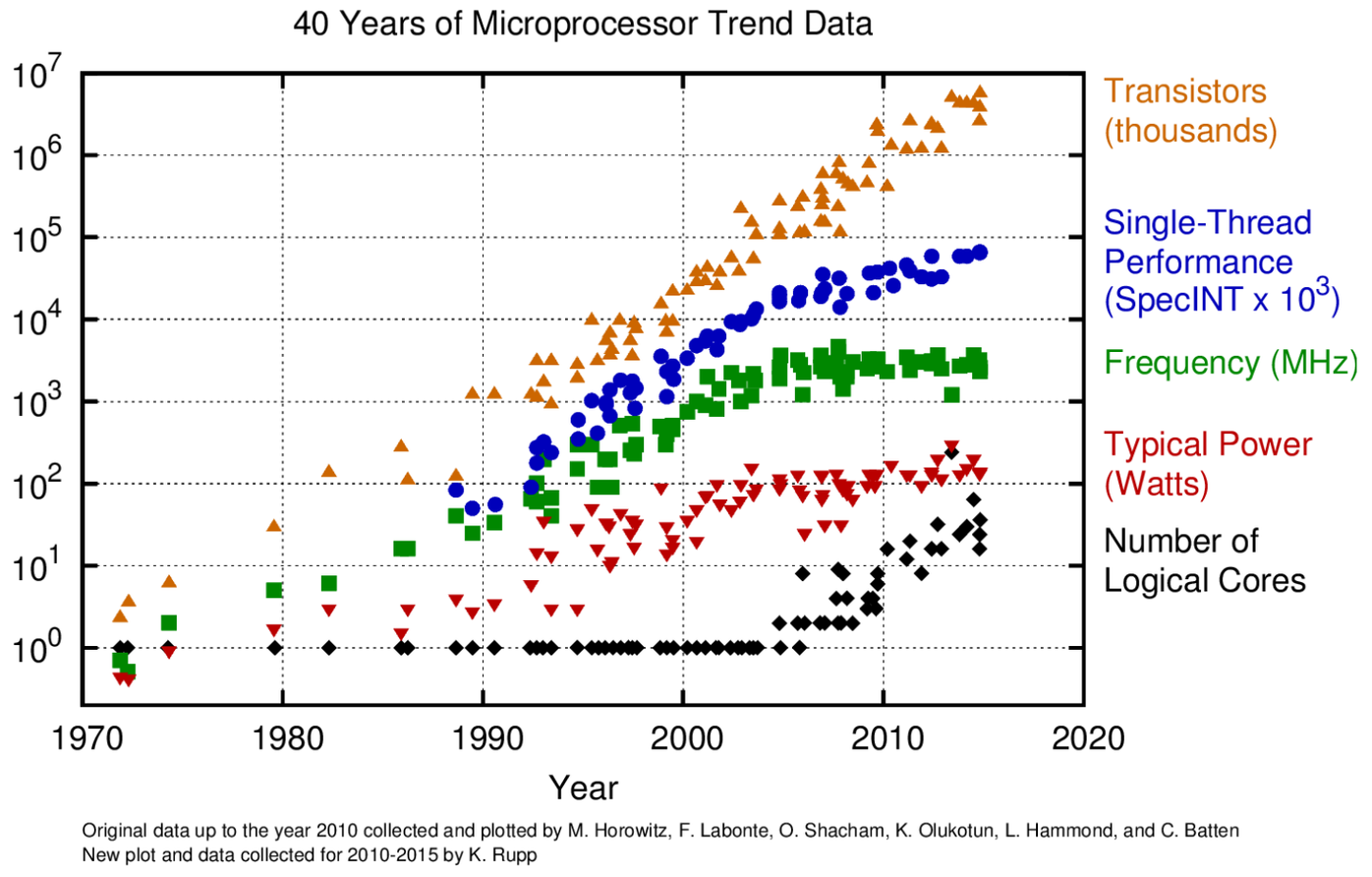
REXI: breaking the time step constraint

David Acreman,
Jemma Shipton, Colin Cotter and Beth Wingate

Why REXI?

- Trends in processor design are towards increasing number of cores
- Strong scaling of domain decomposition is limited
- Timestep limits weak scaling
- We need to find parallelism elsewhere



https://www.karlrupp.net/2015/06/40-years-of-microprocessor-trend-data/

Rational approximation of exponential integrator (REXI)

$$\frac{du}{dt} = \lambda u, \qquad u(t_0) = u_0, \qquad \lambda \in \mathbb{C}$$

Apply n forward Euler time steps

$$\tilde{u}(T) = \underbrace{(1 + \lambda \Delta t)^n}_{\approx e^{\lambda (T - t_0)}} u_0$$

Approximate the exponential

$$e^{\lambda t} \approx \sum_{k=-N}^{N} \frac{\beta_k}{\lambda t + \alpha_k}$$

α_k and β_k are pre-computed complex numbers. Terms in the summation can be calculated in parallel

Schreiber et al, 2017, Beyond spatial scalability limitations with a massively parallel method for linear oscillatory problems, International Journal of High Performance Computing Applications

Rational approximation of exponential integrator (REXI)

No. of Gaussians

Width of Gaussian

$$e^{ix} \approx \sum_{m=-M}^{M} b_m \psi_h(x + mh)$$

Approximate the exponential using Gaussian basis functions

$$= \sum_{m=-M}^{M} b_m \sum_{l=-L}^{L} \mathrm{Re}\left(\frac{h a_l}{ix + h(\mu + i(m+l))}\right)$$

Approximate Gaussians as sum of rational terms

a_l and μ are pre-computed constants (Haut et al, 2015)
Terms in the sum over M can be calculated in parallel

$$hM > |t\lambda_{MAX}|$$

Haut et al, 2015, A high-order time-parallel scheme for solving wave propagation problems via the direct construction of an approximate time-evolution operator, IMA Journal of Numerical Analysis (2016) 36, 688–716

REXI study

- REXI results presented in Schreiber et al (2017) for benchmark problems applied to shallow water equations

- We will also solve the shallow water equations but with some significant differences:

  - Finite difference or spectral → finite elements (Firedrake)

  - Regular unit square → icosahedral sphere in physical co-ordinates

  - Looking for speed up over conventional time stepping

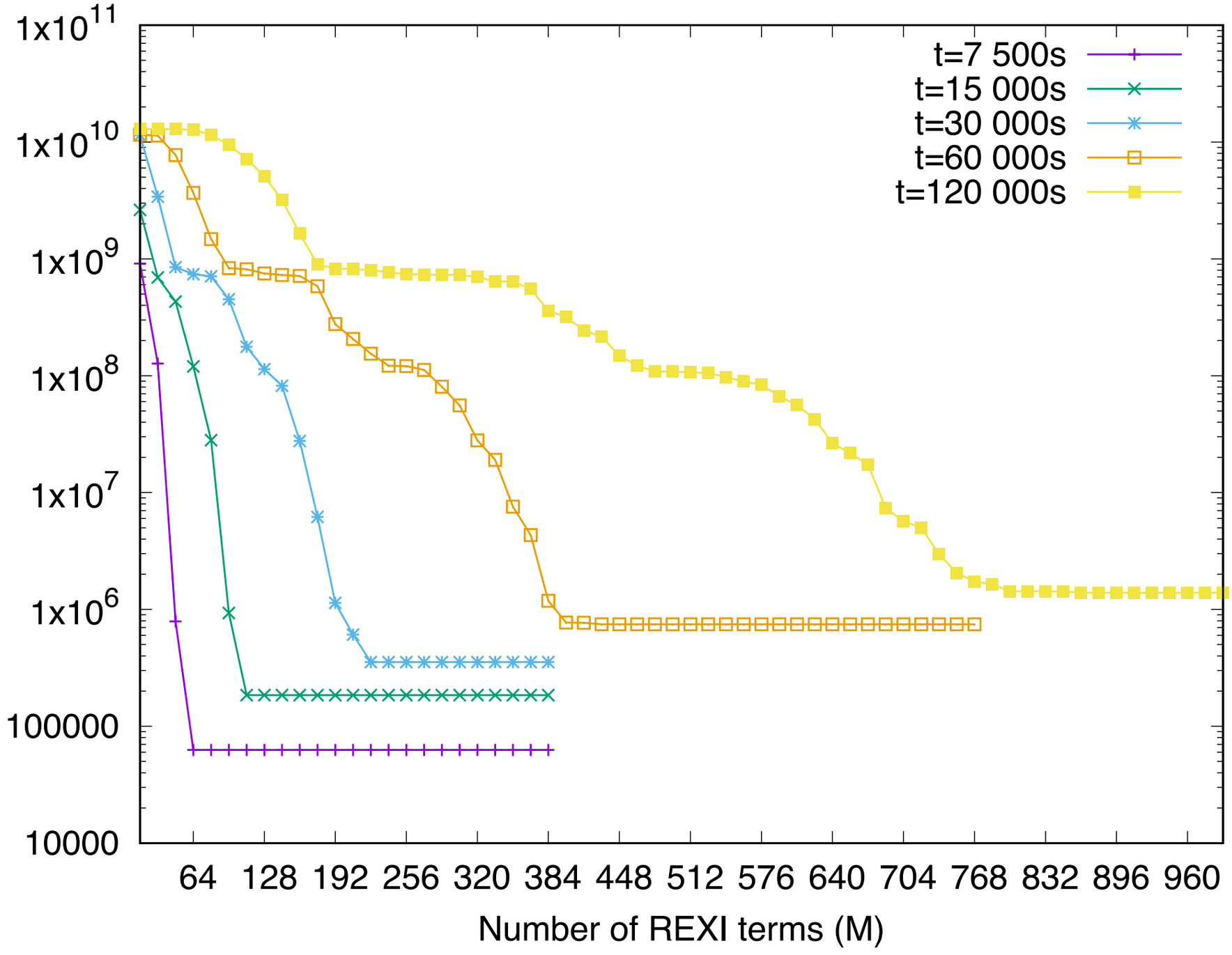
Schreiber et al, 2017, Beyond spatial scalability limitations with a massively parallel method for linear oscillatory problems, International Journal of High Performance Computing Applications

Convergence tests

- Initial conditions: polar wave
- Run REXI with varying number of terms (M) with h=0.2 (width of Gaussian)
- Check L2 error norm vs reference solution (implicit mid-point method with 25s time step)
- Increase REXI time step (t) and determine the number of terms (M) required to achieve convergence
- Expect: $hM > |t\lambda_{MAX}|$

$$hM > |t\lambda_{MAX}|$$

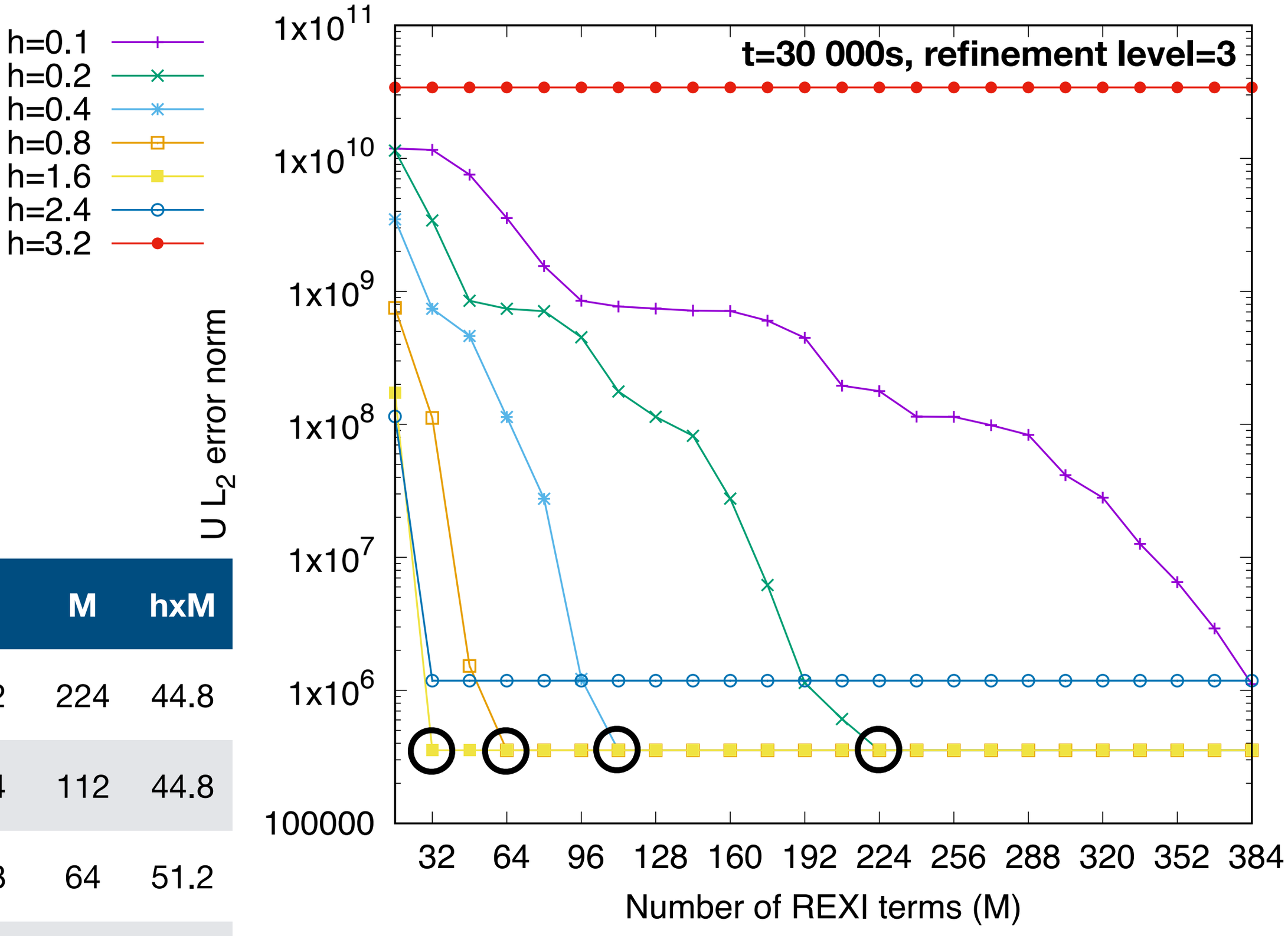
t/ks	M	λ_{MAX}
7.5	64	0.0017
15	112	0.0015
30	224	0.0015
60	432	0.0014
120	864	0.0016



Increasing t requires larger M (linear) ✅

Increasing t increases error ✅

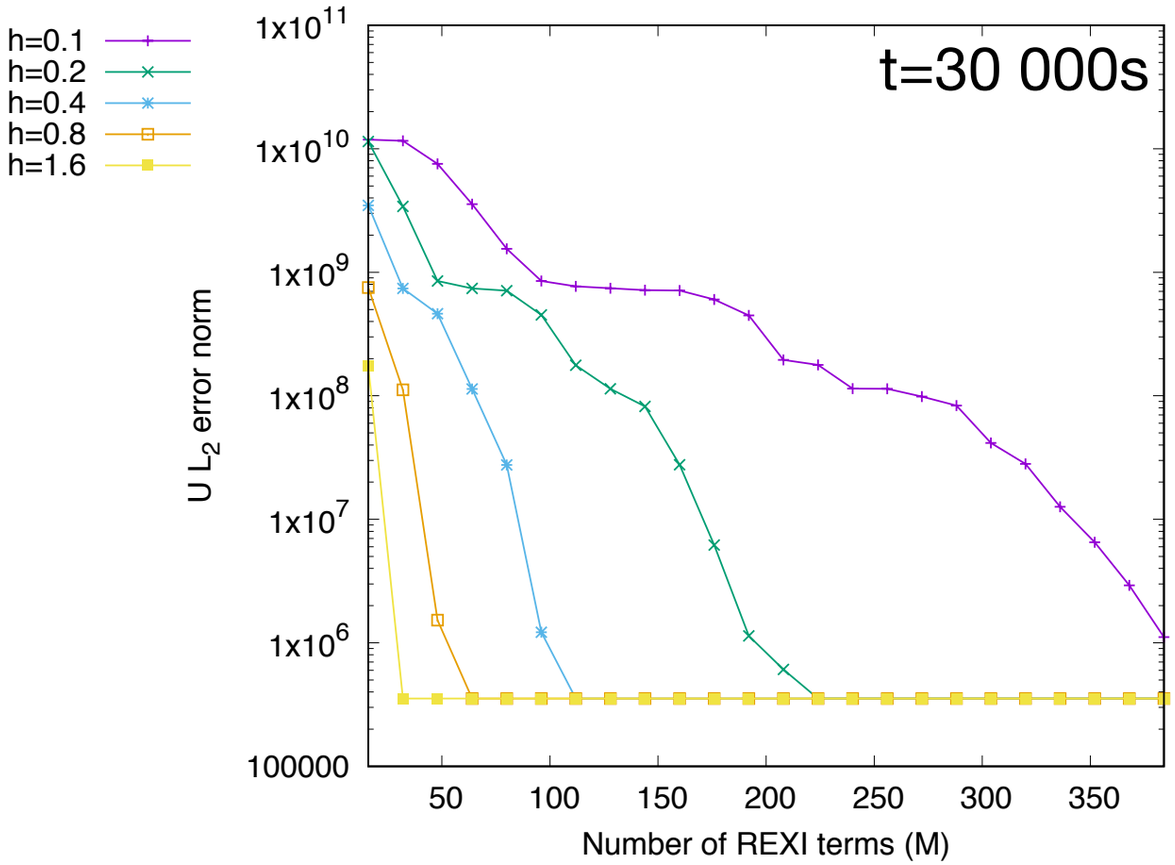
hM is constrained but what about h on its own?

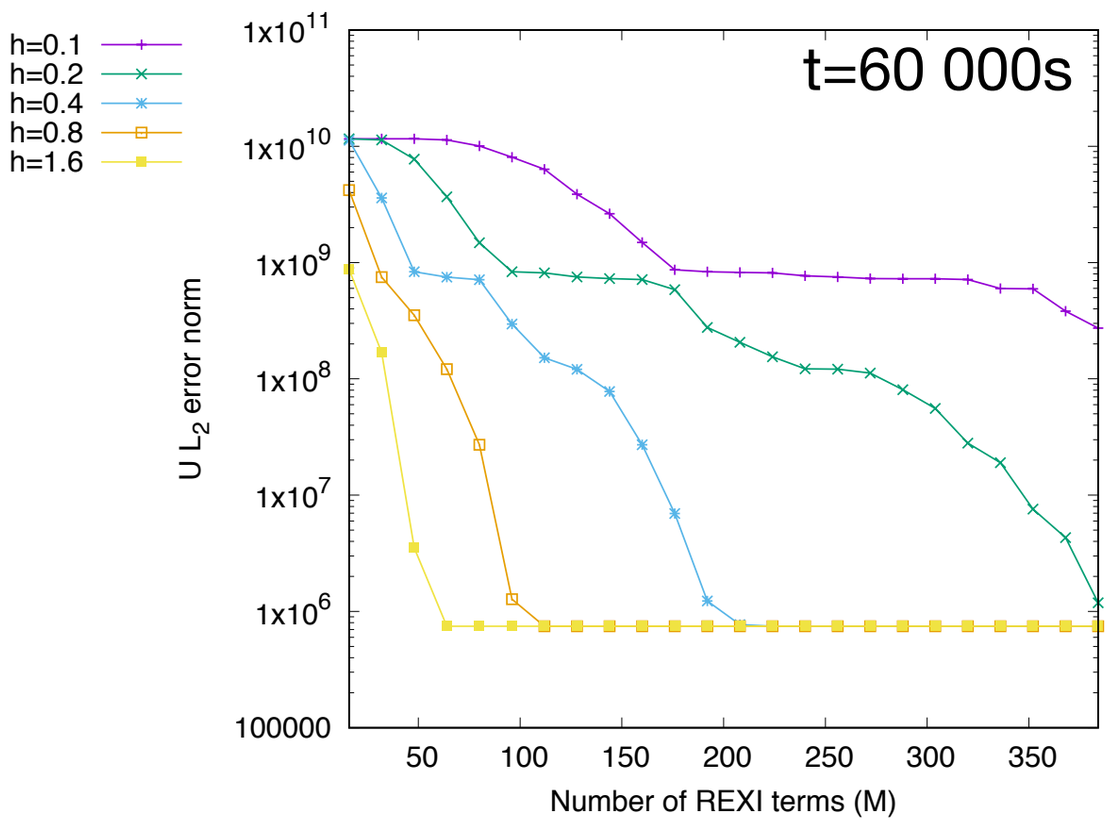


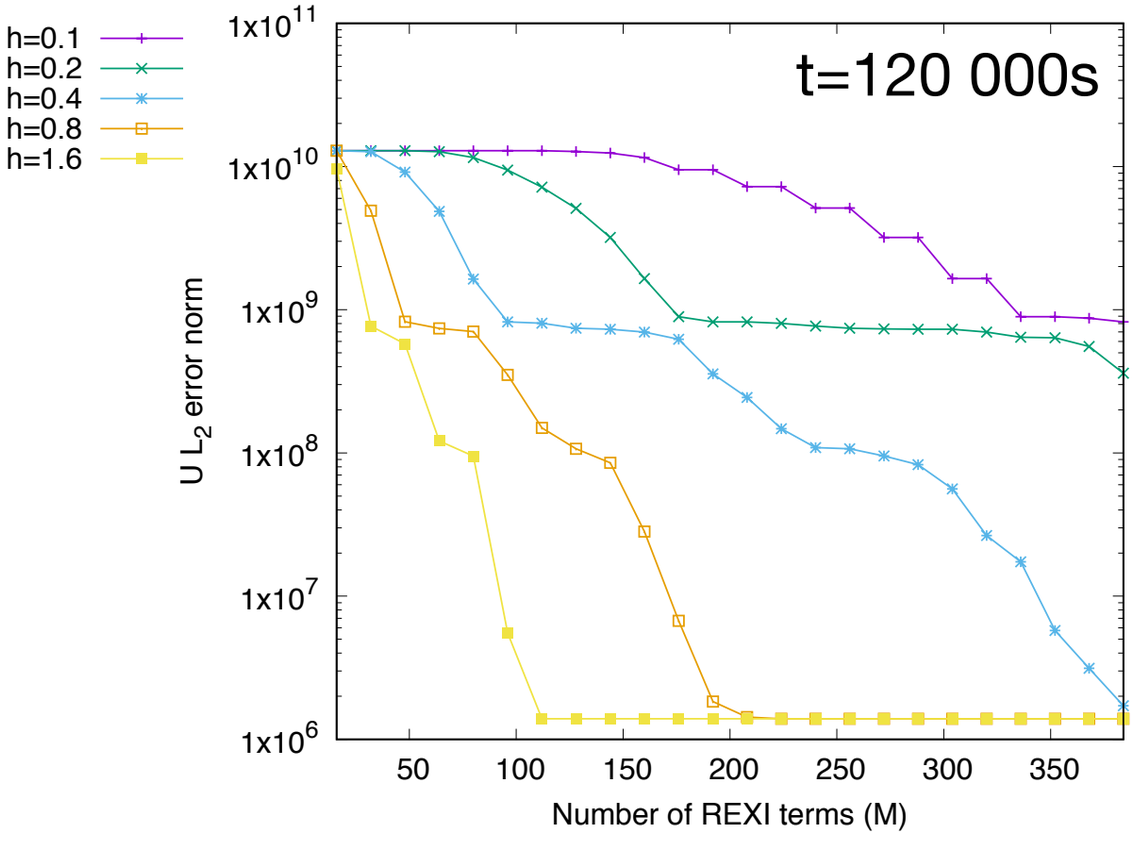
h	M	hxM
0.2	224	44.8
0.4	112	44.8
0.8	64	51.2
1.6	32	51.2

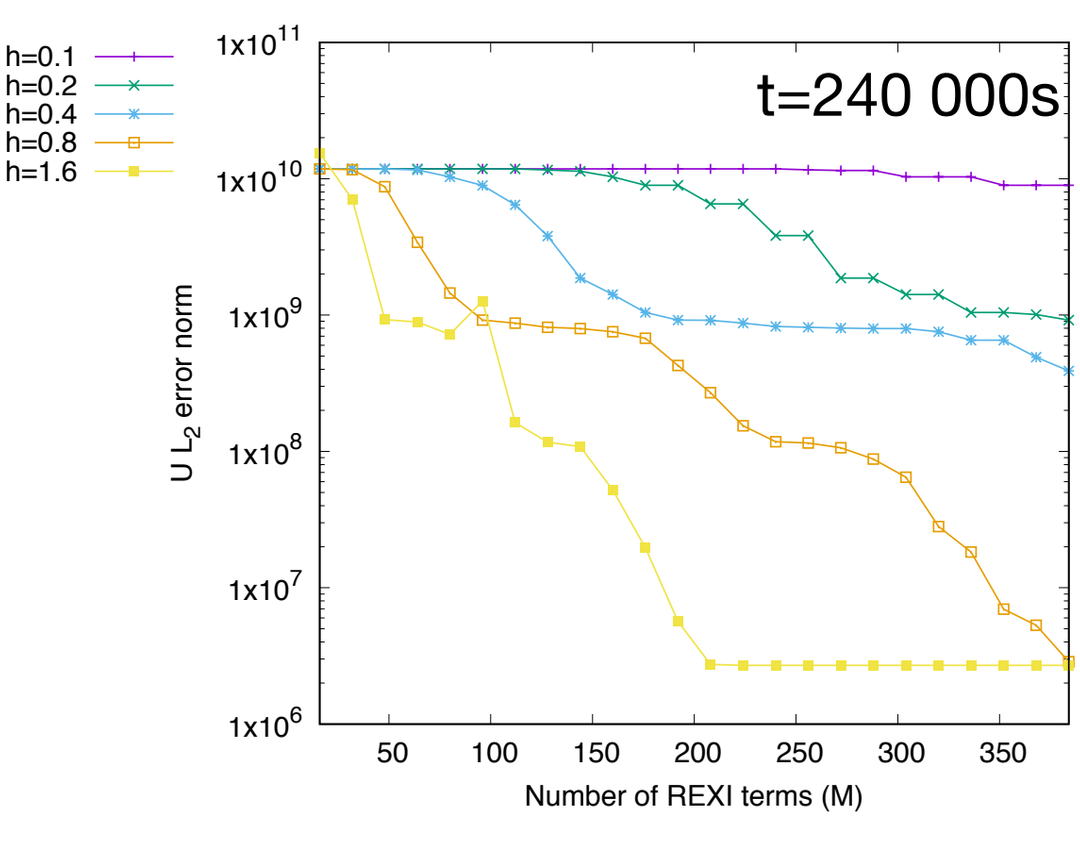
$$hM > |t\lambda_{MAX}| \approx 45 \Rightarrow \lambda_{MAX} \approx 0.0015$$

Can we use h=1.6 with a larger t?

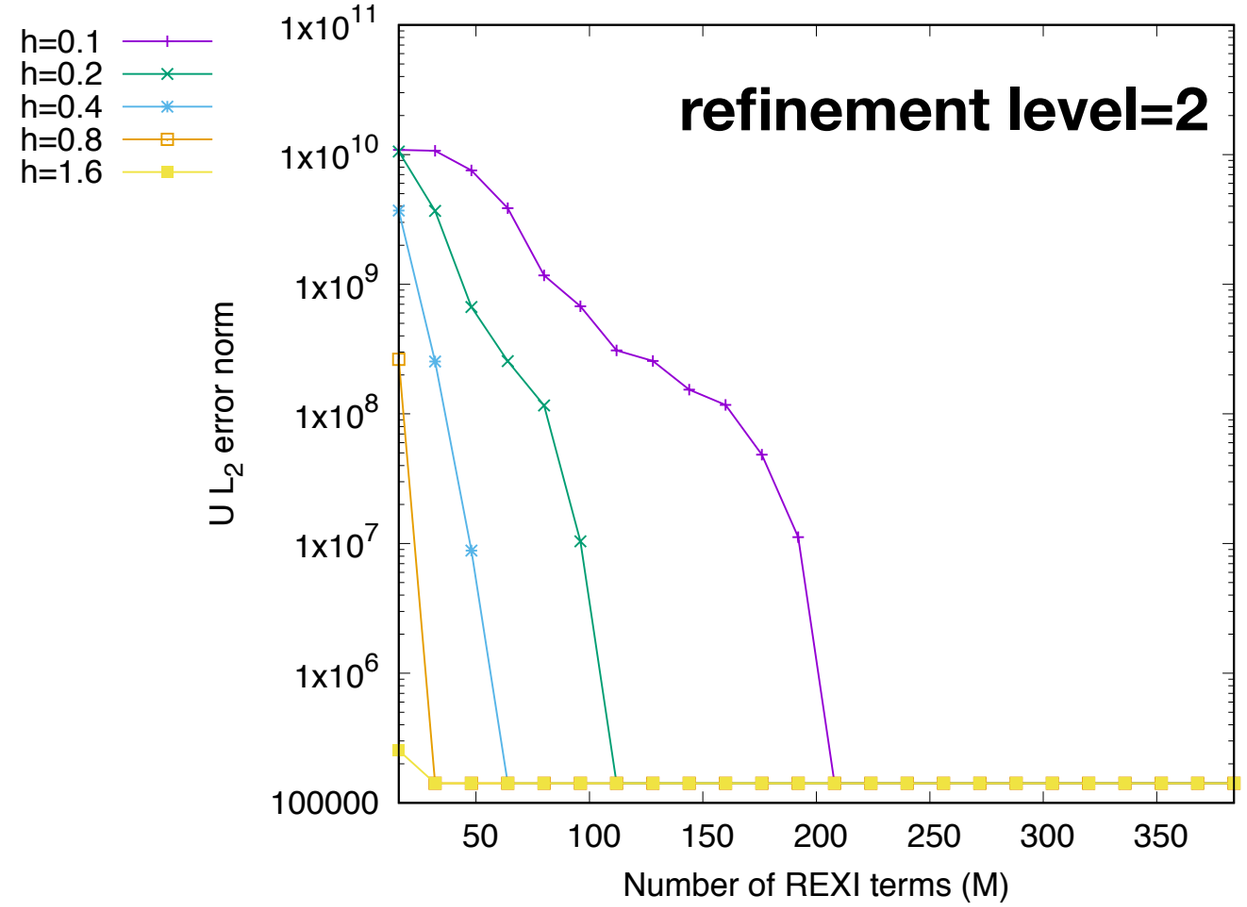


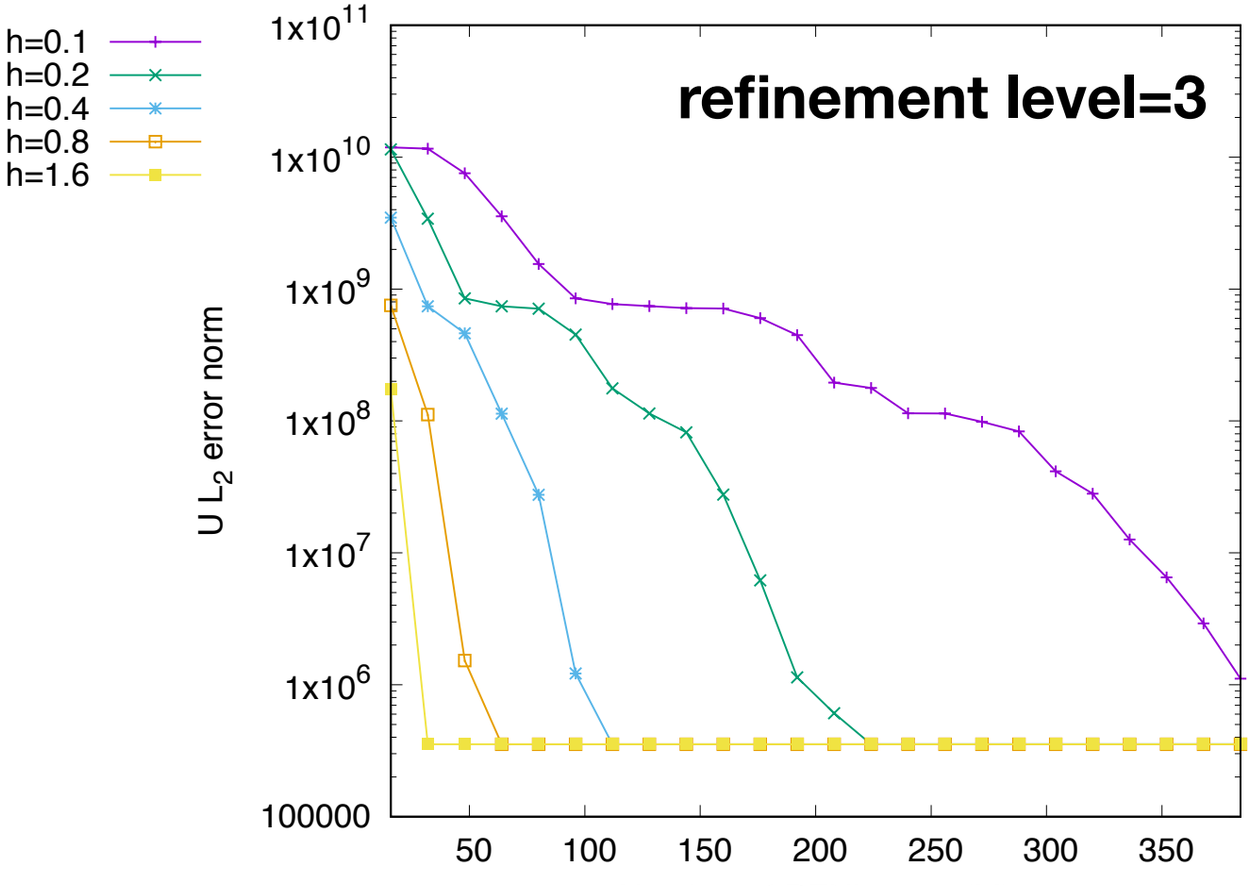


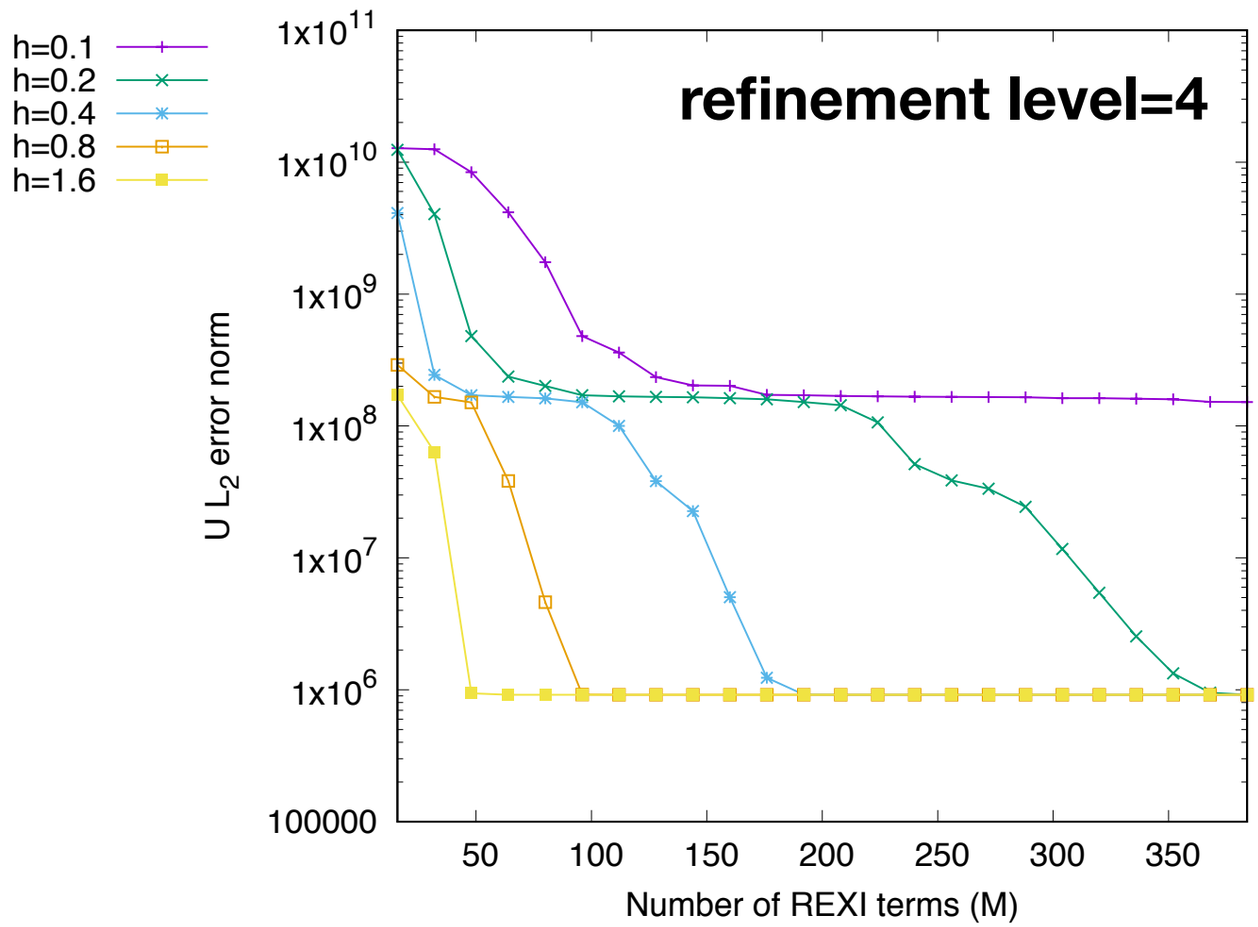


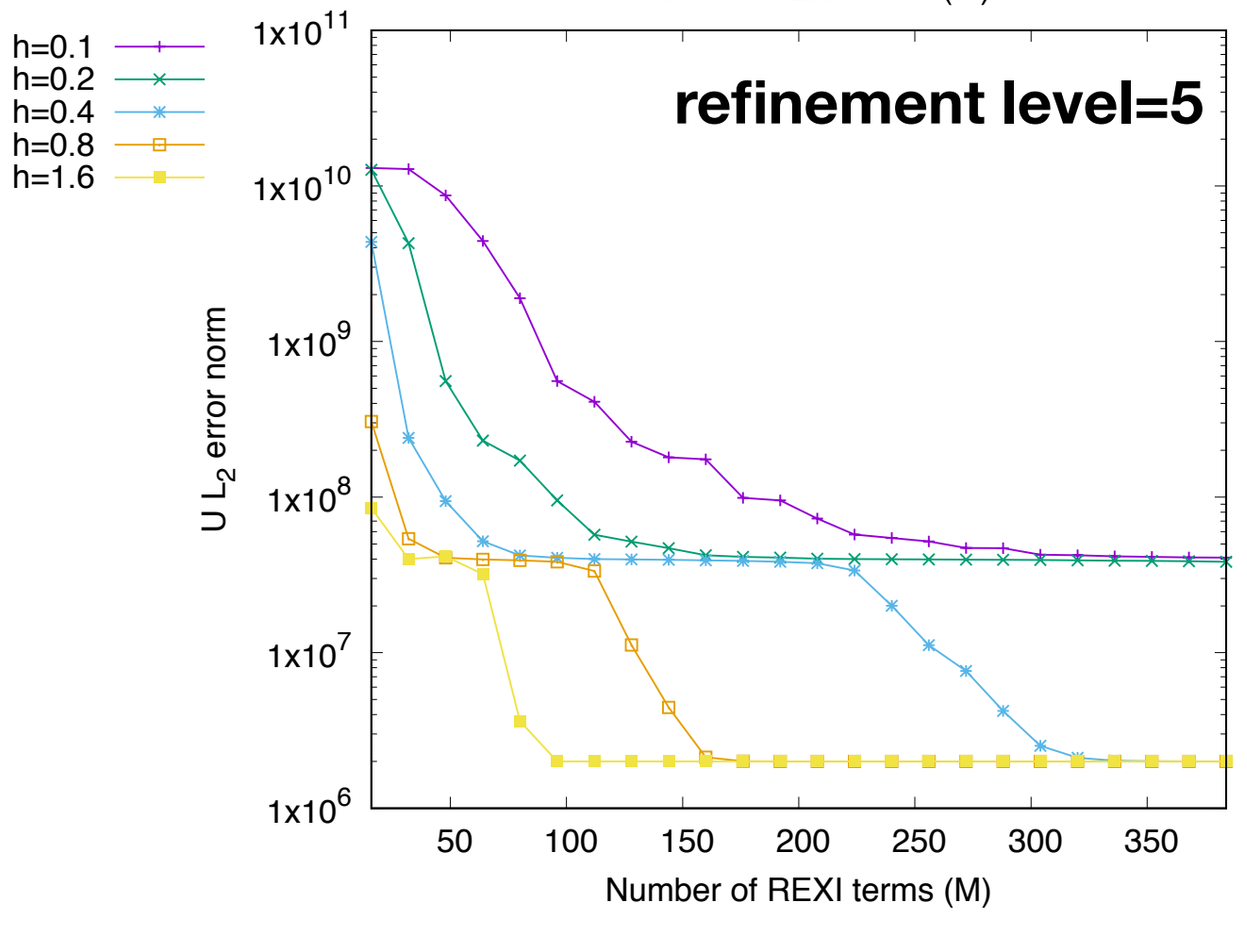


What about resolution (λ_{max})?





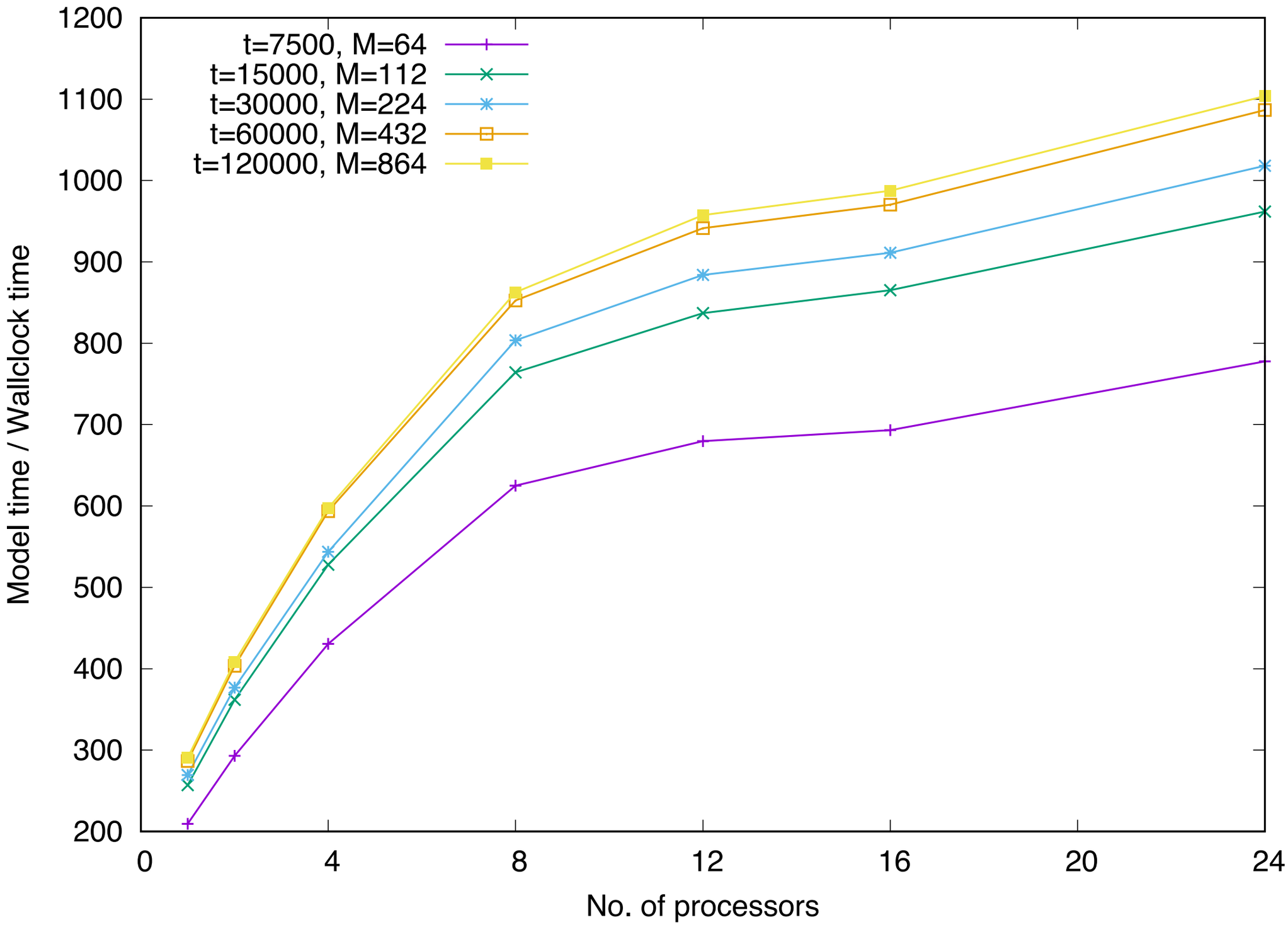




Scaling tests

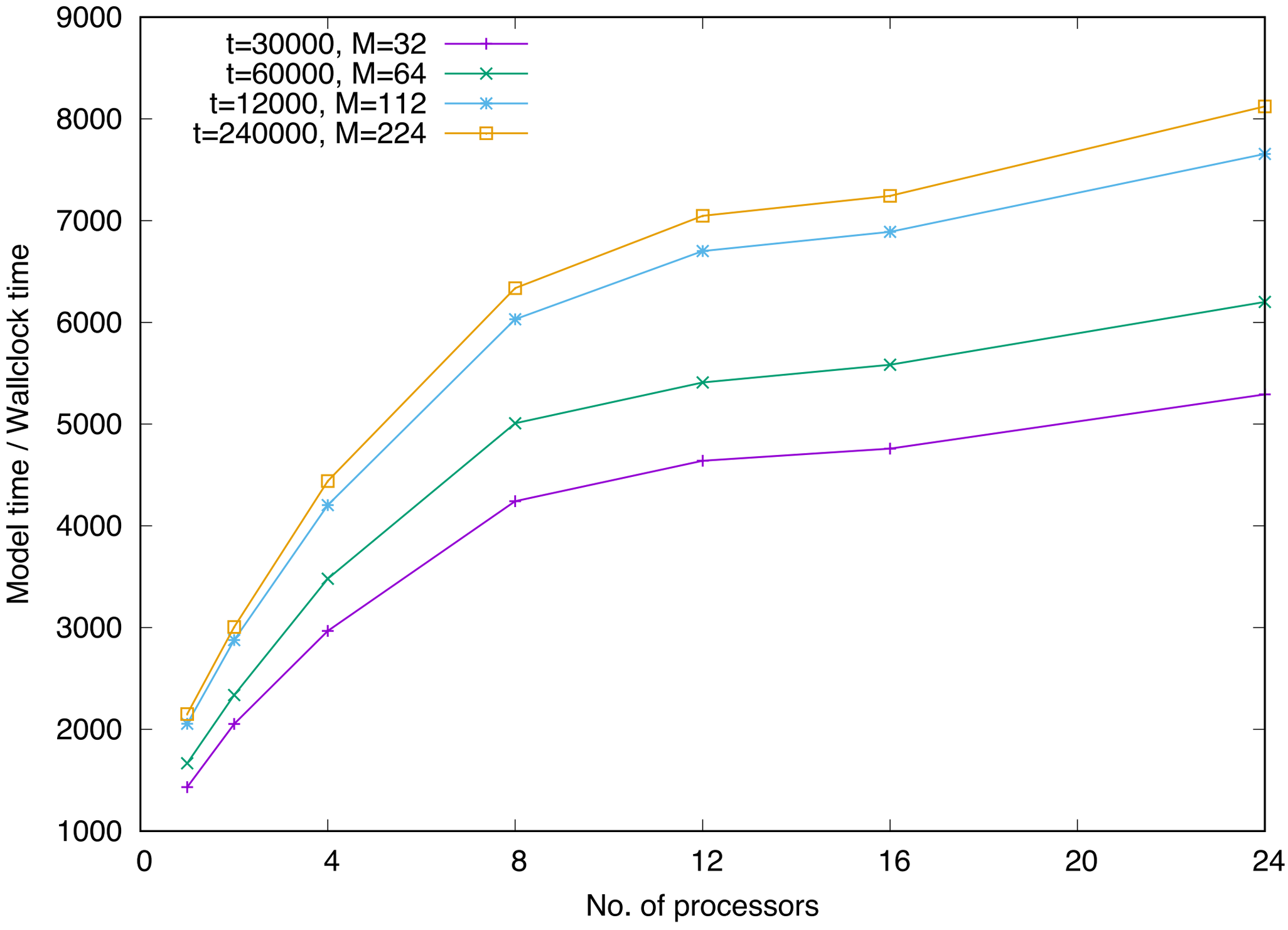
- Measure time for a single REXI step using PyOP2 timed stage (average over three runs, no I/O in timed region)
- h=0.2 and 1.6, minimum M for convergence, refinement level 3
- Single node scaling on Archer: 24 cores per node (2x12)
- Specify placement to ensure MPI processes are distributed evenly between sockets

h=0.2, refinement level=3



Reference solution: 115 (1 proc) → 1300 (24 procs)

h=1.6, refinement level=3



Reference solution: 115 (1 proc) → 1300 (24 procs)

Future work

- What value of h to use? Does this depend on the initial conditions (or other factors)?

- How to trade-off speed and accuracy?
  - For a given spatial resolution (affects λ_{MAX}) and t
  - Determine maximum h and minimum M for convergence ($hM > |t\lambda_{MAX}|$)
  - Measure error vs reference solution and time to solution

- Improve time to solution by reducing MPI overhead: examine in more detail with profiler (e.g. determine load balance)

Build with Intel toolchain and run DG advection example under MPI profiler: