Last time,

Strong and weak scaling.

- I had an error in my eq^n for efficiency of weak scaling.

→ Corrected on pdf & webpage.

This time

- Machine models
- Measurements → to find parameters
- Scaling limits (Fischer 2015)

→ Data structure design for scalability.

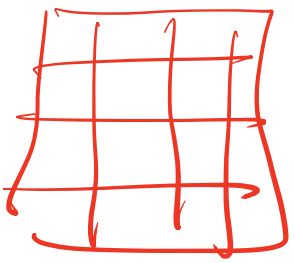
Translating our abstract scaling onto hardware.

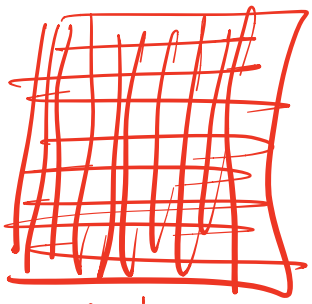
⇒ PDE solvers.

What are the scaling limits?

PDE I want to solve that has
N total degrees of freedom
(dofs)
↳ parameter that we can control by changing resolution.

I have a parallel computer available.
I can pick # processes P as big as I want.


N small


N big

If I can divide work (N) by P, at what pt should I stop?
(When does the efficiency drop below 80% ? 50%).

How small can I make $\frac{N}{P}$ without wasting too much resource?

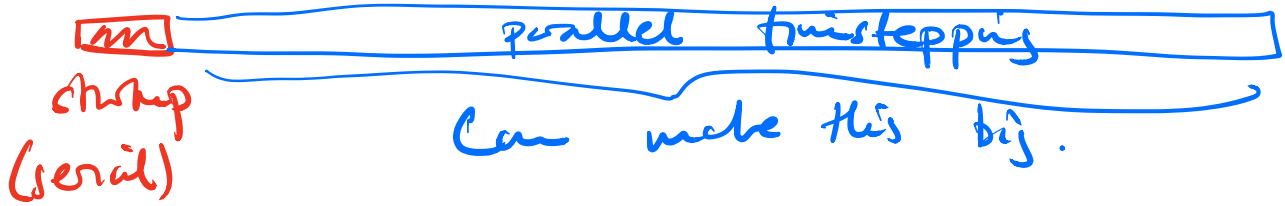
Strong scaling problem:
N fixed & P varies.

Amdahl model: $T_p = f T_1 + (1-f)\frac{T_1}{P}$

Where does this serial fraction, f, come from?

1. Unavoidably serial parts of our code. eg loading configuration, or printing summary data.

   ⟹ Typically can amortise this because for time-dep PDEs only need load config once



2. Communication takes time.
   → eg two processes have to exchange information every timestep.

say they need to exchange
m bytes of data.

$$t(m) = \alpha + \beta m$$

α → [s]

βm → [bytes]

β → [s/byte].

non-dimensionalise

relevant timescale is how long it takes to do some computational work.

→ problem-specific

eg FD model, this time might be "how long to update 1 point in the grid".

$u_{ij} \leftarrow -4u_{ij} + (u_{i+1,j} + u_{i-1,j} + u_{i,j+1} + u_{i,j-1})$

Call this t_f

→ measure our code. → and find eg # updates/s.

$$t^{nd}(m) = \frac{\alpha}{t_f} + \frac{\beta}{t_f} m$$

→ hope hardware specs. ⇒ unfortunately not.

how long to send a message vs.

how long to do some compute?

α, β: need to know this for the parallel programming library we're using.

$\Longrightarrow$ measure.

ping-pong measurements.

send message back and forth.

This should give nice clean data?

→ sometimes.

Let's and measure this.

Write some ping pong code

→ when uploaded

have a go doing these measurements for with & without numpy.