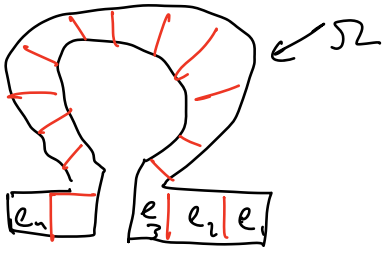
$$-\nabla^2 u = f \qquad u \in \mathbb{R}^n$$

On a computer?

Need to solve in finite dimensional subspace

Want to solve on domain $\Omega$:

subject to some boundary conditions.




$\Rightarrow$ Finite dim subspace of infinite dimensional continuous solution space.

Find $u_h \in V_h$ s.t.

$$\int \nabla u_h \cdot \nabla v_h \, dx = \int f v_h \, dx$$

$\forall v_h \in V_h$.

On a computer: numerical quadrature.

$$\sum_{e_i} \underbrace{\int \nabla u_h|_{e_i} \cdot \nabla v_h|_{e_i} \, dx_i}$$

$$\sum w_q \phi_{qj} \phi_{qi}$$

Eventually: we end up with a big sparse matrix equation

known ↓

$$\begin{bmatrix} & & \\ & A_h & \\ & & \end{bmatrix} \begin{bmatrix} \\ u_h \\ \\ \end{bmatrix} = \begin{bmatrix} \\ f_h \\ \\ \end{bmatrix}$$

↑ known ↑ unknown

Can find $u_h = A_h^{-1} f_h$.

Use an iterative method that only needs

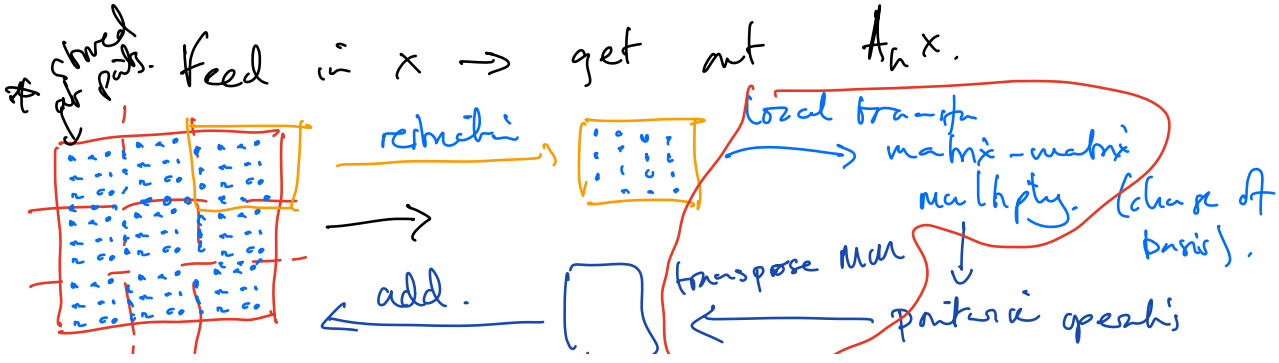
$A_h x$ for known vectors $x$.

Computational challenge boils down to:

- How to compute $A_h x$ fast?

Lib CEED (benchmark code).

Feed in $x$ → get out $A_h x$.



computationally expensive part.

What libCEED offers is
a number of different algorithms for
applying these M-M multiplies.

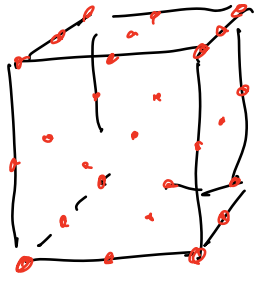
→ what options are there?
How to vectorize?
→ Dimension lifted transposition approach.
→ "blocked".
Grabs entries from 4 elements at once,
→ vectorizes across elements.

→ Backends that call specialized M-M
libraries. ⇒ optimal.

---

What parameters can we control?

Domain is divided up into some # of elements.



Can control polynomial / approximate
order of solution vector.
⇒ How many values live
in each element.

p = 2 ↑ 1 element.

degree p has $(p+1)^d$ values
in d-dimensions.

p>2 : M-M multiply has dimension 27×27.

→ Exploit structure to do these multiplications

in $O(p^{d+1})$ flops. on each element
[sum factorisation].

⇒ Code does $O(p)$ flops per byte of data moved.

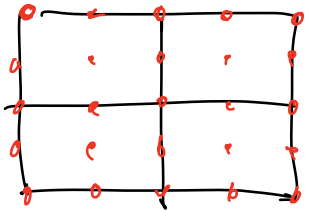
$p=1$ ⇒ low arithmetic intensity

$p=16$ ⇒ high " "

How much data moves?

Need to load $\times$ $(p+1)^d$ / element.
and store result $(p+1)^d$ / element.

There's some reuse between elements.



Nested and perfect cache models are not far apart.

Benchmarks show output: in DOFs/sec.
"useful throughput".

Gotchas: compile with appropriate optimisation flags ⇒ check docs.

: picking problem sizes
→ when $p$ is large make sure you have enough elements.

eg $q = 19 \Rightarrow$ 8000 dofs / element.

$\rightarrow$ 64kB (because each dof is 8 bytes).

$\Rightarrow$ make sure global problem size is big enough for more than one element.

---

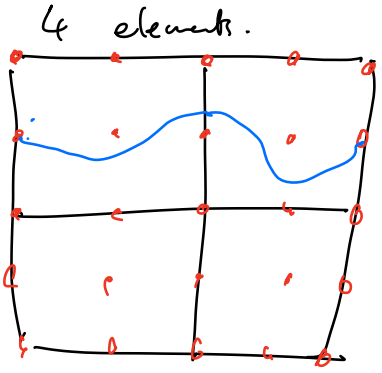
Concrete example:

Want to:

find $u_h \in V_h$ s.t.

$$\int u_h v_h dx = \int f v_h dx \quad \forall v_h \in V_h.$$

$V_h$ = set of all bicubic functions on each element.

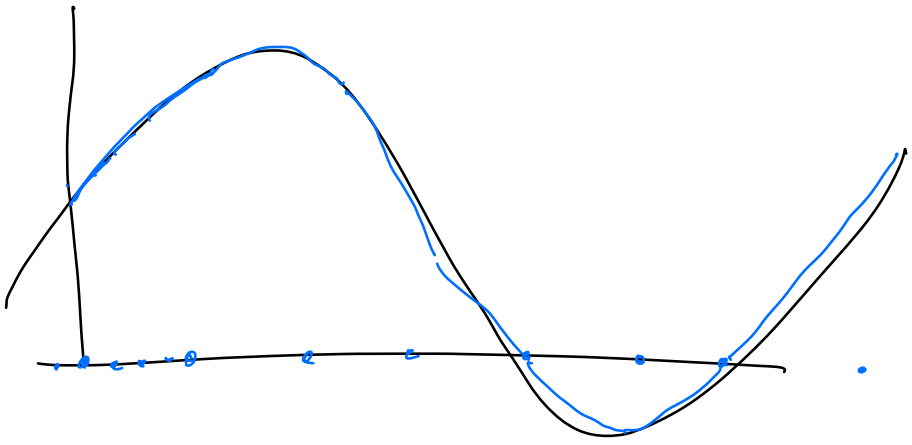



Basis on each element is

$$\{1, x, x^2, y, y^2, xy, xy^2, x^2y, x^2y^2\}$$

$X$ is a vector with 25 entries, each of which is a coefficient for one basis vector

↓ "degrees of freedom".



Computation proceeds

gather from global vector to per-element vector. } data movement

compute on that element ⟵ compute.

scatter result into global vector. } data movement