A very short introduction to pyop3

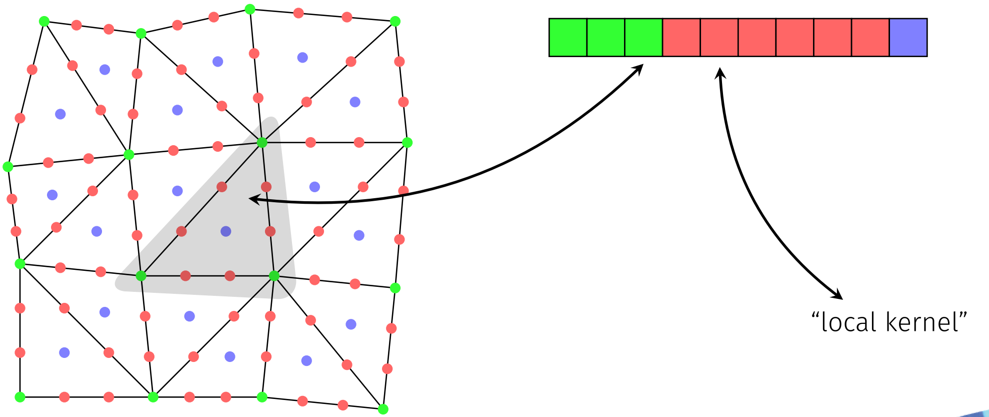
Connor Ward

24 March 2023

What is pyop3?

- Domain-specific language for writing stencil computations
- Embedded in Python
- Intended to be the successor to PyOP2
- **Work in progress**

What is a stencil computation?

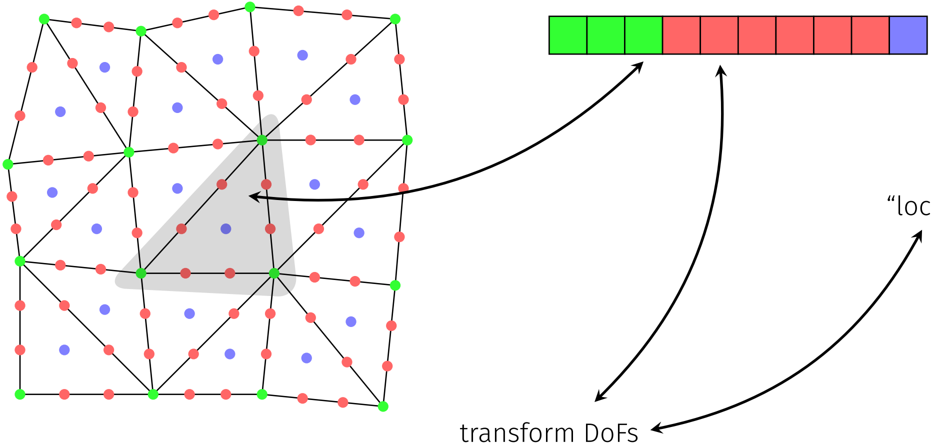


pyop3 interface example: FEM assembly

```
loop(
  c := mesh.cells.index(),
  kernel(dat1[closure(c)], dat2[closure(c)])  # READ, INC
)
```

- This loop expression gets compiled to fast C/OpenCL/CUDA code
- Loops can be nested
- Loops can execute more than one statement
- Maps can be composed (e.g. `closure(star(v))`)
- Works with structured, unstructured and partially structured (e.g. extruded) meshes

“Extraction operators”



“local kernel”

transform DoFs

```
loop(c := mesh.cells.index(),
  [
    t0 := alloc(dat1[closure(c)]),
    t1 := alloc(dat2[closure(c)]),
  ],
  [
    read(dat1[closure(c)], t0),
    zero(t1),
    kernel(t0, t1),
    inc(t1, dat2[closure(c)]),
  ]
)
```

Orienting edges

```
loop(c := mesh.cells.index(), [t0 := ..., t1 := ...],
  [
    read(dat1[closure(c)], t0),
    zero(t1),
    loop(e := t0.edges.index(), maybeflip(t0[e], o[c, e])),
    kernel(t0, t1),
    loop(e := t1.edges.index(), maybeflip(t1[e], o[c, e])),
    inc(t1, dat2[closure(c)]),
  ]
)
```

- `maybeflip` is just another (loopy) kernel
- `o` is an array storing orientations for each cell closure
- `t0` and `t1` “know” which DoFs are edge DoFs

This approach should generalise to other types of constraints that we want in our stencil computations:

- h and p adaptivity
- “Zany” elements
- Non-slip boundary conditions