PSyclone LFRic distributed memory support

Rupert Ford, Andy Porter, Sergi Siso, STFC Hartree Centre
Iva Kavcic, Chris Maynard, Andrew Coughtrie, UK Met Office
Joerg Henrichs, Australian Bureau of Meteorology

ESIWACE2 training course on Domain-specific Languages in Weather and Climate, 23rd-27th November 2020

Overview

- 90 minute session
- Hands on part 1 : 15 minutes
  - Going parallel
- Introduction to distributed memory : 30 minutes
- Hands on part 2 : 45 minutes
  - 3 parts
    - Annexed dofs
    - Asynchronous comms
    - Reductions
- Any issues/questions on the slack channel

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Hands-on : Let's go parallel

- 15 minutes
- `cd <psyclone_home>/tutorial/practicals/LFRic/distributed_memory/1_distributed_memory`
- No compilation, just code generation
- Follow the `README.md` in the directory
  - A browser will display `README.md` files nicely
  - `https://github.com/stfc/PSyclone/tree/master/tutorial/practicals/LFRic/distributed_memory/1_distributed_memory`
- Any issues/questions on slack
  - Use the psyclone channel
  - Please use threads for replies

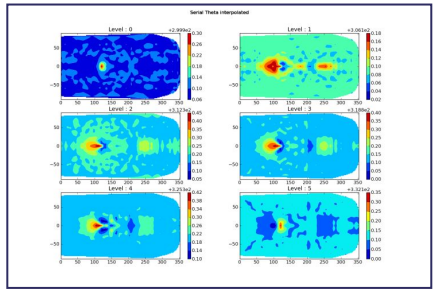
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

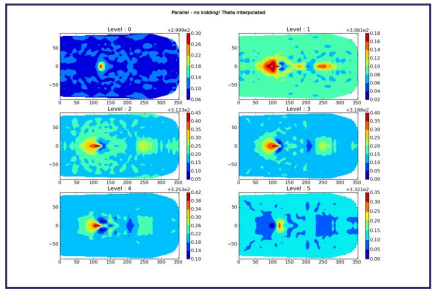
Going parallel: Summary

- Example code extracted from LFRic – most computationally costly part of the dynamical core
- The same algorithm and kernel code written by the scientist is used to run serially or in parallel
  - Single-source science code
  - Science code is not concerned with parallel implementation
- For a user, generating serial or distributed memory parallel code is controlled by a single PSyclone command-line argument

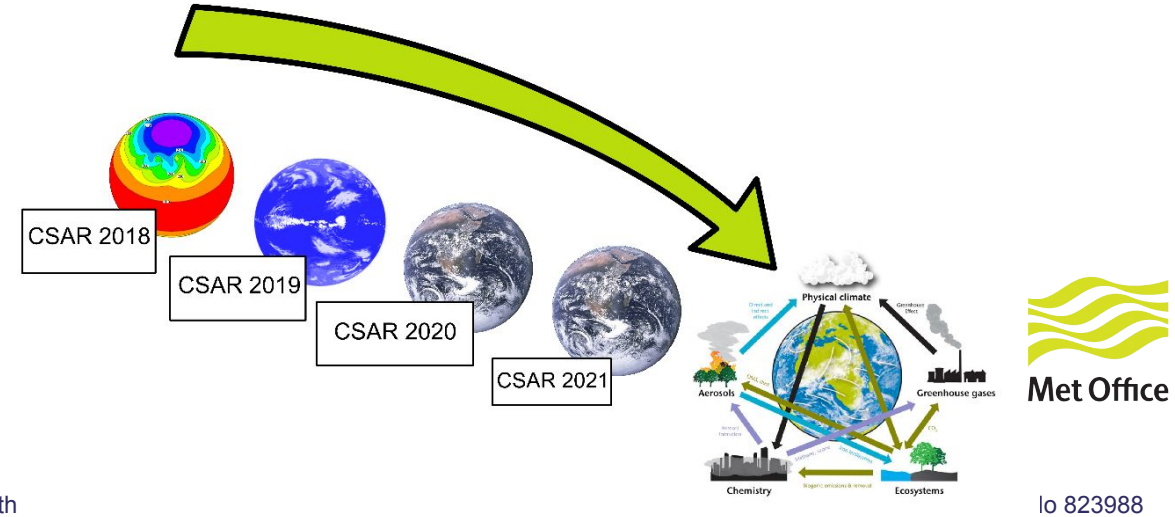
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Going parallel: In practice



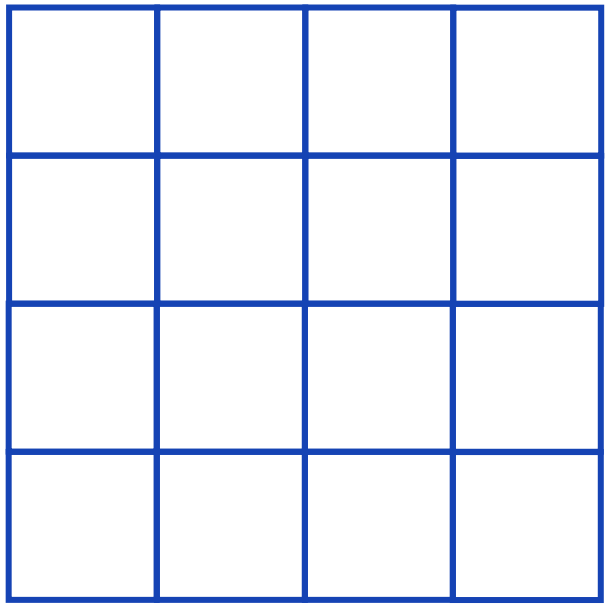


- PSyclone integrated into LFRic build system in September 2015 - serial
- LFRic went parallel (MPI + OpenMP) in March 2016
  - Switch was essentially immediate (but took 1 week in practice due to simple PSyclone OpenMP bug for reductions)
  - No change to science code from serial to parallel
- Science development has continued since then (including adding Physics)



ESiWACE2 has received funding from th lo 823988

Cells/Elements

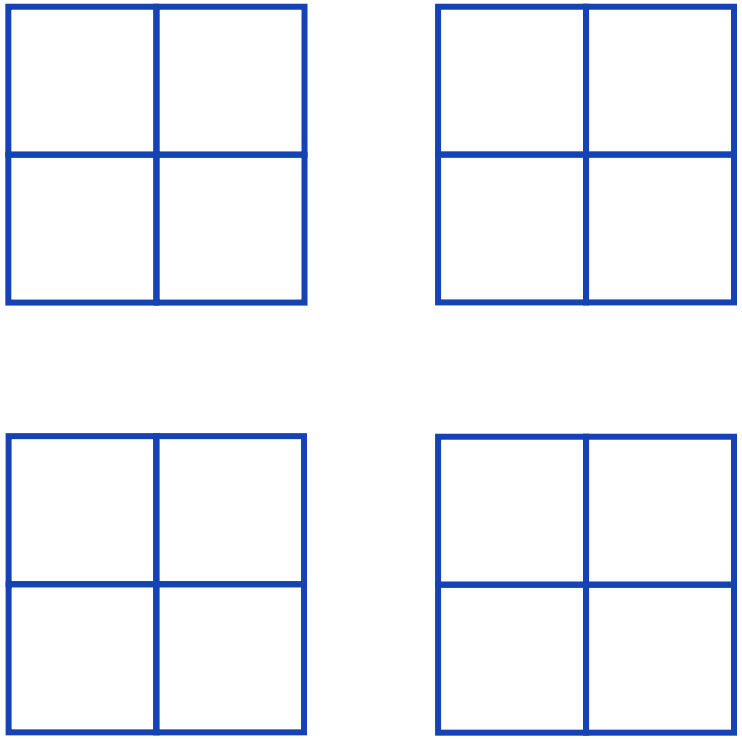


Met Office

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

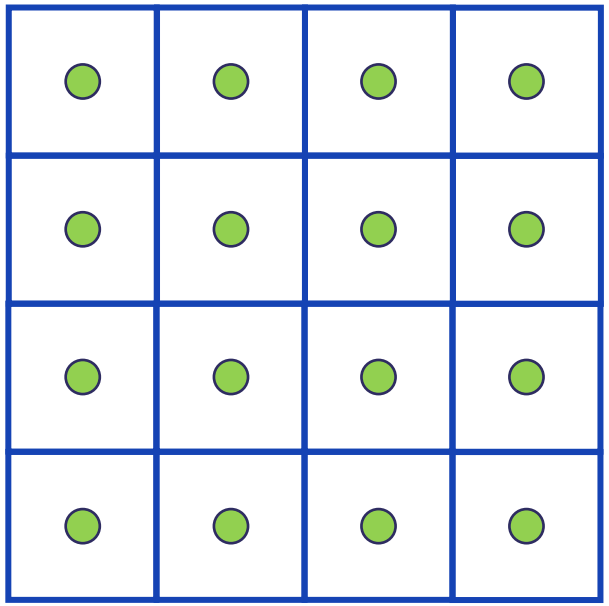
esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Partitioned Cells/Elements



ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

DOFs on discontinuous function space

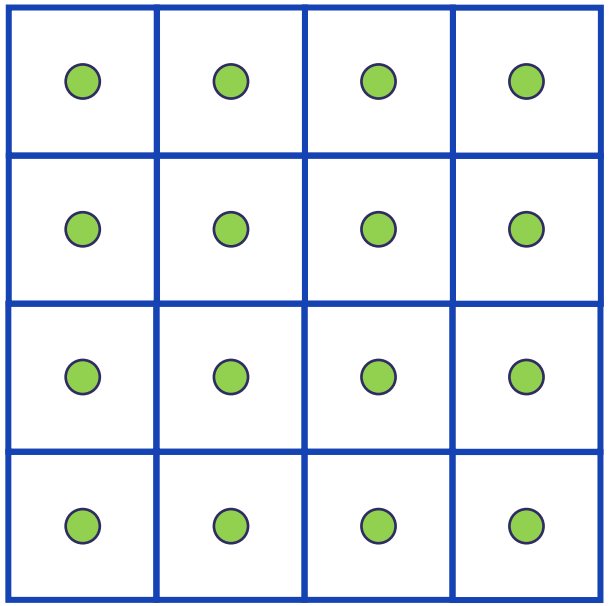


ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Iterating over discontinuous DOFs

Iterate over dofs

```
DO i=1,ndofs
    ! builtin code
```



Accesses

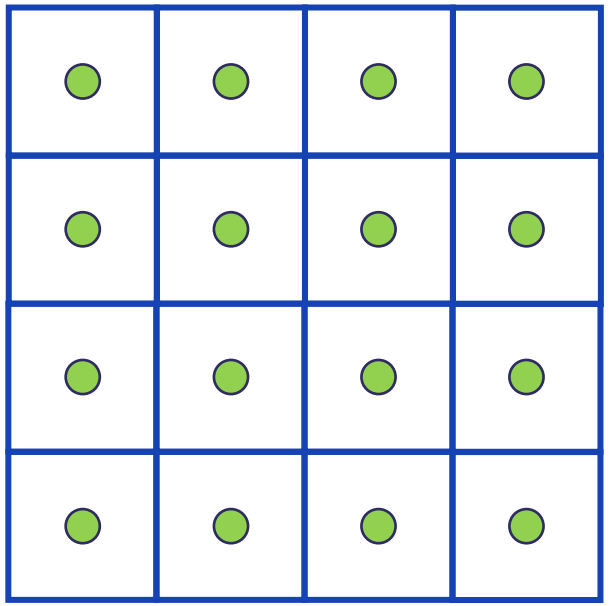
```
read
write
readwrite
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Iterating over cells with discontinuous DOFs

Iterate over cells

```
DO i=1,ncells
    call kern(…)
```



Accesses

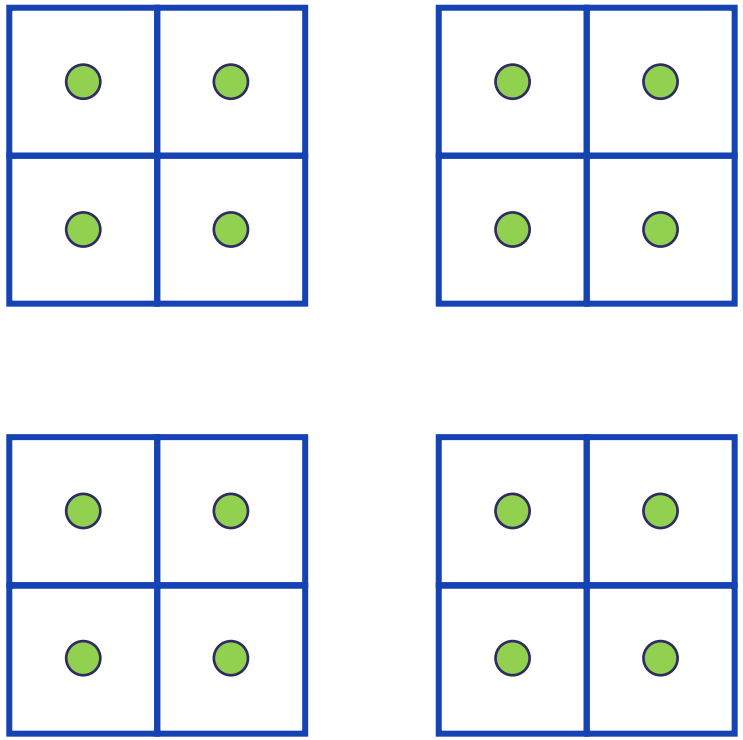
```
read
write
readwrite
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Partitioned discontinuous DOFs

Iterate over dofs

```
DO i=1,ndofs
    ! builtin code
```



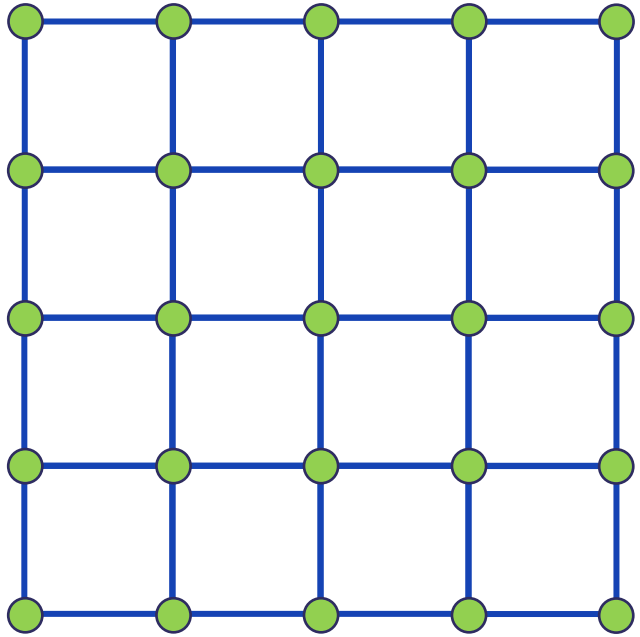
Iterate over cells

```
DO i=1,ncells
    call kern(…)
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

DOFs on a continuous function space

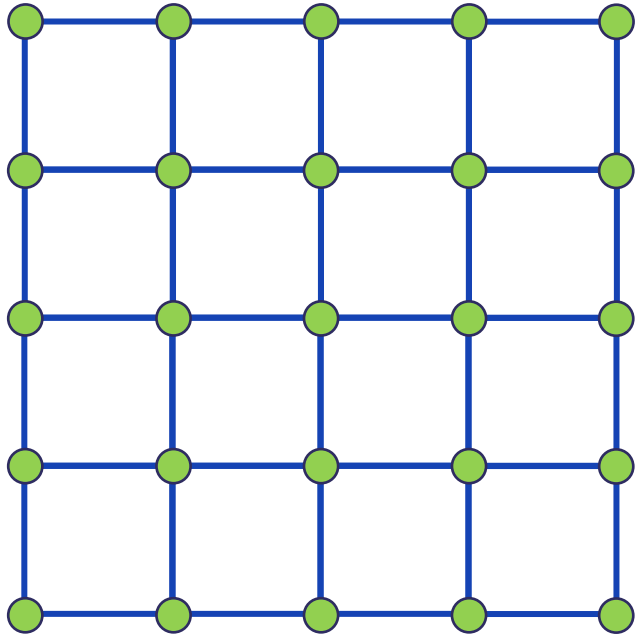


ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Iterating over continuous DOFs

Iterate over dofs

```
DO i=1,ndofs
    ! builtin code
```



Accesses

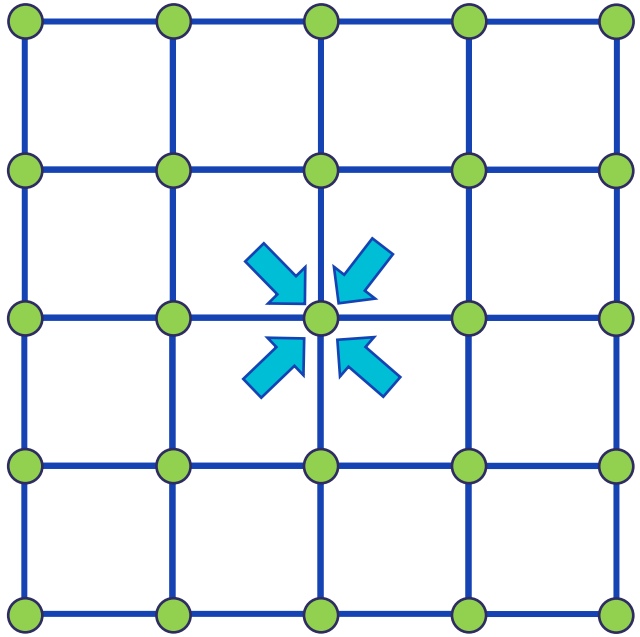
```
read
write
readwrite
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Iterating over cells with continuous DOFs

Iterate over cells

```
DO i=1,ncells
    call kern(…)
```



Accesses

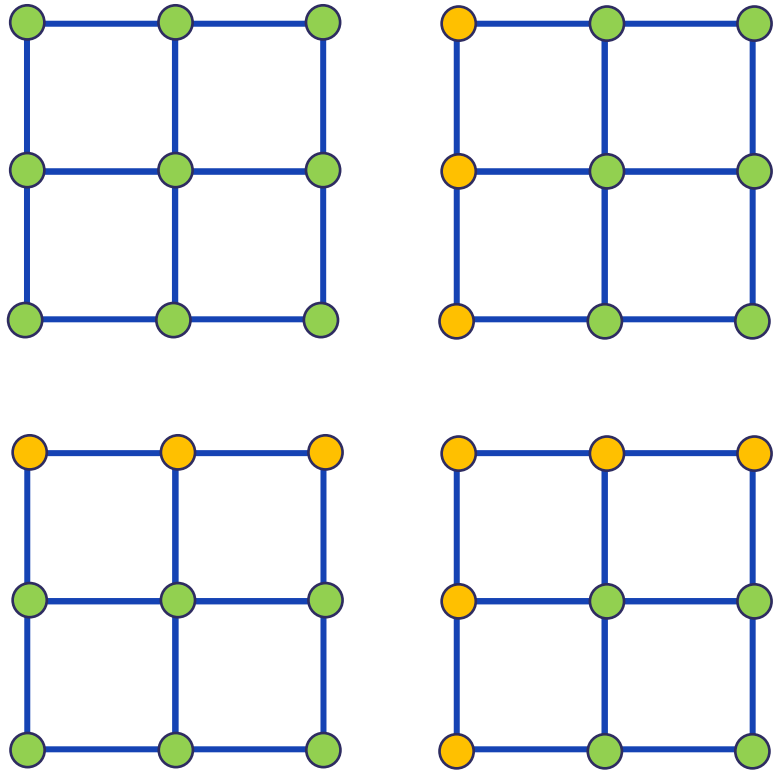
```
read
inc
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

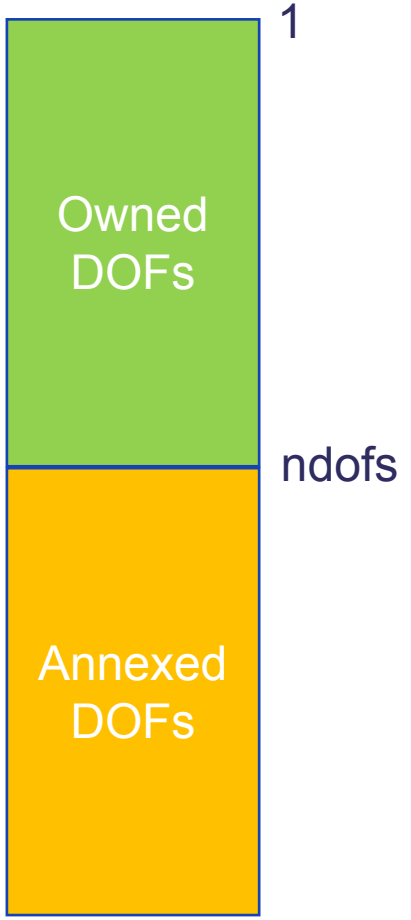
Partitioned continuous DOFs

Iterate over dofs

```
DO i=1,ndofs
    ! builtin code
```



DOFs Numbering scheme



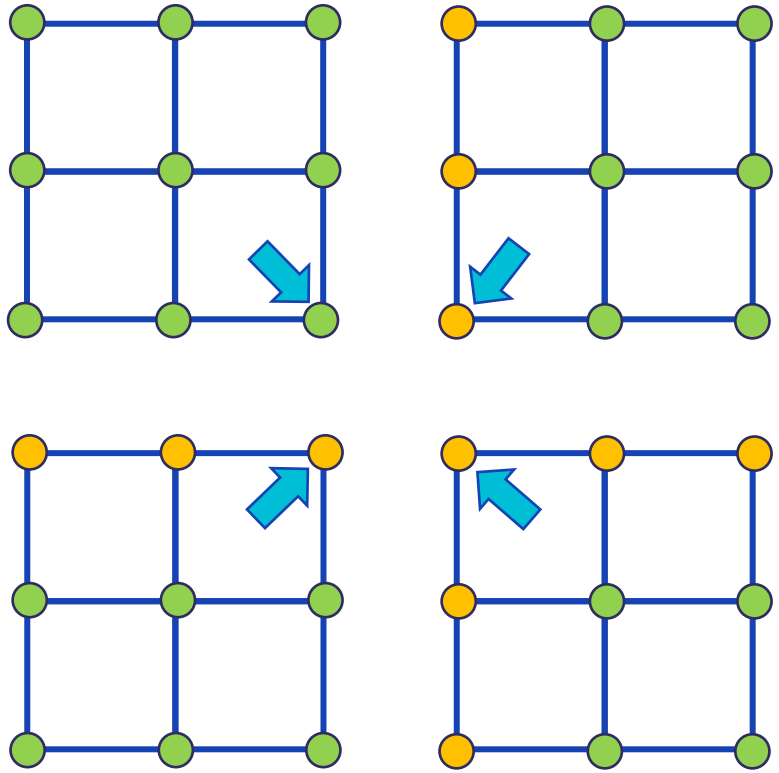
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Partial sums

Iterate over cells

```
DO i=1,ncells
   call kern(…)
```

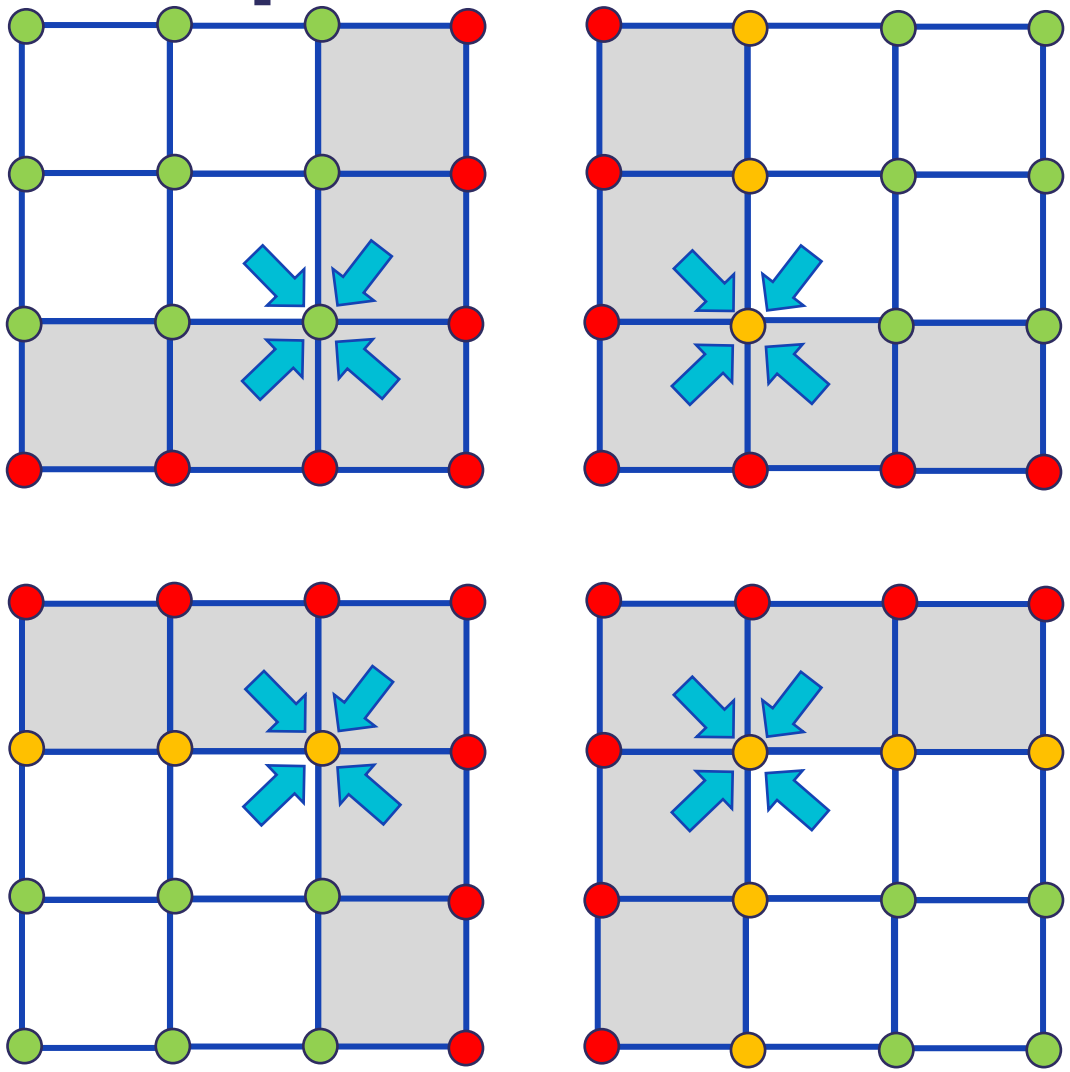


ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

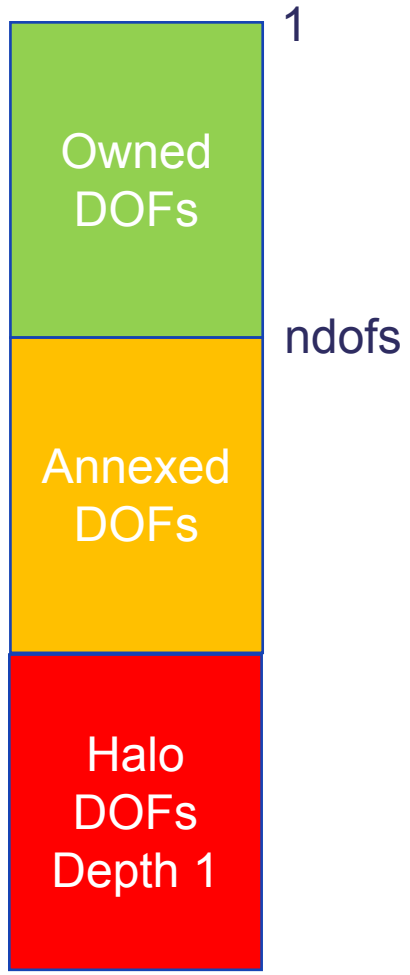
Redundant computation

Iterate over cells

```
DO i=1,cell_halo(1)
    call kern()
```



DOFs Numbering scheme



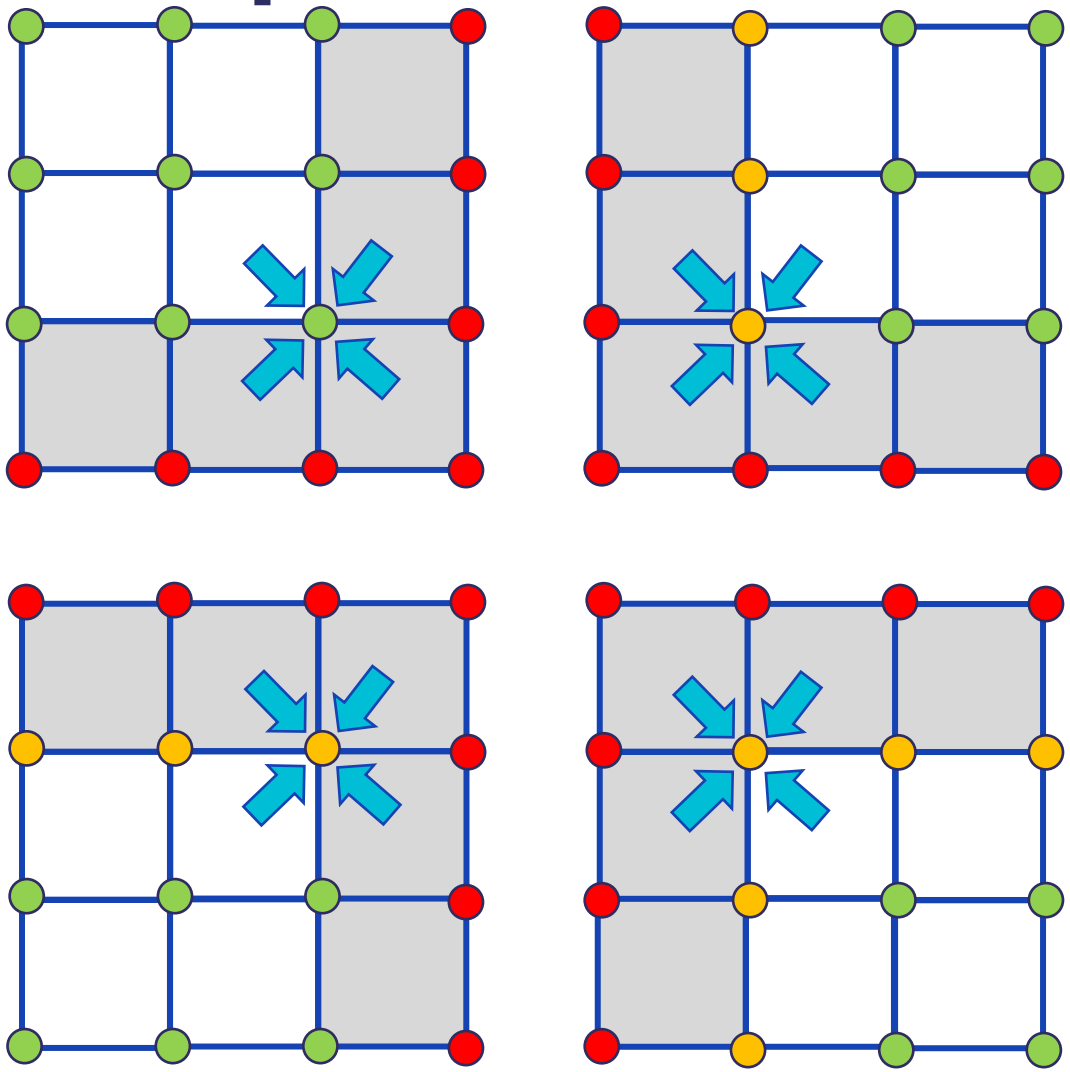
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

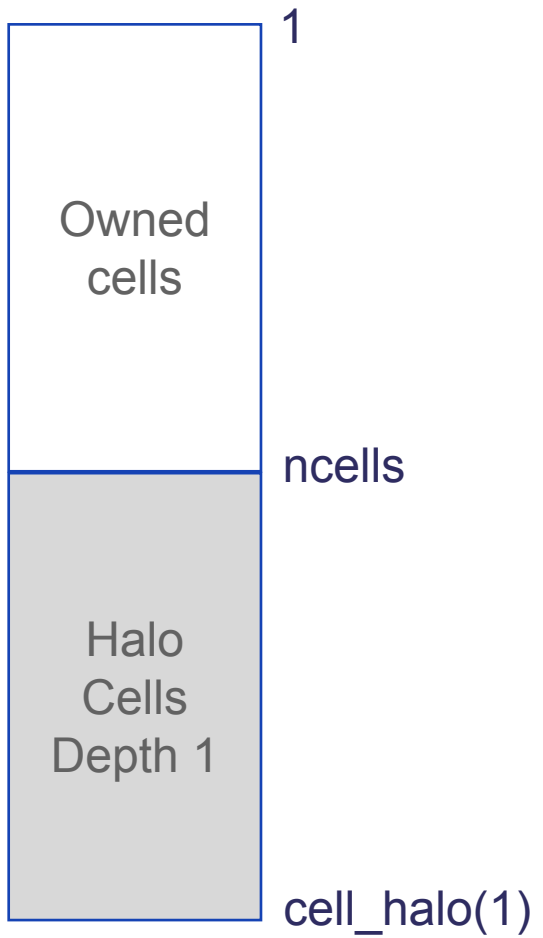
Redundant computation

Iterate over cells

```
DO i=1,cell_halo(1)
    call kern()
```



Cells Numbering scheme



ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

PSyIR view

```
InvokeSchedule[invoke='invoke_0', dm=True]
    0: Loop[type='dofs', field_space='any_space_1', it_space='dof', upper_bound='ndofs']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: BuiltIn setval_c(grad_p,0.0_r_def)
    1: HaloExchange[field='grad_p', type='region', depth=1, check_dirty=False]
    2: HaloExchange[field='p', type='region', depth=1, check_dirty=True]
    3: HaloExchange[field='div_star', type='region', depth=1, check_dirty=True]
    4: HaloExchange[field='hb_inv', type='region', depth=1, check_dirty=True]
    5: Loop[type='', field_space='any_space_1', it_space='cell_column', upper_bound='cell_halo(1)']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern scaled_matrix_vector_code(grad_p,p,div_star,hb_inv) [module_inline=False]
    6: Loop[type='', field_space='any_space_1', it_space='cell_column', upper_bound='cell_halo(1)']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern enforce_bc_code(grad_p) [module_inline=False]
    7: HaloExchange[field='mt_lumped_inv', type='region', depth=1, check_dirty=True]
    8: Loop[type='', field_space='w3', it_space='cell_column', upper_bound='ncells']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern apply_variable_hx_code(hp,grad_p,mt_lumped_inv,p,compound_div,p3theta,ptheta2,m3_exner_star,tau_t,timeste
p_term) [module_inline=False]
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Halo exchange logic

```
InvokeSchedule[invoke='invoke_0', dm=True]
    0: Loop[type='dofs', field_space='any_space_1', it_space='dof', upper_bound='ndofs']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: BuiltIn setval_c(grad_p,0.0_r_def)
    1: HaloExchange[field='grad_p', type='region', depth=1, check_dirty=False]
    2: HaloExchange[field='p', type='region', depth=1, check_dirty=True]
    3: HaloExchange[field='div_star', type='region', depth=1, check_dirty=True]
    4: HaloExchange[field='hb_inv', type='region', depth=1, check_dirty=True]
    5: Loop[type='', field_space='any_space_1', it_space='cell_column', upper_bound='cell_halo(1)']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern scaled_matrix_vector_code(grad_p,p,div
    6: Loop[type='', field_space='any_space_1', it_space='cell_
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFI
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFI
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern enforce_bc_code(grad_p) [module_inline
    7: HaloExchange[field='mt_lumped_inv', type='region', depth
    8: Loop[type='', field_space='w3', it_space='cell_column',
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFI
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFI
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern apply_variable_hx_code(hp,grad_p,mt_lu
p_term) [module_inline=False]
```

Loop 1
Writes to `grad_p`
`1` to `ndofs`
Makes `grad_p` halo dofs “dirty”
Makes `grad_p` annexed dofs “dirty”
No reads
No halo exchange needed

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

Halo exchange logic

```
InvokeSchedule[invoke='invoke_0', dm=True]
    0: Loop[type='dofs', field_space='any_space_1', it_space='dof', upper_bound='ndofs']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: BuiltIn setval_c(grad_p,0.0_r_def)
    1: HaloExchange[field='grad_p', type='region', depth=1, check_dirty=False]
    2: HaloExchange[field='p', type='region', depth=1, check_dirty=True]
    3: HaloExchange[field='div_star', type='region', depth=1, check_dirty=True]
    4: HaloExchange[field='hb_inv', type='region', depth=1, check_dirty=True]
    5: Loop[type='', field_space='any_space_1', it_space='cell_column', upper_bound='cell_halo(1)']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER,
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER,
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern scaled_matrix_vector_code(grad_
    6: Loop[type='', field_space='any_space_1', it_space
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER,
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER,
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern enforce_bc_code(grad_p) [module
    7: HaloExchange[field='mt_lumped_inv', type='region'
    8: Loop[type='', field_space='w3', it_space='cell_co
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER,
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER,
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern apply_variable_hx_code(hp,grad_
p_term) [module_inline=False]
```

Loop 2
modifies `grad_p` [continuous?]
`1` to `cell_halo(1)`
Makes `grad_p` halo depth 1 dofs “dirty”
Makes `grad_p` annexed dofs “clean”
Needs `grad_p` annexed dofs
Loop 1 makes them “dirty”
Halo exchange needed

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

Halo exchange logic

```
InvokeSchedule[invoke='invoke_0', dm=True]
    0: Loop[type='dofs', field_space='any_space_1', it_space='dof', upper_bound='ndofs']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: BuiltIn setval_c(grad_p,0.0_r_def)
    1: HaloExchange[field='grad_p', type='region', depth=1, check_dirty=False]
    2: HaloExchange[field='p', type='region', depth=1, check_dirty=True]
    3: HaloExchange[field='div_star', type='region', depth=1, check_dirty=True]
    4: HaloExchange[field='hb_inv', type='region', depth=1, check_dirty=True]
    5: Loop[type='', field_space='any_space_1', it_space='cell_column', upper_bound='cell_halo(1)']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern scaled_matrix_vector_code(grad_p,p,div_star,hb_inv) [module_inline=False]
    6: Loop[type='', field_space='any_space_1', it_space='cell_column', upper_bound='cell_halo(1)']
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFINED>]
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFIN
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern enforce_bc_code(grad_p) [module_inline=
    7: HaloExchange[field='mt_lumped_inv', type='region', depth=
    8: Loop[type='', field_space='w3', it_space='cell_column', u
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFIN
        Literal[value:'NOT_INITIALISED', Scalar<INTEGER, UNDEFIN
        Literal[value:'1', Scalar<INTEGER, UNDEFINED>]
        Schedule[]
            0: CodedKern apply_variable_hx_code(hp,grad_p,mt_lum
p_term) [module_inline=False]
```

Loop 2
reads `p`, `div_star`, `hb_inv`
`1` to `cell_halo(1)`
Halo exchange needed if halos are "dirty": runtime check

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Runtime dirty flags

```
      DO df=1,grad_p_proxy%vspace%get_last_dof_owned()
        grad_p_proxy%data(df) = 0.0_r_def
      END DO
      !
      ! Set halos dirty/clean for fields modified in the above loop
      !
      CALL grad_p_proxy%set_dirty()
      !
      CALL grad_p_proxy%halo_exchange(depth=1)
      !
      IF (p_proxy%is_dirty(depth=1)) THEN
        CALL p_proxy%halo_exchange(depth=1)
      END IF
      !
      IF (div_star_proxy%is_dirty(depth=1)) THEN
        CALL div_star_proxy%halo_exchange(depth=1)
      END IF
      !
      IF (hb_inv_proxy%is_dirty(depth=1)) THEN
        CALL hb_inv_proxy%halo_exchange(depth=1)
      END IF
      !
      DO cell=1,mesh%get_last_halo_cell(1)
        !
        CALL scaled_matrix_vector_code(nlayers, grad_p_proxy%data, p_proxy%data, div_star_proxy%data, hb_inv_proxy%data, ndf_aspc1_gr
ad_p, undf_aspc1_grad_p, map_aspc1_grad_p(:,cell), ndf_aspc2_p, undf_aspc2_p, map_aspc2_p(:,cell), ndf_w3, undf_w3, map_w3(:,cell))
      END DO
      !
      ! Set halos dirty/clean for fields modified in the above loop
      !
      CALL grad_p_proxy%set_dirty()
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

“Annexed dofs” optimisation

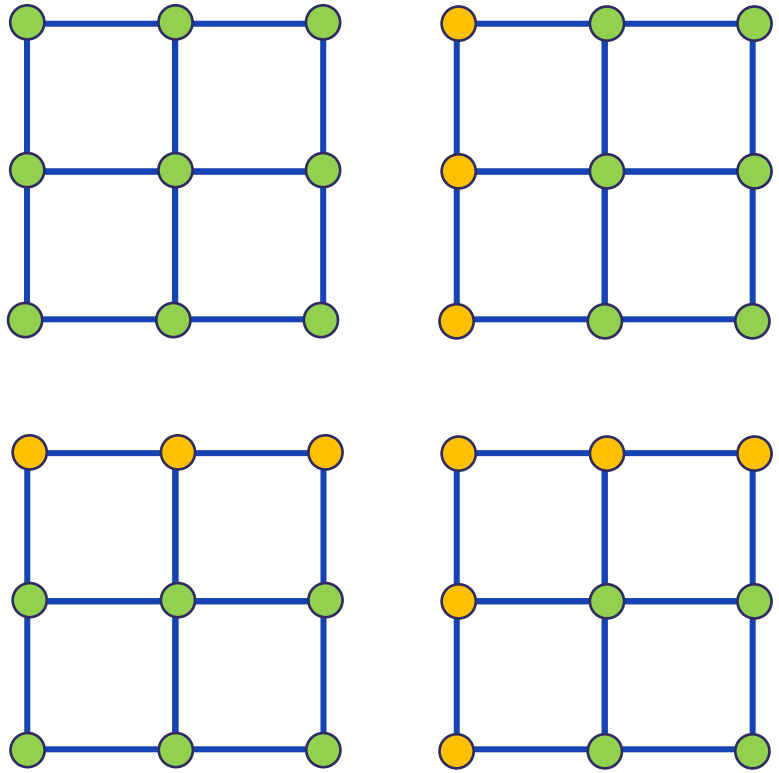
- LFRic either loops over cells or dofs
- If we iterate over cells, we always guarantee that the annexed dofs for a modified continuous field will be clean due to redundant computation
- If we always redundantly compute annexed dofs when we iterate over dofs there will be no additional halo exchanges required
- We have seen in example 1 that dirty annexed dofs may result in halo exchanges
- So …
- Redundantly compute “annexed dofs” for the whole code then we can assume that annexed dofs are always clean and potentially reduce the number of halo exchanges

ESiWACE2 has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement No 823988

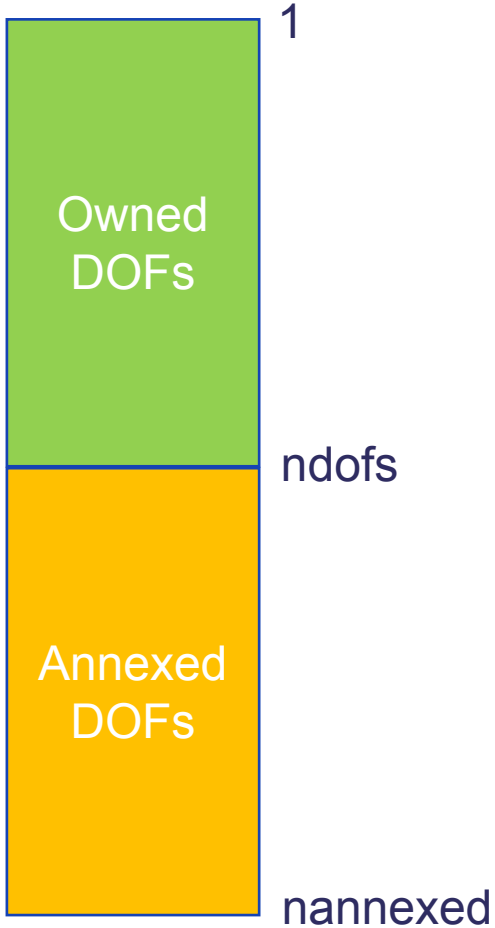
Partitioned continuous DOFs

Iterate over dofs

```
DO i=1,nannexed
    ! builtin code
```



DOFs Numbering scheme



ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Annexed dofs: Halo exchange reduction

Number of halo exchanges	% reduction	Description
63,668		LFRic Fallow Deer release
63,000	1%	GH_INC : no halo exchange for increments
46,274	27%	ANNEXED : redundantly compute annexed dofs
19,892	69%	ANNEXED + GH_INC

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Overlapping communication with computation

- If you can't avoid a halo exchange then you could try to overlap it with computation
- PSyclone provides 2 transformations to help with this
  1. A transformation that changes the default synchronous halo exchange into an asynchronous halo exchange
  2. A transformation that moves PSyIR nodes which, in this case, can be used to make halo exchanges overlap with computation

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Supporting reductions

- LFRic and PSyclone support two types of communication pattern
  - nearest neighbour (stencil) communication patterns - halo exchanges
  - Reductions

```
do i = 1, n
  a = a + data(i)
end do
```

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

What else?

- Now
  - Arbitrary depth redundant computation via a PSyclone transformation

- Future
  - Loop splitting (into internal computation and halo computation)
  - Communication aggregation
  - Eager (asynchronous) halo exchange protocol
  - Test working with partial sums and annexed dofs instead of redundant computation in the halo?
  - Non-MPI based comms?

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

esiwace
CENTRE OF EXCELLENCE IN SIMULATION OF WEATHER AND CLIMATE IN EUROPE

Hands on

- ~45 minutes
- `cd <psyclone_home>/tutorial/practicals/LFRic/distributed_memory`
- 3 parts `2_annexed_dofs, 3_overlapping_comms, 4_reductions`
- No compilation, just code generation
- Follow the `README.md` in the directories
  - A browser will display `README.md` files nicely
  - https://github.com/stfc/PSyclone/tree/master/tutorial/practicals/LFRic/distributed_memory
- Any issues/questions on slack
  - Use the psyclone channel
  - Please use threads for replies

Have fun!

ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988