Introduction to DSLs

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

Simulation

- Levels
    - Maths
    - Discretisation
    - Algorithm
    - Implementation

Application scientist

Numericist

HPC expert

Abstract Description

Concrete Implementation

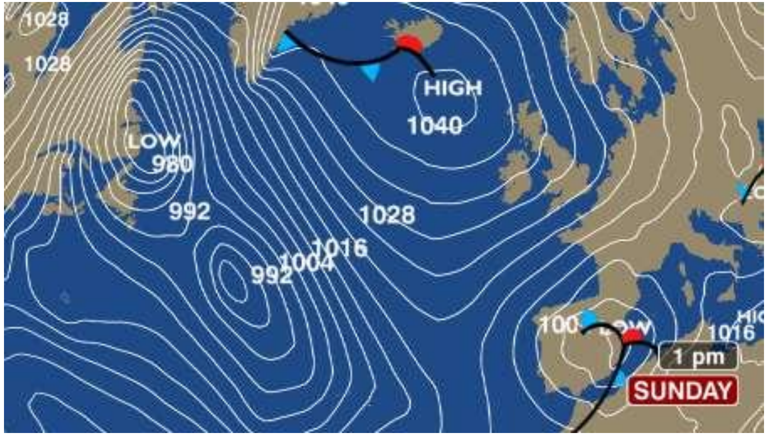
- Schulthess, T. Programming revisited. *Nature Phys* 11, 369–373 (2015). https://doi.org/10.1038/nphys3294

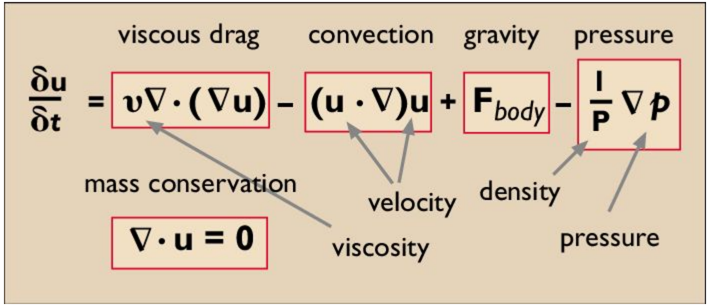
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Worked example: Computing a gradient

- Why? Frequently used in Weather and Climate models

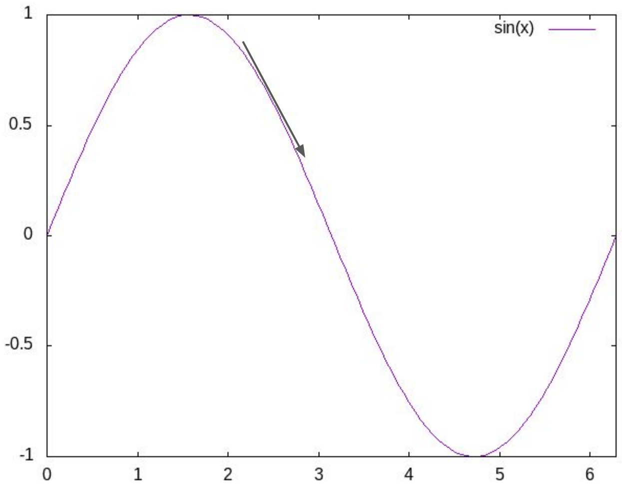
- For example: tightly packed isobars (pressure gradient) means strong winds
- Wind acceleration is proportional to pressure gradient





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

Maths level



$$grad f = \nabla f$$

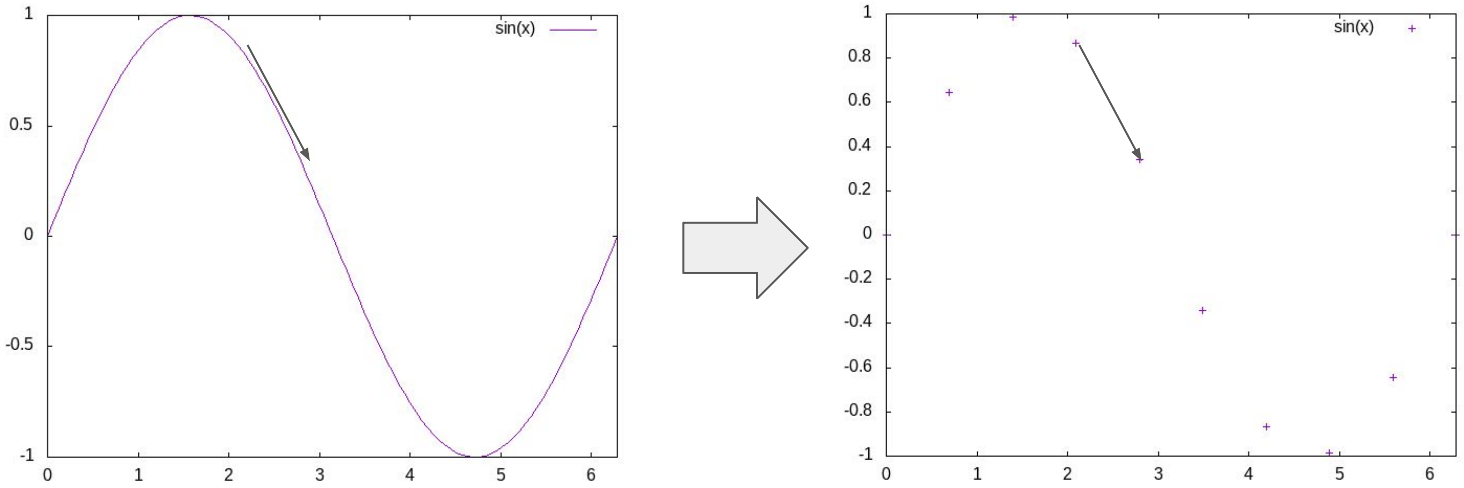
https://www.youtube.com/watch?v=M0u9Qy3SERI

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

Discretisation level

- Choose finite elements, finite volume, finite difference

https://www.youtube.com/watch?v=9WE4zKCLxW8

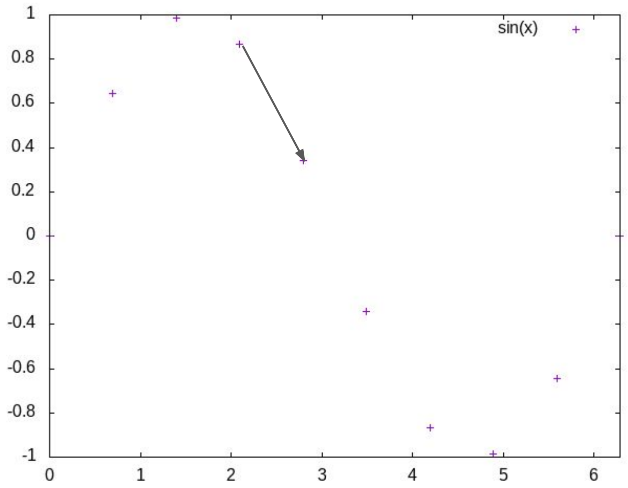


ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Algorithm level

Compare with calculus ...

- Choose multigrid, order of the scheme, etc.
- Different ways to work out gradient
- Here we use a simple 1st order scheme



$$f'(a) = \lim_{h \to 0} \frac{f(a+h) - f(a)}{h}$$

$$\underline{\nabla}_{\underline{n}}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

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

Implementation level

- This is what is compiled and run
- Code taken from the ICON model
- 3D mesh
- Fortran

$$\underline{\nabla}_{\underline{n}}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
DO jk = slev, elev
 DO je = i_startidx, i_endidx
   grad_norm_psi_e(je,jk) =
      (psi_c(iidx(je,2),jk)-psi_c(iidx(je,1),jk))/lhat(je)
 ENDDO
END DO
```

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

Running fast/parallel

- Original serial code
- (very) straight forward implementation
- "actual science" + mesh

```
DO jk = slev, elev
 DO je = i_startidx, i_endidx
   grad_norm_psi_e(je,jk) =
      (psi_c(iidx(je,2),jk)-psi_c(iidx(je,1),jk))/lhat(je)
 ENDDO
END DO
```

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

Running fast/parallel

- turns out that the mesh is too large for one machine and therefore runs slowly, so add blocks

```
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, jb, i_startblk, i_endblk, &
                   i_startidx, i_endidx, rl_start, rl_end)
 DO jk = slev, elev
   DO je = i_startidx, i_endidx
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
```

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

Running fast/parallel

- add directives to exploit multiple cores on shared memory machines

```
#ifdef _OMP
!$OMP PARALLEL
!$OMP DO PRIVATE(jb, i_startidx, i_endidx, je, jk)
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, jb, i_startblk, i_endblk, &
                    i_startidx, i_endidx, rl_start, rl_end)
 DO jk = slev, elev
   DO je = i_startidx, i_endidx
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP END DO NOWAIT
!$OMP END PARALLEL
#endif
```

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

Running fast/parallel

- code also needs to target an architecture with a GPU accelerator ...
- ... which has a different optimal memory layout

```
#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

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

Running fast/parallel

$$\underline{\nabla}_{\underline{n}}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
        psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

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

Running fast/parallel

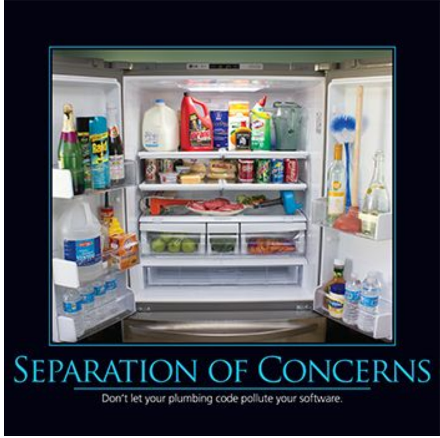
What if

- Requirements change, e.g. it turns out that this gradient should have been approximated using a higher order stencil?
- A third (fourth...) architecture needs to be supported?
- The mesh library needs to be replaced?
- Loops should be fused together for greater performance on a particular architecture?
- A compiler has a bug that needs a workaround?

```
#ifdef _OMP
!$OMP ....
#else
!$ACC ....
#endif
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 #ifdef __LOOP_EXCHANGE
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
 #else
   DO jk = slev, elev
     DO je = i_startidx, i_endidx
 #endif
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
#ifdef _OMP
!$OMP ...
#else
!$ACC ...
#endif
```

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

Separation of Concerns



- Performance portable maintainable code is difficult to achieve

  What can we do?

- Can we separate the specification/coding of the science from its optimisation?
- This would
  - allow the scientists to concentrate on developing the science
  - Allow HPC experts to concentrate on optimising the code

  Domain-specific languages (DSLs) offer a way to do this ...

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

Domain Specific Languages

- Languages tailored to a (very) specific purpose
  - as opposed to general purpose programming languages like C, C++, Java, Python...
- This definition is quite general and includes things like:
  - HTML for web pages
  - PostScript for documents
  - MATLAB for maths processing
- However, we focus on DSLs for High Performance Computing (HPC)

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

Domain Specific Languages

- DSL Frameworks are becoming a more and more viable approach for device-specific code generation, often achieving performance numbers unatainable for general purpose compilers
- Since DSLs are, well, domain specific, they are very expressive for the domain they are tailored to
  - shorter code, better maintainability
- Some application domains for HPC using DSLs include
  - Image Processing (Halide)
  - Deep Learning (XLA)
  - Climate & Numerical Weather Prediction (Stella, Gridtools, dawn, PSyclone)

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

Benefit of DSL vs coding

The DSL Idea

$$\underline{\nabla_n}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
grad_norm_psi_e =
      reduce( psi_c,
              CELL > EDGE,
              [1/lhat, -1/lhat]
           )
```

OMP

DSL compiler

```
!$OMP PARALLEL
!$OMP DO PRIVATE(jb, i_startidx, i_endidx, je, jk)
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 DO je = i_startidx, i_endidx
   DO jk = slev, elev
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
!$OMP END DO NOWAIT
!$OMP END PARALLEL
```

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

Benefit of DSL vs coding

The DSL Idea

$$\underline{\nabla_n}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
grad_norm_psi_e =
      reduce( psi_c,
               CELL > EDGE,
               [1/lhat, -1/lhat]
             )
```

Open ACC

DSL compiler

```
!$ACC PARALLEL &
!$ACC PRESENT(ptr_patch, iidx, iblk, pci_c, grad_...)
!$ACC LOOP GANG
DO jb = i_startblk, i_endblk
 CALL get_indices_e(ptr_patch, ...)
 DO jk = slev, elev
   DO je = i_startidx, i_endidx
     grad_norm_psi_e(je,jk,jb) =  &
       ( psi_c(iidx(je,jb,2),jk,iblk(je,jb,2)) -
         psi_c(iidx(je,jb,1),jk,iblk(je,jb,1)) )
       / ptr_patch%edges%lhat(je,jb)
   ENDDO
 END DO
END DO
!$ACC END PARALLEL
!$ACC END DATA
```

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

Benefit of DSL vs coding

The DSL Idea

$$\underline{\nabla_n}\psi(e) = \frac{\psi(c_1(e)) - \psi(c_0(e))}{\hat{l}}$$

```
grad_norm_psi_e =
      reduce( psi_c,
              CELL > EDGE,
              [1/lhat, -1/lhat]
           )
```

C++

DSL compiler

```
for(int k = 0 + 0; k < m_k_size; ++k) {
 for(auto const& loc : getEdges(LibTag{}, m_mesh)) {
   for(auto inner_loc :
     grad_norm_psi_e(loc, k + 0) = reduce(
         LibTag{}, m_mesh, loc, (::dawn::float_type)0.0,
         std::vector<dawn::LocationType>
         {dawn::Edges, dawn::Cells},
         [&](auto& lhs, auto red_loc1, auto const& weight)
         {
           lhs += weight *psi_c(red_loc1, k + 0);
          return lhs;
         },
         std::vector<::dawn::float_type>({-1.0 ,1.0});
   }
   grad_norm_psi_e(loc, k + 0) /= lhat_e(loc, k + 0)
 }
}
```

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

Existing code & DSLs

Evolution rather than Revolution

- Although DSLs are very powerful, an application must be re-written in order to use them
- Applications in the weather/climate domain are large and under continuous development
- DSLs are relatively new and untested in this domain
    - Concerns over longevity of necessary tool chains
- To stop development on existing code and re-develop from scratch is expensive (time and effort)
- Community has a lot of skill and knowledge in existing coding approaches (Fortran)

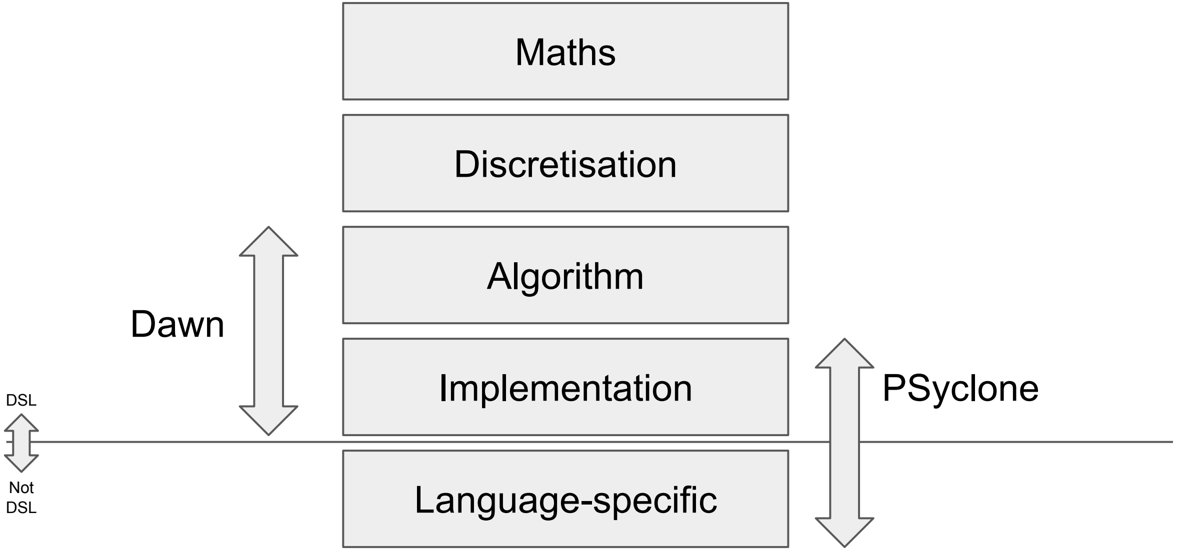
Very attractive to be able to *translate* existing code into a DSL or use existing code in a DSL rather than re-write:

- Support science that cannot be specified in the DSL language
- Transition to high level DSLs by evolution not revolution
- Support code generation and translation

Need to regain lost information

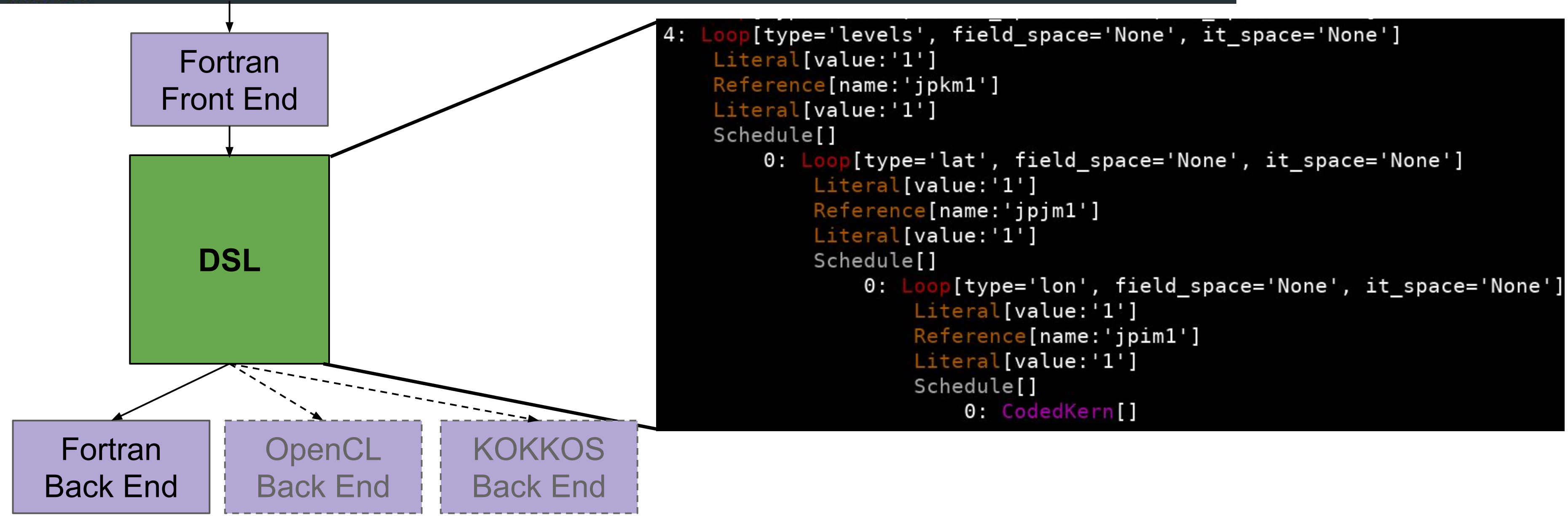
ESiWACE2 has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 823988

Levels of abstraction



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

```fortran
! Horizontal tracer gradient
DO jk = 1, jpkm1
   DO jj = 1, jpjm1
      DO ji = 1, jpim1   ! vector opt.
         zdit(ji,jj,jk) = ( ptb(ji+1,jj  ,jk,jn) - ptb(ji,jj,jk,jn) ) * umask(ji,jj,jk)
         zdjt(ji,jj,jk) = ( ptb(ji  ,jj+1,jk,jn) - ptb(ji,jj,jk,jn) ) * vmask(ji,jj,jk)
      END DO
   END DO
END DO
```



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

Summary

- Modelling required expertise in multiple disciplines (co-design)
- These disciplines work at different levels of abstraction
- Mixing science and performance can produce complex code
- Good to separate these concerns
- DSLs offer a way to do this
- DSLs support working at a high level of abstraction
- Higher level of abstraction allows a greater choice of implementation -> more performance
- Different DSLs can work at different levels of abstraction
- DSLs might support revolution and/or evolution

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

Next

- Break
- Dawn intro
- PSyclone intro
- Tutorial

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