A python implementation of the ICON dynamical core for operational NWP

D. Hupp[1], C. Müller[1], N. Burgdorfer[1], A. Gopal[2/3], N. Farabullini[2], T. Ehrengruber[2/3], S. Kellerhals[2], P. Kardos[2], M. Luz, M. Röthlin[1], E. G. Paredes[2/3], B. Weber[1], R. Häuselmann[2/3], F. Thaler[2/3], J. Jucker[2], L. Groner[2/3], H. Vogt[2/3], M. Bianco[2/3], A. Dipankar[2], C. Osuna[1], X. Lapillonne[1]
[1] MeteoSwiss, [2] ETH Zürich, [3] CSCS, [4] DWD

ICON EXCLAIM

- The ICON (ICOsahedral Non-hydrostatic) modeling framework1 has been developed by the Deutscher Wetterdienst (DWD) and the Max Planck Institute for Meteorology (MPI-M).
- EXCLAIM - an open ETH Zurich project that aims to develop a Python framework based on the ICON model, that is capable of running kilometer-scale climate simulations at both regional and global scales.

ICON partners

DWD KIT DKRZ MAX-PLANCK-INSTITUT FÜR METEOROLOGIE

ICON dev. partners

COSMO CSCS and more …

GT4Py vs Fortran Source Code

Divergence in GT4Py:

```
neighbor_sum(vn(CellToEdge) * geofac_div,
axis=CellToEdgeDim)
```

Divergence in Fortran:

```
DO jb = i_startblk,i_endblk

 CALL get_indices_c(p_patch, jb, i_startblk, i_endblk, &
   i_startidx, i_endidx, rl_start, rl_end)

 DO jk = 1, nlev
   DO jc = i_startidx, i_endidx

    div(jc,jk) =  &
       p_nh_prog%vn(ieidx(jc,jb,1),jk,ieblk(jc,jb,1)) * &
       p_int%geofac_div(jc,1,jb) + &

       p_nh_prog%vn(ieidx(jc,jb,2),jk,ieblk(jc,jb,2)) * &
       p_int%geofac_div(jc,2,jb) + &

       p_nh_prog%vn(ieidx(jc,jb,3),jk,ieblk(jc,jb,3)) * &
       p_int%geofac_div(jc,3,jb)

   ENDDO
 ENDDO
ENDDO
```

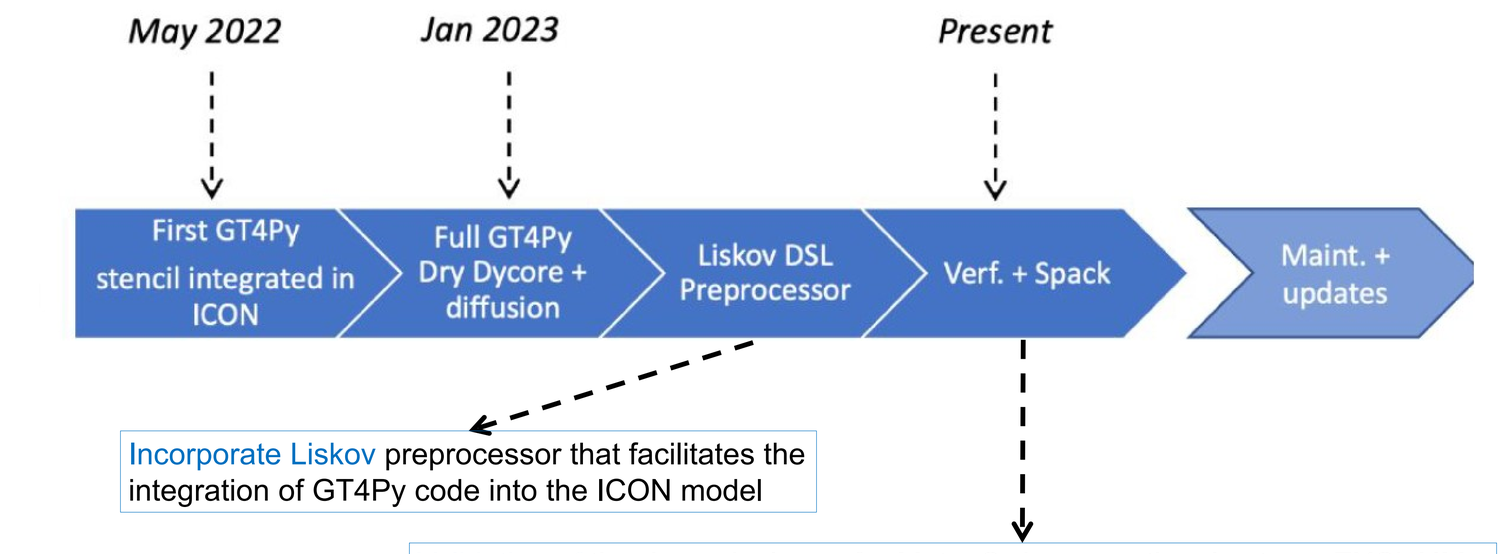
Comparison:

- No loops or explicit indices
- No blocking (compiler handles this internally as optimization)
- No explicit iteration over neighbors (still possible in GT4Py)
- No !$ACC or !$OMP directives

Future goals and plans

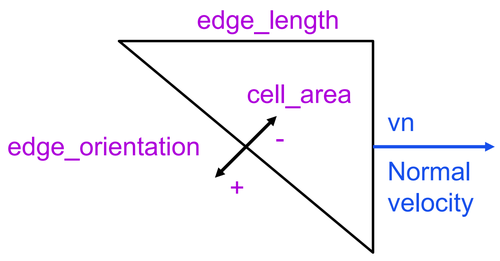
- Fully implemented dycore for MCH operational setup.
- Python dycore needs to show a performance improvement of the overall runtime.
- Work in progress, reduce number of kernel executions.
- Diffusion: 6 instead of 18 kernels 1.3x speedup (on Tsa).
- The GT4Py dycore should be as easily debug-able as possible.
- Continuously adding features to approve quality of life.
- The GT4Py dycore needs to be robustly tested.
- Weekly pipeline that takes less than an hour.

Recent updates and progress



GT4Py: framework for weather and climate modeling

Stencil code is written in an abstracted Python based language and executed with high performance on different machine architectures, e.g. CPUs and GPUs.




```
geofac_div(cell_idx,1:3) =

edge_length(1:3) *
edge_orientation(cell_idx,1:3) /
cell_area(cell_idx)
```

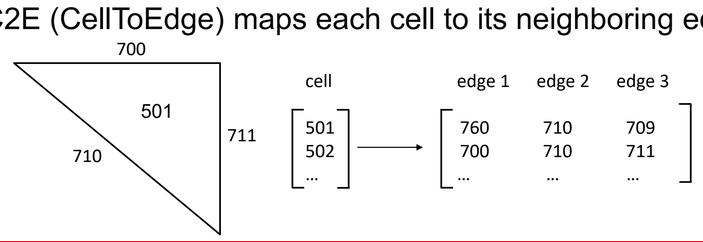
Divergence operator in GT4Py:

```
@field_operator
def divergence(
 vn:Field[[EdgeDim,KDim], float], geofac_div:Field[[CellDim,C2EDim], float],
) -> Field[[CellDim, KDim], float]:
   return neighbor_sum(vn(C2E) * geofac_div, axis=C2EDim)
```

neighbor_sum is shorthand for:

```
vn(C2E(1)) * geofac_div(1) +
vn(C2E(2)) * geofac_div(2) +
vn(C2E(3)) * geofac_div(3)
```

C2E (CellToEdge) maps each cell to its neighboring edges:



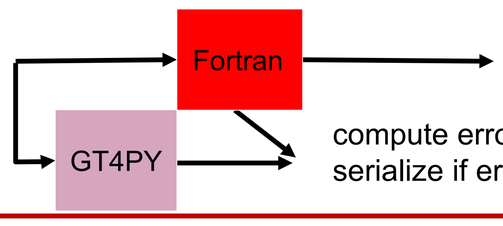
Fortran Integration and Verification with ICON Liskov

ICON Liskov is a directive-based preprocessor which parses comments and substitutes them with code, facilitating the integration of the GT4Py generated code into the ICON model.

```
!$DSL START STENCIL(name=mo_nh_diffusion_stencil_10; w=p_nh_prog%w(:,:,1);
diff_multfac_n2w=diff_multfac_n2w(:); &
!$DSL        cell_area=p_patch%cells%area(:,1); z_nabla2_c=z_nabla2_c(:,:,1); vertical_lower=2; &
!$DSL        vertical_upper=nrdmax(jg); horizontal_lower=i_startidx; horizontal_upper=i_endidx)
DO jk = 2, nrdmax(jg)
  DO jc = i_startidx, i_endidx
    p_nh_prog%w(jc,jk,jb) = p_nh_prog%w(jc,jk,jb) + diff_multfac_n2w(jk) * p_patch%cells%area(jc,jb) * &
        z_nabla2_c(jc,jk,jb)
  ENDDO
ENDDO
!$DSL END STENCIL(name=mo_nh_diffusion_stencil_10)
```

ICON Liskov generates code in two modes, verification and substitution

Verification mode
execute both GT4Py and Fortran versions
continue with Fortran results



Substitution mode
only execute the GT4Py version
continue with GT4Py results

Preliminary Performance results of ICON-CH2-EPS on Alps (MeteoSwiss)

ICON-CH2-EPS (142'000 column per GPU, 1000 timesteps)

	ICON GPU OpenACC	ICON GPU GT4Py	ICON GPU tuned* GT4Py
nh_solve	0.02192	0.02327	0.02204
nh_solve.veltend	0.00310	0.00346	0.00299
nh_diff	0.00536	0.00543	0.00524

*Individual tuned stencils with the equivalent of **TILE** tested over 3 configurations: **128/1, 128/4, 256/1**.

HPC Computing Services on Alps Plattform

- GPU nodes:
  - 4 x NVIDIA A100
  - 1 x AMD Epyc 64-cores CPU
- CPU nodes:
  - 2 x AMD Epyc 64-cores CPUs
- 2 Virtual Clusters (VC)
  - Production: 42 GPU / 15 CPU Nodes
  - R&D: 30-50 GPU / ~15 CPU Nodes (elastic)