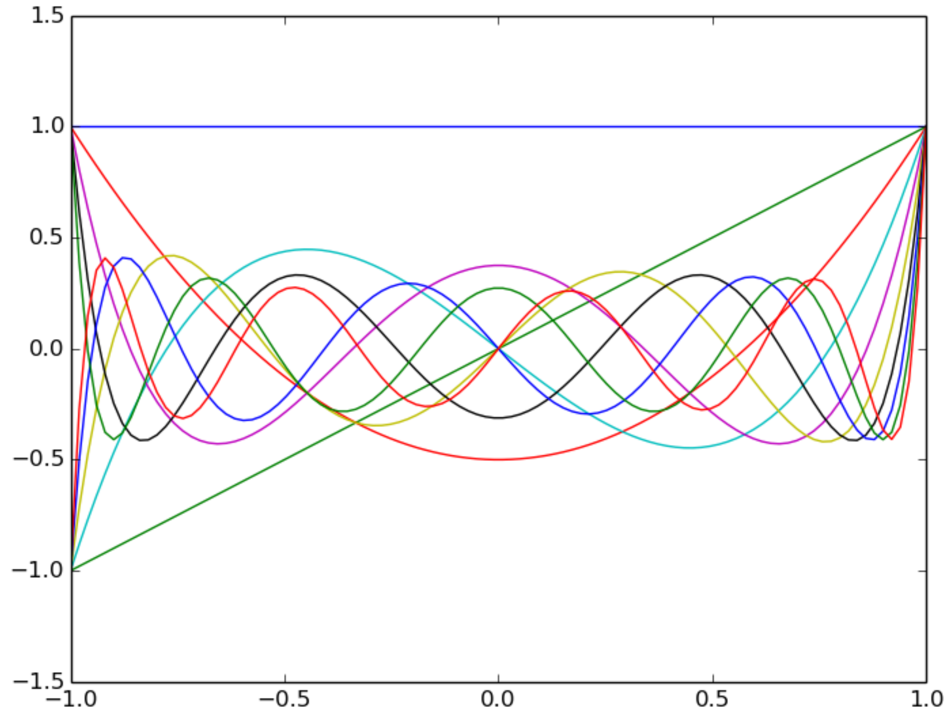
High Order Simulations in Ateles

A brief primer

Modal Representation

- Ateles uses modal and nodal representations as needed
- Legendre modes:



- Nodes accordingly

The Polynomial Degree

- Accuracy of solution representation within elements is given by maximal polynomial degree
- Configured by `scheme.spatial.m`
- Scheme order is actually m+1
- Arbitrary polynomial degrees can be used

Projections

- Ateles uses modal and nodal representations as needed
- Therefore, need to change from one to the other

- Achieved by projection, as given in the projection table
- Two methods:
  - Fast Polynomial Tansform (FPT)
  - L_2-Projection (L2P)
- Factor for dealiasing to deal with nonlinear operations

- Use FPT for high orders (m>20)

Stabilization

- High orders suffer from oscillations in the proximity of discontinuities
- Need for stabilization to avoid unphysical states
- 2 approaches available:
  - Spectral filtering: Damp down high modes
  - Positivity preserving: Scales modes down, to maintain positivity at integration points
- Reduces quality of solution, not needed for smooth solutions, but required in most flows

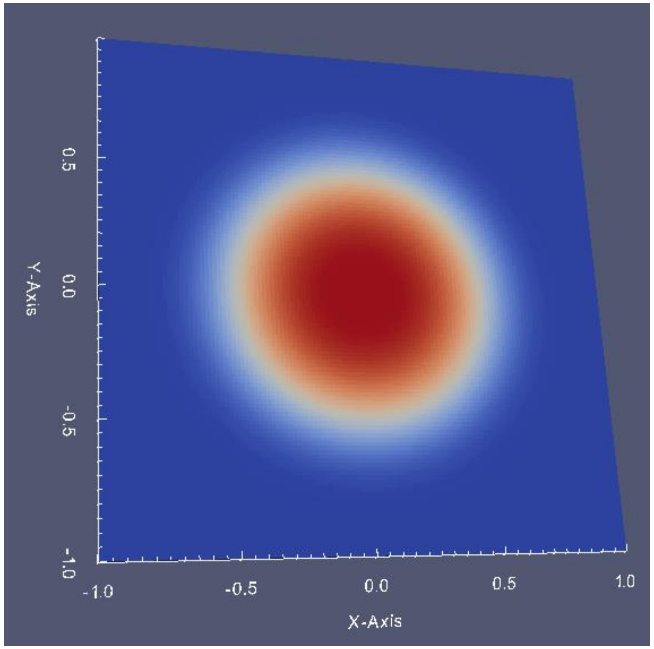
Stabilization Configuration

- Both stabilization methods can be used simultaneously

```
scheme.stabilization = {
  {
    -- Spectral viscosity, higher order = less strong
    name  = 'spectral_viscosity',
    order = 8
  },
  {
    -- Positivity preserving limiter:
    -- eps denotes the limit below which density and
    -- pressure will be considered unphysical.
    name = 'positivity_preserv',
    eps = 1.0e-08
  }
}
```

Task Description

- We will have a look at a Gaussian pulse in density, that is transported by a constant velocity through a periodic domain



- This is a pure transport problem, and the shape of the pulse should be maintained over the simulation
- The quality of the solution can therefore be evaluated after one complete period, by comparing the solution with the initial condition.

Task

- Copy the input scripts into your workspace and have a look at the ateles.lua
  `cp -r $KURS/exercises/hpcfdx7 $MYWS`
- There are 2 variables we want to vary:
  - level (**h**-refinement)
  - max_polynomial_degree (**p**-refinement)
- Evaluate the error with harvester

Error analysis using Atl-Harvesting

- The analytical solution as a variable is added to the variable table (lets say, dens_ref) in the ateles config file (ateles.lua)

```
variable = {
  {
    name = 'dens_ref',
    ncomponents = 1,
    dens_ref = ic_gauss_density
  }
)
```

Error analysis using Atl-Harvesting

- Now the difference between the variable (density) and the analytical solution (dens_ref) can be evaluated, using the predefined “difference” variable (as the name suggests, calculates the difference between two variables)

```
variable = {
  {
    name = 'dens_diff',
    ncomponents = 1,
    vartype = 'operation',
    operation = {
      kind = 'difference',
      input_varname = { 'density', 'dens_ref' },
    }
  }
)
```

Error analysis using Atl-Harvesting

- Use reduction functionality of Atl-Harvesting to perform operations on the variables (like sum, average, l2norm, max or min)
- Give as many reductions as there are variables, the first reduction belongs to the first variable and so on

Error analysis using Atl-Harvesting

- Therefore a tracking table is defined inside the config file harvester.lua

```
tracking = {
  {
    label = 'gauss',
    variable = { 'density', 'dens_ref', 'dens_diff' },
    reduction = { 'l2norm', 'l2norm', 'l2norm' },
    shape = {
      kind = 'canoND',
      object = {
        origin = { -1.0, eps, -1+eps },
        vec =  {{ 2.0, 0.0, 0.0 }},
        segments = { 100 },
        distribution = 'equal'
      }
    },
    folder = './',
    output = { format = 'ascii', use_get_point = true }
  },
}
```

Workflow

- Change the level and the maximum polynomial degree in the configuration file
  `gedit ateles.lua`
- Adapt the job script (do not use more processes than you have elements!)
  `gedit hp.job`
- Submit the computation job:
  `sbatch hp.job`
- Run Atl_harvesting by submitting the res job:
  `sbatch res.job`
- Find the results appended to gauss_p00000.res

P refinement : max_polynomial_degree →

H refinement : refinement_level ↓

h/p	3	5	7	9	11	13	21
3							
4							
5							
6							
7							

P refinement : max_polynomial_degree →

H refinement : refinement_level →

h/p	3	5	7	9	11	13	21
3							
4							
5							
6							
7							

Other things to look out for

- Checking the compute time needed for the respective simulations (timing.res, column simLoop)

- Visualizing the restart data in paraview

- Trying other density distributions, like a cone or cylinder (non-smooth distributions) in ateles.lua:

```
function ic_gauss_density(x,y)
  d= x*x+y*y
  fact = -0.5/(halfwidth*halfwidth)
  dens = background_density + (ampl_density * math.exp(fact*d*d))
  return(dens)
end
```

P refinement : max_polynomial_degree →

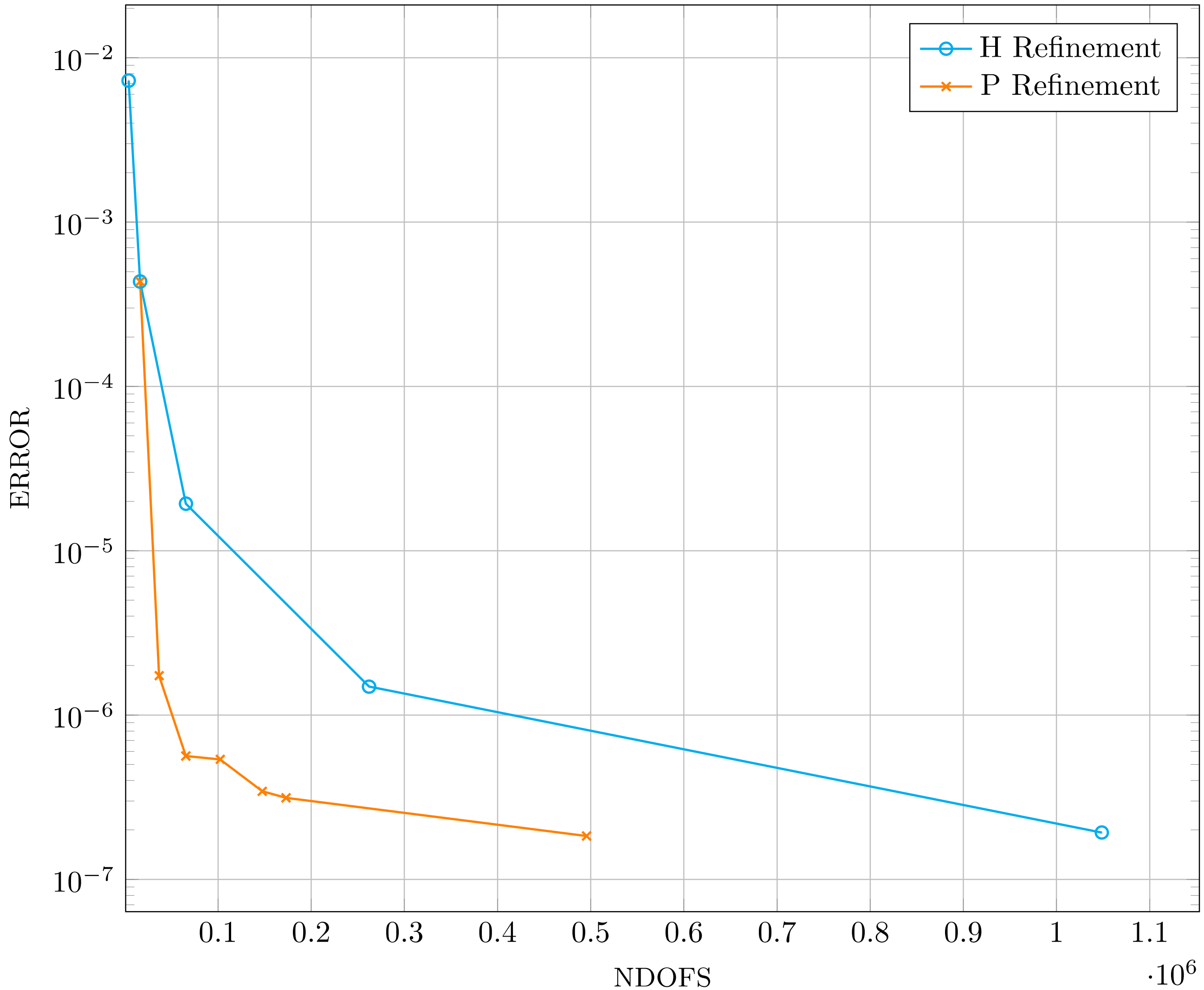
H refinement : refinement_level ↓

h/p	3	5	7	9	11	13	21
3	0.47×10^{-1}						
4	0.17×10^{-2}	0.81×10^{-5}	0.21×10^{-5}	0.16×10^{-5}	0.12×10^{-5}	0.12×10^{-5}	
5	0.48×10^{-4}						
6	0.33×10^{-5}						
7	0.11×10^{-5}						

- Total degree of freedom (ndofs) for 2D

$$= (p+1)^2 * nVars * total\ Elements$$

  - p = polynomial degree
  - nVars = 4 for euler 2D
  - total elements = $(2^{level})^2$
    - So, for level=1, the total number of elements is 4
    - On level = 3, the total elements are 16


H Refinement
P Refinement
ERROR
10−2
10−3
10−4
10−5
10−6
10−7
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
NDOFS
·10^6