HAL open science

Non-uniform meshes in gyrokinetic simulations

Bourne Emily, Grandgirard V., Munschy Y., Leleux P., Kruse C., Kormann K.

▶ **To cite this version:**

Bourne Emily, Grandgirard V., Munschy Y., Leleux P., Kruse C., et al.. Non-uniform meshes in gyrokinetic simulations. NumKin 2022 - Numerical methods for the kinetic equations of plasma physics, Eric Sonnendrücker, Nov 2022, Garching, Germany. cea-03949656

HAL Id: cea-03949656
https://cea.hal.science/cea-03949656

Submitted on 20 Jan 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

DE LA RECHERCHE À L'INDUSTRIE

Non-uniform meshes in gyrokinetic simulations

NumKin – 07/11/2022

Emily Bourne, V. Grandgirard, Y. Munschy, P. Leleux, C. Kruse, K. Kormann

- GYSELA code developed at IRFM/CEA since 2001 - strong collaboration between physicists, mathematicians and computer scientists
- 5D gyrokinetic code based on a backward semi-lagrangian scheme used to study plasma turbulence in tokamaks
- Requires state-of-the-art development on HPCs
  - Hybrid MPI/OpenMP parallelism
  - Optimized for up to 750,000 cores

https://gyselax.github.io

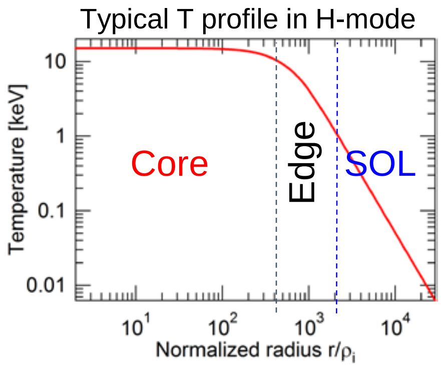
Intensive use of petascale resources

- > 100s of millions of hours / year
  - (GENCI + PRACE + HPC Fusion resources)

1 simulation:
- **100 billion points** (5D mesh: 3D space + 2D velocity)
- **> 8 million hours** (20 days / 18 432 cores),

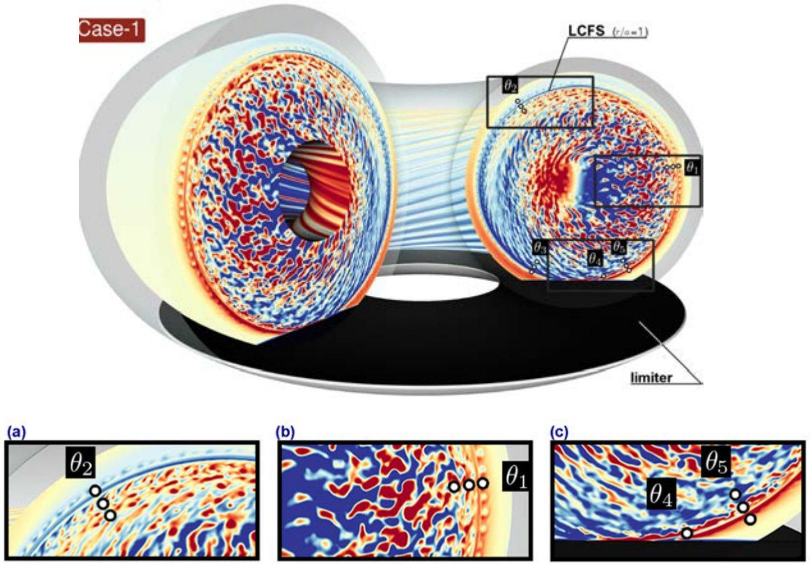
Exascale needed for ITER plasma turbulence simulation with electromagnetic effects

- Strong temperature gradients in Scrape-Off Layer (SOL)



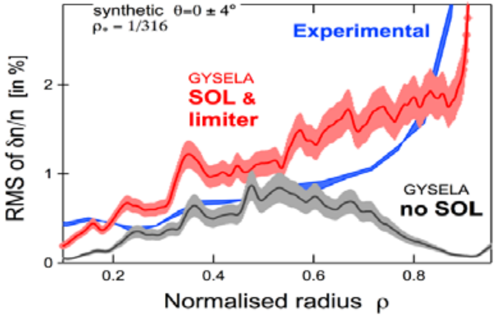
- Smaller structures in edge-region

[Dif-Pradalier 2020, 2022]



- More accurate results when modelling SOL

[Caschera PhD (2018), Dif-Pradalier (2018)]



1 – Vlasov-Poisson Simulations on Non-Uniform Points:
A reduced dimension case study

2 – 2D Poisson Solvers for Non-Uniform Points

GYSELA

5D Vlasov Solver

$$B^{\star}_{\parallel s}\frac{\partial F_s}{\partial t} + \nabla\cdot\left(\frac{dx_G}{dt}B^{\star}_{\parallel s}F_s\right) + \frac{\partial}{\partial v_{G\parallel}}\left(\frac{dv_{G\parallel}}{dt}B^{\star}_{\parallel s}F_s\right)$$
$$= C(F_s) + S + K_{buff}(F_s) + D_{buff}(F_s)$$

- Backward semi-Lagrangian Advection on cubic splines
- Penalisation for walls
- Collision operator

3D Poisson Solver

$$\frac{e}{T_{e,eq}}(\phi - \langle\phi\rangle) - \frac{1}{n_{e0}}\sum_s Z_s\nabla_\perp\cdot\left(\frac{n_{s,eq}}{B\Omega_s}\nabla_\perp\phi\right)$$
$$= \frac{1}{n_{e0}}\sum_s Z_s\int J_0\cdot(F_s - F_{s,eq})d^3v$$

- Finite Differences in r
- Fourier Transform in θ, φ

VOICE

2D Vlasov Solver

$$\partial_t f_s(t,x,v) + v\partial_x f_s(t,x,v) - \frac{q_s}{m_s}\partial_x\phi(t,x)\partial_v f_s(t,x,v)$$
$$= C_{ss}(t,x,v) + S_{s,w_1} + S_{s,w_2} + S_{s,k}$$

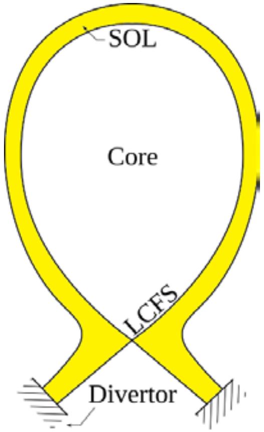
- Backward semi-Lagrangian Advection on arbitrary degree splines
- Penalisation for walls
- Collision operator

1D Poisson Solver

$$\partial_x^2\phi(t,x) = -\frac{\rho_q(t,x)}{\varepsilon_0}$$

- Finite Elements

- Sheath physics
- Kinetic electrons and ions
- Penalisation describes the wall



Vlasov Equation

$$\partial_t f_s(t,x,v) + v\partial_x f_s(t,x,v) - \frac{q_s}{m_s}\partial_x\phi(t,x)\partial_v f_s(t,x,v) = S_s(t,x,v) + C_{ss}(t,x,v)$$

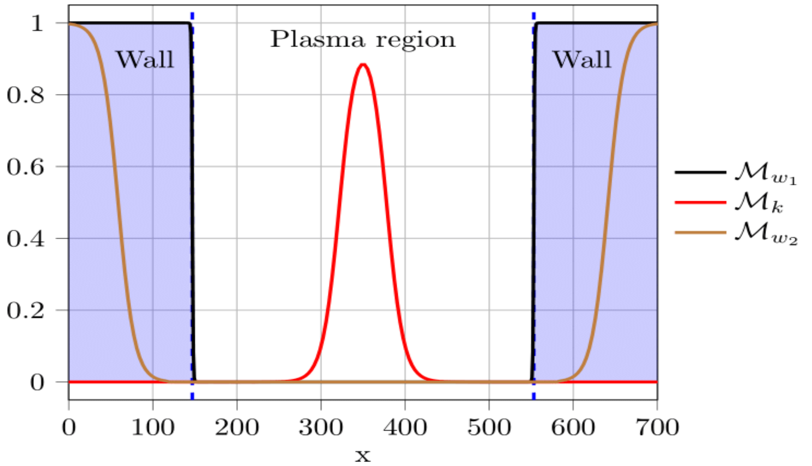
Poisson Equation

$$\partial_x^2\phi(t,x) = -\frac{\rho_q(t,x)}{\varepsilon_0}$$

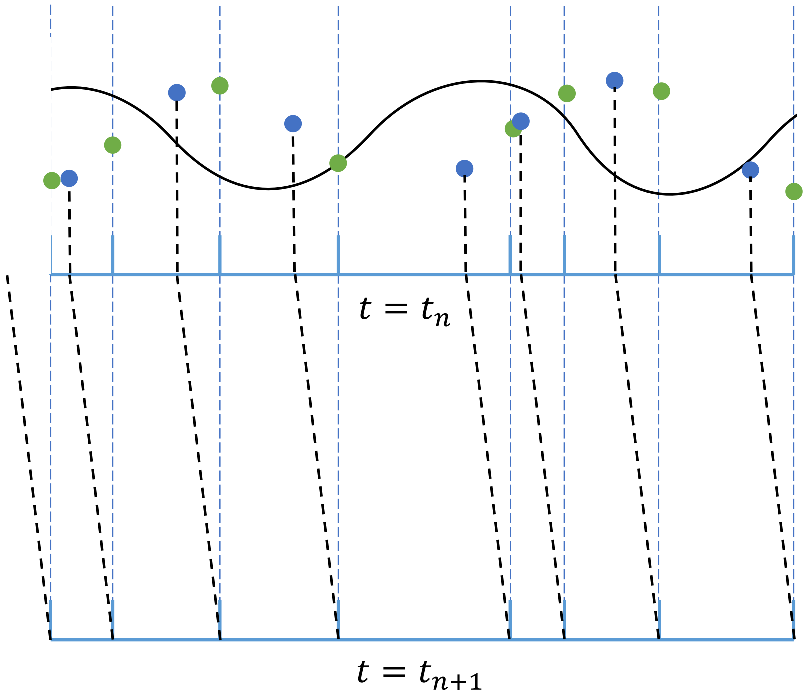
Sources

$$S_s(t,x,v) = S_{s,w_1} + S_{s,w_2} + S_{s,k}$$

- Particle sink
- Momentum/Energy sink
- Kinetic source



[1]E. Bourne, Y. Munschy et al, submitted *2022*
[2]Y. Munschy, E. Bourne et al, submitted *2022*



- The semi-Lagrangian method is a method for solving an advection equation

$$\partial_t F(x) + v(x) \cdot \nabla F(x) = 0$$

- It uses the fact that an advection is a translation

Method Steps

- Find the spline interpolation of the function
  - Requires non-uniform splines
- Find feet of the characteristics
- Approximate the value at the feet of the characteristics
  - Requires a binary search to find spline cell
- Update the values
- Handle boundary conditions

Non-Uniform Quadrature from Splines

- Newton-Cotes methods are optimised for equidistant points
- Spline representation can be used to calculate a quadrature scheme

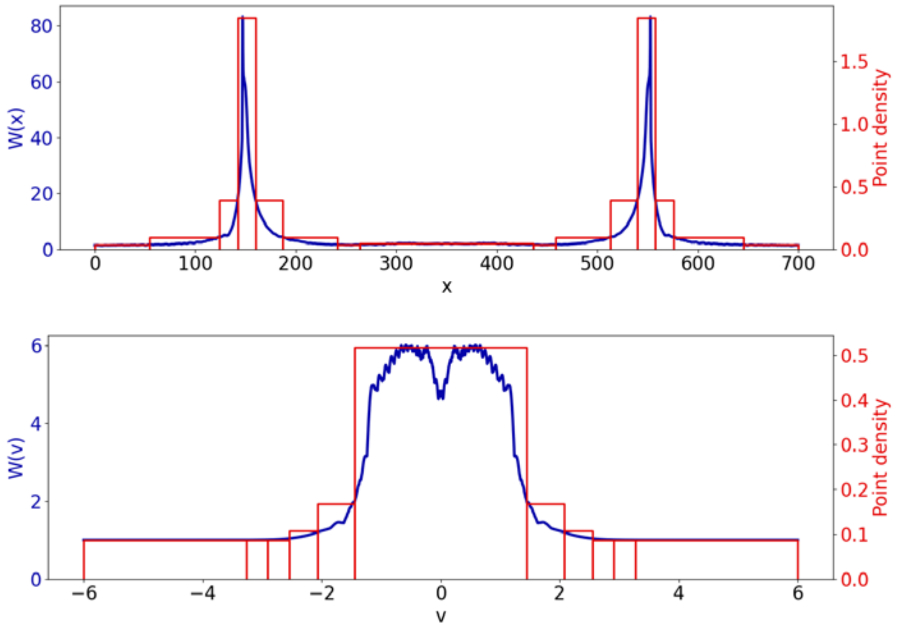
$$\sum_i b_j(x_i)q_i = \int b_j(x)dx \qquad \forall\, j$$

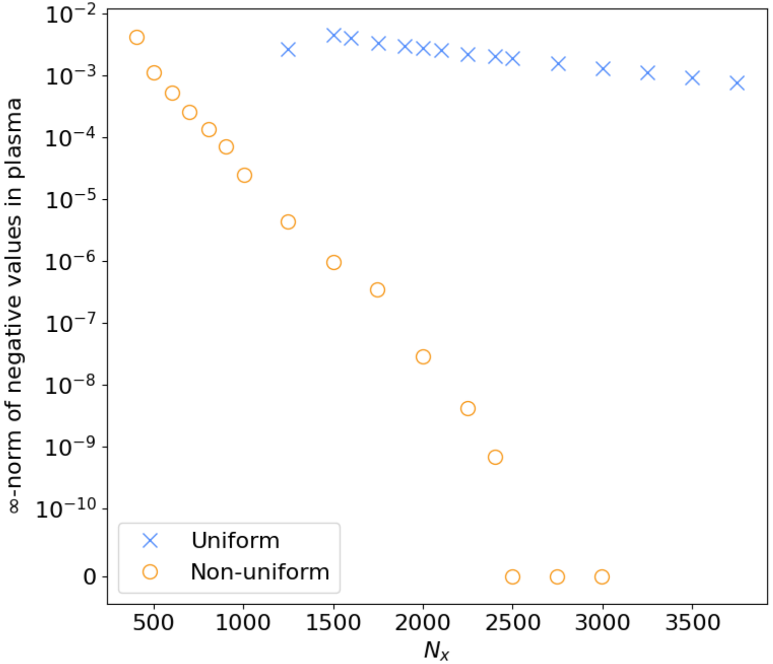
- Boundary conditions must be chosen such that no additional values are required
  - Greville interpolation points with knots at boundary
    - Same number of interpolation points and basis splines
  - Natural boundary condition = Schoenberg's "best" quadrature
    (well conditioned calculation described in *[Bourne et al, submitted 2022]*)
    - Largest derivatives set to 0 at boundaries
    - Quadrature order is equal to number of free derivatives at boundary

- Penalisation confines the plasma
- Lack of refinement leads to negative values which are problematic for physical interpretations
- Steep gradients at wall
- Tends towards an equilibrium *[Y. Munschy et al, submitted 2022]*
- Wider simulations would be more realistic but require even more points

- Points are spaced proportionally to a weight function:

$$W(x) = \sqrt{1 + \alpha^2 \left(\max_{t,v} \ \partial_x f(t,x,v) \right)^2}$$





[1]E. Bourne, Y. Munschy et al, submitted *2022*

Non-Uniform Splines reach better conservation errors faster

$$n_{s(t,x)} = \int f_s(t,x,v_s)dv_s \qquad \Pi_{s(t,x)} = \int v_s^2 f_s(t,x,v_s)dv_s$$

$$\Gamma_{s(t,x)} = \int v_s f_s(t,x,v_s)dv_s \qquad Q_{s(t,x)} = \int \frac{1}{2} v_s^3 f_s(t,x,v_s)dv_s$$

- Density

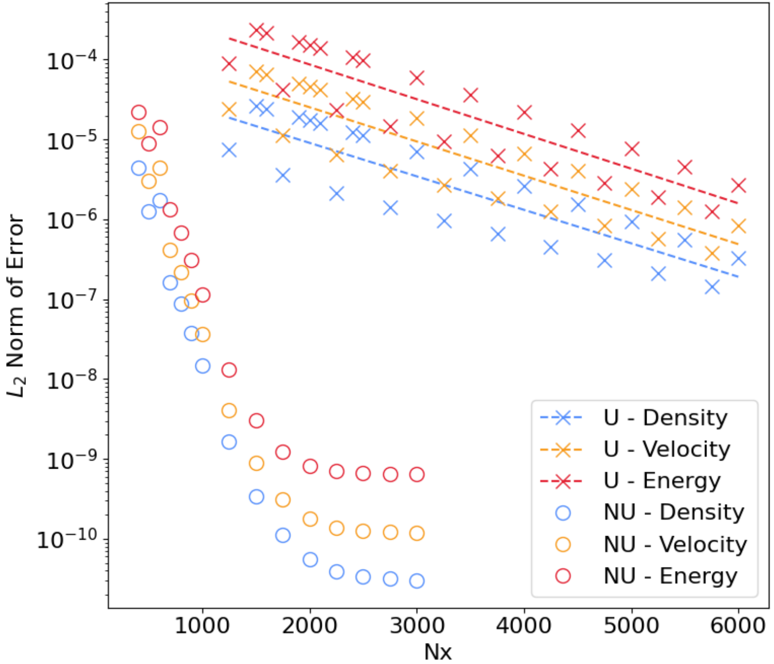
$$\partial_t n_s(t,x) + \frac{1}{\sqrt{m_s}} \partial_x \Gamma_s(t,x) = \int S_s(t,x,v_s) + C_{ss}(t,x,v_s)dv_s$$

- Velocity

$$\partial_t \Gamma_s(t,x) + \frac{1}{\sqrt{m_s}}\left(\partial_x \Pi_s(t,x) + q_s \partial_x \phi(t,x) n_s(t,x)\right) = \int v_s S_{s(t,x,v_s)} + v_s C_{ss}(t,x,v_s)dv_s$$

- Energy

$$\partial_t \Pi_{s(t,x)} + 2\frac{1}{\sqrt{m_s}}\left(\partial_x Q_{s(t,x)} + q_s \partial_x \phi(t,x)\Gamma_s\right) = \int v_s^2 S_{s(t,x,v_s)} + v_s^2 C_{ss}(t,x,v_s)dv_s$$

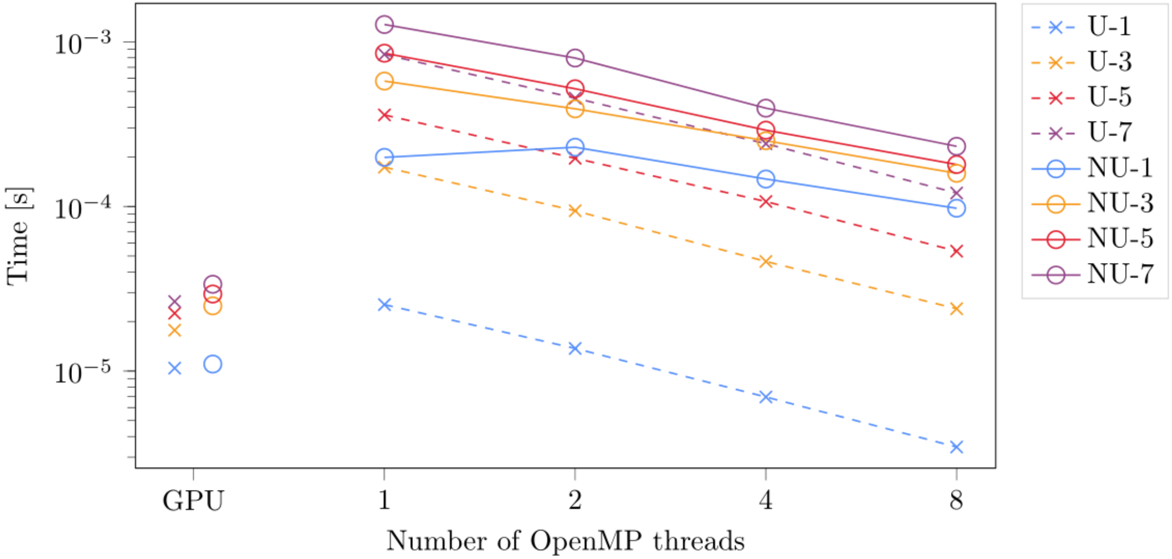


[1]E. Bourne, Y. Munschy et al, submitted *2022*

GPUs are crucial to make non-uniform splines competitive

- Backward semi-Lagrangian advection on non-uniform splines is slower than uniform splines
- The cost difference is much smaller on a GPU

(U-1 = Uniform splines of degree 1)



[1]E. Bourne, Y. Munschy et al, submitted *2022*

- Non-uniform knots reduce the memory requirements by 89%

- Non-uniform simulations run 5.5 times faster than uniform simulations providing equivalent results on GPUs

- Non-uniform points allow simulations on much larger domains with more realistic (steeper) walls

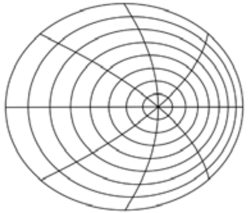
[1] E. Bourne, Y. Munschy et al, submitted *2022*

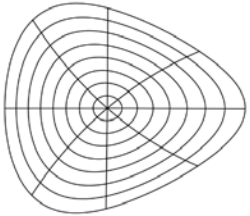
FDM/FFT

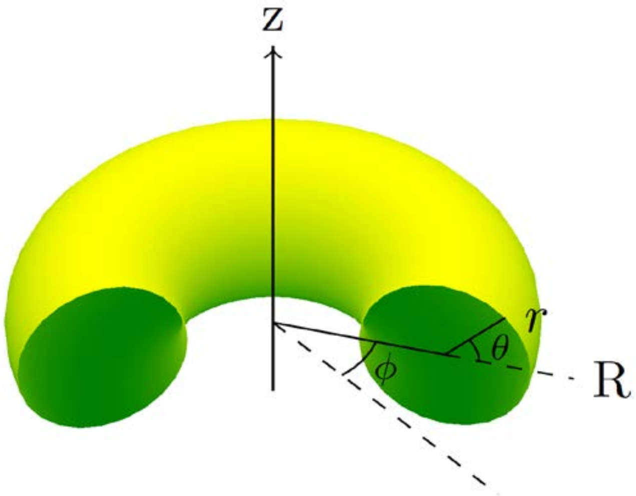
- Solves the equation:

✓ $$\frac{e}{T_{e,eq}}(\phi - \langle\phi\rangle) - \frac{1}{n_{e0}}\sum_s Z_s \nabla_\perp \cdot \left(\frac{n_{s,eq}}{B\Omega_s}\nabla_\perp \phi\right) = \frac{1}{n_{e0}}\sum_s Z_s \int J_0 \cdot (F_s - F_{s,eq}) d^3v$$

- Handles complex geometry

✘ 





! Easy refinement

✓ Fast

✓ Good parallelisation

✓ Accurate

✓ Low memory footprint

! Good representation of the boundary

Spline FEM Solver (IPP)

[Zoni, Güçlü 2019]

- Finite Element Method
- Conjugate Gradient solver
- Curvilinear coordinates
- Arbitrary order
- C^1 polar splines

Embedded Boundary Solver (IPP)

[Zhang, Weiqun 2019]

- Finite Volume Method
- Multi-grid solver
- Cartesian coordinates
- Second order
- Uses AMReX library

GmgPolar Solver (CERFACS)

[Kühn, Kruse 2022]

- Finite Difference Method
- Multi-grid solver
- Curvilinear coordinates
- ~4^{th} order
- Matrix free

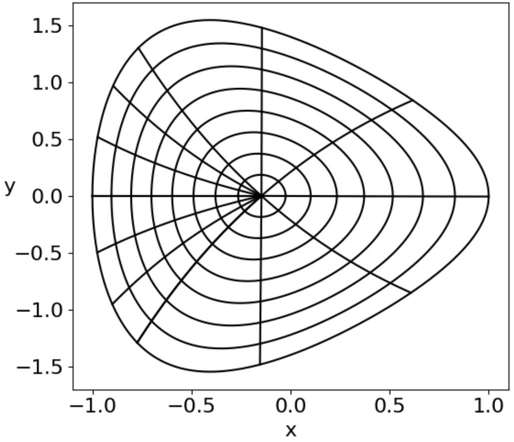
Common Problem

$$-\frac{1}{n_{e_0}}\sum_s Z_s \nabla_\perp \cdot \left(\frac{n_{s,eq}}{B\Omega_s}\nabla_\perp \phi\right) + \frac{e}{T_{e,eq}}(\phi - \langle\phi\rangle)$$

$$= \frac{1}{n_{e_0}}\sum_s Z_s \int J_0 \cdot (\bar{F}_s - \bar{F}_{s,eq}) d^3 v$$

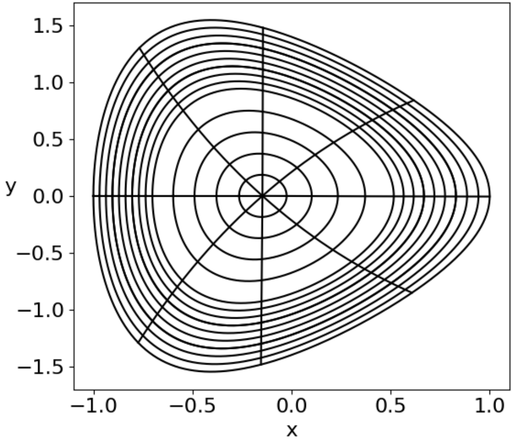
$$\langle\phi\rangle = \frac{\int \phi R J d\theta d\phi}{\int RJ d\theta d\phi}$$

[1]E. Bourne, P. Leleux et al, submitted *2022*



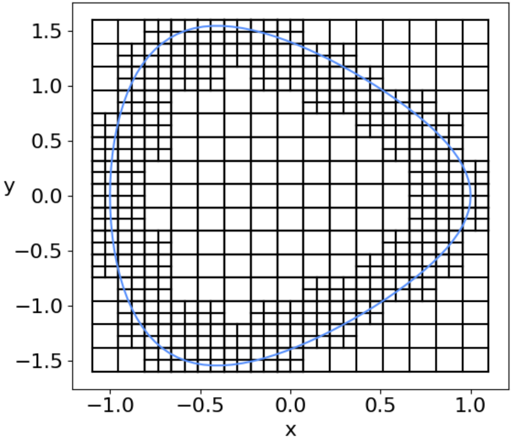
Poloidal refinement
(Spline FEM / GmgPolar)

+ Handles anisotropy



Radial refinement
(Spline FEM / GmgPolar)

+ Easy to refine in target area
- Doesn't handle poloidally small objects



Refinement with Patches
(Embedded Boundary)

+ Good for handling small objects
- Difficult to find required patches
- Lots of patches to handle

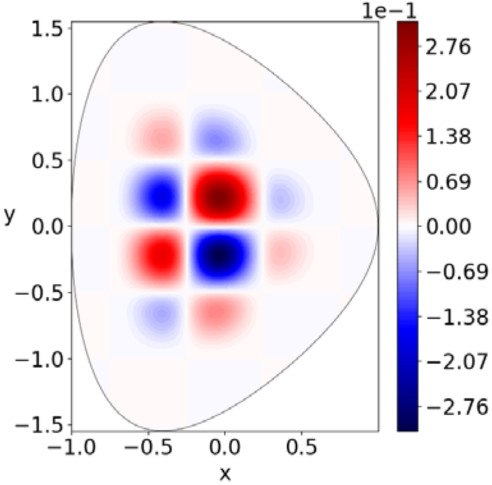
[1]E. Bourne, P. Leleux et al, submitted *2022*

$$-\nabla \cdot (\alpha \nabla u) + \beta u = f$$

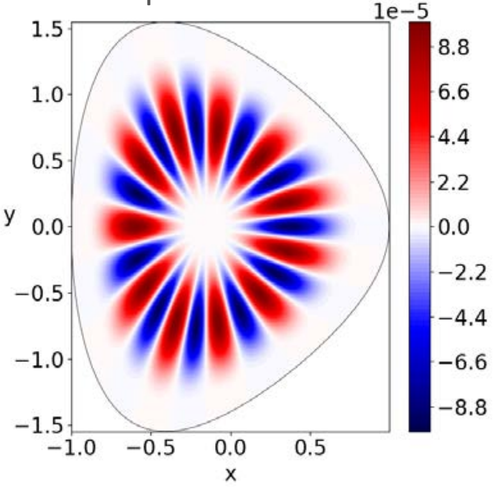
$$\alpha(r) = \exp\left(-\tanh\left(\frac{r - r_p}{\delta_r}\right)\right)$$

$$\beta(r) = \exp\left(\tanh\left(\frac{r - r_p}{\delta_r}\right)\right)$$

Cartesian solution not aligned with coordinate system

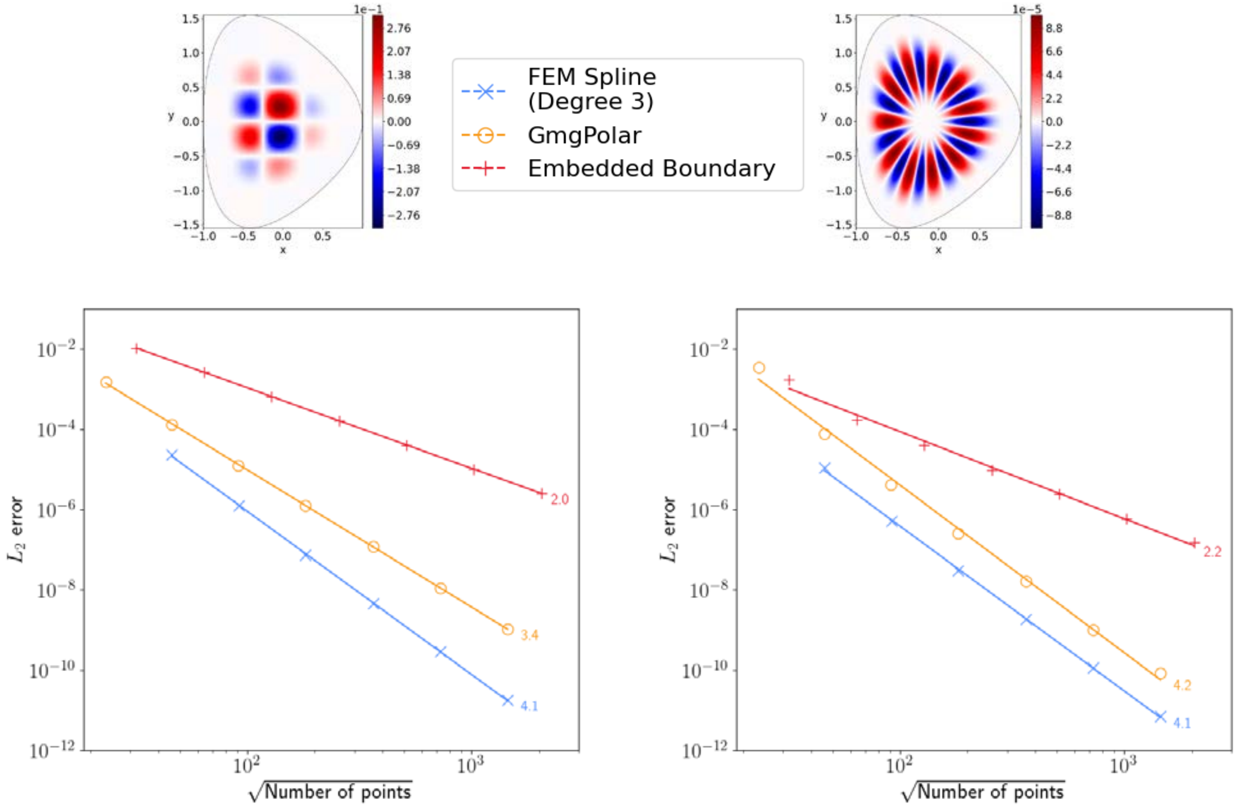


Solution aligned with curvilinear coordinates (used as initial perturbation in GYSELA)



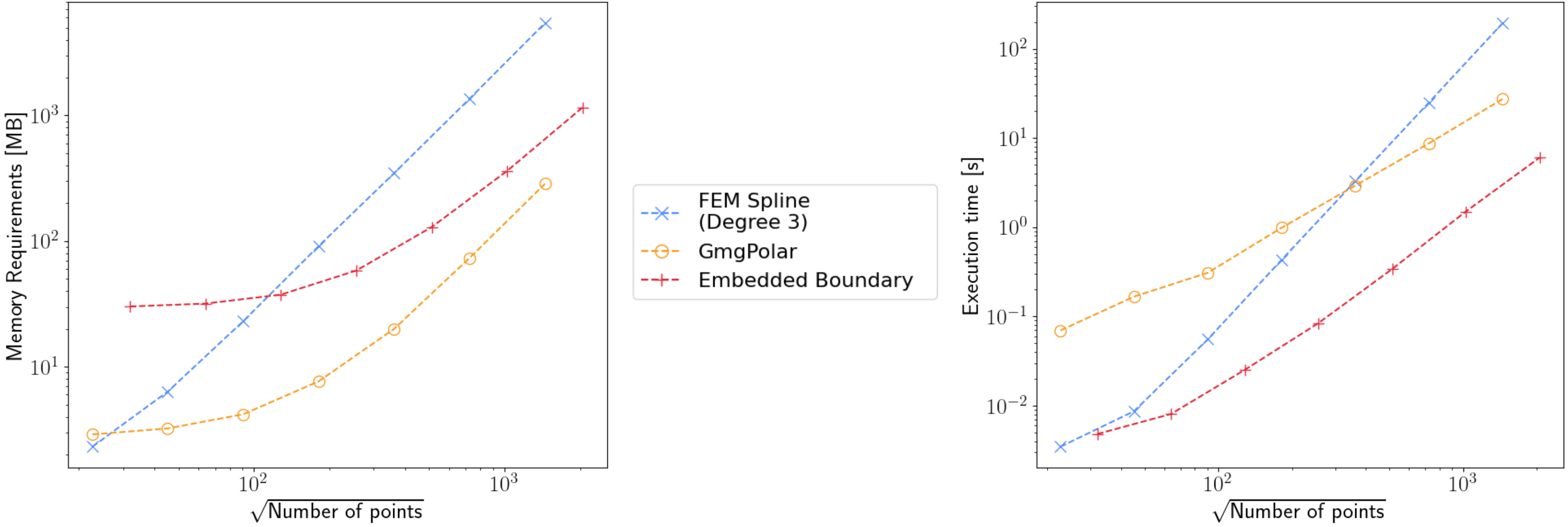
[1]E. Bourne, P. Leleux et al, submitted *2022*

- Spline FEM solver is most accurate
- The singular point is a source of imprecision for the GmgPolar solver
- The lower order of the Embedded Boundary solver makes it much less accurate

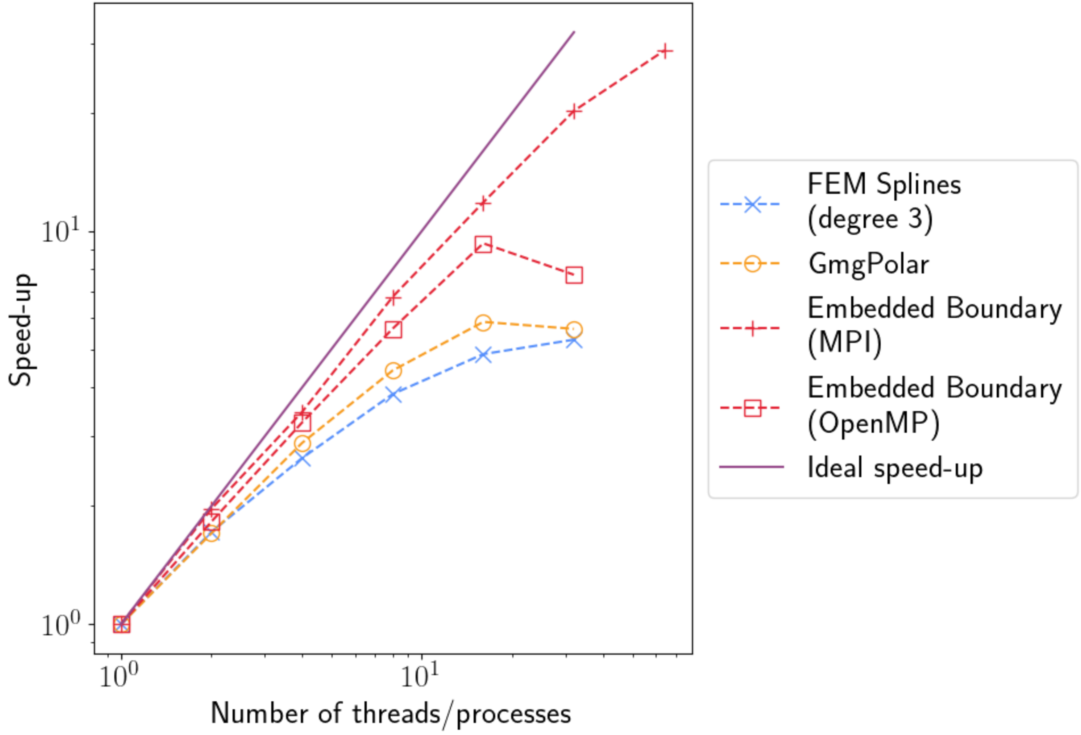


[1]E. Bourne, P. Leleux et al, submitted *2022*

AMReX is highly optimised but matrix-free GmgPolar has low memory



[1]E. Bourne, P. Leleux et al, submitted *2022*



[1]E. Bourne, P. Leleux et al, submitted *2022*

Spline FEM Solver (IPP)

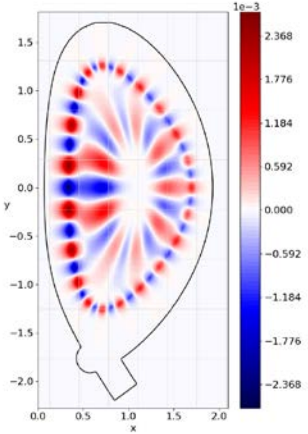
- \+ Highly accurate
- – Weaker performance for large problems
- – Does not handle the flux-surface average
- – Refinement must preserve tensor-product mesh

Embedded Boundary Solver (IPP)

- – Not very accurate
- – Does not handle the flux-surface average
- – Difficult refinement
- \+ Good performance
- \+ Possible to solve problems with complex boundary conditions

GmgPolar Solver (CERFACS)

- \+ Good accuracy
- \+ Good performance
- – Does not handle the flux-surface average
- – Refinement must preserve tensor-product mesh

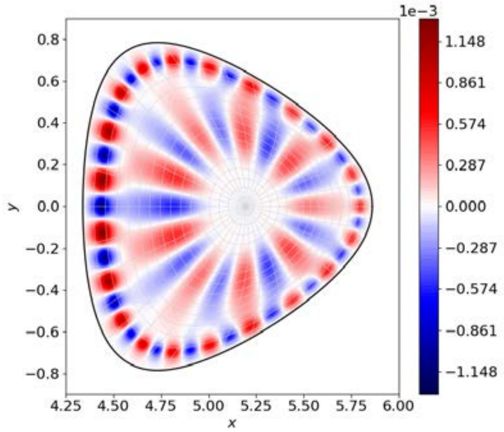


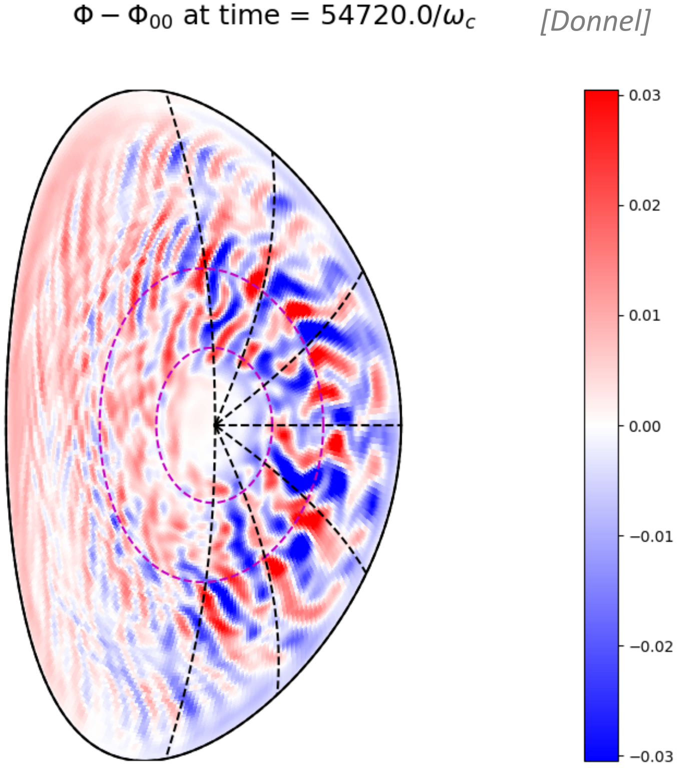
[1]E. Bourne, P. Leleux et al, submitted *2022*

- Culham Geometry has been implemented in GYSELA

$$x(r,\theta) = r\cos\theta - E(r)\cos\theta + T(r)\cos 2\theta - P(r)\cos\theta + \delta(r) + R_0$$
$$y(r,\theta) = r\sin\theta + E(r)\sin\theta - T(r)\sin 2\theta - P(r)\sin\theta$$

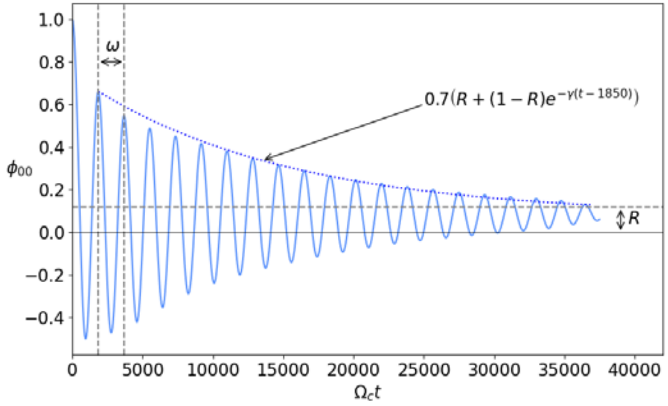
- The Spline FEM Solver has also been implemented using an iterative method to handle the flux-surface average

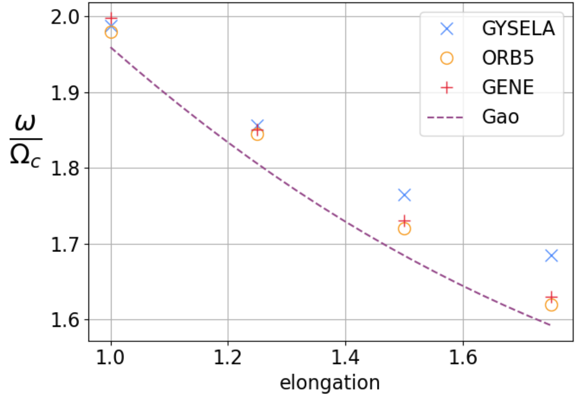




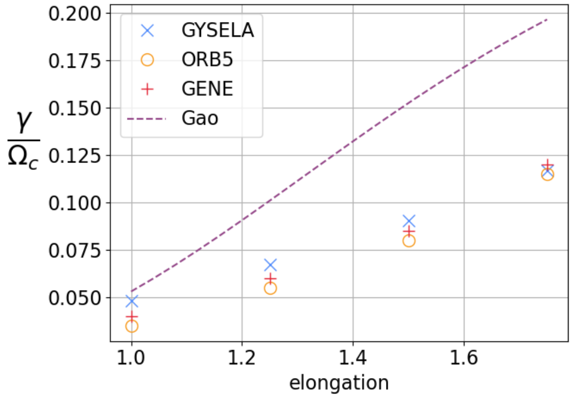
[Donnel]

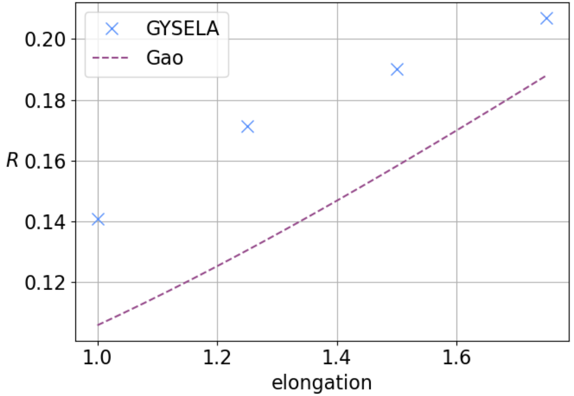
Geodesic Acoustic Mode (GAM) test





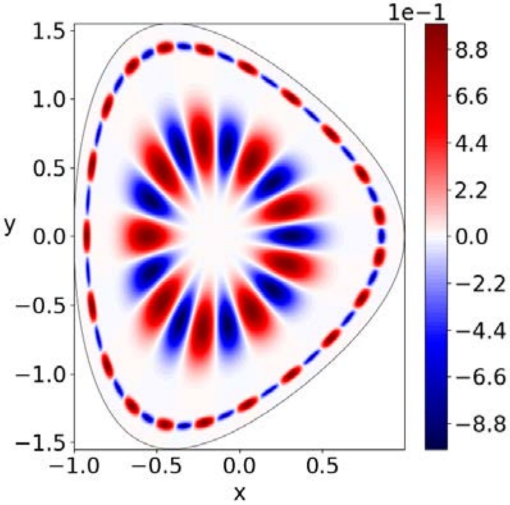
- Results agree with other codes and follow trend predicted by theory

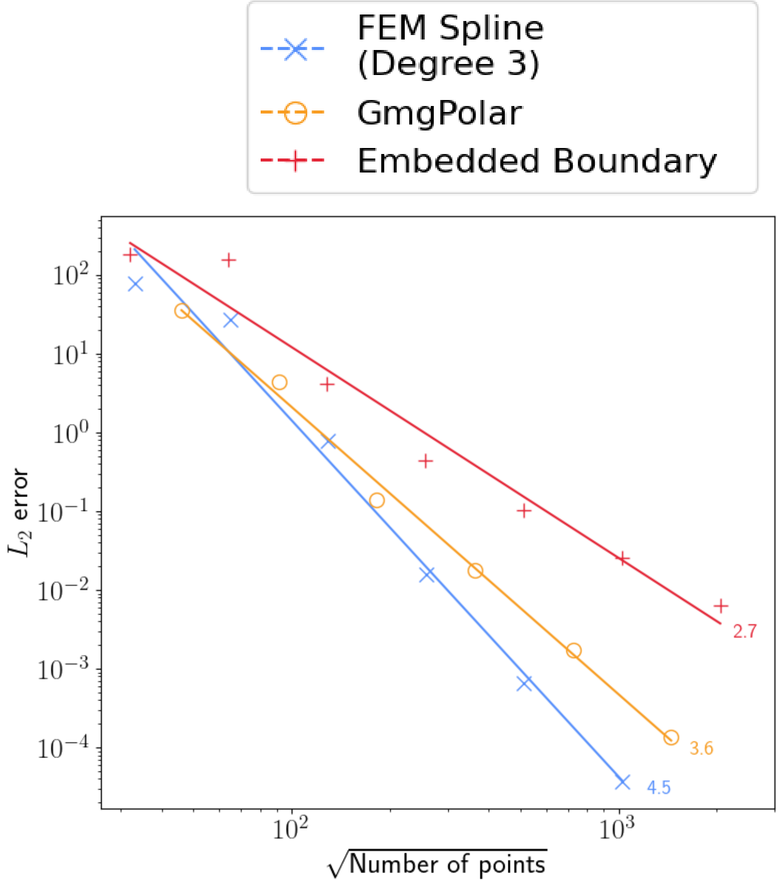




Thank you for your attention

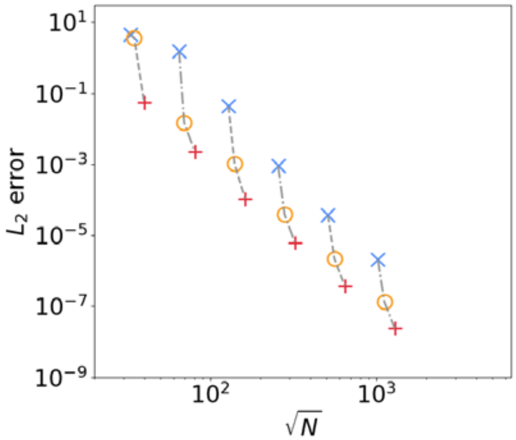
- At least 500^2 points are required for an error. But using more than 2000^2 points is too costly for the machine where the tests were run



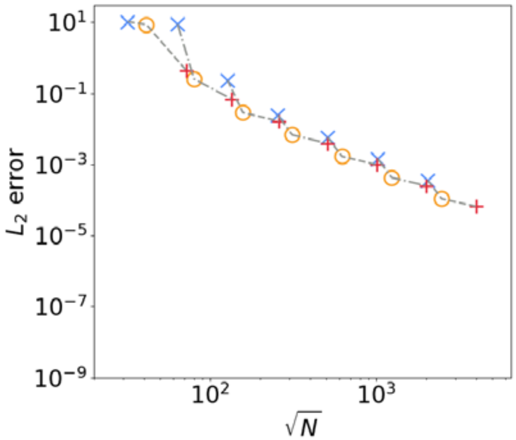


- Lack of poloidal refinement in the poloidal direction hampers Spline FEM and GmgPolar convergence
- Overlarge patches and low convergence order hampers Embedded Boundary convergence

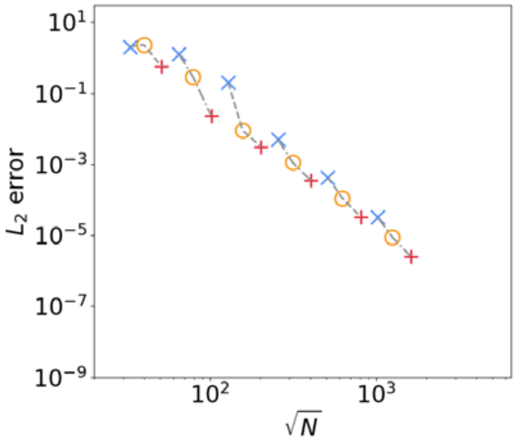
Spline FEM Solver



Embedded Boundary Solver

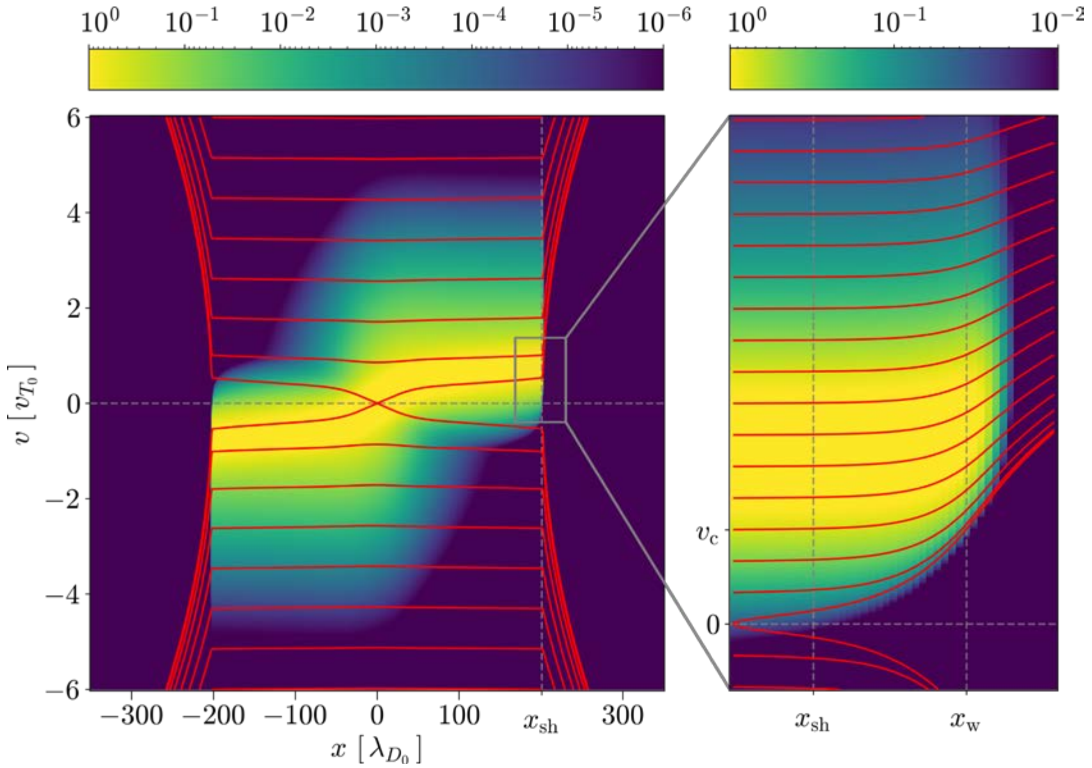


GmgPolar Solver



× No refinement ○ 1 refinement + 2 refinements

- VOICE equilibrium for ions
- Red lines show the trajectory of the particles
  - Trajectories in the wall are nonsensical as all ions in the wall are absorbed



FEM Spline Solver is Performant if the Required Error Determines the Number of Points