Numerical and parallel computing challenges for the global full-$f$ semi-lagrangian code

GYSELA

V. Grandgirard, P. Angelino, N. Crouseilles†, G. Dif-Pradalier, X. Garbet, Ph. Ghendrih, M. Haefele†, G. Latu†, Ch. Passeron, Y. Sarazin, E. Sonnendrücker†

CEA, IRFM, F-13108 Saint-Paul-Lez-Durance.

†(INRIA-Calvi), IRMA-Univ. Louis Pasteur, F-67084 Strasbourg.
Gyrokinetic theory

Phase space in 6D

- 3D in space: $\langle r, \theta, \varphi \rangle$
- 3D in velocity: $(v_\parallel, v_\perp, \alpha)$

$$\omega_{\text{turb}} \approx 10^5 \text{s}^{-1} \ll \dot{\varphi}_c \equiv \omega_c \approx 5 \times 10^8 \text{s}^{-1}$$

- gyroaverage essential
- adiabatic invariant $\mu = \frac{mv_\perp^2}{2B} \rightarrow \mu$ as a parameter

4D+1D gyrokinetic problem $f(r, \theta, \varphi, v_\parallel, \mu)$

↓

Numerical complexity: Conservation properties of the scheme

Memory(5D) + CPU time $\rightarrow$ massively parallel
Why can’t we avoid parallelisation?

Simple question for developers but the theoreticians have not always an idea of the memory size and the CPU time required.

Memory size:

- Example in 4D: ”small” mesh = \((128 \times 128 \times 128 \times 128)\)
  - \(~\) 280 millions of mesh points
  - Already 2 Gbytes just for one distribution function in 4D

So in 5D: with only 4 points in the 5\(^{th}\) direction → not possible on actual PC.

CPU time:

- Typical simulations → 5 days on 1000 processors
  - \(\equiv 120.000\) hours / monoprocessor → more than 13 years on 1 processor
Purpose of the talk

An idea of what are the numerical and parallel computing challenges for the development of semi-lagrangian code

- There is a gap between "academic" cases you can run on your laptop in 5 minutes and 5D simulations which require more than 1000 processors during several weeks.

- It requires a strong collaboration between physicists, mathematicians
  - Numerical methods adapted to your physical problem,
- But also a strong collaboration with specialists of parallel computing.
  - Performant algorithms for improving the parallelisation,
  - Parallelisation can constrainst your physical problem.
Challenges of physics for GYSELA code
GYrokinetic SEMi-LAGRangian code

- **Full-\(f\) code** (\(\neq \delta f\)):
  - Equilibrium & fluctuations: no scale separation assumption
  - Non-Maxwellian background equilibrium
    - Careful calculation of \(E_r\) (canonical initial equilibrium)
      - [Idomura '03, Angelino '06, Dif-Pradalier '07]
  - Neoclassical theory
    - The full \(f\) must relax \(\rightarrow f_{\text{Maxwell}}\)

- **Global code** (\(\neq\) flux-tube):
  - Large scale fast transport events
  - Profile relaxation \(\rightarrow\) steady state \(\rightarrow\) flux-driven conditions
  - Numerical challenge

- **Semi-lagrangian scheme** (mixed between PIC and eulerian)

---

Virginie GRANDGIRARD

Vienna - September 17, 2008
The **GYSELA code** – hypotheses

i. Electrostatic approximation \( \mathbf{E} = -\nabla \phi \);  

ii. Axisymmetric (\( \partial \phi = 0 \)), concentric and circular nested magnetic flux surfaces;  

iii. Ion distribution function;  

iv. Adiabatic electrons;  

v. Low \( \beta \) approximation  

\[
\frac{\nabla N}{R} \approx \frac{\nabla B}{B};
\]

vi. \( \mathbf{B}^* \) at first order in \( \rho^* \);  

vii. Bessel function \( J_0(k_{\perp} \rho_i) \approx \frac{1}{1 + \frac{k_{\perp}^2 \rho_i^2}{4}} \) (Padé).
The equations –consistent up to the first order in \( \rho_\star \)

**A. Gyrokinetic equation (4D + 1D)**

\[
\partial_t \bar{f} + \mathbf{v}_E \cdot \nabla_\perp \bar{f} + \mathbf{v}_D \cdot \nabla_\perp \bar{f} + \mathbf{v}_\parallel \nabla_\parallel \bar{f} + \dot{v}_\parallel \partial_{v_\parallel} \bar{f} = 0 \quad \text{or} \quad C(f)
\]

\[
\mathbf{v}_E = \frac{\mathbf{B} \times \nabla \bar{\phi}}{B^2} \quad \dot{v}_\parallel = \frac{dv_\parallel}{dt} = -\frac{e}{m} \nabla_\parallel \bar{\phi} - \frac{\mu}{m} \nabla_\parallel \mathbf{B} + \frac{m v_\parallel}{B} \mathbf{v}_E \cdot \nabla \mathbf{B}
\]

\[
\mathbf{v}_D = \frac{m_i v_\parallel^2 + \mu B}{eB} \frac{\mathbf{B} \times \nabla \mathbf{B}}{B^2} \quad \nabla_\parallel = \frac{1}{R} \left( \partial_\varphi + \frac{1}{q(r)} \right)
\]

**B. Quasi-neutrality equation (3D)**

\[
\delta n_e = \delta n_i \equiv n - n_{init}
\]

\[
-\frac{1}{n_0(r)} \nabla_\perp \cdot \left[ \frac{n_0}{B \omega_c} \nabla_\perp \phi \right] + \frac{e}{T_e(r)} (\phi - \langle \phi \rangle) = \frac{2\pi B}{mn_0} \int d\mu \ dv_\parallel J_0 \left( \bar{f} - f_{init} \right)
\]

\( \langle \cdot \rangle \equiv \text{flux surface average} \)
The **GySELA code** – boundary conditions

- Periodic in $\theta$ and $\varphi$
- Vanishing perturbations in non-periodic $r$ and $v_\parallel$
- "Source": thermal baths
  - profile relaxation

**Alternatives to sustain the mean gradient:**

- adaptative source at each radial position [GYRO ; GTC ; ...]
- diffusive buffer zones [ORB5, GySELA]
- realistic flux-driven: controled input heat flux [GySELA, in progress]
A thermal bath to sustain the mean gradient

\[ R / L_T \]

\[ \text{Time} \times \omega_c \]

- Blue line: \( D=0 \); \( \nu=0 \)
- Green line: \( D=0.01; \nu=0 \)
- Red line: \( D=0.01; \nu=0.005 \)
Challenges in numerics and parallel computing for a Semi-Lagrangian scheme

Virginie Grandgirard
Vienna - September 17, 2008
Semi-Lagrangian basic concepts

1. Vlasov equation
   - f constant along the trajectories
   - \( f(X_{\text{node}}, t+\Delta t) = f(X^*, t) \)

2. Equations of motion

3. Cubic spline interpolation

[Sonnendrücker '98]

😊 Good property of energy conservation

[Grandgirard '05, for 4D-drift kinetic ITG]

😡 A supplementary step: INTERPOLATION

Virginie Grandgirard  Vienna - September 17, 2008
Parallelisation of a Semi-Lagrangian method

**Advantage** *(due to the eulerian aspect)*:  
- fixed grid ➔ perfect load balancing

**Drawback** *(due to interpolation)*:  
- Several choices for the interpolation  
- But we use cubic splines interpolation:  
  - ☀️ Good compromise between accuracy and simplicity  
  - ☹️ Loss of locality  
    (value of $f$ on one grid point requires $f$ over the whole grid)

- Not possible to use a simple domain decomposition
New approach: Local cubic splines

A new numerical tool has been developed:

» Hermite Spline interpolation on patches

[Latu-Crouseilles '07]

Computational domain decomposed in subdomains

Definition of local splines on each subdomains with Hermite boundary conditions

Derivatives are defined so that they match as closely as possible those of global splines

Boundary adapted conditions for $C^1$ reconstruction, including patch boundaries
Parallelisation in GYSELA – good performance

\{MPI + OpenMP\} parallelisation

5D mesh = (512 × 128 × 256 × 16 × 32) = 8.6 billions of points

Efficiency of 93% sur 2048 processor
Efficiency of 83% on 4096 processors
How can the parallelisation constrainst the physics?

- Choice of the coordinate system
- Choice of the collision operator
Distribution of the 5D mesh

- $\mu =$ adiabatic invariant
  - $\rightarrow$ one $\mu$ per processor $\Rightarrow$ very performing in $//$

- Each processor is solving separately the vlasov equation

- Only communications for $\int d\mu$ (RHS of the Poisson equation)

- $(r, \theta)$ cross-section with the most important discretisation
  - $\leftarrow$ 2D domain decomposition in $(r, \theta)$ (2D local splines)

- sequential distribution in $\varphi$ and $v_\parallel$ directions
Local splines not completely adapted to \((r, \theta, \varphi)\)

😊 Efficient parallelisation but only if you minimize the number of
points in buffer regions

\[ \text{Minimize the size of the messages to transfer} \]

😊 It imposes a CFL condition (displacement of no more than one
cell)

\[ \text{As } \frac{d\theta}{dt} \gg \frac{dr}{dt} \rightarrow \text{need to decrease drastically the time step} \]

▷ In practice, local spline just in radial direction \(\rightarrow\) strong
constraint for memory size per node

**Idea:** Change the coordinates system to separate low
and fast variables (field aligned coordinates).
Collision operator on $\mathbf{v}_\parallel$ only

- If you want to keep an efficient parallelisation in $\mu$
  - no differential operator in $\mu$
- A strong constraint in the choice of the collision operator

Try to find the best compromise: mini. model $\leftrightarrow$ max. physics? exact NC transport at low computational cost [Garbet, Varenna ’08]

Lorentz operator on $\mathbf{v}_\parallel$ only:

$$\frac{d}{dt} f = \partial_{\mathbf{v}_\parallel} \left( D \partial_{\mathbf{v}_\parallel} f - \mathbf{V} f \right)$$

Goal: interaction gyrokinetic turb. $\leftrightarrow$ neoclassical theory [Dif-Pradalier, Varenna ’08]
To conclude ➤ the road is long...

Example: CPU time and memory size for a $\rho^* = \rho_i/a$ scaling

Numerical problem: $\rho^*/2 \rightarrow \text{mesh } \times 2^3 + \text{time step } /2$

<table>
<thead>
<tr>
<th></th>
<th>$\rho^* = 1/128$</th>
<th>$\rho^* = 1/256$</th>
<th>$\rho^* = 1/512$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(tore)</td>
<td>(1/2-tore)</td>
<td>(1/8-tore)</td>
</tr>
<tr>
<td>Mesh (Nr x Ntheta x Nphi x 32 x 8)</td>
<td>128x256x128</td>
<td>256x512x128</td>
<td>512x1024x64</td>
</tr>
<tr>
<td>Number of mesh points</td>
<td>1.15 billions</td>
<td>4.3 billions</td>
<td>8.6 billions</td>
</tr>
<tr>
<td>Number of processors (8 threads at each time)</td>
<td>256</td>
<td>512</td>
<td>512</td>
</tr>
<tr>
<td>Memory required per node</td>
<td>1.94 Go</td>
<td>4.5 Go</td>
<td>9.3 Go</td>
</tr>
<tr>
<td>Total CPU time</td>
<td>14 h</td>
<td>62 h (2.5 days)</td>
<td>255 h (10.5 days)</td>
</tr>
<tr>
<td>Number of monoprocessor hours</td>
<td>3.600 h</td>
<td>32.000 h</td>
<td>130.500 h</td>
</tr>
<tr>
<td>Memory for saving</td>
<td>400 GBytes</td>
<td>3 TBytes</td>
<td>12 TBytes</td>
</tr>
</tbody>
</table>

Difficult to simulate a complete ITER tore with the actual HPCs.