# Comparison of high-order Eulerian methods for electron hybrid model

A. Crestetto [1]  N. Crouseilles [2,3]  Y. Li [4]  J. Massot [3,2]

[1]LMJL, Université de Nantes

[2]Inria Rennes – Bretagne Atlantique

[3]IRMAR, Université de Rennes

[4]Max Planck Institute for Plasma Physics, Garching, Germany

November 9, 2021

# Outline


1. Introduction
2. Numerical methods
3. Application for hybrid Vlasov-Maxwell model
   - With splitting method
   - With Lawson method
4. Numerical results
5. Conclusion

# Outline


1. Introduction
2. Numerical methods
3. Application for hybrid Vlasov-Maxwell model
   - With splitting method
   - With Lawson method
4. Numerical results
5. Conclusion

# Vlasov-Maxwell $1dz - 3dv$ model

Transport of electron density distribution $f = f(t, z, \boldsymbol{v})$,
$\boldsymbol{B}(t, z) = (B_x, B_y, 0)(t, z), \boldsymbol{E}(t, z) = (E_x, E_y, 0)(t, z) \in \mathbb{R}^2$,
$z \in [0, 2\pi]$, $\boldsymbol{B}_0 = (0, 0, B_0)^\top$, $\boldsymbol{v} \in \mathbb{R}^3$, $v_\perp = (v_x, v_y)^\top \in \mathbb{R}^2$:

$$\begin{cases} \partial_t f + v_z \partial_z f - (\boldsymbol{E} + \boldsymbol{v} \times (\boldsymbol{B} + \boldsymbol{B}_0)) \cdot \nabla_{\boldsymbol{v}} f = 0 \\ \partial_t \boldsymbol{B} = -\partial_z \boldsymbol{E} \\ \partial_t \boldsymbol{E} = \partial_z \boldsymbol{B} + \int_{\mathbb{R}^3} v_\perp f \, \mathrm{d}\boldsymbol{v} \end{cases}$$

**Motivation:**

- We consider an initial condition of the form $f = f_c + f_h$ with:
  $f_c(t = 0, z, \boldsymbol{v}) = \rho_c(t, z) \delta_{\boldsymbol{v} = \boldsymbol{u}_c(t,z)}(\boldsymbol{v})$
- We want high order methods in $(z, \boldsymbol{v})$
  - FFT in $z$ + WENO in $\boldsymbol{v}$
- We want high order methods in time $t$
  - splitting method vs exponential integrator

# The idea

- Grid methods can't have an initial condition like:
  $f_0(z, \boldsymbol{v}) = \rho_{c,0}(z)\delta_{\boldsymbol{v}-\boldsymbol{u}_c}(\boldsymbol{v}) + f_{h,0}(z, \boldsymbol{v})$
- Idea is to derive an hybrid model:
  - *Cold plasma approximation*: $\frac{T_c}{T_h} \ll 1 \rightarrow \cancel{f_c(t, z, \boldsymbol{v})} \rightarrow \boldsymbol{j}_c(t, z)$
    - Fluid dynamic for cold particles (no velocity grid)
  - Hypothesis on hot particles: $\int_{\mathbb{R}^3} f_h(t, z, \boldsymbol{v}) \, \mathrm{d}\boldsymbol{v} \ll \rho_c(t, z)$
    - Kinetic dynamic for hot particles

➜ Split $f = f_c + f_h$ + Compute momentum of $f_c$ with *cold plasma approximation* + Linearize the model

Holderied et al. 2020, *Journal of Computational Physics*

# Linearized hybrid Vlasov-Maxwell $1dz - 3dv$ model

The new model: a nonlinear transport in $(z, v_x, v_y, v_z) \in \Omega \times \mathbb{R}^3$ of:

- a cold (**fluid**) electron density distribution, reconstruction from current variable $\boldsymbol{j}_c(t,z) = q_e \rho_c(t,z) \boldsymbol{u}_c(t,z) = (j_{c,x}, j_{c,y}, 0)(t,z)$
- a hot (**kinetic**) electron density distribution $f_h(t, z, \boldsymbol{v})$

$$
\begin{cases}
\partial_t \boldsymbol{j}_c = \Omega_{pe}^2 \boldsymbol{E} - J \boldsymbol{j}_c B_0 \\
\partial_t \boldsymbol{B} = J \partial_z \boldsymbol{E} \\
\partial_t \boldsymbol{E} = -J \partial_z \boldsymbol{B} - \boldsymbol{j}_c + \int_{\mathbb{R}^3} v_\perp f_h \,\mathrm{d}\boldsymbol{v} \\
\partial_t f_h + v_z \partial_z f_h - (\boldsymbol{E} + \boldsymbol{v} \times (\boldsymbol{B} + \boldsymbol{B}_0)) \cdot \nabla_{\boldsymbol{v}} f_h = 0
\end{cases}
$$

with:

$$
J = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}
$$

# Convergence when $T_c \to 0$

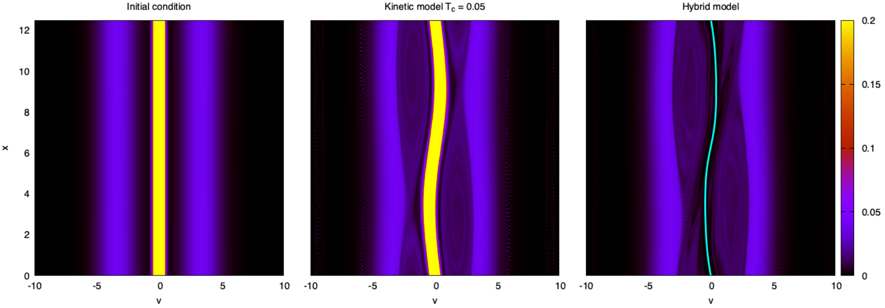


Figure: Simulation of initial condition (left) with kinetic model with $T_c = 0.05$ (middle) and hybrid model (right) to the time $T_f = 200$

✔ Good agreement between kinetic ($f$) model and hybrid model ($f_h + u_c$)

# Outline


1. Introduction
2. Numerical methods
3. Application for hybrid Vlasov-Maxwell model
   - With splitting method
   - With Lawson method
4. Numerical results
5. Conclusion

# Numerical methods

Two time integrators to compute a numerical solution of:

$$\dot{u} = L(t,u) + N(t,u), \quad u(0) = u_0$$

$u \in \mathbb{R}^d$, $L$ and $N$ functions $(t,u) \in \mathbb{R}_+ \times \mathbb{R}^d \mapsto \mathbb{R}^d$, $d \in \mathbb{N}$.

- Splitting method (Lie, Strang, Suzuki)
- Lawson method (LRK(4,4), LDP4(3))

In space $z$: we use a Fourier transform (FFT).
In velocity $\boldsymbol{v}$: we use WENO5 or Lagrange 5.

Successive resolution of:

$$\dot{u} = L(t,u) \qquad \rightarrow \tilde{u}_t = \varphi_t^{[L]}(u_0)$$

$$\dot{u} = N(t,u) \qquad \rightarrow \tilde{u}_t = \varphi_t^{[N]}(u_0)$$

Solution at time $t$:

**Lie:** order 1 method, composition of sub-steps:
$\varphi_t(u_0) \approx \varphi_t^{[L]} \circ \varphi_t^{[N]}(u_0)$

**Strang:** order 2 method: $\varphi_t(u_0) \approx \mathcal{S}_t(u_0) = \varphi_{t/2}^{[L]} \circ \varphi_t^{[N]} \circ \varphi_{t/2}^{[L]}(u_0)$

Strang 1968, *SIAM Journal on Numerical Analysis*

**Suzuki:** order 4 method, composition of 5 Strang methods:

$$\varphi_t(u_0) \approx \mathcal{S}_{\alpha_1 t} \circ \mathcal{S}_{\alpha_2 t} \circ \mathcal{S}_{\alpha_3 t} \circ \mathcal{S}_{\alpha_2 t} \circ \mathcal{S}_{\alpha_1 t}(u_0)$$

with: $\alpha_1 = \alpha_2 = \frac{1}{4-\sqrt[3]{4}}$ and $\alpha_3 = \frac{1}{1-4^{\frac{2}{3}}}$

Suzuki 1990, *Physics Letters A*

Casas and Escorihuela-Tomàs 2020, *Mathematics* (for some higher order methods)

# Splitting method

## Pros & Cons

- ✔ Good behavior in long time
- ✔ Error in time only depends on splitting method
- ✔ Split a difficult problem into small easier sub-problems
- ✗ Numerical cost for high order method
- $\sim$ Needs to find a way to solve exactly **in time** each step

# Lawson method

$$\partial_t u = Lu + N(t,u)$$

Change of variable: $v = e^{-tL}u$, we obtain:

$$\dot{v}(t) = -Le^{-tL}u(t) + e^{-tL}\underbrace{(Lu(t) + N(t,u))}_{\dot{u}(t)}$$

$$= e^{-tL}N(t, e^{tL}v)$$

which can be solved with a **Runge-Kutta method** in $v$, that can be rewritten in $u$, for example with Euler method:

$$v(t^n + \Delta t) \approx v^{n+1} = v^n + \Delta t e^{-t^n L}N(t^n, e^{t^n L}v^n)$$

or as an expression of $u$ :

$$u^{n+1} = e^{\Delta t L}u^n + \Delta t e^{\Delta t L}N(t^n, u^n)$$


Lawson 1967, *SIAM Journal on Numerical Analysis*

Hochbruck, Leibold, and Ostermann 2020, *Numerische Mathematik*

# Lawson method
## Pros & Cons

- ✔ Numerically efficient (order increases linearly-ish with the number of stages)
- ✔ Literature on Runge-Kutta method (embedded-RK, IMEX methods, low storage methods, ...)
- ✔ Linear part is solved exactly
- ✗ Stability constraint (not from the linear part ✔)
- ✗ Behavior in long time
- $\sim$ Needs to compute (efficiently) $e^{\tau L}$ for any $\tau = c_j \Delta t$ and $L$

# Main idea of adaptive time step methods (error estimate)

For a generic ODE $\dot{u} = f(t, u)$, adaptive time step method needs 2 numerical estimations of solution $u(t^{n+1})$ of different order, $p$ and $p+1$:

$$u_{[p]}^{n+1} = u(t^{n+1}) + \mathcal{O}\left(\Delta t^{p+1}\right), \qquad u_{[p+1]}^{n+1} = u(t^{n+1}) + \mathcal{O}\left(\Delta t^{p+2}\right)$$

Estimate of local error: $L_{[p]}^{n+1} = \left| u_{[p+1]}^{n+1} - u_{[p]}^{n+1} \right|$

**If $L_{[p]}^{n+1} > tol$:** we reject the step and start again from time $t^n$. **Else** we accept the step. **In both cases**, the optimal new time step is:

$$\Delta t_{\text{opt}} = \sqrt[p]{\frac{tol}{L_{[p]}^{n+1}}}\Delta t^n$$

In practice $u_{[p]}^{n+1}$ is computed from sub-steps of $u_{[p+1]}^{n+1}$.

- Dormand and Prince 1978, *Celestial mechanics* (for RK method)
- Blanes, Casas, and Thalhammer 2019, *Applied Numerical Mathematics* (for splitting method)

# Outline


1. Introduction
2. Numerical methods
3. Application for hybrid Vlasov-Maxwell model
   - With splitting method
   - With Lawson method
4. Numerical results
5. Conclusion

# Linearized hybrid Vlasov-Maxwell model

$U = (\boldsymbol{j}_c, \boldsymbol{B}, \boldsymbol{E}, f_h)^\top$, $\boldsymbol{j}_c(t,z), \boldsymbol{B}(t,z), \boldsymbol{E}(t,z) \in \mathbb{R}^2$

$$\begin{cases} \partial_t \boldsymbol{j}_c = \Omega_{pe}^2 \boldsymbol{E} - J \boldsymbol{j}_c B_0 \\ \partial_t \boldsymbol{B} = J \partial_z \boldsymbol{E} \\ \partial_t \boldsymbol{E} = -J \partial_z \boldsymbol{B} - \boldsymbol{j}_c + \int v_\perp f_h \,\mathrm{d}v_\perp \\ \partial_t f_h + v_z \partial_z f_h - (\boldsymbol{E} + \boldsymbol{v} \times (\boldsymbol{B} + \boldsymbol{B}_0)) \cdot \nabla_{\boldsymbol{v}} f_h = 0 \end{cases}$$

we define the Hamiltonian as :

$$\mathcal{H} = \underbrace{\frac{1}{2}\int \|\boldsymbol{E}\|^2 \,\mathrm{d}z}_{\mathcal{H}_E} + \underbrace{\frac{1}{2}\int \|\boldsymbol{B}\|^2 \,\mathrm{d}z}_{\mathcal{H}_B} + \underbrace{\frac{1}{2}\int \frac{1}{\Omega_{pe}^2}\|\boldsymbol{j}_c\|^2 \,\mathrm{d}z}_{\mathcal{H}_{j_c}} + \underbrace{\frac{1}{2}\int\int \|\boldsymbol{v}\|^2 f_h \,\mathrm{d}\boldsymbol{v}\,\mathrm{d}z}_{\mathcal{H}_{f_h}}$$

Following the Hamiltonian we built a Hamiltonian splitting.

# Splitting method

5 subsystems $\varphi^{[E]}$, $\varphi^{[B]}$, $\varphi^{[j_c]}$, $\varphi^{[f_h]}$

- Solution with Lie splitting method:

$$U^{n+1} = \varphi^{[E]}_{\Delta t} \circ \varphi^{[B]}_{\Delta t} \circ \varphi^{[j_c]}_{\Delta t} \circ \varphi^{[f_h]}_{\Delta t}(U^n)$$

- or Strang method:

$$U^{n+1} = \varphi^{[E]}_{\Delta t/2} \circ \varphi^{[B]}_{\Delta t/2} \circ \varphi^{[j_c]}_{\Delta t/2} \circ \varphi^{[f_h]}_{\Delta t} \circ \varphi^{[j_c]}_{\Delta t/2} \circ \varphi^{[B]}_{\Delta t/2} \circ \varphi^{[E]}_{\Delta t/2}(U^n)$$

One of sub-steps of Hamiltonian splitting:

$$\varphi^{[E]}(U) = \begin{cases} \partial_t \boldsymbol{j}_c = \Omega_{pe}^2 \boldsymbol{E} \\ \partial_t \boldsymbol{B} = J\partial_z \boldsymbol{E} \\ \partial_t \boldsymbol{E} = 0 \\ \partial_t f_h = \boldsymbol{E} \cdot \nabla_{v_\perp} f_h \end{cases} \to \varphi_t^{[E]}(U^0) = \begin{pmatrix} \boldsymbol{j}_c(0) + t\Omega_{pe}^2 \boldsymbol{E}(0) \\ \boldsymbol{B}(0) + tJ\partial_z \boldsymbol{E}(0) \\ \boldsymbol{E}(0) \\ f_h(0, z, v_\perp + t\boldsymbol{E}(0), v_z) \end{pmatrix}$$

**Numerical tools:**

- 2D interpolation with 2 Lagrange 5 interpolations to approximate $f_h(0, z, v_\perp + t\boldsymbol{E}(0), v_z)$

# Lawson method

$$\partial_t U = LU + N(t, U)$$

with:

$$L = \begin{pmatrix} 0 & -B_0 & 0 & 0 & \Omega_{pe}^2 & 0 & 0 \\ B_0 & 0 & 0 & 0 & 0 & \Omega_{pe}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \partial_z & 0 \\ 0 & 0 & 0 & 0 & -\partial_z & 0 & 0 \\ -1 & 0 & 0 & -\partial_z & 0 & 0 & 0 \\ 0 & -1 & \partial_z & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -v_z\partial_z \end{pmatrix}, \quad N : t, U \mapsto \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \int v_x f_h \,\mathrm{d}\boldsymbol{v} \\ \int v_y f_h \,\mathrm{d}\boldsymbol{v} \\ (\boldsymbol{E} + \boldsymbol{v} \times \boldsymbol{B})\cdot\nabla_{\boldsymbol{v}} f_h \end{pmatrix}$$

But $e^{\tau L}$ can't be computed with symbolic computation software.

# How to compute $e^{\tau L}$?

2 solutions are proposed:

1. Remove some terms of the linear part $L$ and put them in nonlinear part $N$.
   - ✔ symbolic computation to write efficient code
   - ✗ add CFL stability condition
2. Approximate $e^{\tau L}$ with Taylor series or Padé approximant.
   - ✔ no CFL stability from all (local) linear terms
   - ✗ add error of approximation

# Remove terms

Remove Maxwell equations from linear part $L$, and add them in nonlinear term $N$:

$$L = \begin{pmatrix} 0 & -B_0 & 0 & 0 & \Omega_{pe}^2 & 0 & 0 \\ B_0 & 0 & 0 & 0 & 0 & \Omega_{pe}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -v_z\partial_z \end{pmatrix},\ N(t,U) = \begin{pmatrix} 0 \\ 0 \\ \partial_z E_y \\ -\partial_z E_x \\ -\partial_z B_y + \int v_x f_h \,\mathrm{d}\boldsymbol{v} \\ \partial_z B_x + \int v_y f_h \,\mathrm{d}\boldsymbol{v} \\ (\boldsymbol{E} - \boldsymbol{v}\times\boldsymbol{B})\cdot\nabla_{\boldsymbol{v}} f_h \end{pmatrix}$$

- ✔ $e^{\tau L}$ is exactly computed with symbolic computation
- ✗ Add a CFL stability condition in $z$ (coming from explicit resolution of Maxwell equations) which can be estimated.

# Approximation of $e^{\tau L}$

Complete linear part $L$, after Fourier transform in $z$: $\partial_z \mapsto i\kappa$

$$L = \begin{pmatrix} 0 & -B_0 & 0 & 0 & \Omega_{pe}^2 & 0 & 0 \\ B_0 & 0 & 0 & 0 & 0 & \Omega_{pe}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & i\kappa & 0 \\ 0 & 0 & 0 & 0 & -i\kappa & 0 & 0 \\ -1 & 0 & 0 & -i\kappa & 0 & 0 & 0 \\ 0 & -1 & i\kappa & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -i\kappa v_z \end{pmatrix}$$

We have:

$$\forall \kappa, \sigma(L(\kappa)) \subset i\,\mathbb{R}$$

# Taylor series

Simplest approximation:

$$T_p(\tau L) = \sum_{k=0}^{p} \frac{\tau^k}{k!} L^k = e^{\tau L} + \mathcal{O}\left(\tau^{p+1}\right)$$

**Theorem**

*$sp(L) \subset i\mathbb{R} \setminus i[-1,1]$ implies eigenvalues diverge*

*Proof:* compute Taylor series outside of its convergence radius

**Conclusion:**

- ✗ Bad behavior of eigenvalues
- ✗ Numerical instability in scheme

# Eigenvalues of Taylor series

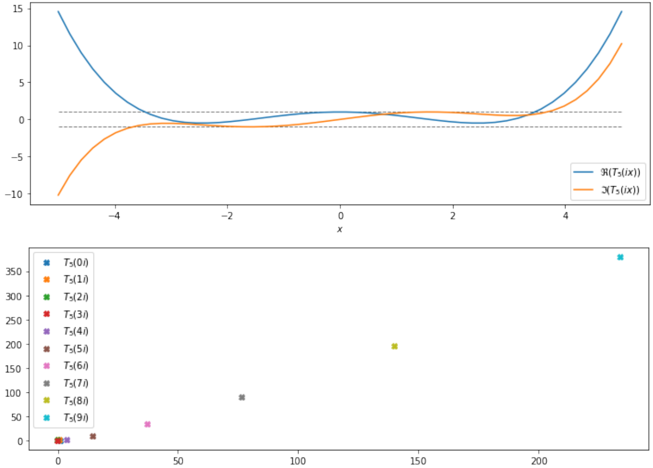


Figure: $T_5(ix)$, $x \in [\![0, 9]\!]$

# Padé approximant

Best rational approximation of exponential function.
Defined (for order $(p, q)$) as:

$$h_{p,q}(M) = \sum_{i=0}^{p} \frac{\frac{p!}{(p-i)!}}{\frac{(p+q)!}{(p+q-i)!}} \frac{M^i}{i!} \quad , \quad k_{p,q}(M) = \sum_{j=0}^{q} (-1)^j \frac{\frac{q!}{(q-j)!}}{\frac{(p+q)!}{(p+q-j)!}} \frac{M^j}{j!}$$

Finally Padé approximant is:

$$P_{p,q}(\tau L) = h_{p,q}(\tau L) \left(k_{p,q}(\tau L)\right)^{-1} = e^{\tau L} + \mathcal{O}\left(\tau^{p+q+1}\right)$$

**Theorem**

$$sp(L) \subset i\mathbb{R} \implies sp(P_{p,p}(tL)) \subset \mathcal{C}(0,1)$$

**Conclusion:**

- ✗ Needs matrix inversion, or some tricks:
  - Li, Zhu, and Gu 2011, *Applied Mathematics*
- ✔ Best approximation for this numerical cost
- ✔ Preserves eigenvalues

# Proof

$L$ diagonalizable ➜ study only on diagonal terms

$$P_{p,p}(iy) = \left(\sum_{k=0}^{p} \frac{1}{k!}(iy)^k\right) \cdot \left(\sum_{\ell=0}^{p} (-1)^\ell \frac{1}{\ell!}(iy)^\ell\right)^{-1}$$

$$\sum_{k=0}^{p} \frac{1}{k!}(iy)^k = \sum_{k=0}^{\lfloor \frac{p}{2} \rfloor} (-1)^k \frac{y^{2k}}{2k!} + i \sum_{k=0}^{\lfloor \frac{p}{2} \rfloor - 1} (-1)^k \frac{y^{2k+1}}{(2k+1)!}$$

$$\sum_{\ell=0}^{p} (-1)^\ell \frac{1}{\ell!}(iy)^\ell = \sum_{\ell=0}^{\lfloor \frac{p}{2} \rfloor} (-1)^\ell \frac{y^{2\ell}}{2\ell!} - i \sum_{\ell=0}^{\lfloor \frac{p}{2} \rfloor - 1} (-1)^\ell \frac{y^{2\ell+1}}{(2\ell+1)!}$$

$\lambda^- = \overline{\lambda^+}$ so $\left|\frac{\lambda^+}{\lambda^-}\right| = 1.$

# Eigenvalues of Padé approximant

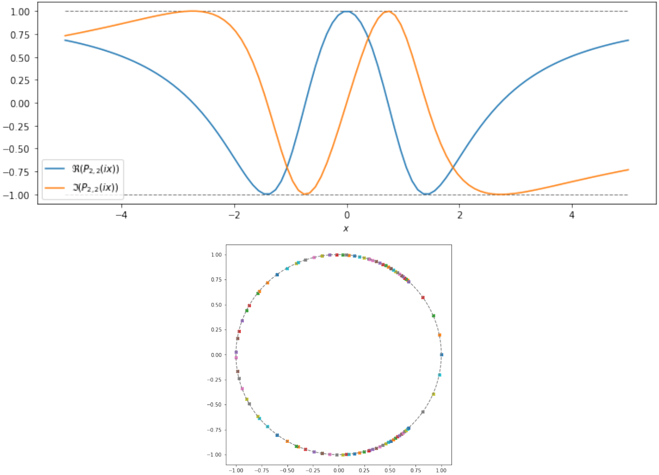


Figure: $P_{2,2}(ix)$, $x \in [-5, 5]$

# Eigenvalues of assymetric Padé approximants

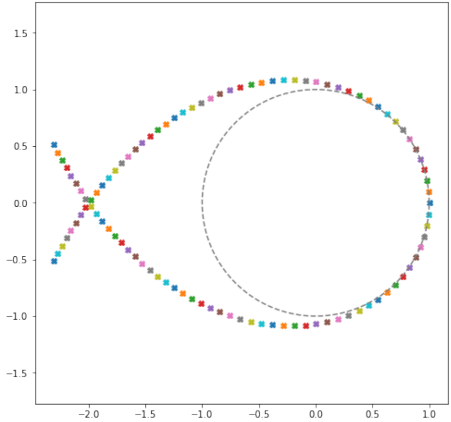


Figure: $P_{2,1}(ix)$ $x \in [-5, 5]$

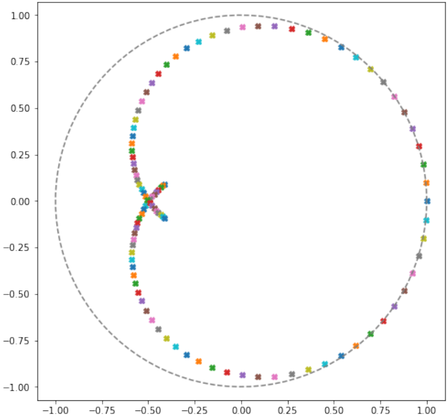


Figure: $P_{1,2}(ix)$ $x \in [-5, 5]$

# Error on approximate Lawson method

We note $\mathfrak{exp}(L) = P_{p.q}(L)$ or $T_p(L)$, generic approximation.

$$\mathfrak{exp}(L) = e^L + \mathcal{O}(L^{r+1})$$

Lawson RK(3,3) method:

$$\begin{aligned}
u^{(1)} &= e^{\Delta t L} u^n + \Delta t e^{\Delta t L} N(t^n, u^n) \\
u^{(2)} &= \frac{3}{4} e^{\frac{\Delta t}{2} L} u^n + \frac{1}{4} e^{-\frac{\Delta t}{2} L} u^{(1)} + \frac{\Delta t}{4} e^{-\frac{\Delta t}{2} L} N(t^n + \Delta t, u^{(1)}) \\
u^{n+1} &= \frac{1}{3} e^{\Delta t L} u^n + \frac{2}{3} e^{\frac{\Delta t}{2} L} u^{(2)} + \frac{2}{3} \Delta t e^{\frac{\Delta t}{2} L} N(t^n + \frac{\Delta t}{2}, u^{(2)})
\end{aligned}$$

**If** $L$ and $N$ commute: $u^{n+1} = e^{\Delta t L} \left( I + N + \frac{N^2}{2} + \frac{N^3}{6} \right) u^n$, stability is same as RK(3,3). **Else**. . .

# Error on approximate Lawson method

**If** $L$ and $N$ don't commute:

$$u^{n+1} = \Big[e^{\Delta tL} + \Delta t\left(\frac{2}{3}e^{\frac{\Delta t}{2}L}Ne^{\frac{\Delta t}{2}L} + \frac{1}{6}e^{\Delta tL}N + \frac{1}{6}Ne^{\Delta tL}\right) \rightsquigarrow e^{\Delta L}\Delta tN$$

$$+ \frac{\Delta t^2}{2}\left(\frac{1}{3}Ne^{\Delta tL}N + \frac{1}{3}e^{\frac{\Delta t}{2}L}Ne^{\frac{\Delta t}{2}L}N + \frac{1}{3}e^{\frac{\Delta t}{2}L}Ne^{-\frac{\Delta t}{2}L}Ne^{\Delta tL}\right)$$

$$\rightsquigarrow e^{\Delta L}\frac{(\Delta tN)^2}{2}$$

$$+ \frac{\Delta t^3}{6}e^{\frac{\Delta t}{2}L}Ne^{-\frac{\Delta t}{2}L}Ne^{\Delta tL}N\Big]u^n \rightsquigarrow e^{\Delta L}\frac{(\Delta tN)^3}{6}$$

Same results if $e^{\Delta tL} \mapsto \mathfrak{exp}(\Delta tL) = e^{\Delta tL} + \mathcal{O}(\Delta t^{r+1})$

## Lemma

*Error for Lawson RK(s,m) is always in* $\mathcal{O}(\Delta t^{m+1}) + \mathcal{O}(\Delta t^{r+1})$

# Test 1

Simulation of $\partial_t u + a\partial_x u + b\partial_y u = 0$ (2D translation test case) and measure order.

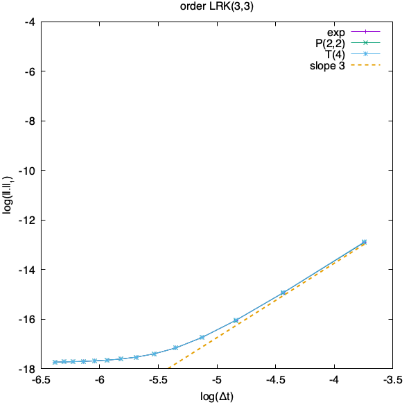


Figure: Order of Lawson RK(3,3) method, and Lawson RK(3,3), $P(2,2)$ approximant method and Lawson RK(3,3) $T(4)$ series method.

# Test 1

Simulation of $\partial_t u + a\partial_x u + b\partial_y u = 0$ (2D translation test case) and measure order.

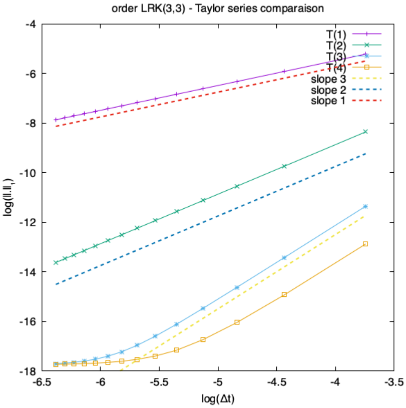


Figure: Order of Lawson RK(3,3) $T(n)$ series method, $n = 1, \ldots, 4$.

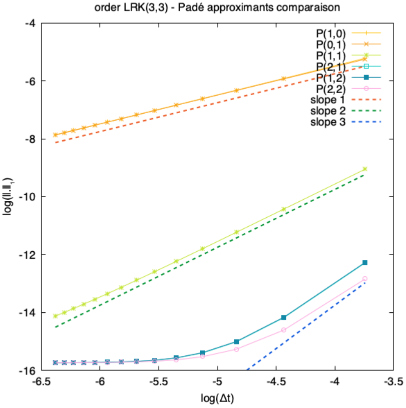


Figure: Order of Lawson RK(3,3) $P(p,q)$ approximant, $p = 1, 2$, $q = 1, 2$

# Test 2

Simulation of $\partial_t u - y\partial_x u + x\partial_y u = 0$ (2D rotation) and test instability

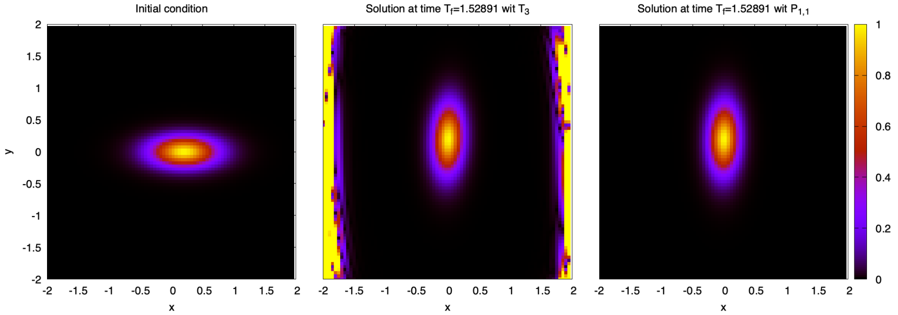


Figure: Initial condition (left), solution with Lawson RK(3,3) $T(3)$ series (middle) and Lawson RK(3,3) $P(1,1)$ approximant (right)

# Outline


1 Introduction

2 Numerical methods

3 Application for hybrid Vlasov-Maxwell model
- With splitting method
- With Lawson method

4 Numerical results

5 Conclusion

We compare:

- Splitting method:
  - Strang (order 2)
  - Suzuki (order 4)
- Lawson method:
  - LRK(4,4) (order 4)
  - LDP4(3) (adaptive time step method)
- Lawson method with approximation of linear part:
  - LRK(4,4) with Padé $(2,2)$ (order 4 + approximation of order $2+2=4$)
  - LDP4(3) with Padé $(2,2)$ (adaptive time step method)

**But:** Padé approximant implies a huge rational function (with invert of matrix), high order Lawson methods have a lot of coefficients, with 7 variables problem. . . ➜ bug source !!!

# Code generator

The main idea of code generator:

1. Write with SymPy the Lawson method with a vector $U \in \mathbb{C}^7$, an abstract matrix $L \in \mathcal{M}_7(\mathbb{C})$ and an abstract nonlinear part $N : t, U \mapsto N(t, U) \in \mathbb{C}^7$
2. Compute $e^{\tau L}$ with our $L$ matrix, and given approximation of exp
3. Loop for each stage of Lawson method into a code template (Jinja2)
4. Save the file, compile and run with a given configuration file

# Numerical results

## $N_z \times N_{v_x} \times N_{v_y} \times N_{v_z} = 27 \times 32 \times 32 \times 41$

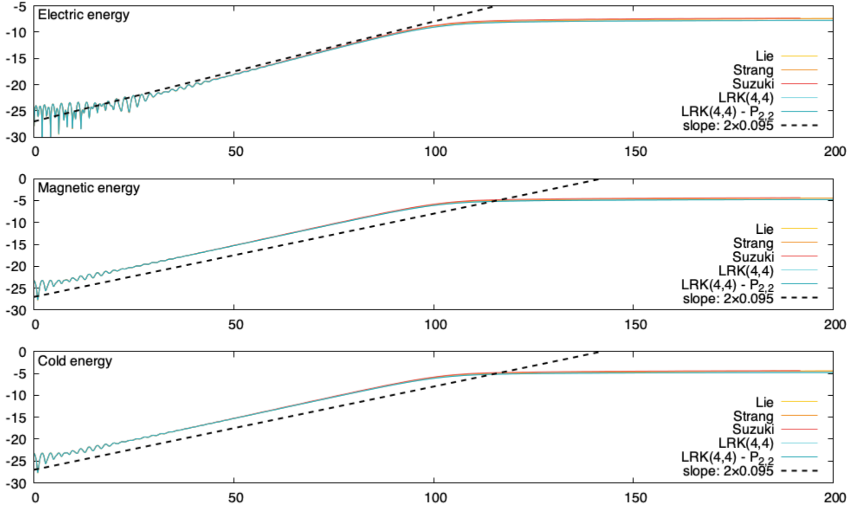


Figure: Energies evolution, $\Delta t = 0.05$

# Numerical results

$N_z \times N_{v_x} \times N_{v_y} \times N_{v_z} = 27 \times 32 \times 32 \times 41$

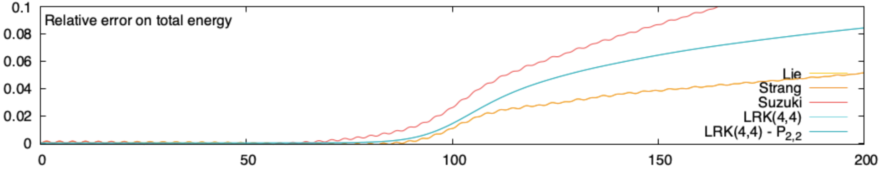


Figure: Relative error on total energy, $\Delta t = 0.05$

# Numerical results

$N_z \times N_{v_x} \times N_{v_y} \times N_{v_z} = 27 \times 32 \times 32 \times 41$

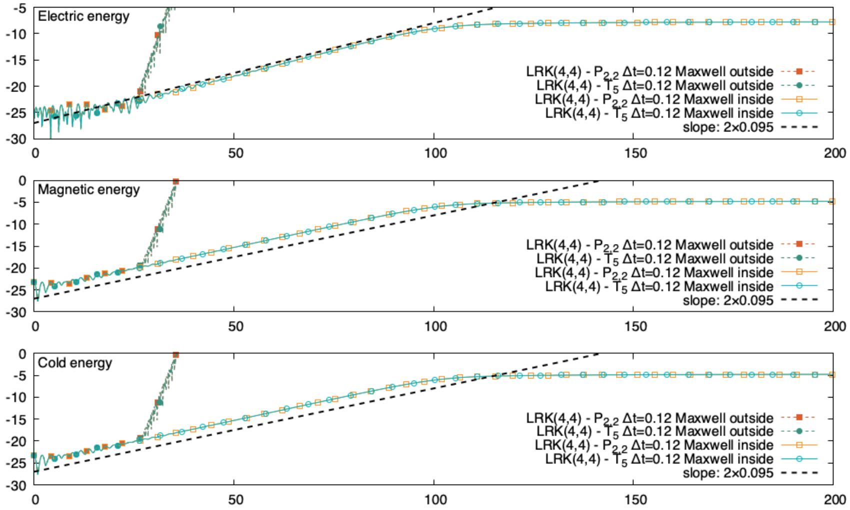


Figure: Energies evolution, Lawson with Taylor or Padé approximation, $\Delta t = 0.12$

# Numerical results

$N_z \times N_{v_x} \times N_{v_y} \times N_{v_z} = 27 \times 32 \times 32 \times 41$

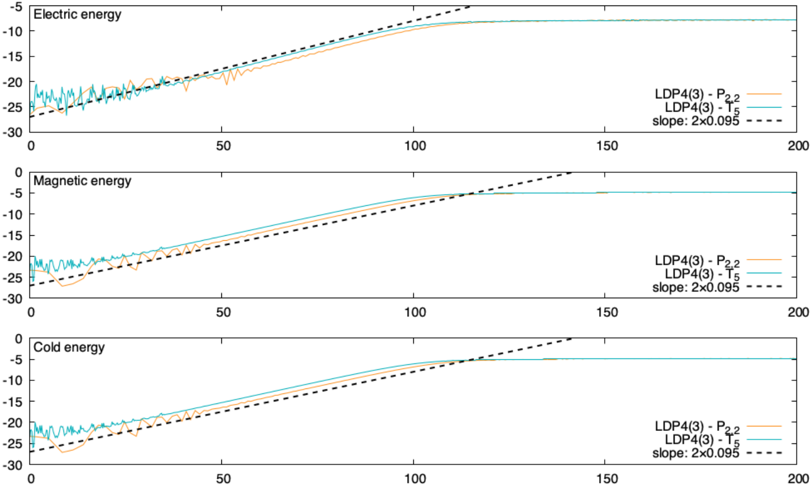


Figure: Energies evolution, Lawson with Taylor or Padé approximation, $\Delta t^n$

# Numerical results

$N_z \times N_{v_x} \times N_{v_y} \times N_{v_z} = 27 \times 32 \times 32 \times 41$

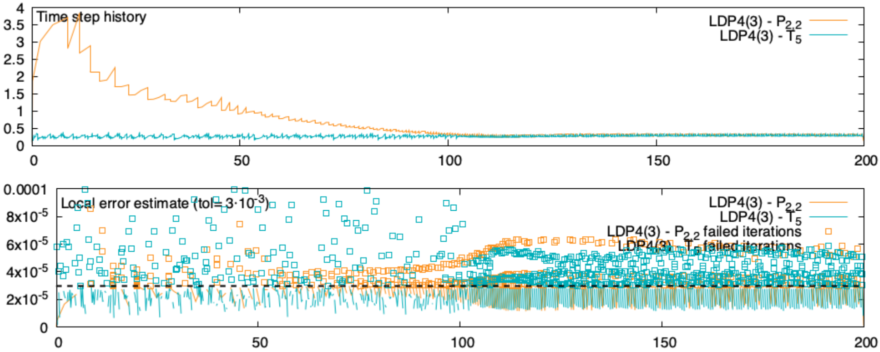


Figure: Time step evolution and estimate of local error, Lawson with Taylor or Padé approximation, $\Delta t^n$

# Simulation time

| time integrator | simulation time |
|---|---|
| Lie splitting | 13 h 25 min 10 s |
| Strang splitting | 17 h 09 min 54 s |
| Suzuki splitting | 3 j 03 h 05 min 24 s |
| LRK(3,3) | 11 h 29 min 09 s |
| LRK(3,3) - $T_4$ | 10 h 53 min 40 s |
| LRK(3,3) - $P_{1,1}$ | 10 h 54 min 11 s |
| LRK(3,3) - $P_{2,2}$ | 10 h 55 min 26 s |
| LRK(4,4) | 14 h 06 min 15 s |
| LRK(4,4) - $T_5$ | 14 h 00 min 03 s |
| LRK(4,4) - $P_{2,2}$ | 13 h 59 min 59 s |
| LDP4(3) | 11 h 44 min 04 s |
| LDP4(3) - $P_{2,2}$ | 04 h 09 min 44 s |

Table: Simulation time for some simulation, on mesh $N_z \times N_{v_x} \times N_{v_y} \times N_{v_z} = 27 \times 32 \times 32 \times 41$ and time step $\Delta t = 0.05$ (initial time step for adaptive time step strategy).

# Outline


1. Introduction
2. Numerical methods
3. Application for hybrid Vlasov-Maxwell model
   - With splitting method
   - With Lawson method
4. Numerical results
5. Conclusion

# Conclusion

- ✔ First simulations of this system using Hamiltonian splitting
- ✗ Numerical cost of splitting methods (not bad in $1dz - 1dv$ but very bad in $1dz - 3dv$, must be very very very bad in $3dx - 3dv$)
- ✔ Numerical cost of Lawson methods
- $\sim$ Behavior of total energy of Lawson method (but we can use high order method easily)
- ✔ Error of approximation with Padé approximant can be lower than time integrator

# Future works

- Compare Lawson method with Padé approximant with a PIC simulation
- Add $\int \boldsymbol{v} f_h \,\mathrm{d}\boldsymbol{v}$ in linear part (for $1dx - 1dv$ model)

Thank you for your attention

Backup

# Adaptive time step method for splitting method

Blanes, Casas, and Thalhammer 2019, *Applied Numerical Mathematics* for Suzuki splitting method

$$u_{[4]}^{n+1} = \mathcal{S}_{\Delta t}(u^n) = \underbrace{S_{\alpha_1\Delta t} \circ \underbrace{S_{\alpha_2\Delta t} \circ \underbrace{S_{\alpha_3\Delta t} \circ \underbrace{S_{\alpha_2\Delta t} \circ \underbrace{S_{\alpha_1\Delta t}(u^n)}_{u^{(1)}}}_{u^{(2)}}}_{u^{(3)}}}_{u^{(4)}}.$$

We compute an order 3 approximation from $U^n$ and $U^{(s)}$, $s = 1, 2, 3, 4$ :

$$u_{[3]}^{n+1} = -u^n + w_1(u^{(1)} + u^{(4)}) + w_2(u^{(2)} + u^{(3)})$$

with:

$$w_1 = \frac{g_2(1-g_2)}{g_1(g_1-1) - g_2(g_2-1)}, \quad w_2 = 1 - w_1, \quad \begin{array}{l} g_1 = \alpha_1 \\ g_2 = \alpha_1 + \alpha_2 \end{array}$$

and $L_{[3]}^n = \left\| u_{[4]}^{n+1} - u_{[3]}^{n+1} \right\|_2$

# Adaptive time step method for Lawson method

Lawson methods are built on Runge-Kutta method, embedded Lawson method are written with an underlying embedded Runge-Kutta method.

Dormand and Prince 1978, *Celestial mechanics*

With DP4(3) (Dormand-Prince method of order 4, with embedded 3 method):

$$\left.\begin{array}{c|ccccc}
0 & & & & & \\
\frac{1}{2} & \frac{1}{2} & & & & \\
\frac{1}{2} & 0 & \frac{1}{2} & & & \\
1 & 0 & 0 & 1 & & \\
\hline
1 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} & \\
\hline
 & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{2}{30} & \frac{1}{10}
\end{array}\right\}\text{Classical RK(4,4)}$$

We compute a 3rd order approximation from $u^n$, $u^{(s)}$, $s = 1, 2, 3, 4$ done by the last line of Butcher tableau.

And $L^n_{[3]} = \left\| u^{n+1}_{[4]} - u^{n+1}_{[3]} \right\|_2$

# Poisson bracket

For two given functionals $\mathcal{F}$, $\mathcal{G}$ of $\boldsymbol{j}_c$, $\boldsymbol{B}$, $\boldsymbol{E}$, $f_h$, the Poisson bracket is given by

$$
\begin{aligned}
\{\mathcal{F},\mathcal{G}\}[\boldsymbol{j}_c,\boldsymbol{B},\boldsymbol{E},f_h] = {} & \frac{1}{m_e}\int_\Omega\int_{\mathbb{R}^3} f_h\left[\frac{\delta\mathcal{F}}{\delta f_h},\frac{\delta\mathcal{G}}{\delta f_h}\right]_{\boldsymbol{xv}}\mathrm{d}\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} \\
& + \frac{q_e}{m_e\varepsilon_0}\int_\Omega\int_{\mathbb{R}^3} f_h\left(\nabla_{\boldsymbol{v}}\frac{\delta\mathcal{F}}{\delta f_h}\cdot\frac{\delta\mathcal{G}}{\delta\boldsymbol{E}} - \nabla_{\boldsymbol{v}}\frac{\delta\mathcal{G}}{\delta f_h}\cdot\frac{\delta\mathcal{F}}{\delta\boldsymbol{E}}\right)\mathrm{d}\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} \\
& + \frac{q_e}{m_e^2}\int_\Omega\int_{\mathbb{R}^3} f_h(\boldsymbol{B}+\boldsymbol{B}_0)\cdot\left(\nabla_{\boldsymbol{v}}\frac{\delta\mathcal{F}}{\delta f_h}\times\nabla_{\boldsymbol{v}}\frac{\delta\mathcal{G}}{\delta f_h}\right)\mathrm{d}\boldsymbol{v}\,\mathrm{d}\boldsymbol{x} \\
& + \frac{1}{\varepsilon_0}\int_\Omega\left(\nabla\times\frac{\delta\mathcal{F}}{\delta\boldsymbol{E}}\cdot\frac{\delta\mathcal{G}}{\delta\boldsymbol{B}} - \nabla\times\frac{\delta\mathcal{G}}{\delta\boldsymbol{E}}\cdot\frac{\delta\mathcal{F}}{\delta\boldsymbol{B}}\right)\mathrm{d}\boldsymbol{x} \\
& + \int_\Omega \Omega_{pe}^2\left(\frac{\delta\mathcal{F}}{\delta\boldsymbol{j}_c}\cdot\frac{\delta\mathcal{G}}{\delta\boldsymbol{E}} - \frac{\delta\mathcal{G}}{\delta\boldsymbol{j}_c}\cdot\frac{\delta\mathcal{F}}{\delta\boldsymbol{E}}\right)\mathrm{d}\boldsymbol{x} \\
& + \frac{q_e\varepsilon_0}{m_e}\int_\Omega \Omega_{pe}^2\boldsymbol{B}_0\cdot\left(\frac{\delta\mathcal{F}}{\delta\boldsymbol{j}_c}\times\frac{\delta\mathcal{G}}{\delta\boldsymbol{j}_c}\right)\mathrm{d}\boldsymbol{x}\,.
\end{aligned}
$$

# Splitting method $\varphi^{[j_c]}$

$$\varphi^{[j_c]}(U) = \begin{cases} \partial_t \boldsymbol{j}_c = -J\boldsymbol{j}B_0 \\ \partial_t \boldsymbol{B} = 0 \\ \partial_t \boldsymbol{E} = -\boldsymbol{j}_c \\ \partial_t f_h = 0 \end{cases} \quad \rightarrow \varphi_t^{[j_c]}(U^0) = \begin{pmatrix} e^{-tJ}\boldsymbol{j}_c(0)B_0 \\ \boldsymbol{B}(0) \\ \boldsymbol{E}(0) - J(e^{-tJ} - I)\boldsymbol{j}_c(0) \\ f_h(0) \end{pmatrix}$$

Obtain because: $\int_0^t \exp(-sJ)\boldsymbol{j}_c(0)\,\mathrm{d}s = J(\exp(-tJ) - I)\boldsymbol{j}_c(0)$, with:

$$\exp(-tJ) = \begin{pmatrix} \cos(t) & -\sin(t) \\ \sin(t) & \cos(t) \end{pmatrix}$$

# Splitting method $\varphi^{[B]}$

$$\varphi^{[B]}(U) = \begin{cases} \partial_t \boldsymbol{j}_c = 0 \\ \partial_t \boldsymbol{B} = 0 \\ \partial_t \boldsymbol{E} = -J\partial_z \boldsymbol{B} \\ \partial_t f_h = 0 \end{cases} \quad \rightarrow \varphi_t^{[B]}(U^0) = \begin{pmatrix} \boldsymbol{j}_c(0) \\ \boldsymbol{B}(0) \\ \boldsymbol{E}(0) - tJ\partial_z \boldsymbol{B}(0) \\ f_h(0) \end{pmatrix}$$

**Numerical tools:**

- Solve in Fourier space

# Splitting method $\varphi^{[E]}$

$$\varphi^{[E]}(U) = \begin{cases} \partial_t \boldsymbol{j}_c = \Omega_{pe}^2 \boldsymbol{E} \\ \partial_t \boldsymbol{B} = J \partial_z \boldsymbol{E} \\ \partial_t \boldsymbol{E} = 0 \\ \partial_t f_h = \boldsymbol{E} \cdot \nabla_{\boldsymbol{v}} f_h \end{cases} \quad \rightarrow \varphi_t^{[E]}(U^0) = \begin{pmatrix} \boldsymbol{j}_c(0) + t\Omega_{pe}^2 \boldsymbol{E}(0) \\ \boldsymbol{B}(0) + tJ\partial_z \boldsymbol{E}(0) \\ \boldsymbol{E}(0) \\ f_h(0, z, \boldsymbol{v} + t\boldsymbol{E}(0), v_z) \end{pmatrix}$$

**Numerical tools:**

- 2D interpolation with 2 Lagrange 5 interpolations to approximate $f_h(0, z, \boldsymbol{v} + t\boldsymbol{E}(0), v_z)$

# Splitting method $\varphi^{[f_h]}$

$$\varphi^{[f_h]}(U) = \begin{cases} \partial_t \boldsymbol{j}_c = 0 \\ \partial_t \boldsymbol{B} = 0 \\ \partial_t \boldsymbol{E} = \int \boldsymbol{v} f_h \,\mathrm{d}\boldsymbol{v} \\ \partial_t f_h = -v_z \partial_z f_h + (\boldsymbol{v} \times (\boldsymbol{B} + \boldsymbol{B}_0)) \cdot \nabla_{\boldsymbol{v}} f_h \end{cases}$$

This step is split again onto 3 parts.

# Splitting method $\varphi^{[f_h]}$, $f_{h,\star}$, $\star \in \{x, y\}$

$$\varphi^{[f_{h,x}]}(U) = \begin{cases} \partial_t \boldsymbol{j}_c = 0 \\ \partial_t \boldsymbol{B} = 0 \\ \partial_t E_x = \int v_x f_h \,\mathrm{d}\boldsymbol{v} \\ \partial_t E_y = 0 \\ \partial_t f_h = -v_x B_0 \partial_{v_y} f_h + v_x B_y \partial_{v_z} f_h \end{cases} \to \varphi_t^{[f_{h,x}]}(U^0) = \begin{pmatrix} \boldsymbol{j}_c(0) \\ \boldsymbol{B}(0) \\ E_x(0) + t \int v_x f_h(0) \,\mathrm{d}\boldsymbol{v} \\ E_y(0) \\ f_h(0, z, v_x, v_y - t v_x B_0, v_z + t B_y v_x) \end{pmatrix}$$

**Numerical tools:**

- 2D interpolation with Lagrange 5 interpolation to approximate $f_h(0, z, v_x, v_y - t v_x B_0, v_z + t B_y v_x)$

Same thing for $\varphi^{[f_{h,y}]}$ in $v_y$ direction.

# Splitting method $\varphi^{[f_h]}$, $f_{h,z}$

$$\varphi^{[f_{h,z}]}(U) = \begin{cases} \partial_t \boldsymbol{j}_c = 0 \\ \partial_t \boldsymbol{B} = 0 \\ \partial_t \boldsymbol{E} = 0 \\ \partial_t f_h = -v_z \partial_z f_h + (-v_z B_y \partial_{v_x} f_h + v_z B_x \partial_{v_y} f_h) \end{cases}$$

**Numerical tools:**

- Split **again** onto 3 parts, with change of variable $g(t, z, \boldsymbol{v}) := f(t, z + tv_z, \boldsymbol{v})$
- 2D interpolation with Lagrange 5 interpolation to approximate $g(0, z, v_x - \sum_k \hat{B}_y(0,k)\frac{1}{ik}e^{ikz}(e^{iktv_z} - 1), v_y + \sum_k \hat{B}_x(0,k)\frac{1}{ik}e^{ikz}(e^{iktv_z} - 1), v_z)$
- Revert change of variable with Fourier transform

# Splitting method

For Lie method:

$$U^{n+1} = \varphi_{\Delta t}^{[j_c]} \circ \varphi_{\Delta t}^{[B]} \circ \varphi_{\Delta t}^{[E_{v_x}]} \circ \varphi_{\Delta t}^{[E_{v_y}]} \circ \varphi_{\Delta t}^{[f_{h,x,v_x}]} \circ \varphi_{\Delta t}^{[f_{h,x,v_z}]} \circ \varphi_{\Delta t}^{[f_{h,y,v_y}]} \circ \varphi_{\Delta t}^{[f_{h,y,v_z}]} \circ \varphi_{\Delta t}^{[f_{h,z,1}]} \circ \varphi_{\Delta t}^{[f_{h,z,2}]} \circ \varphi_{\Delta t}^{[f_{h,z,3}]}(U^n)$$