Exponential methods for solving hyperbolic problems with application to kinetic equations

N. Crouseilles [1,2]   L. Einkemmer [3]   J. Massot [2,1]

[1]Inria Rennes – Bretagne Atlantique

[2]IRMAR, Université de Rennes

[3]University of Innsbruck

December 10, 2019

Outline

Outline

Vlasov-Poisson equations 1D×1D

Our model: a non-linear transport in $(x, v) \in \Omega \times \mathbb{R}$ of an electron density distribution $f = f(t, x, v)$:

$$\begin{cases} \partial_t f + v\partial_x f + E\partial_v f = 0 \\ \partial_x E = \int_{\mathbb{R}} f \,\mathrm{d}v - 1 \end{cases}$$

Motivation:

- We want high order methods in (x, v)
- We want high order methods in time t:
  - Splitting methods: could have a lot of steps
  - Runge-Kutta methods: stability constraints (CFL condition)
    - The most restrictive CFL condition is associated with the linear part $(\partial_t f + v\partial_x f = 0)$

➜We want to propose a compromise: exponential integrators.

Vlasov-Poisson equations 1D×1D

Fourier transform in x direction of Vlasov, amenable to exponential integrators:

$$\partial_t \hat{f} + ikv\hat{f} + \widehat{E\partial_v f} = 0$$

Vlasov is of the form:

$$\dot{u} = iau + F(u)$$

Variation of constant: $\partial_t(e^{-iat}u) = e^{-iat}F(u)$. No more CFL in x of the form $\Delta t \leq \sigma\frac{\Delta x}{v_{\max}}$ with $[-v_{\max}, v_{\max}] \equiv \mathbb{R}$.
Time integration:

$$u(t_n + \Delta t) = \exp(ia\Delta t)u(t_n) + \int_0^{\Delta t} \exp(ia(\Delta t - s))F(u(t_n + s))\,\mathrm{d}s$$

with $\Delta t > 0$, $t_n = n\Delta t$ with $n \in \mathbb{N}$
Linear part is exact! ✔

Idea of exponential integrators

2 classes of methods:

exponential Runge-Kutta: solve exactly what we can, and interpolate the rest. For example first order exponential Euler method:

$$u(t_n + \Delta t) \approx u^{n+1} = e^{-ia\Delta t} u^n + \Delta t \varphi_1(ia\Delta t) F(u^n)$$

where $\varphi_1(z) = \dfrac{e^z - 1}{z}$

Hochbruck and Ostermann (2010)

Lawson: Change of variable: $v(t) = e^{-iat} u(t)$, we solve with a RK method: $\dot{v} = \tilde{F}(t, v) = e^{-iat} F(e^{iat} v(t))$
For example, Lawson Euler method:

$$v(t_n + \Delta t) \approx v^{n+1} = v^n + \Delta t e^{-iat_n} F(e^{iat_n} v^n)$$

or as an expression of u:

$$u^{n+1} = e^{-ia\Delta t} u^n + \Delta t e^{ia\Delta t} F(u^n)$$

Isherwood, Grant, and Gottlieb (2018)

Outline

Reminder of stability tools

If we want to study stability of:

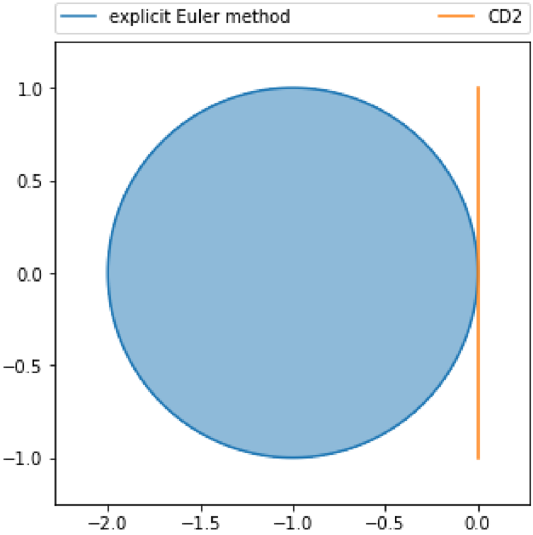
$$\partial_t u + \partial_x u = 0$$

with centered scheme (CD2) $(\partial_x u)_j \approx \frac{1}{2\Delta x}(u_{j+1} - u_{j-1})$. After a Fourier transform (*von Neumann analysis*):

$$\dot{u} + i\frac{\sin(k\Delta x)}{\Delta x}u = 0$$

Explicit Euler method in time: we have to stretch **eigenvalues** (or **Fourier symbol**) of CD2 into explicit Euler **stability domain**.

Reminder of stability tools



From linear Vlasov equation to toy model

Linear Vlasov equation:

$$\partial_t f + a\partial_x f + b\partial_v f = 0$$

Fourier transform in x, CD2 in v plus a Fourier transform in v, formally:

$$\frac{\mathrm{d}f}{\mathrm{d}t} + iakf + b\frac{i\sin(\varphi)}{\Delta v}f = 0$$

Toy model:

$$\dot{u} + iau + \lambda u = 0$$

with $a \in \mathbb{R}$, $\lambda \in \mathbb{C}$ (diffusive scheme for example).
λ is the Fourier symbol (or eigenvalues) of FD method to approximate $\partial_v f$.

Phase discretization

In v direction we use a FD method:

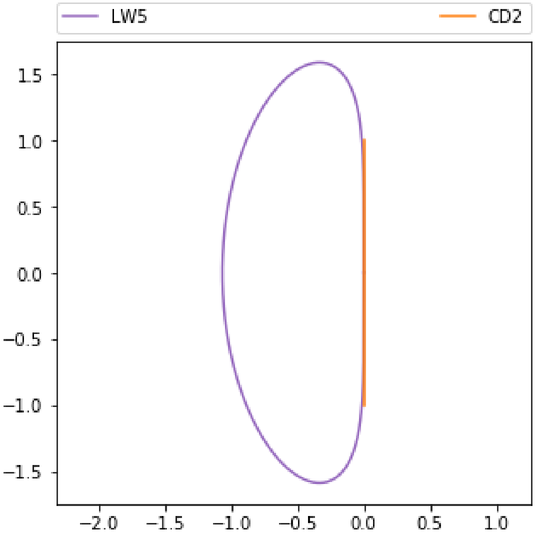
- CD2 (centered difference of order 2): $(\partial_v f)(v_j) \approx \dfrac{f_{j+1} - f_{j-1}}{2\Delta v}$
- WENO5 (weighted essentially non-oscillatory of order 5):
  - WENO5: non linear scheme: ~~Von Neumann analysis~~
  - LW5 (linearized WENO5): linear scheme (this is Lagrange interpolation of order 5)

$$(\partial_v f)(v_j) \approx \frac{1}{\Delta v}\left(-\frac{1}{30}f_{j-3} + \frac{1}{4}f_{j-2} - f_{j-1} + \frac{1}{3}f_j + \frac{1}{2}f_{j+1} - \frac{1}{20}f_{j+2}\right)$$

Wang and Spiteri (2007)

Motamed, Macdonald, and Ruuth (2010)

Fourier symbols



Lawson methods stability domain

For our toy model:

$$\dot{u} = iau + \lambda(u)$$

Change of variable: $v(t) = e^{-iat}u(t)$

$$\dot{v} = e^{-iat}\lambda e^{iat}v$$

Apply a Runge-Kutta method to compute stability function of Lawson method:

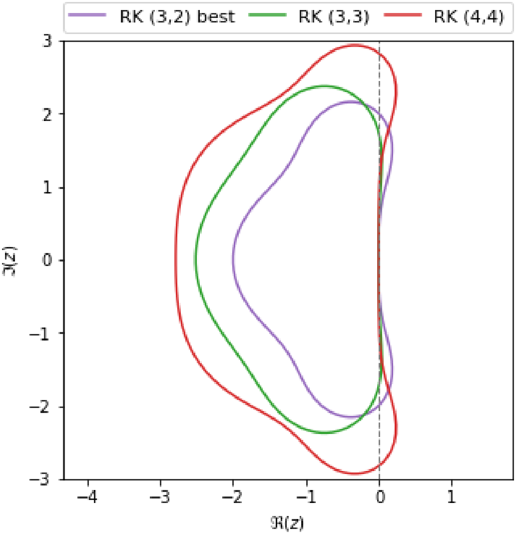
$$v^{n+1} = \underbrace{p(\lambda\Delta t)}_{\text{stability function of RK}} v^n$$

i.e.:

$$u^{n+1} = \overbrace{p(\lambda\Delta t)e^{-ia\Delta t}}^{\text{stability function of Lawson}} u^n$$

Stability domain: $\mathcal{D} = \{z \in \mathbb{C}, |p(z)| \leq 1\}$ of Lawson method is **the same** as the underlying Runge-Kutta method **because** $ia \in i\mathbb{R}$

Considered $Lawson(RK(s,p))$ methods



Lawson methods – CD2

For stability between a Lawson method and CD2, we solve:

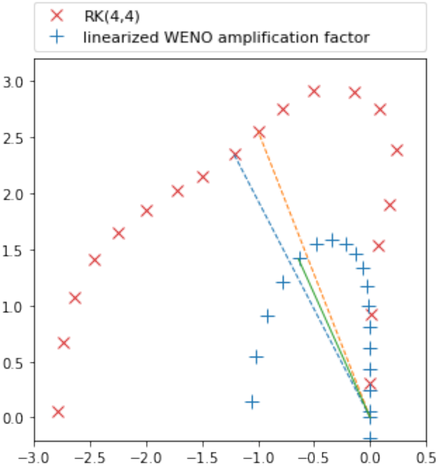
$$|p(iy)| = 1, \quad y \in \mathbb{R}$$

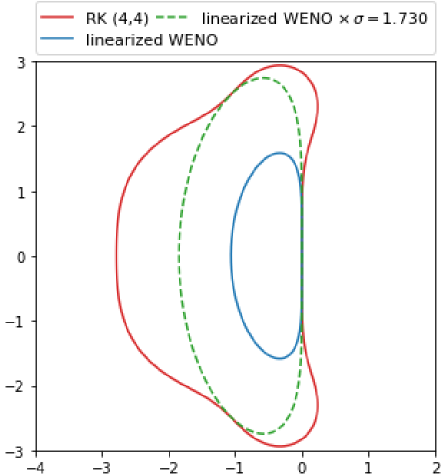
Methods	Lawson($RK(3,2)$ *best*)	Lawson($RK(3,3)$)	Lawson($RK(4,4)$)
$y_{\max}$	2	$\sqrt{3}$	$2\sqrt{2}$

Table: CFL number for some Lawson schemes

Baldauf (2008)

Lawson methods – LW5





Lawson methods – LW5: CFL estimates

Methods	Lawson($RK(3,2)$ *best*)	Lawson($RK(3,3)$)	Lawson($RK(4,4)$)
σ	1.344	1.433	1.73

Table: CFL number for some Lawson schemes.

Motamed, Macdonald, and Ruuth (2010)

Lunet et al. (2017)

Exponential Runge-Kutta methods

$$\dot{u} = iau + F(u)$$

Example on ExpRK(2,2):

$$\begin{aligned} u^{(1)} &= e^{-ia\Delta t} u^n - \Delta t \varphi_1 F(u^n) \\ u^{n+1} &= e^{-ia\Delta t} u^n - \Delta t \left[(\varphi_1 - \varphi_2) F(u^n) + \varphi_2 F(u^{(1)}) \right] \end{aligned}$$

Stability function becomes:

$$p_{\text{ExpRK(2,2)}}(z) = \frac{1}{2}\varphi_1 \varphi_{1,2} z^2 + (\varphi_1 + i\frac{\varphi_1 \varphi_{1,2}}{2} a) z + 1 + i\varphi_1 a$$

Stability domain depends of $a\Delta t$... ✗

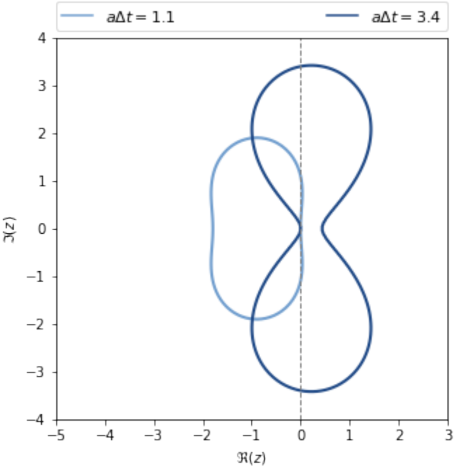


Figure: Stability domain of ExpRK(2,2) for $a\Delta t \in \{1.1, 3.4\}$

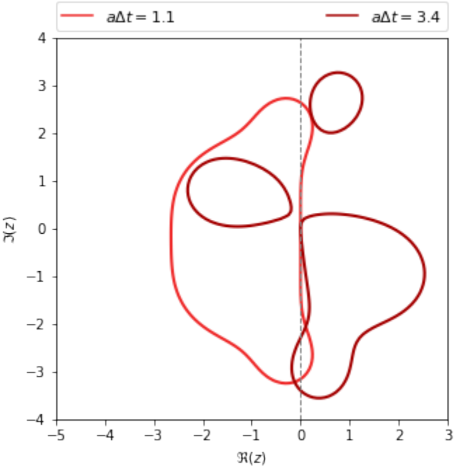


Figure: Stability domain of Cox-Matthews for $a\Delta t \in \{1.1, 3.4\}$

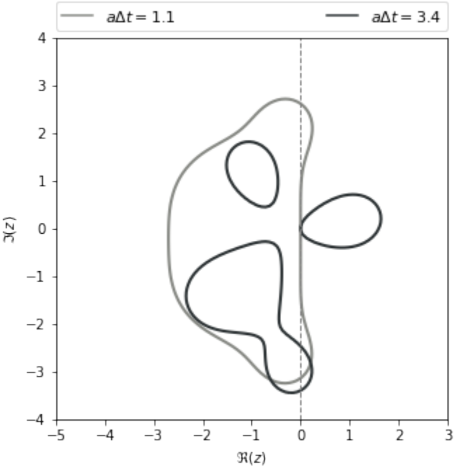


Figure: Stability domain of Krogstad for $a\Delta t \in \{1.1, 3.4\}$

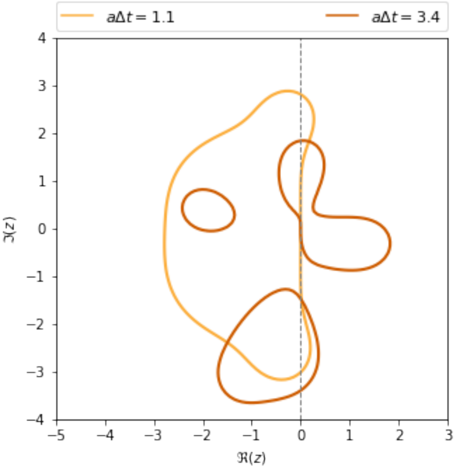


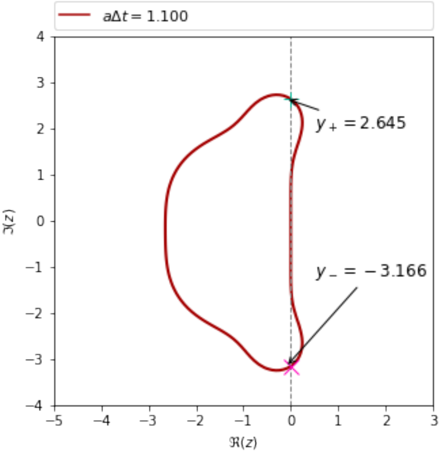
Figure: Stability domain of Hochbruck–Ostermann for $a\Delta t \in \{1.1, 3.4\}$

Stability domain informations

Fourier symbol must fit in the stability domain of ExpRK method **for all** values of $a\Delta t \in \mathbb{R}$.

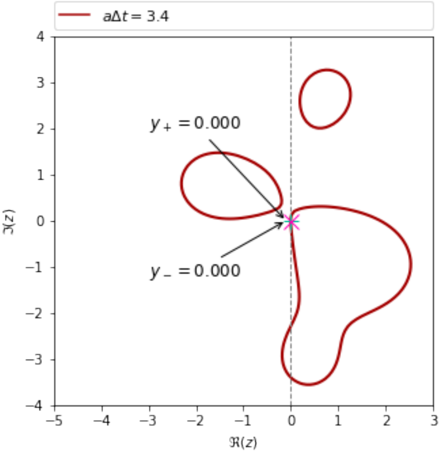
- ✗ Impossible with WENO5 (LW5 Fourier symbol)
  - $\rightarrow$ Numerical test: unstable in very short time
- ✔ Singleton $\{0\}$ is alway in stability domain of ExpRK method for each values of $a\Delta t$
  - $\rightarrow$ We can try to stabilize CD2
  - ✗ SPOILER: CFL is equal to zero

ExpRK – CD2



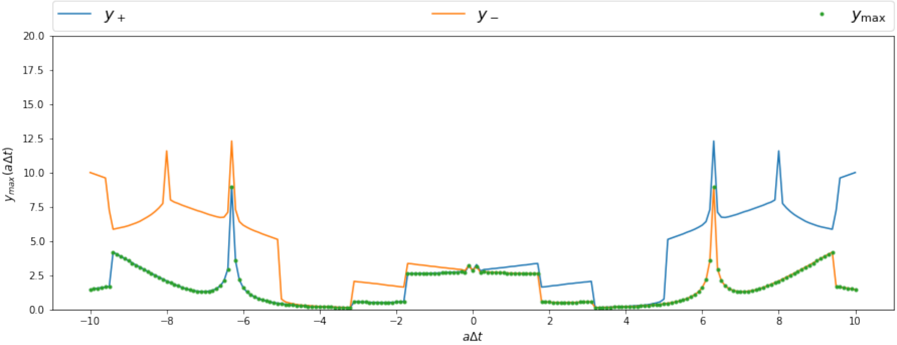
$y_{\max}^{exp} = \min(y_+, |y_-|)$ the largest value to stretch $i[-1, 1]$ into the stability domain at $a\Delta t$

ExpRK – CD2



$y_{\max}^{exp} = \min(y_+, |y_-|)$ the largest value to stretch $i[-1, 1]$ into the stability domain at $a\Delta t$

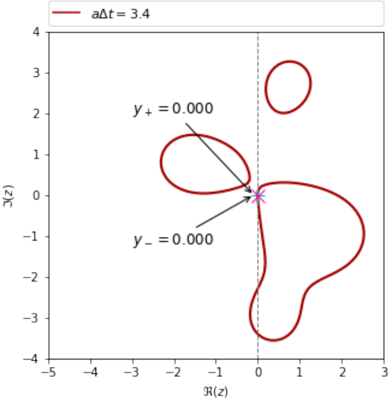
ExpRK – CD2: CFL number



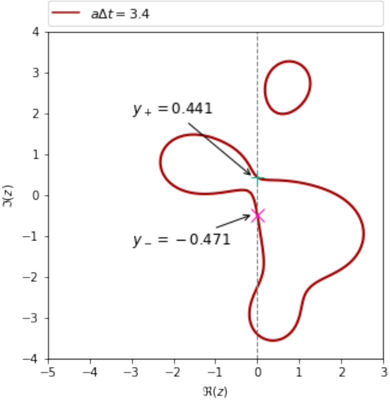
CFL $y_{\max} = \min_{a\Delta t} y_{\max}^{exp}$ is still 0. . .

ExpRK – CD2: Relaxed CFL condition

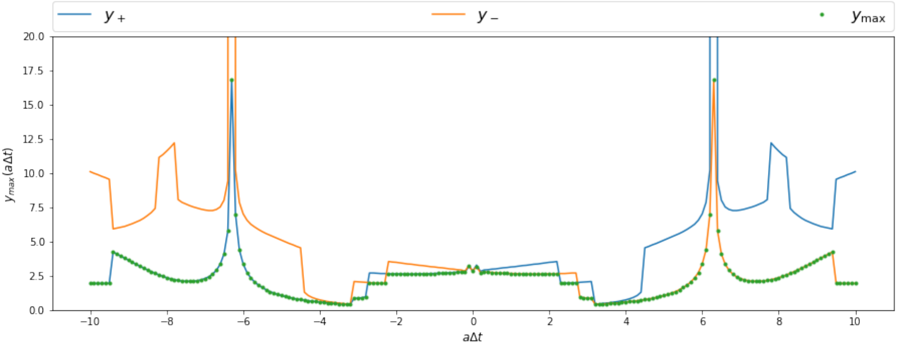
$$\mathcal{D}_\varepsilon = \{z \in \mathbb{C}, |p(z)| \leq 1 + \varepsilon\}$$



Cox-Matthews stability domain, relaxation $\varepsilon = 0$



Cox-Matthews stability domain, relaxation $\varepsilon = 10^{-2}$



Relaxed CFL $y_{\max}(\varepsilon = 10^{-2}) \approx 0.450 \neq 0$! (but unstable in theory)

ExpRK – CD2: Relaxed CFL estimates

Methods	ExpRK22	Krogstad	Cox–Matthews	Hochbruck –Ostermann
$y_{\max}(\varepsilon = 10^{-3})$	0.300	0.100	0.150	0.250
$y_{\max}(\varepsilon = 10^{-2})$	0.551	0.200	0.450	0.501
$y_{\max}(\varepsilon = 10^{-1})$	1.001	0.601	1.351	1.702

Table: CFL number, assuming the relaxed stability constraint, for some exponential integrators.

It's unstable in theory, in practice, number of iterations is finished, so amplification is controlled.

Outline

Vlasov-Poisson equations

$$\begin{cases} \partial_t f + v\partial_x f + E\partial_v f = 0 \\ \partial_x E = \int_{\mathbb{R}} f \,\mathrm{d}v - 1 \end{cases}$$

Numerical tools:

- FFT in x direction
- CD2 or WENO5 in v direction
- $Lawson(RK(s,p))$ or ExpRK method in time t

CFL: $\Delta t_n \leq \dfrac{C\Delta v}{||E^n||_\infty} \leq \dfrac{C\Delta v}{\max_n ||E^n||_\infty}$ where $C = y_{\max}$ or σ from the linear theory.

We can choose: $\Delta t = \min\left(0.1, \dfrac{C\Delta v}{\max_n ||E^n||_\infty}\right)$

Landau damping

$$f(t=0,x,v) = f_0(x,v) = \frac{1}{\sqrt{2\pi}} e^{-\frac{v^2}{2}} (1 + 0.001\cos(0.5x))$$

$x \in [0, 4\pi]$, $v \in [-8, 8]$, $N_x = 81$, $N_v = 128$

Because of damping:

$$\max_n ||E^n||_\infty = ||E^0||_\infty$$

So, we choose $\Delta t = 0.1$ (with $\Delta t = 100$ it is still stable!)

Landau damping: numerical results

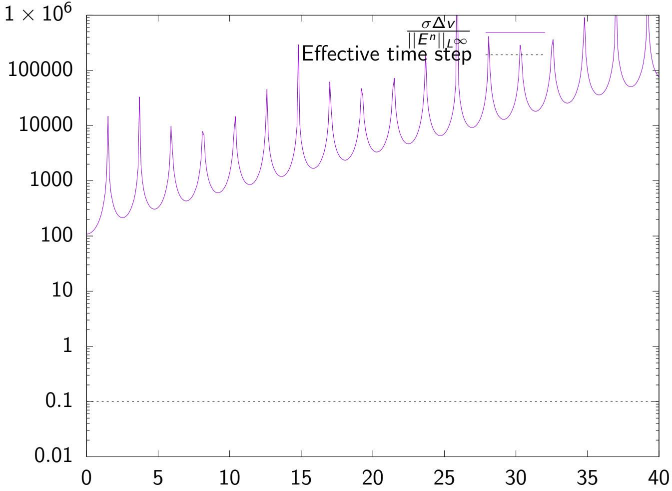


Figure: Landau damping test: time history of the CFL condition (semi-log scale).

Landau damping: numerical results

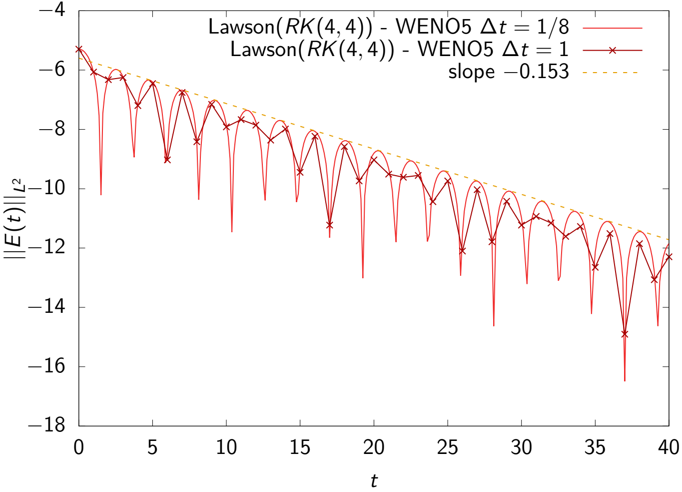


Figure: Landau damping test: time history of $\|E(t)\|_{L^2}$ (semi-log scale) obtained with Lawson($RK(4,4)$) and WENO5 with $\Delta t = 1/8$ and $\Delta t = 1$.

Bump on Tail (BoT)

$$f(t=0,x,v) = \left[\frac{0.9}{\sqrt{2\pi}}e^{-\frac{v^2}{2}} + \frac{0.2}{\sqrt{2\pi}}e^{-2(v-4.5)^2}\right](1+0.001\cos(0.5x))$$

$x \in [0, 20\pi]$, $v \in [-8, 8]$, $N_x = 135$, $N_v = 256$
Numerical estimation of $\max_n ||E^n||_\infty \approx 0.6$, we choose $\Delta t = \frac{C\Delta v}{0.6}$

BoT: numerical results

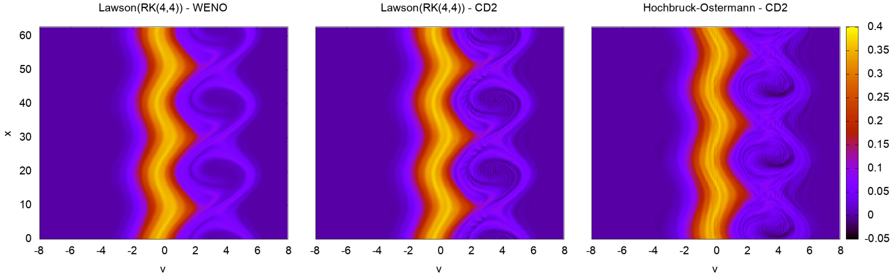


Figure: Distribution function at time $t = 40$ as a function of x and v for Lawson($RK(4,4)$) + WENO5 (left), Lawson($RK(4,4)$) + centered scheme (center), Hochbruck–Ostermann + centered scheme (right).

BoT: numerical results

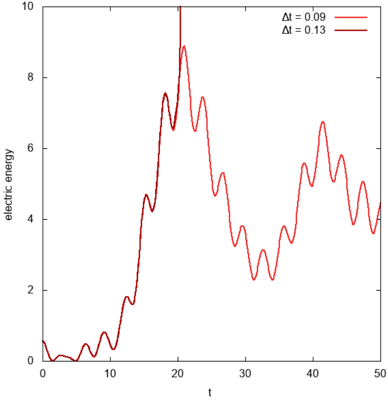


Figure: Illustration of the accuracy of the CFL estimate obtained from the linear theory. History of electric energy with Lawson($RK(4,4)$) + WENO5

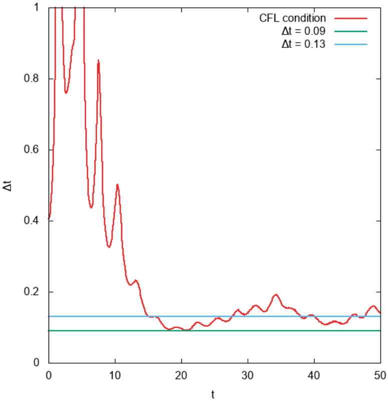
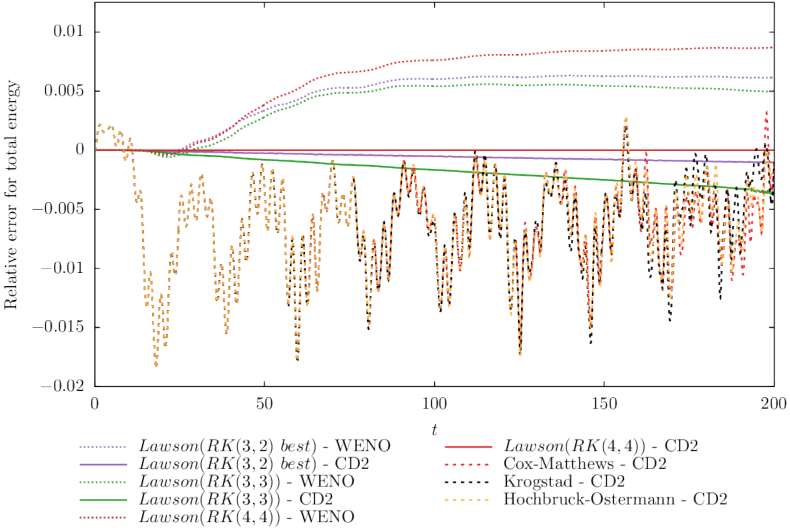


Figure: History of CFL condition for Lawson($RK(4,4)$) + WENO5 case

BoT: numerical results



Adaptive time step size

$\max_n ||E^n||_\infty$ **is not accessible in practice.**

To capture correctly the phenomena involved in the bump on tail test, we take the following time step size:

$$\Delta t_n = \min\left(0.1, \frac{C\Delta v}{||E^n||_\infty}\right)$$

with $C = y_{\text{max}}$ or σ from the linear theory.
➜Good estimate in practice for Lawson methods.

More Lawson methods

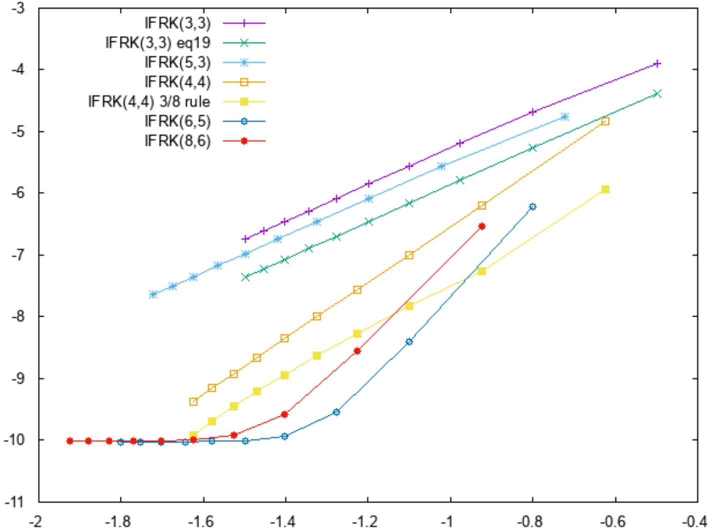
We are interested in the numerical cost $\dfrac{\Delta t}{s}$ of RK(s,n). To compare each time integrator, we compute total energy in Vlasov-Poisson system:

$$H(t) = \int_\Omega \int_{\mathbb{R}} v^2 f \,\mathrm{d}x\mathrm{d}v + \int_\Omega E^2 \,\mathrm{d}x$$

which is preserved in time. We propose to select the best method by considering:

$$h_{s,n} : \frac{\Delta t}{s} \mapsto \left\| \frac{H(t) - H(0)}{H(0)} \right\|_\infty$$

More Lawson methods



Outline

Drift-Kinetic equations

$f = f(t, r, \theta, z, v)$

$$\begin{cases} \underbrace{\partial_t f + v\partial_z f}_{\text{linear part}} \underbrace{- \frac{\partial_\theta \phi}{r}\partial_r f + \frac{\partial_r \phi}{r}\partial_\theta f - \partial_z \phi \partial_v f}_{\text{non linear part}} = 0 \\ -\left[\partial_r^2 \phi + \left(\frac{1}{r} + \frac{\partial_r n_0(r)}{n_0(r)}\right)\partial_r \phi + \frac{1}{r^2}\partial_\theta^2 \phi\right] + \frac{1}{T_e(r)}(\phi - \langle \phi \rangle) = \frac{1}{n_0(r)}\int_{\mathbb{R}} f \,\mathrm{d}v - 1 \end{cases}$$

$(r, \theta, z, v) \in [0.1, 14.5] \times [0, 2\pi] \times [0, L] \times \mathbb{R}$
After a Fourier transform in z, formally, the equation is still of the form of:

$$\partial_t f + ikvf + F(f) = 0$$

Compatible with all previous time integrators.
This is more complicated to use linear stability analysis, we use an other adaptive time step method.

Adaptive time step size (error estimate)

For adaptive time step size with any time integrator φ:

$$f^{n+1} = \varphi_{\Delta t_n}(f^n) \qquad ; \qquad \tilde{f}^{n+1} = \varphi_{\Delta t_n/2} \circ \varphi_{\Delta t_n/2}(f^n)$$

Richardson extrapolated numerical solution of the method of order p:

$$f_R^{n+1} = \frac{2^{p+1}\tilde{f}^{n+1} - f^{n+1}}{2^{p+1} + 1}$$

estimate of the local error:

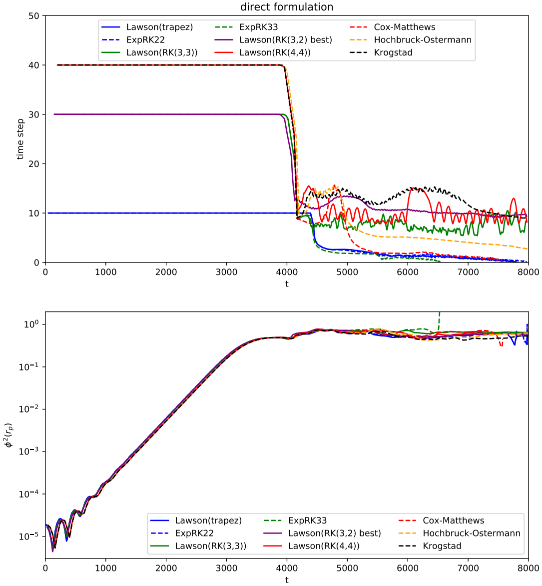
$$e_{n+1} = ||f_R^{n+1} - f^{n+1}||_{L^\infty} + \mathcal{O}(\Delta t_n^{p+2})$$

If $e_{n+1} > \text{tol}$: we reject the step and start again from time t_n. Else we determine the new time step size:

$$\Delta t_{new} = s\Delta t_n \left(\frac{\text{tol}}{e_{n+1}}\right)^{1/(p+1)}$$

$s = 0.8$ is safety factor.

Ion temperature gradient instability: numerical results



Outline

1. Motivation for Vlasov-Poisson equations
2. Linear analysis
   - Lawson methods
   - Exponential Runge-Kutta methods
3. Numerical simulation: Vlasov-Poisson equations
4. Numerical simulation: drift-kinetic equations
5. Conclusion
6. Future works

Conclusion

Summary

- Better understanding on stability of Lawson or ExpRK methods in transport equations
- Python script with `sympy` to compute estimates of CFL of Lawson – CD2, Lawson – WENO (5 or 3) or ExpRK – CD2 (with relaxing CFL)
- An adaptive time step size which works with any time integrators

Future works

- We can improve method with an embedded Runge-Kutta method (Dormand-Prince method, used in `ode45` of Matlab)
- Compare performance between exponential integrators and splitting methods (same stages/step, same order?)
- Use semi-Lagrangian method to remove dependency on periodic space (Fourier transform)

Outline

VPHL

$$\begin{cases} \partial_t f + v\partial_x f + E\partial_v f = 0 \\ \partial_x E = \int_{\mathbb{R}} f \,\mathrm{d}v - 1 \end{cases}$$

We linearized around an equilibrium and we suppose:

$$f(t=0,x,v) = \underbrace{f_c(v)}_{(1-\alpha)\delta_0(v)} + \underbrace{f_h(x,v)}_{\mathcal{M}_{[\alpha/2,u,1]}+\mathcal{M}_{[\alpha/2,-u,1]}}$$

and add Ampère equation to obtain:

$$\begin{cases} \partial_t u_c = E \\ \partial_t E = -\rho_c u_c - \int v f_h \,\mathrm{d}v \\ \partial_t \hat{f}_h = -ikv\hat{f}_h - \widehat{E\partial_v f_h} \end{cases}$$

Schemes

3 possibilities to built a scheme:

- Full-kinetic model with $f_0(x, v) = \mathcal{M}_{[(1-\alpha),0,T_c]} + \mathcal{M}_{[\alpha/2,u,1]} + \mathcal{M}_{[\alpha/2,-u,1]}$ with $T_c \ll 1$.
- Hybrid version with splitting method.
- Hybrid version with Lawson method.

Hybrid splitting

$\varphi_{\Delta t}^{[a]}$:

$$\begin{cases} \partial_t f_h + v \partial_x f_h = 0 \\ \partial_t u_c = 0 \\ \partial_t E = -\int v f_h \, \mathrm{d}v \end{cases}$$

$\varphi_{\Delta t}^{[b]}$:

$$\begin{cases} \partial_t f_h + E \partial_v f_h = 0 \\ \partial_t u_c = E \\ \partial_t E = 0 \end{cases}$$

$\varphi_{\Delta t}^{[c]}$:

$$\begin{cases} \partial_t f_h = 0 \\ \partial_t u_c = 0 \\ \partial_t E = -\rho_c u_c \end{cases}$$

$$U^{n+1} = \varphi_{\Delta t}^{[a]} \circ \varphi_{\Delta t}^{[b]} \circ \varphi_{\Delta t}^{[c]} (U^n)$$

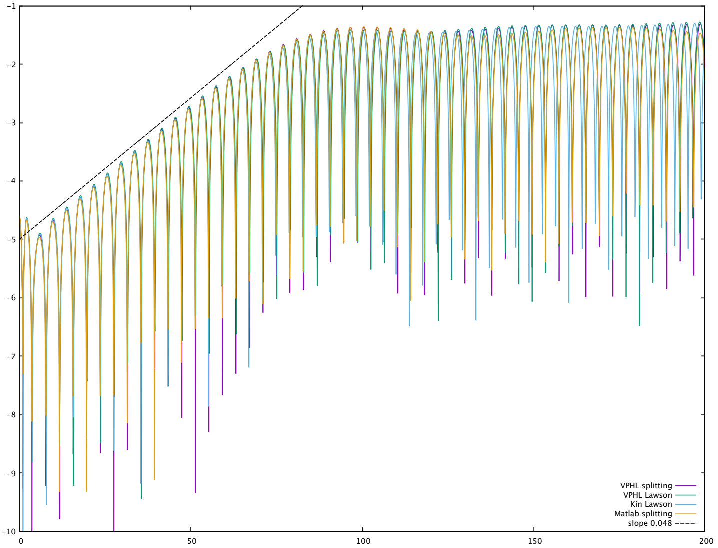
Hybrid Lawson

$$\partial_t U = AU + N(U)$$

with:

$$U = \begin{pmatrix} u_c \\ E \\ \hat{f}_h \end{pmatrix} \quad A = \begin{pmatrix} 0 & 1 & 0 \\ -\rho_c & 0 & 0 \\ 0 & 0 & -ikv \end{pmatrix} \quad N(U) = \begin{pmatrix} 0 \\ -\int v f_h \,\mathrm{d}v \\ -\widehat{E\partial_v f_h} \end{pmatrix}$$

Numerical result



Conclusion of future works

- A Python script to compute slope of dispersion relation of *any* input distribution.
- Kinetic, and hybrid simulation converge.
- Two hybrid simulations to compare (it's the future works to do!)

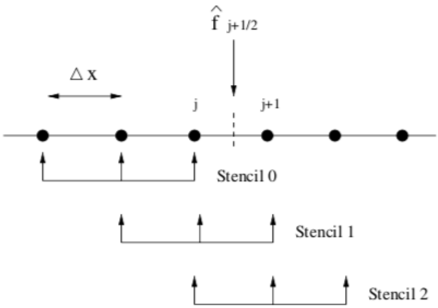
Thank you for your attention

For more questions, I will be at the sauna

Backup

WENO5 method

Weighted **E**ssentially **N**on-**O**scillatory method of order 5: 3 estimates on 3 different stencils weighted with nonlinear weights.



3 steps:

1. Indicator of smoothness
2. Weights
3. Flux

WENO5 method

Indicator of smoothness $\beta_i^{\pm}$

To approximate $\partial_x f(u)$:

Split f as:

$$f(u) = f^{+}(u) + f^{-}(u) \quad , \quad \frac{df^{+}}{du} \geq 0 \text{ et } \frac{df^{-}}{du} \leq 0$$

Indicators of smoothness:

$$\beta_i^{\pm} \leftarrow (f^{\pm}_{[\![j-2,j+3]\!]}) \ , \ i = 0, 1, 2$$

Approximations of derivatives of order 1 and 2 on 3 stencils.

WENO5 method

$$
\begin{aligned}
\beta_0^+ &= \frac{13}{12}\left(f_{j-2}^+ - 2f_{j-1}^+ + f_j^+\right)^2 + \frac{1}{4}\left(f_{j-2}^+ - 4f_{j-1}^+ + 3f_j^+\right)^2\\
\beta_1^+ &= \frac{13}{12}\left(f_{j-1}^+ - 2f_j^+ + f_{j+1}^+\right)^2 + \frac{1}{4}\left(f_{j-1}^+ - f_{j+1}^+\right)^2\\
\beta_2^+ &= \frac{13}{12}\left(f_j^+ - 2f_{j+1}^+ + f_{j+2}^+\right)^2 + \frac{1}{4}\left(3f_j^+ - 4f_{j+1}^+ + f_{j+2}^+\right)^2
\end{aligned}
$$

$$
\begin{aligned}
\beta_0^- &= \frac{13}{12}\left(f_{j+1}^- - 2f_{j+2}^- + f_{j+3}^-\right)^2 + \frac{1}{4}\left(3f_{j+1}^- - 4f_{j+2}^- + f_{j+3}^-\right)^2\\
\beta_1^- &= \frac{13}{12}\left(f_j^- - 2f_{j+1}^- + f_{j+2}^-\right)^2 + \frac{1}{4}\left(f_j^- - f_{j+2}^-\right)^2\\
\beta_2^- &= \frac{13}{12}\left(f_{j-1}^- - 2f_j^- + f_{j+1}^-\right)^2 + \frac{1}{4}\left(f_{j-1}^- - 4f_j^- + 3f_{j+1}^-\right)^2
\end{aligned}
$$

WENO5 method

Weights $w_i^{\pm}$

Unnormalized weights:

$$\alpha_i^{\pm} \leftarrow \frac{\gamma_i}{(\epsilon + \beta_i^{\pm})^2} \ , \ \gamma_i \in \mathbb{R}_+^* \ : \sum_k \gamma_k = 1$$

where $\gamma_0 = \frac{1}{10}, \gamma_1 = \frac{6}{10}, \gamma_2 = \frac{3}{10}$. Parameter $\epsilon = 10^{-6}$
Linearized weights (LW5): $\alpha_i^{\pm} = \gamma_i + \mathcal{O}(\Delta x^2)$
Normalized weights:

$$w_i^{\pm} \leftarrow \frac{\alpha_i^{\pm}}{\sum_k \alpha_k^{\pm}}$$

WENO5 method

Flux $f^{\pm}_{i+\frac{1}{2}}$

$$f^+_{j+\frac{1}{2}} \leftarrow w_0^+ \left(\frac{2}{6} f^+_{j-2} - \frac{7}{6} f^+_{j-1} + \frac{11}{6} f^+_j\right) + w_1^+ \left(-\frac{1}{6} f^+_{j-1} + \frac{5}{6} f^+_j + \frac{2}{6} f^+_{j+1}\right) + w_2^+ \left(\frac{2}{6} f^+_j + \frac{5}{6} f^+_{j+1} - \frac{1}{6} f^+_{j+2}\right)$$

$$f^-_{j+\frac{1}{2}} \leftarrow w_2^- \left(-\frac{1}{6} f^-_{j-1} + \frac{5}{6} f^-_j + \frac{2}{6} f^-_{j+1}\right) + w_1^- \left(\frac{2}{6} f^-_j + \frac{5}{6} f^-_{j+1} - \frac{1}{6} f^-_{j+2}\right) + w_0^- \left(\frac{11}{6} f^-_{j+1} - \frac{7}{6} f^-_{j+2} + \frac{2}{6} f^-_{j+3}\right)$$

$$\boxed{(\partial_x f(u))_j \approx \frac{1}{\Delta x}\left[(f^+_{j+\frac{1}{2}} - f^+_{j-\frac{1}{2}}) + (f^-_{j+\frac{1}{2}} - f^-_{j-\frac{1}{2}})\right]}$$