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 4, 2020

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:

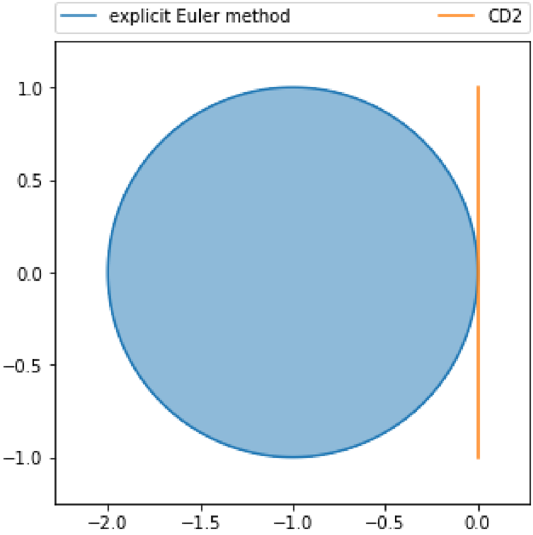
$$\partial_t u + \partial_x u = 0$$

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:

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

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:

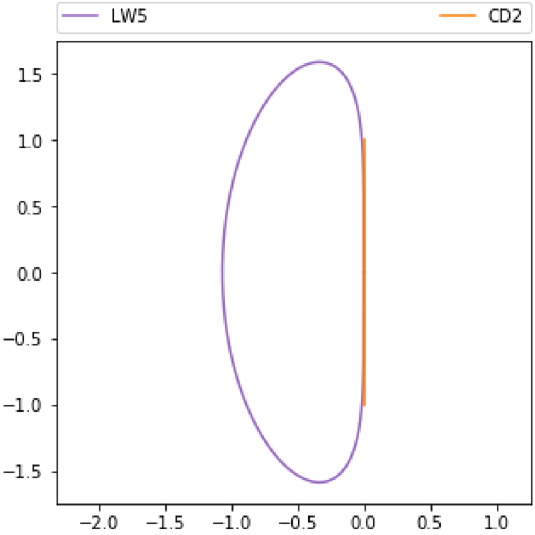
- CD2 (centered difference of order 2): $(\partial_v f)(v_j) \approx \dfrac{f_{j+1} - f_{j-1}}{2\Delta v}$
- WENO5 (weighted essentially non-oscillatory of order 5):
  - WENO5: non linear scheme: ~~Von Neumann analysis~~
  - LW5 (linearized WENO5): linear scheme (this is Lagrange interpolation of order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

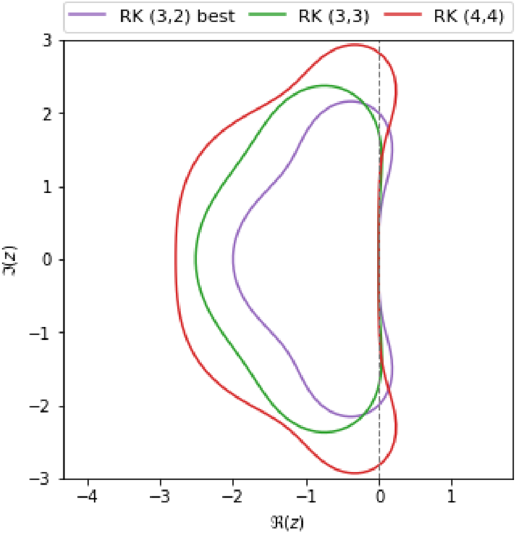
$$v^{n+1} = \underbrace{p(\lambda\Delta t)}_{\text{stability function of RK}} v^n$$

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:

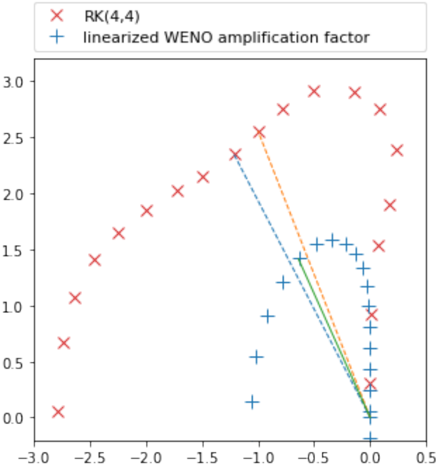
$$|p(iy)| = 1, \quad y \in \mathbb{R}$$

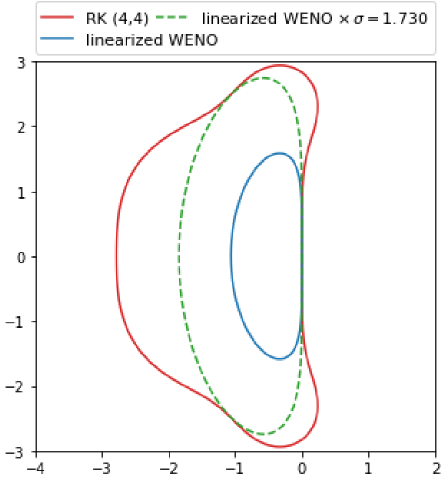
Methods	Lawson($RK(3,2)$ $best$)	Lawson($RK(3,3)$)	Lawson($RK(4,4)$)
$y_{\max}$	2	$\sqrt{3}$	$2\sqrt{2}$

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):

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

Stability function becomes:

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

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

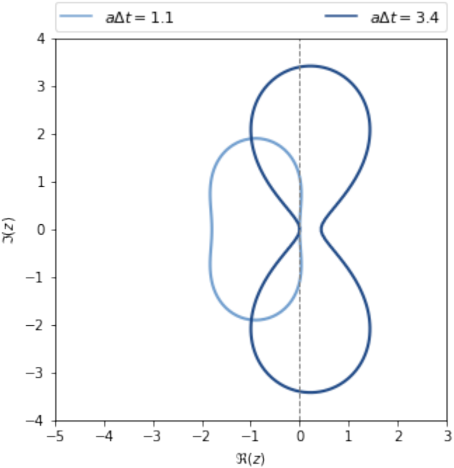


Figure: Stability domain of ExpRK(2,2) for $a\Delta t \in \{1.1, 3.4\}$

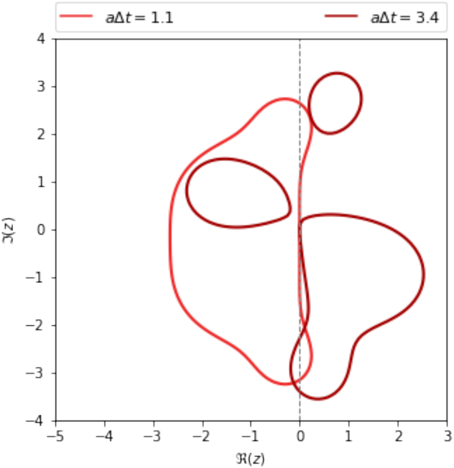


Figure: Stability domain of Cox-Matthews for $a\Delta t \in \{1.1, 3.4\}$

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

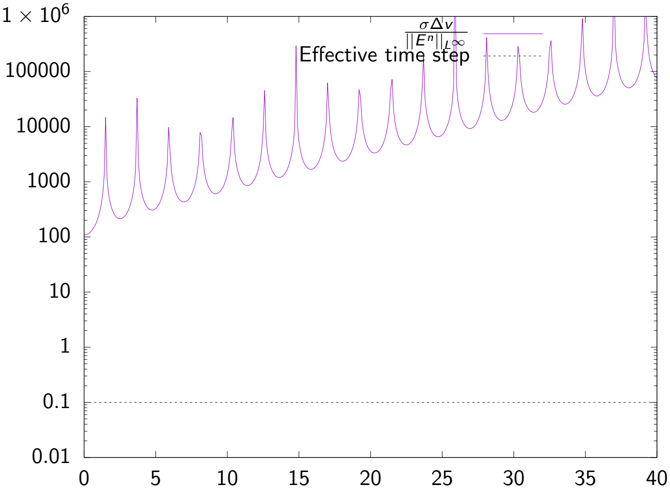


Figure: Landau damping test: time history of the CFL condition (semi-log scale).

Landau damping: numerical results

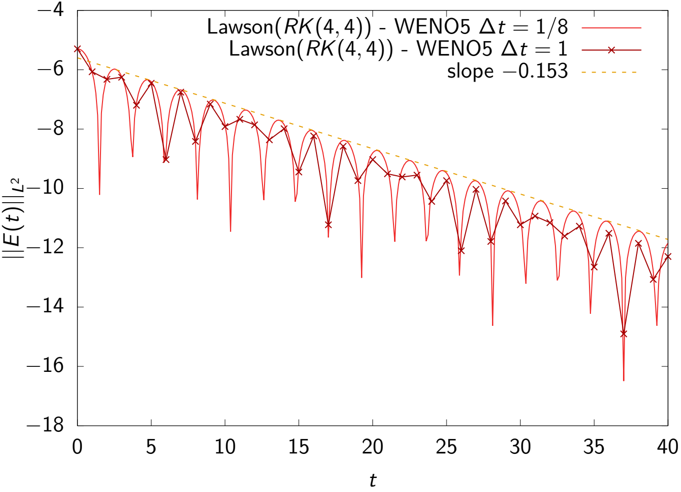


Figure: Landau damping test: time history of $\|E(t)\|_{L^2}$ (semi-log scale) obtained with Lawson($RK(4,4)$) and WENO5 with $\Delta t = 1/8$ and $\Delta t = 1$.

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

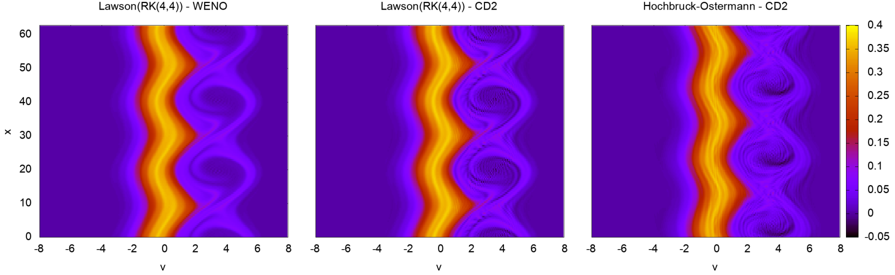


Figure: Distribution function at time $t = 40$ as a function of x and v for Lawson($RK(4,4)$) + WENO5 (left), Lawson($RK(4,4)$) + centered scheme (center), Hochbruck–Ostermann + centered scheme (right).

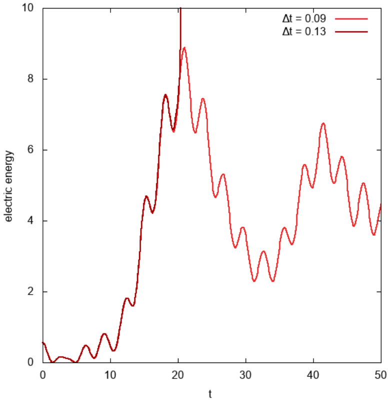


Figure: Illustration of the accuracy of the CFL estimate obtained from the linear theory. History of electric energy with Lawson($RK(4,4)$) + WENO5

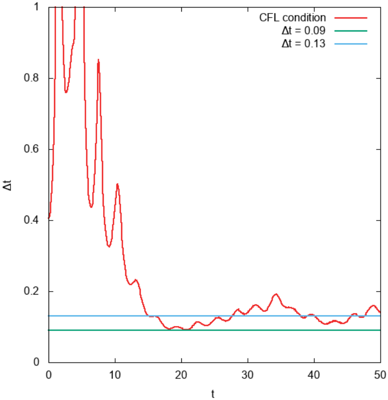
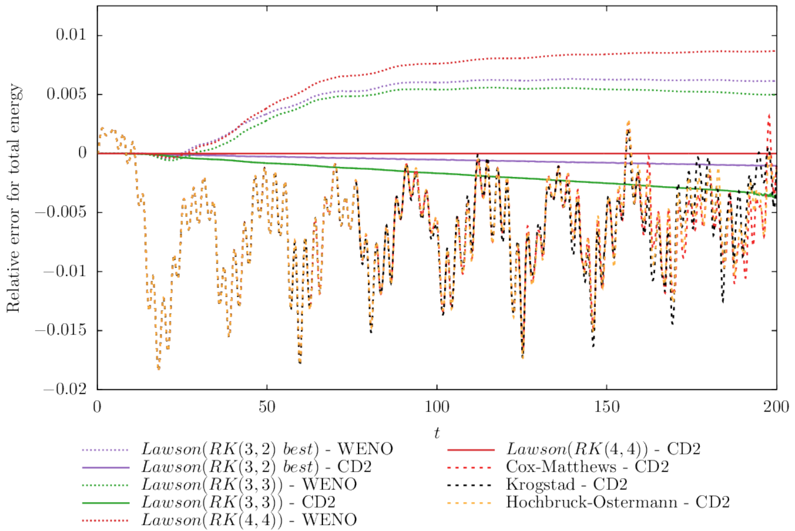


Figure: History of CFL condition for Lawson($RK(4,4)$) + WENO5 case

BoT: numerical results



Outline

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)

Thank you for your attention

Backup