SMR 1673/46

AUTUMN COLLEGE ON PLASMA PHYSICS

5 - 30 September 2005

Numerical Vlasov Simulations: Problems and Applications

B. Eliasson

Ruhr University, Bochum, Germany

Autumn College on Plasma Physics, Trieste, 5–30 September 2005

Numerical Vlasov Simulations: Problems and Applications

BENGT ELIASSON

RUHR-UNIVERSITÄT BOCHUM, GERMANY

Outline

A. Some elements of numerical theory

B. Vlasov simulations – problems with filamentation

C. Fourier method in velocity space. 1D and 2D. Outflow boundary conditions

D. Examples of numerical simulations

- ☞ Kinetic tunnelling (recurrence) effects
- ☞ Electron Bernstein waves (Bernstein-Landau paradox)
- ☞ Electromagnetic waves
- ☞ Coupling of nonlinear wave

Solution of nonlinear equations

Ex: In linear theory, we often want to solve a dispersion $f(\omega) = 0$, where f can be a high-order polynomial of or some more complicated function of ω. The standard method to do this is Newton's (Newton-Raphson's) method: First make a reasonable guess of the root, say $\omega = \omega^{(0)}$.

Then update ω iteratively as

$$\omega \leftarrow \omega + \Delta\omega$$

where $\Delta\omega = -f(\omega)/f'(\omega)$. The process converges when $|\Delta\omega|$ is small enough.

Solution of nonlinear equations

Example: We want to find the root to $f(\omega) = \cos(\omega) - \omega = 0$. Here, $f'(\omega) = -\sin(\omega) - 1$ and hence $\Delta\omega = (\cos(\omega) - \omega)/(\sin(\omega) + 1)$. Let the first guess be $\omega = \omega^{(0)} = 1$. Then we have

1. $\Delta\omega = -0.24963613215976, \quad \omega = 1 - 0.24963613215976 = 0.75036386784024$
2. $\Delta\omega = -0.01125097692888, \quad \omega = 0.73911289091136$
3. $\Delta\omega = -0.00002775752607, \quad \omega = 0.73908513338528$
4. $\Delta\omega = -0.00000000017012, \quad \omega = 0.73908513321516$

and we have the solution $\omega = 0.739085133$ with 9 significant digits. Newton's methods converges extremely fast (it doubles the number of significant digits in each iteration) if the initial guess is good enough and if $f'(\omega^*) \neq 0$, where ω^* is the exact solution. If $f'(\omega^*) = 0$, then Newton's method converges more slowly. Note that Newton's method also works for *complex* ω (which is common in applications!)

Solution of nonlinear systems of equations

Nonlinear systems of equations $\vec{F}(\vec{x}) = 0$ can also be solved with Newton's method: First make a reasonable guess of the root, say $\vec{x} = \vec{x}^{(0)}$. Then update $\vec{x}$ iteratively as

$$\vec{x} \leftarrow \vec{x} + \Delta\vec{x}$$

where the correction term $\Delta\vec{x}$ is obtained by solving the *linear* equation system $\vec{\vec{J}}\Delta\vec{x} = -\vec{f}(\vec{x})$, where $\vec{\vec{J}} = \frac{\partial \vec{f}}{\partial \vec{x}}$ is the Jacobian matrix of $\vec{F}$. The process converges when the norm $||\Delta\vec{x}||$ is small enough.

Solution of nonlinear systems of equations

Here, we have used the matrix notation

$$\vec{F}(\vec{x}) = \begin{bmatrix} F_1 \\ F_2 \\ \vdots \\ F_N \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix}, \quad \Delta\vec{x} = \begin{bmatrix} \Delta x_1 \\ \Delta x_2 \\ \vdots \\ \Delta x_N \end{bmatrix} \tag{1}$$

and

$$\vec{\vec{J}} = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_N} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & & \vdots \\ \vdots & & \ddots & \\ \frac{\partial F_N}{\partial x_1} & \cdots & & \frac{\partial F_N}{\partial x_N} \end{bmatrix} \tag{2}$$

The norm can for example be the Euclidian norm
$||\Delta x|| = \sqrt{(\Delta x_1)^2 + (\Delta x_2)^2 + \cdots + (\Delta x_N)^2}$

Solution of nonlinear systems of equations

Example: We want to solve the nonlinear system of equations

$$F_1(x_1, x_2) = x_1 x_2 + \sin(x_2) - 0.1 = 0 \tag{3}$$

$$F_2(x_1, x_2) = x_1 + x_2 \exp(x_1) = 0 \tag{4}$$

The Jacobian matrix is

$$\vec{J} = \frac{\partial \vec{F}}{\partial \vec{x}} = \begin{bmatrix} x_2 & x_1 + \cos(x_2) \\ 1 + x_2 \exp(x_1) & \exp(x_1) \end{bmatrix} \tag{5}$$

First give some values on x_1 and x_2, say $x_1 = x_2 = 0.1$. To obtain values on Δx_1 and Δx_2, solve the *linear* system of equations

$$x_2 \Delta x_1 + (x_1 + \cos(x_2))\Delta x_2 = -(x_1 x_2 + \sin(x_2) + 1) \tag{6}$$

$$(1 + x_2 \exp(x_1))\Delta x_1 + \exp(x_1)\Delta x_2 = x_1 + x_2 \exp(x_1) \tag{7}$$

and then update $x_1 \leftarrow x_1 + \Delta x 1$ and $x_2 \leftarrow x_2 + \Delta x_2$, etc.

Solution of nonlinear systems of equations (continued)

The first iterations give:

1. $x_1 = -0.09868723775962$, $x_2 = 0.10916462918347$, $||\Delta x|| = 0.19889848887465$
2. $x_1 = -0.10077711054130$, $x_2 = 0.11145795777505$, $||\Delta x|| = 0.00310272852060$
3. $x_1 = -0.10077814555465$, $x_2 = 0.11146377694368$, $||\Delta x|| = 0.0000059104971192$

We thus have the solution $x_1 = -0.100778$ and $x_2 = 0.111463$ after three iterations.

Some theory in numerical simulations

A numerical scheme for a well-posed problem should

- ❑ be consistent with the problem, i.e. it should approximate the problem locally as the step size (space and time) goes to zero.
- ❑ be stable, which means that the numerical solution should remain bounded as the step size goes to zero.
- ❑ converge to the solution when the timestep goes to zero.

It is also desirable that the numerical is as accurate (high order) as possible.

Well-posedness, example 1

The problem

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} = 0, \quad f(0) = 0 \tag{1}$$

is well-posed, while the problem

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} = 0, \quad f(1) = 0$$

is not. Proof of (1); non-increasing energy norm:

$$\frac{d}{dt}||f||^2 = \frac{d}{dt}\int_0^1 f^2\,dx = \int_0^1 2f\frac{\partial f}{\partial t}\,dx =$$

$$-\int_0^1 2f\frac{\partial f}{\partial x} = \int_0^1 \frac{\partial f^2}{\partial x} = -(f(1)^2 - f(0)^2) = -f(1)^2 \leq 0$$

Well-posedness, example 2

The problem

$$\frac{\partial f}{\partial t} = \frac{\partial^2 f}{\partial x^2}, \quad f(0) = f(1) = 0 \tag{1}$$

is well-posed while the problem

$$\frac{\partial f}{\partial t} = -\frac{\partial^2 f}{\partial x^2}, \quad f(0) = f(1) = 0$$

is not. Proof of (1): similar as in previous example.

Consistency, example

Discretize the problem

$$\frac{\partial f}{\partial t} + \frac{\partial f}{\partial x} = 0, \quad f(1) = f(0) \tag{1}$$

as $f(x_j, t_k) \approx f_j^k$ where $x_j = j\Delta x$, $j = 0,\, 1,\, N-1$, $\Delta x = L/N$, $t^m = m\Delta t$, $m = 0,\, 1,\, M-1$, $\Delta t = T/M$, and the scheme

$$f_j^k = f_j^{k-1} - \frac{\Delta t}{\Delta x}(f_j^{k-1} - f_{j-1}^{k-1})$$

The scheme is consistent since $f_j^{k-1} \approx f(x, t-\Delta t) \approx f(x,t) - \Delta t f_t(x,t)$ and
$f_{j-1}^k \approx f(x-\Delta x, t) \approx f(x,t) - \Delta x f_x(x,t)$ and
$f_{j-1}^{k-1} \approx f(x-\Delta x, t-\Delta t) \approx f(x,t) - \Delta x f_x(x,t) - \Delta t f_t(x,t)$, and we recover Eq. (1) as Δt, $\Delta x \to 0$ (with $\Delta t/\Delta x = \lambda =$constant).

Stability, von Neumann analysis, example

Apply the numerical scheme

$$f_j^k = f_j^{k-1} - \frac{\Delta t}{\Delta x}(f_j^{k-1} - f_{j-1}^{k-1})$$

on the function $f_j^k = g^k \exp(iKx_j)$ where K is real constant while g is complex constant. This gives (after eliminating common factors)

$$1 = g^{-1} - \frac{\Delta t}{\Delta x}(g^{-1} - g^{-1}\exp(-iK\Delta x))$$

Solving for g, we have

Stability, von Neumann analysis, example

$$g = 1 - \lambda(1 - \exp(-iK\Delta x))$$

where $\lambda = \Delta t/\Delta x$. The scheme is stable if $|g| < 1$ and unstable (useless) if $|g| > 1$ for $-\pi \leq K\Delta x \leq \pi$. Here,

$$|g|^2 = 1 + 2\lambda\,(\lambda - 1)\,[1 - \cos(K\Delta x)]$$

and hence $|g| \leq 1$ for $\lambda = \Delta t/\Delta x \leq 1$.

Convergence

Convergence is more difficult to show in general, but fortunately we have the following *Lax-Richtmyer equivalence theorem*:

> A consistent finite difference scheme for a partial (or ordinary) differential equation, for which the initial problem is well-posed, converges to the true solution if and only if it is stable.

(see J. Strikwerda, *Finite Difference Schemes and Partial Differential Equations* for a formal proof)

So we only have to worry about consistency, well-posedness and numerical stability – the convergence follows.

Some popular numerical methods

$$\frac{\partial u}{\partial t} = f(u, t)$$

Leapfrog scheme:

$$u^{i+1} \leftarrow u^{i-1} + 2\Delta t f(u^i, t^i)$$

Advantages: Simple to implement, is fast. Suitable for Hamiltonian systems like particle systems and Maxwell equations etc. Non-dissipative, symplectic integrator. Very efficient variants (Yee scheme) for Maxwell equations.
Disadvantage: Not suitable (unstable) for dissipative equations. Multi-step method, initial conditions on the two first time-steps.

Some popular numerical methods (continued)

4th order Runge-Kutta:

$$
\begin{aligned}
&k_1 \leftarrow f(u^i, t^i)\\
&k_2 \leftarrow f(u^i + k_1\Delta t/2, t + \Delta t/2)\\
&k_3 \leftarrow f(u + k_2\Delta t/2, t^i + \Delta t/2)\\
&k_4 \leftarrow f(u^i + k_3\Delta t, t^i + \Delta t)\\
&u^{i+1} \leftarrow u^i + (\Delta t/6)(k_1 + 2k_2 + 2k_3 + k_4)
\end{aligned}
$$

Advantages: Simple to implement, robust. Suitable for both Hamiltonian systems and dissipative systems. High accuracy. Disadvantage: f must be calculated four times per timestep.

Finite difference methods

To approximate 1st spatial derivative:

$$\frac{\partial f}{\partial x} \approx \frac{f_{j+1} - f_{j-1}}{2\Delta x} \tag{8}$$

To approximate 2nd derivative:

$$\frac{\partial^2 f}{\partial x^2} \approx \frac{f_{j+1} - 2f_j + f_{j-1}}{\Delta x^2} \tag{9}$$

where $x = j\Delta_x$, $j = 0,\ 1,\ \ldots N_x$, and $\Delta x = L/N_x$.

Pseudo-spectral methods

Pseudo-spectral methods are used to approximate x derivatives and are based on trigonometric interpolation

$$f(x) \approx \phi(x) = \sum_{j=-(N_x/2-1)}^{N_x/2} \widehat{\phi}_j \exp(ik_j x) \tag{10}$$

where $k_j = 2\pi j/L$ and $0 \leq x \leq L$. Differentiation of the interpolating polynomial $\phi(x)$ gives

$$\frac{\partial \phi(x)}{\partial x} = \sum_{j=-(N_x/2-1)}^{N_x/2} ik_j \widehat{\phi}_j \exp(ik_j x) \approx \frac{\partial f(x)}{\partial x} \tag{11}$$

Pseudo-spectral method (continued)

The weights $\widehat{\phi}_j$ are obtained from the Discrete Fourier Transform (DFT)

$$\widehat{\phi}_j = \frac{1}{N_x} \sum_{m=0}^{N_x-1} \phi(x_m) \exp\left(- i2\pi m \frac{j}{N_x} \right) \tag{12}$$

Using the Fast Fourier Transform algorithm, the x derivatives are approximated as

$\widehat{\phi} = \mathrm{FFT}(\phi)$ Make DFT
$\widehat{\psi} = ik\widehat{\phi}$ Multiply by ik
$\phi_x = \mathrm{IFFT}(\widehat{\psi})$ Make inverse DFT

Normally the accuracy is superior compared to finite difference methods, except for problems having discontinuities. Drawback: Requires periodic solutions.

Example, simulation in Matlab

main.m:

```
clear
N1=400;  % Number of x intervals
Nt=8000;  % Number of time steps
Nprints=200; % Number of times to save data
interval=Nt/Nprints;
L1=20000;  % box length
dx1=L1/N1; % Delta x
x1=(0:(N1-1))*L1/N1-L1/2; % x
dt=0.5; % Time step
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
alpha=0.25; % n_e0/n_i0
eta=0.1;    % Ti/Te
```

Example, simulation in Matlab (Continued)

main.m (continued):

```
%%%% Create Fourier weights %%%
  % Calculate wavenumber k
  jj=(0:(N1/2-1));
  k1=2*pi/(N1*dx1)*jj;  % Obs Real-valued!
  k_minus=2*pi/(N1*dx1)*(jj-ones(1,N1/2)*N1/2);
  k1=[k1 k_minus];
  % Calculate k^2
  kk1=k1.*k1;

%%% Initial conditions %%%%%%%%%%%
  for i1=1:N1
     N(i1)=1.5-0.5*tanh(3*sin(2*pi*x1(i1)/L1)+1.5);
     u(i1)=F(N(i1),eta,alpha);
  end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

Example, simulation in Matlab (Continued)

main.m (continued):

```
for j=1:Nt
  [N,u]=RungeKutta(N,u,dt,k1,kk1,eta,alpha);
  if mod(j,interval)==0
    subplot(2,1,1)
    plot(x1,real(N));
    title('Density')
    subplot(2,1,2)
    plot(x1,real(u));
    title('Velocity')
    pause(0.01);
  end
end
```

Implementation in Matlab, Runge-Kutta

RungeKutta.m:

```
function [N,u]=RungeKutta(N, u,dt,k,kk1,eta,alpha)

  [R1_N, R1_u]=f(N, u,k,kk1,eta,alpha);
  [R2_N, R2_u]=f(N+0.5*dt*R1_N, u+0.5*dt*R1_u,k,kk1,eta,alpha);
  [R3_N, R3_u]=f(N+0.5*dt*R2_N, u+0.5*dt*R2_u,k,kk1,eta,alpha);
  [R4_N, R4_u]=f(N+dt*R3_N, u+dt*R3_u,k,kk1,eta,alpha);

 N=N+dt/6*(R1_N+2*R2_N+2*R3_N+R4_N);
 u=u+dt/6*(R1_u+2*R2_u+2*R3_u+R4_u);
```

Implementation in Matlab, the right-hand side

f.m:

```
function [R_N,R_u]=f(N, u, k,kk1,eta,alpha)
  diss=3;
  R_N=real(-d1(N.*u,k)+10*d2(N,kk1));
  R_u=real(-d1(u.^2/2+log(N+alpha-1)+1.5*eta*N.^2,k)+10*d2(u,kk1));
```

Solves the system (dust ion-acoustic waves)

$$\frac{\partial N}{\partial t} = -\frac{\partial (Nu)}{\partial x}$$
$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}(u^2/2 + \log(N + \alpha - 1) + 1.5\eta N^2)$$

with some numerical dissipation. (Eliasson and Shukla, Phys. Plasmas 12, 024502/1-4)

Calculation of derivatives

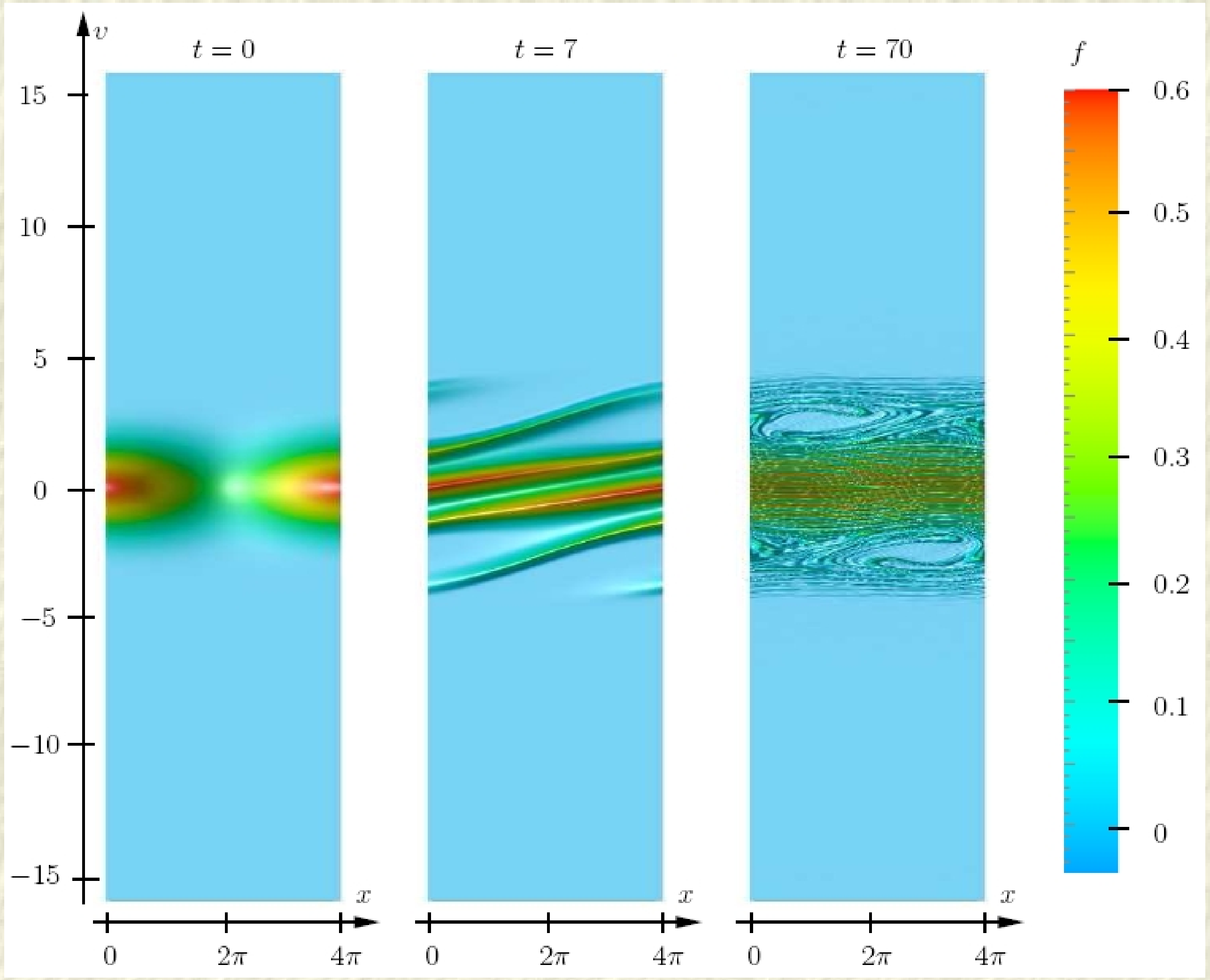
d1.m

```
%Function for calculating d/dx.
function d1y=d1(y,k)

  d1y=fft(y);     % Make FFT
  d1y=i*d1y.*k;   % Multiply by i*k element-wise
  d1y=ifft(d1y); % Make inverse FFT
```

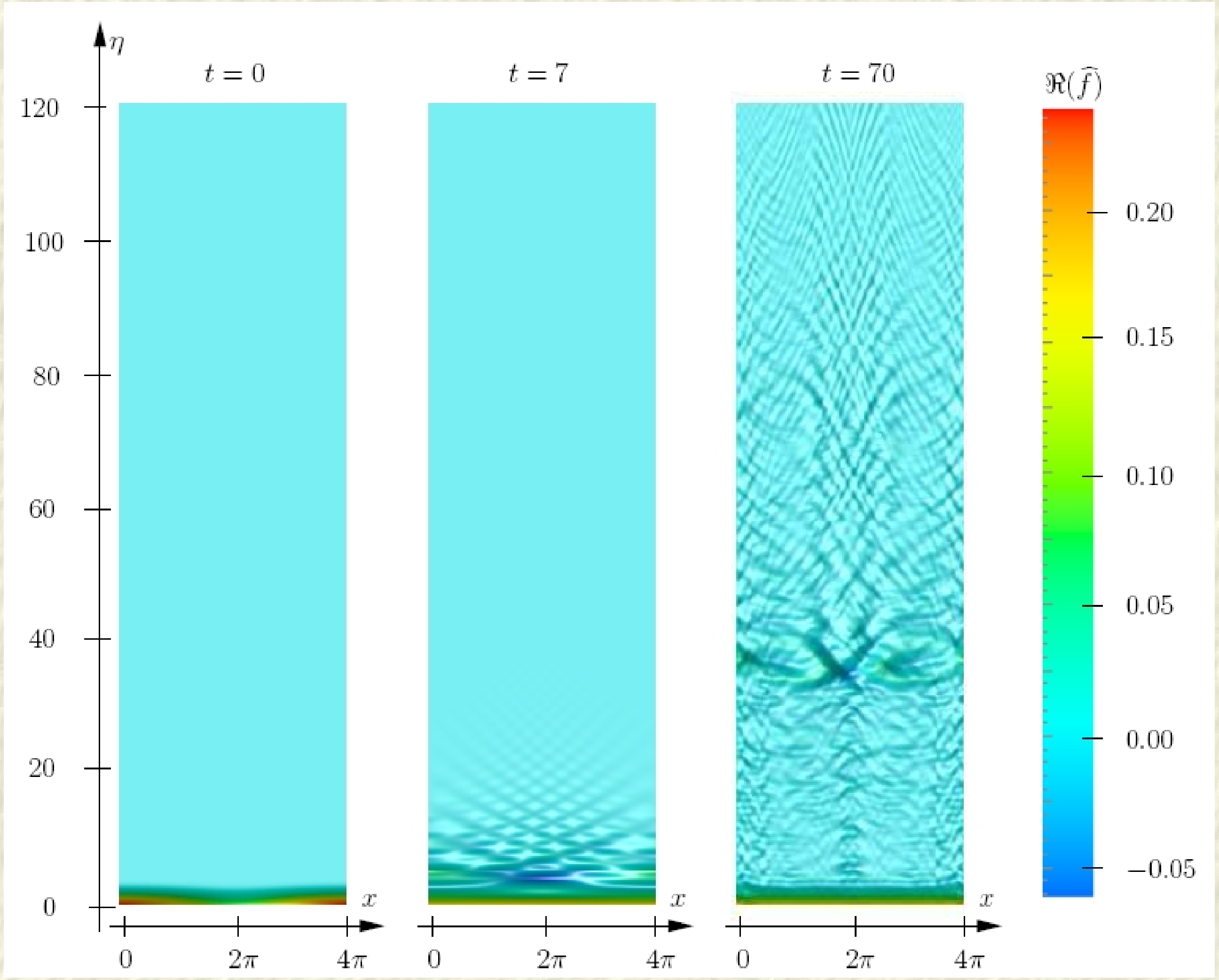
d2.m

```
% Function for calculating d^2/dx^2.
function d2=d2(y,kk1)

  d2=fft(y);     % Make FFT
  d2=-d2.*kk1;   % Multiply by -k^2 elementwise
  d2=ifft(d2);   % Make inverse FFT
```

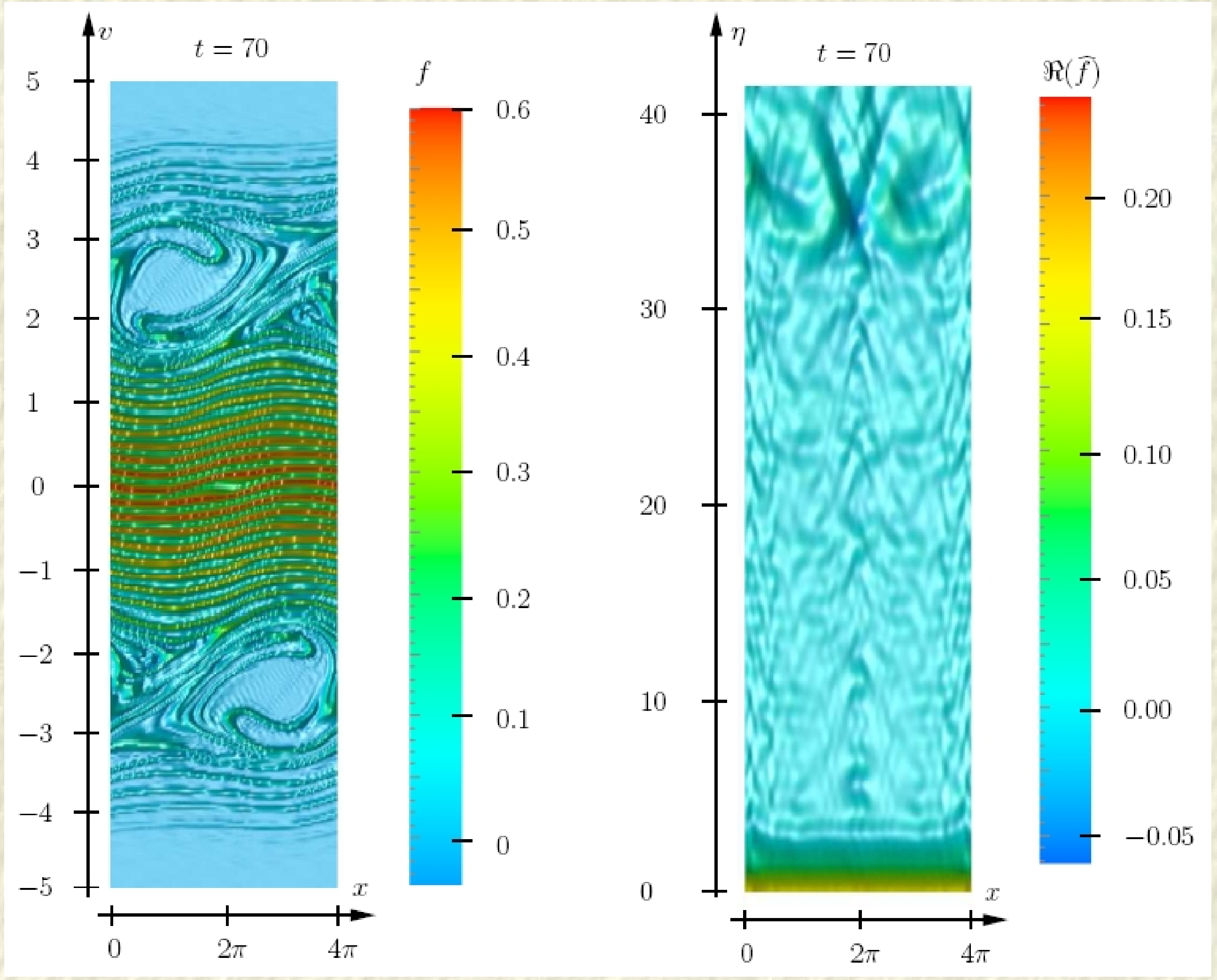
Electron phase space distribution



Fourier transformed velocity space



Closeup of solution



Fourier transformed Vlasov-Poisson system

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} - E\frac{\partial f}{\partial v} = 0, \qquad \frac{\partial E}{\partial x} = 1 - \int_{-\infty}^{\infty} f(x,v,t)\,dv \tag{13}$$

The Fourier transform pair

$$f(x,v,t) = \int_{-\infty}^{\infty} \widetilde{f}(x,\eta,t)\mathrm{e}^{-i\eta v}\,d\eta, \qquad \widetilde{f}(x,\eta,t) \quad = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(x,v,t)e^{i\eta v}\,dv \tag{14}$$

gives

$$\frac{\partial \widetilde{f}}{\partial t} - i\frac{\partial^2 \widetilde{f}}{\partial x \partial \eta} + \ E\eta\widetilde{f} = 0, \qquad \frac{\partial E(x,t)}{\partial x} = 1 - 2\pi\widetilde{f}(x,\eta,t)_{\eta=0} \tag{15}$$

Boundary conditions

Study the reduced problem

$$\frac{\partial \widetilde{f}}{\partial t} - i\frac{\partial^2 \widetilde{f}}{\partial x \partial \eta} = 0 \tag{16}$$

Fourier transform i space ($\partial/\partial x \to ik$) gives

$$\frac{\partial \widehat{f}}{\partial t} + k\frac{\partial \widetilde{f}}{\partial \eta} = 0 \tag{17}$$

where we know that a well-posed boundary condition is to set $\widehat{f}$ to zero at $\eta = \eta_{max}$ if $k < 0$. This idea has been used for 1D and 2D (2 spatial and 2 velocity dimensions) Vlasov equation. (B. Eliasson: J. Scientific Computing, 16(1), pp. 1-28 (2001); J. Computational Physics, 181(1), pp. 98-125 (2002); Computer Physics Communications 170(2), pp. 205-230)

Numerical recurrence effects

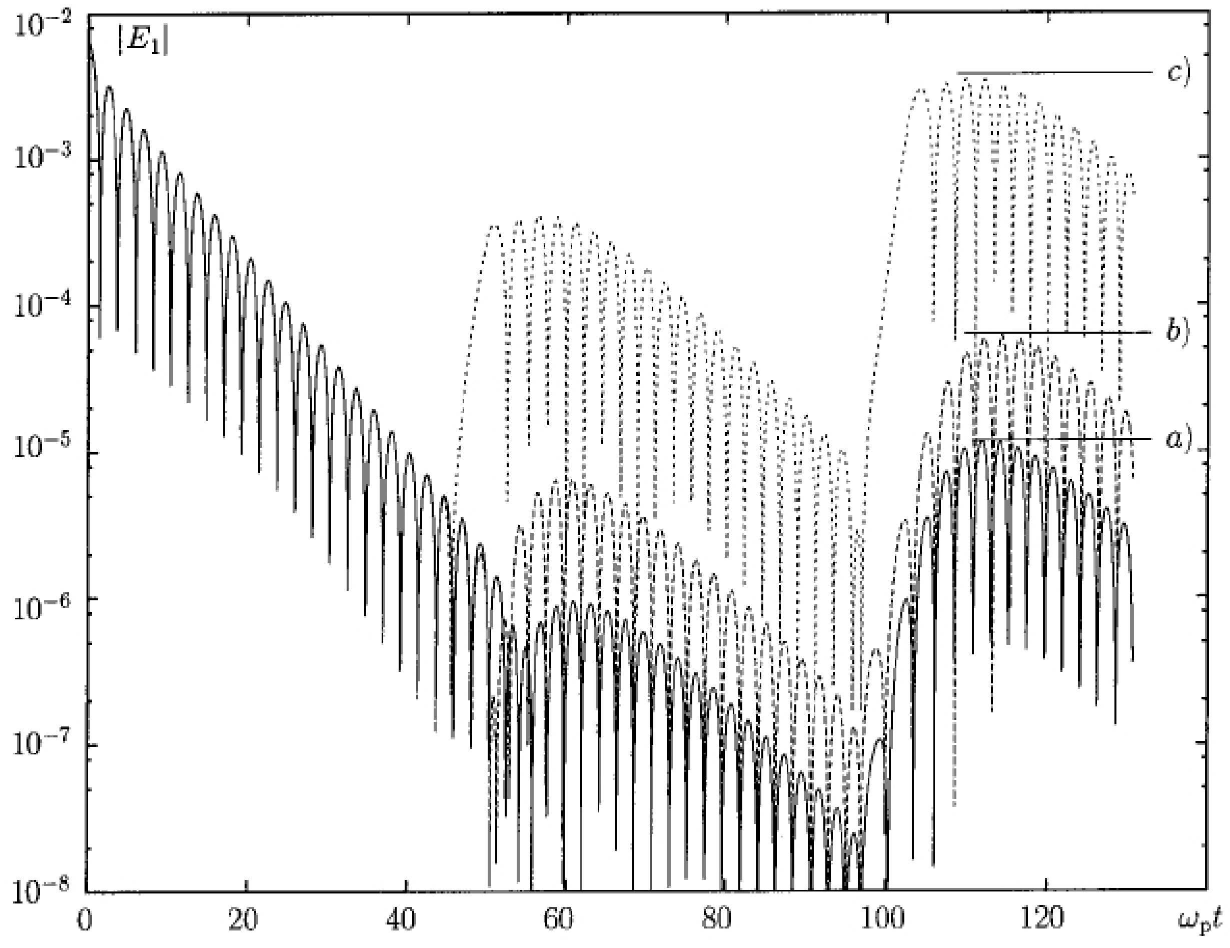
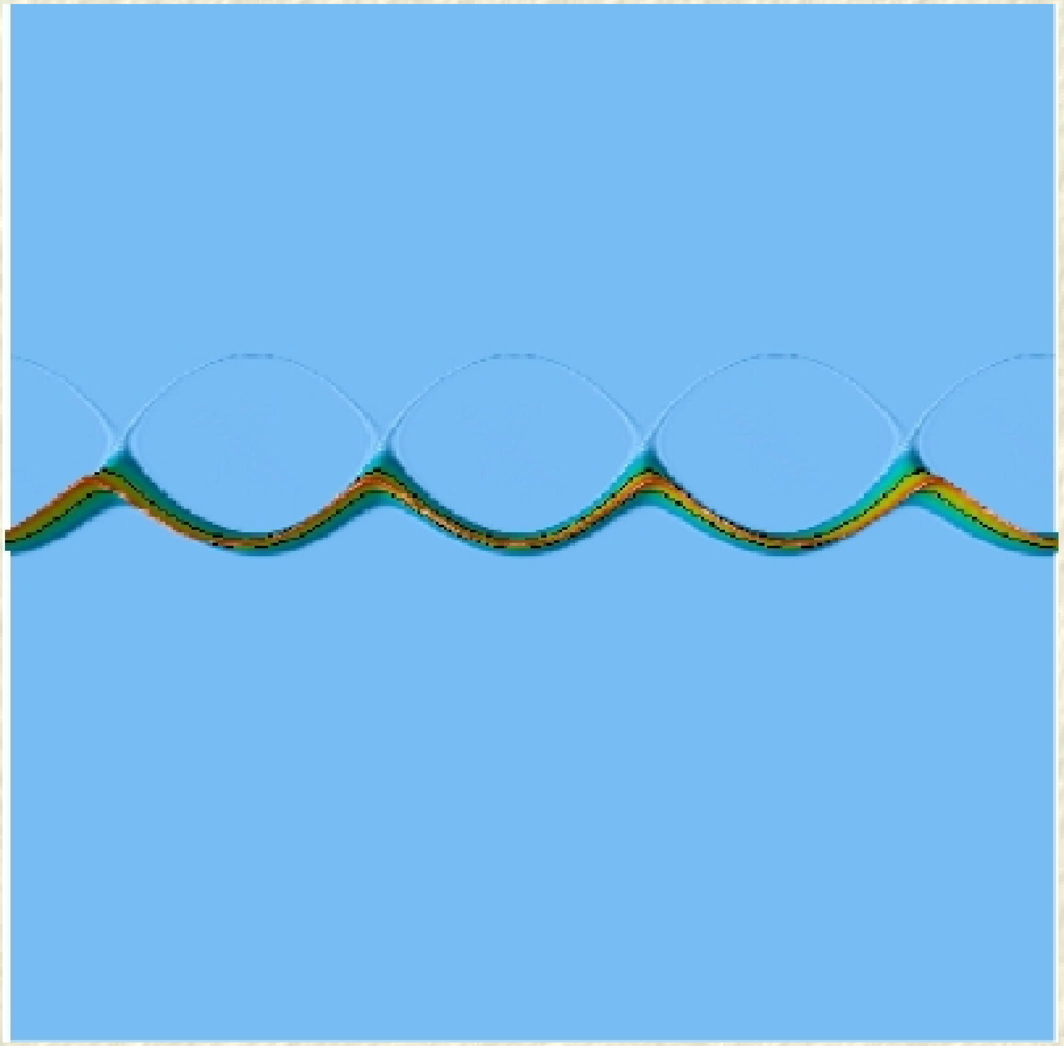
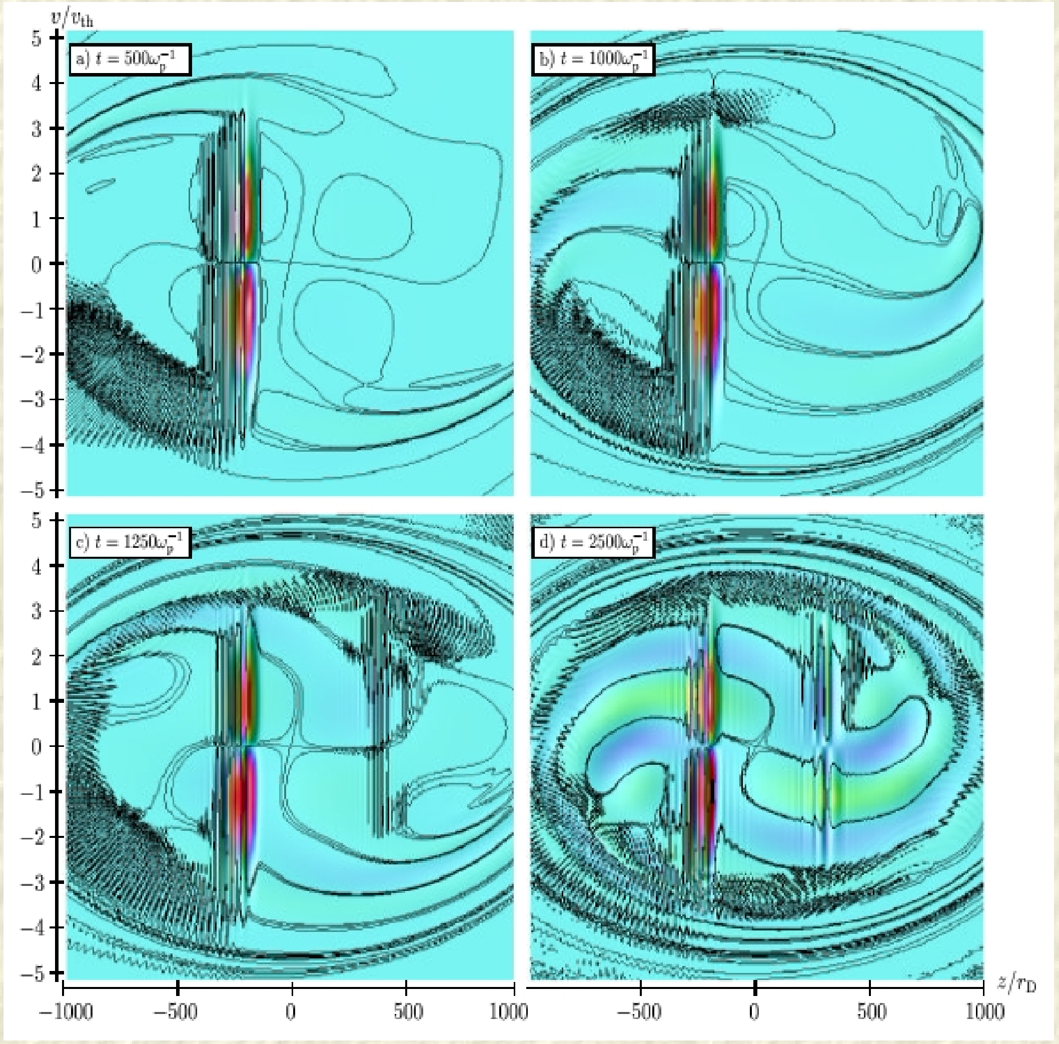


Fig. 1. Reflections of waves against the boundary $\eta = \eta_{\max}$.

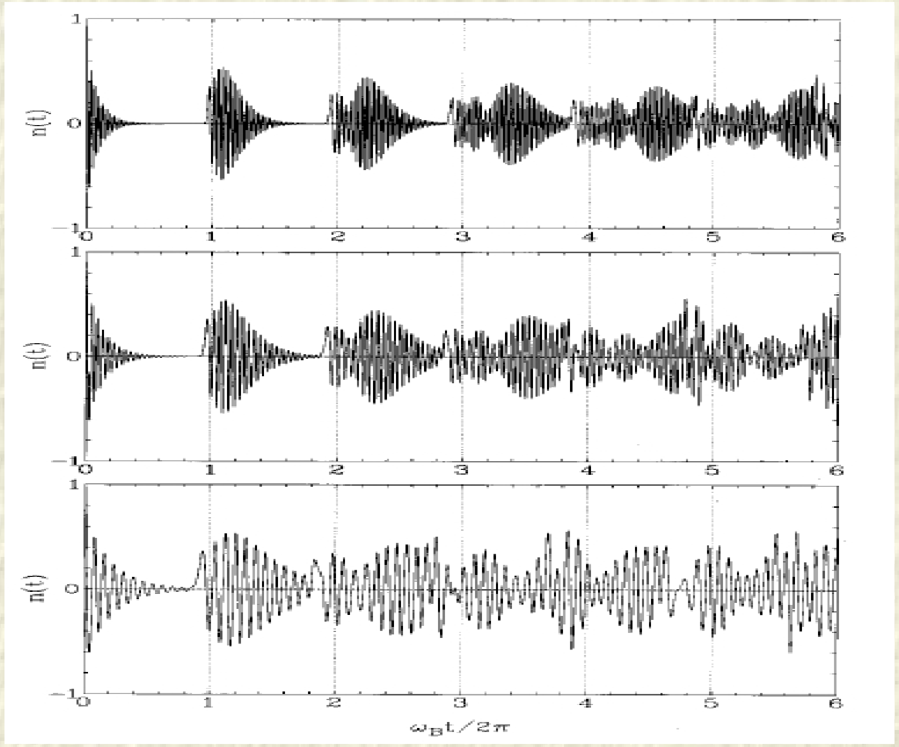
Buneman instability

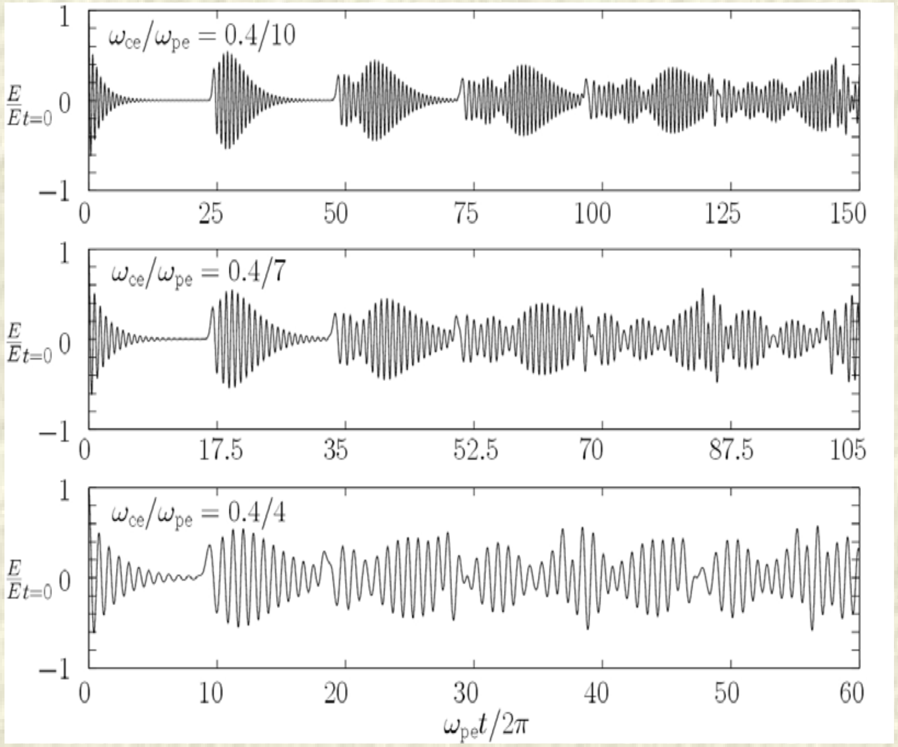


Kinetic tunneling effects



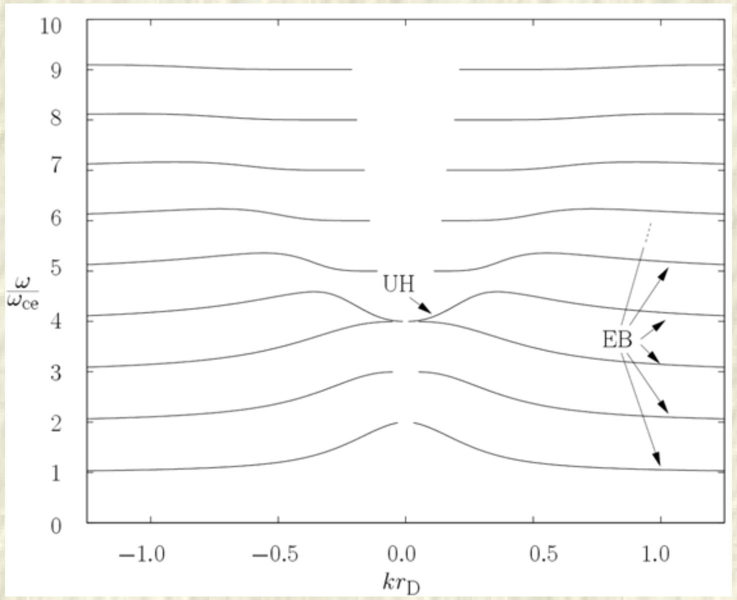
Magnetized plasmas, undamped UH waves

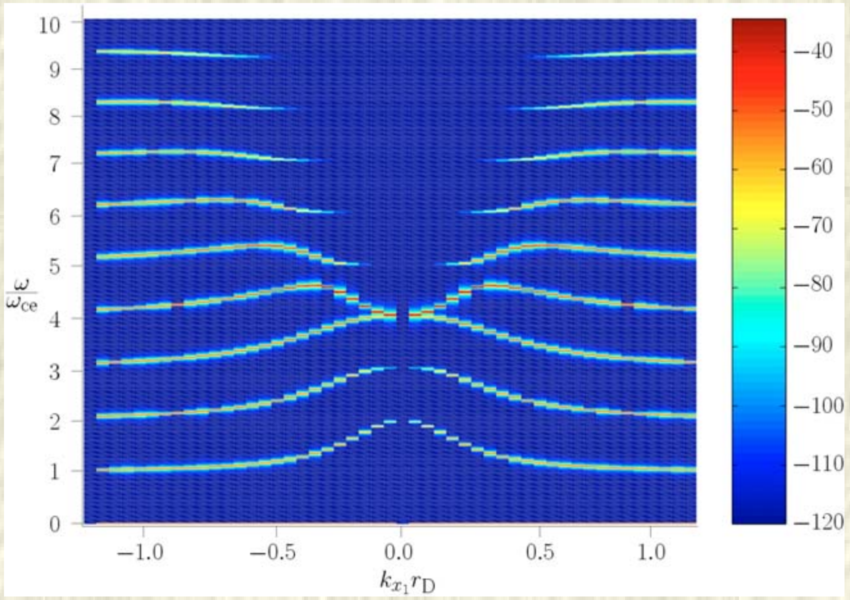




Left Fig. from Stubbe & Sukhorukov, PoP 4, 2497 (1997). Right Fig. from B. Eliasson (http://www.it.uu.se/research/reports/2002-028/)

Electron Bernstein and upper-hybrid waves



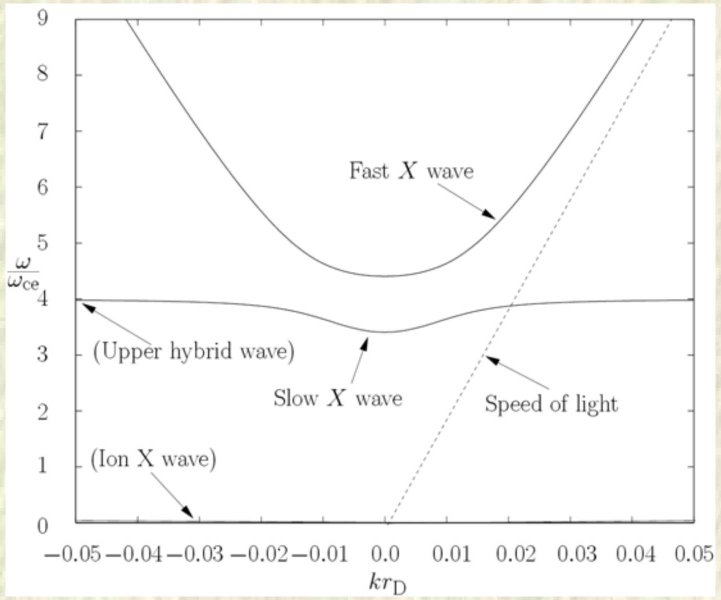


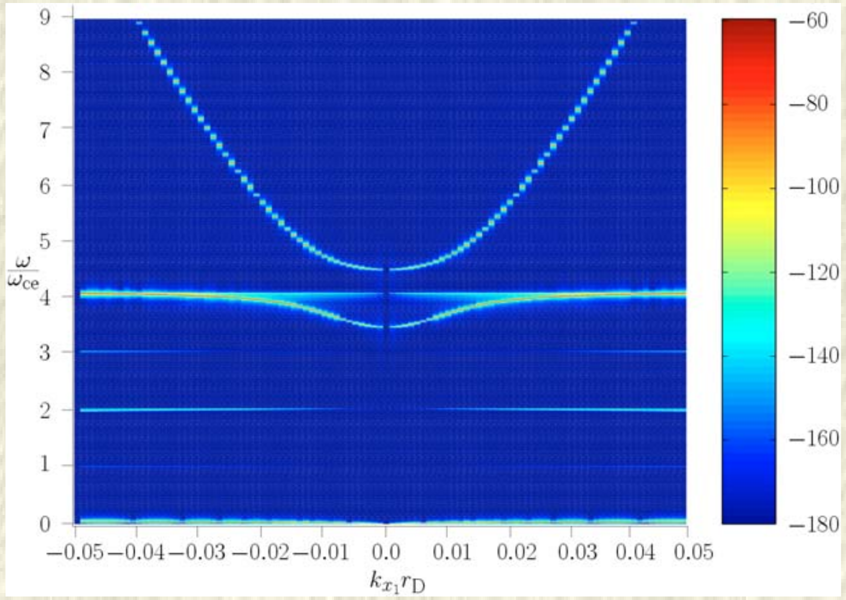
$$1 + \left(\frac{\omega_{pe}}{\omega_{ce}}\right)^2 \exp(-k^2 r_L^2) \int_{\psi=0}^{\pi} \frac{\sin(\psi\omega/\omega_{ce})\sin(\psi)\exp[-k^2 r_L^2 \cos(\psi)]}{\sin(\pi\omega/\omega_{ce})} \mathrm{d}\psi = 0$$

Left panel: F. W. Crawford & J. A. Tataronis, J. Appl. Phys. 36, 2930 (1965).

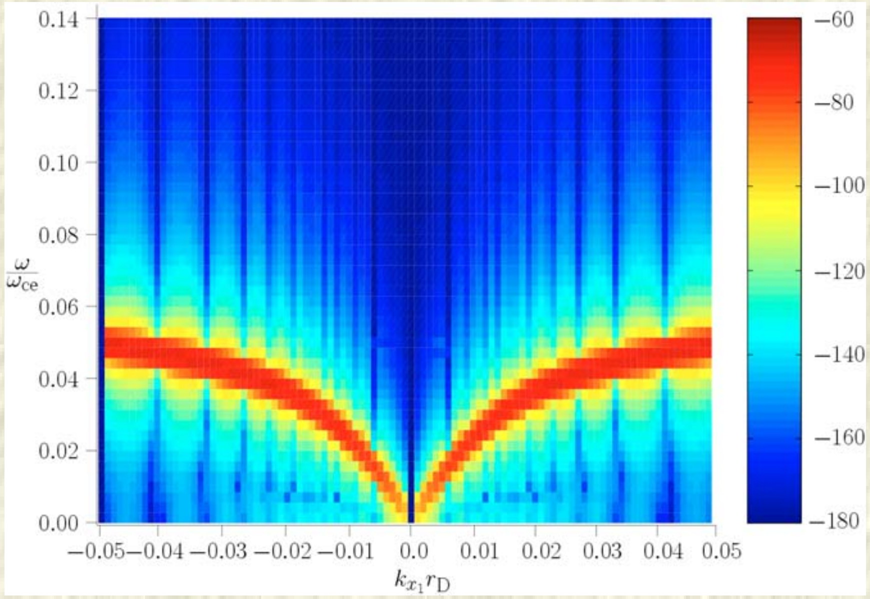
Right panel: B. Eliasson, (http://www.it.uu.se/research/reports/2002-028/

Electromagnetic waves

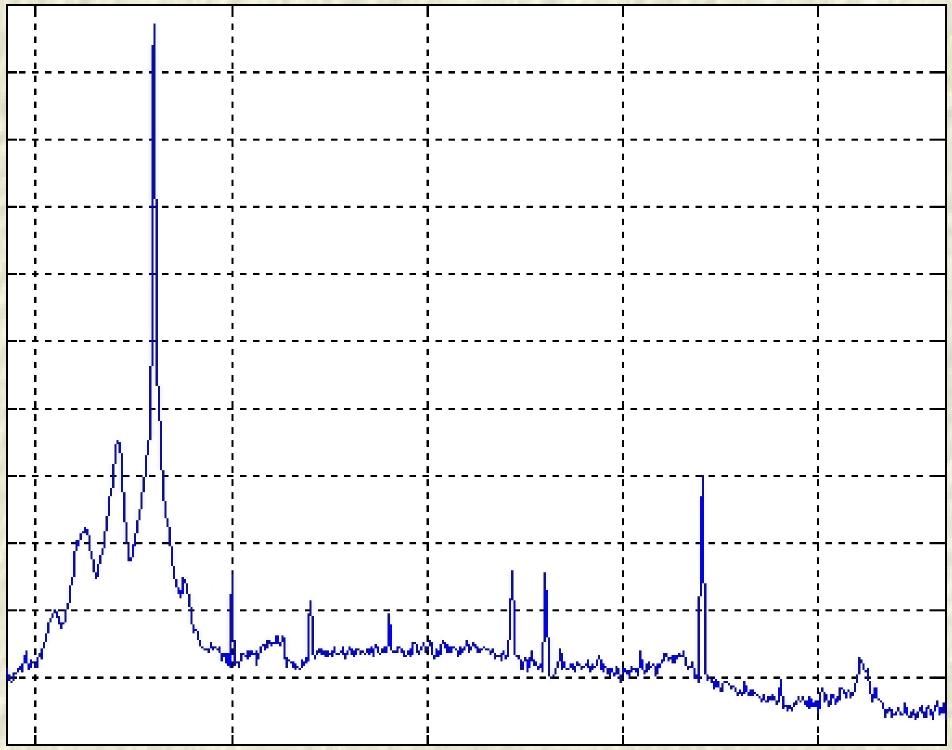




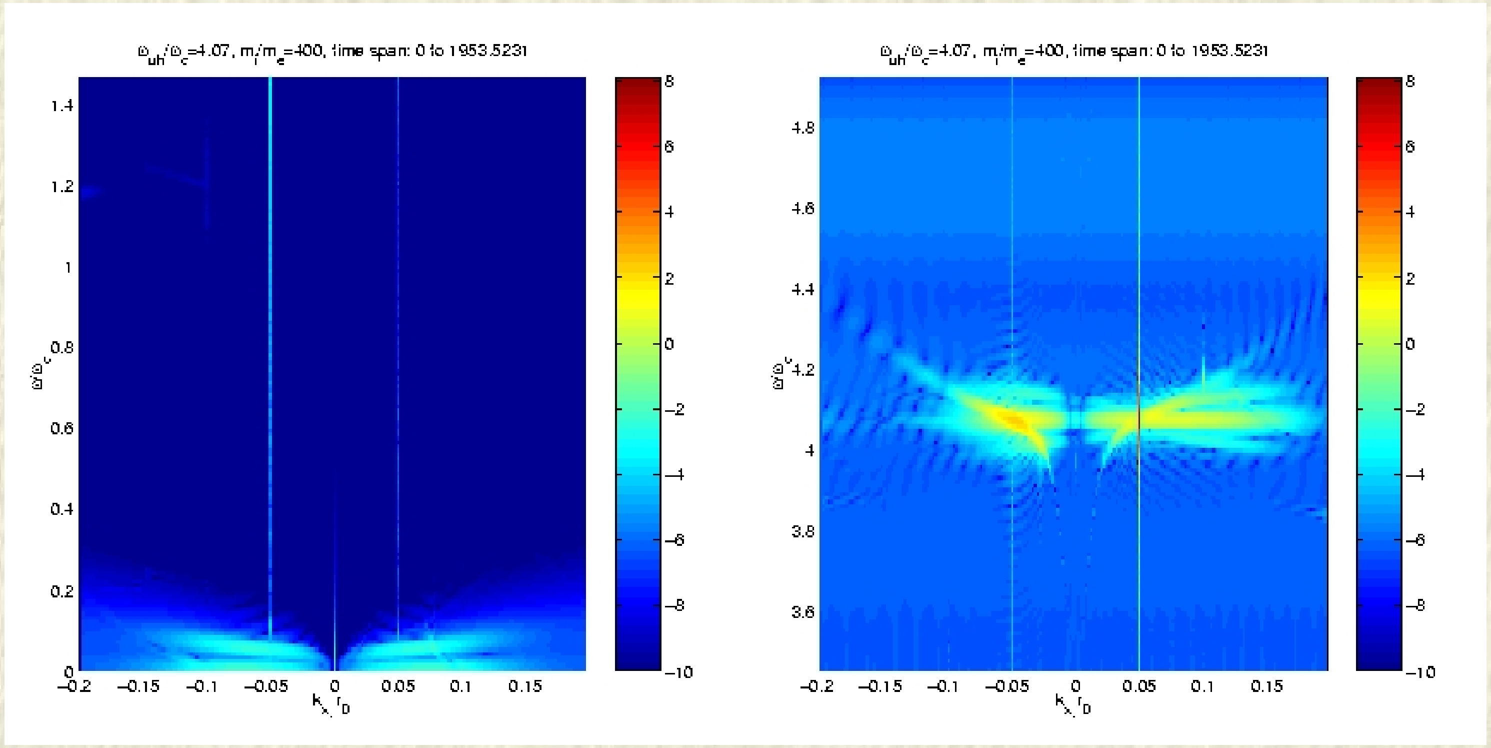
Magnetosonic and lower hybrid waves



Experiment



Nonlinear wave coupling



Summary

A. Numerics: Well-posedness, consistency, stability, convergence. Some methods.

B. Vlasov simulations – problems with filamentation

C. Fourier method in velocity space. 1D and 2D. Outflow boundary conditions

D. Examples of numerical simulations, unmagnetized and magnetized plasmas