Numerical resolution of partial differential equations (PDE)

Sébastien Charnoz & Andrea Ciardi

University of Paris

Références :

Numerical Recipes in Fortran, Press. Et al. Cambridge University press

http://homepage.univie.ac.at/Franz.Vesely/cp0102/dx/dx.html

Many problems in physics are described by evolutionary equations which involve several parameters (t and x most often)

$$c^2 \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial t^2} = f(x,t)$$ Wave equation

$$D \frac{\partial^2 u}{\partial x^2} - \frac{\partial u}{\partial t} = f(x,t)$$ Diffusion equation (heat)

$$\frac{\hbar^2}{2m} \frac{\partial^2 u}{\partial x^2} + i\hbar \frac{\partial u}{\partial t} - U(x)\, u = 0$$ Shrodinger' equation

Etc…

Because they involve several parameters, the differential equation involves partial derivatives w.r.t. the parameters

$$c^2 \frac{\partial^2 u}{\partial x^2} - \frac{\partial^2 u}{\partial t^2} = f(x,t)$$ Wave equation

$$D \frac{\partial^2 u}{\partial x^2} - \frac{\partial u}{\partial t} = f(x,t)$$ Heat equation

$$\frac{\hbar^2}{2m} \frac{\partial^2 u}{\partial x^2} + i\hbar \frac{\partial u}{\partial t} - U(x)\, u = 0$$ Shrodinger esquation

« PDE » = « Partial Differential Equation »

Means : 2, or more, parameters at the denominator of the derivative

Their resolution is **more complex** than single-parameter equations (ODE), and involve several grids because it will be necessary to discratize all parameters

For example: A 3D grid for space and a 1D grid for time

Resolution techniques change a little depending on the size of the problem

We will deal here most often with 1Dx1D systems:
1 dimension of space and 1 dimension of time.

But the methods are mostly the same in N dimensions.

Like with ODEs, we will encounter problems with:
 * accuracy
 * speed
 * stability

We will look for in priority :
Stability
Then : accuracy
Then : speed

There are two main methods of resolution

Either we work in the "physical" space (real space) and the methods will be either :

- Finite difference (the basics)
- Finite volumes (mass conservative method)
- Finite elements (mass conservative method)

Either we work in the Fourrier space (decomposition of functions on a eigen base) and the methods will be called « spectral ». We're working on a finite base (so incomplete) solutions on which solutions will decompose. So we'll have a grid that will contain the terms of the decomposition in the eigen space.

We will study here the method of « finite differences », the most basics, but The most widespread

To begin with: :

The advection equation …

simple but displays a lot of important concepts

The advection equation describes how a quantity is transported in a current (e.g. a poluant in water). It is an HYPERBOLIC equation.
U (x,y,z,t) - concentration of an element at the position (x,y,z) at time t

Total derivative

Pollution production at the moment t at the r position

$$\frac{Du}{Dt} = f(r,t) \Leftrightarrow$$

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial u}{\partial y}\frac{\partial y}{\partial t} + \frac{\partial u}{\partial z}\frac{\partial z}{\partial t} = f(r,t) \Leftrightarrow$$

$$\frac{\partial u}{\partial t} + \vec{v}\cdot\vec{\nabla}u = f(r,t)$$

"v scalar product with gradient of u"

If we only look at the case at 1D
And if the water speed is constant - C
Then the advection equation becomes:

Source term.
=0 if nothing is added
to the system

$$\partial_t u(x,t) + c\partial_x u(x,t) = f(x,t)$$

partial derivative
w.r.t time

partial derivative w.r.t
space

We will first study some properties of this equation
to be inspired on how to solve it numerically

Studying the solution of advection equation

Hyperbolic equations have : Characteristic curves

A remarkable property of the advection equation (found also in the wave equations) is that the solution propagates linearly at speed C.
This is called a « characteristic curve ».

This property is going to be very useful for us.

$$\partial_t u(x,t) + c\partial_x u(x,t) = f(x,t)$$

Let’s do the following change of variables:

$X = \alpha x + \beta t$ and $T = \gamma x + \mu t$
u(x,t)= U (X,T).

We'll take f(x,t)=0

How are derivatives written as a function of the new variables X et T ?

$$u(x,t) = U(X,T) \Rightarrow$$

$$du = \frac{\partial U}{\partial X} dX + \frac{\partial U}{\partial T} dT \Rightarrow$$

$$\begin{cases} \dfrac{\partial u}{\partial x} = \dfrac{\partial U}{\partial X}\dfrac{\partial X}{\partial x} + \dfrac{\partial U}{\partial T}\dfrac{\partial T}{\partial x} \\ \dfrac{\partial u}{\partial t} = \dfrac{\partial U}{\partial X}\dfrac{\partial X}{\partial t} + \dfrac{\partial U}{\partial T}\dfrac{\partial T}{\partial t} \end{cases} \Rightarrow$$

$$\begin{cases} \dfrac{\partial u}{\partial x} = \alpha \dfrac{\partial U}{\partial X} + \gamma \dfrac{\partial U}{\partial T} \\ \dfrac{\partial u}{\partial t} = \beta \dfrac{\partial U}{\partial X} + \mu \dfrac{\partial U}{\partial T} \end{cases}$$

So how does the advection equation is rewritten?

The equation is as follows :

$$(\beta + c\alpha)\frac{\partial U}{\partial X} + (\mu + c\gamma)\frac{\partial U}{\partial T} = 0$$

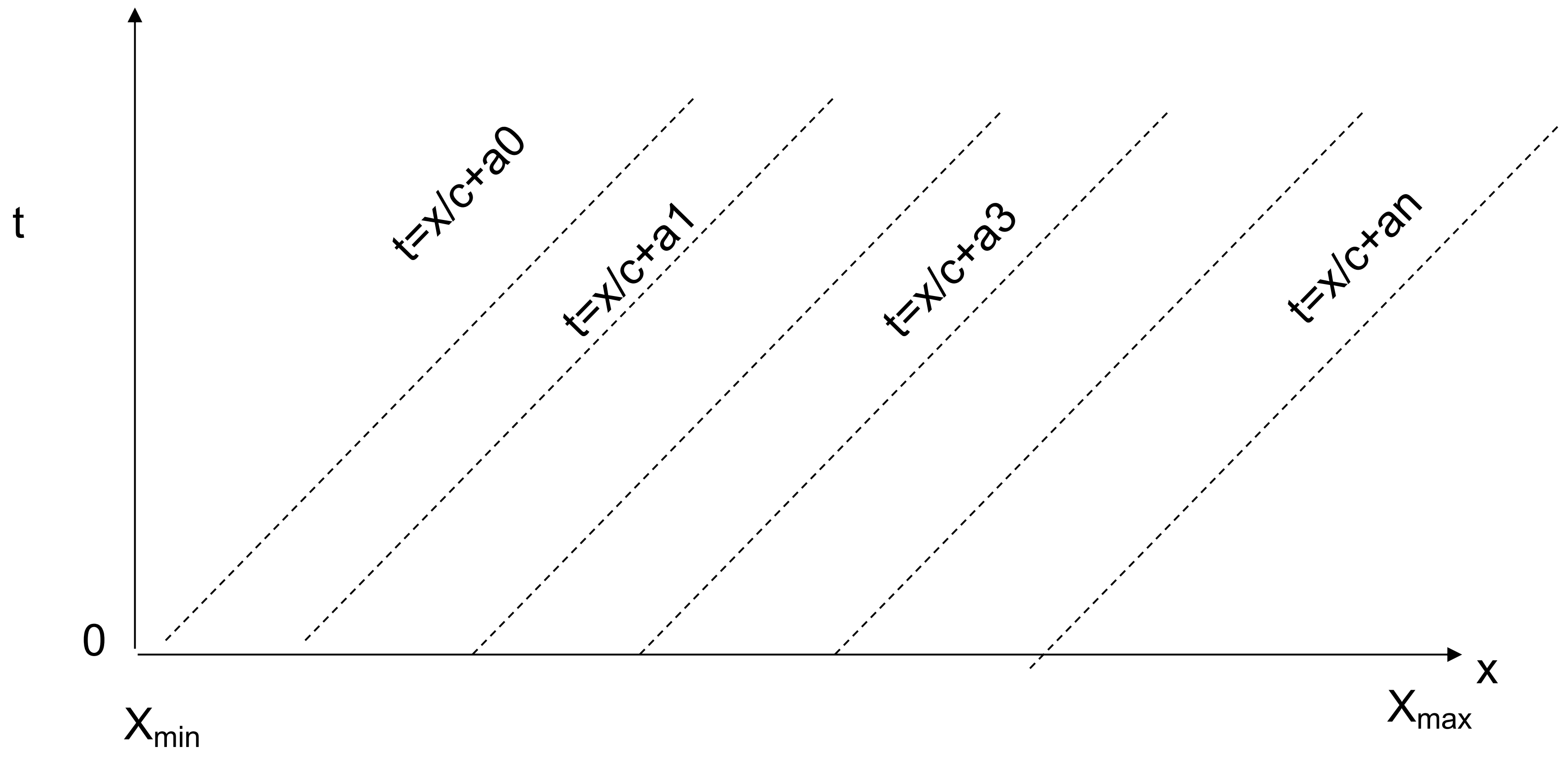
Then , if we choose : $\beta = -c\ \alpha$ (which is always possible) we simply get

$$\frac{\partial U}{\partial T} = 0$$

Then U(X,T) = U(X) only

Therefore u(x,t)=U(X)=u(αx-cαt,t=0)=u(x-ct,t=0)

So u stay constant (=U(x,t=0)) along the lines with equations
x-ct=cste ⇔
t=x/c+ cste



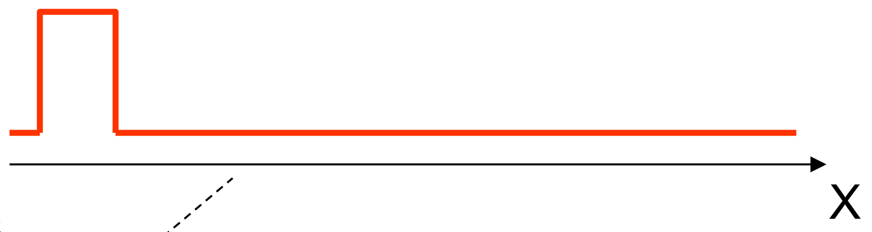
These lines (with slope 1/C) are called « characteristic curves » of the advection equation . T

the solution u (x, 0) propagates along these lines, at speed C

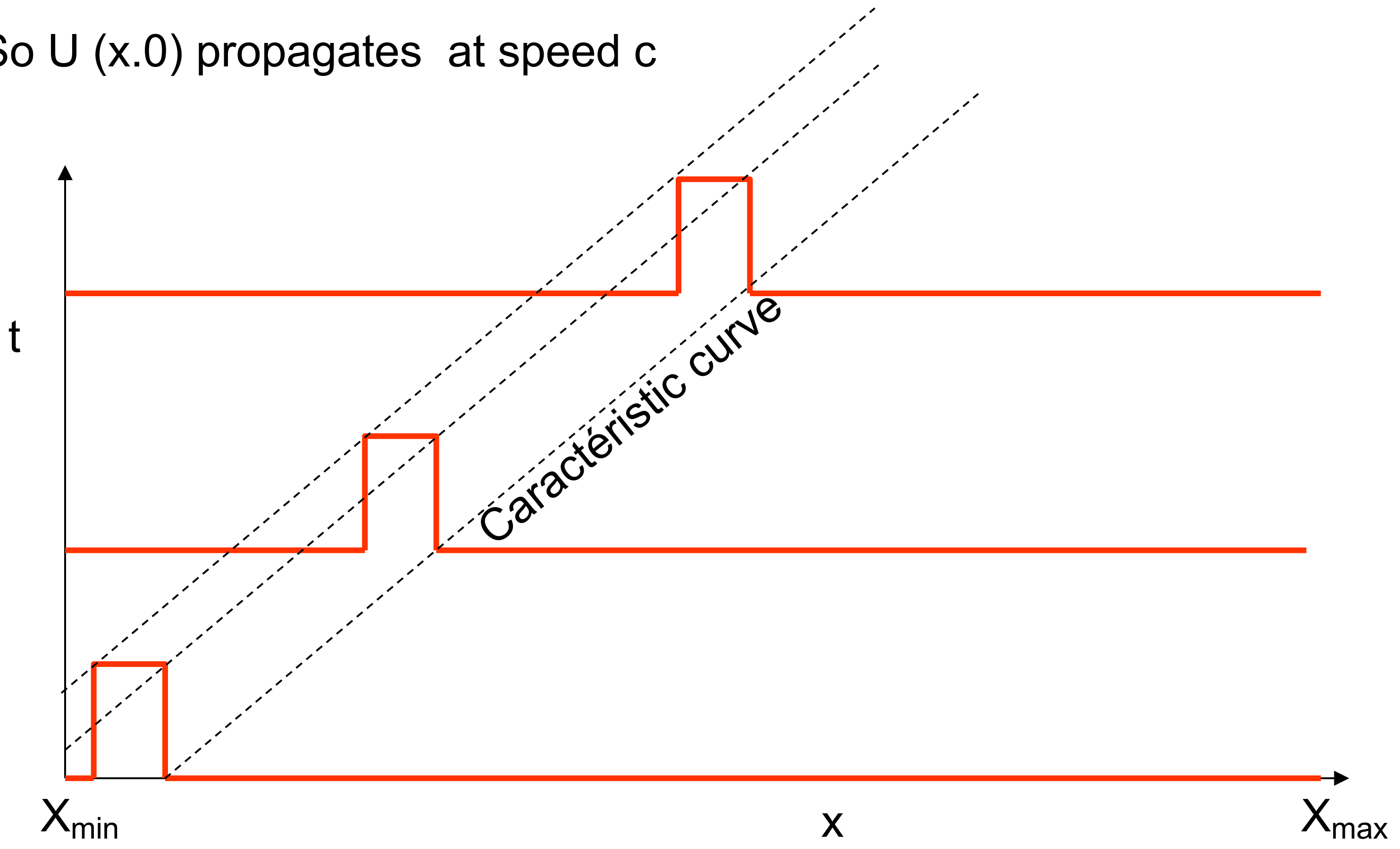
What is u(x,t) ? if x-ct=a then u(x,t)=u(a-ct, t=0)

If a-ct falls outside the definition domain X initial =[X_{min},X_{max}], then we need **a boundary condition** ⇔ si (a-ct) $\notin$ =[Xmin,Xmax] alors u(x,t)=φ(a-ct)

Example:
Suppose U (x.0) is like that :



So U (x.0) propagates at speed c



The same behaviour is found in the sound equation
Note that the initial data U (x.0) **is evacuated after a time (Xmax-Xmin)/c**
Which is the crossing time of the resolution domai. So we need to choose boundary conditions.

This property of propagation of the initial state along the « characteristic curves » will impose serious constraints on numerical resolution.

It can also be used as a criterion to check the validity of the solver.

How to builf the numerical resolution

of the advection equation ?

i.e.

How to advance the solution in time, with steps dt ?

THE GRID

The 1D advection equation is solved on a spaceXtime grid, so each box (or ‘cell’) has dimensions : dx x dt

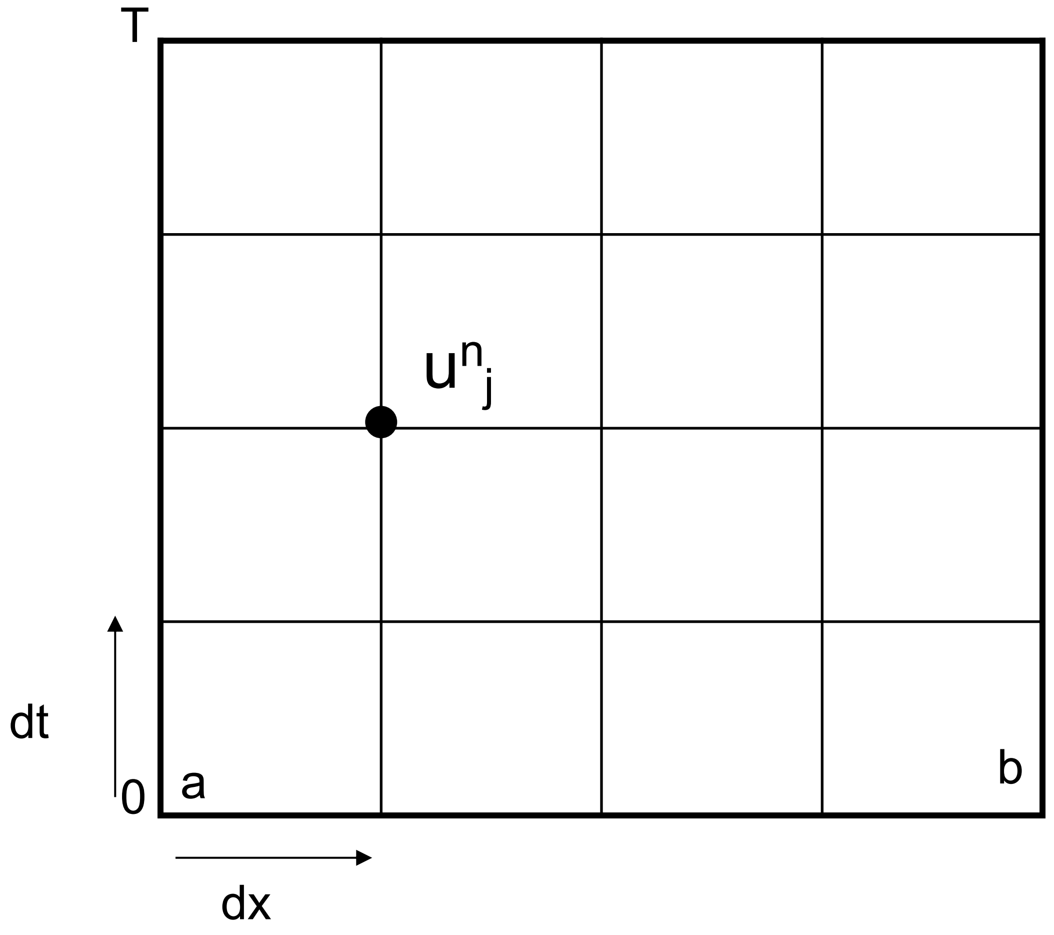
x_j=a+jdx et dx=(b-a)/J

t_n=ndt et dt=T/N

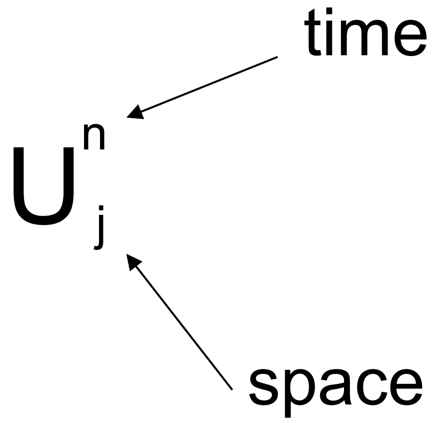
J=nb of cells in X
N= nb of cells in T

Let $U^n{}_j$ the numerical approximation to $u(x_j, t_n)$

The initial state of the system is $U^0{}_j$ for all j = $u(x_j,0)$



Let's agree on the symbols and notation



n goes from 0 to N
j goes from 0 to J

n=0 : initial condition

j=0 ou J+1 : boundary condition

The evolution of the system will be done by gradually calculating values U^{n+1}_j for all J frome the knowledge of U^n_j

Resolution using finite differences

The simplest method is that of finite differences:
approximation of derivatives by finite differences

The equation to be solved is

$$\partial_t u(x,t) + c\partial_x u(x,t) = 0$$

Let's apply order 1 methods :

How to compute d U^n_j/dt ⇔ d $u(x_j,t_n)$/dt ?

How to compute d U^n_j/dx ⇔ d $u(x_j,t_n)$/dx ?

At first order, we can just useTaylor Expansion:

$$\frac{\partial U_j^n}{\partial t} \approx \frac{U_j^{n+1} - U_j^n}{dt} \longleftarrow$$

Explicit forward diference :
$D^+ (f_k) = f_{k+1} - f_k$

$$et$$

$$\frac{\partial U_j^n}{\partial x} \approx \frac{U_j^n - U_{j-1}^n}{dx} \longleftarrow$$

Explicit backward diference :
$D^- (f_k) = f_k - f_{k-1}$

If C >0, then the choice of D- (backward difference)
is physically motivated by the
fact that the information moves forward
from j to j+1 (in space).
In other words the « solution moves to the right »
(because C>0) so it is better to use backward difference that
Uses the left value to compute the right one

$$\partial_t u(x,t) + c\partial_x u(x,t) = 0$$

$$\frac{\partial U_j^n}{\partial t} \approx \frac{U_j^{n+1} - U_j^n}{dt} \qquad \frac{\partial U_j^n}{\partial x} \approx \frac{U_j^n - U_{j-1}^n}{dx}$$

$$\frac{U_j^{n+1} - U_j^n}{dt} + c\frac{U_j^n - U_{j-1}^n}{dx} = 0 \Rightarrow$$

$$\boxed{U_j^{n+1} = U_j^n - \left(\frac{cdt}{dx}\right)\left(U_j^n - U_{j-1}^n\right)}$$

Here is the 1st order in space and time , explicit, integration scheme

"UPWIND SCHEME"

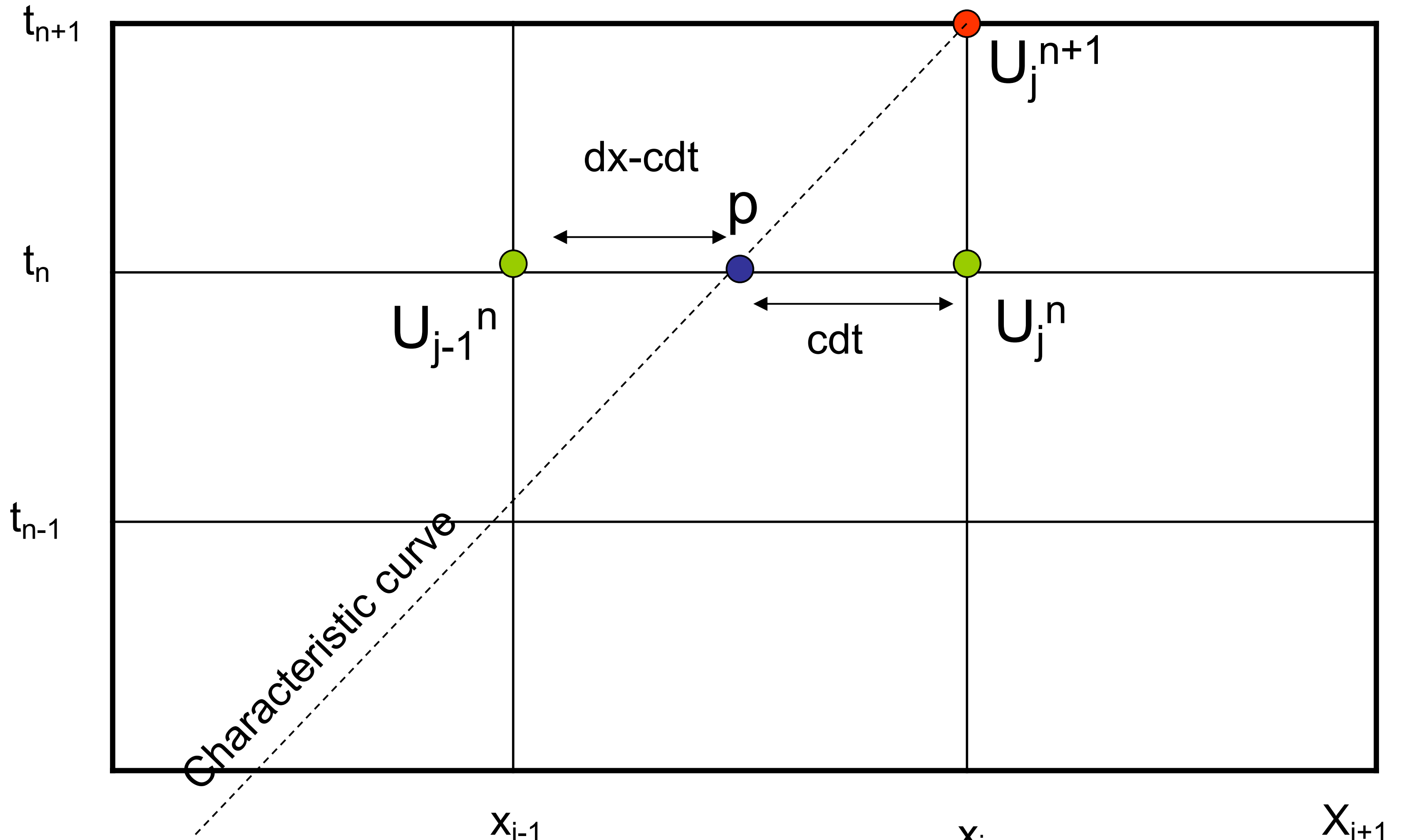
$$U_j^{n+1} = U_j^n - \left(\frac{cdt}{dx}\right)\left(U_j^n - U_{j-1}^n\right)$$

Note that at the edges of the system U^n_{J+1} et U^n_0 are given by the boundary conditions of the problem

We note: $$\sigma = \frac{cdt}{dx}$$

What is the condition of validity of the scheme on σ ??
In what conditions for σ will we obtain physical solutions ?

We note that the system at $\mathbf{U_j^{n+1}}$ comes from the propagation of U_p^n following the characteristic curve. Note that x_p is between x_{j-1} and x_j



We shoud have $\mathbf{U_j^{n+1}= U_p^n}$ **with** $\mathbf{X_p}$ **between** $\mathbf{x_{j-1}}$ **et** $\mathbf{x_j}$

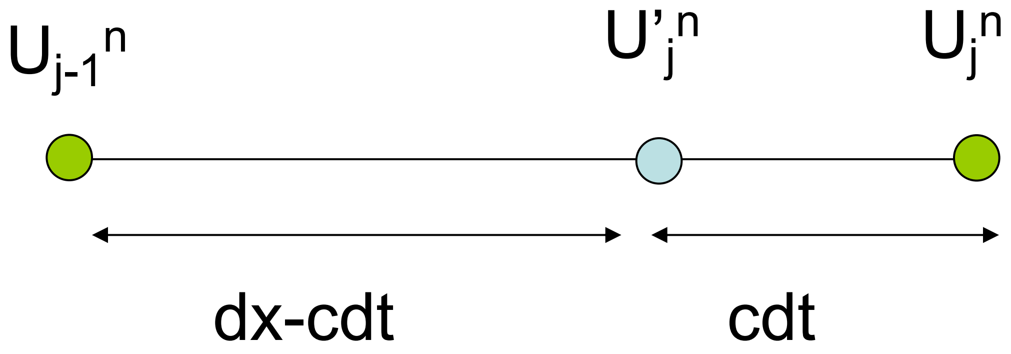
The integration scheme proposes : $U^{n+1}{}_j=U_j^n-cdt/dx\,(U^n{}_j-U_{j-1}{}^n)$

$\Rightarrow$ We did well to take D- for the spatial derivative because the information, comes well from the left!!!

But what happens when the information comes from between two cells ?

Since $U_j^{n+1}=U_j^n-cdt/dx\,(U_j^n-U_{j-1}^n)=(1-\sigma)\,U_j^n+\sigma U_{j-1}^n$

⇨ U_j^{n+} is just a linear interpolation of U_p^n using U_j^n and U_{j-1}^n



For interpolation to be accurate x_p must be very close to x_j or x_{j+1} . In the middle of the segment the interpolation Is the worst. So $cdt << dx \Rightarrow \sigma << 1$:

« COURANT » CONDITION (or CFL)

(CFL : Courant-Friedrichs-Levy)

The CFL condition is.. very common
when you integrate PDEs. It is always found in one form or another
It can be understood as follows:
If C is the speed of transmission of the info.
If dx is the space step
If dt is the time

The dt time step must be much shorter than the time to transmit the information on a length dx:
$dt \ll dx/c \Leftrightarrow cdt/dx \ll 1$ *CONDITION CFL*

Dr. Richard COURANT

German-American mathematician

Göttigen Univ.
NYU (New-York)

Worked with David HILBERT

Died 1972

Numerical implementation: Matrix notation

$U^{n+1}_{0<j<J} = U^{n+1}$

$$\begin{pmatrix} U_0^{n+1} \\ U_1^{n+1} \\ \dots \\ U_J^{n+1} \end{pmatrix}$$

Vector state of the system at time T=(n+1)dt

$U^{n}_{0<j<J} = U^{n}$

$$\begin{pmatrix} U_0^{n} \\ U_1^{n} \\ \dots \\ U_J^{n} \end{pmatrix}$$

Vector state of the system at time T=n dt

We also know that : $U_j^{n+1}=(1-\sigma)\, U_j^n + \sigma U_{j-1}^n$

How to write the solver in the form $U^{n+1}=AU^n$ with A a matrix ?

Simple matrix relationship between U^{n+1} and U^n

CL="boundary condition"

$$\begin{pmatrix} U_0^{n+1} \\ U_1^{n+1} \\ \dots \\ \dots \\ U_J^{n+1} \end{pmatrix} = \begin{pmatrix} CL & 0 & 0 & 0 & 0 \\ \sigma & 1-\sigma & 0 & 0 & 0 \\ 0 & \sigma & 1-\sigma & 0 & 0 \\ 0 & 0 & \sigma & \dots & \dots \\ 0 & 0 & 0 & \sigma & 1-\sigma \end{pmatrix} \begin{pmatrix} U_0^{n} \\ U_1^{n} \\ \dots \\ \dots \\ U_J^{n} \end{pmatrix}$$

A is a bi-diagonal matrix

Important
Memory storage

Boundary conditions (CL)?

If the limit condition is then fixed we can imposes U^{n+1}_0=cst & U^{n+1}_J=cst

Boundary Conditions

Boundary condition in time :

N:0 => Initial state

Boundary condition in space for j=0 (left edge) or J=J (right edge):

To be defined.

Periodic : the most simple U(j=J)=U(j=0)

Non periodic : if the domain is bounded, then many things are possible:

- Reflective : U(J=J-1, N+1)=U(J=J,N)
- Absorbing U(J)=0

etc…

For an infinite domain with “open” boundaries, interpolations are possible for outgoing fluxes, but you have to define what happens for an ingoing flux (for example at j=0).

Note : Since information is propagating from left to right (C>0) we may not need to impose a boundary conditions for U(j=J) because we have all information to compute U(j=J) from U(j=J-1).

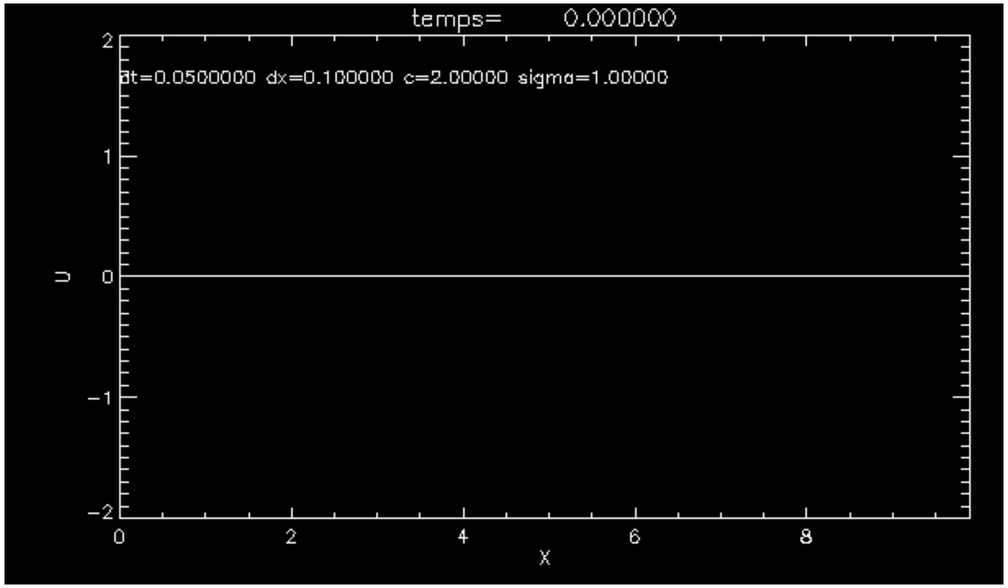
How do we proceed step by step

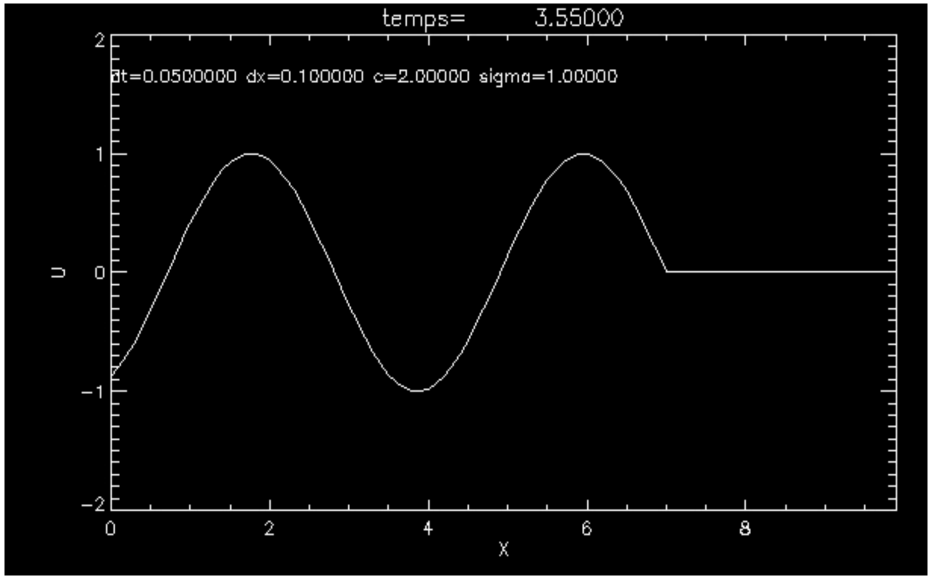
1. Initialize U^0, dx, dt, c
2. Calculate $U^{n+1}=A\ U^n$
3. CL: Put U_0^{n+1}= cste for boundary conditions
4. back to 2

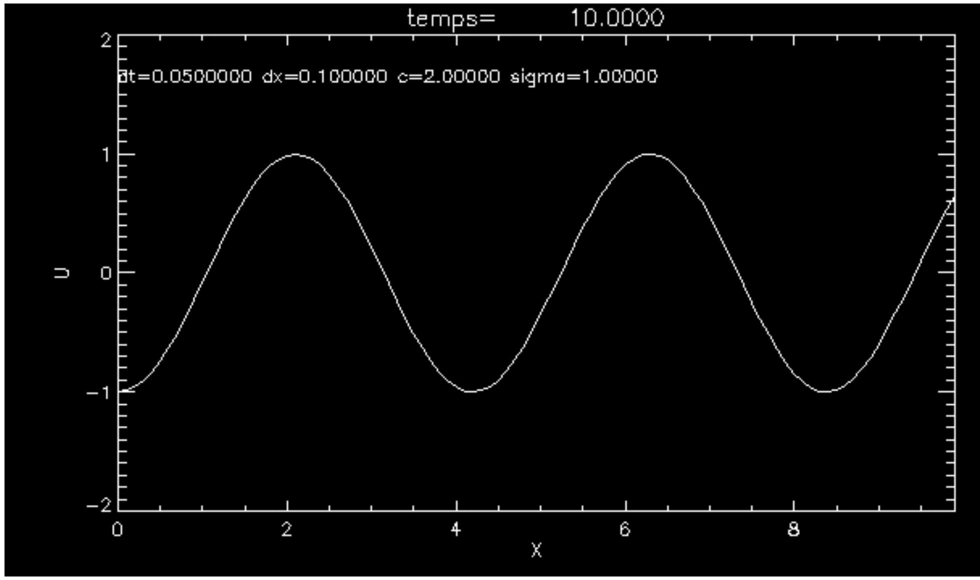
Exemple

Boundary conditons:
U(j=0, t)= sin(t)
We shoudl see a sinus wave
Propagating to the right.

Sigma=1 here
(specific case that works here)





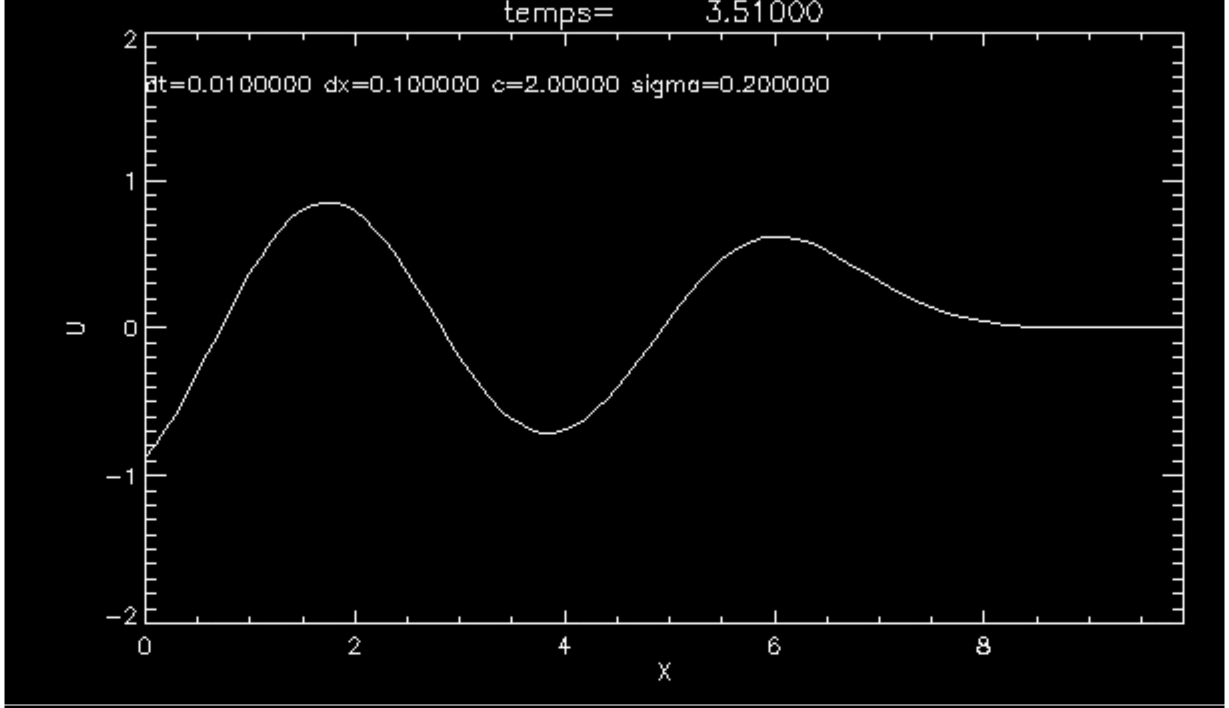


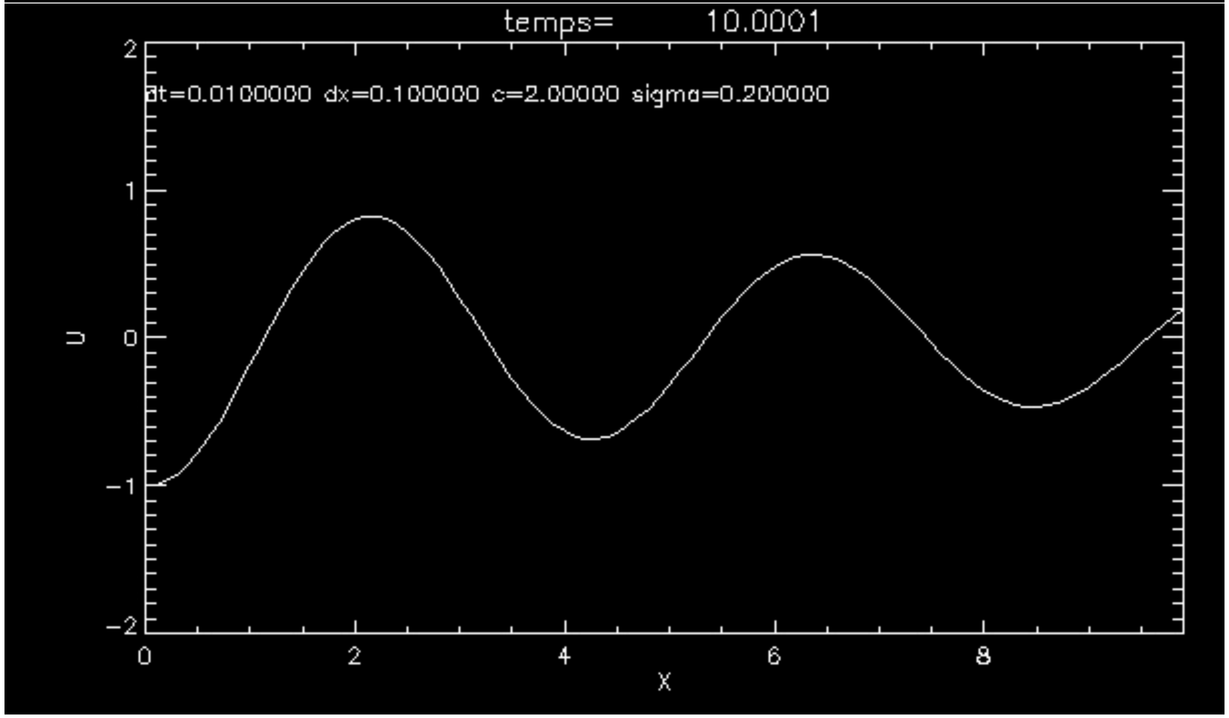
Pathological case sigma=1, exact solution Ujn-1-Uj-1n..., specific to THE UPWIND sigma=1, in practice IMPOSSIBLE ..

A more realistic case :

Let’s take dt=0.01
=> sigma=0.2

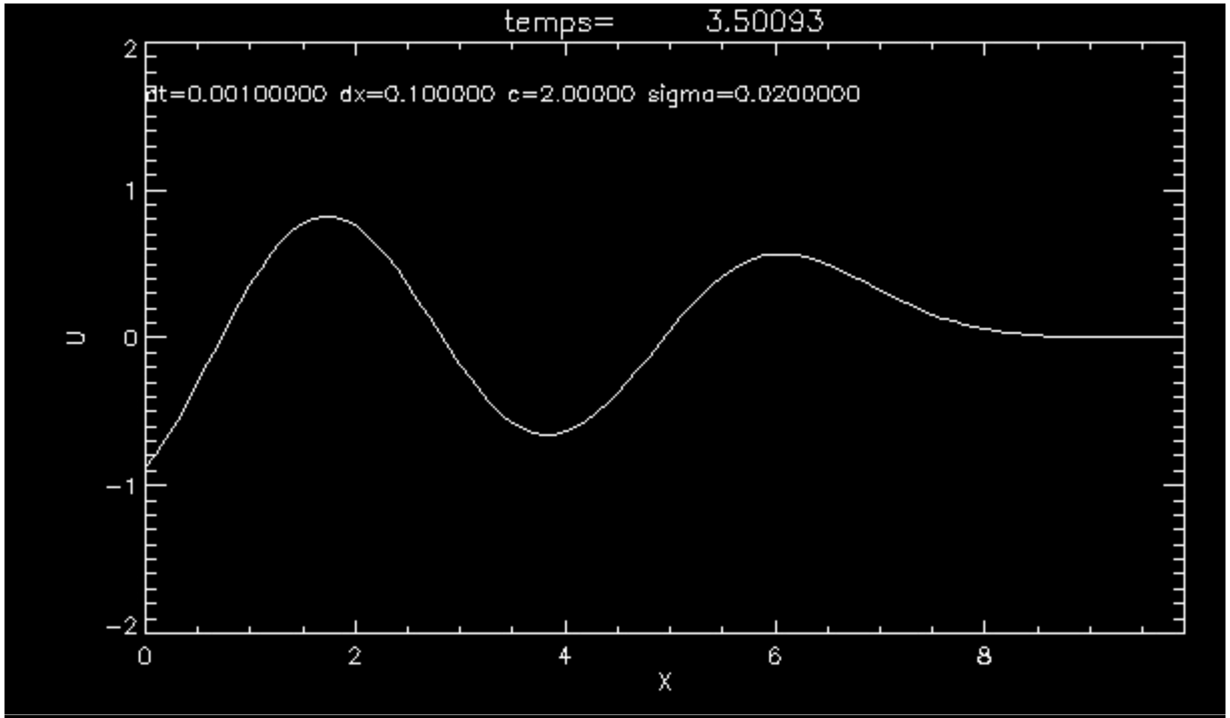
What happens ?

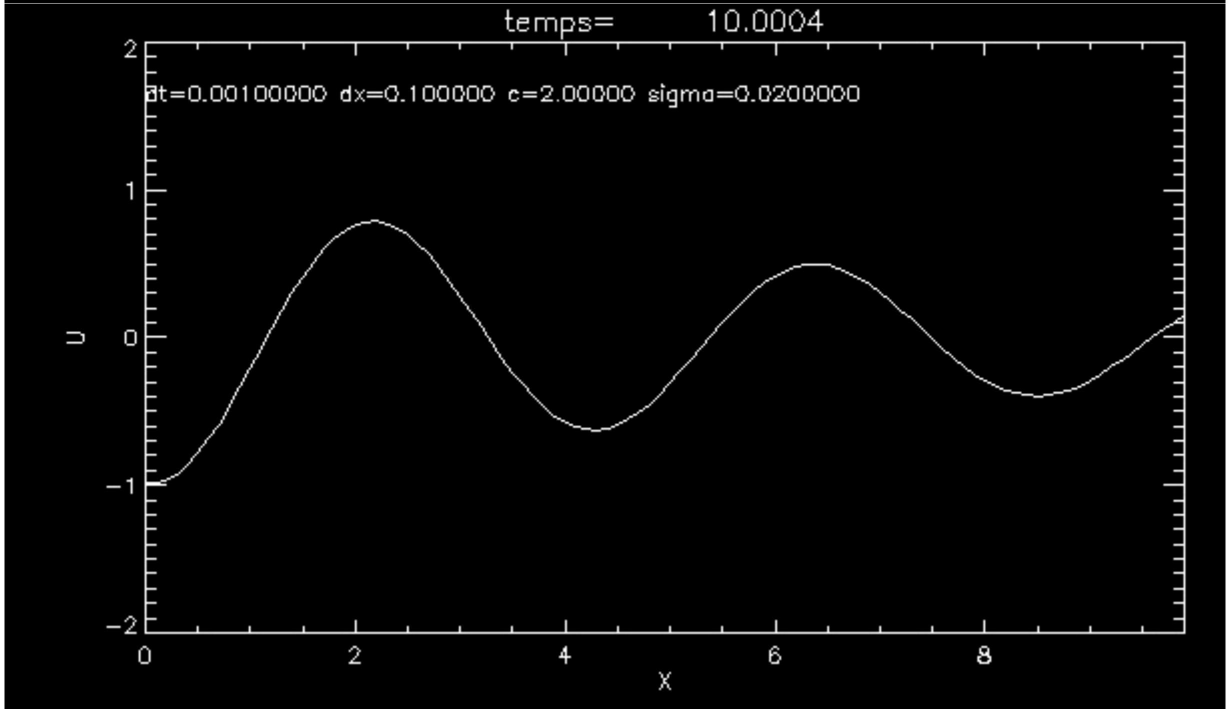




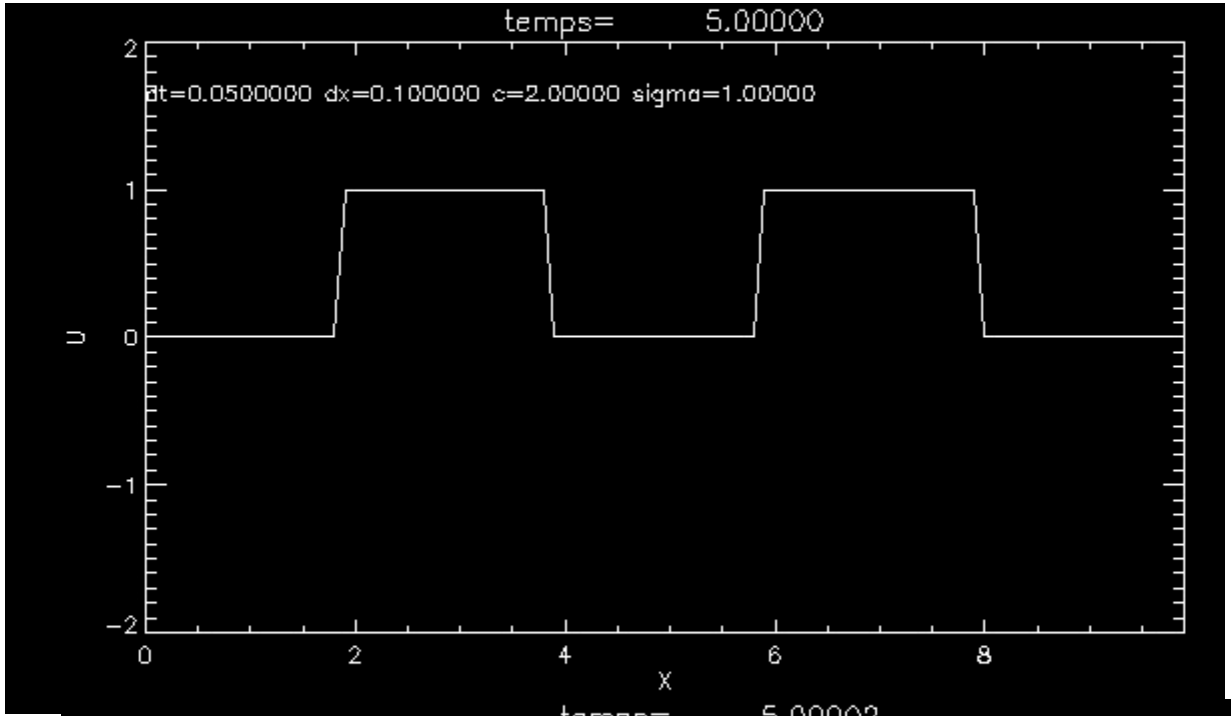
dt=0.001
sigma)=0.02

We observe
« numerical dispersion»
Or « numerical damping »
=>
Artificial diffusion intrinsic
to the numerical scheme

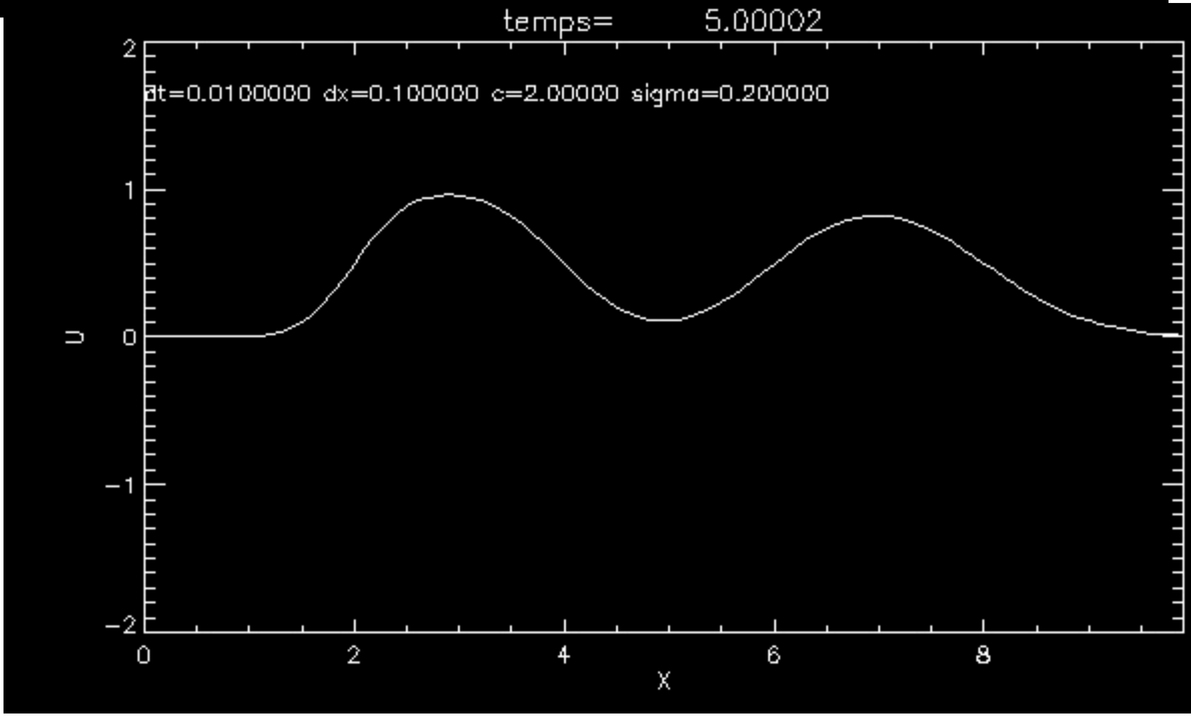




Another examples :
2 Top-hat functions as initial state
U(j=0,t)=0 at all times



Dt=0.05
sigma=1



Dt=0.01
Sigma=0.2

Numerical dissipation…. Where are you from ?

Our derivatives are too aproximatives....
Let's check the 2nd order taylor development

TRUTH

$$U_j = U_{j-1} + dx\frac{\partial U_{j-1}}{\partial x} + \frac{dx^2}{2}\frac{\partial^2 U_{j-1}}{\partial x^2} + O(dx^2)$$

donc

UPWIND SCHEME

$$\frac{\partial U_j^n}{\partial x} \approx \frac{U_j^n - U_{j-1}^n}{dx}$$

$$\begin{tikzcd}
\frac{U_j - U_{j-1}}{dx} = \frac{\partial U_{j-1}}{\partial x} + \frac{dx}{2}\frac{\partial^2 U_{j-1}}{\partial x^2} + O(dx) \arrow[dr, "\text{truth}"] & & \frac{\partial U_j^n}{\partial x} \approx \frac{U_j^n - U_{j-1}^n}{dx} \arrow[dl, "\text{approximation}"'] \\
& \partial_t u(x,t) + c\partial_x u(x,t) = 0 &
\end{tikzcd}$$

By neglecting the 2nd order term in our approximate derivative, we introduce Artificially a 2nd order "parasitic" term in our numerical approximation that behaves like a diffusion process (like viscosity) that "smoothes" the solution artificially.
We will do better later…

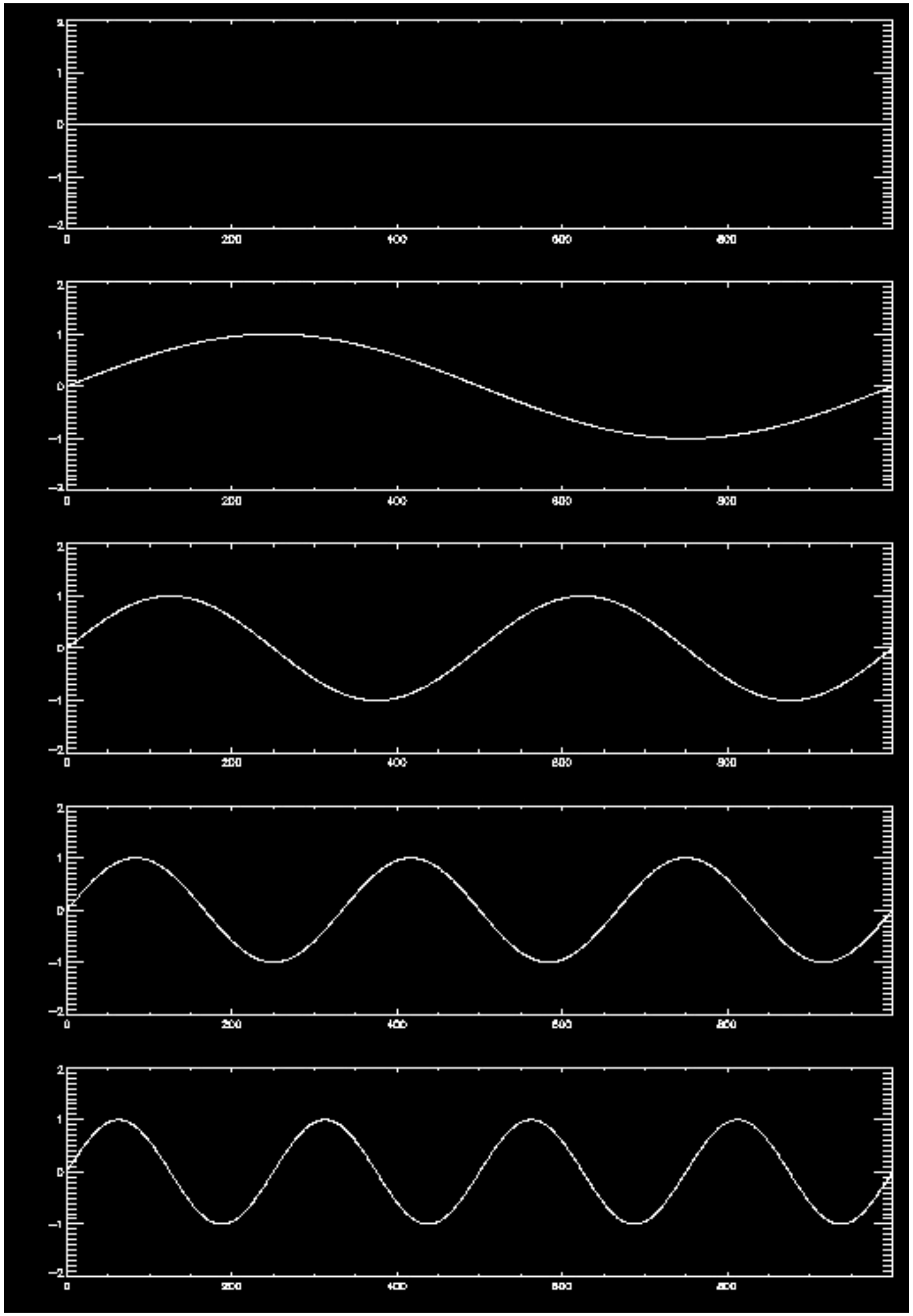
Stability Analysis: Von Neuman Method
Let's check upwind's method stabitily (not easy !)

One must consider the transformation of Fourrier ' Spectral approach'
Approche spectrale

We decompose $U^n(x)$ on a (sinus, cosinus) base
Considering that $U^n_j= U^n(x_j)$ avec $x_j=j\, dx$

$$U_j^n = \sum_k f_k^n e^{ikx_j} = \sum_k f_k^n e^{ik\ jdx}$$

The wavelength of the k mode is $\lambda=L/k$
k = number of times that the mode enters in the box of length L



K=0

K=1

K=2

K=3

K=4 etc…

Decomposition
a solution
on eigen basis of modes

domain : 0 < x < 1000

$$U_j^n = \sum_k f_k^n e^{ik\, jdx}$$

spectral approach:
work on f rather
than on the U

$$U_j^n = \sum_k f_k^n e^{ik\ jdx}$$

We then consider the evolution of f^n_k which are the coefficients of each mode.

Over time: $f^{n+1}_k = g(k)\, f^n_k$, g(k) is the amplification factor

Stability ⇔ || g(k) || < 1 for any k

We're looking at whether the modes are amplified or not

How does the upwind scheme translate into a fourier base?

$$U_j^{n+1} = (1-\sigma)U_j^n + \sigma U_{j-1}^n$$

Integration scheme

$$\Rightarrow$$

We replace U with his FT

$$\sum_k f_k^{n+1} e^{ikj\,dx} = (1-\sigma)\sum_k f_k^n e^{ikj\,dx} + \sigma \sum_k f_k^n e^{ik(j-1)\,dx} =$$

$$\Rightarrow$$

$$\sum_k f_k^{n+1} e^{ikj\,dx} = (1-\sigma)\sum_k f_k^n e^{ikj\,dx} + \sigma \sum_k \left(f_k^n e^{-kidx}\right) e^{ikj\,dx}$$

$$\Rightarrow$$

Find the amplification factor of the k mode : $u_k^{\,n+1} = g(k)\, u_k^n$

$$f_k^{n+1} = (1-\sigma) f_k^n + \sigma \cdot e^{-kidx} f_k^n$$

$$f_k^{n+1} = \underbrace{\left(1-\sigma+\sigma e^{-ikdx}\right)} f_k^n$$

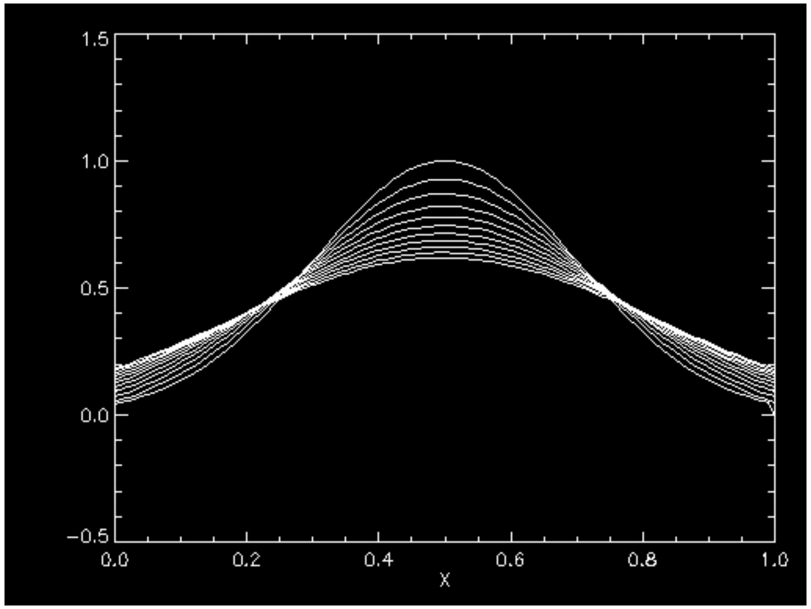
$$g(k) = \left(1-\sigma+\sigma e^{-ikdx}\right) = 1+\sigma\left(e^{-ikdx}-1\right)$$

$$\left\|g(k)\right\|^2 = 1-2\sigma(1-\sigma)\big(1-\cos(kdx)\big)$$

Stability: if ||g(k)|| < 1 for any k
⇔ SIGMA less than or equal to 1

We still find the CFL condition

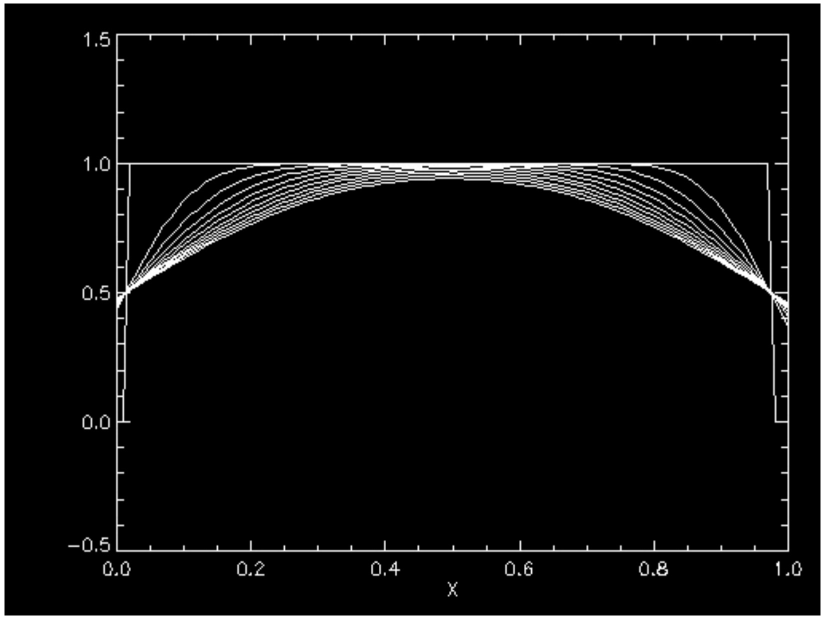
Our pattern is stable but dispersive
We also see that there is a coupling of the modes
=> origin of the dispersion



Sigma=0.2

Evolution of a Gaussian pulse

(note : the gausian is moving to
The right be here I have just
Recentered all of them for easy
Comparison)



Top hat evolution

Conclusion on the UPWIND scheme:

A UPWIND scheme to solve the advection eqaution is easily obtained by writing.
The scheme must respect the natural direction of transport of the information (forward or backward difference in space)

time $$\left\{ \frac{\partial U_j^n}{\partial t} = \frac{U_j^{n+1} - U_j^n}{dt} \right.$$

space $$\left\{ \begin{array}{ll} \dfrac{\partial U_j^n}{\partial x} = \dfrac{U_j^n - U_{j-1}^n}{dx} & \text{si } C > 0 \\ \dfrac{\partial U_j^n}{\partial x} = \dfrac{U_{j+1}^n - U_j^n}{dx} & \text{si } C < 0 \end{array} \right.$$

If C>0

If C< 0

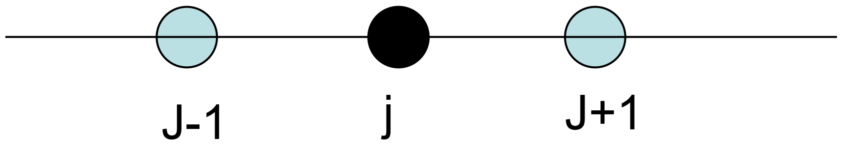
It is stable if the CFL condition is met
But it is dispersive(« smoothing » of solutiions)
It is mainly used to propagate functions with sharp edges
(shock waves)

Exploring other integration schemes:

1) Higher order explicit

2) Implicit schemes

Using Taylor's development to design other integration schemes



$$(1)\ \ U_{j+1}^{n} = U_{j}^{n} + dx\frac{\partial U_{j}^{n}}{\partial x} + \frac{dx^{2}}{2!}\frac{\partial^{2}U_{j}^{n}}{\partial x^{2}} + O(dx^{2})$$

Taylor expansion at X_J+dx

$$(2)\ \ U_{j-1}^{n} = U_{j}^{n} - dx\frac{\partial U_{j}^{n}}{\partial x} + \frac{dx^{2}}{2!}\frac{\partial^{2}U_{j}^{n}}{\partial x^{2}} + O(dx^{2})$$

Taylor expansion at X_j - dx

$\Rightarrow (1)-(2)$ donne

$$\frac{\partial U_{j}^{n}}{\partial x} = \frac{U_{j+1}^{n} - U_{j-1}^{n}}{2dx} + O(dx^{2})$$

2nd order accurate !

2nd order method
« central difference»

The same Reasoning on **time** gives :

$$\frac{\partial U_{j}^{n}}{\partial t} = \frac{U_{j}^{n+1} - U_{j}^{n-1}}{2dt} + O(dt^{2})$$

NOTE :
This new expression is actually the average of Right and Left derivatives.
Indeed the centered drivative is also rewritten ;

$$\frac{\partial U_j^n}{\partial x} = \frac{U_{j+1}^n - U_{j-1}^n}{2dx} = \frac{1}{2}\left[\frac{U_{j+1}^n - U_j^n}{dx} + \frac{U_j^n - U_{j-1}^n}{dx}\right]$$

We can continue the game for higher orders

$$(1)\; U_{j+2}^{n} = U_j^n + 2dx\frac{\partial U_j^n}{\partial x} + \frac{4dx^2}{2!}\frac{\partial^2 U_j^n}{\partial x^2} + \frac{8dx^3}{3!}\frac{\partial^2 U_j^n}{\partial x^3} + O(dx^3)$$

$$(2)\; U_{j+1}^{n} = U_j^n + dx\frac{\partial U_j^n}{\partial x} + \frac{dx^2}{2!}\frac{\partial^2 U_j^n}{\partial x^2} + \frac{dx^3}{3!}\frac{\partial^2 U_j^n}{\partial x^3} + O(dx^3)$$

$$(3)\; U_{j-1}^{n} = U_j^n - dx\frac{\partial U_j^n}{\partial x} + \frac{dx^2}{2!}\frac{\partial^2 U_j^n}{\partial x^2} - \frac{dx^3}{3!}\frac{\partial^3 U_j^n}{\partial x^3} O(dx^3)$$

$$\Rightarrow$$

$$\frac{\partial U_j^n}{\partial x} = \frac{-U_{j+2}^n + 6U_{j+1}^n - 3U_j^n - 2U_{j-1}^n}{2dx} + O(dx^3)$$

3nd order accurate !

This is a 4 points forward method : the « stencil » goes from j-1 to j+2

Etc…

But these methods involve seeking information further "away" in time or space.

Step N-1 must be stored in memory

Let's build new diagrams for the advection equation, order 2 to time and in space

$$\begin{tikzcd}
& \partial_t u(x,t) + c\partial_x u(x,t) = 0 \arrow[dl] \arrow[dr] & \\
\frac{\partial U_j^n}{\partial t} \approx \frac{U_j^{n+1} - U_j^{n-1}}{2dt} & & \frac{\partial U_j^n}{\partial x} = \frac{U_{j+1}^n - U_{j-1}^n}{2dx}
\end{tikzcd}$$

NEW INTEGRATION SCHEME?

Full method 2nd order space and time method :

$$\partial_t u(x,t) + c\partial_x u(x,t) = 0$$

$$\frac{1}{2dt}\left(U_j^{n+1} - U_j^{n-1}\right) + c\,\frac{U_{j+1}^n - U_{j-1}^n}{2dx} = 0 \Rightarrow$$

$$U_j^{n+1} = U_j^{n-1} - \sigma\left(U_{j+1}^n - U_{j-1}^n\right)$$

New Integration Algorithm

Notice the simplicity..
But requires more memory

We « jump » over position j,and n

This method is called "leap-frog" or " sheep jump " in French (.... and not frog jump...)

Implementation of the Leap-Frog

$$U^{n+1} = AU^{n} + U^{n-1}$$

$$A = \begin{pmatrix} 0 & -\sigma & 0 & 0 & 0 \\ +\sigma & 0 & -\sigma & 0 & 0 \\ 0 & +\sigma & 0 & -\sigma & 0 \\ 0 & 0 & +\sigma & \dots & \dots \\ 0 & 0 & 0 & +\sigma & 0 \end{pmatrix}$$

Problème : How to Calculate the 1st time step (N=1) because we don't know
The system at n=-1.

Solution: Take the first step with a UPWIND, and then continue with LEAPFROG

1st step with upwind

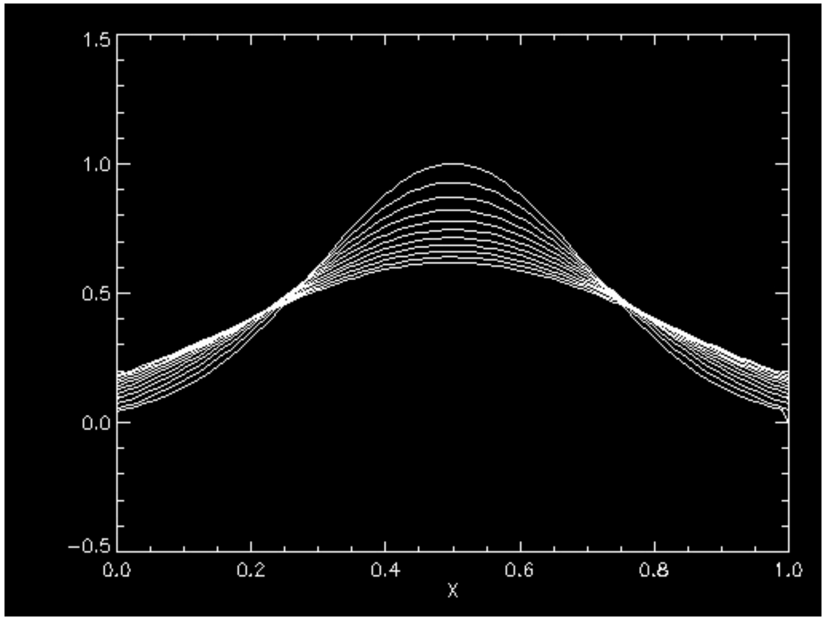
Next step with leap-frog

1. Initialize the systemdt,dx,U0
2. compute U1= A_{upwind}U0
3. compute U2=$A_{leapfrog}$U1+U0
4. U0=U1 et U1=U2
5. Go back to 3

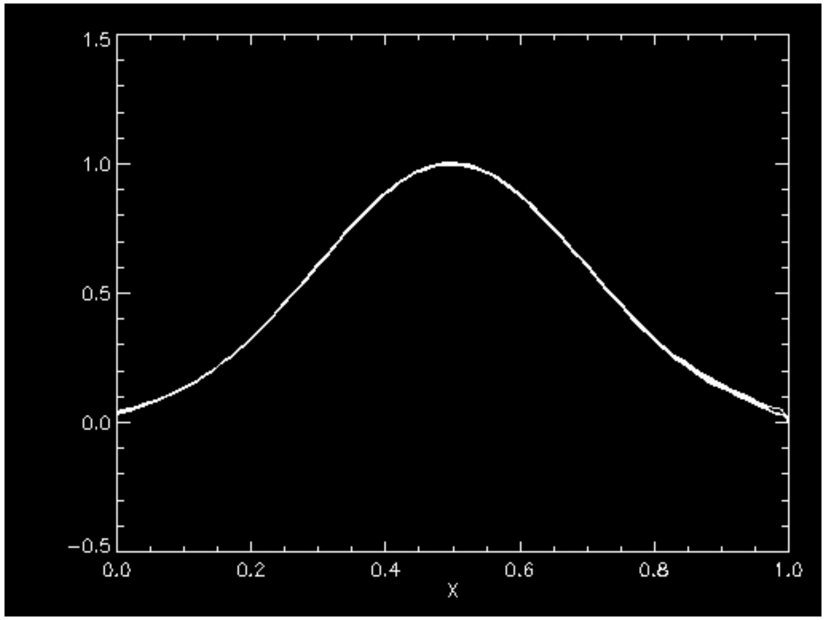
Keep in memory U^{n-1}

Comparison upwind Vs leap frog, sigma=0.2

Gaussian pulse



Upwind



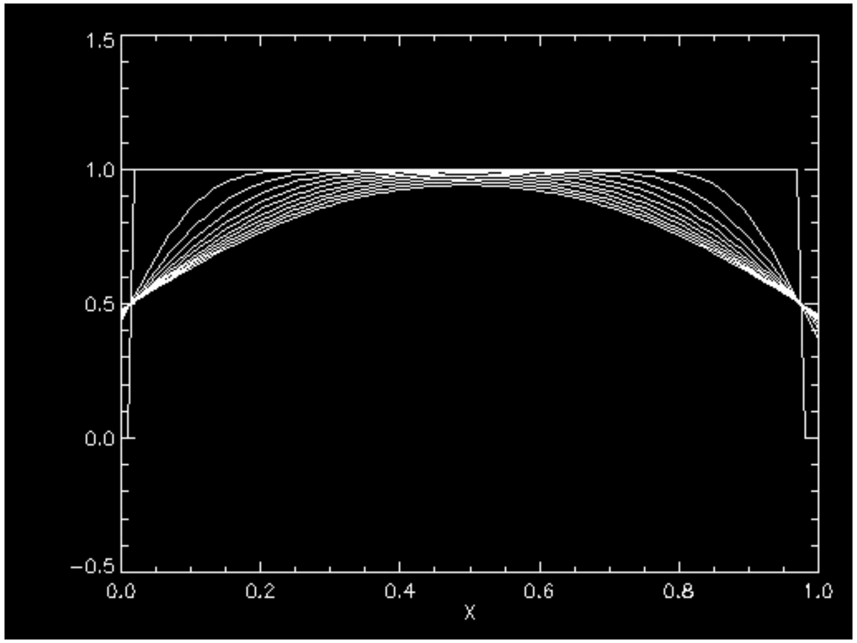
Leap Frog

the leap-frog seembs better!!

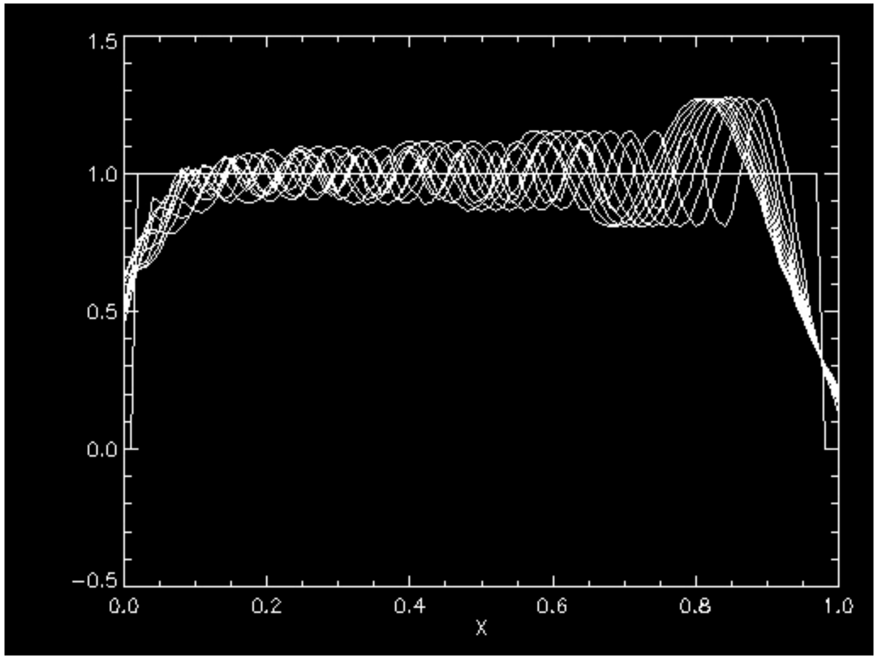
Is it always the case?

Not always... When the function is very steep and sharp the Leap-Frog can becomes somewhat unstable. So choose sigma << 1

Evolution of a top-hat impulse



upwind



Leap frog

This can be checked with Von Neuman stability analysis

Implicit Method:

Building a simple first order implicit method

$$\partial_t u(x,t) + c\partial_x u(x,t) = 0$$

$$\frac{\partial U_j^n}{\partial t} \approx \frac{U_j^{n+1} - U_j^n}{dt}$$

Space derivative is evaluated at time N+1 (ant not tim N)=> it is then an implicit equation!

$$\frac{\partial U_j^{n+1}}{\partial x} \simeq \frac{U_j^{n+1} - U_{j-1}^{n+1}}{dx}$$

So the implicit equation to solve is

$$\frac{U_j^{n+1} - U_j^n}{dt} + c\,\frac{U_j^{n+1} - U_{j-1}^{n+1}}{dx} = 0$$

We want to find U_j^{n+1} and U_{j-1}^{n+1} =>

$$U_j^{n+1}\left(\frac{1}{dt} + \frac{c}{dx}\right) - U_{j-1}^{n+1}\,\frac{c}{dx} = U_j^n\,\frac{1}{dt}$$

$$U_j^{n+1}\left(\frac{1}{dt}+\frac{c}{dx}\right)-U_{j-1}^{n+1}\frac{c}{dx}=U_j^n\frac{1}{dt}$$

First methode : Iterative (no matrix)

$$\boxed{U_j^{n+1}\left(\frac{1}{dt}+\frac{c}{dx}\right)=U_j^n\frac{1}{dt}+U_{j-1}^{n+1}\frac{c}{dx}}$$

Since U_j^n are computed in increasing order of j for every N the above equation
May be applied to each cell at time N+1, knowing all U_j^n

You need also U_0^{n+1} that in general you know because of boundary conditions.

So IF you know U_j^n and U_0^{n+1} THEN YOU CAN COMPUTE ALL U_j^{n+1} using the
Above relation .

Note : it require a simple boundary condition for U_0^{n+1}

Better technics using Matrix Algebra

$$U_j^{n+1}\left(\frac{1}{dt}+\frac{c}{dx}\right)-U_{j-1}^{n+1}\frac{c}{dx}=U_j^n\frac{1}{dt}$$

$$U_j^{n+1}\left(\frac{dx+cdt}{dtdx}\right)-U_{j-1}^{n+1}\frac{c}{dx}=U_j^n\frac{1}{dt}$$

Let define : a=(dx+cdt)/dtdx , b=-c/dx , c=1/dt

We write it in matrix Form : $AU^{+1}=BU^N$

$$\begin{array}{cccc} A & U^{N+1} = & B & U^N \\ \begin{pmatrix} a & & \\ b & a & \\ & b & a \end{pmatrix} & \begin{pmatrix} \dots \\ U_{j-1}^{N+1} \\ U_j^{N+1} \\ U_{j+1}^{N+1} \\ \dots \end{pmatrix} .= & \begin{pmatrix} c & & \\ & c & \\ & & c \end{pmatrix} & \begin{pmatrix} \dots \\ U_{j-1}^{N} \\ U_j^{N} \\ U_{j+1}^{N} \\ \dots \end{pmatrix} \end{array}$$

$$\begin{array}{cccc} A & U^{+1} & = \ B & U^{N} \\ \begin{pmatrix} a & & \\ b & a & \\ & b & a \end{pmatrix} & \begin{pmatrix} \dots \\ U_{j-1}^{N+1} \\ U_{j}^{N+1} \\ U_{j+1}^{N+1} \\ \dots \end{pmatrix} . = & \begin{pmatrix} c & & \\ & c & \\ & & c \end{pmatrix} & \begin{pmatrix} \dots \\ U_{j-1}^{N} \\ U_{j}^{N} \\ U_{j+1}^{N} \\ \dots \end{pmatrix} \end{array}$$

Then taking advantage of matrix algebra we can write :

$A\,U^{N+1} = BU^{N}$ => $\mathrm{A}^{-1}A\,U^{N+1} = A^{-1}BU^{N} \Rightarrow$

$$\boxed{U^{N+1} = A^{-1}BU_{N}}$$

So implicit scheme requires a matrix inversion

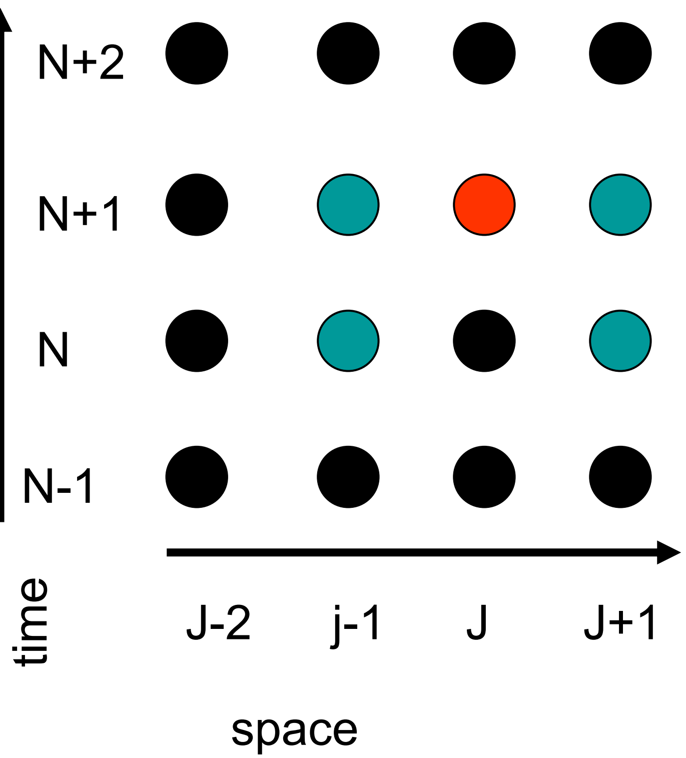
But we now KNOW HOW to compute $\mathrm{U}^{\mathrm{N+1}}$ as a function of U^{N}

A popular Implicit Method: Cranck-Nicholson

They are unconditionally stable.

Order 2 in space and Order 2 in time

Idea: Make time averages
(rather than space average)



$$\frac{\partial u}{\partial t} = \frac{U_j^{n+1} - U_j^n}{dt}$$

$$\frac{\partial u}{\partial x} = \frac{1}{2}\left[\left.\frac{\partial u}{\partial x}\right|_{t=n+1} + \left.\frac{\partial u}{\partial x}\right|_{t=n} \right] = \frac{1}{2}\frac{\left(U_{j+1}^{n+1} - U_{j-1}^{n+1}\right) + \left(U_{j+1}^{n} - U_{j-1}^{n}\right)}{2dx}$$

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0 \Rightarrow$$

$$U_j^{n+1} = U_j^n - \frac{cdt}{4dx}\left[\left(U_{j+1}^{n+1} - U_{j-1}^{n+1}\right) + \left(U_{j+1}^{n} - U_{j-1}^{n}\right)\right]$$

Implicit scheme

$$U_j^{n+1} = U_j^n - \frac{cdt}{4dx}\left[\left(U_{j+1}^{n+1} - U_{j-1}^{n+1}\right) + \left(U_{j+1}^n - U_{j-1}^n\right)\right]$$

$$\sigma = cdt/dx$$

By grouping the terms into U^{n+1} with U^n

$$U_j^{n+1} + \frac{\sigma}{4}\left(U_{j+1}^{n+1} - U_{j-1}^{n+1}\right) = U_j^n - \frac{\sigma}{4}\left(U_{j+1}^n - U_{j-1}^n\right)$$

$$\Rightarrow$$

$$AU^{n+1} = BU^n$$

$$U^{n+1} = A^{-1}BU^n$$

$$U_j^{n+1}+\frac{\sigma}{4}\left(U_{j+1}^{n+1}-U_{j-1}^{n+1}\right)=U_j^{n}-\frac{\sigma}{4}\left(U_{j+1}^{n}-U_{j-1}^{n}\right)$$

$$\Rightarrow AU^{n+1}=BU^{n}\Rightarrow U^{n+1}=A^{-1}BU^{n}$$

So implicit scheme requires a matrix inversion:

$$A=\begin{pmatrix}1 & \sigma/4 & 0 & 0 & 0\\ -\sigma/4 & 1 & \sigma/4 & 0 & 0\\ 0 & -\sigma/4 & 1 & \sigma/4 & 0\\ 0 & 0 & -\sigma/4 & \ldots & \ldots\\ 0 & 0 & 0 & -\sigma/4 & 1\end{pmatrix};B=\begin{pmatrix}1 & -\sigma/4 & 0 & 0 & 0\\ \sigma/4 & 1 & -\sigma/4 & 0 & 0\\ 0 & \sigma/4 & 1 & -\sigma/4 & 0\\ 0 & 0 & \sigma/4 & \ldots & \ldots\\ 0 & 0 & 0 & \sigma/4 & 1\end{pmatrix}$$

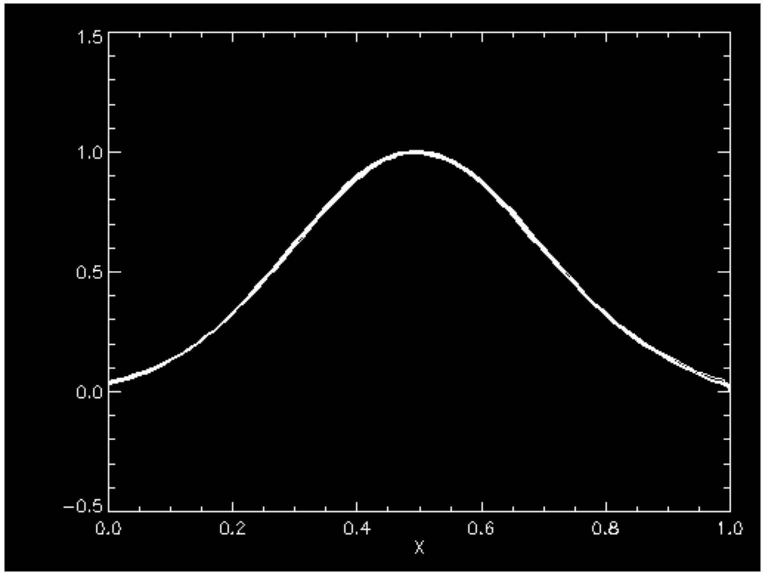
On prendra comme condition limite : U(j=0)=0, et U(j=J_{max})=0

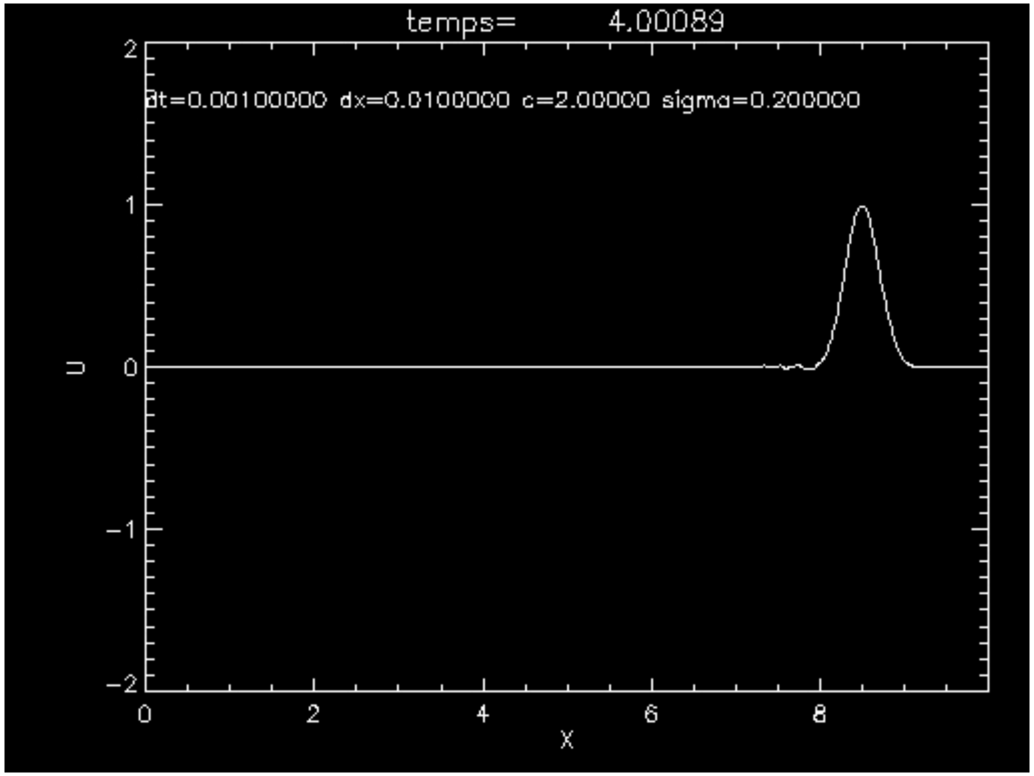
Cranck Nicholson Stability:

$$g(k) = \frac{1 - \sigma / 2i \sin(dx)}{1 + \sigma / 2i \sin(dx)}$$

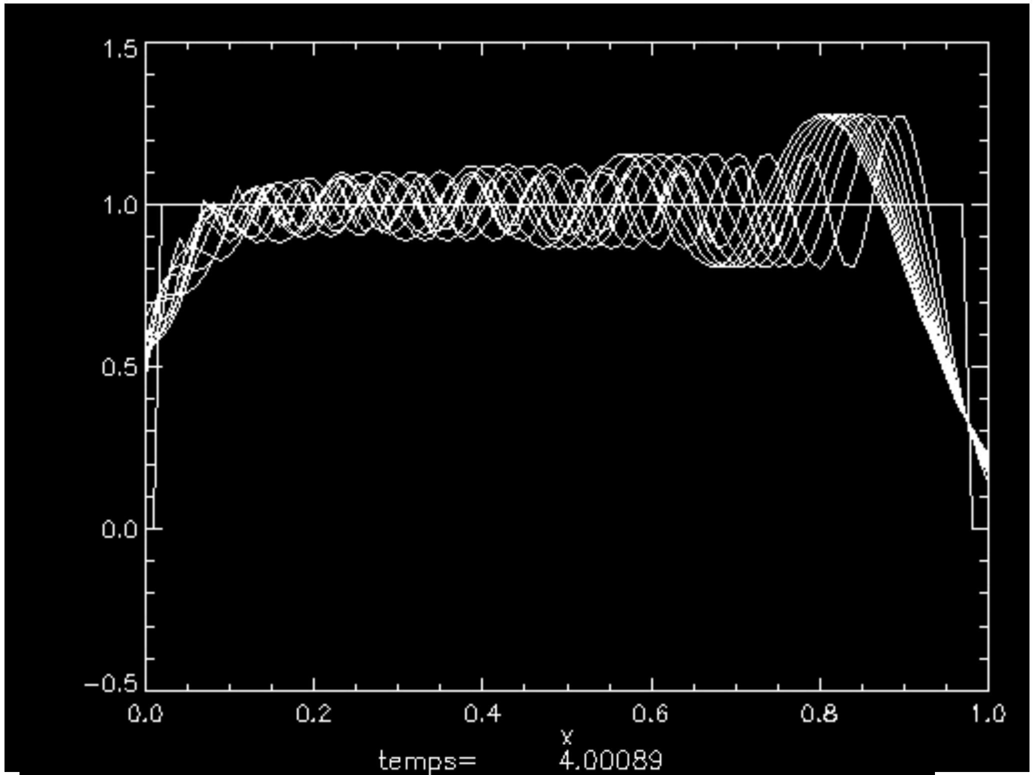
So the g(k) norm is 1 for any k, so it's a method unconditionally Convergent.

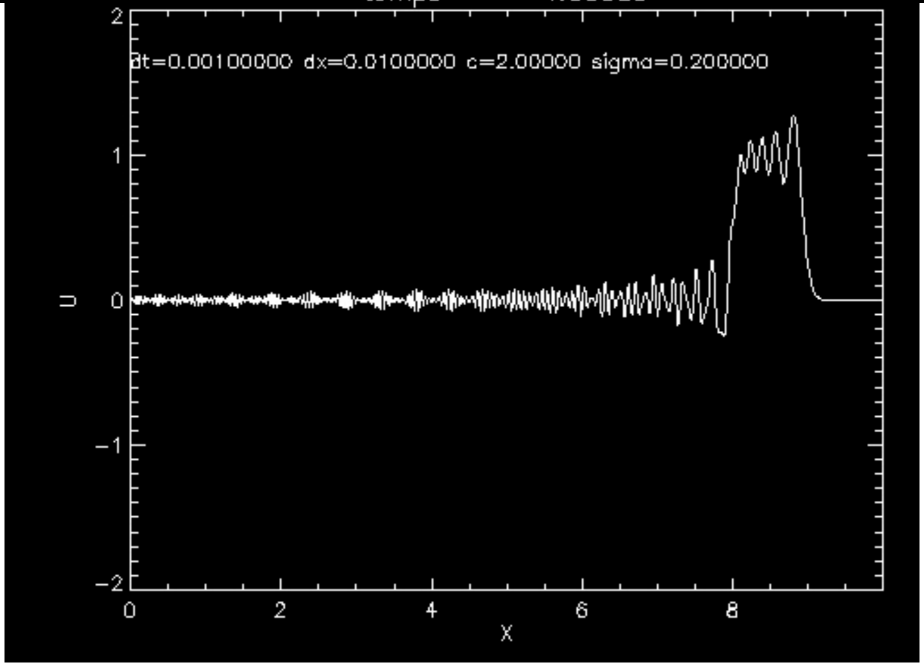
It is suitable for very smooth problems, but produces spurious oscillations where there are discontinuities.





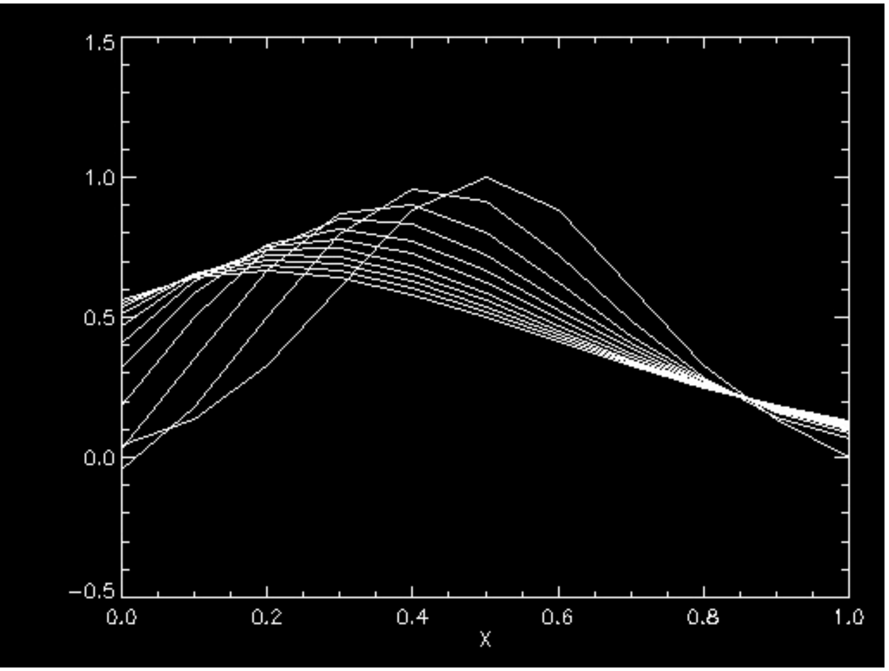
Evolution of the solution
The solution is stable
Here we have 1000 cells in x
Notice the small oscillations
at the base of the peak

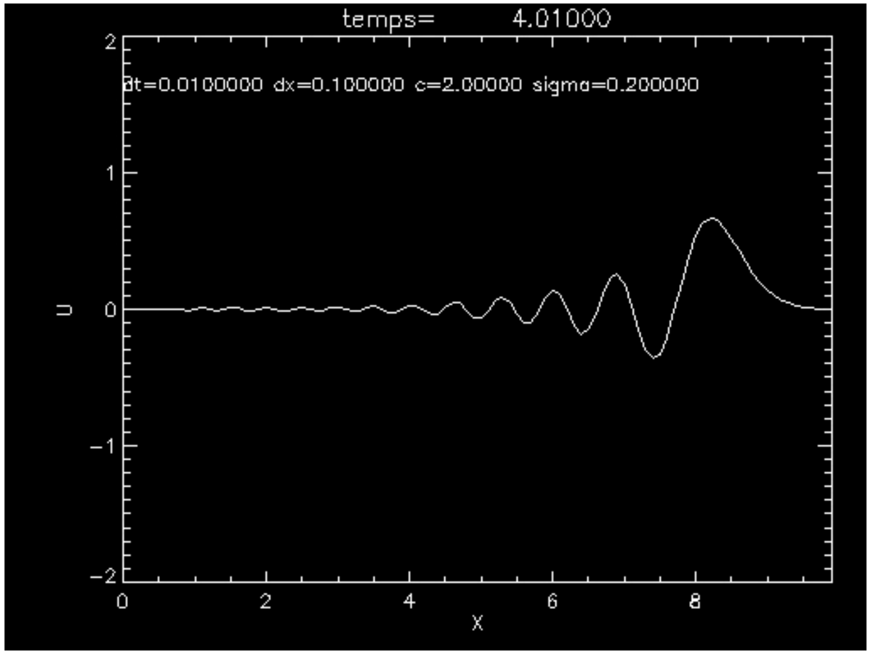




But for a function discontinuous, we get Oscillations
(here a top-hat function)
....

Let's take over the Gaussian function, but let's reduce the number of cells in X





Evolution of function global view

We see that when we decrease the nb of boxes in X, the function becomes more discontinuous and spurious oscillationsappear

Cranck Nicholson est souvent une bonne méthode

Mais plutôt pour des fonctions douces

Pour une fonction très discontinue, on utilisera plutôt la méthode upwind

The classification of partial derivatives equations of order 2:

General form of a second-order equa.diff:
u is a vector: (x1, x2, ...,xn)

$$A\frac{\partial^2 u}{\partial x^2} + B\frac{\partial^2 u}{\partial x \partial y} + C\frac{\partial^2 u}{\partial y^2} + D\frac{\partial u}{\partial x} + E\frac{\partial u}{\partial y} + Fu + G = 0$$

with $A, ..G = const.$ This equation is called $\left\{ \begin{array}{ll} elliptic\, for & B^2 - 4AC < 0 \\ parabolic\, for & B^2 - 4AC = 0 \\ hyperbolic\, for & B^2 - 4AC > 0 \end{array} \right.$

If G=0 , the equation is called "homogeneous."

We're calculating The "Discriminant":
The sign of Δ gives us the type of equation

if $\Delta > 0$ then the equation is Hyperbolic : (A>0, B=0, C<0 => B^2-4AC>0)

Wave propagation : $$\frac{\partial^2 f}{\partial t^2} - c^2 \frac{\partial^2 f}{\partial x^2} = 0$$

Advection Equation: $$\frac{\partial f}{\partial t} + c \frac{\partial f}{\partial x} = 0$$

if Δ = **0** the equation is called Parabolic : (A=0, B=0, C>0 => B^2-4AC=0)

Heat diffusion: $$\frac{\partial T}{\partial t} = k \frac{\partial^2 T}{\partial x^2}$$

Shrodinger Equation : $$\frac{\partial U}{\partial t} = -i\left(\frac{\partial^2 U}{\partial x^2} + U\right)$$

Si $\Delta<0$ the equation is called Elliptical : (A>0, B=0, C>0 => B^2-4AC<0)

Laplace Equation $$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0$$

Poisson Equation : $$\rho(x,y,z) = -\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2}\right)V(x,y,z)$$

ρ= charge density, V: electric potential

Second order derivatives

Many equations in physics involve a second order derivative.
It can be approximated with the same type of reasoning as for the first derivative

$$U_{j+1}^{n} = U_j^n + dx\frac{\partial U_j^n}{\partial x} + \frac{dx^2}{2!}\frac{\partial^2 U_j^n}{\partial x^2} + O(dx^2)$$

$$U_{j-1}^{n} = U_j^n - dx\frac{\partial U_j^n}{\partial x} + \frac{dx^2}{2!}\frac{\partial^2 U_j^n}{\partial x^2} + O(dx^2)$$

Once again we are inspired by the Taylor's expansion...

$$\Rightarrow$$

$$(1)\;\; U_{j+1}^{n} - U_j^n = dx\frac{\partial U_j^n}{\partial x} + \frac{dx^2}{2!}\frac{\partial^2 U_j^n}{\partial x^2} + O(dx^2)$$

$$(2)\;\; U_{j+2}^{n} - U_j^n = 2dx\frac{\partial U_j^n}{\partial x} + 2dx^2\frac{\partial^2 U_j^n}{\partial x^2} + O(dx^2)$$

So we find:

$$\frac{\partial^2 U_j^n}{\partial x^2} \approx \frac{U_{j+2}^n - 2U_{j+1}^n + U_j^n}{dx^2} + o(dx)$$ FORWARD difference

By shifting the position, we can find other approximations

$$\frac{\partial^2 U_j^n}{\partial x^2} \approx \frac{U_{j-2}^n - 2U_{j-1}^n + U_j^n}{dx^2} + o(dx)$$ BACKWARD diference

$$\frac{\partial^2 U_j^n}{\partial x^2} \approx \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{dx^2} + o(dx^2)$$ CENTERED difference (order 2!)

The same formulas apply for time derivatives (replace x by t)
Note that the CENTERED difference is always better (order 2)

Cross derivatives can also be calculated in the same way:
Example

$$\frac{\partial^2 U_j^n}{\partial x \partial t} \approx \frac{U_{j+1}^{n+1} - U_j^{n+1} - U_{j+1}^n + U_j^n}{dxdt} + o(dx) + o(dt)$$ FORWARD difference

$$\frac{\partial^2 U_j^n}{\partial x \partial t} \approx \frac{U_{j+1}^{n-1} - U_j^{n-1} - U_{j-1}^n + U_j^n}{dxdt} + o(dx) + o(dt)$$ BACKWARD difference

$$\frac{\partial^2 U_j^n}{\partial x \partial t} \approx \frac{U_{j+1}^{n+1} - U_{j-1}^{n+1} - U_{j+1}^{n-1} + U_{j-1}^{n-1}}{4dxdt} + o(dx^2) + o(dt^2)$$ CENTERED difference

Once again the centered diferences are of order 2

HYPERBOLIC equations

We have already studied the advection equation

Another example: the propagation of the sound waves

These equations have all "characteristics" , they propagate an initial condition.

Order 2 methods work well for soft functions

For stiff functions one would prefer a 1"UPWIND" method

See course

PARABOLIC equations (or diffusion equation). Quite simple to solve in general

Example: heat equation (Fourrier)
$$\boxed{\frac{\partial U}{\partial t} - \lambda \frac{\partial^2 U}{\partial x^2} = 0}$$

λ=thermal diffusivity

Simplest Method: Explicit

Time: forward difference
$$\frac{\partial U_j^n}{\partial t} \approx \frac{U_j^{n+1} - U_j^n}{dt}$$

Space: centered difference
$$\frac{\partial^2 U_j^n}{\partial x^2} \approx \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{dx^2} + o(dx^2)$$

We replace and find the scheme:

$$U_j^{n+1} = (1-2\sigma)U_j^n + \sigma(U_{j-1}^n + U_{j+1}^n)$$

$$où\ \sigma = \lambda dt / dx^2$$

Note the new expression of σ (CFL)

Stability? VON NEUMAN METHODE again $u_k^{n+1} = g(k)\, u^n_k$

$$U_j^{n+1} = (1-2\sigma)U_j^n + \sigma(U_{j+1}^n + U_{j-1}^n)$$

$$\Rightarrow$$

$$\sum_k u_k^{n+1} e^{ikj\,dx} = (1-2\sigma)\sum_k u_k^n e^{ikj\,dx} + \sigma\left(\sum_k u_k^n e^{ik(j+1)\,dx} + \sum_k u_k^n e^{ik(j-1)\,dx}\right)$$

$$\Rightarrow$$

$$\sum_k u_k^{n+1} e^{ikj\,dx} = (1-2\sigma)\sum_k u_k^n e^{ikj\,dx} + (e^{-ikdx} + e^{+ikdx})\sigma \sum_k u_k^n e^{ikj\,dx}$$

By pooling term to term we find $\quad g(k) = 1 - 4\sigma \sin^2\left(\frac{kdx}{2}\right)$

$$\left\| g(k)^2 \right\| \leq 1 \Leftrightarrow dt \leq \frac{dx^2}{2\lambda} \Leftrightarrow \sigma < \frac{1}{2}$$

A condition close to the courrant condition (CFL) is again found

In practice: this explicit method is therefore STABLE if CFL is respected, however
As it is only order 1 in time it is very diffusive and not very good

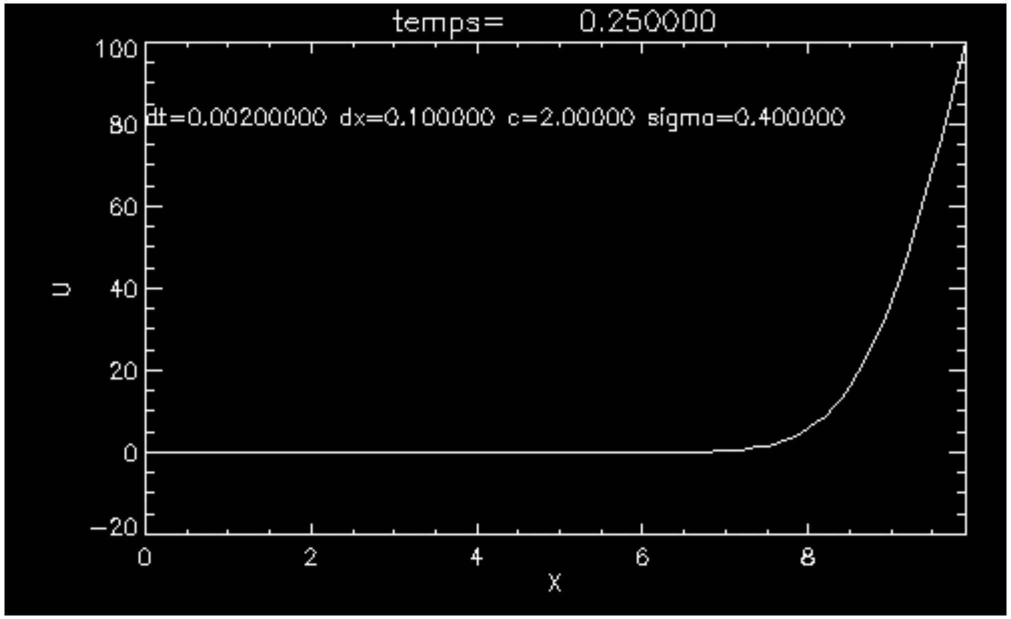
What means $dx^2/2\lambda$?

λ is the coefficient of diffusion In kinetic theory, gases are shown (see the course on the ideal gases) that the time it takes to diffuse over a distance dx is:
dx^2/ λ ... So it's a physical quantity.
This is the new Courrant condition (CFL) for heat.

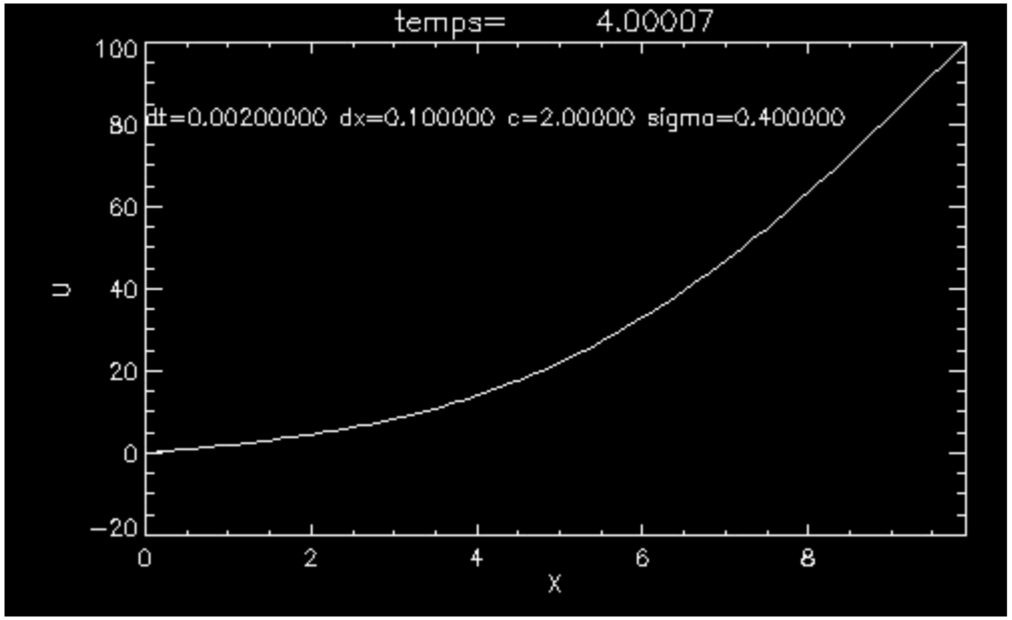
Lecture on heat diffusion :
www.unice.fr/lpmc/bdd/suprex/ download/Cours-thermo-ch3.doc

Ex: heat diffusion in a bar

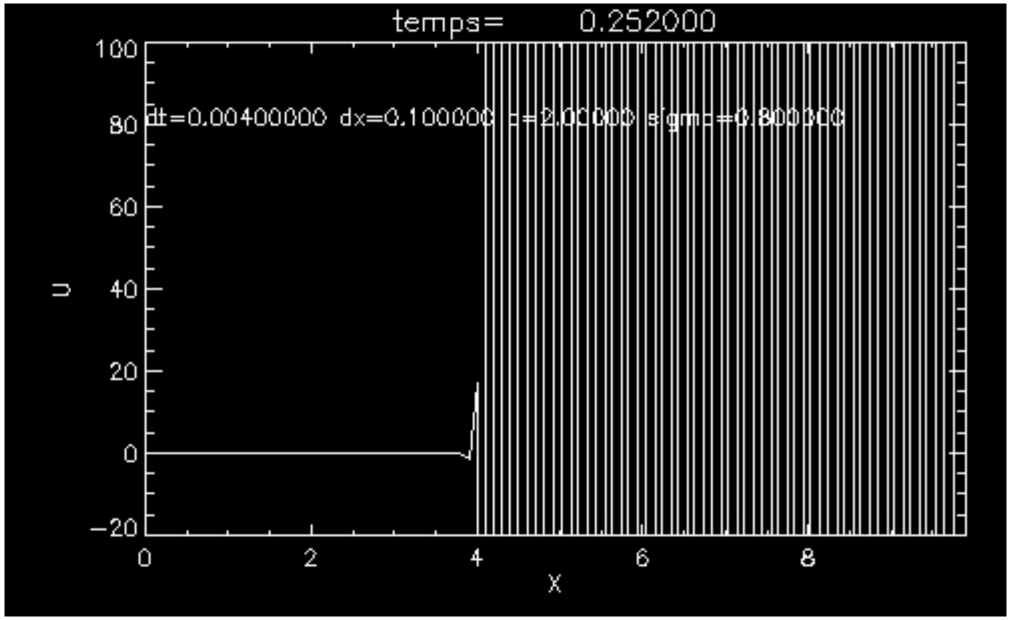
here SIGMA=0.4



Not bad…



What if we increase the time step ?



here SIGMA=0.8

…. Results is violently wrong…

Implicit methods work better

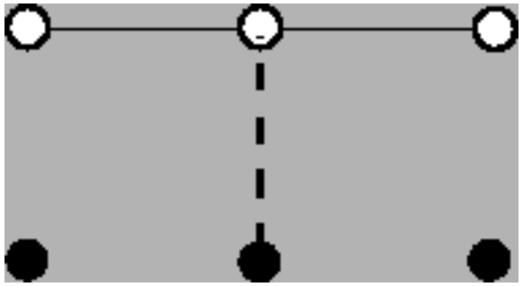
Implicit Method of Order 1 (in time)

Time : idem

$$\frac{\partial U_j^n}{\partial t} \approx \frac{U_j^{n+1} - U_j^n}{dt}$$

Space : same as before, but
Taking time n+1 (implicit)

$$\frac{\partial^2 U_j^n}{\partial x^2} \approx \frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{dx^2} + o(dx^2)$$



We get the following equation , that must be solved (implicit)

$$-\sigma U_{j+1}^{n+1} + (1+2\sigma)U_j^{n+1} - \sigma U_{j-1}^{n+1} = U_j^n$$

$$o\grave{u}\ \sigma = \lambda dt / dx^2$$

To solve, we take a matrix approach again

$$\mathrm{A}U^{n+1} = U^{n} \quad \Rightarrow \quad U^{n+1} = \mathrm{A}^{-1}U^{n}$$

With A :

$$A = \begin{pmatrix} 1+2\sigma & -\sigma & 0 & 0 & 0 \\ -\sigma & 1+2\sigma & -\sigma & 0 & 0 \\ 0 & -\sigma & 1+2\sigma & -\sigma & 0 \\ 0 & 0 & -\sigma & \dots & \dots \\ 0 & 0 & 0 & -\sigma & 1+2\sigma \end{pmatrix}$$

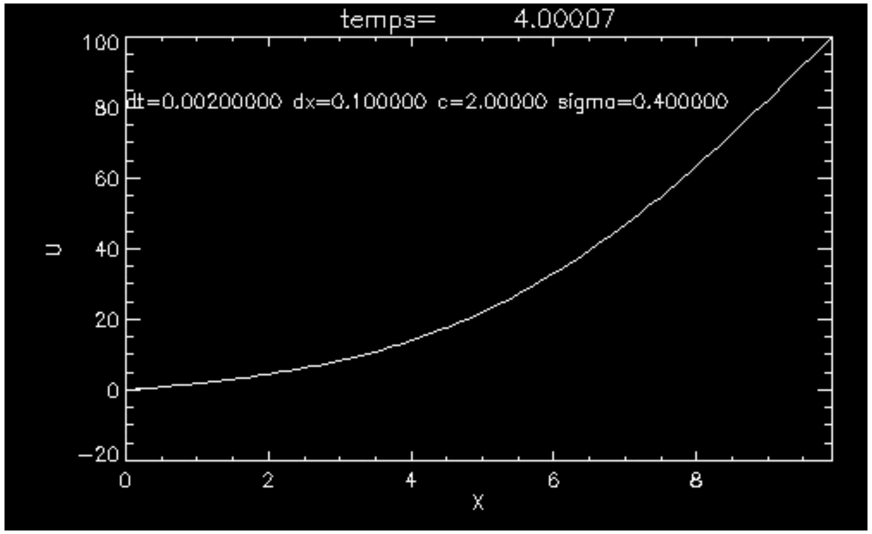
Matrix again tri-diagonal easily inversible

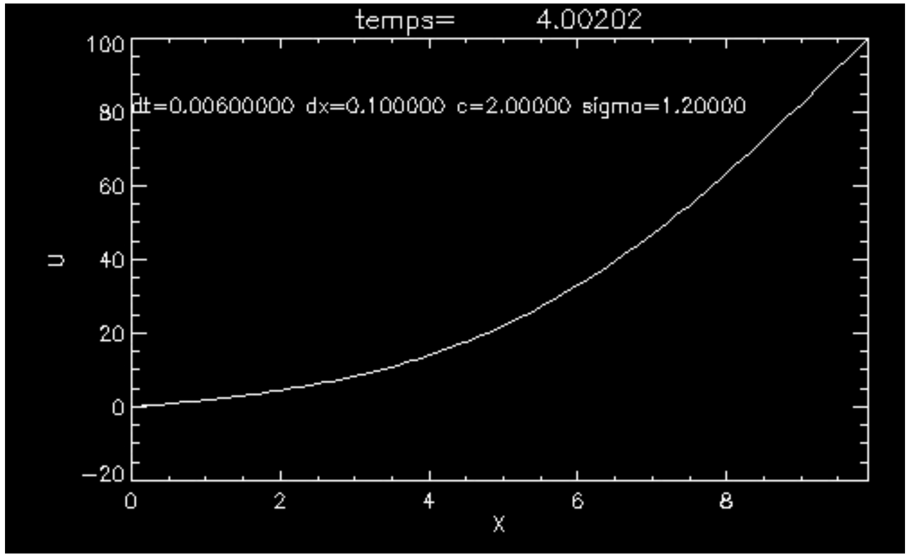
Attention will be paid to the boundary conditions when integrating

Stability? Von Neuman's method again

It is found that
$$g(k) = \frac{1}{1 + 4\sigma \sin^2\left(\frac{kdx}{2}\right)}$$

threrefore $|| g(k)||^2 < 1$ in all circumstances - unconditionally stable





Small time step, sigma=0.4

big time step: sigma=1.2

Here we can clearly see the superiority of the implicit method

Cranck Nicholson (very popular solver)

Time : as usual

$$\frac{\partial U_j^n}{\partial t} \approx \frac{U_j^{n+1} - U_j^n}{dt}$$

Space : we average at time n-1 and n+1

$$\frac{\partial^2 U_j^n}{\partial x^2} \approx \frac{1}{2}\left(\frac{U_{j+1}^{n+1} - 2U_j^{n+1} + U_{j-1}^{n+1}}{dx^2} + \frac{U_{j+1}^{n} - 2U_j^{n} + U_{j-1}^{n}}{dx^2} \right) + o(dx^2)$$

In fact it's a 2-order scheme at time time n+1/2 (magic trick !!)

We get

$$-\sigma U_{j-1}^{n+1} + (1+2\sigma)U_j^{n+1} - \sigma U_{j+1}^{n+1} = \sigma U_{j-1}^{n} + (1-2\sigma)U_j^{n} + \sigma U_{j+1}^{n}$$

That one writes using matrices:

$$AU^{n+1} = BU^n$$

It will therefore be necessary to calculate A^{-1}. If you define $C=A^{-1}B$ we then get

The simple relationship $U^{n+1}= C\ U^{n}$

$$\mathbf{A}\equiv\begin{pmatrix} 1 & 0 & 0 & . & . & 0 \\ -a & 1+2a & -a & 0 & . & 0 \\ 0 & . & . & . & 0 & . \\ . & . & . & . & . & . \\ . & . & . & -a & 1+2a & -a \\ . & . & . & 0 & 0 & 1 \end{pmatrix} \quad \mathbf{B}\equiv\begin{pmatrix} 1 & 0 & 0 & . & . & 0 \\ a & 1-2a & a & 0 & . & 0 \\ 0 & . & . & . & 0 & . \\ . & . & . & . & . & . \\ . & . & . & a & 1-2a & a \\ . & . & . & 0 & 0 & 1 \end{pmatrix}$$

Ici σ=a

Stability : $||g(k)||^{2}=$ $\dfrac{1-2a\sin^2(k\Delta x/2)}{1+2a\sin^2(k\Delta x/2)}\leq 1\,,$

TESTS: it works well, it's more accurate than previous methods

ELLIPTICAL equations (or boundary conditions problems)

Elliptical equations **do not depend** on time. They only depend on boundaries.
For example: Equation for Electric Potential

laplacian $\longrightarrow$

$$\nabla^2 U = -\rho(x,y,z) \Leftrightarrow$$

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} + \frac{\partial^2 U}{\partial z^2} = \frac{-\rho(x,y,z)}{\varepsilon_0}$$

U is the potential field, is the distribution of electrical charges in space

We encounter the same type of equation in fluid mechanics:
For a permanent, viscous, non-turbulent flow, incompressible, we have:

$$\nabla^2 v = -\frac{grad(P)}{\rho\eta}$$

Poiseuille flow

In any case, we can see that time does not play .
This means that one calculates A FIELD : you have to calculate the value of U for all space point.

For example the potential field, the speed field, the gravitational field etc...

This field depends solely on its boundary conditions:
The distribution of loads, masses, flow at the edges of the domain etc...

These problems are called » Boundary conditions problems »

They are usually quite complex to solve. We'll fly over them quickly

There are a few matrix methods, but the best are the methods
spectral, i.e. where the system is resolved in the Fourrier space and then converted

in real space.

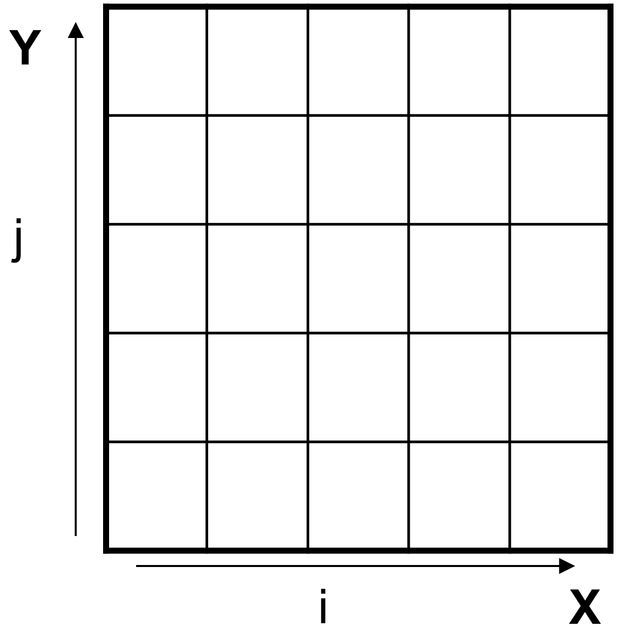
Example: Let's solve the field generated by a 2D electric charge

$$\frac{\partial^2 U}{\partial x^2} + \frac{\partial^2 U}{\partial y^2} = \frac{-\rho}{\varepsilon_0}$$

Taking the expression of the second derivative as the second derivative, the difference centered, one Leads to an equation like this

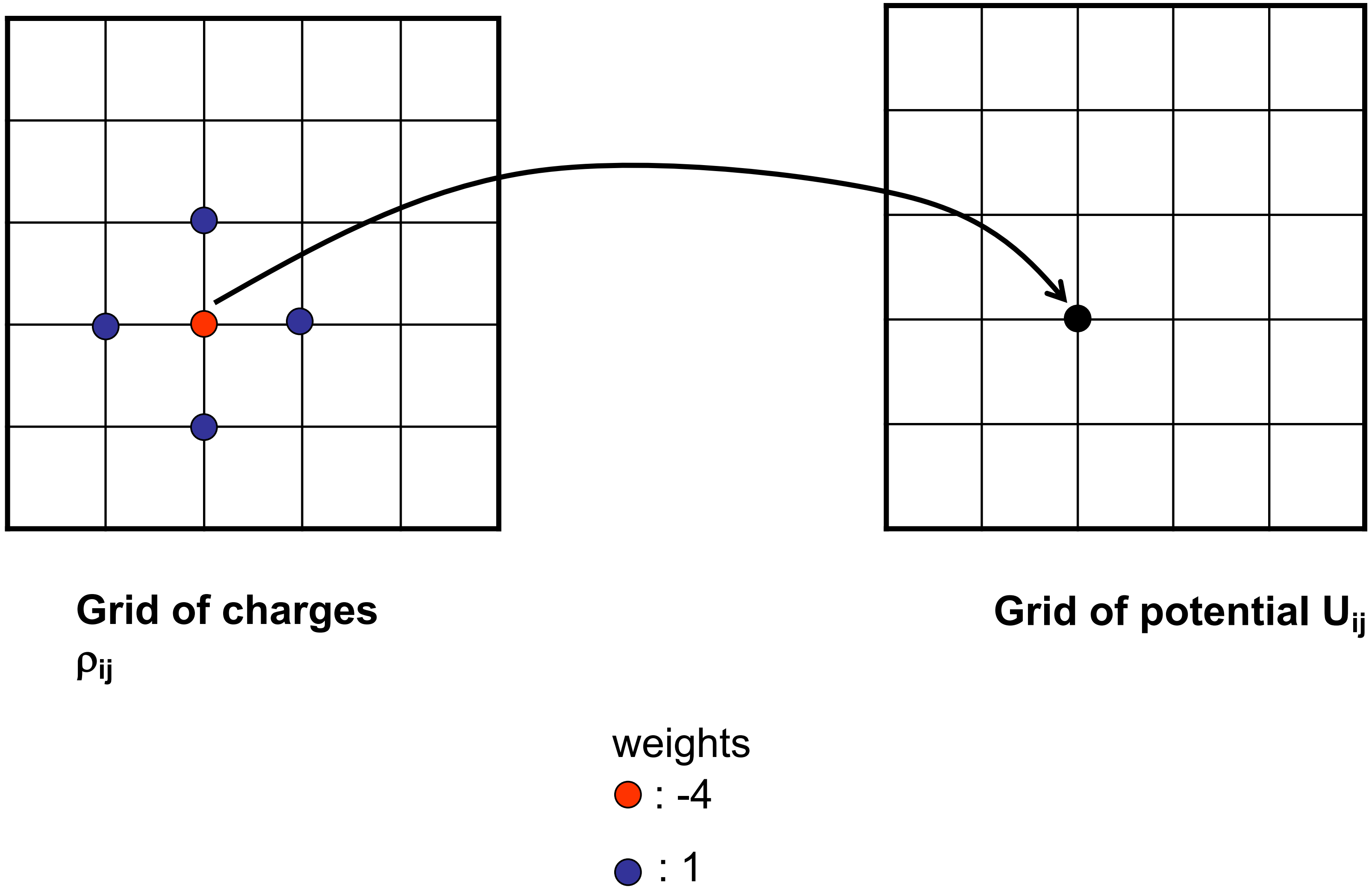
$$\frac{1}{(\Delta l)^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right] = -\rho_{i,j}$$

Electric charge at the i,j knot



U(i,j) is U's value at the position :
X=i x dL et Y= j x dl

$$\frac{1}{(\Delta l)^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right] = -\rho_{i,j}$$

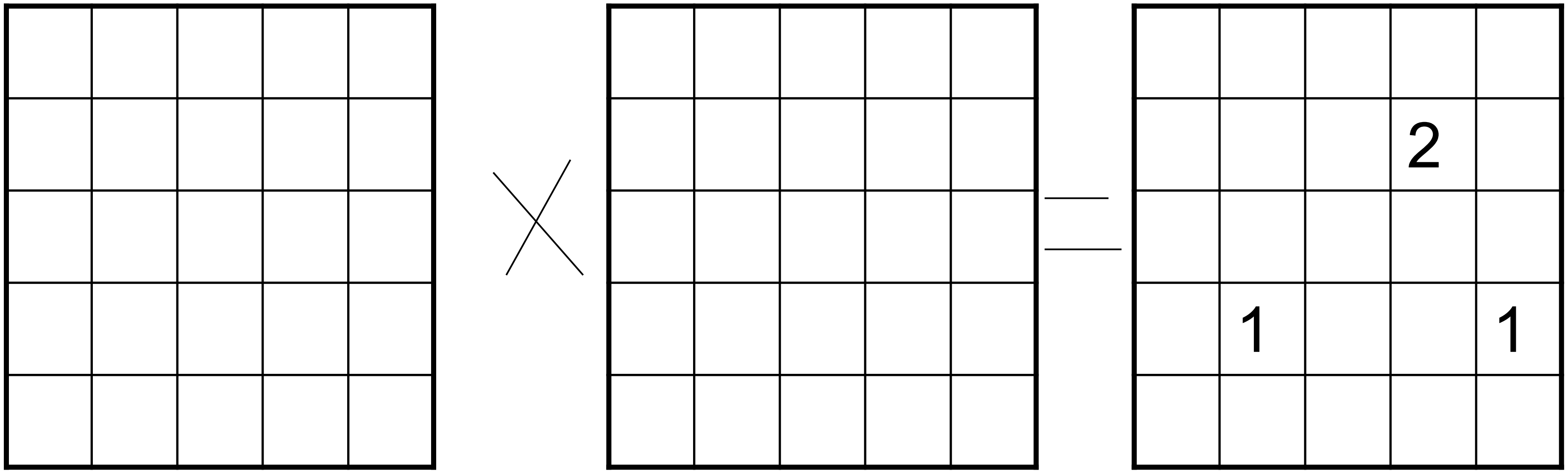


Why it is not easy to solve an elliptical equation Simply by using matrixes like for Heat or transport equations ?

$$\frac{1}{(\Delta l)^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right] = -\rho_{i,j}$$

How to write the solved in matric form ? **Is it possible ??**

L x U = C



LAPLACIAN

U

Charge distribution

Let’s have a look at the equation more closely

$$\frac{1}{(\Delta l)^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right] = -\rho_{i,j}$$

Shifting a line Down

Shifting a line Up

Shifting a column on the right

Shifting a Column on the left

To shift the lines we can multiply by:

$$\begin{array}{c} \begin{bmatrix} & & & \\ 1 & & & \\ & 1 & & \\ & & 1 & \end{bmatrix} \\ S^{-1} \end{array} \quad \text{or} \quad \begin{array}{c} \begin{bmatrix} & 1 & & \\ & & 1 & \\ & & & 1 \\ & & & \end{bmatrix} \\ S^{+1} \end{array}$$

But to shift the columns?

The only way is to use the transposed matrix

Shifting a a column: shift the lines of the transposed and take the transposed result $(S\text{-}1\ A^T)^T$

Let's think about it....

Problem !!!! =>

Matrix transposition is NOT a linear operation

it is not possible to compute a« column shift » by a simple matrix multiplication...!!!!

An equation of the type : $A X + B X^T = C$ is not a linear operation ….

So we can NOT simply write this problem in the form of a
Simple multiplication of matrixes...
You can get away with it by writing a system of N^2 équations with N^2 Unknown
(so a matrix containing N4 terms)... which is long and VERY ineffective...

This would require to consider a matrix with size $N^2 x N^2 = N^4$ elements !

So we see that elliptical equations are very expensive in memory and computation

Example: all problems with N body where the gravity field equation is very
computationally intensive

Solving the Poisson equation

Slow relaxation method : the Gauss Seidel method, and Jacobi Method

$$\frac{1}{(\Delta l)^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right] = -\rho_{i,j}$$

The Gauss-Siedel method is an iterative procedure that converges progressively
To the solution by successive steps of approximation.

Let's call U^n the system at step n, so that $U^{n+1} = f(U^n)$. With f an non-linear
(and maybe complex transformation)

We try to find a a way that when n goes to infinity U^n becomes solution of the above
Equation (the poisson equation)

Starting from the above equation :

$$\frac{1}{(\Delta l)^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right] = -\rho_{i,j}$$

We write

$$U_{i,j} = \frac{\left(U_{i+1,j} + U_{i-1,j} + U_{i,j+1} + U_{i,j-1}\right)}{4} + \frac{\Delta l^2 \rho_{i,j}}{4}$$

Then we interpret this equation as an iterative procedure :

$$\underset{\substack{\uparrow \\ \text{Step N+1}}}{U_{i,j}^{N+1}} = \frac{\left(U_{i+1,j}^{N} + U_{i-1,j}^{N} + U^{N}{}_{i,j+1} + U_{i,j-1}^{N}\right)}{\underset{\substack{\uparrow \\ \text{Step N}}}{4}} + \frac{\Delta l^2 \rho_{i,j}}{4}$$

If U converges to the solution of the poisson equation, then the solution is the Convergence point of the iterative procedure

This is the JACOBI iteration: Every point (i,j) is calculated individually following

$$U_{i,j}^{N+1} = \frac{\left(U_{i+1,j}^{N} + U_{i-1,j}^{N} + U^{N}{}_{i,j+1} + U_{i,j-1}^{N}\right)}{4} + \frac{\Delta l^2 \rho_{i,j}}{4}$$

At every step N you have to chek that $\epsilon(N) = \max\left(abs(U^{N+1} - U^{N})\right)$ gets smaller Than a small pre-defined value

The above equation must be modified for the boundary conditions:
$U^N{}_{0,j}$, $U^N{}_{i,0}$, $U^N{}_{imax,j}$, $U^N{}_{i,jmax}$

So the Jacobi method for solving the Poisson Equation goes as follows :

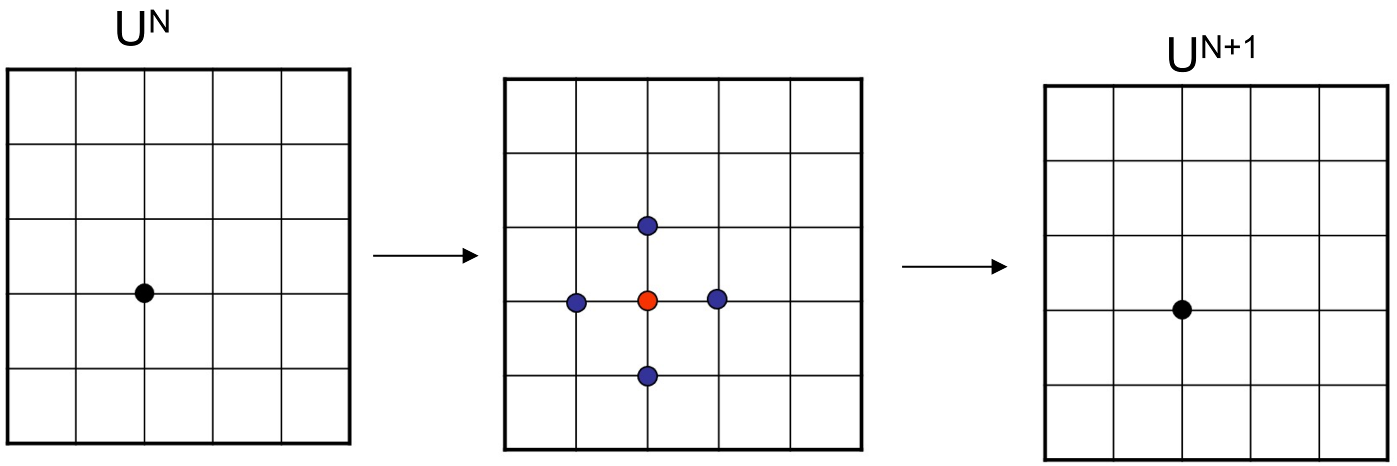
1- Define the grid of charges $\rho_{i,j}$
2- Start with U^0 =0 everywhere
3- Compute

$$U_{i,j}^{N+1} = \frac{\left(U_{i+1,j}^{N} + U_{i-1,j}^{N} + U^{N}{}_{i,j+1} + U_{i,j-1}^{N}\right)}{4} + \frac{\Delta l^2 \rho_{i,j}}{4}$$

4- Apply boundary conditions
5- Check for every (I,j) : $\epsilon_{i,j} = abs(U^{N+1} - U^{N})$, take the max of $\epsilon_{i,j}$
6- if $\max(\epsilon_{i,j}) < \max\_error$ (max_error is predefined) then STOP
7- Otherwise go back to 3

This is the JACOBI iteration: Every point (i,j) is calculated individually following

$$U_{i,j}^{N+1} = \frac{\left(U_{i+1,j}^{N} + U_{i-1,j}^{N} + U^{N}{}_{i,j+1} + U_{i,j-1}^{N}\right)}{4} + \frac{\Delta l^2 \rho_{i,j}}{4}$$



The 4 neighbors are
Used to compute U^{N+1} for each
(I,j)
This operation is also called a
“convolution”

weights

🟥 : -4

🟦 : 1

Variants : The Gauss-Seidel and and SOR method to accelerate convergence

Gauss- Siedel : Going from i=0 to i=i_{max} and from j=0 to j=j_{max}

Note that $U_{i,0}$ and $U_{0,j}$ are given by boundary conditions :

$$U_{i,j}^{N+1} = \frac{\left(U_{i+1,j}^{N} + U_{i-1,j}^{N+1} + U^{N}{}_{i,j+1} + U_{i,j-1}^{N+1}\right)}{4} + \frac{\Delta l^2 \rho_{i,j}}{4}$$

These 2 terms has been computed PREVIOUSLY
When computing $U^{N+1}_{i,j}$

So this is an implicit method

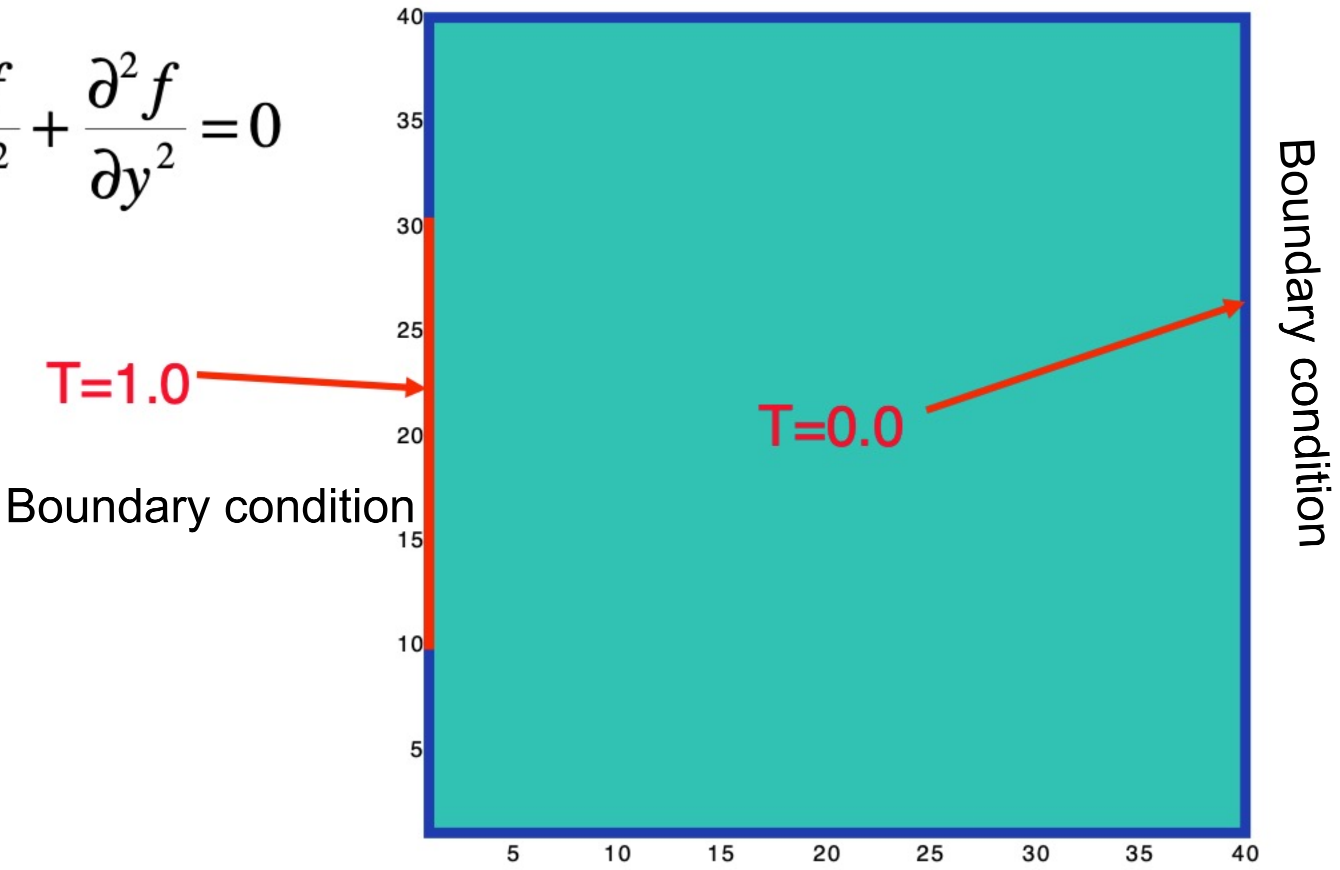
SOR Method : apply a coefficient to accelerate convergence: Over Relaxation

$$U_{i,j}^{N+1} = \frac{\beta\left(U_{i+1,j}^{N} + U_{i-1,j}^{N+1} + U^{N}{}_{i,j+1} + U_{i,j-1}^{N+1} - \Delta l^2 \rho_{i,j}\right)}{4} - \frac{(1-\beta)U_{i,j}^{N}}{4}$$

With 1<β<2. β =1.5 is often a good choice.
This technics is called “Over relaxation”.

Example: solving the steady state temperature distribution in a square Grid with boundary conditions.
The equation is Laplacian(T)=0 + Boundary conditions on the edges Of the domain

$$\frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} = 0$$



The equation to solve written in finite difference is (U=temperature at i,j):

$$U_{i,j} = \frac{(U_{i+1,j}+U_{i-1,j}+U_{i,j+1}+U_{i,j-1})}{4}$$, but on edges $U_{i,j}$ =0 or 1

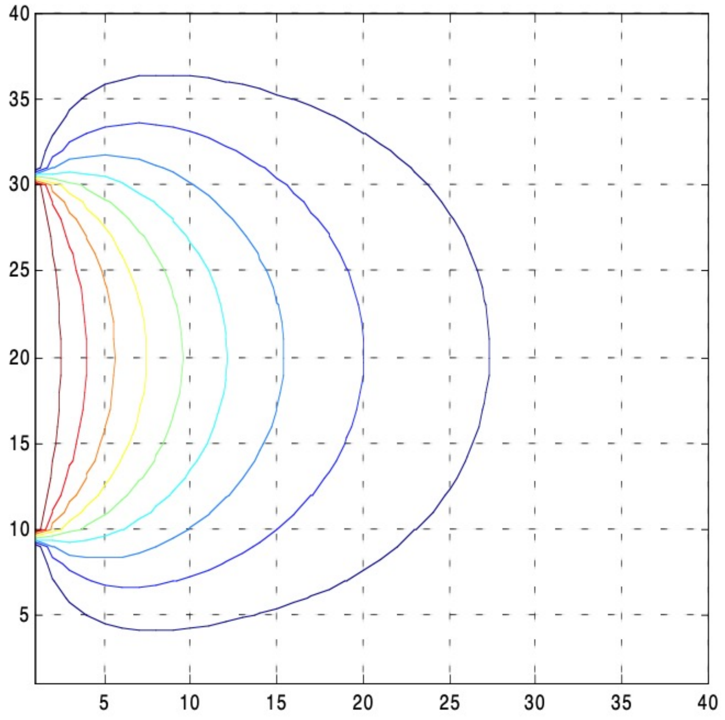
Python implementation

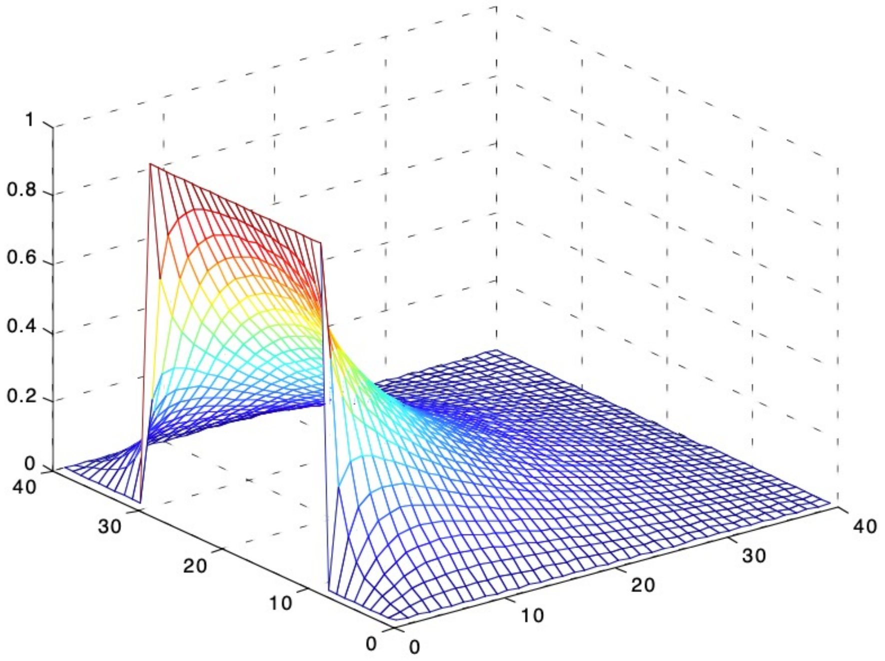
```
% two-dimensional steady-state problem by SOR
n=40;m=40;nstep=5000;alpha=0.05;length=2.0;h=length/(n-1);
T=zeros(n,m);bb=1.7;
T(10:n-10,1)=1.0;
for l=1:nstep,
   for i=2:n-1, for j=2:m-1
      T(i,j)=bb*0.25*(T(i+1,j)+...
             T(i,j+1)+T(i-1,j)+T(i,j-1))+(1.0-bb)*T(i,j);
   end,end
   % find residual
   res=0;
   for i=2:n-1, for j=2:m-1
      res=res+abs(T(i+1,j)+...
             T(i,j+1)+T(i-1,j)+T(i,j-1)-4*T(i,j))/h^2;
   end,end
   l,res/((m-2)*(n-2)) % Print iteration and residual
   if (res/((m-2)*(n-2)) < 0.001), break,end
end;
contour(T);
```

Average absolute error: 0.001

	Number of iterations
Jacobi:	1989
Gauss-Seidler:	986
SOR (1.5):	320
SOR (1.7):	162
SOR (1.9):	91
SOR (1.95):	202

Solution after convergence





Note : you have to keep in mind that we have computed here the “steady-state” , there is NO time evolution here. The iteration (N-> N+1) “mimics” a time evolution but it is not physical.

Other method: Introduction to spectral methods

$$\frac{1}{(\Delta l)^2}\left[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1}\right] = -\rho_{i,j}$$

Principle: Note that this mathematical operation is actually a CONVOLUTION:

Recall on 1D convolution:

$$(f * g)(t) = \int f(\tau)g(t-\tau)\,d\tau$$

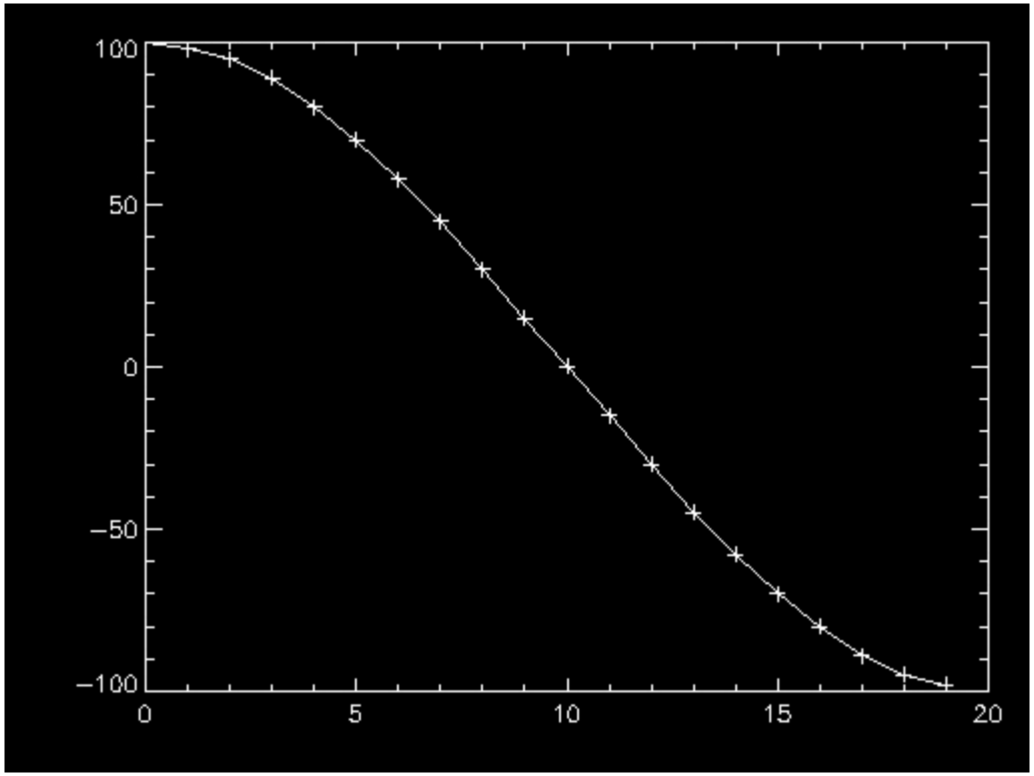
Discreet convolution : « value of f * g at point m »

$$(f * g)(m) = \sum_n f(n)g(m-n)$$

Example:
Either f a series of values (one vector), here 20 values

F(m)=[100, 98,95,89,80,70, 58,45,30,15, 0, -15, -30,-45,-58 ,-70,-80,-89,-95,-98]

f



m

Convolutions
This function
With the function
"hat"

F(x)=1/2 si -1<x<1
F(x)=0 otherwise

=> 3 points smoothing

Let’s do convolution with kernel **G=[1,1,1] /3**

since

$$(f * g)(m) = \sum_n f(n)g(m-n)$$

Let C=f*g , then each point of C is replaced by the average of the 3 neighboring points of f

F(m)=[100, 98,95,89,80,70, 58,45,30,15, 0, -15, -30,-45,-58 ,-70,-80,-89,-95,-98

[1/3,1/3,1/3]

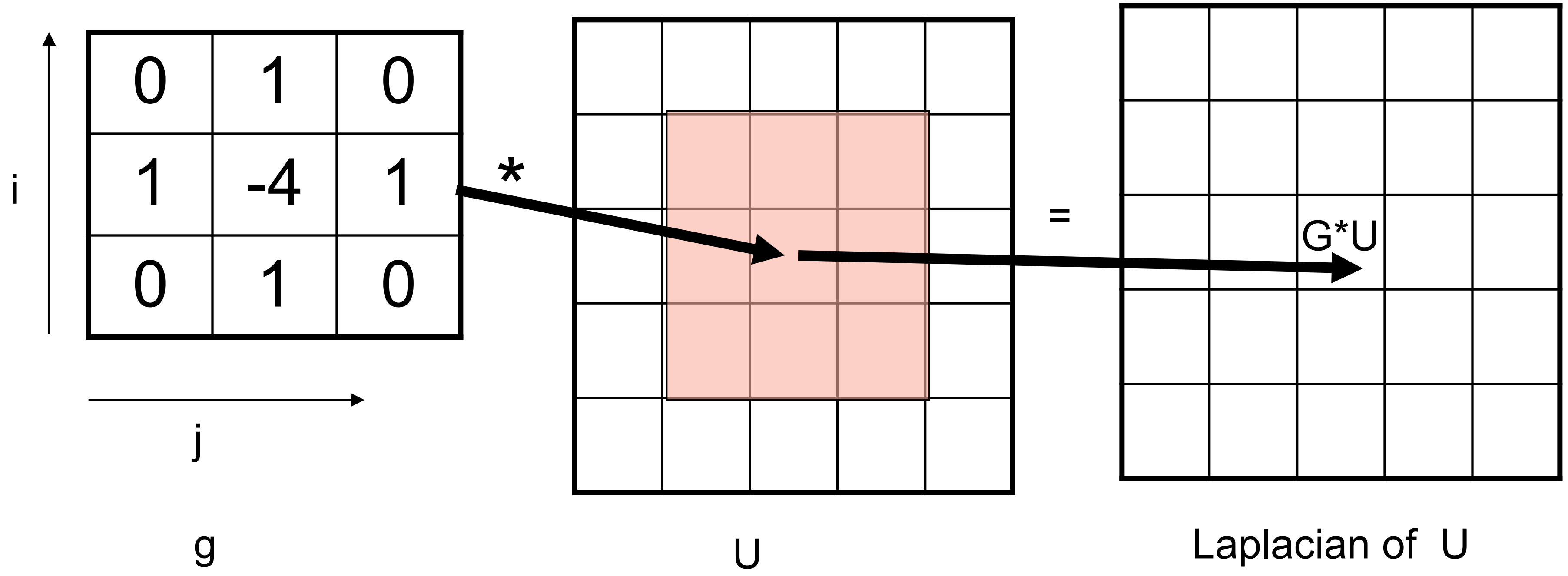
[0.,97.6,94.0,88.0, 79.6, 69.3, 57.6, 44.3, 30.0,15.0, 0.0, 15.0, -30.0, -44.3, -57.6, -69.3, -79.6,…]

Convolution can be seen as a kind of »moving" window acting on the function itself. G is called the "convolution kernel"

2D convolution:

We shift the « window » with the convolution kernel :

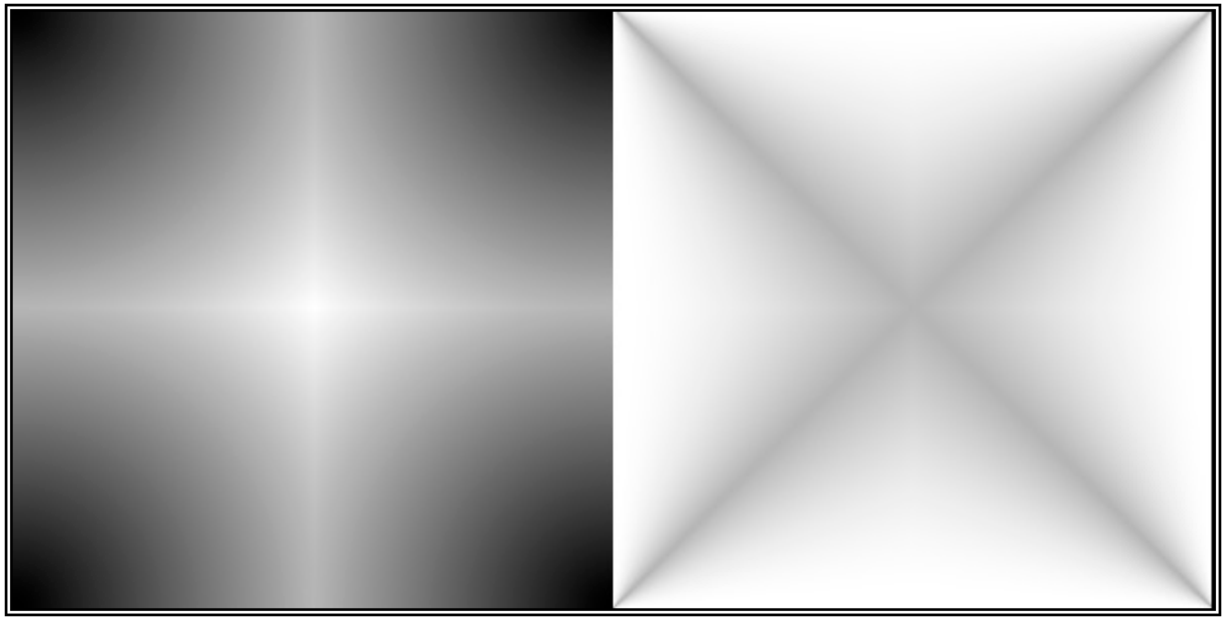
$$\frac{1}{(\Delta l)^2}\underbrace{[u_{i+1,j} - 2u_{i,j} + u_{i-1,j} + u_{i,j+1} - 2u_{i,j} + u_{i,j-1}]}_{\text{Laplacian}} = -\rho_{i,j}$$



Ex :

U(i,j)= 1/i+1/j

Laplacian of U after applying convolution to every point



So the problem can be rewritten

$$\nabla^2 f = (U * lap)(x, y) = -\rho(x, y)$$

Laplacian

Why does it help us???

Well **thanks to this famous theorem linking <u>Fourrier Transform</u> and <u>convolution</u>**

$$\mathcal{F}(f * g) = \sqrt{2\pi}\mathcal{F}(f) \cdot \mathcal{F}(g)$$

« The Fourrier tranform of a convolution product is equal to prodcut Fourrier series » (to a multiplicative constant...)

So our problem is rewritten:

let U*= TF(U), Lap*=TF(lap) , ρ*=TF (ρ)

where TF means « fourrier transform » «

U* x Lap*=ρ* =>

U^*= ρ*/Lap* =>

☺

$$U = TF^{-1}(\rho^*/Lap^*)$$

Advantage : FFT is super radid (Nxlog(N))
INCONVENIENT : limited to periodic boundary conditions!!

This kind of method is the basis of so-called spectral methods...

But they are actually very sensitive to numerical noise and require to be used with care

Exemple : « PARTICLE MESH METHOD», inspired on previous results. Special case of electric charges

$$Laplacian(U) = \rho_{i,j} \; is \; equavalent \; to \; U = \sum_{i,j} \frac{\rho}{r}$$

The equation of the potential of electric charges:

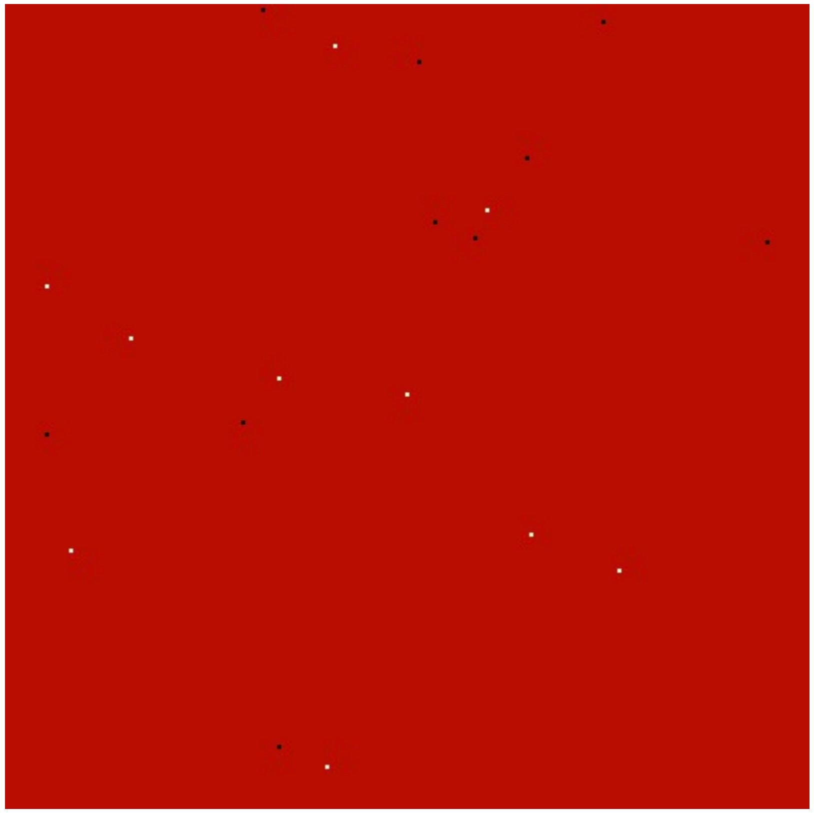
$$U(r) = \sum_i \frac{q_i}{\left\| r - r_i \right\|} = \left(\rho * p\right) \longleftarrow \text{Convolution}$$

where $\rho(r)$= charge at position r
where P(r)=1/R

So we've got $U^* = \rho^* \times P^* \Rightarrow U = TF^{-1} (\rho^* \times P^*)$

EXEMPLE :

Calculate the potential field generated by 10 charges - and 10 charges - placed
In a grid 200x200 (40,000 mesh points)

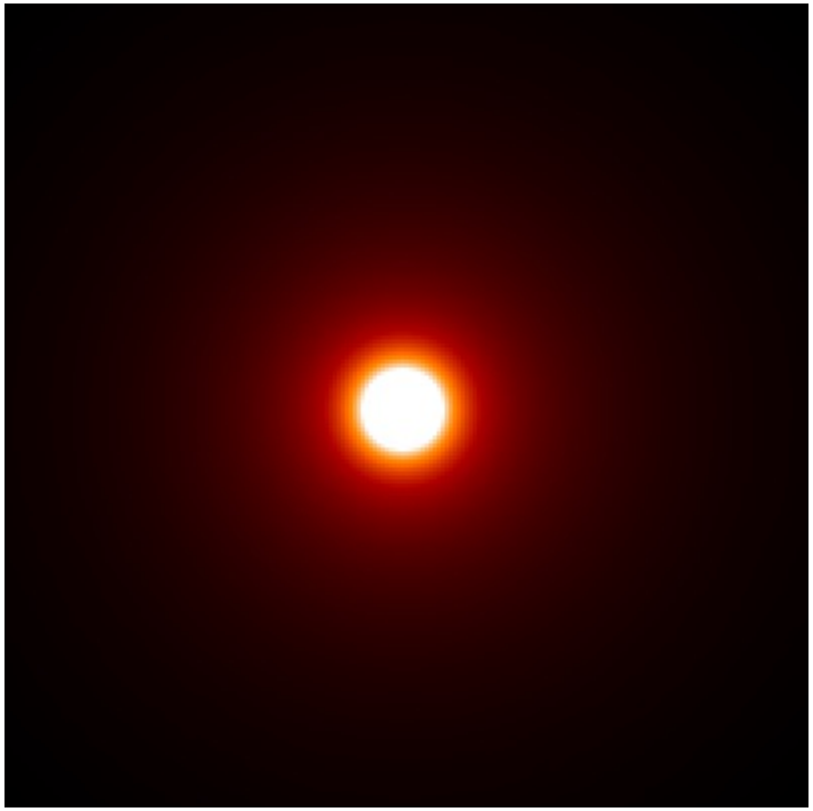


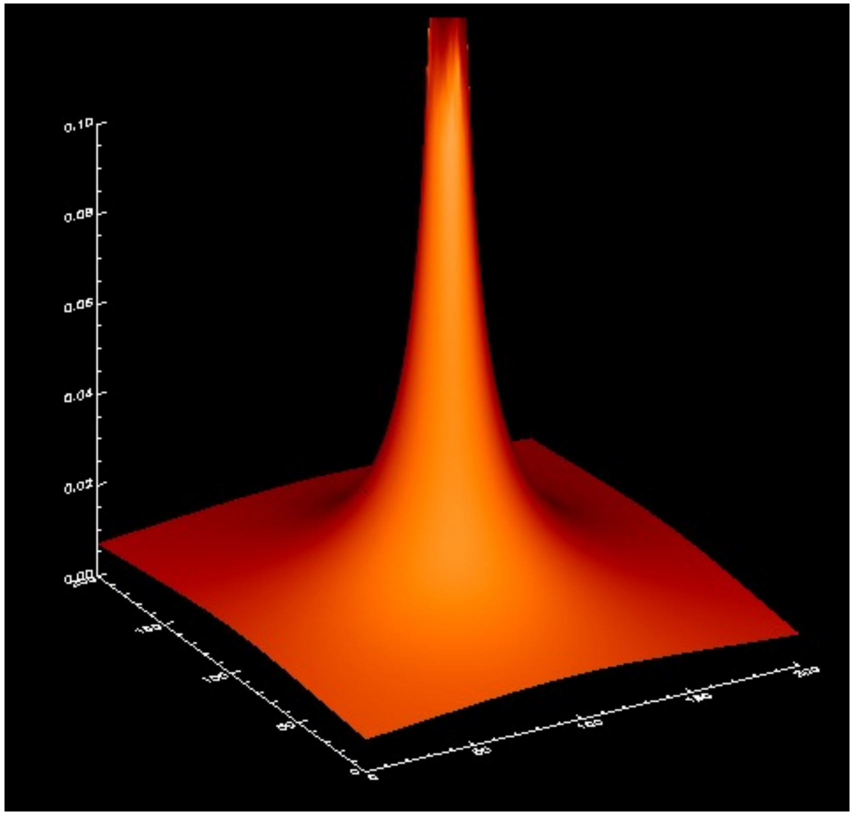
« White » = +1

« Black » =-1

$\rho(r)$

Convolution kernel: 1/R

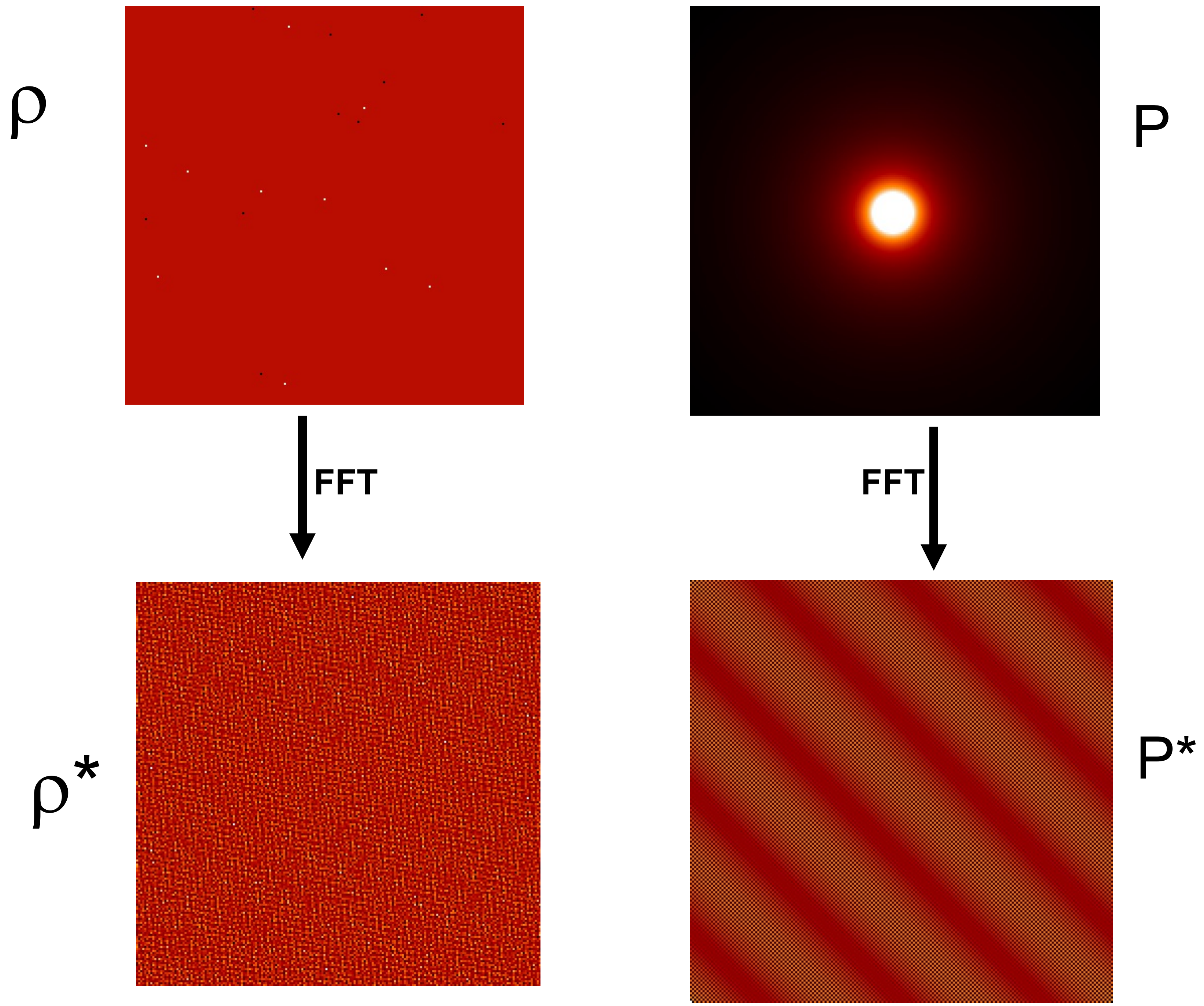


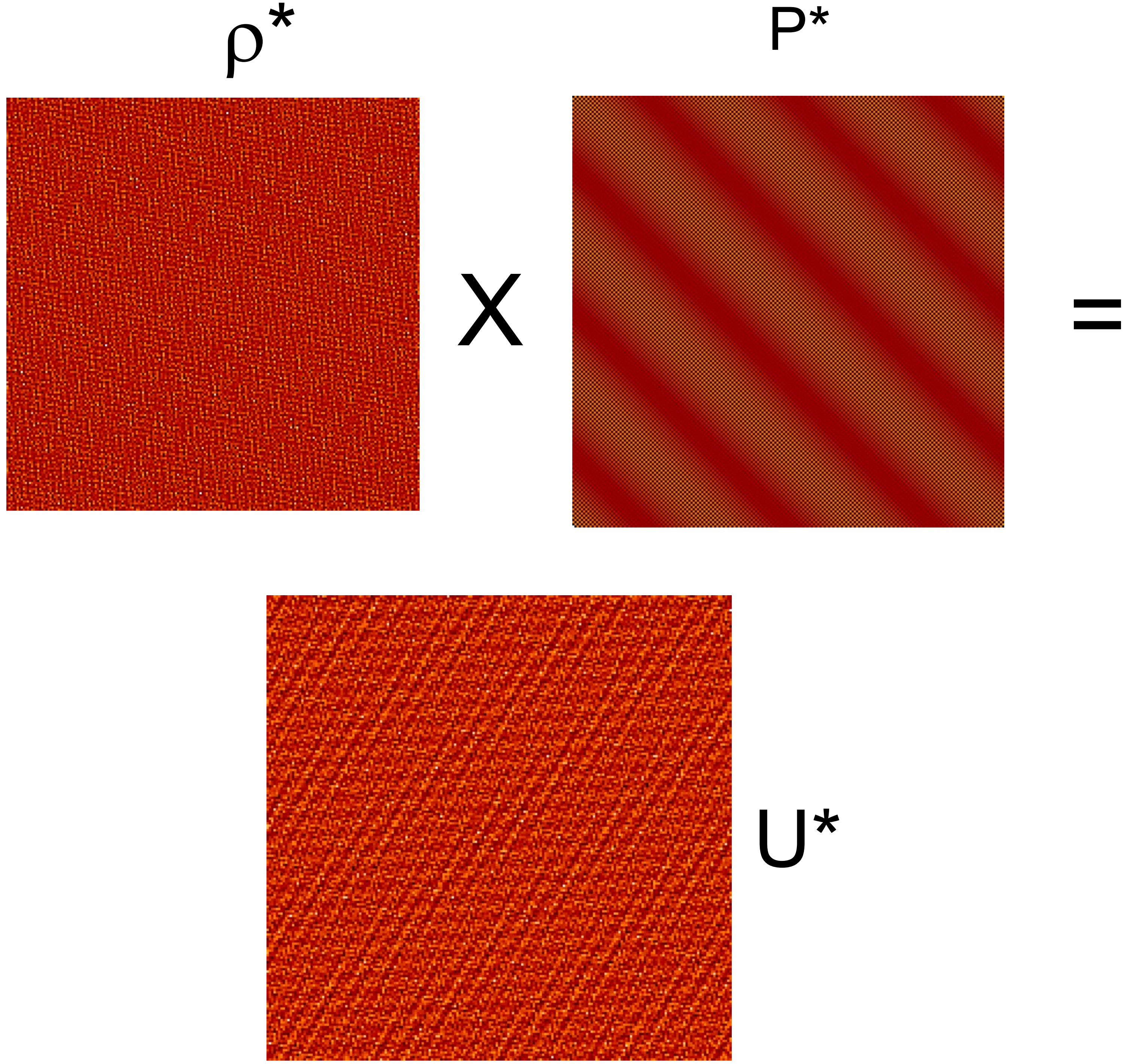


P(r)

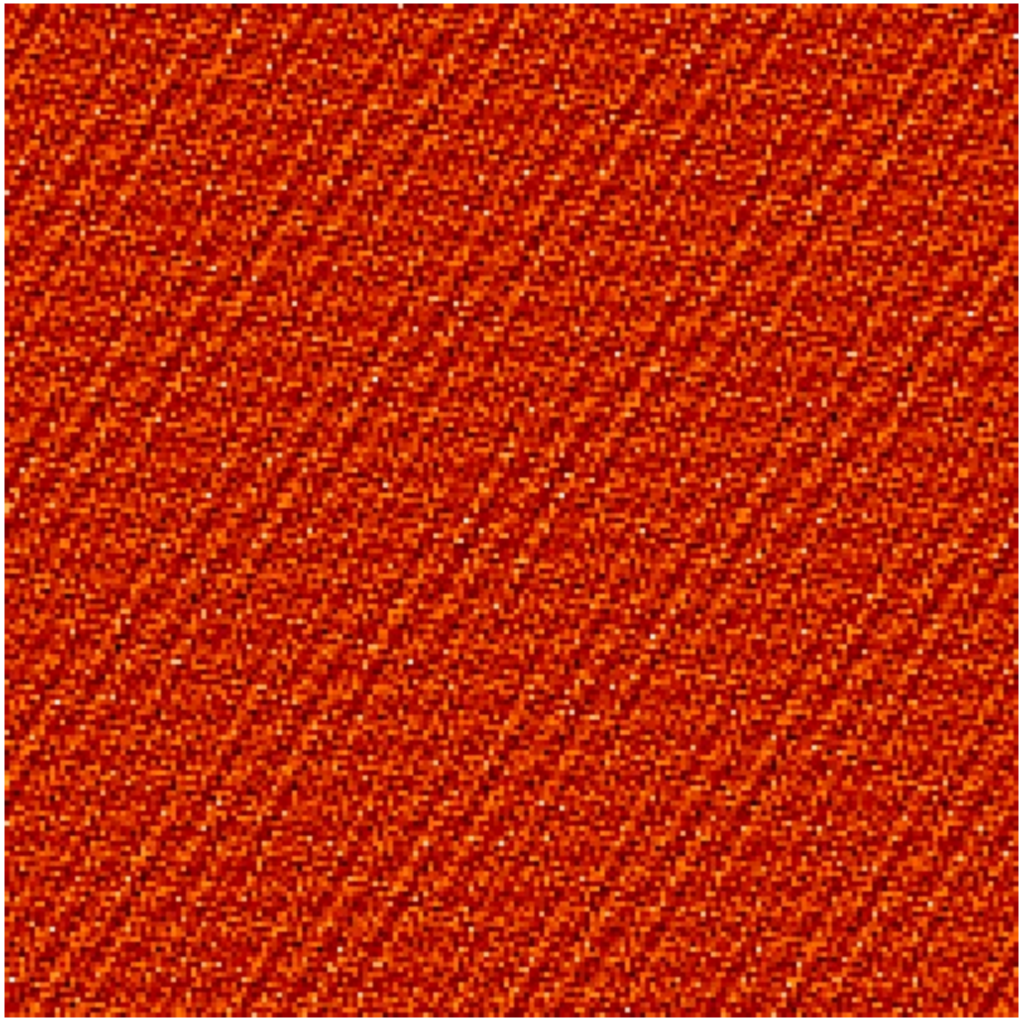
Color coded

3D display of P(x,y)


ρ
P
FFT
FFT
ρ*
P*

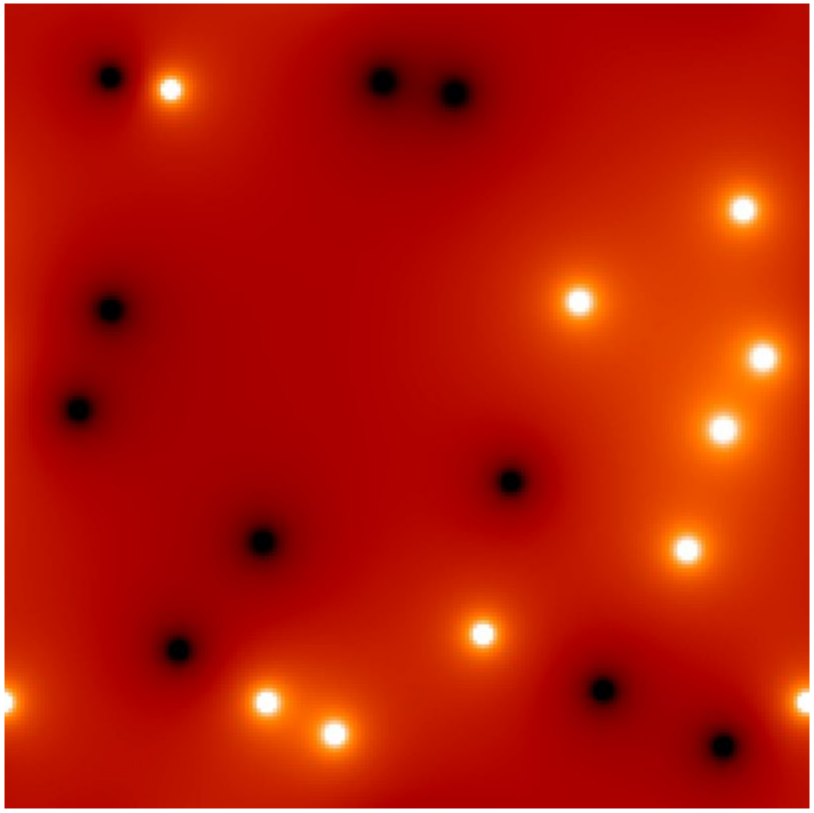

ρ*
X
P*
=
U*

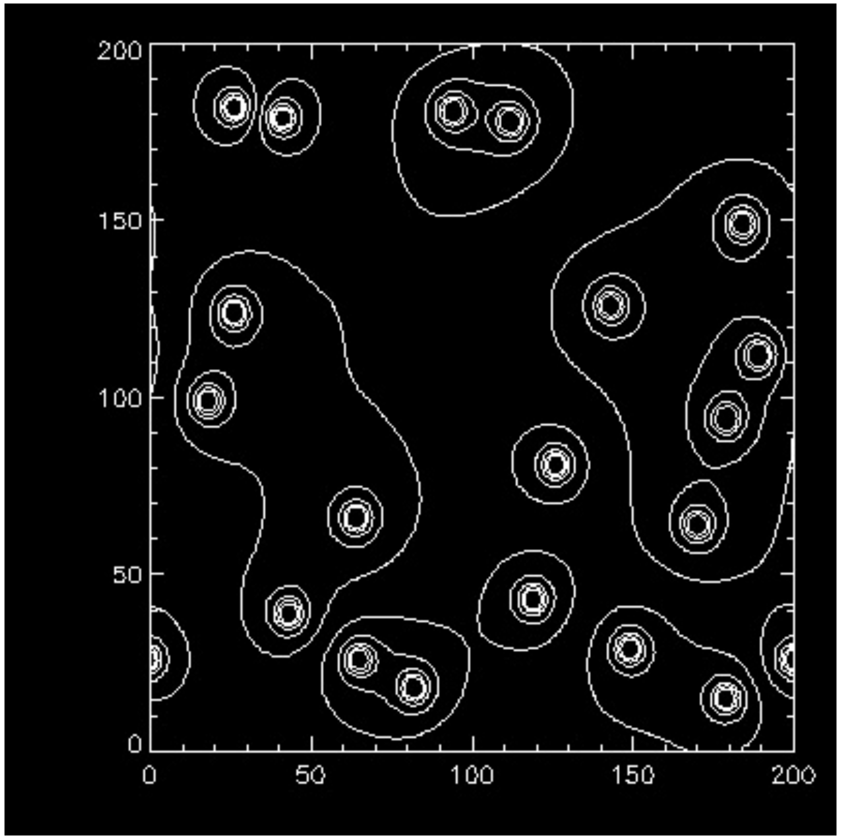
U* : Fourrier transform of field U



To find U, we take the reverse FT of U, and we find ...

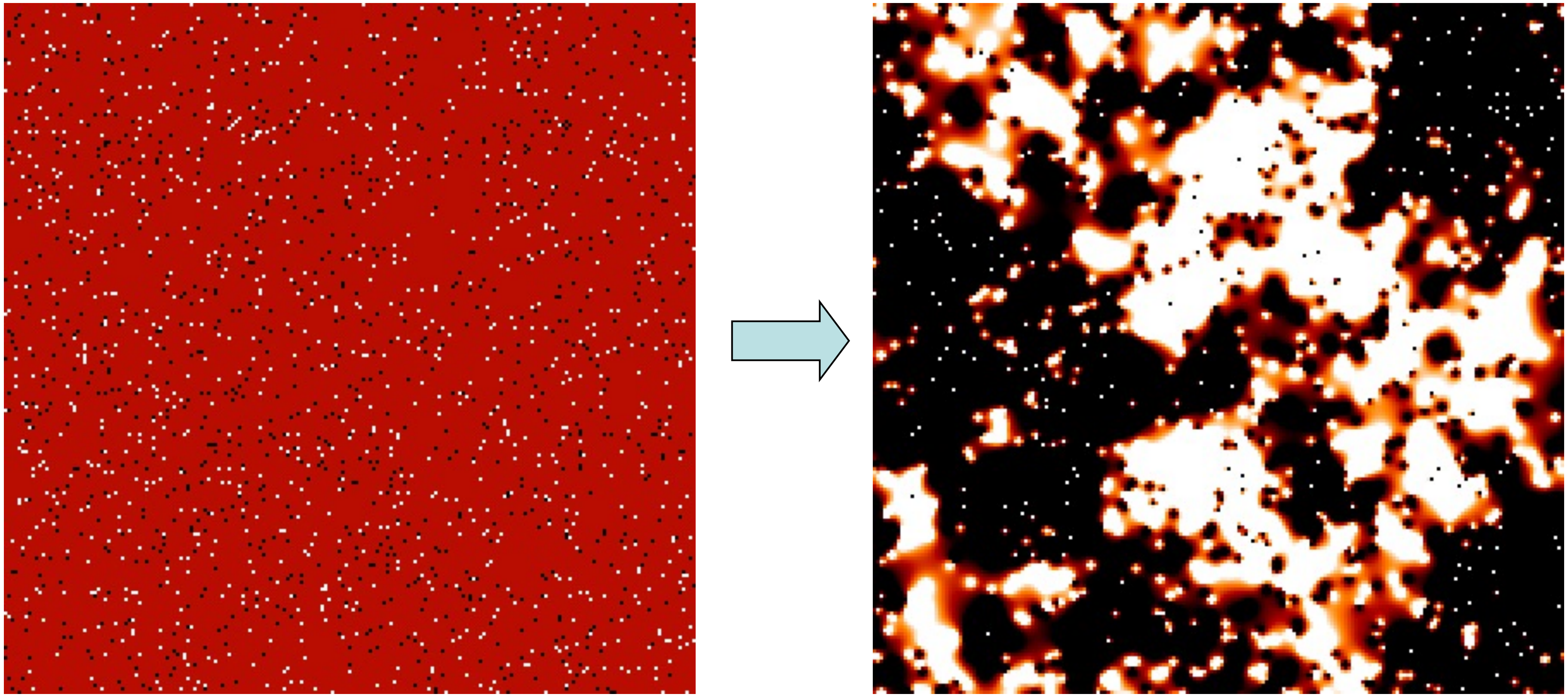
U Potential field





Notice the periodic conditions

The same with 4000 charges (it's no longer)



CONCLUSION on elliptical equations

Elliptical functions (which require calculating a field on a grid) require a rather complex resolution because the equation that allows calculating them is not linear and is not simply written with dies.

They are called "equations with boundary conditions"

Spectral approaches, which pass through a TF, are often the fastest (thanks to FFT algorithme) .

Boundary conditions can sometimes be difficult to handle.
Of course the FFT assumes periodic term conditions

For isolated non-periodic systems ... this still requires a little work.