Courant and all that

$$\mathbf{c}\frac{\mathbf{dt}}{\mathbf{dx}} \le \varepsilon \approx 1$$

- Consistency, Convergence
- Stability
- Numerical Dispersion
- Computational grids and numerical anisotropy

The goal of this lecture is to understand how to find suitable parameters (i.e., grid spacing dx and time increment dt) for a numerical simulation and knowing what can go wrong.

A simple example: Newtonian Cooling

Numerical solution to first order ordinary differential equation

$$\frac{dT}{dt} = f(T,t)$$

We can not simply integrate this equation. We have to solve it numerically! First we need to discretise time:

$$t_j = t_0 + jdt$$

and for Temperature T

$$T_j = T(t_j)$$

A finite-difference approximation

Let us try a forward difference:

$$\left.\frac{dT}{dt}\right|_{t=t_j} = \frac{T_{j+1} - T_j}{dt} + O(dt)$$

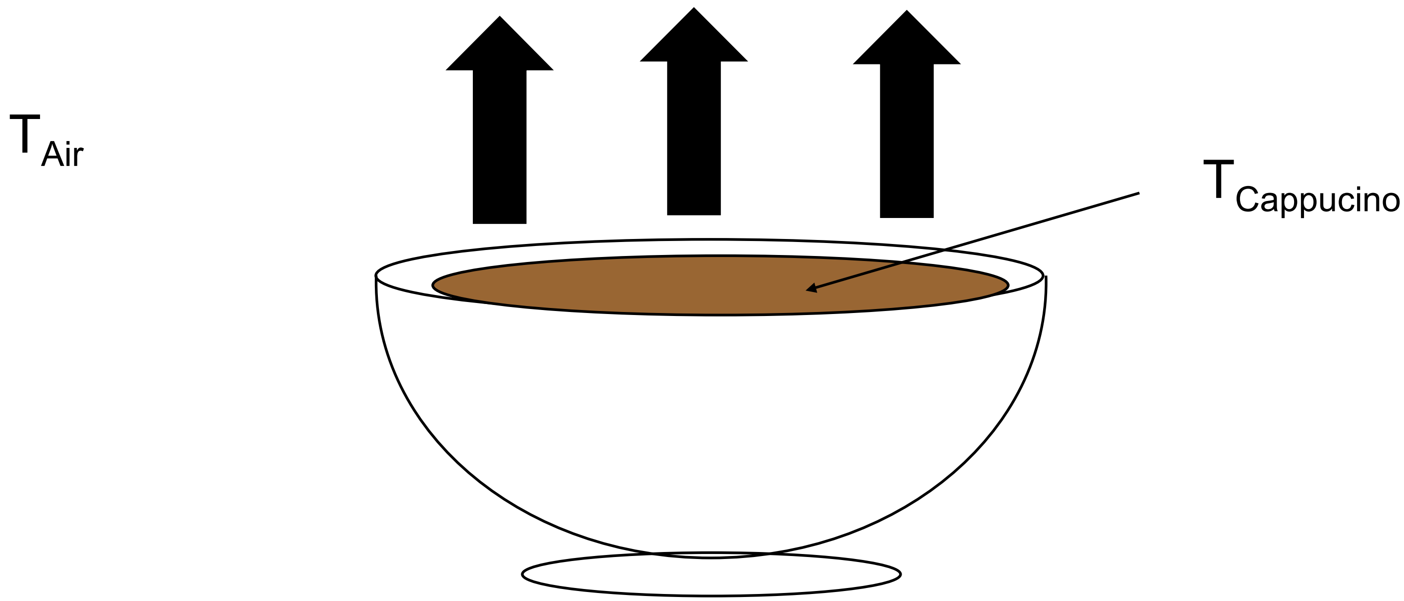
... which leads to the following explicit scheme :

$$T_{j+1} \approx T_j + \mathrm{dt} f(T_j, t_j)$$

This allows us to calculate the Temperature T as a function of time and the *forcing* inhomogeneity f(T,t). Note that there will be an error O(dt) which will accumulate over time.

Coffee?

Let's try to apply this to the Newtonian cooling problem:



How does the temperature of the liquid evolve as a function of time and temperature difference to the air?

Newtonian Cooling

The rate of cooling (dT/dt) will depend on the temperature difference (T_{cap}-T_{air}) and some constant (thermal conductivity). This is called **Newtonian Cooling**.

With $T = T_{cap} - T_{air}$ being the temperature difference and τ the time scale of cooling then $f(T,t) = -T/\tau$ and the differential equation describing the system is

$$\boxed{\frac{dT}{dt} = -T/\tau}$$

with initial condition $T=T_i$ at $t=0$ and $\tau>0$.

Analytical solution

This equation has a simple analytical solution:

$$T(t) = T_i \exp(-t/\tau)$$

How good is our finite-difference appoximation?
For what choices of dt will we obtain a stable solution?

Our FD approximation is:

$$T_{j+1} = T_j - \frac{dt}{\tau} T_j = T_j (1 - \frac{dt}{\tau})$$

$$\boxed{T_{j+1} = T_j (1 - \frac{dt}{\tau})}$$

FD Algorithm

$$\boxed{T_{j+1} = T_j(1 - \frac{dt}{\tau})}$$

1. Does this equation approximation converge for dt -> 0?
2. Does it behave like the analytical solution?

With the initial condition $T=T_0$ at $t=0$:

$$T_1 = T_0(1 - \frac{dt}{\tau})$$

$$T_2 = T_1(1 - \frac{dt}{\tau}) = T_0(1 - \frac{dt}{\tau})(1 - \frac{dt}{\tau})$$

leading to : $$\boxed{T_j = T_0(1 - \frac{dt}{\tau})^j}$$

Understanding the numerical approximation

$$\boxed{T_j = T_0 (1 - \frac{dt}{\tau})^j}$$

Let us use $dt = t_j / j$ where t_j is the total time up to time step j:

$$T_j = T_0 \left(1 + \left[-\frac{t}{j\tau} \right] \right)^j$$

This can be expanded using the *binomial theorem*

$$T_j = T_0 \left[1^j + 1^{j-1} \left[-\frac{t}{j\tau} \right] \binom{j}{1} + 1^{j-2} \left[-\frac{t}{j\tau} \right]^2 \binom{j}{2} + \ldots \right]$$

Courant and all that

... where

$$\binom{j}{r} = \frac{j!}{(j-r)!r!}$$

we are interested in the case that dt-> 0 which is equivalent to j->∞

$$\frac{j!}{(j-r)!} = j(j-1)(j-2)...(j-r+1) \rightarrow j^r$$

as a result

$$\binom{j}{r} \rightarrow \frac{j^r}{r!}$$

substituted into the series for T_j we obtain:

$$T_j \rightarrow T_0\left[1+\frac{j}{1!}\left[-\frac{t}{j\tau}\right]+\frac{j^2}{2!}\left[-\frac{t}{j\tau}\right]^2+\ldots\right]$$

which leads to

$$T_j \rightarrow T_0\left[1+\left[-\frac{t}{\tau}\right]+\frac{1}{2!}\left[-\frac{t}{\tau}\right]^2+\ldots\right]$$

... which is the Taylor expansion for

$$T_j = T_0 \exp(-t/\tau)$$

So we conclude:

For the Newtonian Cooling problem, the numerical solution converges to the exact solution when the time step dt gets smaller.

How does the numerical solution behave?

$$T_j = T_0 \exp(-t/\tau)$$

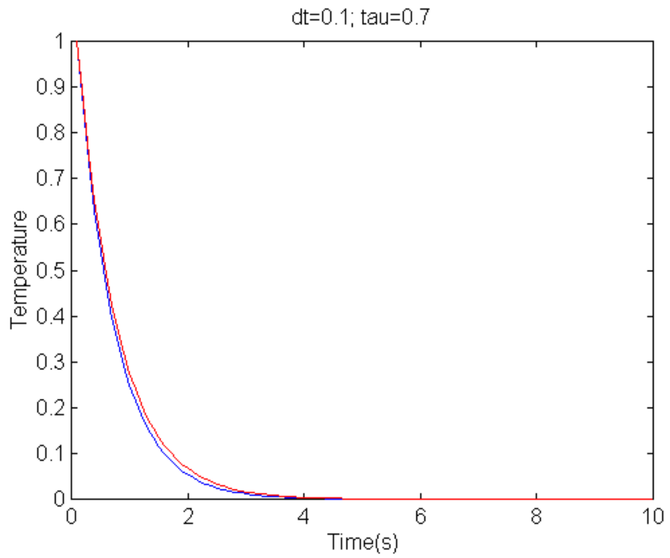
The analytical solution decays monotonically!

$$T_{j+1} = T_j (1 - \frac{dt}{\tau})$$

What are the conditions so that $T_{j+1} < T_j$?

Convergence

Convergence means that if we decrease time and/or space increment in our numerical algorithm we get closer to the true solution.



Stability

$$T_{j+1} = T_j (1 - \frac{dt}{\tau})$$

$T_{j+1} < T_j$ requires

$$0 \le 1 - \frac{dt}{\tau} < 1$$

or

$$\boxed{0 \le dt < \tau}$$

The numerical solution decays only montonically for a limited range of values for dt! Again we seem to have a *conditional stability*.

$$\boxed{T_{j+1} = T_j\left(1 - \frac{dt}{\tau}\right)}$$

if $\tau < dt < 2\tau$ then $\left(1 - \dfrac{dt}{\tau}\right) < 0$

$\Longrightarrow$ the solution oscillates but converges as |1-dt/τ|<1

if $dt > 2\tau$ then $dt/\tau > 2$

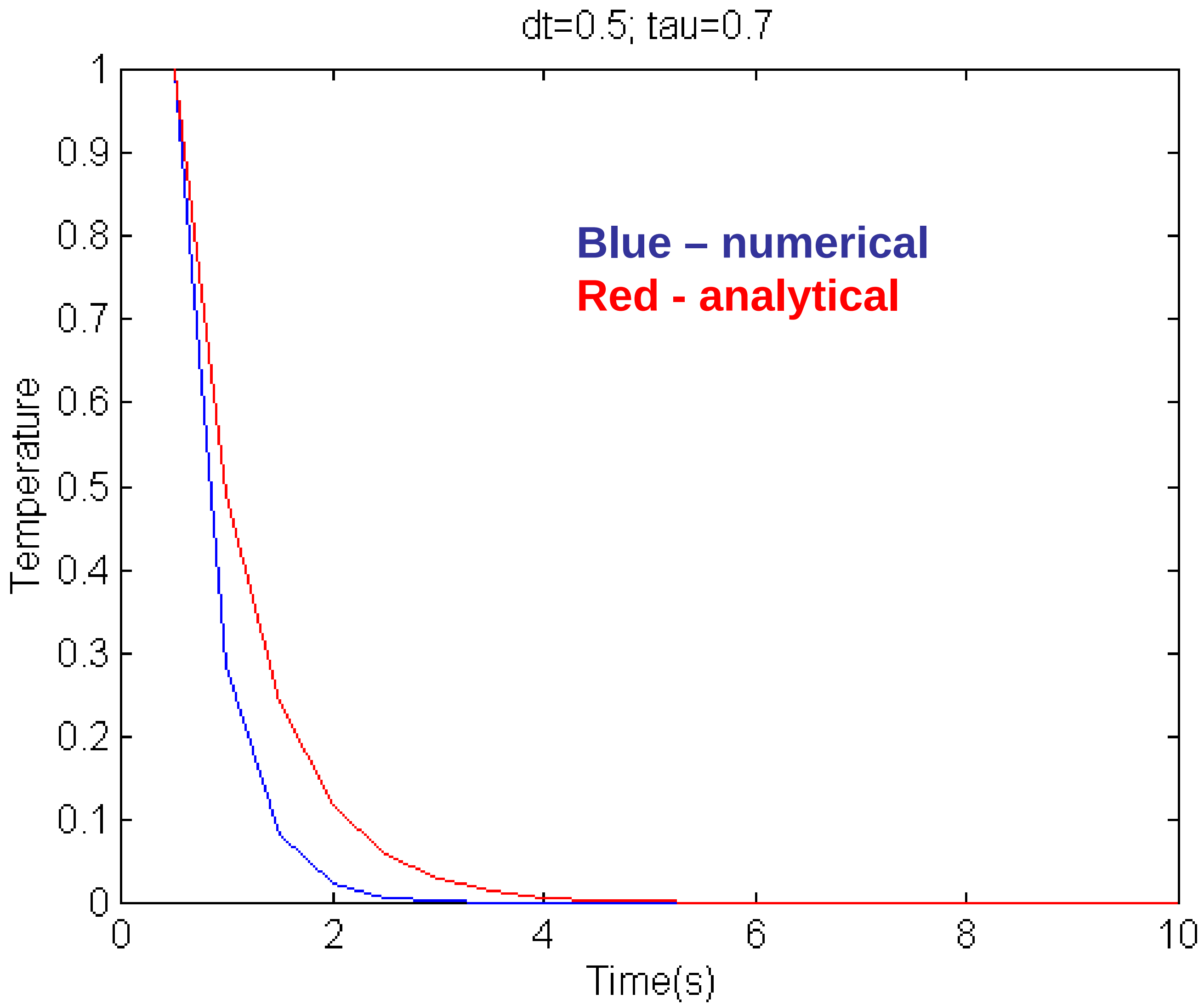
$\Longrightarrow$ 1-dt/τ<-1 and the solution oscillates and diverges

... now let us see how the solution looks like ....

Matlab solution

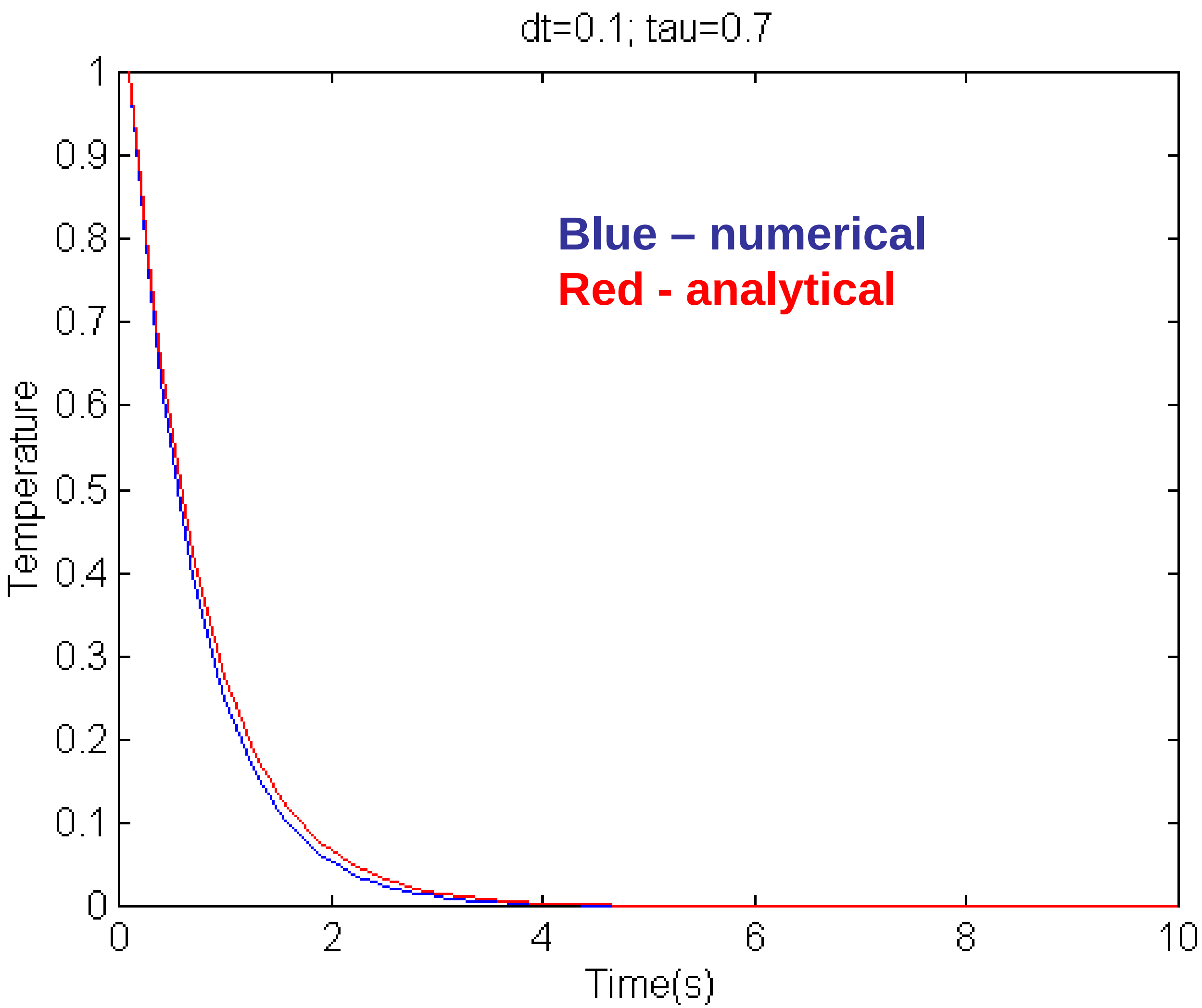
```
% Matlab Program - Newtonian Cooling

% initialise values
nt=10;
t0=1.
tau=.7;
dt=1.

% initial condition
T=t0;

% time extrapolation
for i=1:nt,
T(i+1)=T(i)-dt/tau*T(i);
end

% plotting
plot(T)
```

Example 1

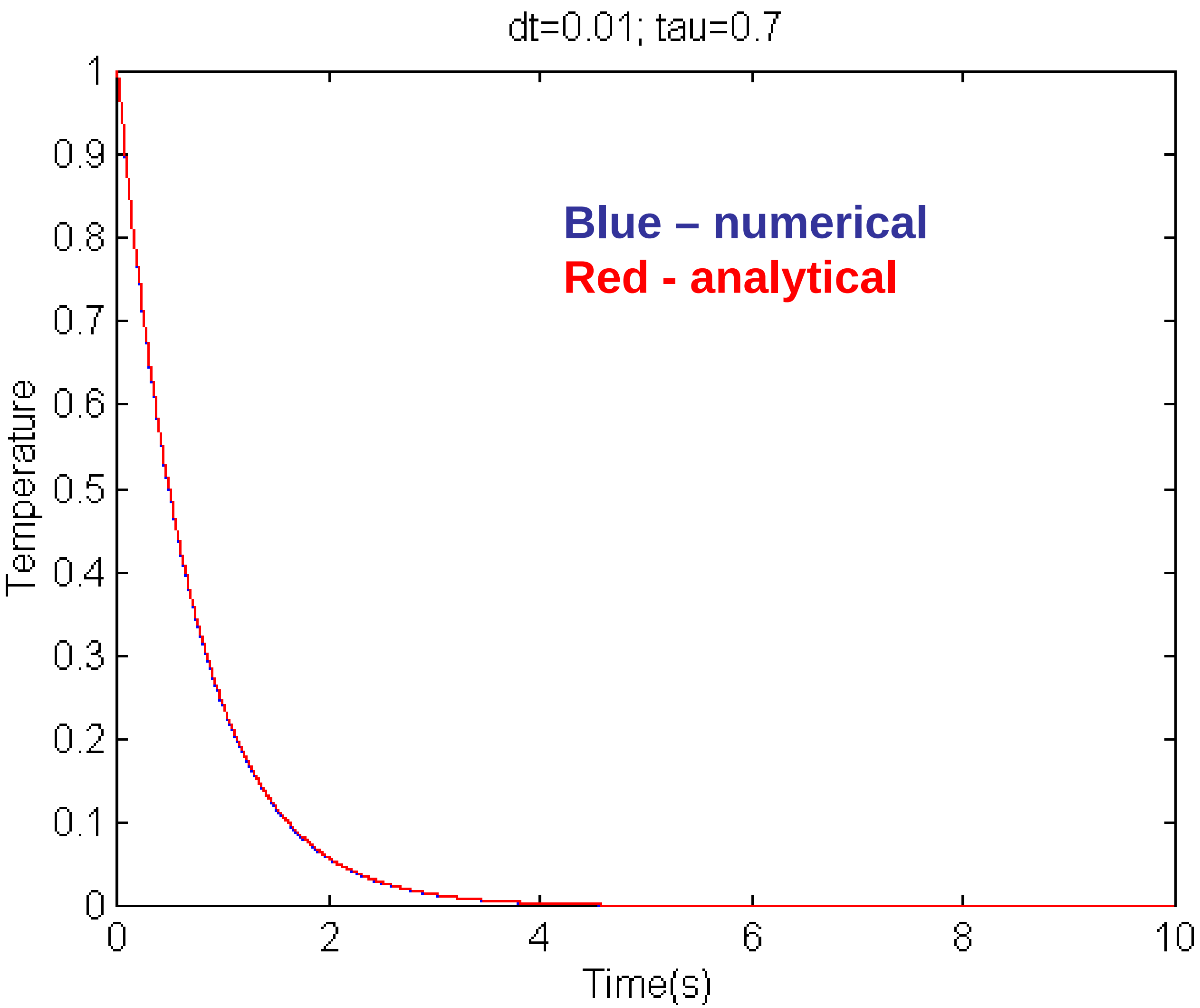


dt=0.5; tau=0.7
Blue – numerical
Red - analytical
Temperature
Time(s)
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
2
4
6
8
10

Example 2

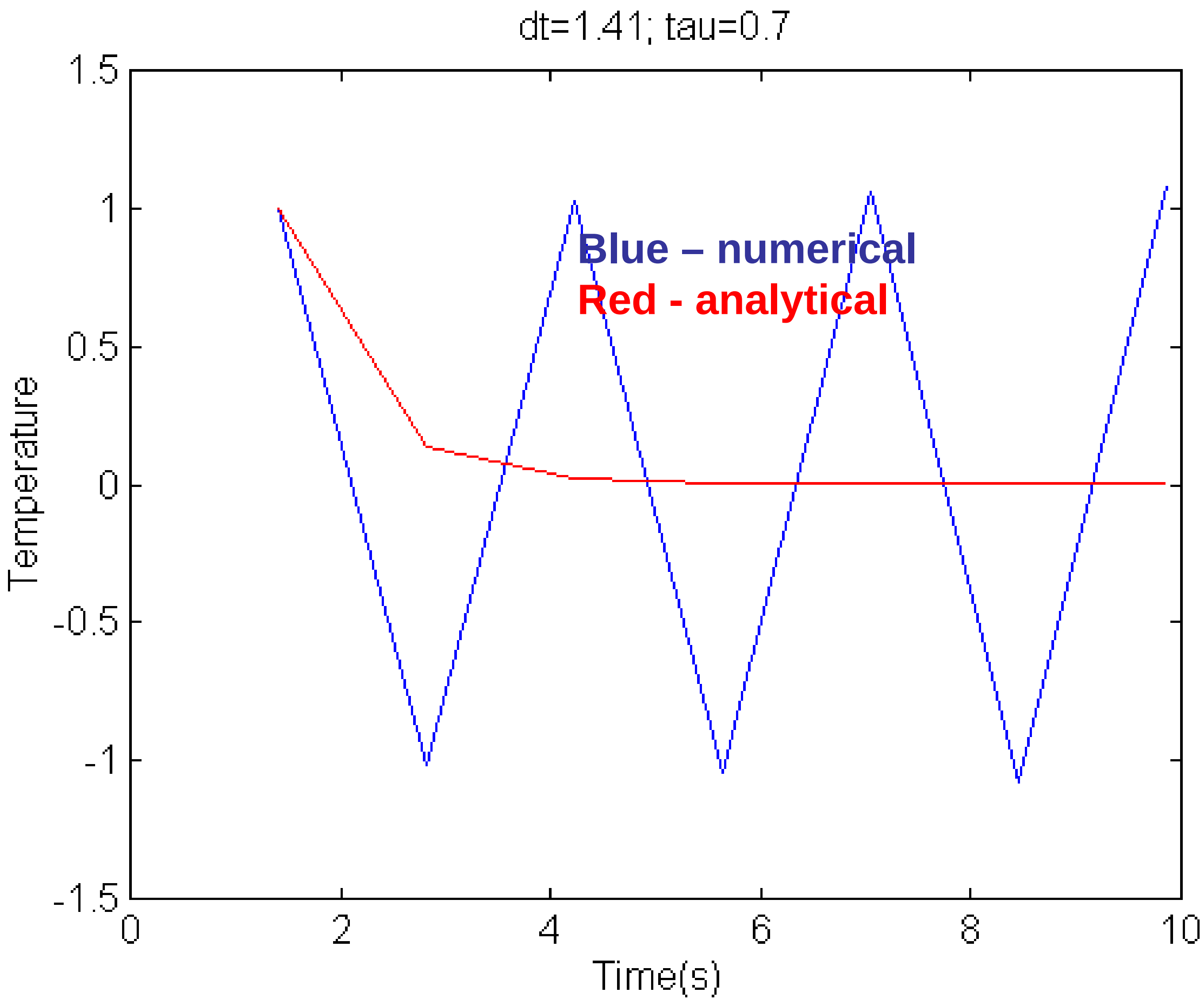

dt=0.1; tau=0.7
Blue – numerical
Red - analytical
Temperature
Time(s)
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
2
4
6
8
10

Example 3



dt=0.01; tau=0.7
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Temperature
0
2
4
6
8
10
Time(s)
Blue – numerical
Red - analytical

Example 4

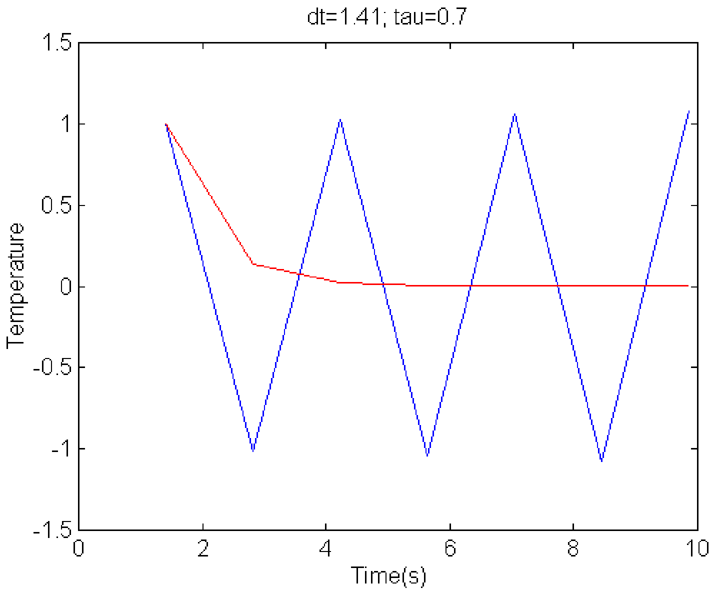


dt=1.41; tau=0.7
Blue – numerical
Red - analytical
Temperature
Time(s)
1.5
1
0.5
0
-0.5
-1
-1.5
0
2
4
6
8
10

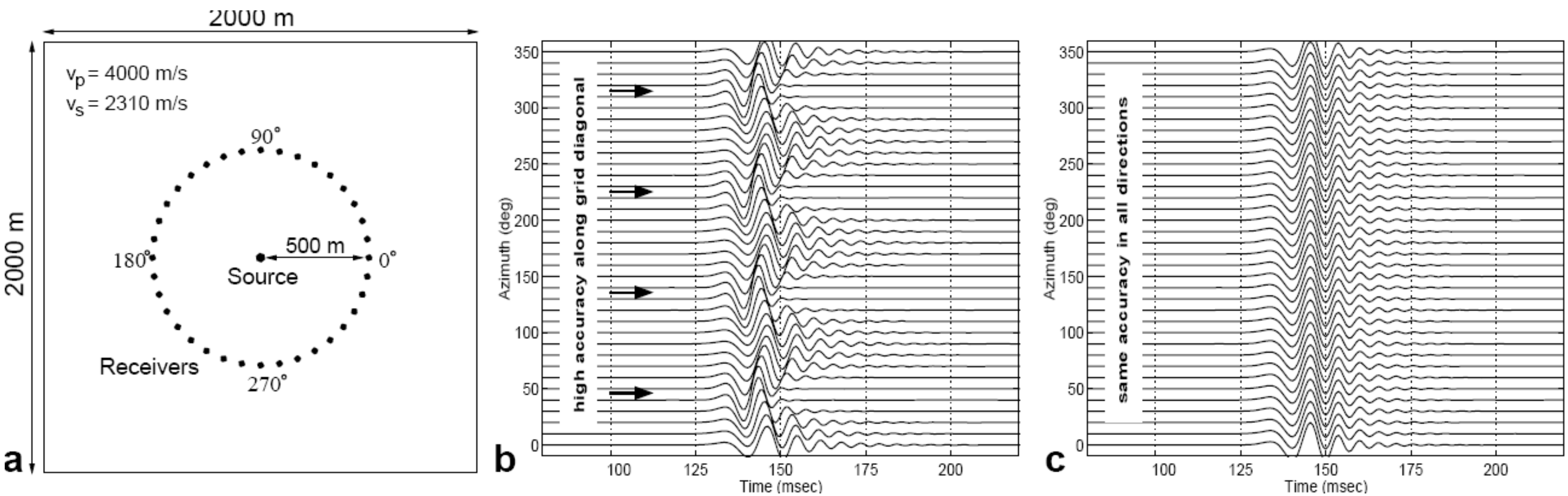
Stability

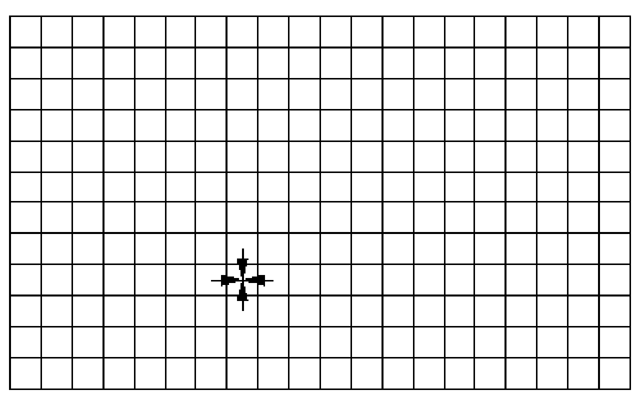
Stability of a numerical algorithm means that the numerical solution does not tend to infinite (or zero) while the true solution is finite.

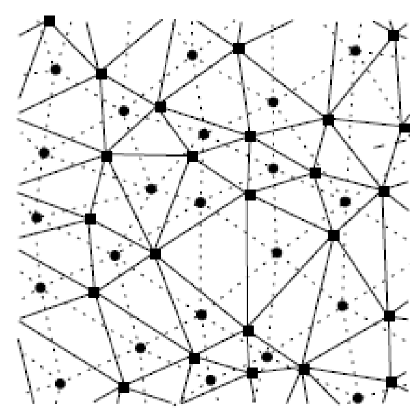
In many case we obtain algoriths with conditional stability. That means that the solution is stable for values in well-defined intervals.



Grid anisotropy

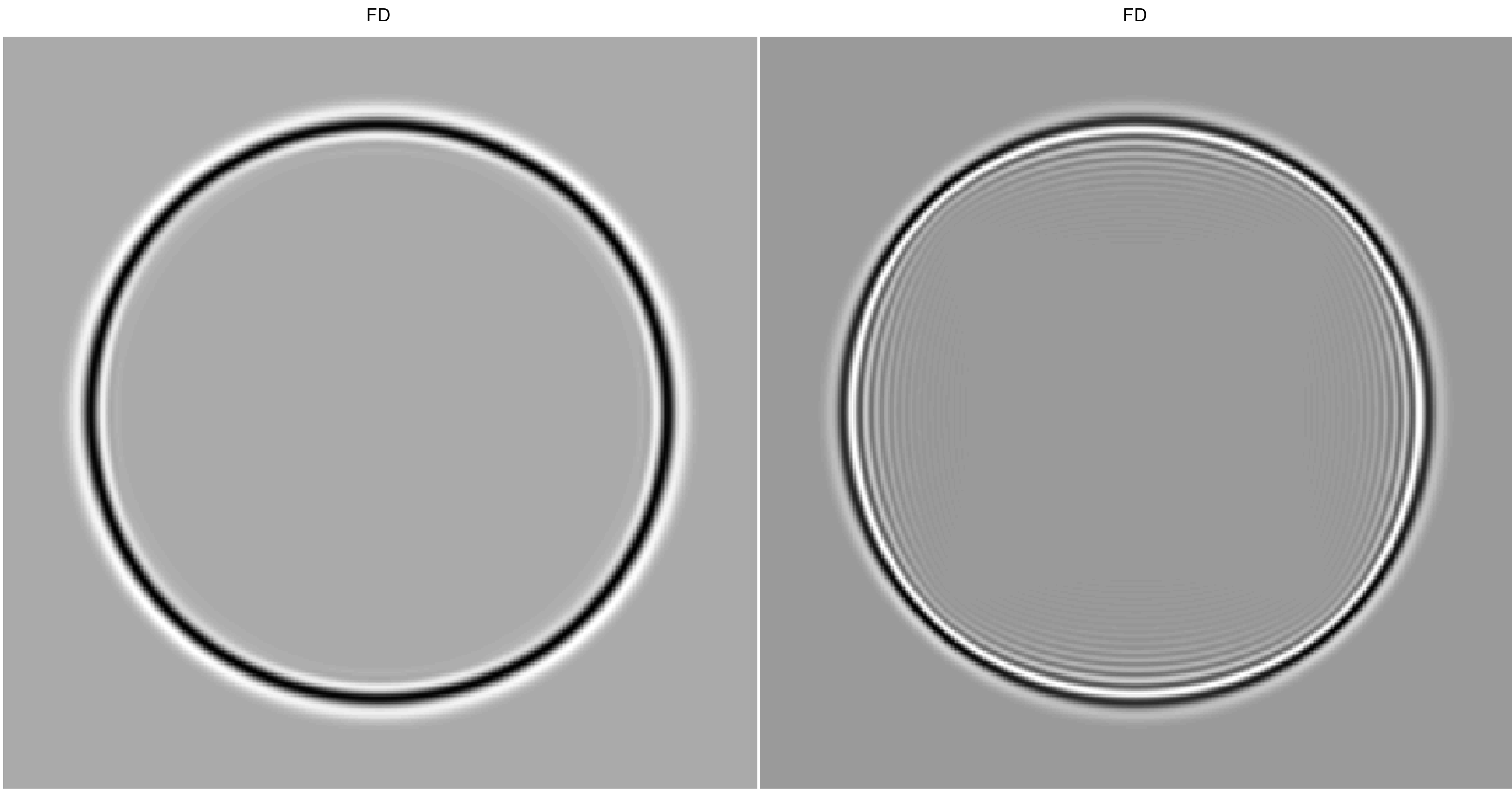




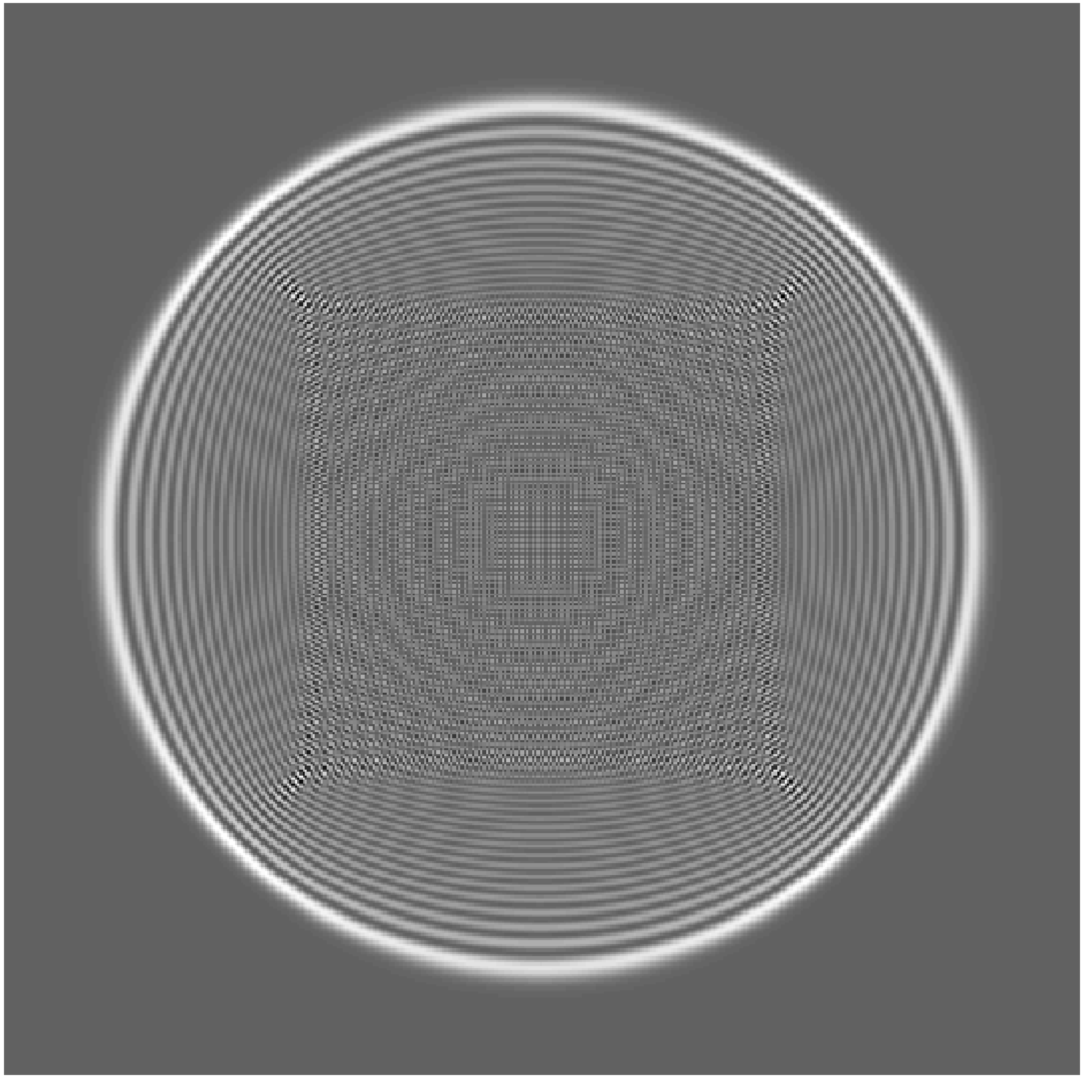


In which direction is the solution most accurate and why?

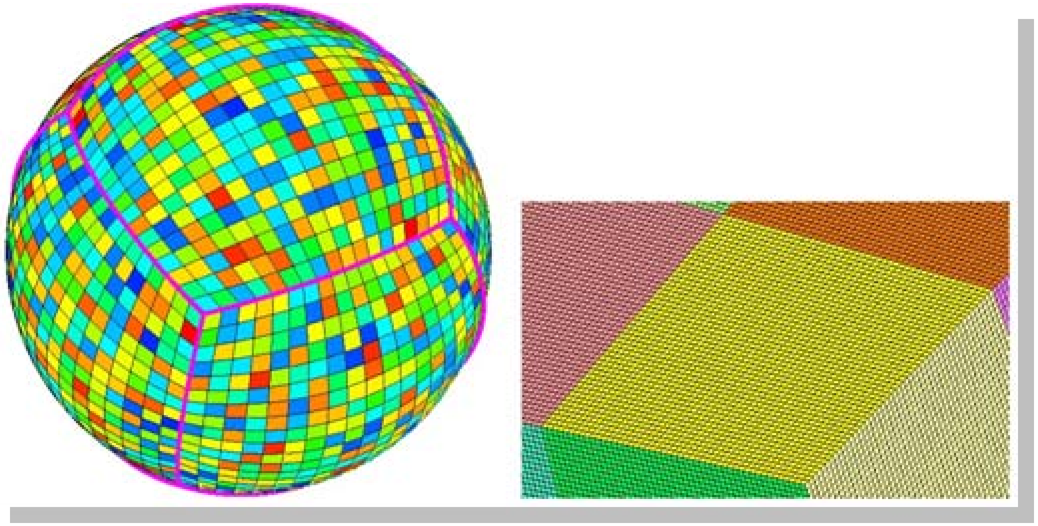
Grid anisotropy with ac2d.m

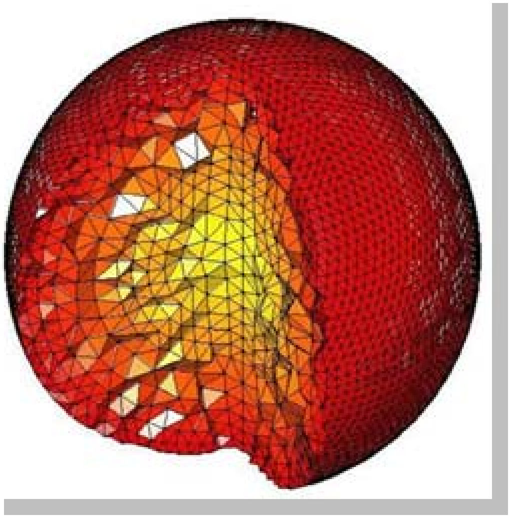


… and more … a Green's function …



Grids

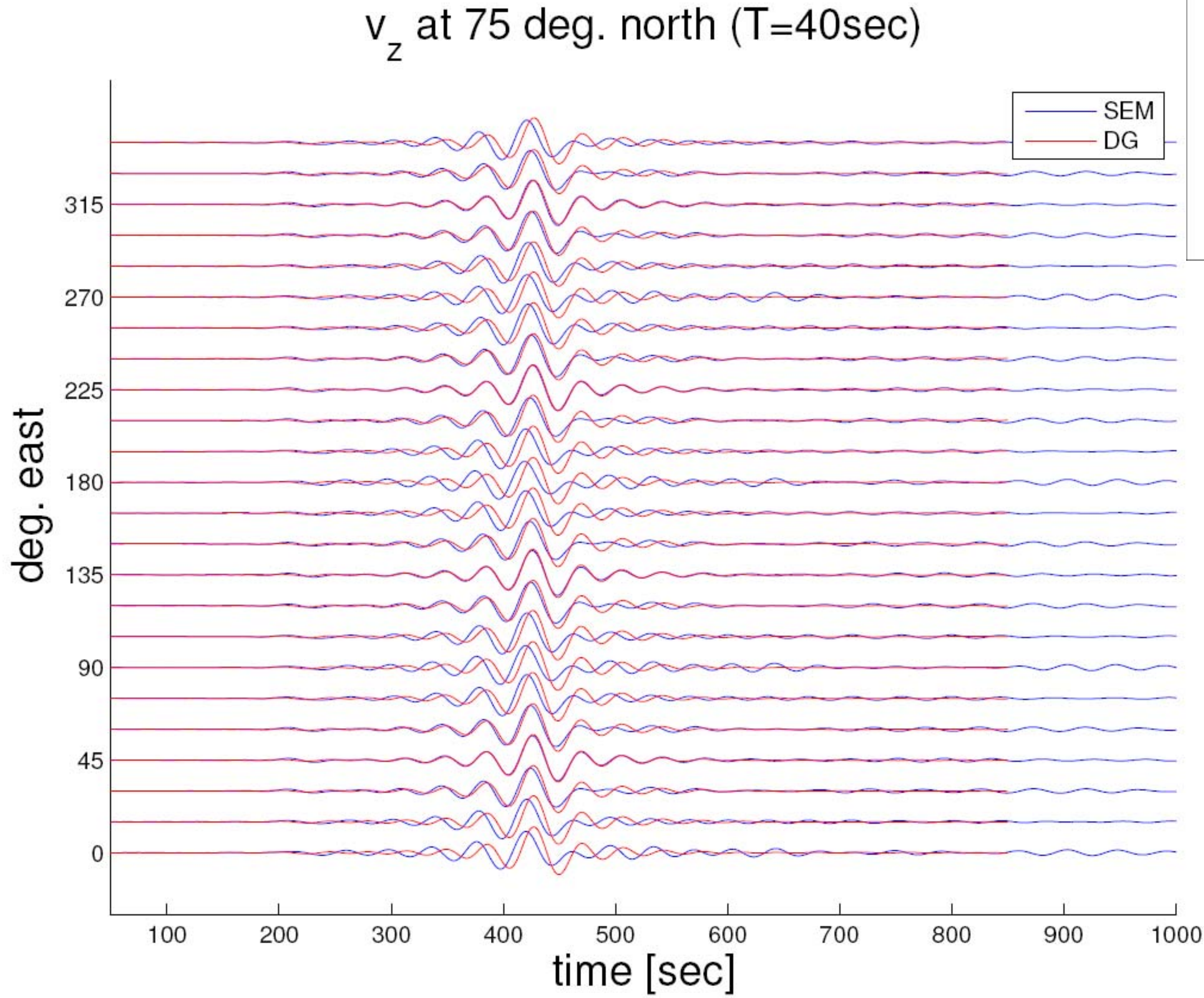


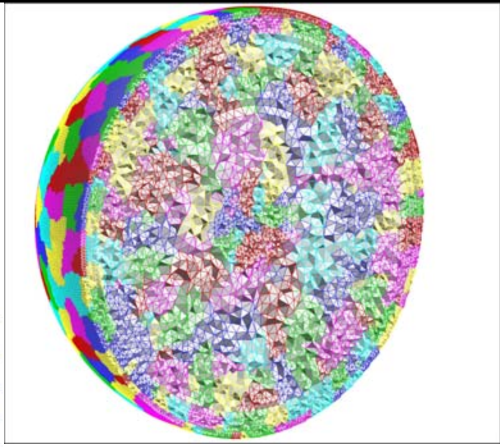


Cubed sphere
Hexahedral grid
Spectral element implementation

Tetrahedral grid
Discontinuous Galerkin method

Grid anisotropy





Courant and all that

Grid anisotropy

Grid anisotropy is the directional dependence of the accuracy of your numerical solution if you do not use enough points per wavelength.

Grid anisotropy depends on the actual grid you are using. Cubic grids display anisotropy, hexagonal grids in 2D do not. In 3D there are no grid with isotropic properties!

Numerical solutions on unstructured grids usually do not have this problem because of averaging effects, but they need more points per wavelength!

Real problems

Alluvial Basin
V_S = 300m/s

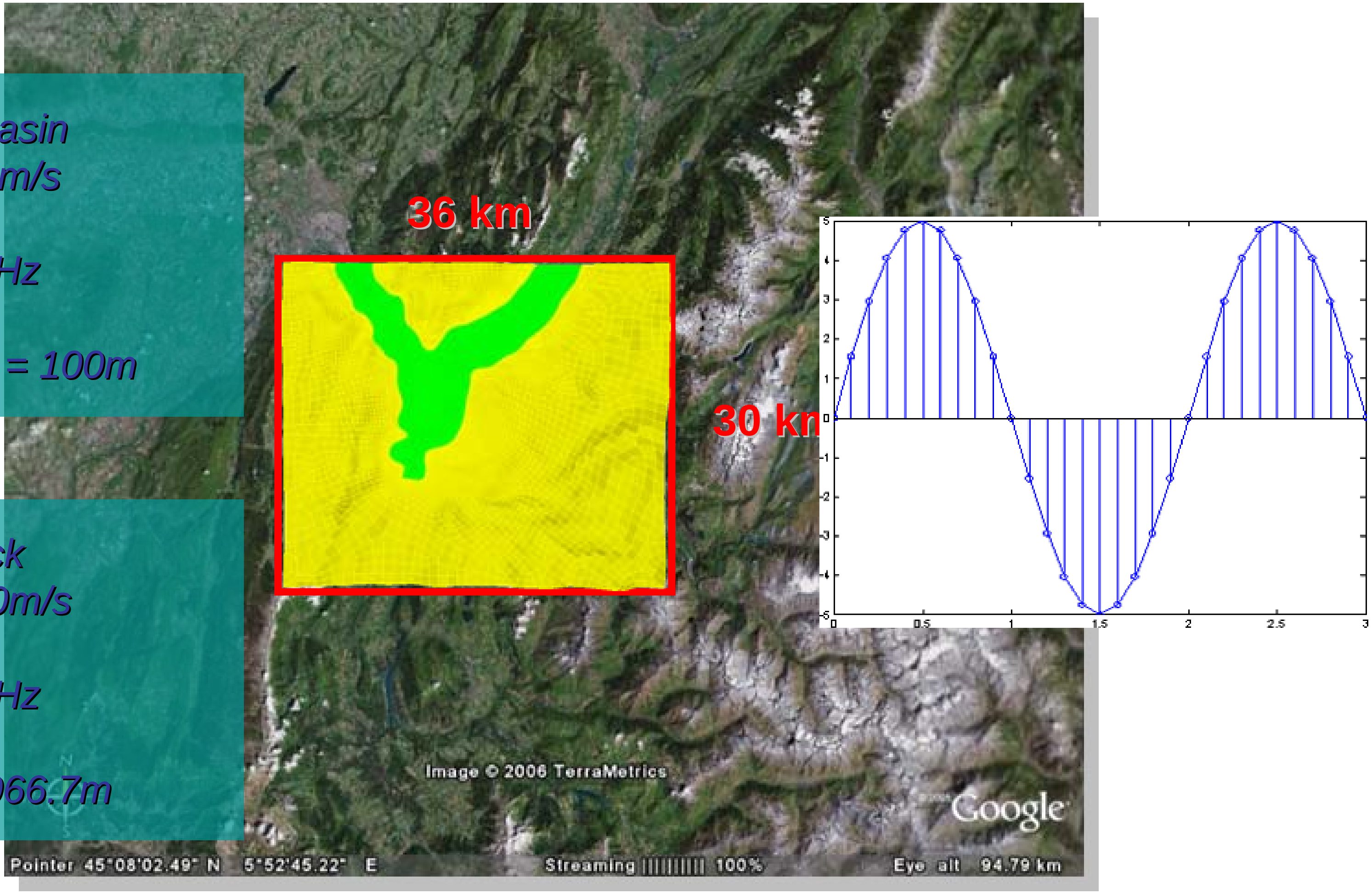
f_{max} = 3Hz

$\lambda_{min} = V_S/f_{max}$ = 100m

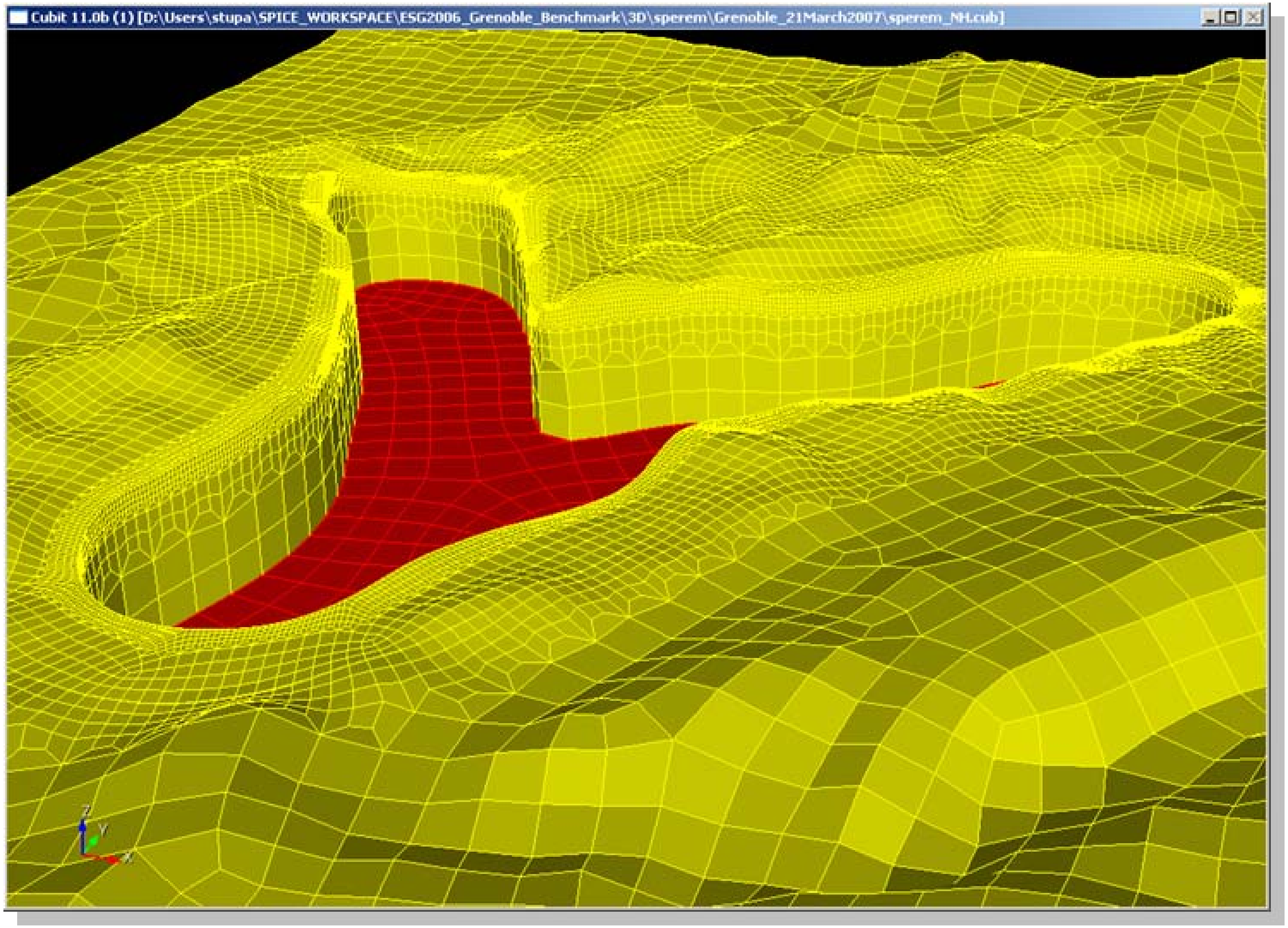
Bedrock
V_S = 3200m/s

f_{max} = 3Hz

$\lambda_{min} = V_S/f_{max}$ = 1066.7m



Hexahedral Grid Generation



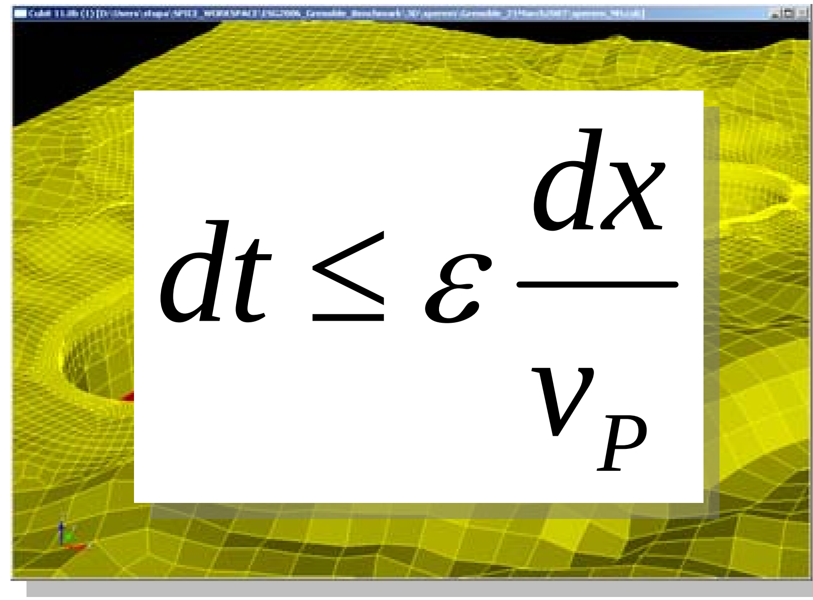
Courant …

Time step

$$v_P \left(\frac{dt}{dx} \right) \le \varepsilon$$

Largest velocity

Smallest grid size



$$dt \le \varepsilon \frac{dx}{v_P}$$

Courant and all that … - Summary

- Any numerical solution has to be checked if it converges to the correct solution (as we have seen there are different options when using FD and not all do converge!)
- The number of grid points per wavelength is a central concept to all numerical solutions to wave like problems. This desired frequency for a simulation imposes the necessary space increment.
- The Courant criterion, the smallest grid increment and the largest velocity determine the (global or local) time step of the simulation

Examples on the exercise sheet.