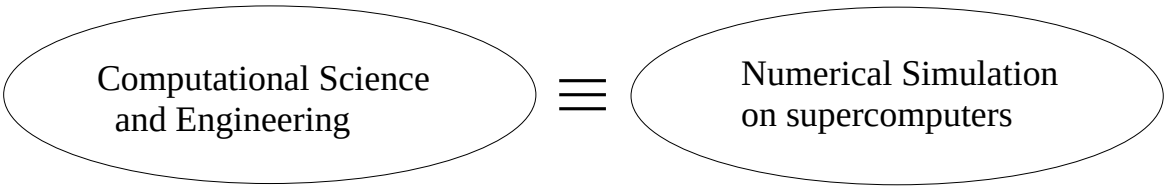
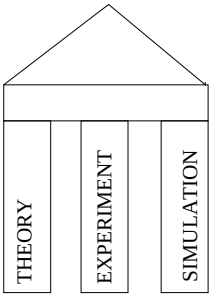
Computational Science and Engineering (CS & E)

CS & E deals with the simulation of processes, in engineering, physical but also in economic sciences. The simulation is typically supercomputer based.

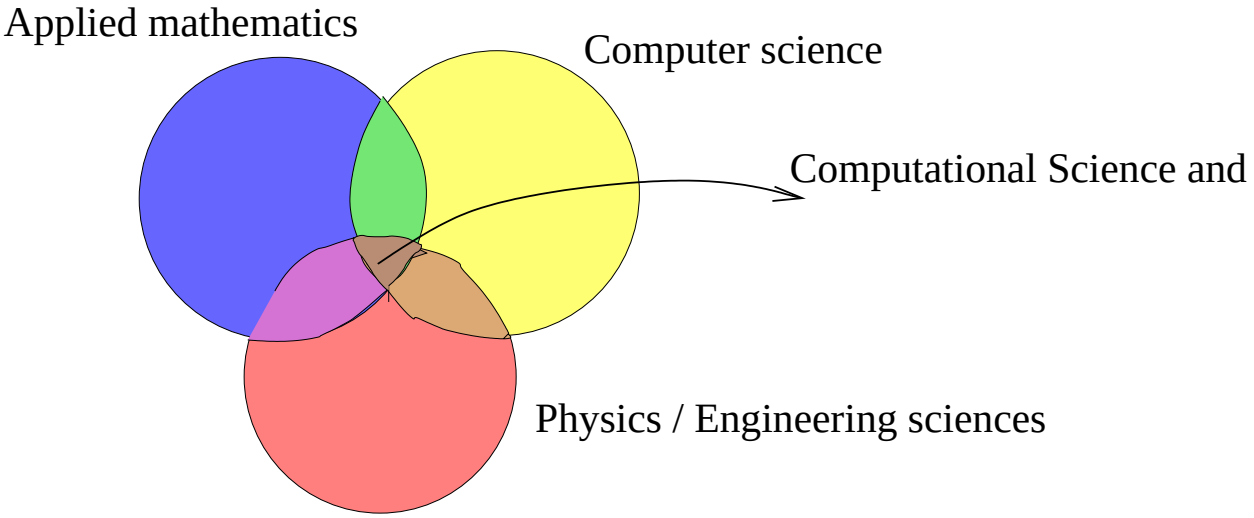


- Examples of application areas are fluid dynamics (the aerodynamics of cars and aircrafts, combustion processes, pollution spreading), semi conductor technology (breeding of cristals, oxidation processes), weather and climate prediction (the growing and tracking of tornados, global warming) but also financial mathematics (prediction of stock and option prices).
- Numerical simulation is nowadays an equal and indispensable partner in the advance of scientific knowledge next to theoretical and experimental research.



Interdisciplinary

- It is characteristic for scientific computing that practical relevant results are only achieved by combining methods and research results from different scientific areas.
- The application area brings the typical problem-dependent know-how and the knowledge to verify the computed results by means of real experiments.
- Applied mathematics deals with the definition of a mathematical model and develops fast numerical methods to solve the model with a computer.
- Computer science typically takes care for the usability, and programming of modern supercomputers and designs powerful software packages.



Six Steps

Basically, we can distinguish six important steps in simulation: setting up a model, numerical treatment, implementation, embedding, visualization and validation:

Setting up a model. At the start of each simulation we have the development of a mathematical model of the process of interest. This must be a simplified image of the reality that contains many relevant phenomena. It must be formulated such that the model can be solved with a computer and has a unique solution. Often we obtain a system of (not analytically solvable) differential equations.

The numerical treatment. As a computer can only handle discrete numbers, we have to discretize the model (the equations and the solution domain), so that the computer can deal with them.
For the solution of the resulting matrix from the discrete equations, mathematicians provide efficient numerical methods.

The implementation. Next to the choice of computer language and data structures, especially the distributed computation is of major importance.

4 The embedding. The numerical simulation is just one step in the product development in industrial applications. Therefore, we need interfaces, so that we can link the simulation programme with CAD tools, for example.

   Only in this way it is, for example, possible to use aerodynamics car simulation results on drag coefficients at an early stage into the design process.

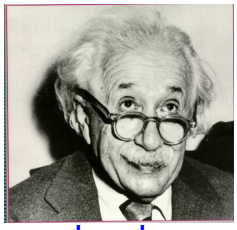
5 The visualisation. Typically, we have after a simulation huge data sets (not only one drag coefficient as in the example above).

   Often we need the velocity in the complete flow domain, or one is interested for the optimal control of a robot in the path of the robot arm. We need to present such results with the help of computer graphics, i.e. visualization is very important.

6 The validation. After long computer times with many many computations, we have the result of a simulation. It is of a high importance to verify the results obtained.

Introduction

Errors in scientific computing



"If mathematical theories refer to reality, then they are not certain. If they are certain, then they do not refer to reality. "

Seemingly: also numerical mathematics is not an "exact" science. We wish to compute an exact solution x, but we obtain an approximation $\hat{x}$

$\Rightarrow$ An important aspect of numerical math is understanding the errors made in computing solutions.

$\Rightarrow$ Find an approximation $\hat{x}$ for the solution x with a known accuracy (we have to consider possible errors and their influence on the accuracy.)

• Obtaining an approximate solution without knowing anything about the approximation error is useless !

Error measures: For $x \in \mathbb{R}$:

$$|x - \hat{x}| \leq \epsilon \quad \text{Estimate of the absolute error}$$

$$\frac{|x - \hat{x}|}{|x|} \leq \epsilon \quad \text{Estimate of the relative error}$$

• Area of conflict: Estimates are only of interest, if they are (more or less) realistic.

Algorithms, Methods, Code

A purpose of scientific computing is the development algorithms.

Algorithm: (s. computer science)

A set of instructions to carry out certain mathematical, arithmetical and logical operations (or already known algorithms) for solving a prescribed problem.

Important: finite algorithm: Solution is obtained after a finite number of iterations

Example of an infinite algorithm: The Babylonian root extraction method:

$$\sqrt{a} : (a > 0) : \quad x_0 > 0 \quad \text{arbitrary}$$
$$n = 1, 2, 3, ... \quad x_n = \frac{1}{2}(x_{n-1} + \frac{a}{x_{n-1}})$$

Practically: infinite algorithms not interesting.

Infinite algorithm $+$ stopping criterion $\Rightarrow$ finite algorithm

(for example: $|x_n - x_{n-1}| \leq \varepsilon$)

Often: an algorithm $=$ finite algorithm

Introductory example

Babylonian root extraction

Given: $a > 0, \quad a \in \mathbb{R}$

Required: $\sqrt{a}$, or a sufficient good approximation for it

Method:

1) Choose $x_0 \in \mathbb{R}, \quad x_0 > 0$ arbitrary (f.e., $x_0 = 1$)

2) Compute for $n = 1, 2, 3...$ $\qquad x_n = 1/2(x_{n-1} + a/x_{n-1})$ $(*)$

Theorem: The method of Babylonian root extraction, i.e. the sequence $\{x_n\}_{n \in \mathbb{N}}$ defined by $(*)$ is well defined for each positive initial approximation x_0. The sequence $\{x_n\}$ converges for each positive initial approximation x_0 towards $\sqrt{a}$. Further, it can be shown that:

(1) $x_n \geq x_{n+1} \geq \sqrt{a}$ for $n = 1, 2, ...$

(2) $x_n \geq \sqrt{a} \geq a/x_n$ for $n = 1, 2, ...$

(3) $(x_n - \sqrt{a})/\sqrt{a} \leq 1/2^{n-1}(x_1 - \sqrt{a})/\sqrt{a}$ for $n = 1, 2, ...$
(a-priori bound)

(4) $(x_n - \sqrt{a})/\sqrt{a} \leq (x_n - x_{n-1})^2/2a$ for $n = 1, 2, ...$
(a-posteriori bound)

Derivation/ motivation of the algorithm:

Desired: zeros of $g(x) = x^2 - a = 0$

Derive via the Newton-Raphson algorithm:

$$\begin{aligned} x_n &= x_{n-1} - \frac{g(x_{n-1})}{g'(x_{n-1})} \\ x_n &= x_{n-1} - \frac{x_{n-1}^2 - a}{2x_{n-1}} = x_{n-1} - \frac{1}{2}x_{n-1} + \frac{1}{2}\frac{a}{x_{n-1}} \\ x_n &= \frac{1}{2}(x_{n-1} + \frac{a}{x_{n-1}}) \end{aligned}$$

Observation: Error bound (3) is correct, but it is not sharp:

($a = 0.64,\ \sqrt{a} = 0.8$)

n	x_n	$x_n - \sqrt{a}$	$(x_n - \sqrt{a})/\sqrt{a}$	(3)
0	0.76	0.04	0.05	0.05
1	0.801052631	0.001052631	0.001316 ...	0.001316
2	0.800000691	0.000000691	0.000000864	0.000658
3	0.800000000 ...	$< 10^{-10}$	$< 1.2510^{-10}$	0.000329

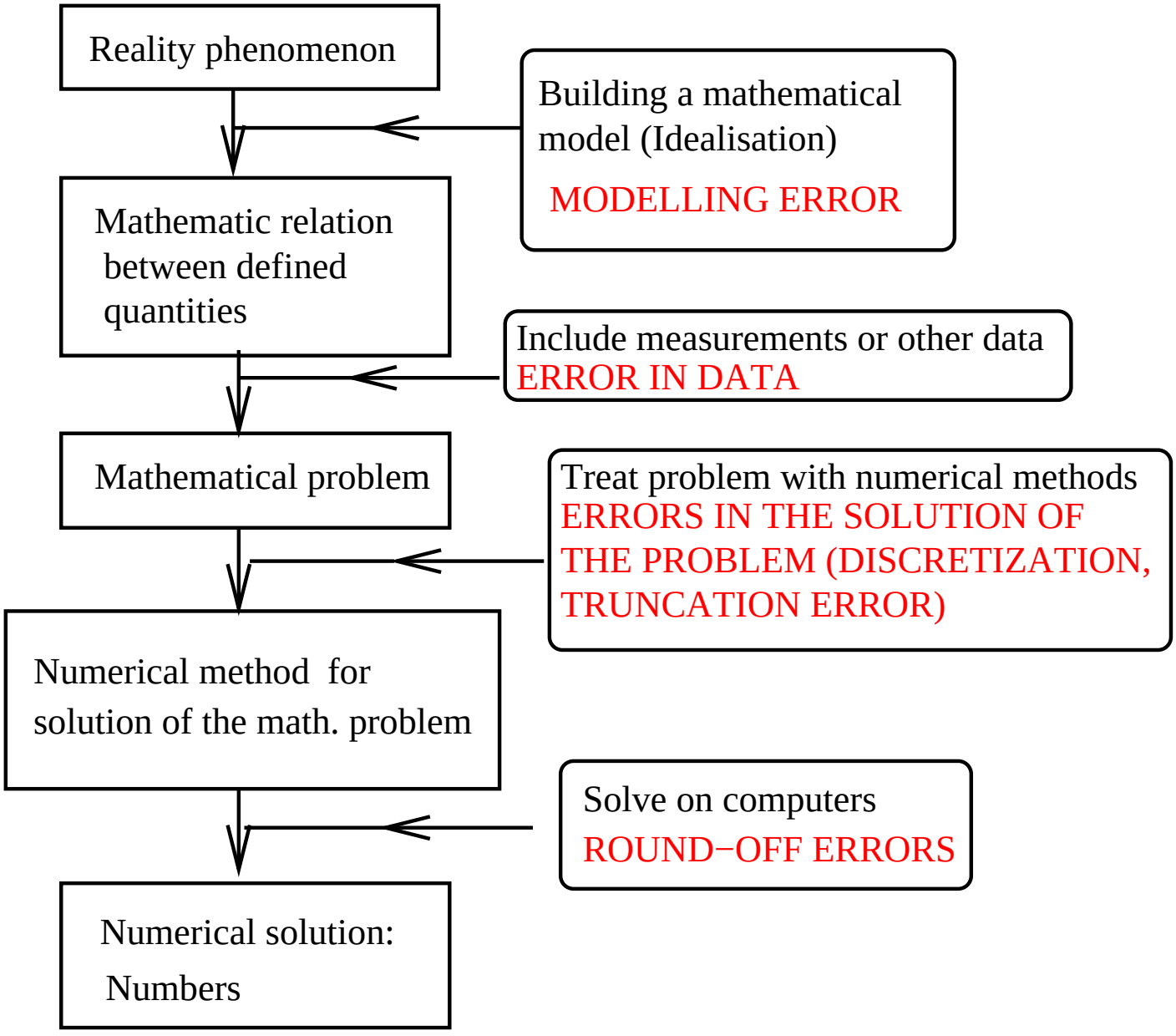
Improved error bound:

$$\frac{x_n - \sqrt{a}}{\sqrt{a}} = \frac{1}{2}\left(\frac{x_{n-1} - \sqrt{a}}{\sqrt{a}}\right)^2 \frac{\sqrt{a}}{x_{n-1}}$$

$\Rightarrow$ If $x_0 \geq \sqrt{a}$:

$$\frac{x_n - \sqrt{a}}{\sqrt{a}} \leq \frac{1}{2^{2^n-1}}(\frac{x_0 - \sqrt{a}}{\sqrt{a}})^{2^n}$$

Scheme


Reality phenomenon
Building a mathematical model (Idealisation)
MODELLING ERROR
Mathematic relation between defined quantities
Include measurements or other data
ERROR IN DATA
Mathematical problem
Treat problem with numerical methods
ERRORS IN THE SOLUTION OF THE PROBLEM (DISCRETIZATION, TRUNCATION ERROR)
Numerical method for solution of the math. problem
Solve on computers
ROUND−OFF ERRORS
Numerical solution: Numbers

Modeling errors

• Practical problems are physical, chemical, biological or economical problems.

It is necessary to set up a model, which is often a partial differential equation (pde).

But: (for example, for simplicity) a linear simplification of nonlinear processes

In fluid mechanics:

Laminar – turbulent flow

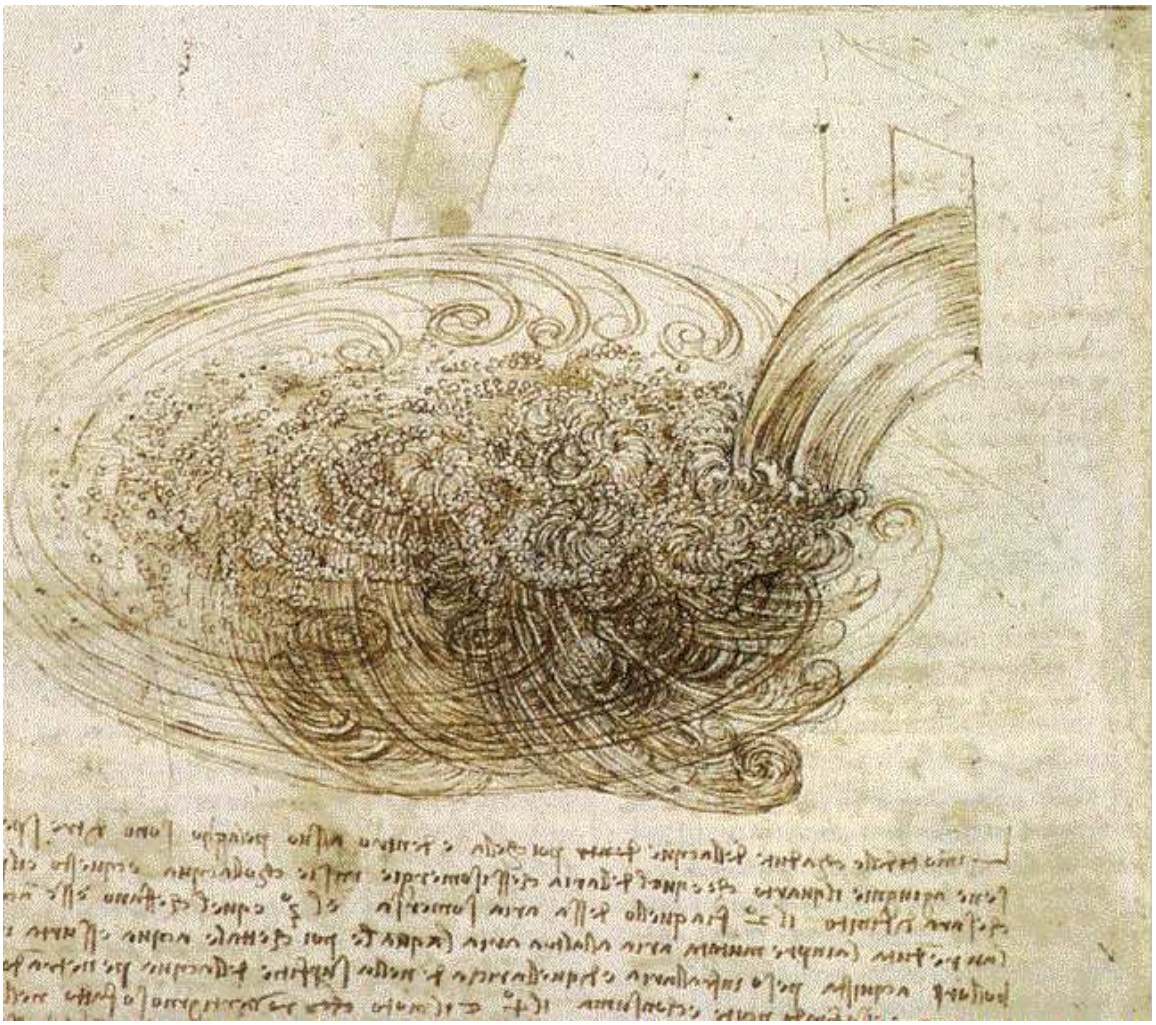
Euler equations $\longleftrightarrow$ Navier-Stokes equations
(without friction – model with friction)

Data errors

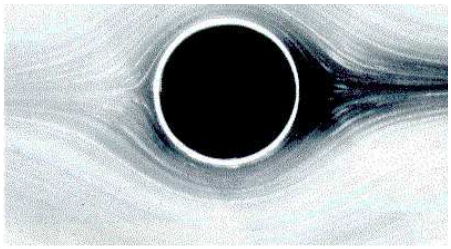
Except for trivial cases, most data that enter a computation are afflicted with errors, for example with measurement errors. These can have a big or a small effect on the numerical solution:

Bad $\leftrightarrow$ good condition of the problem

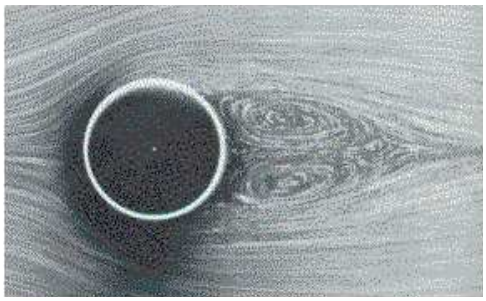
Turbulence: Modeling complex flow phenomena



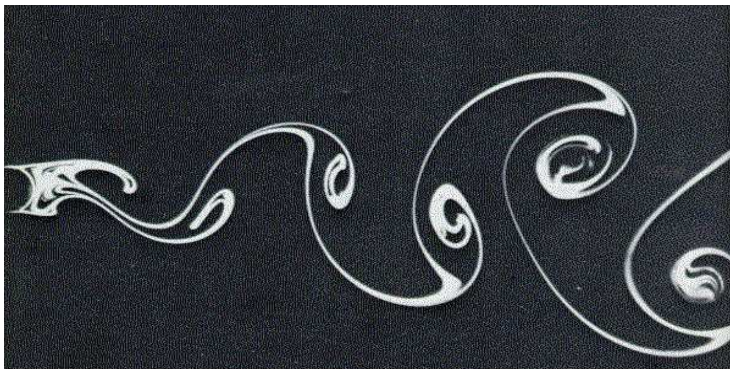
(Leonardo da Vinci ~ 1508)



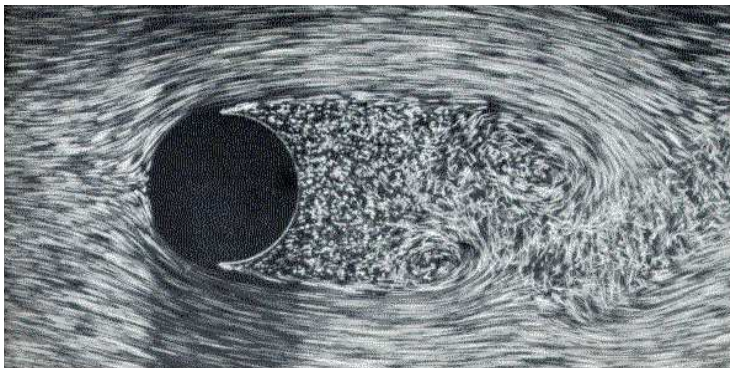
$Re = 0.16$,
Laminar, stationary,
symmetric



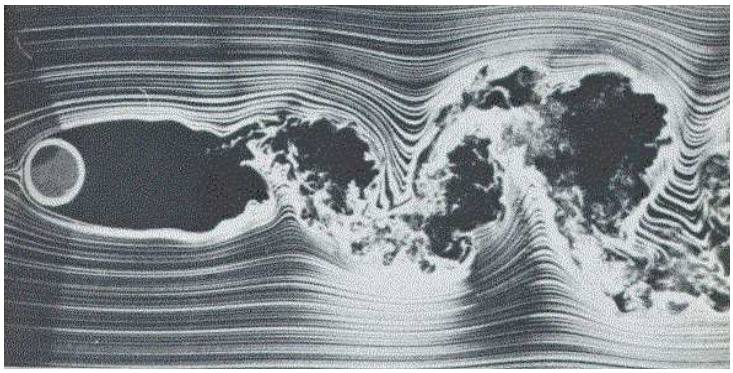
$Re = 26$,
Laminar, stationary,
recirculation



$Re = 140$,
Transition, unsteady,
von Kármán vortex street

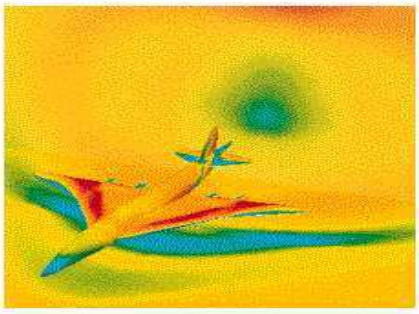


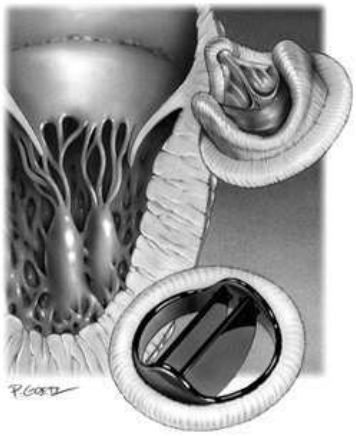
$Re = 2000$,
Turbulent, unsteady

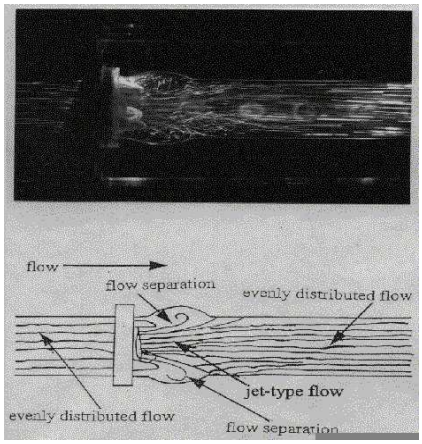


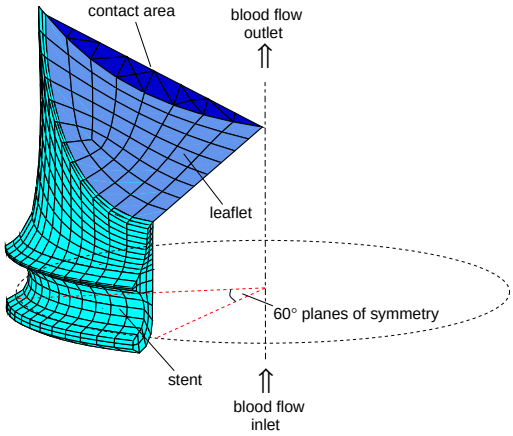
$Re = 10000$,
Turbulent, unsteady

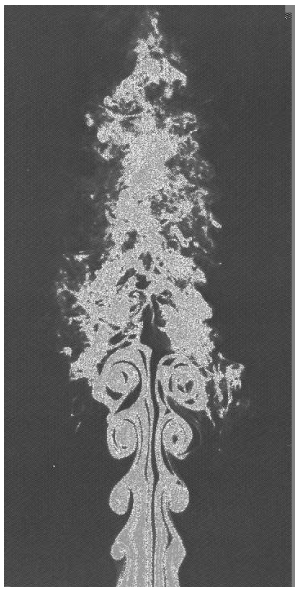
Examples of turbulent flow

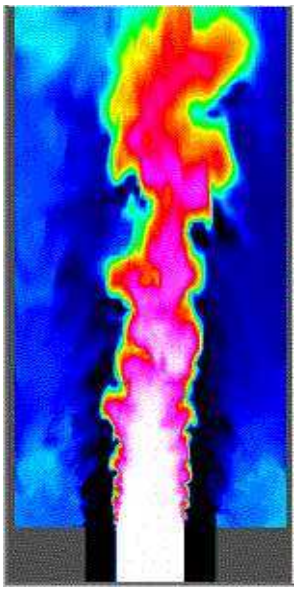












Necessity for numerical simulation

- For optimization of designs, engineers have to predict flow phenomena

- Experiments on the real object are often impossible or very expensive
- Model experiments can be expensive and time consuming, certain flow phenomena cannot be simulated in a laboratory

  $\Rightarrow$ Computations are for certain flows the only way to gain insight, for other flows they are cheaper than experiments.

The mathematical equations

Flow phenomena are usually described by the Navier-Stokes equations. x-momentum:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} + \frac{\partial p}{\partial x} = \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}\right)$$

$$Re = \overline{U}L/\nu \text{ (Reynoldsnumber)}$$

(similar for y- and z-momentum)

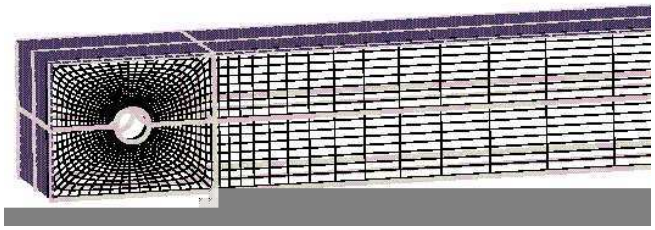
(Navier 1785-1835, Stokes 1819-1903)

continuity equation for an incompressible fluid:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

- $Re < Re_{tr}$: laminar, $Re > Re_{tr}$: turbulent

- System of nonlinear partial differential equations, Solution is 'only' possible with numerical methods
- There are methods, by which it is in principle possible to solve the equations: $\Rightarrow$ direct numerical simulation (DNS)



The Reynoldsnumber

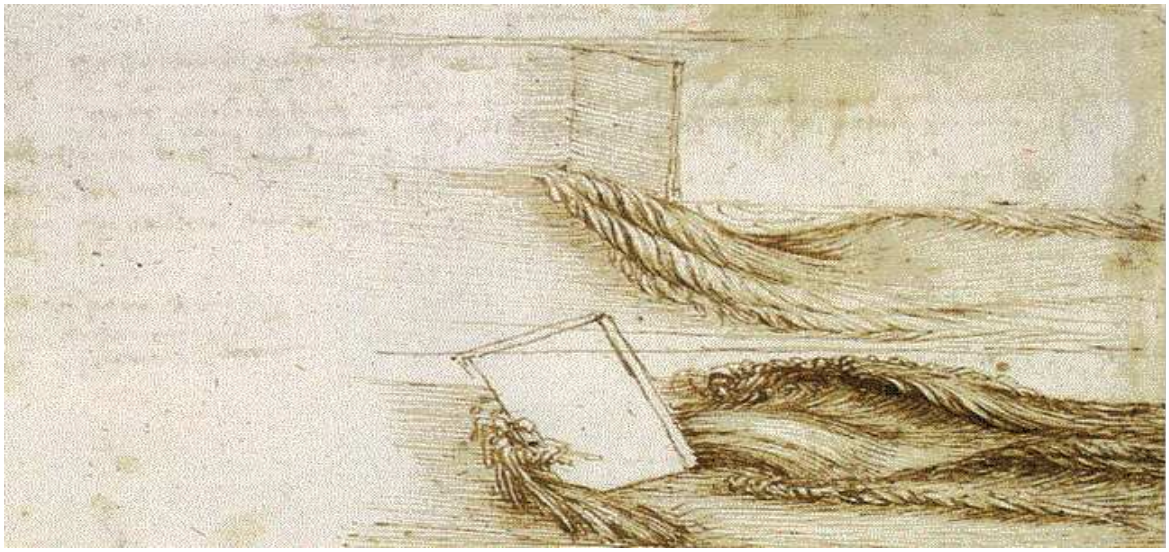
$$\Longrightarrow Re = \overline{U}\, L/\nu$$

- spoon in cup of coffee $Re \sim 10^4$
- air flow around a car ($50km/h$) $Re \sim 3 \cdot 10^6$
- water flow in a river $Re \sim 10^7$
- air flow around an aircraft $Re \sim 3 \cdot 10^8$

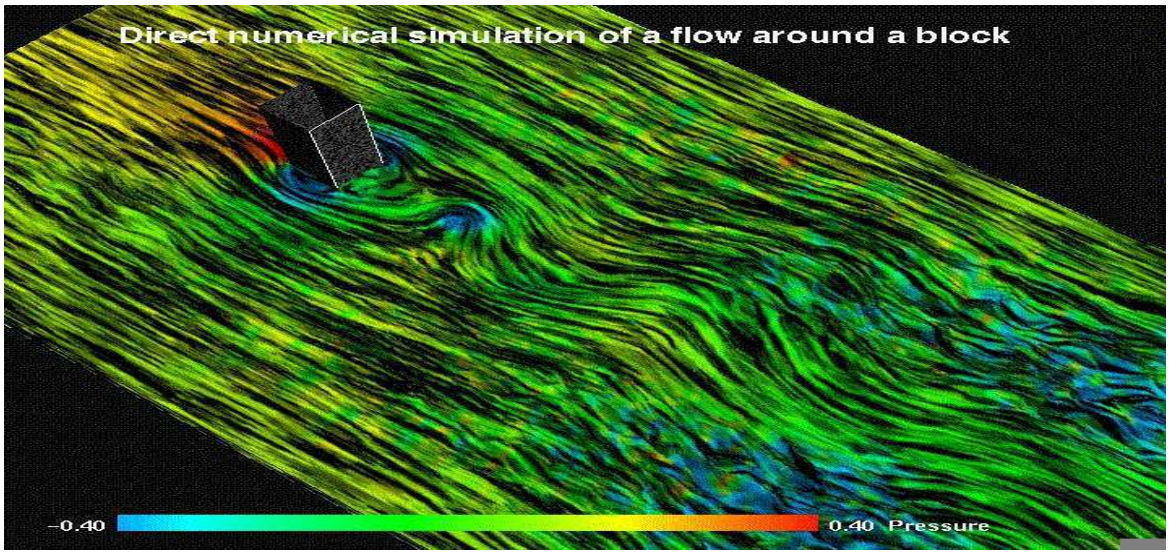
DNS, an example

- channel flow at $Re = \overline{U} \cdot \mathcal{L}/\nu = 10^5$
- smallest length scale $\eta \approx \mathcal{L}/1000$
- resolve the smallest turbulence movements:
  1000 points in each direction: 10^9 points
- computer storage: $> 10 \times 10^9$
- number of computing operations: ~ 500
- number of time steps: ~ 10000
  $\Rightarrow$ Total number of operations 5×10^{15}

- DNS is not yet applicable for practical applications
- With the increasing power of supercomputers, it becomes possible to analyze increasingly larger Reynoldsnumbers.
- Then it is possible to study the basic mechanisms of turbulence.



Leonardo

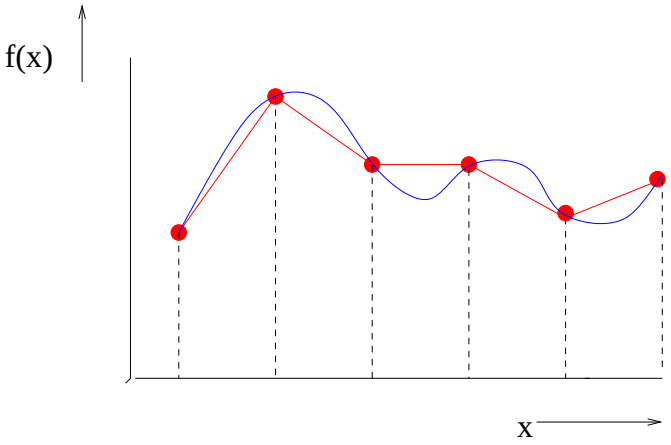


DNS

$Re = 22000$

(numerical simulation)

Discretization error

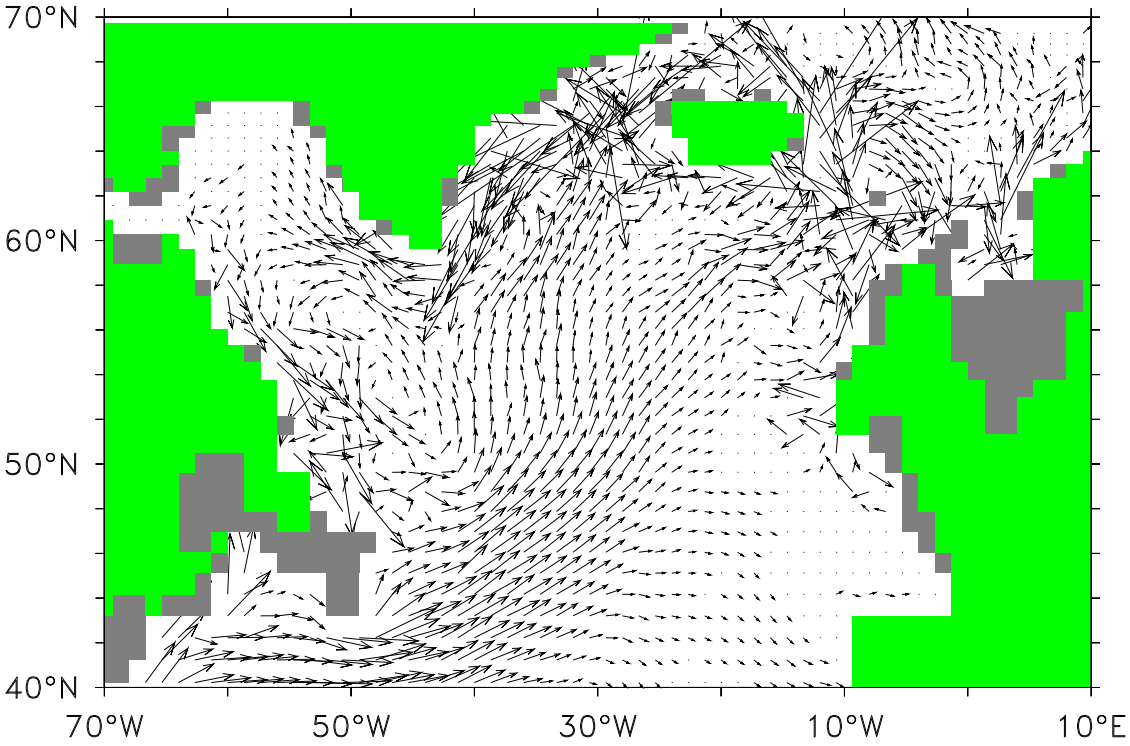


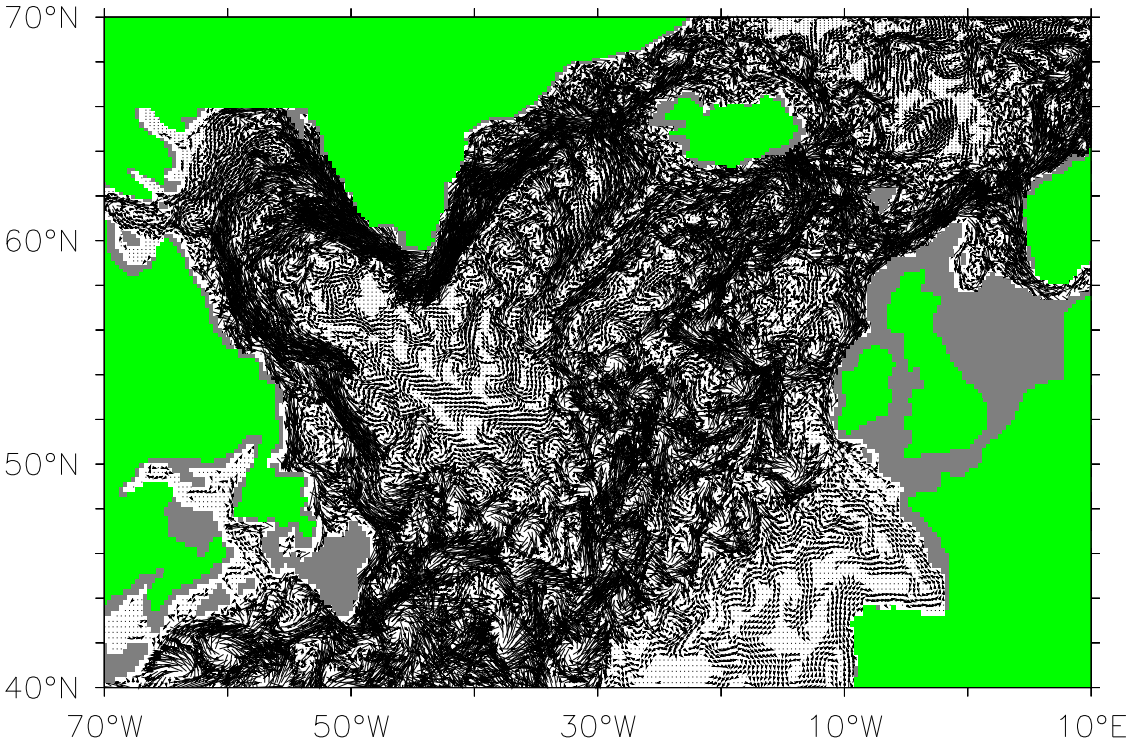
Only n values are obtained: gives an approximate solution curve.

Round off errors

Only numbers of a finite length are used in computers. Practically, this means that already during multiplication/ division numbers are rounded. Each separate round off error may be small, however, during long calculations many small errors can add up, to make a solution totally useless.

Stable $\leftrightarrow$ unstable algorithms


70°N
60°N
50°N
40°N
70°W
50°W
30°W
10°W
10°E


70°N
60°N
50°N
40°N
70°W
50°W
30°W
10°W
10°E

The simulation of recirculation in the Atlantic Ocean

	4/3 Grad Atlantic Model	1/3 Grad Model
Detail near equator	148 km	ca. 27 km
Detail in northern latitudes	70 km	13 km
# grid points	$78 \times 150 \times 45$	$308 \times 600 \times 45$
# Proc. Cray T3E	15	120
elapsed time	1.5 hours	8 hours
(for 1 year simulation)		

Simulation time for future applications:

for simulation of biological-physical processes: 3-5 years

for the spreading of geochemical species: 50 years

for the spreading and absorption of CO2: $>$ 100 years

Remark: For many applications it would be appropriate to consider global models. The costs in CPU time would then be four times as high.

Presently, research considers coupled models, based on a coupling of ocean-atmosphere-geosphere.

Linear system with bad condition

$$\begin{pmatrix} 1 & 1 \\ 1 & 1-\varepsilon \end{pmatrix} x = \begin{pmatrix} 4 \\ 4-\varepsilon \end{pmatrix} \qquad Ax = b$$

$$x^* = \begin{pmatrix} 3 \\ 1 \end{pmatrix}$$

But also:

$$\begin{pmatrix} 1 & 1 \\ 1 & 1-\varepsilon \end{pmatrix} \tilde{x} = \begin{pmatrix} 4+\varepsilon \\ 4-2\varepsilon \end{pmatrix} \qquad A\tilde{x} = \tilde{b}$$

$$\tilde{x}^* = \begin{pmatrix} 1+\varepsilon \\ 3 \end{pmatrix}$$

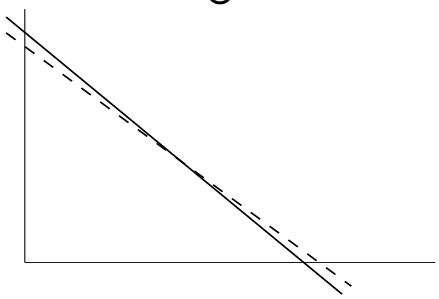
Also: $||b - \tilde{b}||_\infty = \varepsilon$, but $||x - \tilde{x}|| = \max\{|2-\varepsilon|, 2\} = 2$

The error of the approximation is always 2, whereas the change in rhs is only ε !

"Bad condition of the problem"

Geometrical interpretation: Lines cut under a very sharp angle

$\Rightarrow$ Small change in a line changes the cutting point drastically



Influence of round off errors

Problem: Computation of

$$\begin{aligned} x_n &:= \int_0^1 \frac{t^n}{t+10} dt \quad \text{for larger } n \\ &> 0 \quad \text{clearly} \\ x_0 &= \int_0^1 \frac{1}{t+10} dt = \ln(10+t)|_0^1 = \ln \frac{11}{10} = \ln 1.1 \approx 0.0953 \\ x_n &:= \int_0^1 \frac{t^n}{t+10} dt = \int_0^1 \frac{t^{n-1}(t+10) - 10t^{n-1}}{t+10} dt \\ &= \int_0^1 t^{n-1} dt - 10x_{n-1} = \frac{1}{n} - 10x_{n-1} \end{aligned}$$

Recursion: $x_n = 1/n - 10x_{n-1}$

$\Rightarrow$ In principle: Problem solved (?)

Practically: computation with 3 decimals gives the approximations $\tilde{x}_n$

$$\begin{aligned} \tilde{x}_0 &= 0.0953 \\ \tilde{x}_1 &= 1 - 0.953 = 0.047 \quad (x_1 = 0.0469..) \\ \tilde{x}_2 &= 0.5 - 0.470 = 0.030 \quad (x_2 = 0.0310..) \\ \tilde{x}_3 &= 0.333 - 0.300 = 0.033 \quad (x_3 = 0.0232..) \\ \tilde{x}_4 &= 0.250 - 0.330 = -0.08 < 0! \quad (x_4 = 0.0185..) \end{aligned}$$

Nonsense $\tilde{x}_3 > \tilde{x}_2$ is already nonsense

Reason: x_{n-1} contains error ε

$\Rightarrow$ x_n contains error approx. 10ε

$\Rightarrow$ x_{n+1} contains error approx. 100ε. "Unstable recursion"

Remedy here: reversed procedure:

$$x_{n-1} = \frac{1}{10n} - \frac{1}{10}x_n = \frac{1}{10}(\frac{1}{n} - x_n)$$

$\Rightarrow$ Error becomes about $1/10$ smaller

Start ?

$$x_n \leq \int_0^1 \frac{1}{10}t^n dt = \frac{1}{10(n+1)} \to 0 \quad (n \to \infty)$$

employ recursion with $y_n = 0,\ y_{n-1} = 1/10(1/n - y_n)$

Error in $y_n < 1/(10(n+1))$

Error in $y_{n-1} < 1/(100(n+1))$

Error in $y_{n-j} < 1/(10^{j+1}(n+1))$

for example,

$$\begin{aligned} y_8 &= 0 \\ \tilde{y}_7 &= 0.0125 \\ \tilde{y}_6 &= 0.0143 - 0.00125 = 0.0131 \\ \tilde{y}_5 &= 0.0167 - 0.00131 = 0.0154 \\ \tilde{y}_4 &= 0.0200 - 0.00154 = 0.0185 \end{aligned}$$

$= x_4$ is 3 digits accurate