A Short Course in Parallel Scientific Computing

Technical University of Budapest

Graham Horton, University of Erlangen-Nürnberg

Talk 4 : Ordinary Differential Equations

Contents

Time-Dependent Problems

Most phenomena are time-dependent (unsteady, non-stationary)

The stationary PDE is a special case

Example : Switching on an electric oven

Problems that are only meaningful when time-dependent:

- Electrical and electronic circuits
- Turbulent flows
- Many-body problems
- Periodic and chaotic systems

Ordinary Differential Equations

Dynamics are usually described by *ordinary differential equations*

$$\frac{du}{dt} = g(u, t)$$

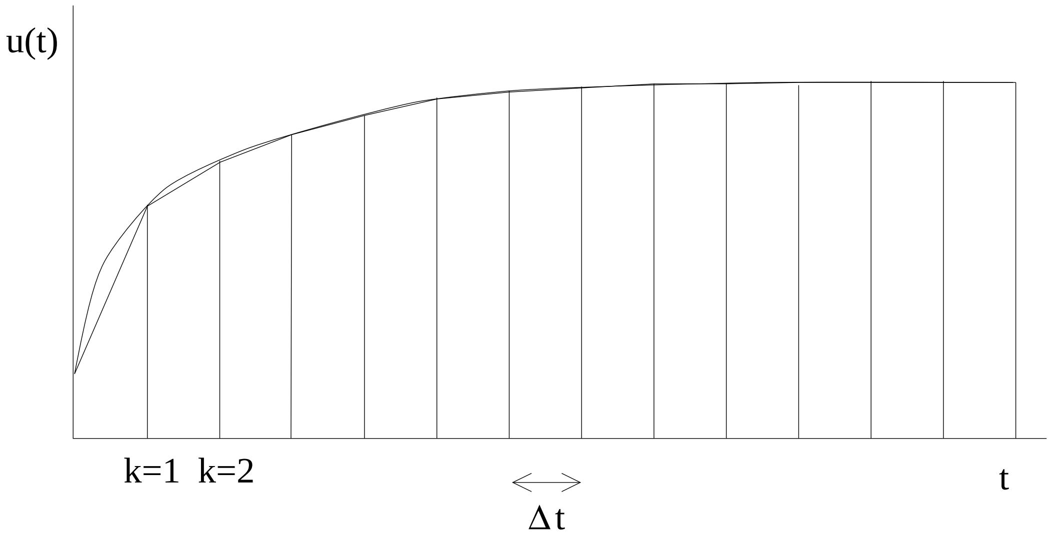
with

$$u(t_0) = u_0 \ \text{ defined}$$

Problem: determine $u(t)$! *(Initial value problem)*

Numerical solution is usually necessary

Discretise the time axis:



Compute approximations for the solution at discrete time points t_k

Example

Speed of a falling object (in a vacuum)

Initial velocity $u(0) = u_0$

$$\frac{du}{dt} = g \qquad (= 9.81 m/s^2)$$

Analytic solution:

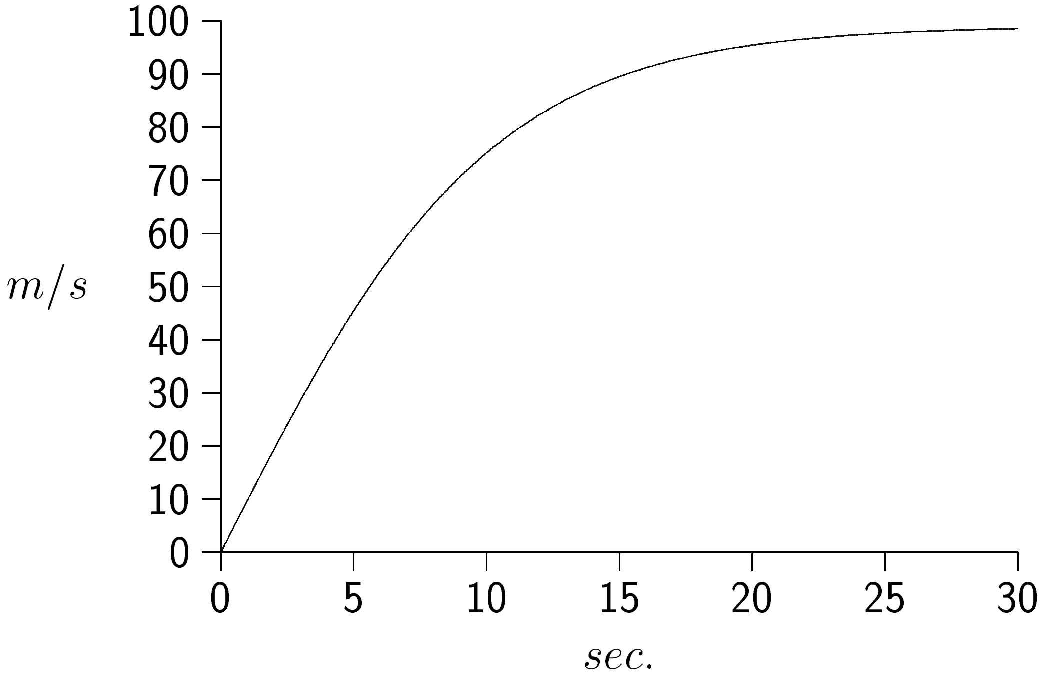
$$u = gt + u_0$$

Introduce wind resistance $= \alpha u^2$

$$\frac{du}{dt} = g - \alpha u^2$$

(Parachute equation)

Numerical solution for $\alpha = 0.001$:



Systems of ODEs

We are usually interested in systems of ODEs

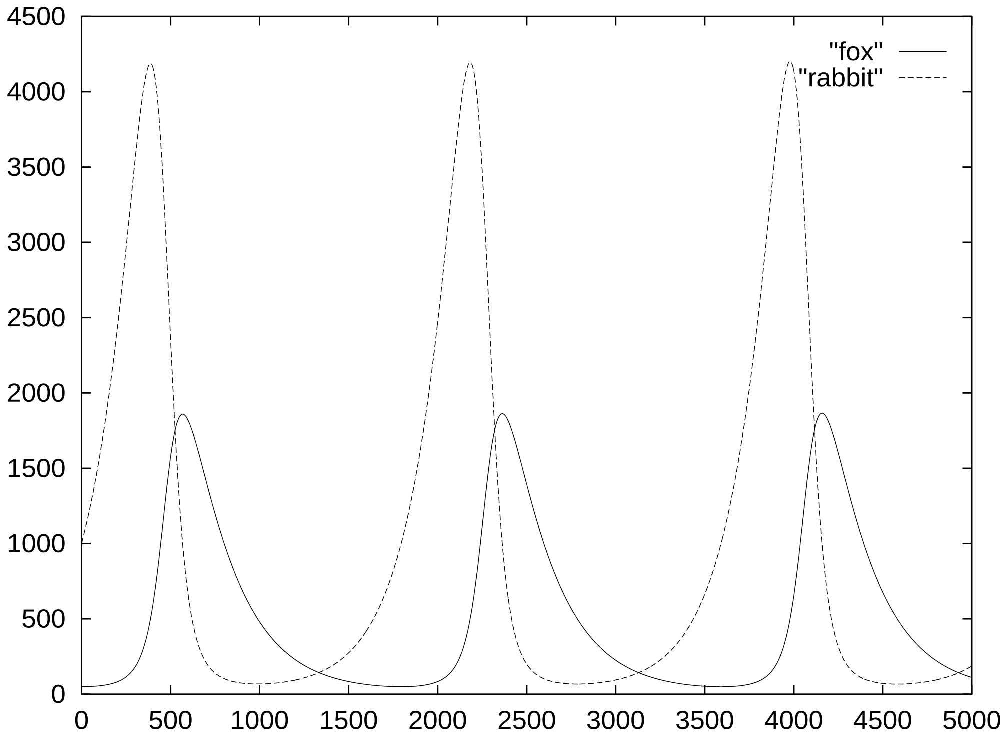
Example: Predator/prey-system

Volterra equations:

$$\begin{aligned}\frac{dr}{dt} &= ar - brf \\ \frac{df}{dt} &= -cf + drf\end{aligned}$$

(f = Fox population, r = rabbit population)

Numerical solution for $a = 50$, $b = 0.1$, $c = 40$, $d = 0.04$:



Volterra's Paradox: (small amount of) fishing increases fish population(!)

Time-Dependent (Parabolic) PDEs

The time-dependent version of our model problem

$$-\frac{\partial^2 \phi}{\partial x^2} - \frac{\partial^2 \phi}{\partial y^2} = f(x, y)$$

is

$$\frac{\partial \phi}{\partial t} - \frac{\partial^2 \phi}{\partial x^2} - \frac{\partial^2 \phi}{\partial y^2} = f(t, x, y)$$

We must define

$$\begin{array}{lll} \phi(t,x,y) & \text{at} \quad t = 0 & \text{and } \forall\ x, y \text{ in domain} \\ \phi(t,x,y) & \quad\ \forall\ t & \text{and } \forall\ x, y \text{ on boundary} \end{array}$$

(Initial and boundary conditions)

This describes the progress of the temperature distribution in a solid

If $f(t, x, y)$ is independent of t, we will reach a steady state

This is described by the steady-state equation, as $\frac{\partial \phi}{\partial t} = 0$

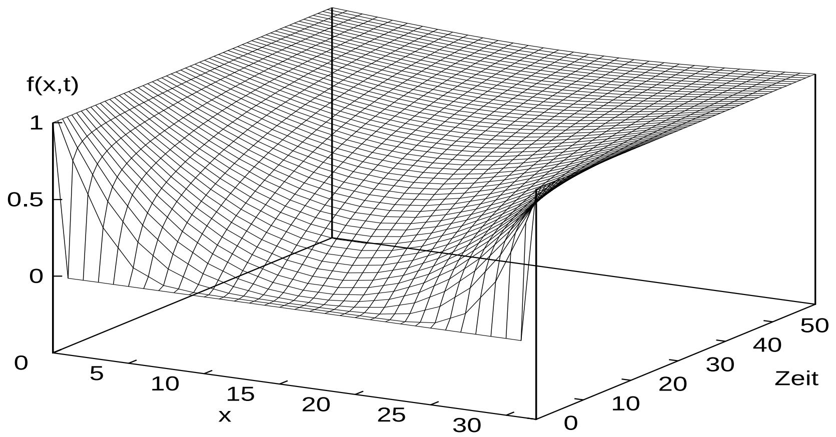
Time-Dependent (Parabolic) PDEs

Examples:

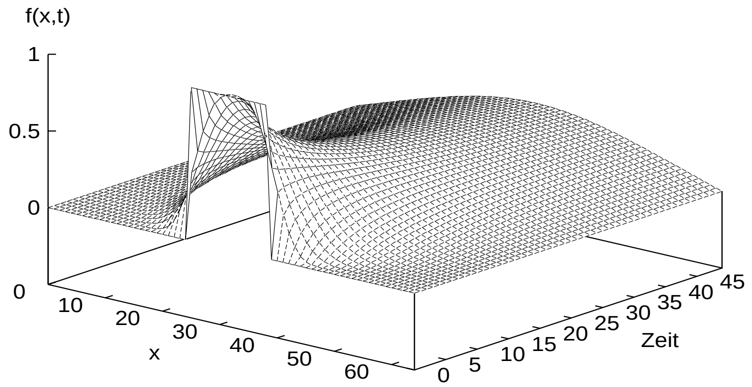
- Concentration of solution ("Sugar in coffee")
- Temperature distribution ("Switching on an oven")

Numerical solution for a 1-D oven plate:

Initial temperature=0, temperature at boundaries=1



Example: 1-D sugar cube in 1-D coffee cup:



Solution of ODEs

Obtain numerical methods from Taylor series approximations

From the Taylor series representation of $u(t)$:

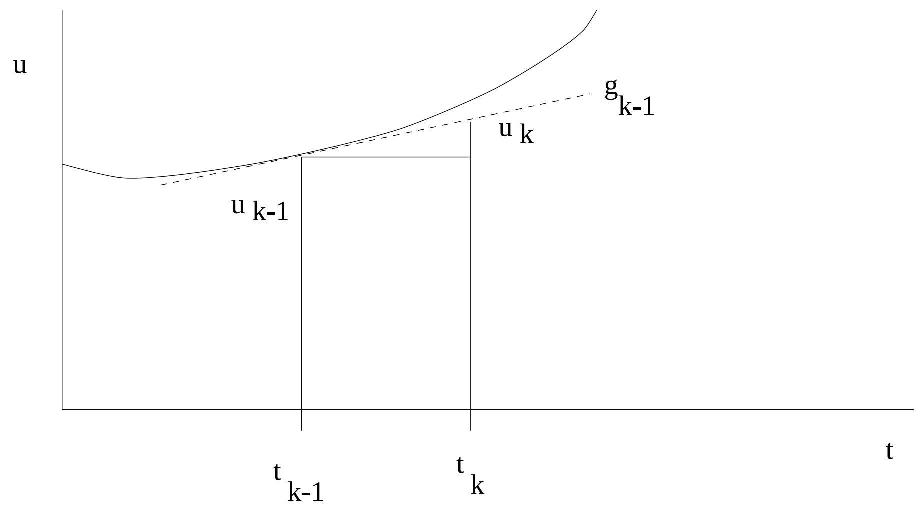
$$u(t+\Delta t) = u(t) + \Delta t u'(t) + \frac{\Delta t^2}{2} u''(t) + O(\Delta t^3)$$

we obtain

$$u'(t) = \frac{u(t+\Delta t) - u(t)}{\Delta t} + O(\Delta t^2)$$

This yields the explicit Euler method:

$$\frac{1}{\Delta t}(u_k - u_{k-1}) = g_{k-1}$$



u_k can be computed directly from the known values $u_{k-1},\ g_{k-1}$

In general *linear multistep methods*

The same holds for systems, if u, g are vectors

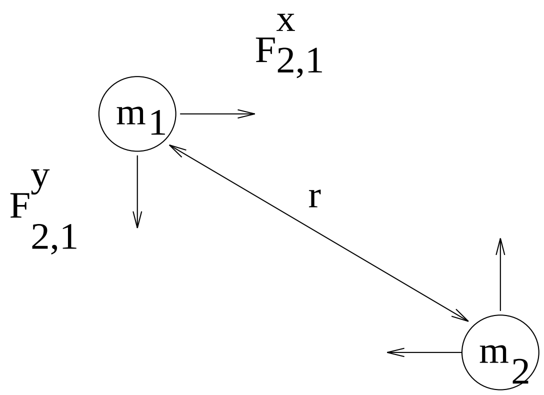
Example of a System of ODEs

Simulation of galaxies

Formation of spirals, clusters etc.

Application of law of gravitation for many ($> 10^5$) bodies

$$F_{1,2} = \Gamma \frac{m_1 m_2}{r^2}$$



Divide the forces and velocities into x, y and z components

Number of objects is very large

Simulation of long periods

Similar problem with $ab\ initio$ computations

Example: Compute structure of a molecule from atomic forces

Example of a System of ODEs

Each body has 6 variables:

Position: x, y, z, velocity: u, v, w

We have : $u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$, $w = \frac{dz}{dt}$

Newton: force $=$ mass $\times$ acceleration

i.e. $\frac{du}{dt} \propto$ sum of all forces acting on body

Form system of ODEs with $6n$ unknowns with n bodies

$$\frac{d}{dt}\begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ u_1 \\ v_1 \\ w_1 \\ x_2 \\ y_2 \\ z_2 \\ u_2 \\ v_2 \\ w_2 \end{pmatrix} = \begin{pmatrix} & & & 1 & & & & & & & & \\ & & & & 1 & & & & & & & \\ & & & & & 1 & & & & & & \\ X & X & X & & & & X & X & X & & & \\ X & X & X & & & & X & X & X & & & \\ X & X & X & & & & X & X & X & & & \\ & & & & & & & & & 1 & & \\ & & & & & & & & & & 1 & \\ & & & & & & & & & & & 1 \\ X & X & X & & & & X & X & X & & & \\ X & X & X & & & & X & X & X & & & \\ X & X & X & & & & X & X & X & & & \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \\ u_1 \\ v_1 \\ w_1 \\ x_2 \\ y_2 \\ z_2 \\ u_2 \\ v_2 \\ w_2 \end{pmatrix}$$

Implementation of the Galaxy Simulation

The explicit Euler method has the structure:

```
for t = t0 to tmax by dt
   for all bodies K_i
      forces = 0
      for all bodies K_j,  j ≠ i
         forces_i += force(K_i,K_j)
      compute new vel. and pos. for K_i
   update positions
end
```

Parallelisation of the Galaxy Simulation

The parallelisation is via the middle loop:

```
for t = t0 to tmax by dt
   for all own bodies K_i
      forces = 0
      for all bodies K_j, j ≠ i
         forces_i += force(K_i,K_j)
      compute new vel. and pos. of K_i
   Update own positions
   communicate all positions
end
```

Parallelisation is simple, since all computations at one timestep are independent

Computational effort for each body is identical

Distribute all bodies among all processors equally

Each processor must know the positions of all other bodies

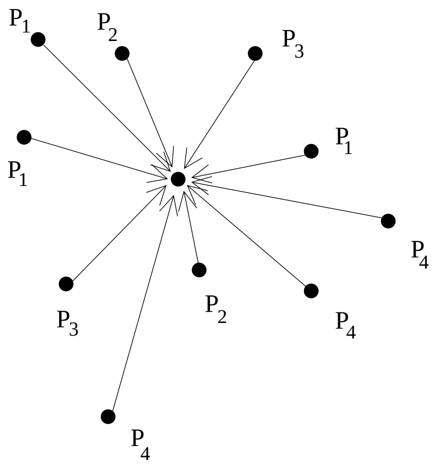
(but not their velocities)

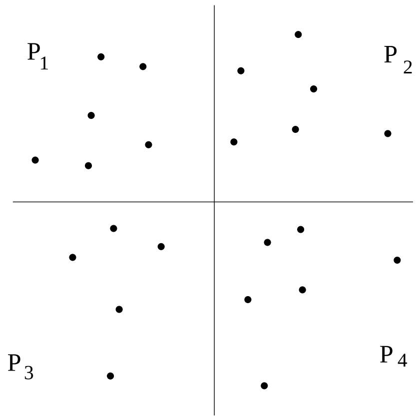
Computation per proc. and timestep : $\mathcal{O}(\frac{n^2}{p})$

Communication per proc. and timestep : $\mathcal{O}(n)$

Parallelisation of the Galaxy Simulation

We can use any load balanced partitioning of bodies to processors





Gravitation is *global* (note difference to PDE!)

All positions must be exchanged at each timestep

No advantage to a domain-based partitioning

Note large communication costs of the method

Discretisation of parabolic PDEs

Consider 1-D model problem

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

Write in the standard form for ODEs

$$\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + f(t, x)$$

or

$$\frac{du}{dt} = g(u, t) \tag{1}$$

Let $u_{i,k}$ be solution value in grid point u_i at time t_k

The discretisation of the space derivative is

$$\frac{1}{h^2}\begin{bmatrix} -1 & 2 & -1 \end{bmatrix}_{i,k}$$

Insert space discretisation into (**??**)

$\Rightarrow$ System of ODEs!

$$\frac{du_i}{dt} = \frac{1}{h^2}(-u_{i-1} + 2u_i - u_{i+1}) + f(t, u_i)$$

Discretisation of parabolic PDEs

Using the explicit Euler method

$$\frac{1}{\Delta t}(u_k - u_{k-1}) = g_{k-1}$$

we obtain

$$\frac{u_{i,k} - u_{i,k-1}}{\Delta t} = \frac{u_{i-1,k-1} - 2u_{i,k-1} + u_{i+1,k-1}}{h^2} + f_{i,k-1}$$

This can be solved immediately for $u_{i,k}$ $\forall i$:

```
for i=1 to n-1 do
   u[i] = uold[i]+dt*((
          uold[i-1] - 2*uold[i] + uold[i+1])/hsq + f[i])
```

N.B. `uold` is the solution at the previous timestep

Very efficient for all types of parallel computer

Example of a directly parallelisable algorithm

Solution of Parabolic PDEs

We obtain the explicit Euler method:

```
double u[0..n], ualt[0..n]

for t = t0 to t1 by dt do              % all timesteps

   u[0] = bound(0.0,t)                 % set boundary values
   u[n] = bound(1.0,t)

   for i = 1 to n-1 do                % new timestep
      u[i] = ualt[i] + dt*((
             ualt[i-1]-2*ualt[i]+ualt[i+1])/hsq + f(t-dt,i*h))

   for i = 0 to n do                   % new initial value
      ualt[i] = u[i]
```

Stability

Stability is an important characteristic of solution methods for ODEs

Stability describes the propagation of errors during the integration

In an *unstable* method errors can grow rapidly

Explicit Euler has for any problem a certain *stability interval* $(0, \epsilon)$

i.e. we must ensure that $0 < \Delta t < \epsilon$

Example : solve time-dependent heat equation

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

Let exact solution be $u(x,t) = e^{-t}e^{-x}$, i.e. $f = -2e^{-t}e^{-x}$

Solve with $n = 16$, $t = (0, 1)$, $x = (0, 1)$

Use $\Delta t = 0.05,\ 0.005,\ 0.001$ and explicit Euler

Observe solution value at $x = 0.5$

Stability

Δt	t	$u_{exakt}(0.5)$	$u_{EE}(0.5)$
0.05	0.1	$5.488116e-01$	$5.473391e-01$
	0.2	$4.965853e-01$	$4.936294e-01$
	0.3	$4.493290e-01$	$4.448643e-01$
	0.5	$3.678794e-01$	$-9.647627e+09$
	1.0	$2.231302e-01$	$-2.699556e+28$
0.005	0.1	$5.488116e-01$	$-8.750352e+06$
	0.2	$4.965853e-01$	$-2.357154e+19$
	0.3	$4.493290e-01$	$-3.904537e+31$
	0.5	$3.678794e-01$	$-9.610017e+55$
	1.0	$2.231302e-01$	$-8.930748e+116$
0.001	0.1	$5.488116e-01$	$5.491465e-01$
	0.2	$4.965853e-01$	$4.970785e-01$
	0.3	$4.493290e-01$	$4.498460e-01$
	0.5	$3.678794e-01$	$3.683364e-01$
	1.0	$2.231302e-01$	$2.234114e-01$

The stability interval for this problem is about $(0 \ldots 0.0019)$

$\Rightarrow$ upper limit for Δt

Small stability intervals lead to large computation times!