Control and optimization for non-local and fractional differential equations

Umberto Biccari and Enrique Zuazua

Chair of Computational Mathematics, Bilbao, Basque Country, Spain

Chair for Dynamics, Control and Numerics, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany.

Universidad Autónoma de Madrid, Spain.

umberto.biccari@deusto.es enrique.zuazua@fau.de

cmc.deusto.es dcn.nat.fau.eu

PART IV: numerical approximation of differential equations and numerical control

LECTURE 11: numerical approximation of ODE and PDE

Fundación Deusto Fundazioa

Friedrich-Alexander-Universität Department Data Science | DDS

European Research Council Established by the European Commission

European Commission

DyCon DYNAMIC CONTROL erc

Chair DYNAMICS, CONTROL AND NUMERICS FAU

Unterstützt von / Supported by Alexander von Humboldt Stiftung/Foundation

UAM Universidad Autónoma de Madrid

Finite difference approximation of derivatives

Approximation of derivatives: the finite difference approximation of derivatives

One of the **simplest** and of the **oldest** methods to solve differential equations.

It was already known by Euler, in one space-dimension, and extended to dimension two by Runge (ca. 1908).

The advent of finite difference techniques in numerical applications began in the early 1950s and their development was stimulated by the emergence of computers that offered a convenient framework for dealing with complex problems of science and technology.

Finite difference approximation of derivatives

The principle of finite difference methods consists in approximating the differential operator by **replacing the derivatives in the equation using differential quotients**.

The domain is partitioned into a mesh (uniform or non-uniform) and approximations of the operator are computed at the mesh points.

The error between the numerical operator and the exact operator is called the **discretization error** or **truncation error**.

The term truncation error reflects the fact that a finite part of a Taylor series is used in the approximation.

Finite difference approximation

The main concept behind any finite difference scheme is related to the definition of the derivative of a smooth function f at a point $x \in \mathbb{R}$

$$f'(x) = \lim_{h \to 0} \frac{1}{h} \Big(f(x+h) - f(x) \Big).$$

When h tends to 0 (without vanishing), the quotient on the right-hand side provides a good approximation of the derivative.

What does **good approximation** mean?

Approximation error

Approximation error

Error committed when replacing the derivative by the differential quotient.

If the function fu is sufficiently smooth in a neighborhood of x, it is possible to quantify this error using a Taylor expansion.

For any $h > 0$, we have

$$f(x+h) = f(x) + f'(x)h + \mathcal{O}(h^2).$$

We deduce that

$$\mathcal{E}_h := \left| \frac{1}{h}\Big(f(x+h) - f(x)\Big) - f'(x) \right| \leq Ch, \quad \text{with} \quad \mathcal{C} = \sup_{y \in [x, x+h]} \frac{|u''(y)|}{2}$$

The approximation error is of order h. The approximation of f' at point x is said to be **consistent at the first order**.

This approximation is known as the **forward difference approximant of** f'.

$$\partial_h^+ f(x) = \frac{1}{h}\Big(f(x+h) - f(x)\Big)$$

Approximation error

Order of approximation

The approximation of the derivative f' at point x is **of order** $p > 0$ if there exists a constant $\mathcal{C} > 0$, independent of h, such that the error between the derivative and its approximation is bounded by $\mathcal{C}h^p$.

Approximation error

Backward difference

Similarly to the previous forward difference approximation of f', we can define the **backward difference** approximation. For any $h > 0$, we have

$$f(x-h) = f(x) - f'(x)h + \mathcal{O}(h^2).$$

Also in this case, we can quantify the error

$$\mathcal{E}_h := \left| \frac{1}{h}\Big(f(x-h) - f(x)\Big) - f'(x) \right| \leq Ch, \quad \text{with} \quad C = \sup_{y \in (x, x+h)} \frac{|f''(y)|}{2}$$

Hence, the backward finite difference

$$\partial_h^- f(x) = \frac{1}{h}\Big(f(x) - f(x-h)\Big)$$

also approximates f' up to the order h.

Central difference

In order to improve the accuracy, we can consider the **central difference approximation** of f', by taking the points $x-h$ and $x+h$ into account.

Suppose that the function f is C^3 in the vicinity of x. By subtracting the two expressions

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f'''(\xi^+)}{6}h^3, \quad \xi^+ \in (x, x+h)$$
$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f'''(\xi^-)}{6}h^3, \quad \xi^+ \in (x-h, x)$$

we get

Central difference

$$\partial_h f(x) = \frac{1}{2h}\Big(f(x+h) - f(x-h)\Big) = f'(x) + \frac{f'''(\xi)}{3}h^2, \quad \xi \in (x-h, x+h).$$

Hence, we have the following bound on the approximation error:

$$\mathcal{E}_h := \left|\frac{1}{2h}\Big(f(x+h) - f(x-h)\Big) - f'(x)\right| \leq Ch^2, \quad \text{with} \quad C = \sup_{y\in(x-h,x+h)} \frac{|f'''(y)|}{3}$$

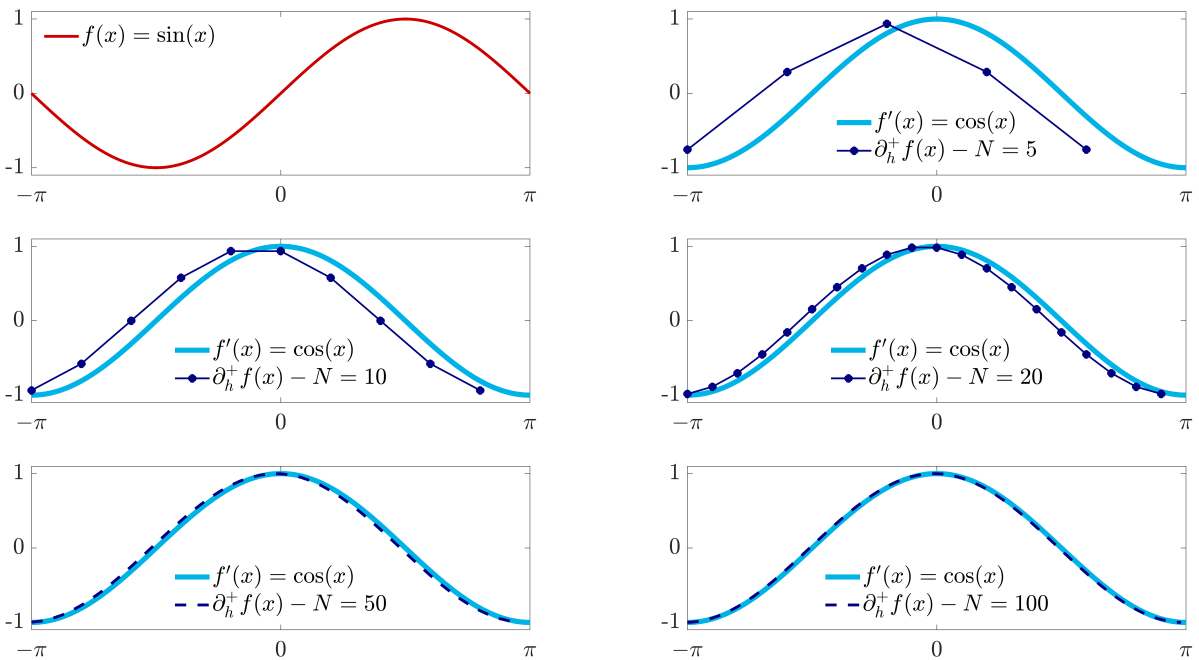
Approximation error

Remark

The order of the approximation is related to the regularity of the function f.

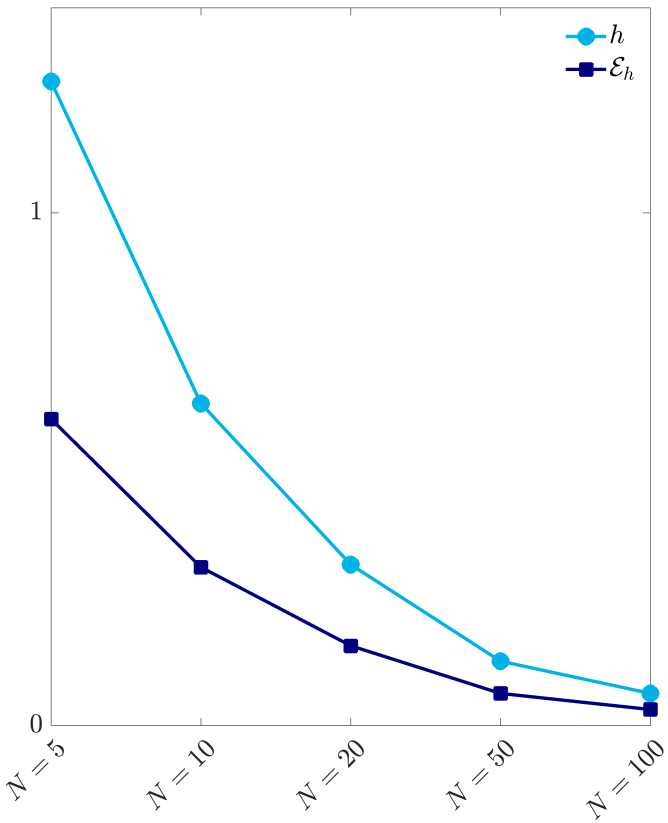
Example - forward finite difference for $f(x) = \sin(x)$

$$f(x) = \sin(x) \qquad f'(x) = \cos(x) \qquad \partial_h^+ = \frac{1}{h}\Big(f(x+h) - f(x)\Big)$$



Example - forward finite difference for $f(x) = \sin(x)$

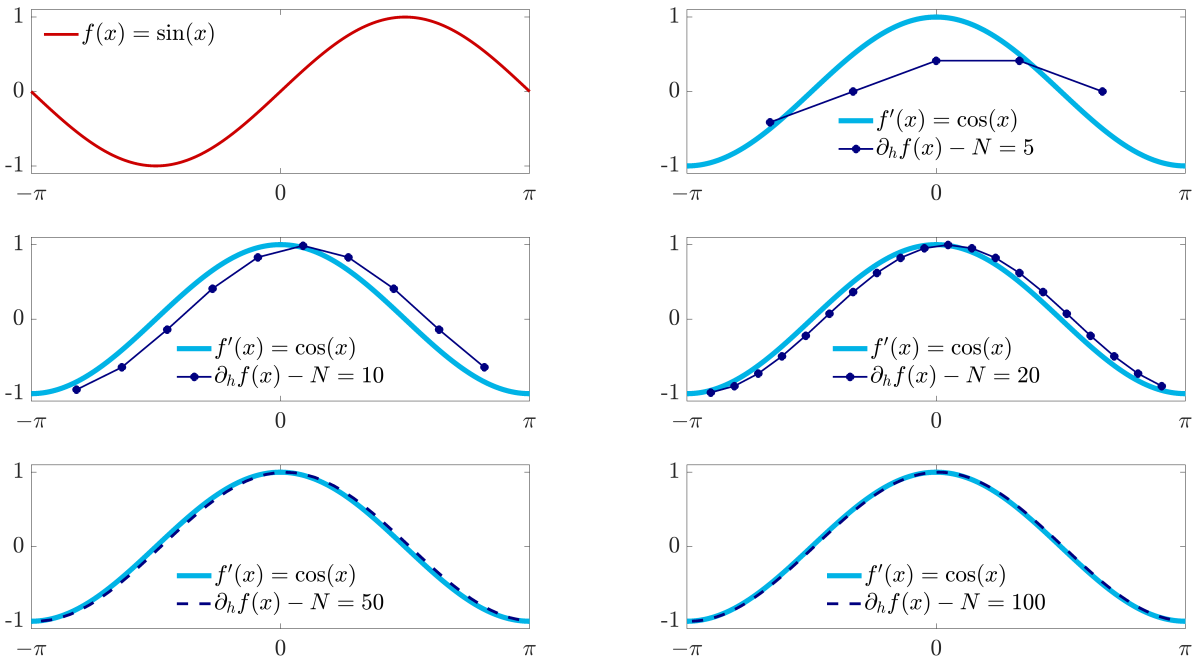
$$f(x) = \sin(x) \qquad f'(x) = \cos(x) \qquad \partial_h^+ = \frac{1}{h}\Big(f(x+h) - f(x)\Big)$$



h	$\mathcal{E}_h$
1.2566	0.5981
0.6283	0.3090
0.3142	0.1558
0.1257	0.0628
0.0628	0.0314

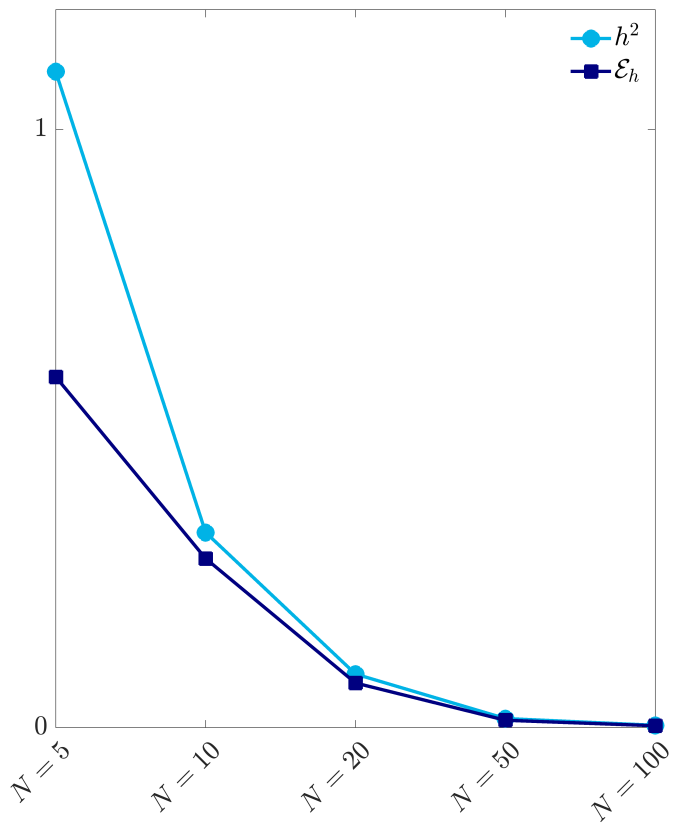
Example - central finite difference for $f(x) = \sin(x)$

$$f(x) = \sin(x) \qquad f'(x) = \cos(x) \qquad \partial_h = \frac{1}{2h}\Big(f(x+h) - f(x-h)\Big)$$



Example - central finite difference for $f(x) = \sin(x)$

$$f(x) = \sin(x) \qquad f'(x) = \cos(x) \qquad \partial_h = \frac{1}{2h}\Big(f(x+h) - f(x-h)\Big)$$



h^2	$\mathcal{E}_h$
1.0966	0.5865
0.3263	0.2827
0.0895	0.0746
0.0152	0.0123
0.0039	0.0031

Approximation of the second-order derivative

Secon-order finite difference

$$\partial_h^2 f(x) := \frac{1}{h^2}\Big(f(x+h) - 2f(x) + f(x-h)\Big)$$

Suppose that the function f is C^4 in the vicinity of x. By subtracting the two expressions

$$f(x+h) = f(x) + f'(x)h + \frac{f''(x)}{2}h^2 + \frac{f'''(x)}{6}h^3 + \frac{f^{(4)}(\xi^+)}{24}h^4, \quad \xi^+ \in (x, x+h)$$

$$f(x-h) = f(x) - f'(x)h + \frac{f''(x)}{2}h^2 - \frac{f'''(x)}{6}h^3 + \frac{f^{(4)}(\xi^-)}{24}h^4, \quad \xi^+ \in (x-h, x)$$

we get

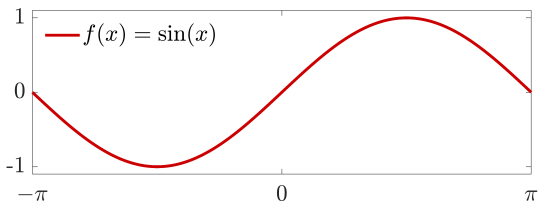
$$\frac{1}{h^2}\Big(f(x+h) - 2f(x) + f(x-h)\Big) = f''(x) + \frac{f^{(4)}(\xi)}{12}h^2, \quad \xi \in (x-h, x+h).$$

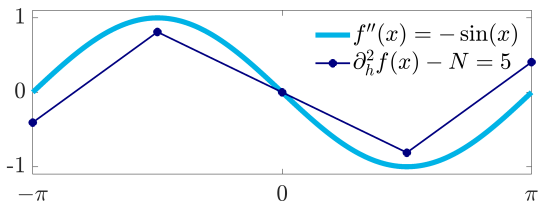
Hence, we have the following bound on the approximation error:

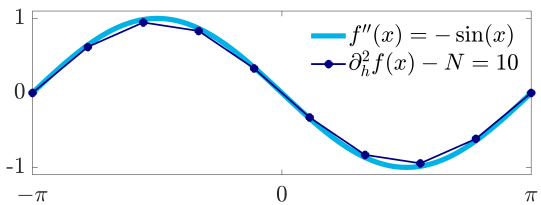
$$\mathcal{E}_h := \left|\frac{1}{h^2}\Big(f(x+h) - 2f(x) + f(x-h)\Big) - f''(x)\right| \leq Ch^2, \quad \text{with} \quad \mathcal{C} = \sup_{y\in(x-h,x+h)} \frac{|f^{(4)}(y)|}{12}$$

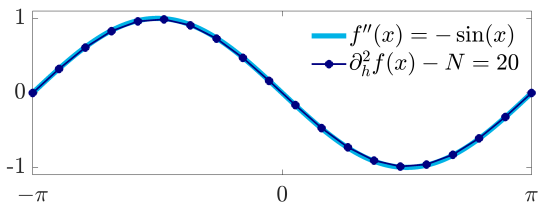
Example - second-order FD for $f(x) = \sin(x)$

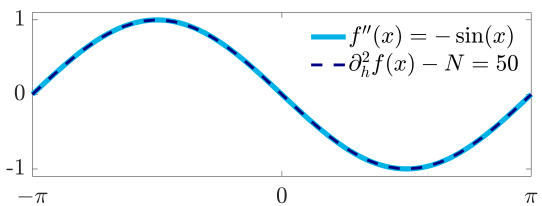
$$f(x) = \sin(x) \qquad f''(x) = -\sin(x) \qquad \partial_h^2 = \frac{1}{h^2}\Big(f(x+h) - 2f(x) + f(x-h)\Big)$$

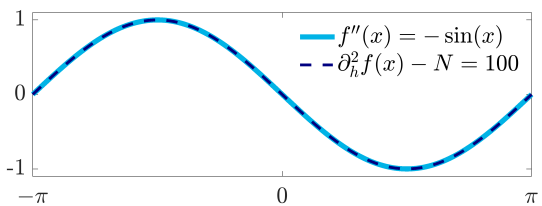






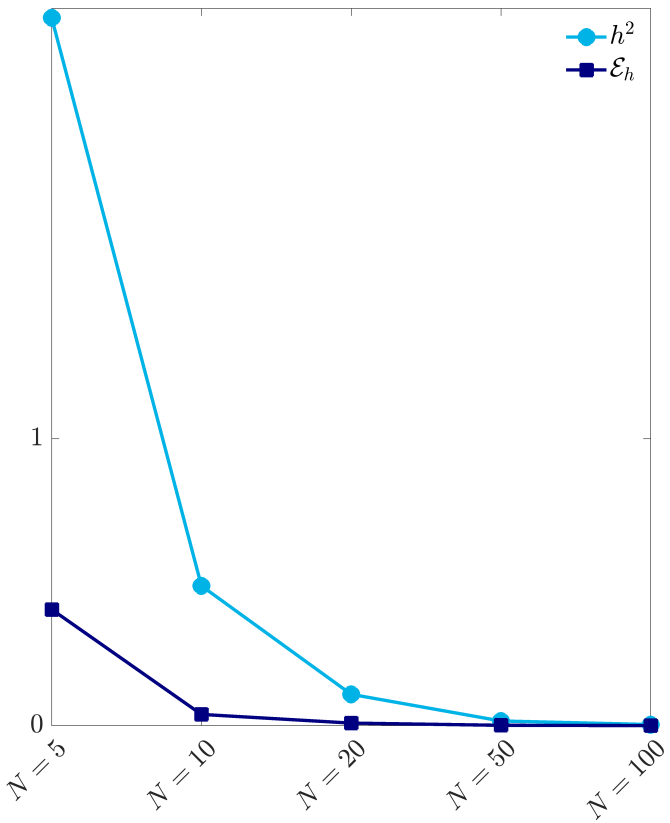






Example - second-order FD for $f(x) = \sin(x)$

$$f(x) = \sin(x) \qquad f''(x) = -\sin(x) \qquad \partial_h^2 = \frac{1}{h2}\Big(f(x+h) - 2f(x) + f(x-h)\Big)$$



h^2	$\mathcal{E}_h$
1.0966	0.4053
0.3263	0.0394
0.0895	0.0090
0.0152	0.0014
0.0039	0.0003

Numerical resolution of ODE

Numerical resolution of ODE

First order autonomous ODE

$$\begin{cases} y'(t) = f(y(t)), & t \in (0, T) \\ y(0) = y_0 \in \mathbb{R} \end{cases} \tag{1}$$

Theorem

Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous function. Then for any initial value $y_0 \in \mathbb{R}$, the ODE (1) admits a **local** solution $y : X \to \mathbb{R}$, where D is a neighborhood of $x : 0$ in $\mathbb{R}$. Moreover, if f is Lipschitz continuous, then the solution is unique.

Numerical resolution of ODE

What about the numerical resolution of (1)?

DIFFERENT APPROXIMATION METHODS:

- Forward (explicit) Euler approximation.
- Backward (implicit) Euler approximation.
- Runge-Kutta approximation.

Forward Euler method

Define a mesh $\{t_k\}_{k=0}^N$ on the interval $(0, T)$ such that

$$0 = t_0 < t_1 < \ldots < t_k < t_{k+1} < \ldots < t_N = T$$

and denote $y^k := y(t_k)$ for all $k \in \{0, \ldots, N\}$.

Forward Euler method

$$y^0 = y_0$$
$$y^{k+1} = y^k + (t_{k+1} - t_k) f(y^k), \quad \text{for all } k \in \{0, \ldots, N-1\}$$

The method is **explicit** because to compute y^{k+1} I only need information about y^k.

Backward Euler method

Define a mesh $\{t_k\}_{k=0}^N$ on the interval $(0, T)$ such that

$$0 = t_0 < t_1 < \ldots < t_k < t_{k+1} < \ldots < t_N = T$$

and denote $y^k := y(t_k)$ for all $k \in \{0, \ldots, N\}$.

Backward Euler method

$$y^0 = y_0$$
$$y^{k+1} = y^k + (t_{k+1} - t_k) f(y^{k+1}), \qquad \text{for all } k \in \{0, \ldots, N-1\}$$

The method is **implicit** because to compute y^{k+1} I need information about y^{k+1} itself.

Stability analysis

Two concepts of **stability**:

- Small perturbations in the initial datum produce small perturbations in the corresponding solution (also known as **continuous dependence on the initial datum**)

$$y_0 \mapsto y_0 + \delta \quad \Rightarrow \quad |y - y_\delta| < \varepsilon.$$

- The solution decays to zero as $t \to +\infty$ (requires the **global** existence of solutions)

$$\lim_{t \to +\infty} y(t) = 0.$$

Test problem

$$\begin{cases} y'(t) = \lambda y(t) & t \in (0, T),\ \lambda \in \mathbb{R} \\ y(0) = y_0 \end{cases} \tag{2}$$

SOLUTION: $y(t) = y_0 e^{\lambda t}$.

- If $\lambda < 0$, then the solution is stable.
- If $\lambda > 0$, then the solution is unstable.

We want this to be preserved by the numerical scheme

Stability analysis - forward Euler

Consider for simplicity a **uniform** partition $\{t_k\}_{k=0}^N$, i.e. $t_{k+1} - t_k = h$ for all $k \in \{0, \dots, N-1\}$.

Assume $\lambda < 0$ and approximate (2) by the **forward Euler method**.

$$y^0 = y_0$$
$$y^{k+1} = y^k + hf(y^k) = y^k + h\lambda y^k = (1 + h\lambda)y^k \quad \Rightarrow \quad y^k = (1 + h\lambda)^k y^0.$$

In order to ensure stability, we then need that $|1 + h\lambda| < 1$. For real $\lambda < 0$ this is equivalent to

$$-2 < h\lambda < 0 \quad \Rightarrow \quad h < -\frac{2}{\lambda}.$$

Conditional stability

The forward Euler method is **conditionally stable**, i.e., the step size has to be chosen sufficiently small to ensure stability.

Stability analysis - backward Euler

Consider for simplicity a **uniform** partition $\{t_k\}_{k=0}^N$, i.e. $t_{k+1} - t_k = h$ for all $k \in \{0, \dots, N-1\}$.

Assume $\lambda < 0$ and approximate (2) by the **backward Euler method**.

$$y^0 = y_0$$

$$y^{k+1} = y^k + h\lambda y^{k+1} \quad \Rightarrow \quad y^{k+1} = \frac{1}{1-h\lambda} y^k \quad \Rightarrow \quad y^k = \left(\frac{1}{1-h\lambda}\right)^k y^0.$$

In order to ensure stability, we then need that $|1-h\lambda|^{-1} < 1 \Rightarrow |1-h\lambda| > 1$. For real $\lambda < 0$ this is equivalent to

$$h\lambda < 0 \quad \text{or} \quad h\lambda > 2.$$

Conditional stability

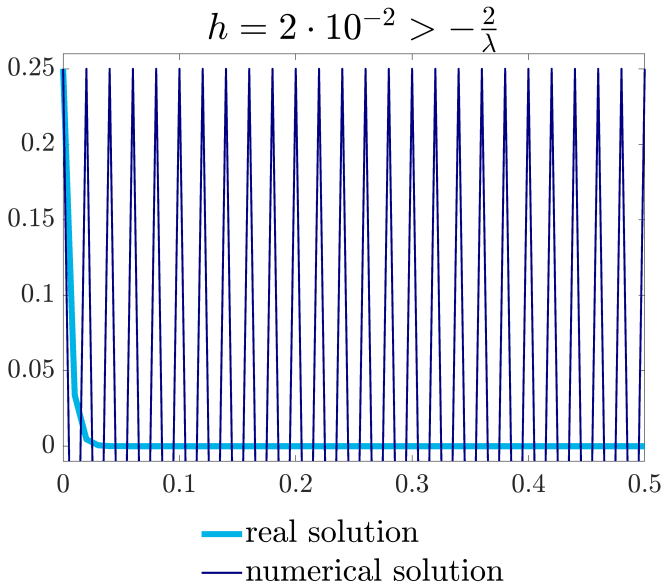
Since $\lambda < 0$ and $h > 0$, we always have $h\lambda < 0$.

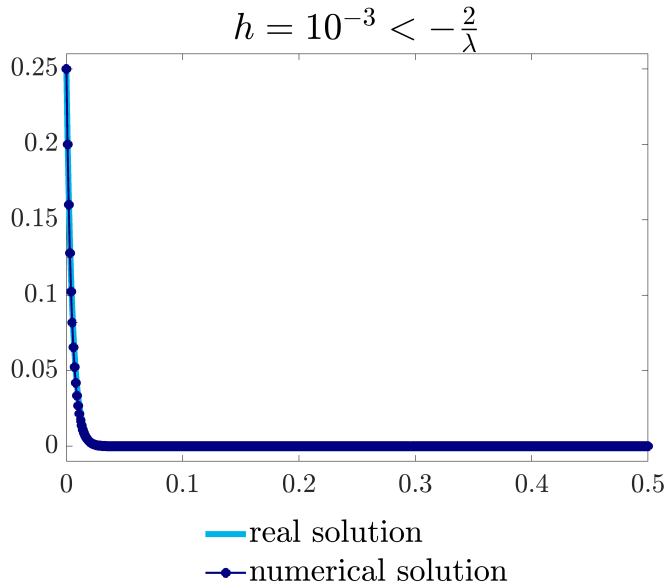
The forward Euler method is **unconditionally stable**, i.e. it is stable no matter the selection of the step size.

Numerical example - forward Euler

Consider (2) with $y_0 = 0.5$, $T = 0.5$ and $\lambda = -200$. We approximate the solution with the **forward Euler method** on two meshes of size

- $h_1 = 2 \cdot 10^{-2} > -\frac{2}{\lambda}$
- $h_2 = 10^{-3} < -\frac{2}{\lambda}$



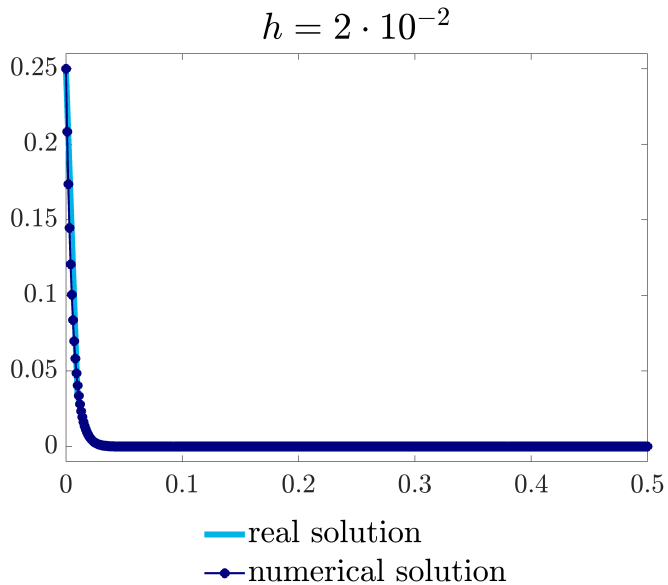


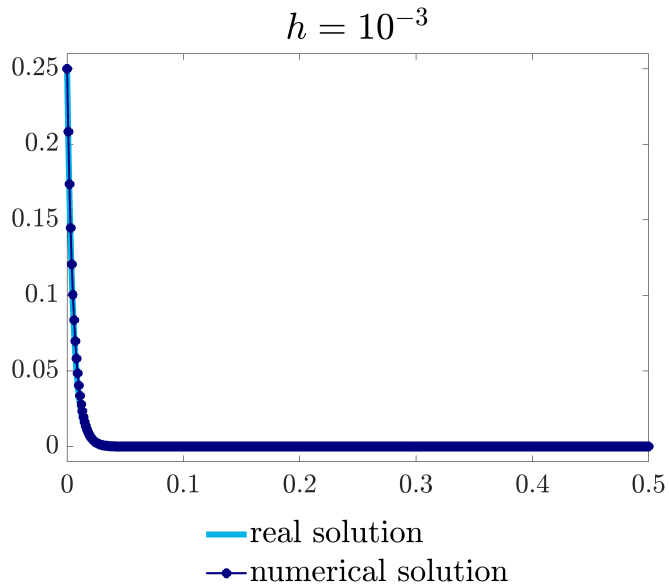
On the coarser mesh, the method is unstable and fails in approximating the real solution of the ODE.

Numerical example - backward Euler

Consider (2) with $y_0 = 0.5$, $T = 0.5$ and $\lambda = -200$. We approximate the solution with the **backward Euler method** on two meshes of size

- $h_1 = 2 \cdot 10^{-2} > -\frac{2}{\lambda}$
- $h_2 = 10^{-3} < -\frac{2}{\lambda}$





On both meshes, the method is stable and correctly approximates the real solution of the ODE.

Numerical resolution of PDE: the finite difference method

The Poisson equation

$$\begin{cases} -u''(x) + u(x) = f(x), & x \in (a,b) \\ u(a) = \alpha \in \mathbb{R},\ u(b) = \beta \in \mathbb{R} \end{cases} \tag{3}$$

Theorem (Schauder)

If $f \in C^m([a,b])$ for $m \geq 0$, there exists a unique **strong** solution $u \in C^{m+2}(a,b)$ of (3).

H. Brezis, *Functional analysis, Sobolev spaces and Partial Differential Equations*, Springer, 2010.

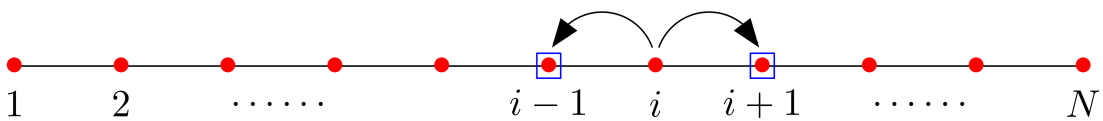
Finite difference approximation

Uniform mesh of the space domain (a, b): $\{x_j\}_{j=0}^{N+1}$

- $N \in \mathbb{N}$.
- $x_j = a + jh$
- $h = (b - a)/(N + 1)$
- $u(x_0) = \alpha$ and $u(x_{N+1}) = \beta$.
- $u_j := u(x_j)$ for $j \in \{0, \ldots, N + 1\}$.
- $\mathbf{u} = (u_1, \ldots, u_N) \in \mathbb{R}^N$ unknown vector.

Finite difference approximation

$$\begin{cases} -\dfrac{1}{h^2}\left(u_{j-1} - 2u_j + u_{j+1}\right) + u_j = f(x_j), \quad j \in \{1, \ldots, N\} \\ u_0 = \alpha, \quad u_{N+1} = \beta \end{cases} \tag{4}$$



Finite difference approximation - matrix form

Matrix form

Problem (4) can be written in matrix form

$$\underbrace{(A_h + I)}_{A_{FD}}\mathbf{u} = \mathbf{f}$$

$$A_h = \frac{1}{h^2}\begin{pmatrix} -2 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & -2 \end{pmatrix} \qquad \mathbf{f} = \begin{pmatrix} f(x_1) - \frac{\alpha}{h^2} \\ f(x_2) \\ \vdots \\ f(x_N) - \frac{\beta}{h^2} \end{pmatrix}$$

Existence and uniqueness of solutions

Proposition

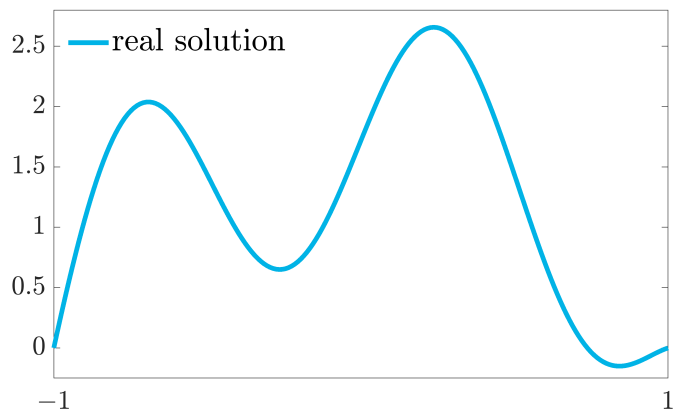
The matrix $A_{FD} \in \mathbb{R}^{N \times N}$ is invertible, since it is a Toeplitz matrix.

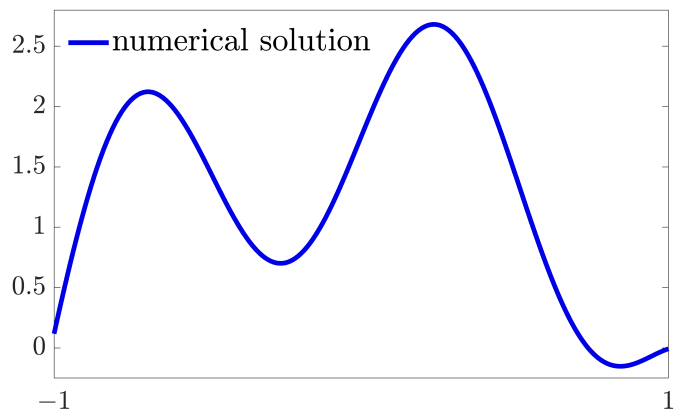
Since A_{FD} is invertible, we have $\mathbf{u} = A_{FD}^{-1}\mathbf{f}$.

Example

$$\begin{cases} -u''(x) + u(x) = (1 + 4\pi^2)\sin(2\pi x) + (1 + 4x^2)e^{x^2} + e, & x \in (-1, 1) \\ u(-1) = u(1) = 0 \end{cases}$$

SOLUTION: $u(x) = \sin(2\pi x) - e^{x^2} + e$.





Two-dimensional problems

$$\begin{cases} -\Delta u(x) + u(x) = f(x), & x \in \Omega \\ u = 0 & x \in \partial\Omega \end{cases} \tag{5}$$

The finite difference method allows approximating the solution of (5) in **simple geometries** for the domain Ω (for instance, $\Omega = [a, b]^2$).

Finite difference approximation

Uniform mesh of the space domain $[a_1, a_2] \times [b_1, b_2]$: $\{(x_j, y_j)\}_{j=0}^{N+1}$

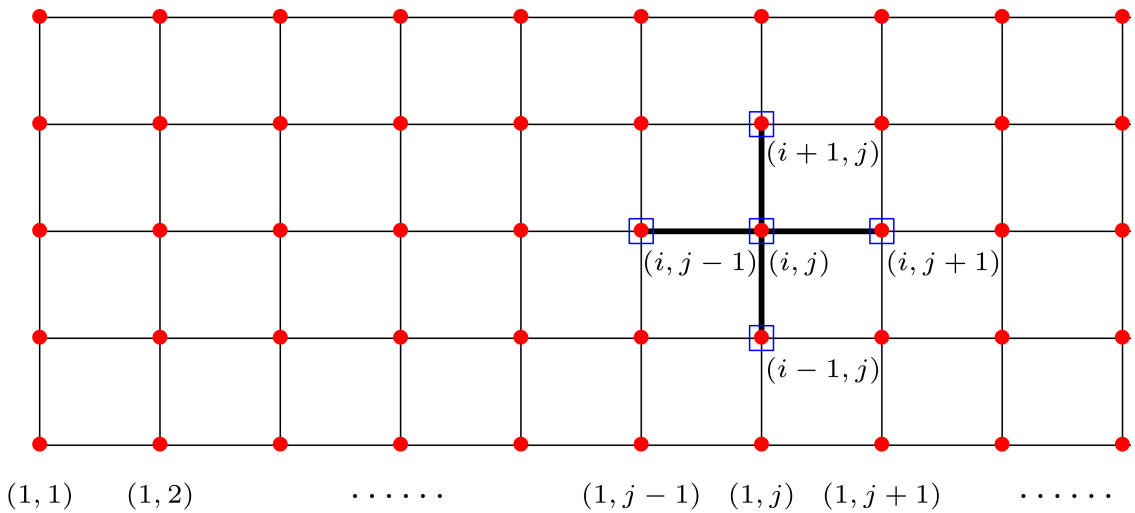
- $N \in \mathbb{N}$.
- $x_j = y_j = a + jh$
- $h = (b - a)/(N + 1)$
- $u_j := u(x_j, y_j)$ for $j \in \{0, \ldots, N + 1\}$.
- $\mathbf{u}_j = (u_{i,j})_{i=1}^{N} \in \mathbb{R}^N$ for all $j \in \{1, \ldots, N\}$.
- $\mathbf{U} = (\mathbf{u}_j)_{j=1}^{N} \in \mathbb{R}^{N^2}$.

Finite difference approximation

$$\begin{cases} -\dfrac{u_{i-1,j} + u_{i+1,j} - 4u_{i,j} + u_{i,j-1} + u_{i,j+1}}{h^2} + u_{i,j} = f(x_i, y_j), \quad (i,j) \in \{1, \ldots, N\}^2 \\ u_{i,0} = u_{i,N+1} = 0 \; i \in 0, \ldots, N + 1 \\ u_{0,j} = u_{N+1,j} = 0, \; j \in 0, \ldots, N + 1 \end{cases}$$

Finite difference approximation

Two-dimensional uniform mesh on a domain $\Omega = [a_1, a_2] \times [b_1, b_2]$



Finite difference approximation - matrix form

Matrix form

$$\underbrace{(A_h + I)}_{A_{FD}}\mathbf{U} = \mathbf{F}$$

$$A_h = \frac{1}{h^2}\begin{pmatrix} P & I & 0 & \cdots & \cdots & 0 \\ I & P & I & 0 & \cdots & 0 \\ 0 & I & P & I & 0 & 0 \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & I & P & I \\ 0 & \cdots & \cdots & 0 & I & P \end{pmatrix} \qquad \mathbf{F} = \begin{pmatrix} \mathbf{f}_1 \\ \mathbf{f}_2 \\ \vdots \\ \mathbf{f}_N \end{pmatrix}$$

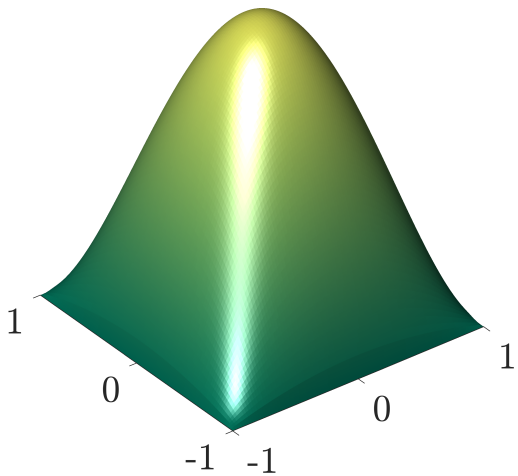
$$P = \begin{pmatrix} -4 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & -4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -4 & 1 & 0 & 0 \\ \vdots & & \ddots & \ddots & & \vdots \\ 0 & \cdots & 0 & 1 & -4 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & -4 \end{pmatrix}$$

Example

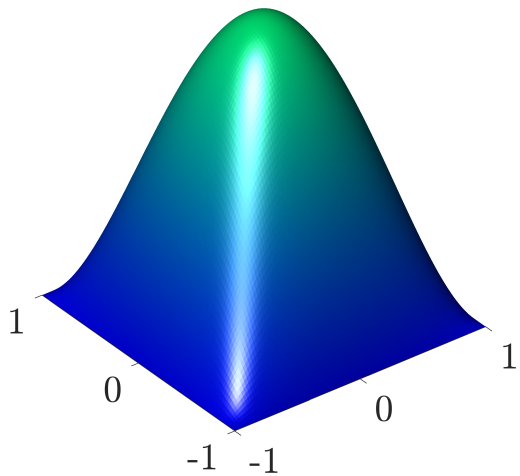
$$\begin{cases} -\Delta u(x,y) + u(x,y) = (1-x^2)(1-y^2) + 2(2-x^2-y^2), & x \in (-1,1)^2 \\ u(x,-1) = u(x,1) = 0 & x \in (-1,1) \\ u(-1,y) = u(1,y) = 0 & y \in (-1,1) \end{cases}$$

SOLUTION: $u(x,y) = (1-x^2)(1-y^2)$.

Numerical solution



Real solution



Numerical resolution of PDE: the finite element method

The Poisson equation

$$\begin{cases} -\Delta u(x) + u(x) = f(x), & x \in \Omega \\ u = 0 & x \in \partial\Omega \end{cases} \tag{6}$$

Theorem

If $\Omega \in C^1$ and $f \in H^{-1}(\Omega)$, there exists a unique **weak** solution $u \in H_0^1(\Omega)$ of (6), characterized through the variational formulation

$$\underbrace{\int_\Omega \nabla u(x) \cdot \nabla v(x)\, dx + \int_\Omega u(x)v(x)\, dx}_{\text{bilinear form } a(u,v)} = \underbrace{\int_\Omega f(x)v(x)\, dx}_{\text{linear functional } F(v)}, \quad \text{for all } v \in H_0^1(\Omega). \tag{7}$$

The finite element method

The finite element (FE) method consists in approximating the variational formulation (7) characterizing the weak solution of (6).

INGREDIENTS:

- A mesh $\mathfrak{M}_h$ on the domain Ω composed by $N \in \mathbb{N}^*$ **elements**.
- A finite-dimensional space V_h with $dim(V_h) = N$ approximating $H_0^1(\Omega)$.
- A basis $(\phi_i)_{i=1}^N$ for V_h.

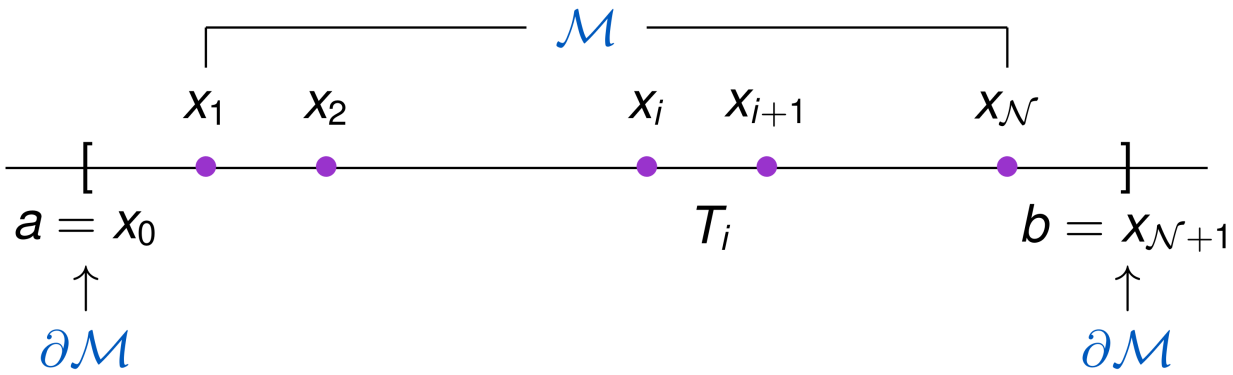
The mesh

Let $\mathcal{M} = \{T_i\}_{i=1}^{\mathcal{N}}$ be a partition of the domain Ω (i.e. $\overline{\Omega} = \bigcup_{i=1}^{\mathcal{N}} T_i$)

- In dimension $N = 1$, we consider a uniform partition of $\Omega = (a, b)$

$$a = x_0 < x_1 < \ldots < x_i < x_{i+1} < \ldots < x_{\mathcal{N}+1} = b,$$

with $x_{i+1} = x_i + h$, $i = 0, \ldots \mathcal{N}$, and we denote $T_i := [x_i, x_{i+1}]$. Moreover, we indicate with $\partial\mathfrak{M} = \{x_0, x_{\mathcal{N}+1}\}$ the boundary nodes.



The mesh

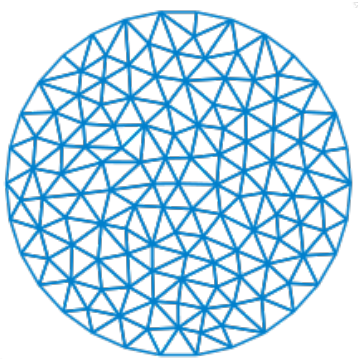
Let $\mathcal{M} = \{T_i\}_{i=1}^{\mathcal{N}}$ be a partition of the domain Ω (i.e. $\overline{\Omega} = \bigcup_{i=1}^{\mathcal{N}} T_i$)

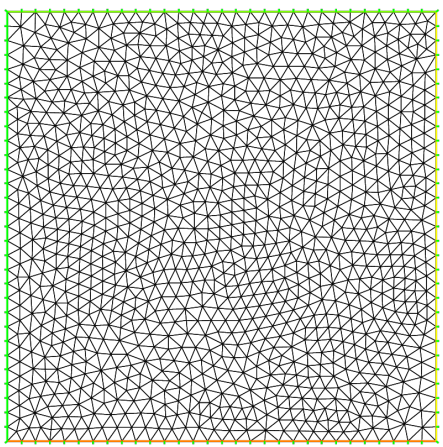
- In dimension $N = 2$, we consider triangular elements T_i, $i = 1, \dots, \mathcal{N}$. We indicate with h_i the diameter of the element T_i and with ρ_i its inner radius, i.e. the diameter of the largest ball contained in T_i. We define

$$h = \max_{i \in \{1,\dots,\mathcal{N}\}} h_i.$$

  Moreover, we require that the triangulation satisfies the **regularity** and **local uniformity** conditions:

  there exists $\sigma > 0$ s.t. $h_i \leq \sigma \rho_i$ for all $i = 1, \dots, \mathcal{N}$,
  there exists $\kappa > 0$ s.t. $h_i \leq \kappa h_j$ for all $i, j = 1, \dots, \mathcal{N}$, $T_i \cap T_j = \emptyset$.





The finite dimensional space

$$V_h := \left\{ v \in H_0^1(\Omega) \ : \ v|_{T_i} \in \mathcal{P}^r \right\},$$

where $\mathcal{P}_r$ denotes the space of polynomial of degree r.

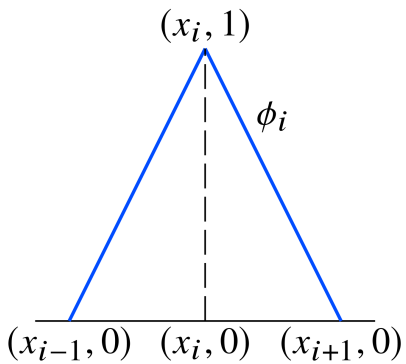
- r is usually known as the **degree** of the FE approximation.
- The order of convergence of the FE method is determined by r and the regularity of the solution to the continuous problem.

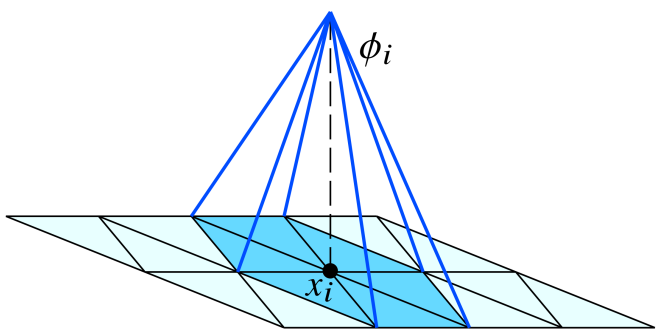
r	$u \in H^1(a,b)$	$u \in H^2(a,b)$	$u \in H^3(a,b)$	$u \in H^4(a,b)$	$u \in H^5(a,b)$
1	converges	$\mathbf{h}$	h	h	h
2	converges	h	$\mathbf{h}^2$	h^2	h^2
3	converges	h	h^2	$\mathbf{h}^3$	h^3
4	converges	h	h^2	h^3	$\mathbf{h}^4$

The basis functions - $p = 1$

- Dimension $N = 1$: $\mathcal{P}^1 = \left\{\phi(x) = a + bx, \text{ with } a, b \in \mathbb{R}\right\}$.
- Dimension $N = 2$: $\mathcal{P}^1 = \left\{\phi(x, y) = a + bx + cy, \text{ with } a, b, c \in \mathbb{R}\right\}$.

The typical choice for basis functions $\{\phi_i\}_{i=1}^{\mathcal{N}}$ has to fulfill $\phi_i(x_j) = \delta_{i,j}$, for all $i,j \in \{1, \ldots, \mathcal{N}\}$.





The method

STEP 1: by decomposing

$$u(x) = \sum_{j=1}^{\mathcal{N}} u_j \phi_j(x), \quad f(x) = \sum_{j=1}^{\mathcal{N}} f_j \phi_j(x), \quad f_j = \int_\Omega f(x)\phi_j(x)\,dx$$

and taking $v = \phi_i$, from (7) we obtain the linear system $A_h \mathbf{u} = M_h \mathbf{f}$, where

- $\mathbf{u} = (u_1, \ldots, u_{\mathcal{N}}) \in \mathbb{R}^{\mathcal{N}}$ is an unknown vector.
- $\mathbf{f} = (f_1, \ldots, f_{\mathcal{N}}) \in \mathbb{R}^{\mathcal{N}}$.
- The **stiffness** matrix $A_h \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ has components

$$a_{i,j} = \int_\Omega \nabla\phi_i(x) \cdot \nabla\phi_j(x)\,dx, \quad \text{for all } i,j \in \{1,\ldots,\mathcal{N}\}. \tag{8}$$

- The **mass** matrix $M_h \in \mathbb{R}^{\mathcal{N}\times\mathcal{N}}$ has components

$$m_{i,j} = \int_\Omega \phi_i(x)\phi_j(x)\,dx, \quad \text{for all } i,j \in \{1,\ldots,\mathcal{N}\}. \tag{9}$$

STEP 2: the unknown vector $\mathbf{u}$ is obtained by solving $\mathbf{u} = A_h^{-1} M_h \mathbf{f}$.

STEP 3: once the vector $\mathbf{u}$ is known, the solution of the Poisson equation is approximated by

$$u(x) = \sum_{j=1}^{\mathcal{N}} u_j \phi_j(x).$$

Remark

By construction, the basis functions $\{\phi_i\}_{i=1}^{\mathcal{N}}$ are such that $supp(\phi_i) \cap supp(\phi_j) \neq \varnothing$ if and only if $j \in \{i-1, i, i+1\}$. Consequently, the stiffness and mass matrices are both tri-diagonal.

SPECIAL CASE: space dimension $N = 1$. We can compute explicitly

$$A_h = \frac{1}{h}\begin{pmatrix} -2 & 1 & 0 & \cdots & \cdots & 0 \\ 1 & -2 & 1 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & -2 & 1 \\ 0 & \cdots & \cdots & 0 & 1 & -2 \end{pmatrix}$$

$$M_h = h\begin{pmatrix} 2/3 & 1/6 & 0 & \cdots & \cdots & 0 \\ 1/6 & 2/3 & 1/6 & 0 & \cdots & 0 \\ 0 & 1/6 & 2/3 & 1/6 & 0 & 0 \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1/6 & 2/3 & 1/6 \\ 0 & \cdots & \cdots & 0 & 1/6 & 2/3 \end{pmatrix}$$

Finite Element VS Finite Difference

ADVANTAGES OF FINITE ELEMENT W.R.T. FINITE DIFFERENCE:

- Working with the variational formulation, FE allows to approximate weak solution and do not requires high regularity to converge.
- FE is applicable to complex problems with complex geometries.

ADVANTAGES OF FINITE ELEMENT W.R.T. FINITE DIFFERENCE:

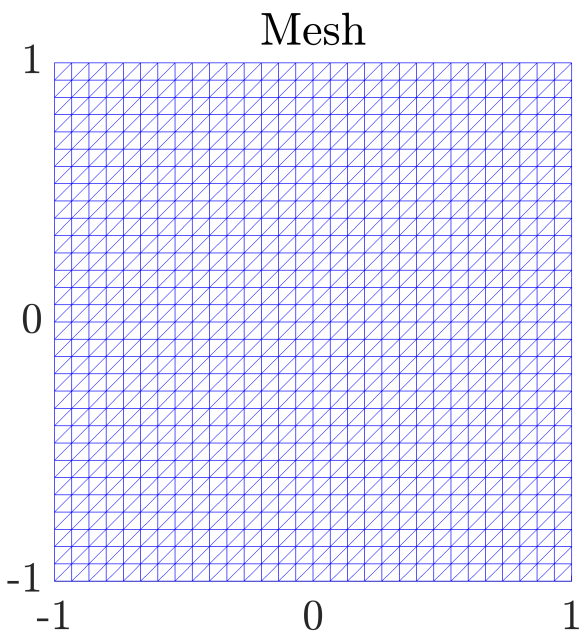
- In simple geometries (namely, rectangles), FD is typically easier to implement than FE.

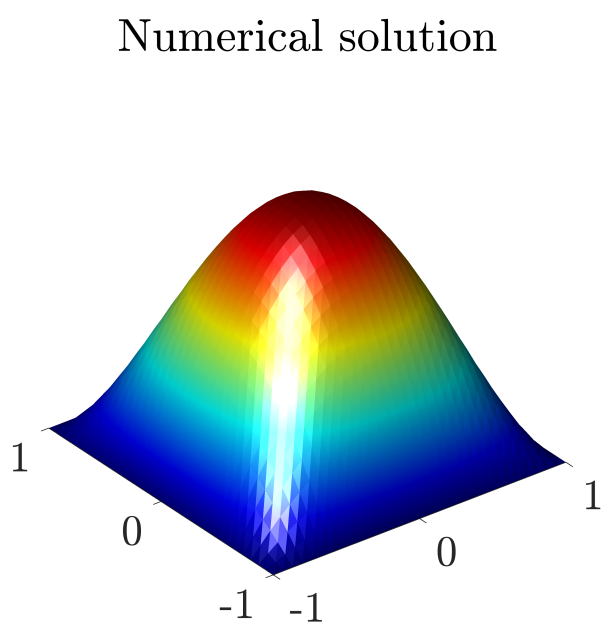
Due to the success that FE methods have had since their introduction, many computational tools for the FE approximation of PDE have been developed and perfected in the last decades.

- PDE toolbox of Matlab.
- Fenics.
- Free Fem.

$$\begin{cases} -\Delta u(x,y) + u(x,y) = (1-x^2)(1-y^2) + 2(2-x^2-y^2), & x \in (-1,1)^2 \\ u(x,-1) = u(x,1) = 0 & x \in (-1,1) \\ u(-1,y) = u(1,y) = 0 & y \in (-1,1) \end{cases}$$

SOLUTION: $u(x,y) = (1-x^2)(1-y^2)$.





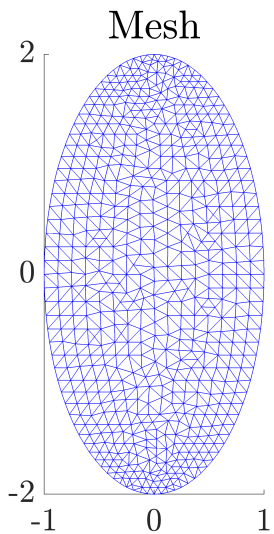
Free FEM code

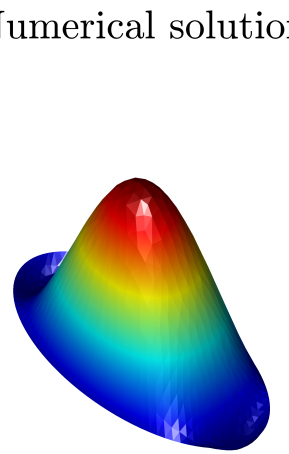
```
include "ffmatlib.idp"                                  // Library connecting
                                                        // FreeFEM to Matlab

// Mesh
real x0 = -1, x1 = 1, y0 = -1, y1 = 1;                  // Vertices of the square
int M = 30;                                             // Number of partition points
                                                        //for each edge
mesh Th = square(M,M,[x0+(x1-x0)*x,y0+(y1-y0)*y]);

fespace Vh(Th, P1);                                     // The FE space defined over Th
Vh u, v;

func f= (1-x^2)*(1-y^2) + 2*(2-x^2-y^2);                // Right-hand side function

// Define and solve the PDE
solve Poisson(u,v) = int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v) + u*v)  // Bilinear form
                   - int2d(Th)(f*v)   // Right-hand side
                   + on(1,2,3,4,u=0); // Boundary condition

// Save
Vh Ex, Ey;
Ex = -dx(u);
Ey = -dy(u);
savemesh(Th, "Th_circle_mesh.msh");
ffSaveVh(Th,Vh,"Poisson_vh.txt");
ffSaveData3(u,Ex,Ey,"solutionPoisson.txt");
```

Free FEM code

In Free FEM, we can easily change the geometry of our domain

```
// Mesh
real a = 2; //The length of the semimajor axis
real b = 1; //The length of the semiminor axis
border Gamma(t=0., 2*pi){x=a*cos(t); y=b*sin(t);}
mesh Th = buildmesh(Gamma(100));
...
// Define and solve the PDE
solve Poisson(u,v) = int2d(Th)(dx(u)*dx(v) + dy(u)*dy(v) + u*v)  // Bilinear form
                   - int2d(Th)(f*v)   // Right-hand side
                + on(Gamma,u=0); // Boundary condition
```

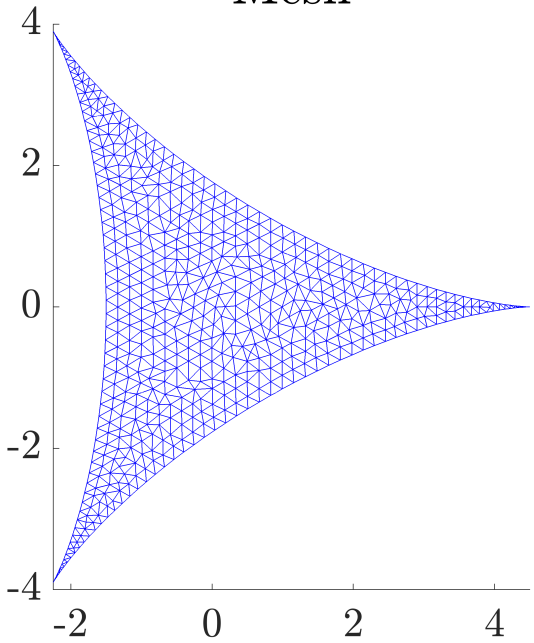




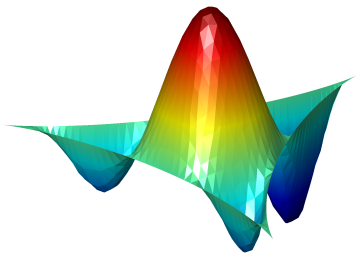
In Free FEM, we can easily change the geometry of our domain

POISSON EQUATION ON A DELTOID: $\begin{cases} x = 2\cos(t) + \cos(2t) \\ y = 2\sin(t) - \sin(2t) \end{cases}$

Mesh



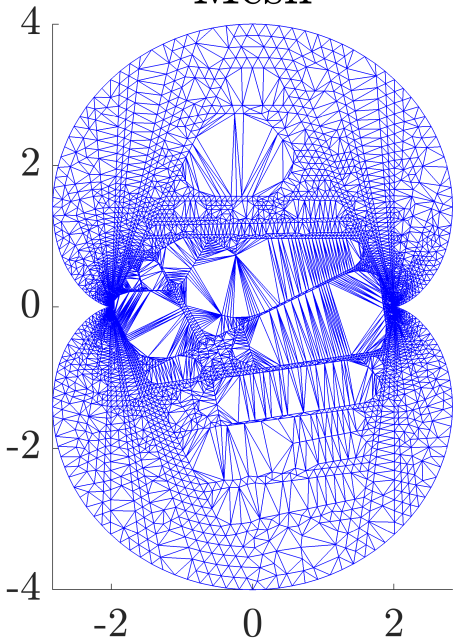
Numerical solution



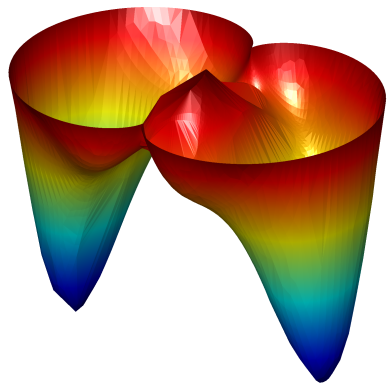
In Free FEM, we can easily change the geometry of our domain

POISSON EQUATION ON A NEFROID: $\begin{cases} x = 3\cos(t) - \cos(3t) \\ y = 4\sin^3(t) \end{cases}$

Mesh



Numerical solution



Free FEM

In Free FEM, we can easily solve complex problems

Free FEM webpage: `freefem.org`

Large collection of tutorials for solving different types of PDE including

- Poisson equation with non-Dirichlet boundary conditions on any type of domain.
- System of elasticity.
- Stokes equations for fluids.
- Maxwell's equations.
- p-Laplace equation.

Time-dependent PDE

Time-dependent PDE

$$\begin{cases} u_t(x,t) - Au(x,t) = f(x,t) & (x,t) \in \Omega \times (0,T) \\ u(x,t) = 0 & (x,t) \in \partial\Omega \times (0,T) \\ u(x,0) = u_0(x) & x \in \Omega \end{cases}$$

Time-dependent PDE

To approximate numerically a time-dependent PDE we shall proceed in two steps.

STEP 1: discretize in space - Finite Difference

$$\begin{cases} \mathbf{u}_t(t) - A_h\mathbf{u}(t) = \mathbf{f}(t), & t \in (0,T) \\ \mathbf{u}(0) = \mathbf{u}_0 \end{cases}$$

$$\begin{aligned} \mathbf{u}(t) &= (u_i(t))_{i=1}^N & u_i(t) &= u(x_i,t) \\ \mathbf{u}_0 &= (u_{0,i})_{i=1}^N & u_{0,i} &= u_0(x_i) \\ \mathbf{f}(t) &= (f_i(t))_{i=1}^N & f_i(t) &= f(x_i,t) \end{aligned}$$

STEP 2: discretize in time

FORWARD EULER:

$$\begin{cases} \frac{1}{\Delta t}\left(\mathbf{u}^{j+1} - \mathbf{u}^j\right) - A_h\mathbf{u}^j = \mathbf{f}^j \\ \mathbf{u}^0 = \mathbf{u}_0 \end{cases}$$

$$\begin{aligned} \mathbf{u}^j &= \left((u_i^j)_{i=1}^N\right)_{j=1}^M & u_i^j &= u(x_i,t_j) \\ \mathbf{u}_0 &= (u_{0,i})_{i=1}^N & u_{0,i} &= u_0(x_i) \\ \mathbf{f}^j &= \left((f_i^j)_{i=1}^N\right)_{j=1}^M & f_i^j &= f(x_i,t_j) \end{aligned}$$

BACKWARD EULER:

$$\begin{cases} \frac{1}{\Delta t}\left(\mathbf{u}^{j+1} - \mathbf{u}^j\right) - A_h\mathbf{u}^{j+1} = \mathbf{f}^{j+1} \\ \mathbf{u}^0 = \mathbf{u}_0 \end{cases}$$

$$\begin{aligned} \mathbf{u}^j &= \left((u_i^j)_{i=1}^N\right)_{j=1}^M & u_i^j &= u(x_i,t_j) \\ \mathbf{u}_0 &= (u_{0,i})_{i=1}^N & u_{0,i} &= u_0(x_i) \\ \mathbf{f}^j &= \left((f_i^j)_{i=1}^N\right)_{j=1}^M & f_i^j &= f(x_i,t_j) \end{aligned}$$

Time-dependent PDE

To approximate numerically a time-dependent PDE we shall proceed in two steps.

STEP 1: discretize in space - Finite Element

$$\begin{cases} M_h \mathbf{u}_t(t) - A_h \mathbf{u}(t) = M_h \mathbf{f}(t), & t \in (0, T) \\ \mathbf{u}(0) = \mathbf{u}_0 \end{cases}$$

$$\begin{aligned} \mathbf{u}(t) &= (u_i(t))_{i=1}^N & u_i(t) &= u(x_i, t) \\ \mathbf{u}_0 &= (u_{0,i})_{i=1}^N & u_{0,i} &= u_0(x_i) \\ \mathbf{f}(t) &= (f_i(t))_{i=1}^N & f_i(t) &= f(x_i, t) \end{aligned}$$

STEP 2: discretize in time

FORWARD EULER:

$$\begin{cases} \frac{1}{\Delta t} M_h \left(\mathbf{u}^{j+1} - \mathbf{u}^j \right) - A_h \mathbf{u}^j = M_h \mathbf{f}^j \\ \mathbf{u}^0 = \mathbf{u}_0 \end{cases}$$

$$\begin{aligned} \mathbf{u}^j &= \left((u_i^j)_{i=1}^N \right)_{j=1}^M & u_i^j &= u(x_i, t_j) \\ \mathbf{u}_0 &= (u_{0,i})_{i=1}^N & u_{0,i} &= u_0(x_i) \\ \mathbf{f}^j &= \left((f_i^j)_{i=1}^N \right)_{j=1}^M & f_i^j &= f(x_i, t_j) \end{aligned}$$

BACKWARD EULER:

$$\begin{cases} \frac{1}{\Delta t} M_h \left(\mathbf{u}^{j+1} - \mathbf{u}^j \right) - A_h \mathbf{u}^{j+1} = M_h \mathbf{f}^{j+1} \\ \mathbf{u}^0 = \mathbf{u}_0 \end{cases}$$

$$\begin{aligned} \mathbf{u}^j &= \left((u_i^j)_{i=1}^N \right)_{j=1}^M & u_i^j &= u(x_i, t_j) \\ \mathbf{u}_0 &= (u_{0,i})_{i=1}^N & u_{0,i} &= u_0(x_i) \\ \mathbf{f}^j &= \left((f_i^j)_{i=1}^N \right)_{j=1}^M & f_i^j &= f(x_i, t_j) \end{aligned}$$

Matrix form

FINITE DIFFERENCE + FORWARD EULER: $\mathbf{u}^{j+1} = \left(I + \Delta_t A_h\right)\mathbf{u}^j + \Delta_t \mathbf{f}^j$

FINITE DIFFERENCE + BACKWARD EULER: $\mathbf{u}^{j+1} = \left(I - \Delta_t A_h\right)^{-1}\left(\mathbf{u}^j + \Delta_t \mathbf{f}^j\right)$

FINITE ELEMENT + FORWARD EULER: $\mathbf{u}^{j+1} = \left(I + \Delta_t M_h^{-1} A_h\right)\mathbf{u}^j + \Delta_t \mathbf{f}^j$

FINITE ELEMENT + BACKWARD EULER: $\mathbf{u}^{j+1} = \left(I - \Delta_t M_h^{-1} A_h\right)^{-1}\left(\mathbf{u}^j + \Delta_t \mathbf{f}^j\right)$

WARNING

Depending on the numerical scheme we select for the time discretization, we have to pay attention to the choice of the time-step Δ_t.

The CFL condition

$$\begin{cases} u_t(x,t) - u_{xx}(x,t) = 0 & (x,t) \in (-1,1) \times (0,T) \\ u(0,t) = u(1,t) = 0 & t \in (0,T) \\ u(x,0) = u_0(x) & x \in (0,1) \end{cases}$$

FINITE DIFFERENCE APPROXIMATION OF $-\Delta$: $A_h = \frac{1}{(\Delta_x)^2}\text{tridiag}\Big(1,-2,1\Big)$

FORWARD EULER METHOD: it is conditionally stable, meaning that we need a small enough time-step to correctly approximate the solution of the equation. In the case of the heat equation, we need

Courant-Friedrichs-Lewy (CFL) condition

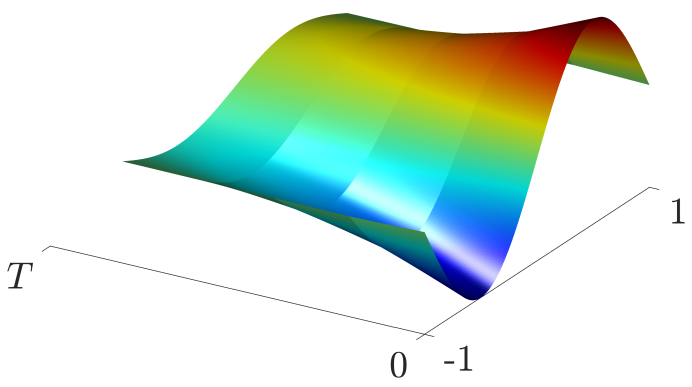
$$\Delta_t = \mu(\Delta_x)^2, \ \mu \in (0,1)$$

BACKWARD EULER METHOD: it is unconditionally stable, meaning that no matter the time-step we can always approximate the solution of the equation.

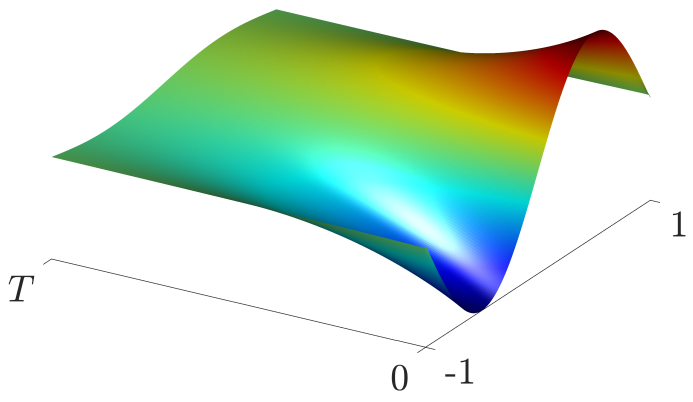
Numerical examples - FD + backward Euler

Finite Difference for the space discretization
Backward Euler for the time discretization

$\Delta_t = 500\Delta_x^2$



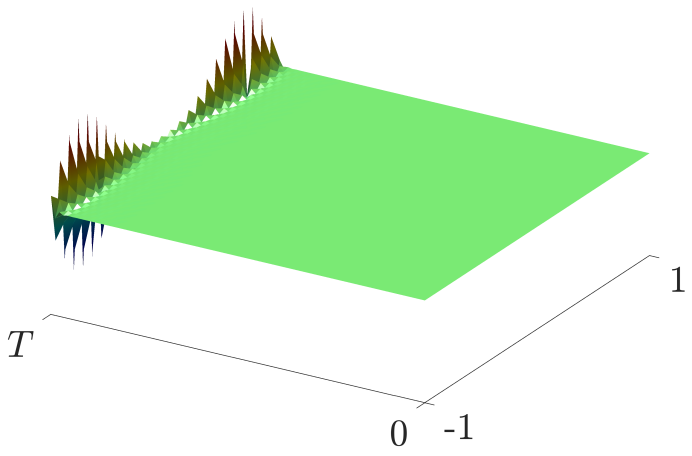
$\Delta_t = 0.5\Delta_x^2$



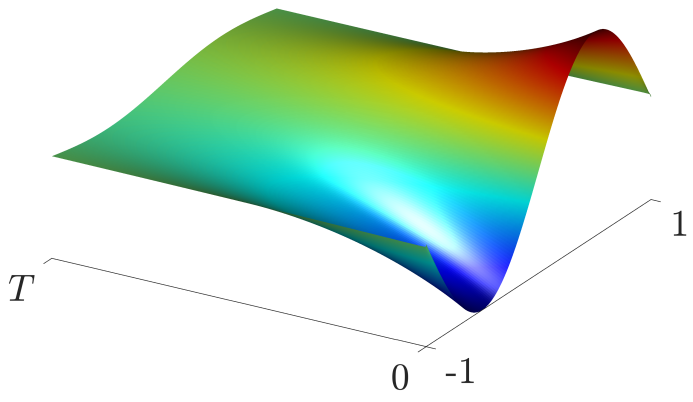
Numerical examples - FD + forward Euler

Finite Difference for the space discretization
Forward Euler for the time discretization

$\Delta_t = \Delta_x^2$



$\Delta_t = 0.5\Delta_x^2$



References

References

- S. C. Brenner and L. R. Scott, *The mathematical theory of finite element methods*, Springer, 2008.
- P. Ciarlet and J. L. Lions, *Handbook of Numerical Analysis - Finite Difference Methods*, Elsevier, 1990.
- E. Isaacson and H. B. Keller, *Analysis of numerical methods*, Courier Corporation, 2012.
- A. Quarteroni and A. Valli, *Numerical approximation of partial differential equations*, Springer, 2008.

THANK YOU FOR YOUR ATTENTION!

Funding

- European Research Council (ERC): grant agreements NO: 694126-DyCon and No.765579-ConFlex.
- MINECO (Spain): Grant PID2020-112617GB-C22 KILEARN
- Alexander von Humboldt-Professorship program
- DFG (Germany): Transregio 154 Project "Mathematical Modelling,Simulation and Optimization Using the Example of Gas Networks"
- COST Action grant CA18232, "Mathematical models for interacting dynamics on networks".