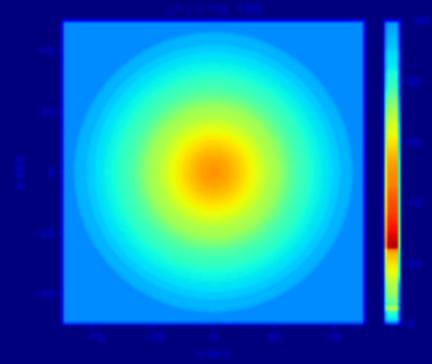
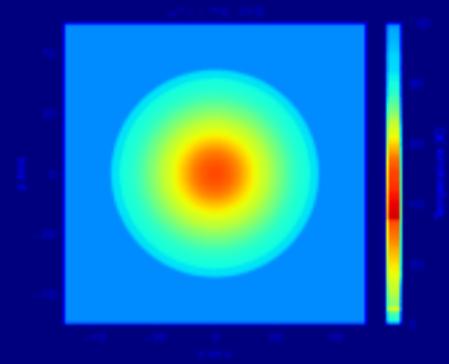
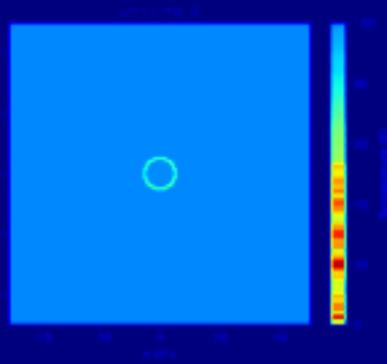


Partial Differential Equations I



Carlos Aznarán Laos

Last change: October 28, 2024 at 12:15am.

Useful links



Click on each vignette for updated resources or the book cover on next slides.

- Pad + general information
- Meeting link Mon, Fri 09:00:00 PM -05
- Beamer slides + Report lecture
- Analytical methods for solve the wave equation (1D, 2D and 3D) course + books
- Live recordings + Jason Bramburger's lectures
- Repository
- Animations with matplotlib
- Shared folder + exercises

Remark

We'll try to follow this outline <https://math.dartmouth.edu/~m53f22>.

VisualPDE

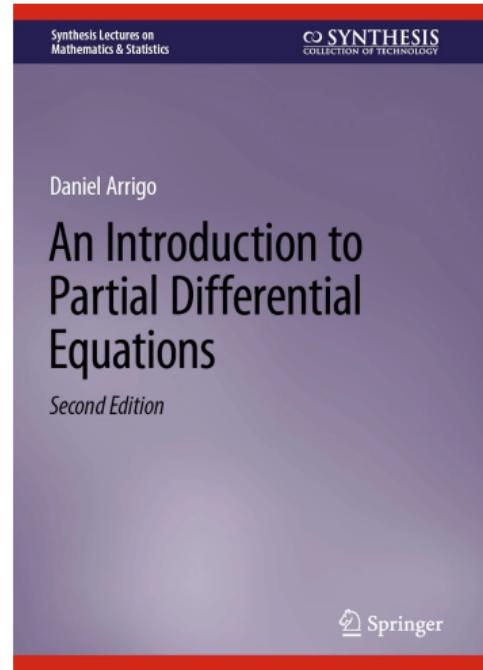
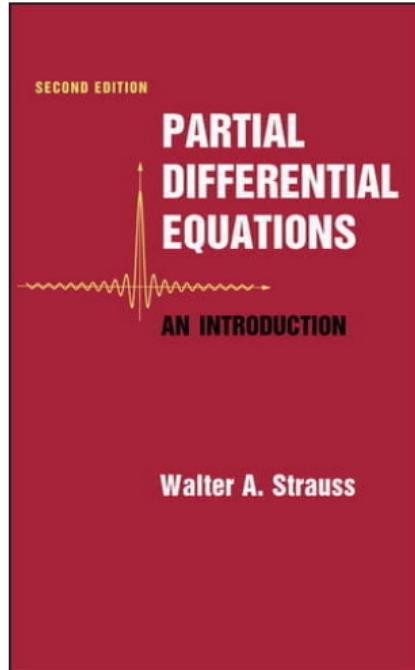
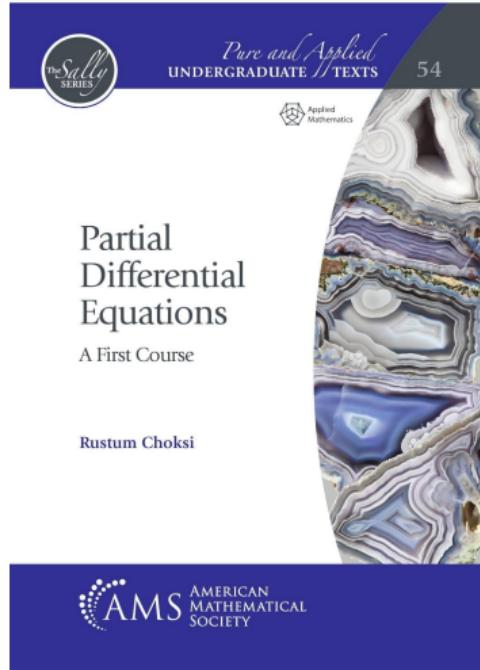


Every time we explore a new PDE we are likelihood to visualize the animation on <https://visualpde.com>.

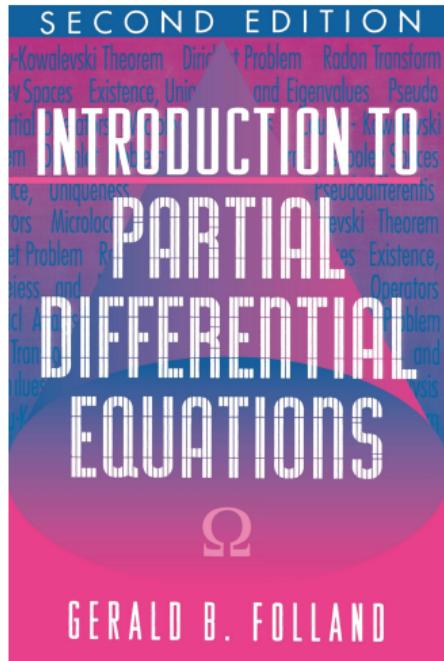
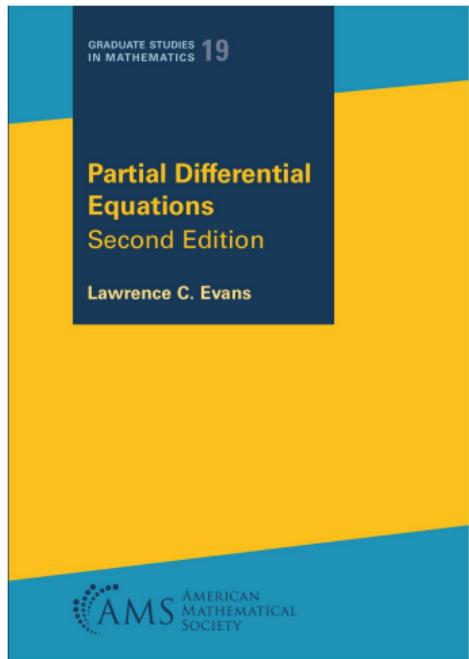
Universal document viewer

Okular is a PDF viewer that allows interaction with forms, e.g., display animations of time dependent PDE solutions.

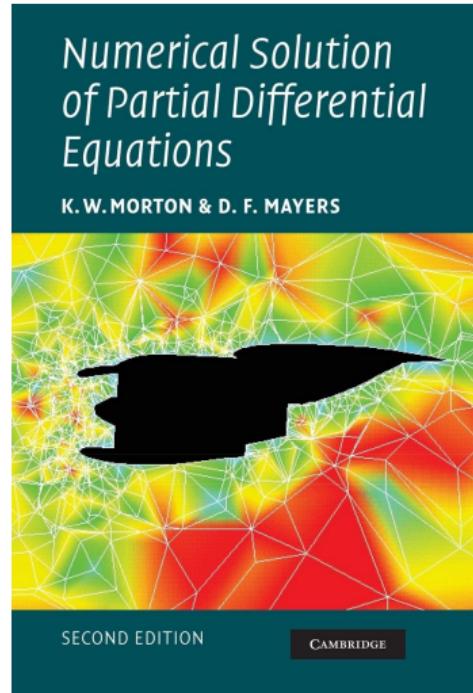
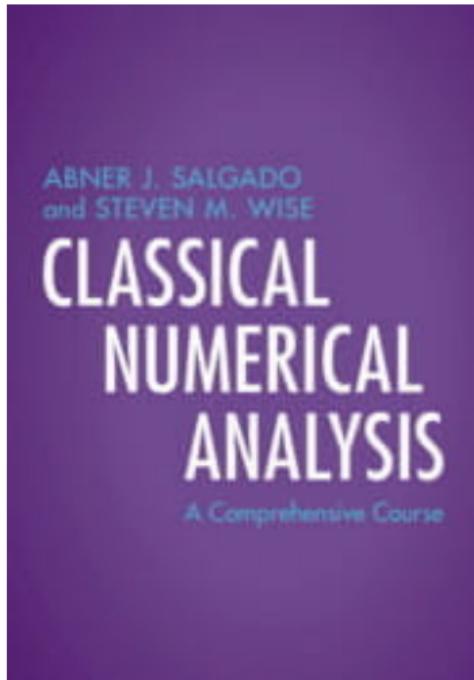
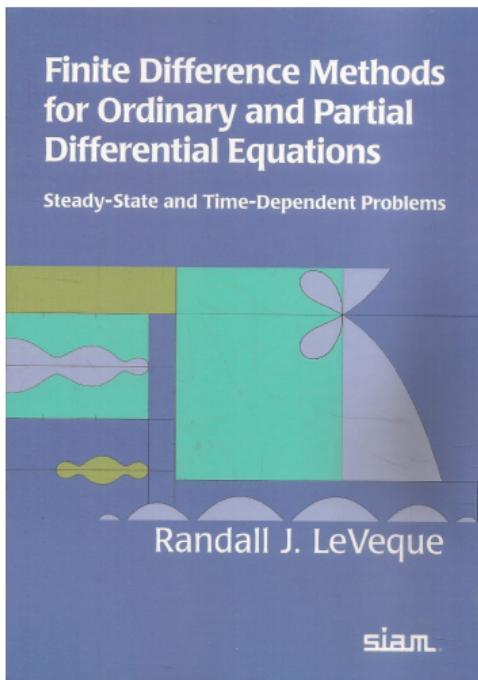
References with foundations on ODE



References with foundations on Functional Analysis

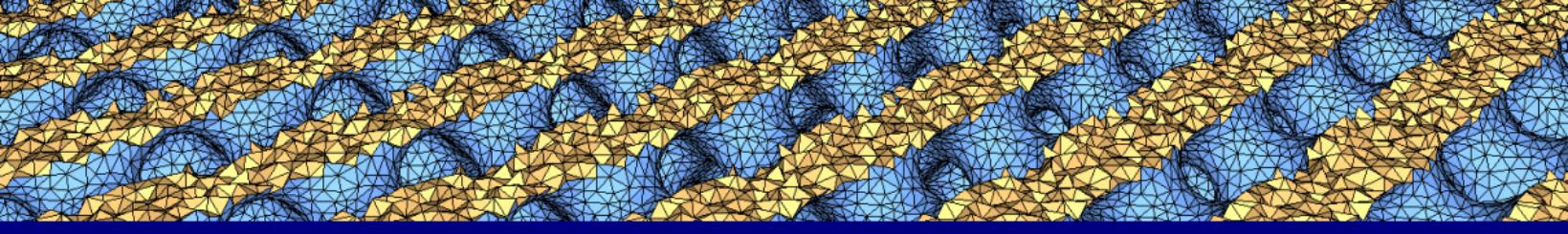


References with foundations on Numerical Analysis



Contents

- 1 Review of ODEs**
- 2 Using Python  to Solve PDEs**
- 3 Fourier stability analysis**
- 4 Basic definitions**
- 5 Classification of Linear Second Order Partial Differential Equations**
- 6 Method of characteristics**
- 7 Trigonometric Fourier Series**
- 8 Fourier transform**
- 9 Distribution**
- 10 Wave operator**
- 11 Wave equation with two spatial dimensions**
- 12 Diffusion operator**
- 13 Laplace operator**
- 14 The Separation of Variables Algorithm for Boundary Value Problems**



Review of ODEs

Review of ODEs

An ordinary differential equation (ODE) is a *functional equation* that relates some function with its derivatives.

Example (Classification of ODEs

- Heterogeneous first-order linear constant coefficient.

$$\frac{du}{dx} = \pi u + \cos(x).$$

- Homogeneous second-order linear.

$$\frac{d^2u}{dx^2} - x \frac{du}{dx} + u = 0.$$

- Homogeneous second-order linear constant coefficient.

$$\frac{d^2u}{dx^2} + \alpha^2 u = 0.$$

- Heterogeneous first-order nonlinear.

$$\frac{du}{dx} = u^5 + 1.$$

Review of ODEs

For functions of several variables, an ODE becomes in a PDE.

Example (PDE models)

- Models the concentration of a substance **flowing** in a fluid at a constant rate $c \in \mathbb{R} \setminus \{0\}$.

(Advection)

$$\partial_t u + c\partial_x u = 0.$$

Its general solution is $u(x, t) = \phi(x - ct)$ where ϕ is an arbitrary function.

- Type of **propagating** disturbance that moves faster than the speed of sound in a medium.

(Shock waves)

$$\partial_x u + u\partial_y u = 0.$$

Like a common wave, a shock wave carries energy and can propagate through a medium, but is characterized by an abrupt, almost discontinuous change in the pressure, temperature, and density of the medium.

- Models the constant **heat flow** in a region where the temperature is fixed at the boundary.

(Laplace)

$$\Delta u = 0.$$

Review of ODEs

More classifications of differential equations

- An integro-differential equation involving both the derivatives and its anti-derivatives of a solution.

(RLC circuit )

$$L \frac{dI(t)}{dt} + RI(t) + \frac{1}{C} \int_0^t I(\tau) d\tau = E(t).$$

- A functional differential equation with deviating argument and more applicable than ODEs.

(Population growth )

$$\frac{du(t)}{dt} = \rho u(t) \left(1 - \frac{u(t-\tau)}{k}\right).$$

- A stochastic differential equation is composed in terms of stochastic process.

(Arithmetic Brownian motion )

$$dX_t = \mu dt + \sigma dB_t.$$

- A differential algebraic equation involves differential and algebraic terms.
- Stiff PDE, Delay PDE, Controlled PDE, Fractional PDE, Neural PDE and so on.

Review of ODEs

Let the IVP

$$\begin{cases} \frac{du}{dt} = -\frac{u}{2}, & t \in [0, 10], \\ u(0) = a_i, \end{cases}$$

where $a_1 = 2$, $a_2 = 4$, $a_3 = 6$ and $a_4 = 8$.

```
import numpy as np
from jaxtyping import Array, Float
from scipy.integrate import solve_ivp

def exponential_decay(
    t: Float[Array, "dim"], u: Float[Array, "dim"]
) -> Float[Array, "2"]:
    return -0.5 * u
```

```
sol = solve_ivp(
    fun=exponential_decay,
    t_span=(0, 10),
    y0=(2, 4, 6, 8),
    t_eval=np.linspace(start=0, stop=10),
    dense_output=True,
)
```

Program  : Recovered

from https://docs.scipy.org/doc/scipy-1.14.1/reference/generated/scipy.integrate.solve_ivp.html.

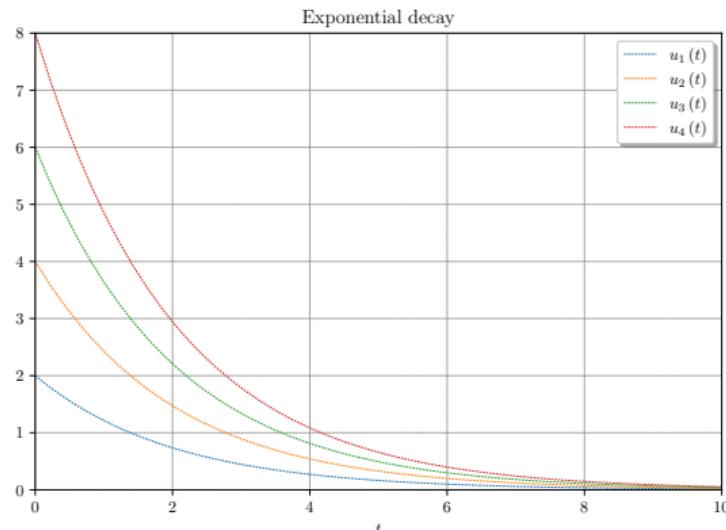


Figure: Numerical solution.

Review of ODEs

The BVP

$$\frac{du}{dx} + \exp(u) = 0, \quad u(0) = u(1) = 0.$$

```
import numpy as np
from jaxtyping import Array, Float
from scipy.integrate import solve_bvp

def fun(x: Float[Array, "dim"], u: Float[Array, "2"]) → Float[Array, "2"]:
    return np.vstack((u[1], -np.exp(u[0])))

def bc(ua: float, ub: float) → Float[Array, "2"]:
    return np.array([ua[0], ub[0]])

x = np.linspace(start=0, stop=1, num=5)
u_a = np.zeros(shape=(2, x.size))
u_b = np.copy(a=u_a)
u_b[0] = 3

sol_a = solve_bvp(fun=fun, bc=bc, x=x, y=u_a)
sol_b = solve_bvp(fun=fun, bc=bc, x=x, y=u_b)
```

Program  : Recovered
from https://docs.scipy.org/doc/scipy-1.14.1/reference/generated/scipy.integrate.solve_bvp.html.

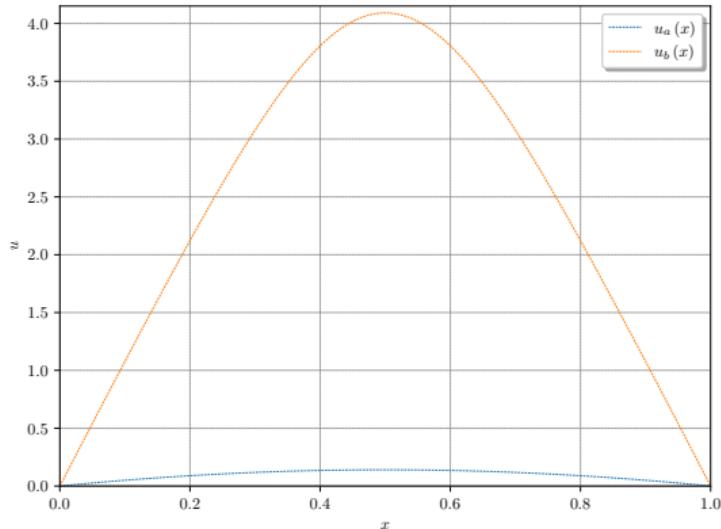


Figure: Numerical solution.

Theorem (Existence and Uniqueness of solutions - Picard-Lindelöf)

Consider the initial value problem

$$(1) \quad \begin{cases} \frac{du}{dx} = f(x, u), \\ u(\xi) = \eta. \end{cases}$$

Here it is assumed that $f(\cdot, \cdot)$ is continuous on $[\xi, \xi + a] \times \mathbb{R}$ where $a > 0$, and furthermore satisfies

(Lipschitz condition) $|f(x, u) - f(x, \bar{u})| \leq L |u - \bar{u}|$

for some $L \in \mathbb{R}_{\geq 0}$; here all $x \in [\xi, \xi + a]$, $u, \bar{u} \in \mathbb{R}$ are allowed. Then (1) admits precisely one C^1 -solution $u(x)$ on $[\xi, \xi + a]$.

ODE - Part 12

$\dot{x} = v(x)$

$x(0) = x_0$

solution?

Existence and Uniqueness Theorem!

Idea of proof.

- 1 Formulation as a fixed point problem.

$$u(x) = \eta + \int_{\xi}^x f(t, u(t)) dt.$$

- 2 Introduction of a Banach space, verifying contraction property.

$$\begin{aligned} T: C^0(I_b) &\longrightarrow C^0(I_b) \\ u &\longmapsto \eta + \int_{\xi}^x f(t, u(t)) dt. \end{aligned}$$

- 3 Application of Contraction Principle, construction of local solution.

□

Theorem (Peano)

For $I = [\xi, \xi + a]$, $J = [\eta - b, \eta + b]$, we have $f \in C^0(I \times J)$, $|f|_{C^0(I \times J)} \leq M$ for some $M, a, b > 0$, there exists a solution $u(x) \in C^1\left([\xi, \xi + \min\{a, \frac{b}{M+1}\}]\right)$.

Idea of proof.

- 1 The idea is to reduce to the situation in Picard's theorem.
- 2 The **mollification** of f is now given by the family of functions.

$$f_\varepsilon(x, u) := f *_u \chi_\varepsilon(x, u) = \int_{\mathbb{R}} f(x, u - z) \chi_\varepsilon(z) dz.$$

- 3 In order to be able to invoke the version of Picard's theorem, we need to extend $f_\varepsilon(x, u)$ to all \mathbb{R} .

$$|f_\varepsilon(x, u) - f_\varepsilon(x, \bar{u})| \leq \frac{C}{\varepsilon} M |u - \bar{u}|.$$

- 4 Use the Arzelà-Ascoli theorem.



Techniques to solve First order ODEs

Separable equation

If the right hand side of the equation

$$\frac{du}{dx} = g(x) p(u)$$

can be expressed as function $g(x)$ that depends only of x times a function $p(u)$ that depends only on u , the differential equation is called **separable**.

Example (Separable equation )

$$\frac{du}{dx} = \frac{x-5}{u^2}.$$

Solution

$$u^2 du = (x-5) dx.$$

$$\int u^2 du = \int (x-5) dx.$$

$$\frac{u^3}{3} = \frac{x^2}{2} - 5x + C \implies u(x) = \left(\frac{3x^2}{2} - 15x + K \right)^{\frac{1}{3}}.$$

Techniques to solve First order ODEs

Linear equation

In order to solve the ODE in the **standard form**

$$(2) \quad \frac{du}{dx} + P(x)u(x) = Q(x).$$

Calculate the **integrating factor** $\mu(x)$ by

$$(3) \quad \mu(x) = \exp \left[\int P(x) dx \right].$$

And multiply (2) by (3)

$$\frac{d}{dx} [\mu(x)u(x)] = \mu(x)Q(x).$$

And obtain the solution

$$u(x) = \frac{1}{\mu(x)} \left[\int \mu(x)Q(x) dx + C \right].$$

Example (Linear equation )

$$\frac{du}{dx} + 2u(x) = 50 \exp(-10x).$$

Techniques to solve Second order ODEs

Homogeneous linear second order ode

Let be $a \in \mathbb{R} \setminus \{0\}$.

$$a \frac{d^2u}{dx^2} + b \frac{du}{dx} + cu = 0.$$

Find a solution of the form $u(x) = e^{rx}$.

$$\begin{aligned} ar^2 e^{rx} + bre^{rx} + ce^{rx} &= 0. \\ e^{rx} (ar^2 + br + c) &= 0. \end{aligned}$$

Since $e^{rx} > 0$

$$ar^2 + br + c = 0.$$

Example (Homogeneous linear second order)

$$\frac{d^2u}{dx^2} + 5 \frac{du}{dx} - 6u = 0.$$

Solution

$$r^2 + 5r - 6 = (r - 1)(r + 6) = 0.$$

e^x and e^{-6x} are solutions.

Techniques to solve Second order ODEs

Example ()

$$\frac{d^3u}{dx^3} + 3\frac{d^2u}{dx^2} - \frac{du}{dx} - 3u = 0.$$

Techniques to solve Second order ODEs

Nonhomogeneous

$$a \frac{d^2u}{dx^2} + b \frac{du}{dx} + cu = f(x).$$

Example (Nonhomogeneous )

$$\frac{d^2u}{dx^2} + 3 \frac{du}{dx} + 2u = 3x.$$

Techniques to solve Second order ODEs

Method of Variation of Parameters

$$a \frac{d^2 u}{dx^2} + b \frac{du}{dx} + cu = f(x).$$

$$u_h(x) = c_1 u_1(x) + c_2 u_2(x).$$

$$u_p(x) = v_1(x) y_1(x) + v_2(x) y_2(x).$$

Example ()

$$\frac{d^2 u}{dx^2} + u = \tan x.$$

Solution

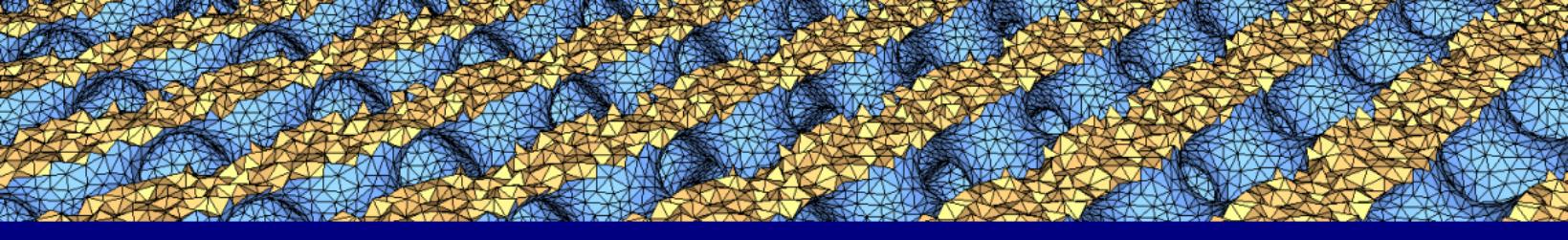
The homogeneous equation $\frac{d^2 u}{dx^2} + u = 0$ are $\cos x$ and $\sin x$.

$$u_p(x) = v_1(x) \cos(x) + v_2(x) \sin(x).$$

$$v_1(x) = \sin(x) - \ln |\sec x + \tan x| + C_1.$$

$$v_2(x) = -\cos x + C_2.$$

$$u(x) = c_1 \cos x + c_2 \sin x - (\cos x) \ln(\sec x + \tan x).$$



Using Python to Solve PDEs

Symbolic computing in Python

- Library for symbolic mathematics.
- It aims to become a full-featured **computer algebra system** while keeping the code as simple as possible in order to be comprehensible and easily extensible.

Recovered from <https://www.sympy.org>.

- 2 hours tutorial by Aaron Meurer: <https://youtu.be/FZWevQ6Xz6U?t=3059>.
- Featured modules: `sympy.calculus`, `sympy.integrals`, `sympy.series.fourier`, `sympy.solvers.ode`, and `sympy.solvers.pde`.
- Latest version 1.13.3 dated September 18, 2024.



Symbolic computing in Python

```
from sympy import Derivative as D
from sympy.abc import U, c, k, r, theta, x, y
from sympy.core import Eq, Function, I, Symbol, pi, var
from sympy.functions import exp
from sympy.integrals import fourier_transform, inverse_fourier_transform

u = Function("u")

laplace = Eq(lhs=D(U, x, 2) + D(U, y, 2), rhs=0)

Δt = Symbol("Δt")
Δx = Symbol("Δx")
u = Function("u")

taylor = Eq(u(x + Δx), u(x + Δx).series(x=Δx, x0=0, n=3).simplify())

expresion1 = (exp(I * theta) + exp(-I * theta)) / 2
expresion2 = (exp(I * theta) - exp(-I * theta)) / (2 * I)
expresion3 = (exp(I * theta) - exp(-I * theta)) / 2

Unpj = var("U^{n+1}_j")
Unj = var("U^n_j")
Unjm1 = var("U^{n_{j-1}}")
FOU = Eq(
    Unpj,
    solve(((Unpj - Unj) / Δt + c * (Unj - Unjm1) / Δx).subs({Δt: r * Δx / c}), Unpj)[
        0
    ],
)
a = fourier_transform(f=exp(-(x**2)), x=x, k=k)
b = inverse_fourier_transform(F=a, k=k, x=x)

s = fourier_series(f=x**2, limits=(x, -pi, pi))
iterator = s.truncate(n=None)
for n in range(10):
    term = next(iterator)
    print(f"\"a_{n}:\")"
    pprint(term.subs(x, 0))
```

Laplace equation:

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

Taylor expansion:

$$u(x + \Delta x) = u(x) + \Delta x \cdot \frac{d}{dx}(u(x)) + \frac{\Delta x^2}{2} \cdot \frac{d^2}{dx^2}(u(x)) + O(\Delta x^3)$$

Simplify:

$$\begin{aligned} &\cos(\theta) \\ &\sin(\theta) \\ &i \cdot \sin(\theta) \end{aligned}$$

Solve equation:

$$U^{n+1}_j = -U^n_j \cdot r + U^n_j + U^{n_{j-1}} \cdot r$$

Laplace transform:

$$\frac{2}{-\pi \cdot k}$$

$$\sqrt{\pi} \cdot e$$

Inverse Laplace transform:

$$\frac{2}{-x} \cdot e$$

py-pde: A Python package for solving partial differential equation

- Contains classes for grids on which scalar and tensorial fields can be defined.
The associated differential operators are computed using a implementation of finite differences.

Recovered from <https://www.zwickergroup.org/software>.

- 60 minutes tutorial by me: <https://youtu.be/2xnK2ubFAt0?t=273>.
- Highlights three useful modules: `pde.pdes.laplace`, `pde.pdes.diffusion` and `pde.pdes.wave`.
- Latest version `0.41.0` dated August 5, 2024.



py-pde: A Python package for solving partial differential equation

```
from math import pi
from pde import CartesianGrid, solve_laplace_equation
res = solve_laplace_equation(
    grid=CartesianGrid(bounds=[[0, 2 * pi]] * 2, shape=2**8),
    bc=[{"value": "sin(y)"}, {"value": "sin(x)"}],
)
```

$\Delta u = 0$ with boundary conditions: $\sin(y), \sin(x)$

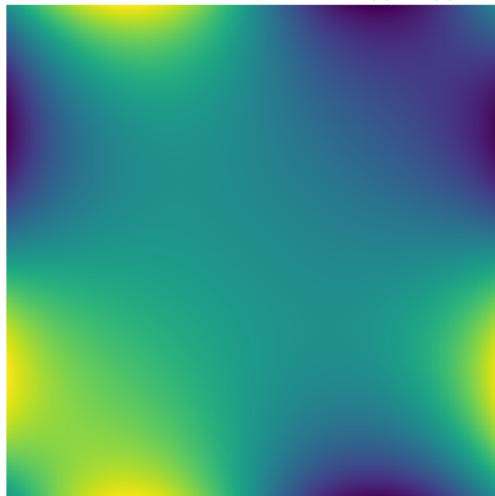


Figure: Solution of Laplace equation using the methods of lines in a rectangular grid.

DOLFINx: Next generation FEniCS problem solving environment

- The FEniCS project is a research and software project aimed at creating mathematical methods and software for solving PDEs.
- The latest version of the FEniCS project, FEniCSx, consists of several building blocks, namely DOLFINx, UFL, FFCx, and Basix.
- DOLFINx, being a state of the art **finite element** solver.

Recovered from <https://jsdokken.com/dolfinx-tutorial>.

- 45 minutes tutorial by Antonio Baiano Svizzero:
<https://youtu.be/uQW2wSDtW5k>.
- Highlights five useful modules: `dolfinx.mesh.create_rectangle`,
`dolfinx.fem.functionspace`, `dolfinx.mesh.locate_entities_boundary`,
`ufl.TestFunction` and `ufl.TrialFunction`.
- Latest version `0.9.0.post0` dated October 9, 2024.



DOLFINx: Next generation FEniCS problem solving environment

```
import numpy as np
from dolfinx.fem import dirichletbc, functionspace, locate_dofs_topological
from dolfinx.fem.petsc import LinearProblem
from dolfinx.io import VTKFile
from dolfinx.mesh import CellType, create_rectangle, locate_entities_boundary
from mpi4py import MPI
from petsc4py.PETSc import ScalarType
from ufl import (
    SpatialCoordinate,
    TestFunction,
    TrialFunction,
    ds,
    dx,
    exp,
    grad,
    inner,
    sin,
)
msh = create_rectangle(
    comm=MPI.COMM_WORLD,
    points=((0.0, 0.0), (2.0, 1.0)),
    n=(32, 16),
    cell_type=CellType.triangle,
)
import numpy as np
from dolfinx.fem import dirichletbc, functionspace, locate_dofs_topological
from dolfinx.fem.petsc import LinearProblem
from dolfinx.io import VTKFile
from dolfinx.mesh import CellType, create_rectangle, locate_entities_boundary
from mpi4py import MPI
from petsc4py.PETSc import ScalarType
from ufl import (
    SpatialCoordinate,
    TestFunction,
    TrialFunction,
    ds,
    dx,
    exp,
    grad,
    inner,
    sin,
)
msh = create_rectangle(
    comm=MPI.COMM_WORLD,
    points=((0.0, 0.0), (2.0, 1.0)),
    n=(32, 16),
    cell_type=CellType.triangle,
)
V = functionspace(mesh=msh, element=("Lagrange", 1))
facets = locate_entities_boundary(
    msh=msh,
    dim=1,
    marker=lambda x: np.logical_or(np.isclose(x[0], 0.0), np.isclose(x[0], 2.0)),
)
dofs = locate_dofs_topological(V=V, entity_dim=1, entities=facets)
bc = dirichletbc(value=ScalarType(0), dofs=dofs, V=V)

u = TrialFunction(function_space=V)
v = TestFunction(function_space=V)
x = SpatialCoordinate(domain=msh)
f = 10 * exp(-((x[0] - 0.5) ** 2 + (x[1] - 0.5) ** 2) / 0.02)
g = sin(f*5 * x[0])
a = inner(a=grad(u), b=grad(v)) * dx
L = inner(a=f, b=v) * dx + inner(a=g, b=v) * ds

problem = LinearProblem(
    a=a, L=L, bcs=[bc], petsc_options={"ksp_type": "preonly", "pc_type": "lu"}
)
uh = problem.solve()
```

DOLFINx: Next generation FEniCS problem solving environment

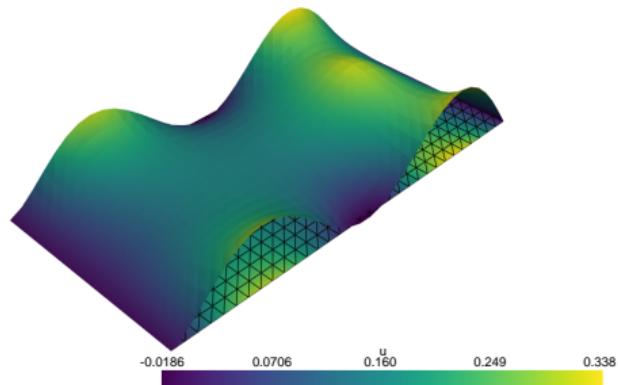


Figure: Warp By Scalar filter over solution.

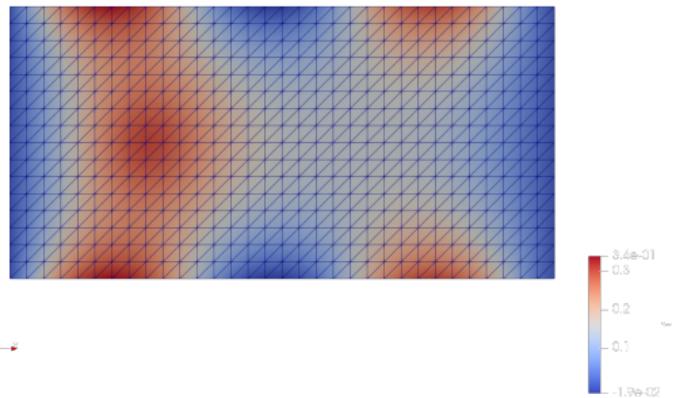


Figure: Solution.

Conservation Laws Package (Clawpack)

- Is a collection of **finite volume methods** for linear and nonlinear hyperbolic systems of conservation laws.
- Employs high-resolution Godunov-type methods with limiters in a general framework applicable to many kinds of waves.

Recovered from <https://www.clawpack.org>.

- 15 minutes tutorial by Felix Köhler: <https://youtu.be/tr348El2A4Q>.
- Latest version 5.11.0 dated August 26, 2024.
- PyClaw: Python version of the hyperbolic PDE solvers that allows solving the problem in Python without explicitly using any Fortran code.
- Friendly tutorial:
https://en.ancey.ch/cours/doctoрат/tutorial_clawpack.pdf
- Randall J. LeVeque and et.al wrote two books around this package:
 - *Finite Volume Methods for Hyperbolic Problems* (2002).
 - *Riemann Problems and Jupyter Solutions* (2020).



Conservation Laws Package (Clawpack)

```
import numpy as np
from clawpack.pyclaw import BC, ClawSolver1D, Controller, Dimension, Domain, Solution
from clawpack.pyclaw.plot import interactive_plot
from clawpack.riemann import advection_1D

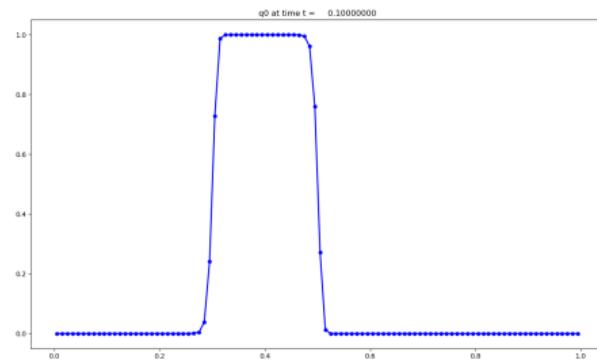
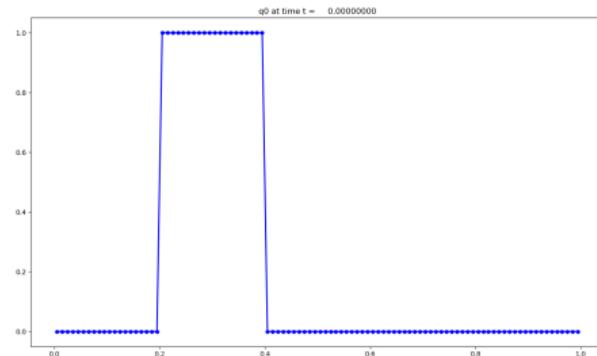
# the library
solver = ClawSolver1D(riemann_solver=advection_1D)
solver.bc_lower[0] = BC.periodic
solver.bc_upper[0] = BC.periodic

x_dimension = Dimension(lower=0.0, upper=1.0, num_cells=100)
domain = Domain(x_dimension)

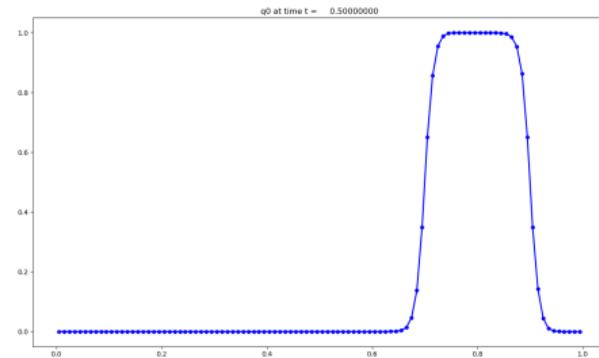
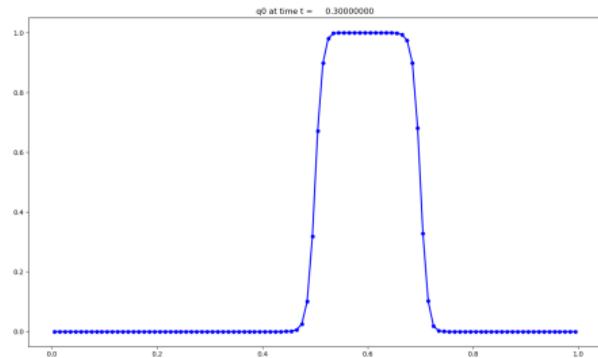
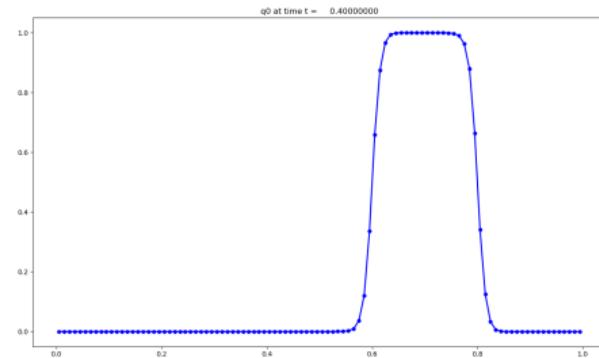
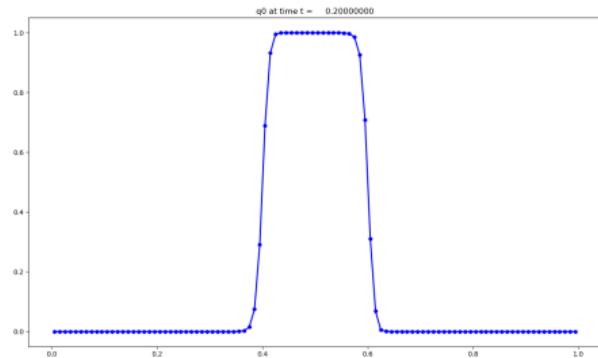
solution = Solution(solver.num_eqn, domain)

state = solution.state
cell_center_coordinates = state.grid.p_centers[0]
state.q[0, :] = np.where(
    (cell_center_coordinates > 0.2) & (cell_center_coordinates < 0.4),
    1.0,
    0.0,
)
state.problem_data["u"] = 1.0

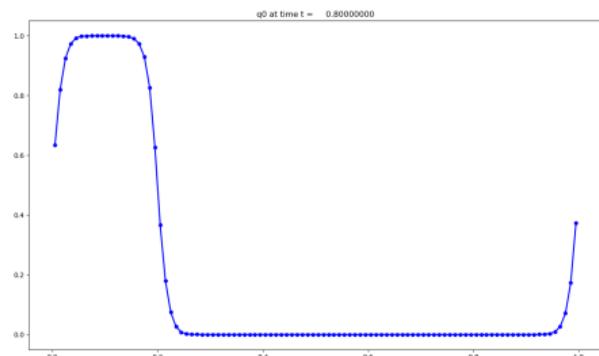
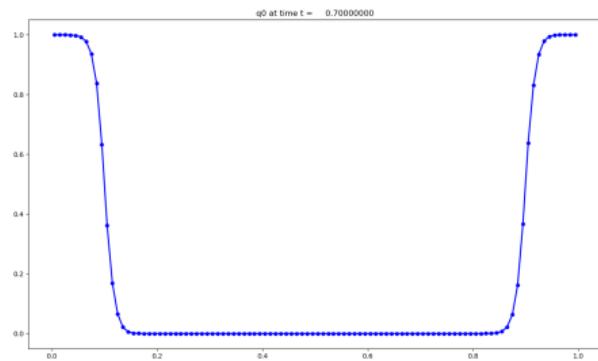
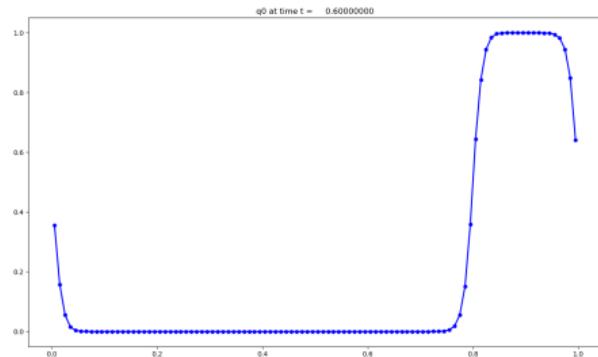
controller = Controller()
controller.solution = solution
controller.solver = solver
controller.tfinal = 1.0
```



Conservation Laws Package (Clawpack)



Conservation Laws Package (Clawpack)



Portable, Extensible Toolkit for Scientific Computation (PETSc)

- is a suite of data structures and routines for the scalable (parallel) solution of scientific applications modeled by **partial differential equations**.
- It employs the MPI standard for all message-passing communication.

Recovered from <https://petsc.org/release/petsc4py>.

- 25 minutes tutorial by Felix Köhler: <https://youtu.be/oqxPyRZKOu4>.
- Latest version 3.22.0 dated September 28, 2024.
- Demo: [Solve a constant coefficient Poisson problem on a regular grid](#).
- Book: [PETSc for Partial Differential Equations: Numerical Solutions in C and Python](#) (2020).



Portable, Extensible Toolkit for Scientific Computation (PETSc)

```
import numpy as np
import petsc4py

from petsc4py.PETSc import KSP, Mat, Vec

N_POINTS = 1001
TIME_STEP_LENGTH = 0.001
N_TIME_STEPS = 100
mesh, Ax = np.linspace(start=0.0, stop=1.0, num=N_POINTS, retstep=True)
# Create a new sparse PETSc matrix, fill it and then assemble it
A = Mat().createAIJ([N_POINTS, N_POINTS])
A.setUp()

diagonal_entry = 1.0 + 2.0 * TIME_STEP_LENGTH / Ax**2
off_diagonal_entry = -1.0 * TIME_STEP_LENGTH / Ax**2

A.setValue(0, 0, 1.0)
A.setValue(N_POINTS - 1, N_POINTS - 1, 1.0)

for i in range(1, N_POINTS - 1):
    A.setValue(i, i, diagonal_entry)
    A.setValue(i, i - 1, off_diagonal_entry)
    A.setValue(i, i + 1, off_diagonal_entry)

A.assemble()

# Define the initial condition
initial_condition = np.where(
    (mesh > 0.3) & (mesh < 0.5),
    1.0,
    0.0,
)
# Assemble the initial rhs to the linear system
b = Vec().createSeq(N_POINTS)
b.setArray(initial_condition)
b.setValue(0, 0.0)
b.setValue(N_POINTS - 1, 0.0)

# Allocate a PETSc vector storing the solution to the linear system
x = Vec().createSeq(N_POINTS)

# Instantiate a linear solver: Krylov subspace linear iterative solver
ksp = KSP().create()
ksp.setOperators(A)
ksp.setFromOptions()

chosen_solver = ksp.getType()
print(f"Solving with {chosen_solver}:")

def animate(ti):
    print(f"Frame: {ti + 1}")
    plt.clf()
    plt.plot(
        mesh,
        initial_condition,
        color="black",
        label="Initial state",
        linewidth=0.4,
        linestyle="dashed",
    )
    ksp.solve(b, x)

    # Re-assemble the rhs to move forward in time
    current_solution = x.getArray()
    b.setArray(current_solution)
    b.setValue(0, 0.0)
    b.setValue(N_POINTS - 1, 0.0)
```


A python hydrodynamics code for teaching and prototyping (pyro2)

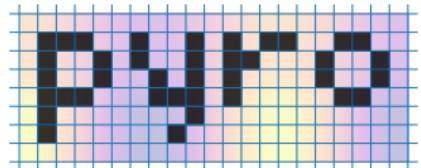
- It is designed to provide a tutorial for students in computational astrophysics (and hydrodynamics in general) and for easily prototyping new methods.

- Builds off of a finite-volume framework for solving PDEs.

Recovered from <https://python-hydro.github.io/pyro2>.

- Latest version 4.4.0 dated September 21, 2024.

- Michael Zingale wrote an open text [Introduction to Computational Astrophysical Hydrodynamics](#) that introduces the core finite-volume methods used in astrophysics simulation codes.



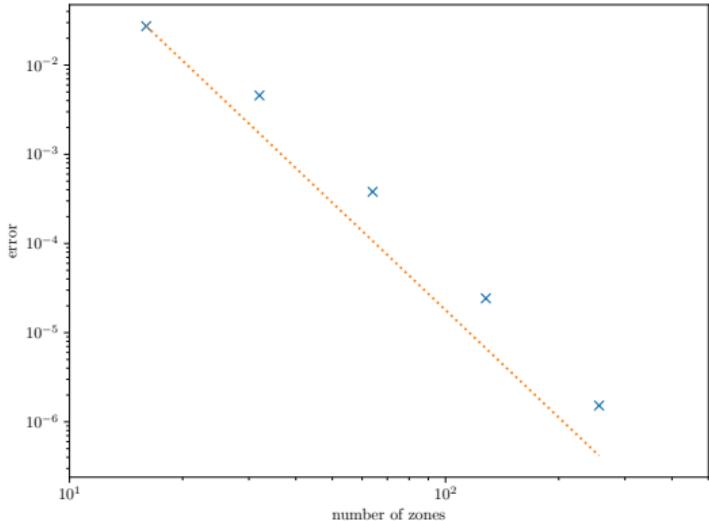
A python hydrodynamics code for teaching and prototyping (pyro2)

```
from pyro import Pyro

nzones = [16, 32, 64, 128, 256]
err = []
params_all = {"driver.cfl": 0.5, "driver.max_steps": 5000}

for N in nzones:
    params = {"mesh.nx": N, "mesh.ny": N}
    p = Pyro("advection_fv4")
    p.initialize_problem(problem_name="smooth", inputs_dict=params | params_all)
    a_init = p.get_var("density").copy()
    p.run_sim()
    print(f"N = {N}, number of steps = {p.sim.n}")
    a = p.get_var("density")
    err.append((a - a_init).norm())

N = 16, number of steps = 64
N = 32, number of steps = 128
N = 64, number of steps = 256
N = 128, number of steps = 512
N = 256, number of steps = 1024
```



Finite differences as approximations of partial derivatives

Definition (Partial derivatives)

Let $u: \mathbb{R}^d \rightarrow \mathbb{R}$ be a sufficiently smooth function and $x \in \mathbb{R}^d$.

$$\forall i = 1, \dots, d : \frac{\partial u(x)}{\partial x_i} := \lim_{\Delta x \rightarrow 0} \frac{u(x + \Delta x e_i) - u(x)}{\Delta x}$$

where e_i is the i -th vector of the canonical basis of \mathbb{R}^d .

Remark

The **finite difference method** (FDM) arises by approximating the derivative of a function u by an expression of differences in its values at certain nearby discrete points, and thus, we convert a differential equation into a finite system of algebraic equations that can be solved on the computer.

The choice of this “finite difference” should be

Consistent The approximation should be as accurate as possible and find a finite difference approximation of the derivatives which is consistent with the highest possible order.

Stable Not only with respect to data perturbations, but in discrete versions of the same rules where the solution to the continuous problem has its own stability properties.

Second-order finite difference approximation for $\partial_x^2 u$

To derive a finite difference approximation of a higher-order derivative, we need to truncate the Taylor series to a higher order than that of the derivative. For example, to derive a finite difference approximation of $\partial_x^2 u$, we require the third-order forward and backward Taylor series:

$$u(x + \Delta x) = u(x) + \Delta x \partial_x u(x) + \frac{(\Delta x)^2}{2} \partial_x^2 u(x) + \frac{(\Delta x)^3}{6} \partial_x^3 u(x) + O((\Delta x)^4).$$
$$u(x - \Delta x) = u(x) - \Delta x \partial_x u(x) + \frac{(\Delta x)^2}{2} \partial_x^2 u(x) - \frac{(\Delta x)^3}{6} \partial_x^3 u(x) + O((\Delta x)^4).$$

We want to approximate $\partial_x^2 u(x)$, so we need to eliminate the terms $\partial_x u(x)$ and $\partial_x^3 u(x)$. Since these terms have opposite signs in the forward and backward Taylor expansions, we can do this by adding the two, i.e.,

$$u(x - \Delta x) + u(x + \Delta x) = 2u(x) + (\Delta x)^2 \partial_x^2 u(x) + O((\Delta x)^4).$$

Rearranging to clear $\partial_x^2 u(x)$, we get:

$$\partial_x^2 u(x) = \frac{u(x - \Delta x) - 2u(x) + u(x + \Delta x)}{(\Delta x)^2} + O((\Delta x)^2).$$

This is the **second order symmetric** difference approximation of $\partial_x^2 u(x)$.

Remark

When adding the forward and backward Taylor expansions, the odd-order terms cancel out. This means that using an odd-order Taylor expansion will result in a first order approximation of accuracy.

Finite differences as approximations of partial derivatives

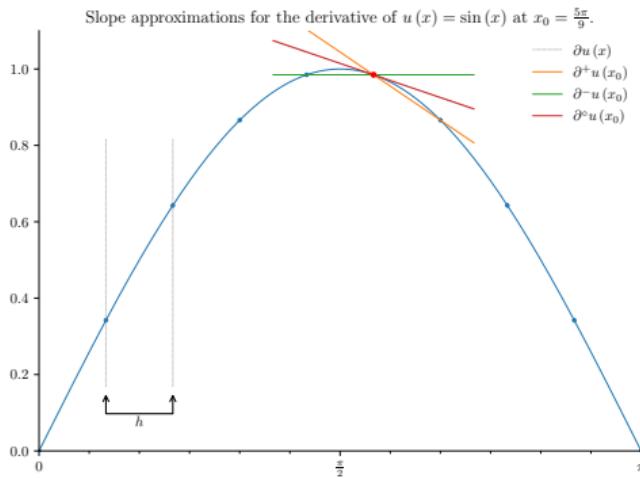


Figure: Forward difference $\partial^+ u(x)$, backward difference $\partial^- u(x)$, centered difference $\partial^\circ u(x)$

Finite differences are analogous to partial derivatives in several variables.

$$\begin{aligned}\partial_x u(x) &\approx \begin{cases} \partial^+ u(x) := \frac{u_{i+1} - u_i}{\Delta x}, & i = 0, \dots, n-1. \\ \partial^- u(x) := \frac{u_i - u_{i-1}}{\Delta x}, & i = 1, \dots, n. \\ \partial^\circ u(x) := \frac{u_{i+1} - u_{i-1}}{2\Delta x}, & i = 1, \dots, n-1. \end{cases} \\ \partial_x^2 u(x) &\approx \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}, \quad i = 1, \dots, n-1. \\ \partial_x^3 u(x) &\approx \frac{u_{i+2} - 2u_{i+1} + 2u_{i-1} - u_{i-2}}{2(\Delta x)^3}, \quad i = 2, \dots, n-2. \\ \partial_x^4 u(x) &\approx \frac{u_{i+2} - 4u_{i+1} + 6u_i - 4u_{i-1} + u_{i-2}}{(\Delta x)^4}, \quad i = 2, \dots, n-2.\end{aligned}$$

Mixed derivatives

Suppose that

$$\partial_x \partial_y u(x, y) = \partial_y \partial_x u(x, y).$$

Let Δx and Δy be the step sizes for variables x and y , respectively, and using a forward approximation ∂^+ for $\partial_x u(x, y)$ and $\partial_y u(x, y)$:

$$\begin{aligned}\partial_x u(x, y) &\approx \partial_x^+ u(x, y) = \frac{u(x + \Delta x, y) - u(x, y)}{\Delta x}. \\ \partial_y u(x, y) &\approx \partial_y^+ u(x, y) = \frac{u(x, y + \Delta y) - u(x, y)}{\Delta y}.\end{aligned}$$

Then,

$$\begin{aligned}\partial_x \partial_y u(x, y) &\approx \partial_x^+ [\partial_y^+ u(x, y)] = \frac{\partial_y^+ u(x + \Delta x, y) - \partial_y^+ u(x, y)}{\Delta x} \\ &= \frac{\frac{u(x+\Delta x, y+\Delta y) - u(x+\Delta x, y)}{\Delta y} - \frac{u(x, y+\Delta y) - u(x, y)}{\Delta y}}{\Delta x} \\ &= \frac{u(x + \Delta x, y + \Delta y) - u(x + \Delta x, y) - u(x, y + \Delta y) + u(x, y)}{\Delta x \Delta y}.\end{aligned}$$

Theorem (Properties of difference operators)

Let be $u_1, u_2: \mathbb{R} \rightarrow \mathbb{R}$ two functions.

$$\partial^+(u_1 u_2)(x) = \partial^+ u_1(x) u_2(x) + u_1(x+h) \partial^- u_2(x+h).$$

$$\partial^-(u_1 u_2)(x) = \partial^- u_1(x) u_2(x) + u_1(x-h) \partial^+ u_2(x-h).$$

$$h \sum_{k=0}^{N-1} \partial^+ u_1(kh) u_2(kh) = u_1(Nh) u_2(Nh) - u_1(0) u_2(0) - h \sum_{k=1}^N u_1(kh) \partial^- u_2(kh).$$

$$\partial^- \partial^+ u(x) = \partial^+ \partial^- u(x).$$

Gradient and Hessian

Let be $u: \mathbb{R}^d \rightarrow \mathbb{R}$ a function.

$$\forall j = 1, \dots, d : [\nabla u(x)]_j \approx \partial_j^+ u(x).$$

$$\forall j, k = 1, \dots, d : [\nabla^2 u(x)]_{jk} = \partial_{x_j} \partial_{x_k} u(x) \approx \partial_k^+ [\nabla u(x)]_j.$$

$$[\nabla^2 u(x)]_{ij} \approx \frac{u(x + \Delta x \hat{e}_i + \Delta x \hat{e}_j) - u(x - \Delta x \hat{e}_i + \Delta x \hat{e}_j) - u(x + \Delta x \hat{e}_i - \Delta x \hat{e}_j) + u(x - \Delta x \hat{e}_i - \Delta x \hat{e}_j)}{4(\Delta x)^2}.$$

Finite differences as approximations of partial derivatives

Remark

Use `np.roll` in FDM, performs periodic shift of an array.

```
import numpy as np  
  
u = np.linspace(start=0, stop=20, num=6)  
print(f"u_{i+1} = {np.roll(a=u, shift=-1)}")  
print(f"u_{i-1} = {np.roll(a=u, shift=1)}")
```

index	$u[:]$	$u[1:]$	$u[1:-1]$	$u[:-1]$
0	0	4	nan	nan
1	4	8	4	0
2	8	12	8	4
3	12	16	12	8
4	16	20	16	12
5	20	nan	nan	16

$u_{i+1} = [4. 8. 12. 16. 20. 0.]$

$u_{i-1} = [20. 0. 4. 8. 12. 16.]$

Program  : `subarrays.py`.

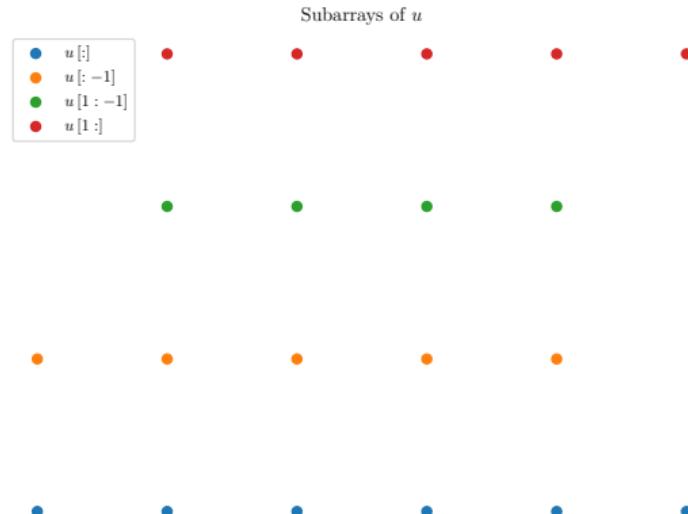


Figure: Point diagram of subarrays of u .

Finite differences as approximations of partial derivatives

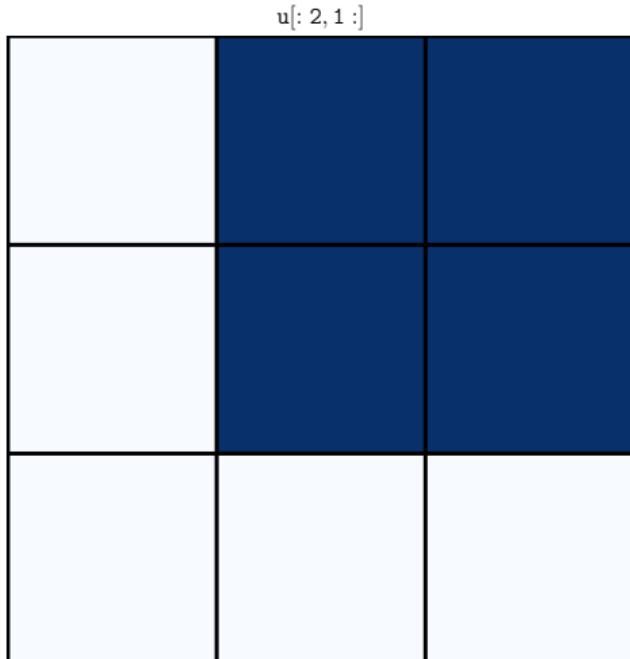


Figure: Select from the **first row to the second** and from the **second column to the end**.

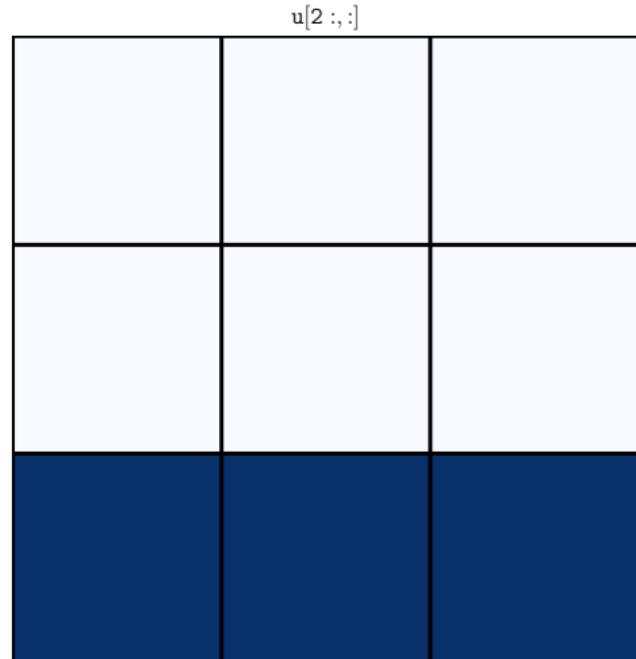


Figure: Select from the **third row to the end** and **every all columns of u**.

Finite differences as approximations of partial derivatives

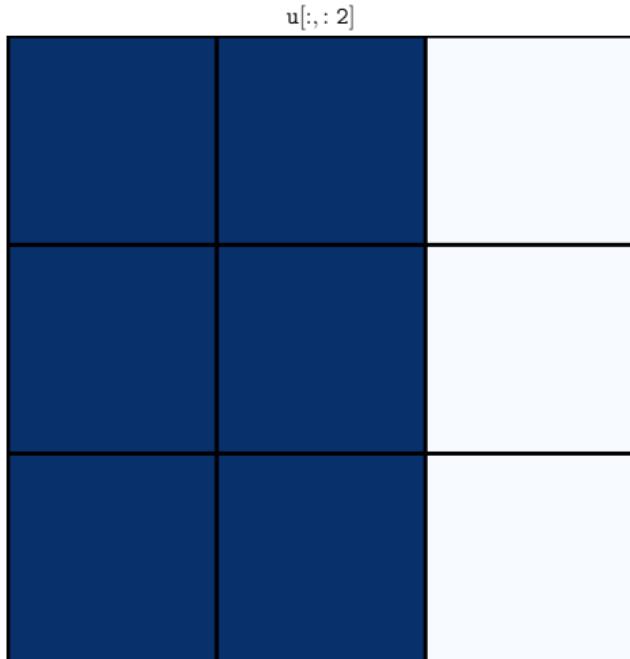


Figure: Select **every all rows** of u and from the **first column to the second**.

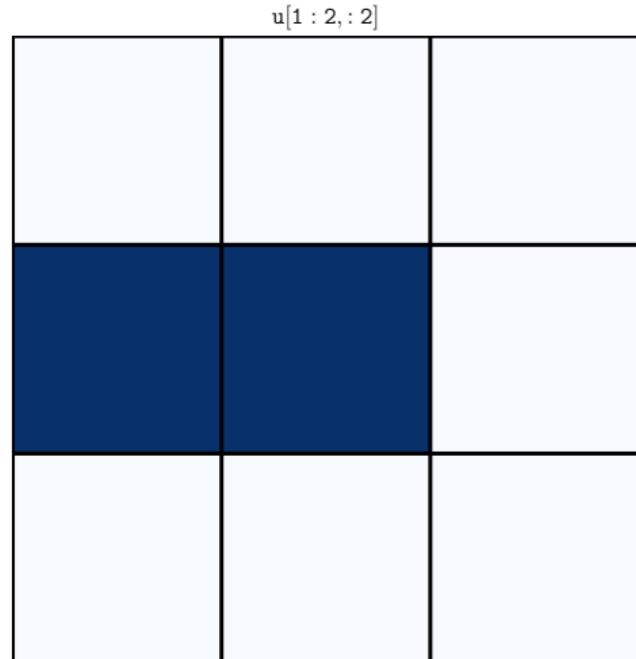


Figure: Select from the **second row to the second** and from the **first column to the second**.

Estimate the error for $\partial^+ u(x)$

By Taylor's expansion, for some $\xi \in [x, x + h]$

$$(4) \quad \partial_x u(x) = \partial^+ u(x) - \frac{\Delta x}{2} \partial_x^2 u(\xi).$$

For $u(x) = \sin(x)$ at $x = \frac{5\pi}{9}$

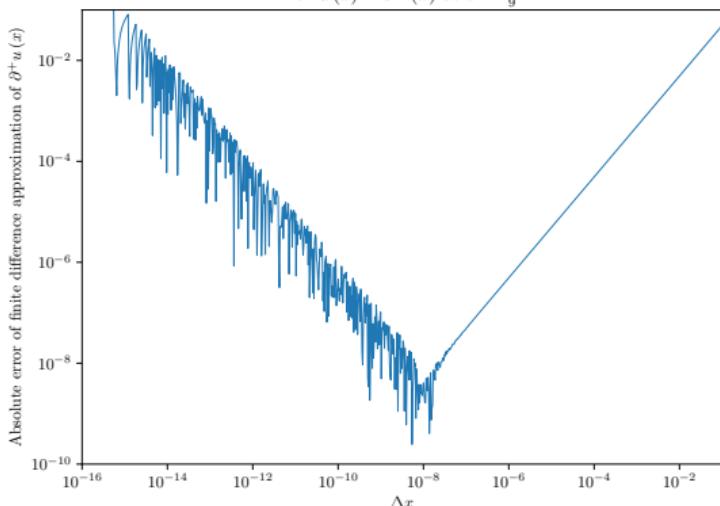


Figure: Error of $\partial^+ u(x)$ for several values of Δx .

Part of the error is due to the inaccuracies in (4) is

$$\text{truncated error} = \frac{\Delta x}{2} |\partial_x^2 u(\xi)|.$$

Depends of $\varepsilon_{\text{mach}}$ (for float64 is $2.22044604925 \times 10^{-16}$).

$$\text{rounded error} \approx \frac{|u(x)| \varepsilon_{\text{mach}}}{\Delta x} + |\partial_x u(x)| \varepsilon_{\text{mach}}.$$

The best value of Δx is obtained by minimizing the total error as function of Δx , i.e.,

$$0 = \frac{1}{2} |\partial_x^2 u(\xi)| - \frac{|u(x)| \varepsilon_{\text{mach}}}{(\Delta x)^2}.$$

$$\Delta x = \sqrt{\frac{2 |u(x)| \varepsilon_{\text{mach}}}{|\partial_x^2 u(\xi)|}}.$$

If $u(x)$ and $\partial_x^2 u(\xi)$ are neither large nor small, then

$$\Delta x \approx \sqrt{\varepsilon_{\text{mach}}} \quad (\text{for float64 is } 1.49011611938 \times 10^{-8}).$$

Truncation error

$$u(x + \Delta x) = \underbrace{u(x) + \Delta x \partial_x u(x)}_{\text{1st order approximation}} + \underbrace{\frac{(\Delta x)^2}{2!} \partial_x^2 u(x) + \dots}_{\text{truncation error}}.$$

$u(x + \Delta x) = \text{approximation} + E(\Delta x) + \text{high order terms.}$

$u(x + \Delta x) \approx u(x) + \Delta x \partial_x u.$

$$E(\Delta x) = \frac{(\Delta x)^{n+1}}{(n+1)!} \partial_x^{n+1} u(x).$$

```
import numpy as np

x = 0
Δx = 0.1
u = np.cos

def dudx(x):
    return -np.sin(x)

def du2dx(x):
    return -np.cos(x)

exact = u(x + Δx)
approximation = u(x) + Δx * dudx(x)
absolute_lower_order_omitted_term = np.abs(np.power(Δx, 2) / 2 * du2dx(x))
truncation_error = np.abs(exact - approximation)
error_attributed_to_other_omitted_terms = np.abs(
    truncation_error - absolute_lower_order_omitted_term
)

First order approximation: 1.0
Exact: 0.9950041652780258
Truncation error: 0.0049958347219741794
Absolute lower order omitted term: 0.0050000000000000001
Error attributed to other terms: 4.165278025821534e-06
```

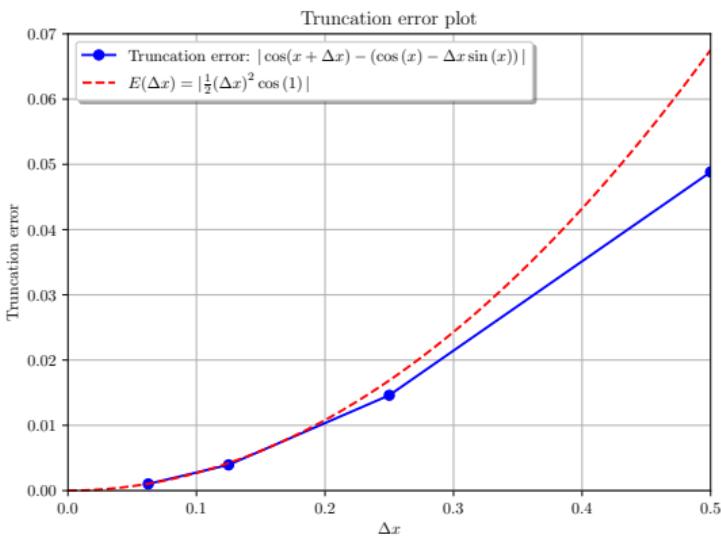


Figure: Truncation error for several values of Δx at $x = 1$

Finite differences as approximations of partial derivatives

Given a known truncation error $E(\Delta x) = O((\Delta x)^n)$ it is possible to estimate the **order** of an approximation.

$$E(\Delta x) \approx C(\Delta x)^n.$$

Taking logarithms of both sides gives

$$\log |E(\Delta x)| = n \log (\Delta x) + \log (C).$$

which is a linear function in Δx . Therefore, one way of approximate the slope n as a gradient is

$$n \approx \frac{\log |E(\max(\Delta x))| - \log |E(\min(\Delta x))|}{\log(\max(\Delta x)) - \log(\min(\Delta x))}.$$

Δx	backward	centered	forward
0.1	-0.670603	-0.705929	-0.741255
0.05	-0.689138	-0.706812	-0.724486
0.025	-0.698195	-0.707033	-0.715872
0.0125	-0.702669	-0.707088	-0.711508

Δx	backward	centered	forward
0.1	0.0365038	0.00117792	0.034148
0.05	0.0179686	0.000294591	0.0173794
0.025	0.00891203	7.36547e-05	0.00876472
0.0125	0.00443777	1.84141e-05	0.00440095

Order of convergence of backward is 1.013379633444132
Order of convergence of centered is 1.9997633049971728
Order of convergence of forward is 0.9853046896368071

```
import numpy as np
from jaxtyping import Array, Float

u = np.cos

def dudx(x):
    return -np.sin(x)

x = np.pi / 4
Δx = np.logspace(start=-3, stop=0, num=4, base=2) / 10

backward: Float[Array, "dim1"] = (u(x) - u(x - Δx)) / Δx
centered: Float[Array, "dim1"] = (u(x + Δx) - u(x - Δx)) / (2 * Δx)
forward: Float[Array, "dim1"] = (u(x + Δx) - u(x)) / Δx

error_backward: Float[Array, "dim1"] = np.abs(dudx(x) - backward)
error_centered: Float[Array, "dim1"] = np.abs(dudx(x) - centered)
error_forward = np.abs(dudx(x) - forward)

def estimate_order(
    Δx: Float[Array, "dim1"], truncation_error: Float[Array, "dim1"]
) → float:
    assert Δx.size == truncation_error.size

    E_max = truncation_error[np.argmax(a=Δx)]
    E_min = truncation_error[np.argmin(a=Δx)]

    return (np.log(np.abs(E_max)) - np.log(np.abs(E_min))) / (
        np.log(Δx.max()) - np.log(Δx.min()))
)

print(f"Order of convergence of backward is {estimate_order(Δx, error_backward)}")
print(f"Order of convergence of centered is {estimate_order(Δx, error_centered)}")
print(f"Order of convergence of forward is {estimate_order(Δx, error_forward)}")
```

Program : Determine order of convergence .

Order of convergence

```
from typing import Callable
import numpy as np
def u(x: float) → float:
    """Sample function
    u:R → R
    x ↦ exp(x^2)
    """
    return np.exp(np.pow(x, 2))
def up(x: float) → float:
    """Derivative of sample function
    u':R → R
    x ↦ 2*x*f(x)
    """
    return 2 * x * u(x)
def ff(u: Callable, Δx: np.array) → np.array:
    """Forward finite difference approximation
    u' = (u(x + Δx) - u(x)) / Δx
    """
    return (u(x + Δx) - u(x)) / Δx
def bf(u: Callable, Δx: np.array) → np.array:
    """Backward finite difference approximation
    u' = (u(x) - u(x - Δx)) / Δx
    """
    return (u(x) - u(x - Δx)) / Δx
def cf(u: Callable, Δx: np.array) → np.array:
    """Centered finite difference approximation
    u' = (u(x + Δx / 2) - u(x - Δx / 2)) / Δx
    """
    return (u(x + Δx / 2) - u(x - Δx / 2)) / Δx
x = 2
exact = up(x)
Δx = np.logspace(start=-16, stop=-1.0, num=16)
```

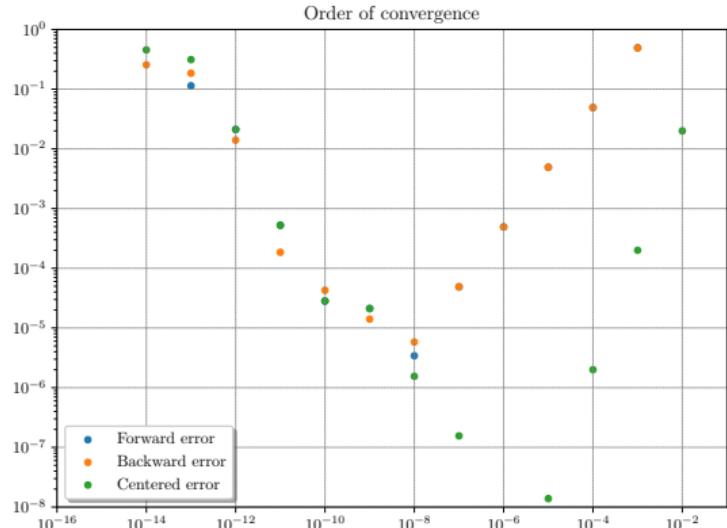


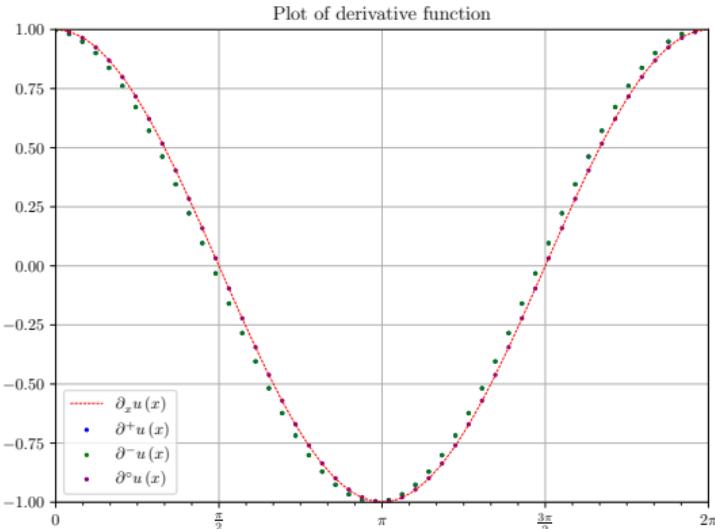
Figure: $\partial^+ u(x), \partial^- u(x), \partial^0 u(x)$ of $u(x) = \exp(x^2)$ for several values of Δx at $x = 2$

Finite differences as approximations of partial derivatives

In the array computation, we do not consider the last element of the difference $u_{i+1} - u_i$. and the first element of the difference $u_{i+1} - u_i$.

$$\partial^+ u(x) = \frac{u_{i+1} - u_i}{\Delta x}, \quad i = 0, \dots, n-1.$$

$$\partial^- u(x) = \frac{u_i - u_{i-1}}{\Delta x}, \quad i = 1, \dots, n.$$



```
import numpy as np  
x, Δx = np.linspace(start=0, stop=2 * np.pi, retstep=True)  
y = np.sin(x)
```

```
forward = (np.roll(y, -1) - y)[-1] / Δx  
backward = (y - np.roll(y, 1))[1:] / Δx  
centered = (np.roll(y, -1) - np.roll(y, 1))[1:-1] / (2 * Δx)  
first_derivative = np.cos(x)  
np.roll(a=u, shift=-1), u, np.roll(a=u, shift=-1) - u
```

u[1]	u[2]	u[3]	u[4]	u[0]
u[0]	u[1]	u[2]	u[3]	u[4]
u[1] - u[0]	u[2] - u[1]	u[3] - u[2]	u[4] - u[3]	u[0] - u[4]

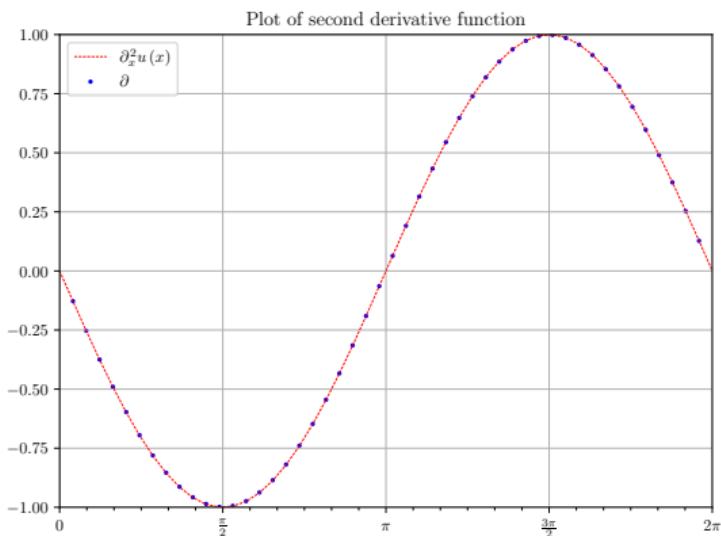
```
np.roll(a=u, shift=1), u, u - np.roll(a=u, shift=1)
```

u[4]	u[0]	u[1]	u[2]	u[3]
u[0]	u[1]	u[2]	u[3]	u[4]
u[4] - u[0]	u[0] - u[1]	u[1] - u[2]	u[2] - u[3]	u[3] - u[4]

Finite differences as approximations of partial derivatives

$$\partial^+ \partial^- u(x) = \frac{u_{i+1} - 2u_i + u_{i-1}}{(\Delta x)^2}, \quad i = 1, \dots, n-1.$$

```
import numpy as np
x, Δx = np.linspace(start=0, stop=2 * np.pi, retstep=True)
y = np.sin(x)
plt.ylim(-1, 1)
plt.grid()
```

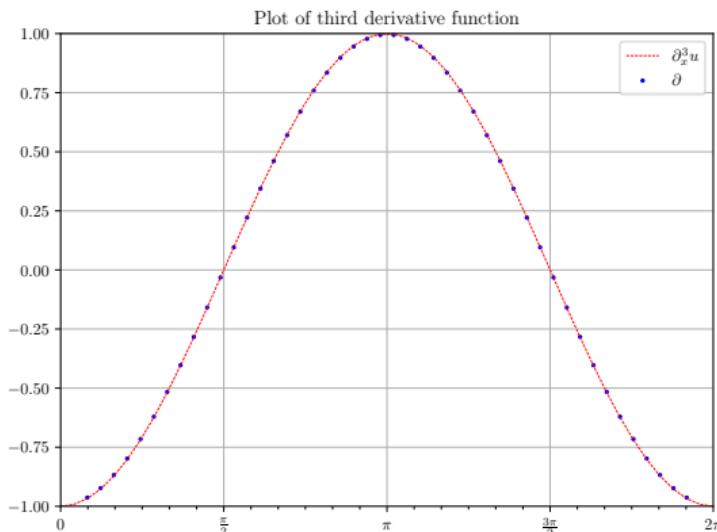


Finite differences as approximations of partial derivatives

$$= \frac{u_{i+2} - 2u_{i+1} + 2u_{i-1} - u_{i-2}}{2(\Delta x)^3}, \quad i = 2, \dots, n-2.$$

```
import numpy as np
x, Δx = np.linspace(start=0, stop=2 * np.pi, retstep=True)
y = np.sin(x)

plt.gca().xaxis.set_minor_locator(plt.MultipleLocator(np.pi / 12))
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(multiple_formatter()))
plt.scatter(x=x[1:-1], y=partial2x, c="blue", s=3, label=r"\partial")
plt.title(label="Plot of second derivative function")
```

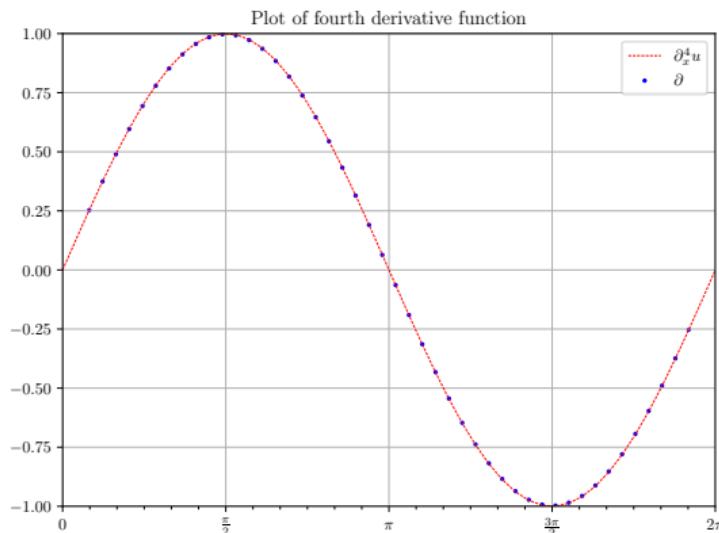


Finite differences as approximations of partial derivatives

$$= \frac{u_{i+2} - 4u_{i+1} + 6u_i - 4u_{i-1} + u_{i-2}}{(\Delta x)^4}, \quad i = 2, \dots, n-2.$$

```
import numpy as np
x, Δx = np.linspace(start=0, stop=2 * np.pi, retstep=True)
y = np.sin(x)

plt.gca().xaxis.set_minor_locator(plt.MultipleLocator(np.pi / 12))
plt.gca().xaxis.set_major_formatter(plt.FuncFormatter(multiple_formatter()))
plt.scatter(x=x[2:-2], y=partial3x, c="blue", s=3, label=r"$\partial$")
plt.title(label="Plot of third derivative function")
```



Complex step method

Definition (Holomorphic function)

Let $D \subset \mathbb{C}$ be a simply connected, open region and $u: D \rightarrow \mathbb{C}$. We say that u is **complex differentiable** at $a \in D$ iff

$$\lim_{z \rightarrow a} \frac{u(z) - u(a)}{z - a}$$

exists. If u is complex differentiable at every point of D , then we say that u is **holomorphic** in D .

The **complex step derivative** approximation is a technique to compute the derivative of a real-valued function $u(x)$. For u analytic,

$$u(x + i\Delta x) = u(x) + i\Delta x \partial_x u(x) + \frac{(i\Delta x)^2}{2!} \partial_x^2 u(x) + \dots$$

$$\operatorname{Re}[u(x + i\Delta x)] + i \operatorname{Im}[u(x + i\Delta x)] \approx u(x) + i\Delta x \partial_x u(x).$$

Comparing imaginary parts of the two sides gives

$$\partial_x u(x) \approx \operatorname{Im} \left[\frac{u(x + i\Delta x)}{\Delta x} \right].$$

Remark

Behind the scenes, the complex step method is a particular case of **automatic differentiation**.

Complex step method

$$u(x + i\Delta x) = u(x) + i\Delta x \partial_x u(x) + \frac{(i\Delta x)^2}{2!} \partial_x^2 u(x) + \dots$$

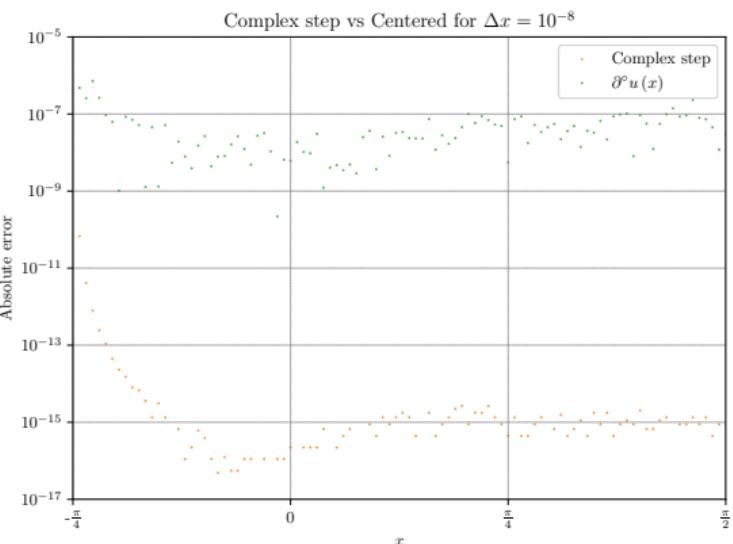
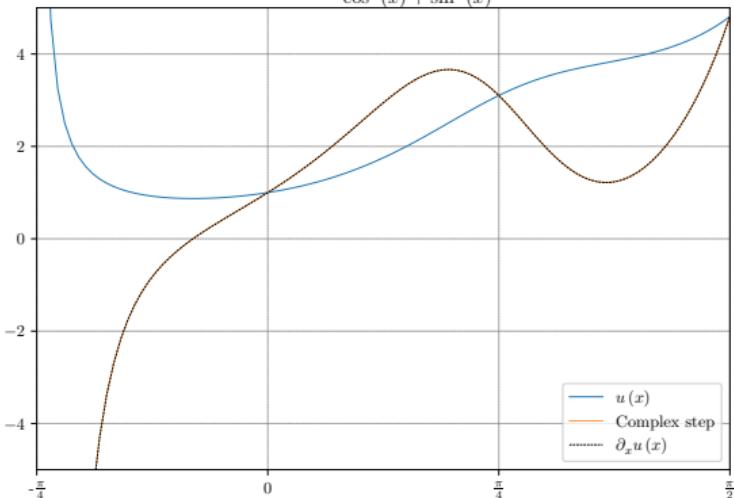
$$\operatorname{Re}[u(x + i\Delta x)] + i \operatorname{Im}[u(x + i\Delta x)] \approx u(x) + i\Delta x \partial_x u(x).$$

Comparing real parts of the two sides gives

$$\partial_x^2 u(x) \approx \frac{2}{\Delta x^2} (u(x) - \operatorname{Re}[u(x + i\Delta x)]).$$

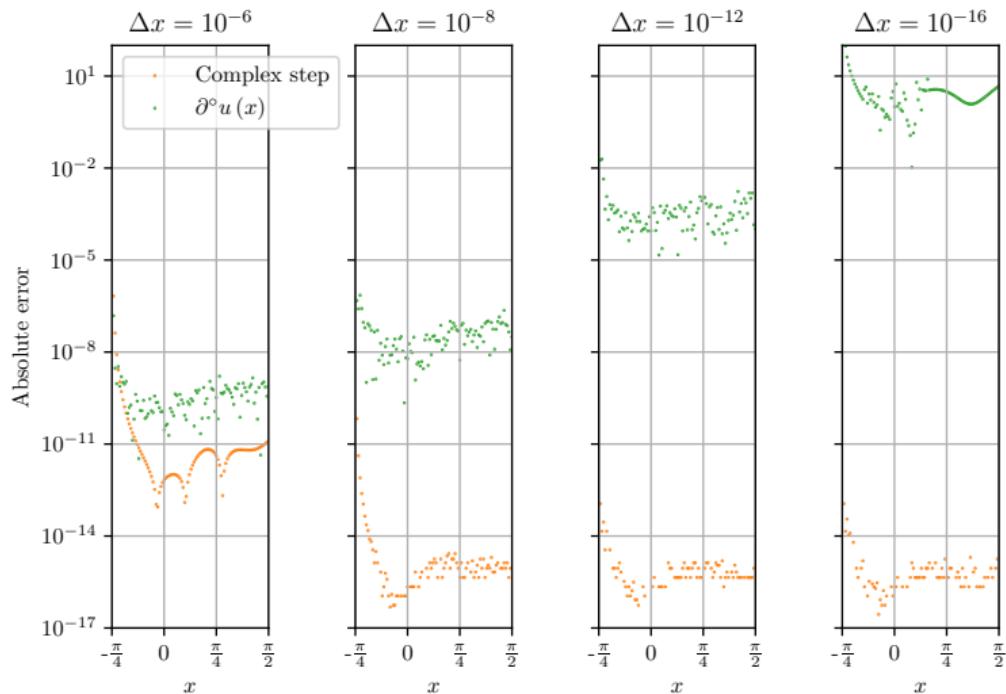
Complex step method

$$u(x) = \frac{\exp(x)}{\cos^3(x) + \sin^3(x)}$$



Complex step method

Complex step vs Centered with varying step size



Solve BVP for ODEs

Let's follow the steps:

- 1 Discretize the domain on which the equation is defined.
- 2 On each **grid point**, replace the derivatives with an approximation, using the values in neighbouring grid points.
- 3 Replace the exact solutions by their approximations.
- 4 Solve the resulting system of equations.

Finite difference for Two-point BVP

We will first see how to find approximations to the derivative of a function, and then how these can be used to solve boundary value problems like

$$\begin{cases} \frac{d^2u}{dx^2} + p(x) \frac{du}{dx} + q(x) u = r(x) & \text{for } a \leq x \leq b, \\ u(a) = u_a, \quad u(b) = u_b \end{cases}.$$

This technique described here is applicable to several other time dependent PDEs, and it is therefore important to try to understand the underlying idea.

Example (Two-point BVP FDM for the 1D Poisson Problem)

Let $f: [0, 1] \rightarrow \mathbb{R}$ be a function. Find a $u: [0, 1] \rightarrow \mathbb{R}$ such that

$$(5) \quad \begin{cases} -\frac{d^2u}{dx^2} = f(x), & x \in (0, 1), \\ u(0) = u_a, \quad u(1) = u_b. \end{cases}$$

Instead of trying to compute $u(x)$ exactly, we will now try to compute a numerical approximation u_Δ of $u(x)$. As many times before we start by defining $n+1$ equally spaced points $\{x_i\}_{i=0}^n$ with a grid size $h = \frac{b-a}{n}$ so that

$$\forall i = 0, 1, \dots, n : x_i := a + ih.$$

Consider a collection of equally spaced points, labeled with an index i , with the physical spacing between them denoted Δx . We can express the first derivative of a quantity a at i as:

$$\frac{\partial a}{\partial x_i} \approx \frac{a_i - a_{i-1}}{\Delta x}$$

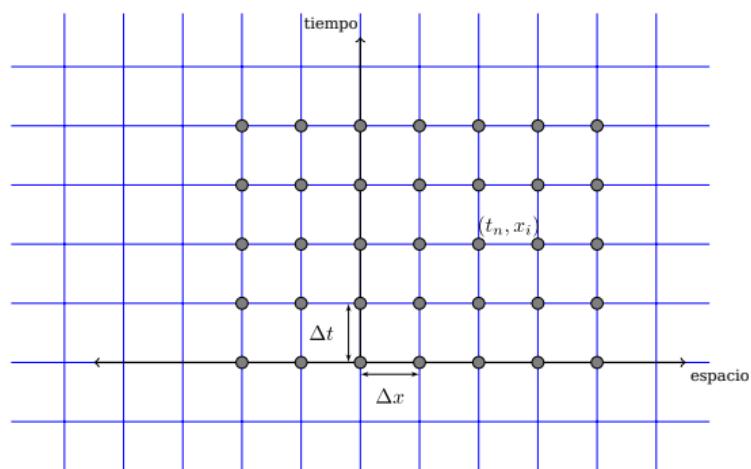
or

$$\frac{\partial a}{\partial x_i} \approx \frac{a_{i+1} - a_i}{\Delta x}$$

$$a_{i+1} = a_i + \Delta x \frac{\partial a}{\partial x} \Big|_i + \frac{1}{2} \Delta x^2 \frac{\partial^2 a}{\partial x^2} \Big|_i + \dots$$

Solving for $\partial a / \partial x|_i$, we see

$$\begin{aligned} \frac{\partial a}{\partial x} \Big|_i &= \frac{a_i - a_{i-1}}{\Delta x} - \frac{1}{2} \Delta x \frac{\partial^2 a}{\partial x^2} \Big|_i + \dots \\ &= \frac{a_i - a_{i-1}}{\Delta x} + \mathcal{O}(\Delta x) \end{aligned}$$



```

import numpy as np
from scipy.sparse import csr_array, diags_array
from scipy.sparse.linalg import spsolve

def fdm_poisson1d_matrix(N: int):
    """Computes the finite difference matrix for the Poisson problem in 1D

    Parameters:
    N (int): Number of grid points :math:`\{x_i\}_{i=0}^N` counting from 0.

    Returns:
    A (scipy.sparse._csr.csr_array): Finite difference sparse matrix

    """
    Δx = 1 / N
    diag = np.concatenate(
        (
            np.ones(shape=1),
            np.full(shape=N - 1, fill_value=2 / Δx**2),
            np.ones(shape=1),
        )
    )
    diag_sup = np.concatenate(
        (np.zeros(shape=1), np.full(shape=N - 1, fill_value=-1 / Δx**2))
    )
    diag_inf = np.flipud(m=diag_sup)

    return diags_array(
        [diag, diag_sup, diag_inf],
        offsets=[0, 1, -1],
        shape=(N + 1, N + 1),
        format="csr",
    )

N = 10
x = np.linspace(start=0, stop=1, num=N + 1)
A = fdm_poisson1d_matrix(N)
F = (2 * np.pi) ** 2 * np.sin(2 * np.pi * x)

```

Program ✎ : `fdmpoisson1d.py`.

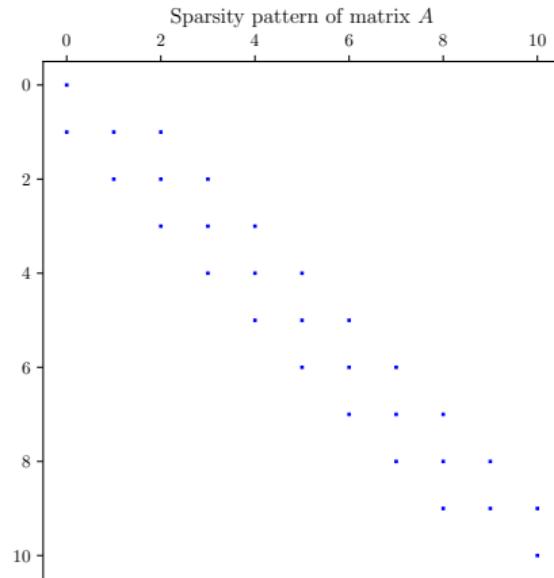
```

xfine = np.linspace(start=0, stop=1, num=10 * N)
# Analytical reference solution
u = np.sin(2 * np.pi * xfine)

# Incorporate boundary condition into rhs vector
F[0], F[-1] = u[0], u[-1]

# Solve AU = F
U = spsolve(A=A, b=F)

```



```

%%MatrixMarket matrix coordinate real general
%
11 11 29
1 1 1
2 1 -9.99999999999999E1
2 2 1.999999999999997E2
2 3 -9.99999999999999E1
3 2 -9.99999999999999E1
3 3 1.999999999999997E2
3 4 -9.99999999999999E1
4 3 -9.99999999999999E1
4 4 1.999999999999997E2
4 5 -9.99999999999999E1
5 4 -9.99999999999999E1
5 5 1.999999999999997E2
5 6 -9.99999999999999E1
6 5 -9.99999999999999E1
6 6 1.999999999999997E2
6 7 -9.99999999999999E1
7 6 -9.99999999999999E1
7 7 1.999999999999997E2
7 8 -9.99999999999999E1
8 7 -9.99999999999999E1
8 8 1.999999999999997E2
8 9 -9.99999999999999E1
9 8 -9.99999999999999E1
9 9 1.999999999999997E2
9 10 -9.99999999999999E1
10 9 -9.99999999999999E1
10 10 1.999999999999997E2
10 11 -9.99999999999999E1
11 11 1

```

Program  : poissonA.mm.

```

%%MatrixMarket matrix coordinate real general
%
1 11 10
1 2 2.3204831651684845E1
1 3 3.754620631564544E1
1 4 3.754620631564544E1
1 5 2.320483165168485E1
1 6 4.8347117754578846E-15
1 7 -2.3204831651684856E1
1 8 -3.754620631564544E1
1 9 -3.754620631564544E1
1 10 -2.3204831651684856E1
1 11 -2.4492935982947064E-16

```

Program  : poissonF.mm.

```

%%MatrixMarket matrix coordinate real general
%
1 11 10
1 2 6.07510379673303E-1
1 3 9.829724428297576E-1
1 4 9.829724428297576E-1
1 5 6.07510379673303E-1
1 6 -5.1532467934608046E-17
1 7 -6.075103796733031E-1
1 8 -9.829724428297577E-1
1 9 -9.829724428297578E-1
1 10 -6.075103796733033E-1
1 11 -2.4492935982947064E-16

```

Program  : poissonU.mm.

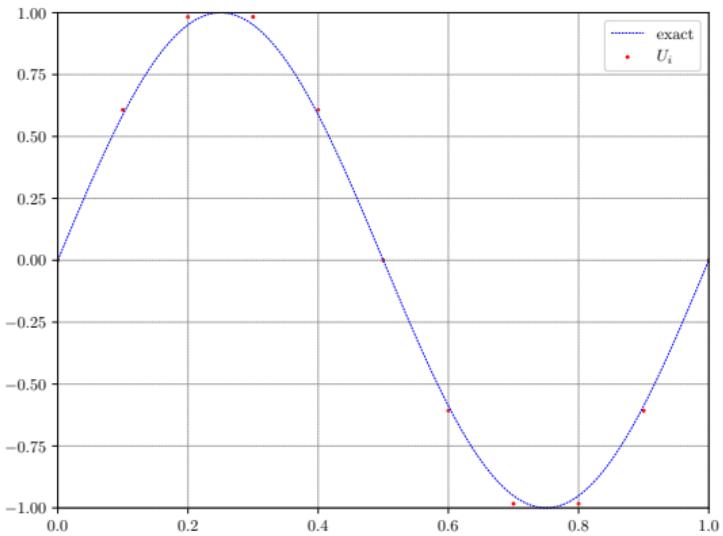


Figure: Solution.

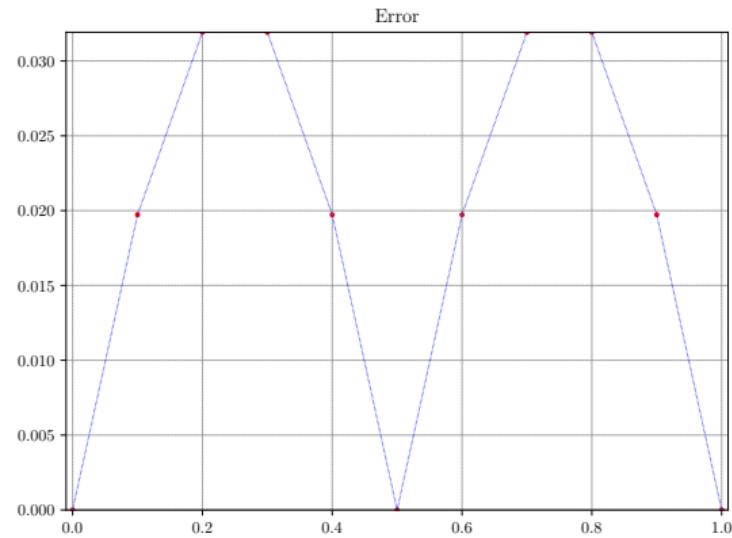


Figure: Error.

```

import numpy as np

from scipy.sparse import csr_array, diags_array
from scipy.sparse.linalg import spsolve

def tridiag(p: np.ufunc, q: np.ufunc, N: int):
    """
    Help function
    Returns a tridiagonal matrix A of dimension N+1 x N+1.
    """
    Δx = 1 / N
    diag = np.concatenate(
        (
            np.ones(shape=1),
            np.full(shape=N - 1, fill_value=-2 + Δx**2 * q),
            np.ones(shape=1),
        )
    )
    diag_sup = np.concatenate(
        (np.zeros(shape=1), np.full(shape=N - 1, fill_value=1 + Δx / 2 * p))
    )
    diag_inf = np.concatenate(
        (np.full(shape=N - 1, fill_value=1 - Δx / 2 * p), np.zeros(shape=1))
    )

    return diags_array(
        [diag, diag_sup, diag_inf],
        offsets=[0, 1, -1],
        shape=(N + 1, N + 1),
        format="csr",
    )

N = 4 # Number of intervals
x, Δx = np.linspace(start=0, stop=1, num=N + 1, retstep=True)

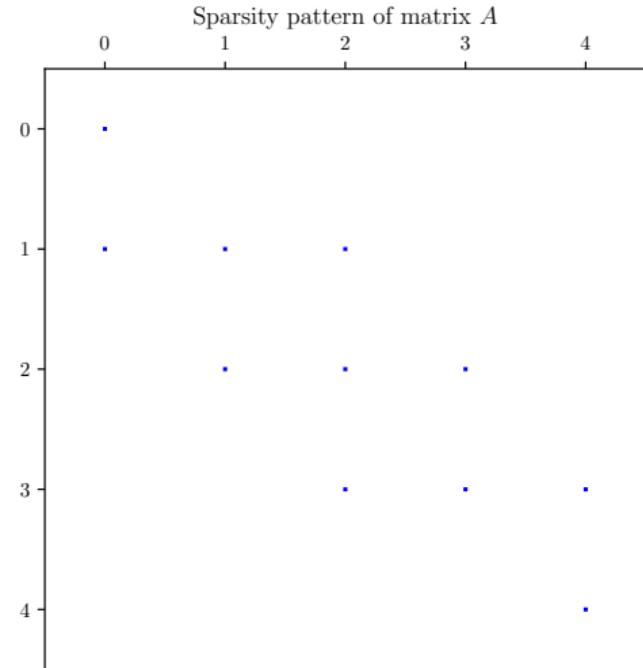
p = 2
q = -3
r = 9 * x

A = tridiag(p, q, N)
b = Δx**2 * r
b[0] = 1
b[N] = np.exp(-3) + 2 * np.exp(1) - 5

U = spsolve(A=A, b=b) # Solve the equation

```

Program : twopointboundary.py.



```
%%MatrixMarket matrix coordinate real general
%
5 5 11
1 1 1
2 1 7.5E-1
2 2 -2.1875
2 3 1.25
3 2 7.5E-1
3 3 -2.1875
3 4 1.25
4 3 7.5E-1
4 4 -2.1875
4 5 1.25
5 5 1
```

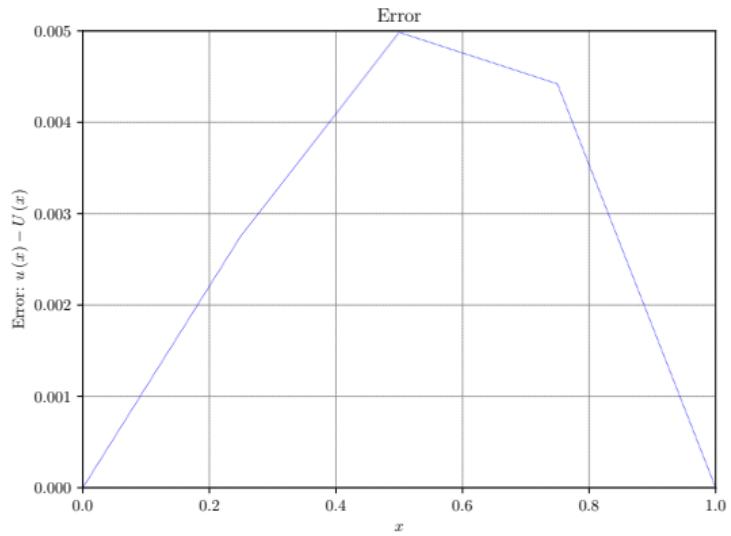
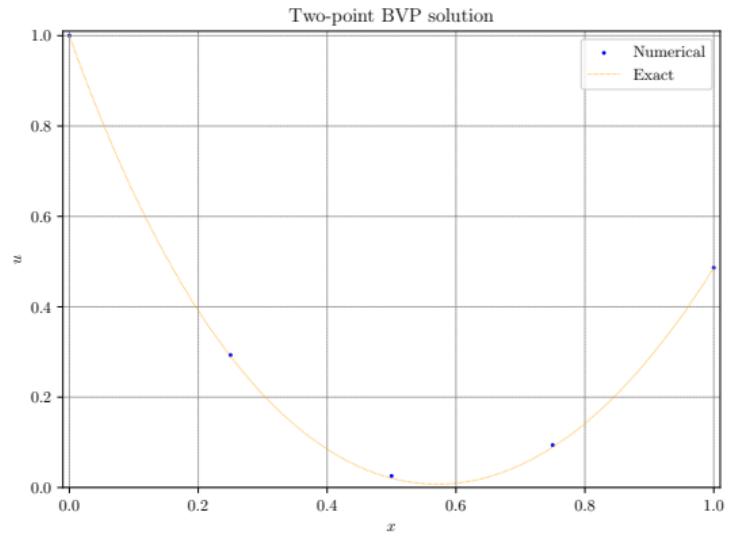
Program  : `twopointboundaryA.mm`.

```
%%MatrixMarket matrix coordinate real general
%
1 5 5
1 1 1
1 2 1.40625E-1
1 3 2.8125E-1
1 4 4.21875E-1
1 5 4.863507252859538E-1
```

Program  : `twopointboundaryb.mm`.

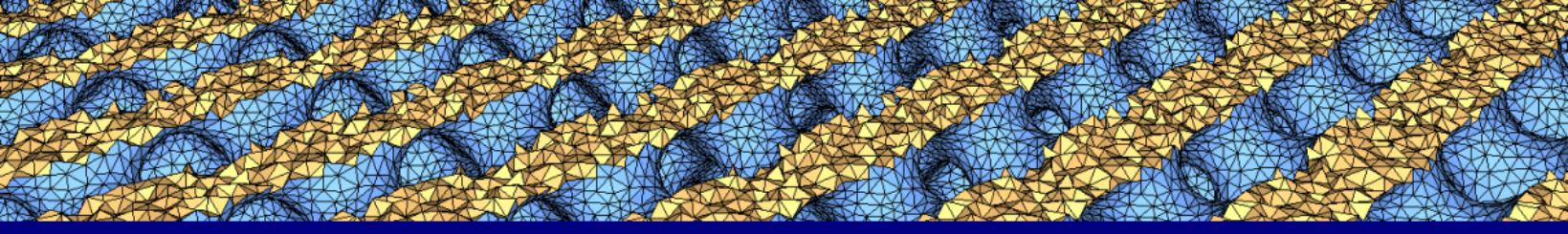
```
%%MatrixMarket matrix coordinate real general
%
1 5 5
1 1 1
1 2 2.931756779400817E-1
1 3 2.5557436395142973E-2
1 4 9.382010692745119E-2
1 5 4.863507252859538E-1
```

Program  : `twopointboundaryU.mm`.



$$u(x, t) = \sum_{\nu=0}^{\infty} a_{\nu}^0 e^{i\nu x} e^{\nu^2 t}$$

$$U_n^m = \sum_{\nu=0}^N a_{\nu}^m e^{i\nu nh} := \sum_{\nu=0}^N a_{\nu}^0 \omega_{\nu}^m e^{i\beta_{\nu} n}.$$



Fourier stability analysis

Fourier stability analysis

We will introduce some simple spatial discretizations for the periodic constant-coefficient advection-diffusion problem on a uniform grid $\Omega_h = \{x_1, \dots, x_m\}$ with grid points $x_j = jh$ and mesh width $h = \frac{1}{m}$.

Discrete Fourier decomposition is an important tool to analyze **linear difference schemes** with spatial periodic conditions. It can be used to study fundamental properties like **stability**, **dissipation** and **dispersion**.

Definition

Fourier modes

$$\forall k \in \mathbb{Z} : \varphi_k(x) = \exp(2\pi i k x).$$

These modes form an orthonormal basis for $L^2[0, 1]$ space, i.e.,

$$\forall v \in L^2[0, 1] : \sum_{k \in \mathbb{Z}} \alpha_k \varphi_k(x),$$

where the right-hand side is a convergent series, which we now call the Fourier series. The Fourier coefficients are given by $\alpha_k = \langle \varphi_k, v \rangle$.

Definition

Discrete Fourier modes

$$\forall k \in \mathbb{Z} : \phi_k(x) = (\varphi_k(x_1), \dots, \varphi_k(x_m)) \in \mathbb{C}^m.$$

The Courant-Friedrichs-Lowy (CFL) condition

Explicit schemes for the advection equation $\partial_t u + c \partial_x u = 0$ give rise to step size restrictions (stability conditions) of the form

$$|c| \frac{\Delta t}{\Delta x} \leq C,$$

with C an appropriate positive constant independent of Δx and Δt .

Remark

In the beginning of numerical analysis, finite difference approximations were used to prove the existence of PDE solutions. For **convergence** of finite difference approximations are necessary in order to guarantee that the mathematical domain of dependence of a PDE problem lies within the numerical counterpart of the finite difference method.

$$U_j^{n+1} = \sum_{k=-r}^r \gamma_k U_{j+k}^n.$$

A necessary condition for stability is that the mathematical domain of dependence of PDE is contained in the numerical domain of dependence.

Fourier stability analysis

In the 1940's, John von Neumann introduced Fourier analysis in the theory of finite difference schemes for time-dependent PDEs. We are interested in the propagation of small errors at different grid points. If these errors are not controlled at each stage of the time iteration, they can grow and create a solution completely different from the desired solution at a later time. An initial pulse which has bounded size but oscillates with frequency k . Take a complex exponential for some $k \in \mathbb{R}$

$$e^{ikx} = \cos(kx) + i \sin(kx).$$

The size of the complex-valued pulse is given by its modulus, and

$$|e^{ikx}| = |\cos(kx) + i \sin(kx)| = 1.$$

The **growth in size** when applying the scheme once to e^{ikx} , captured by an amplification constant called the **growth factor**. At time step n , for some $j \in \mathbb{Z}$ we take

$$U_j^n(k) \equiv \lambda(k)^n e^{ik(j\Delta x)} \implies \begin{cases} U_j^{n+1}(k) &= \lambda(k)^{n+1} e^{ik(j\Delta x)}. \\ U_{j+1}^n(k) &= \lambda(k)^n e^{ik(j+1)\Delta x} e^{ik\Delta x}. \\ U_{j-1}^n(k) &= \lambda(k)^n e^{ik(j-1)\Delta x} e^{-ik\Delta x}. \\ U_j^{n-1}(k) &= \lambda(k)^{n-1} e^{ik(j-1)\Delta x}. \end{cases}$$

Remark

$$\operatorname{Re}(e^{ik\Delta x}) = \cos(k\Delta x) = \frac{e^{ik\Delta x} + e^{-ik\Delta x}}{2}, \quad \operatorname{Im}(e^{ik\Delta x}) = \sin(k\Delta x) = \frac{e^{ik\Delta x} - e^{-ik\Delta x}}{2i}.$$

Example (Transport equation on a periodic domain)

For some constant $c > 0$ and some continuous and bounded function $g: [0, 1] \rightarrow \mathbb{R}$.

$$\begin{cases} \partial_t u + c\partial_x u = 0, & x \in (0, 1), t > 0. \\ u(0, t) = u(1, t), & t > 0. \\ u(x, 0) = g(x), & x \in [0, 1]. \end{cases}$$

We consider the **implicit** FDM

$$(6) \quad \begin{cases} \frac{v_j^{m+1} - v_j^m}{\Delta t} + c \frac{v_j^{m+1} - v_{j-1}^{m+1}}{\Delta x} = 0 & j = 1, \dots, n+1, m = 0, 1, \dots \\ v_0^{m+1} = v_{n+1}^{m+1} & m = 0, 1, \dots \\ v_j^0 = g(x_j) & j \in \mathbb{Z}. \end{cases}$$

Show that for any choice of $\Delta t, \Delta x > 0$, any solution of (6) satisfies

$$\inf_{x \in [0, 1]} g(x) \leq v_j^m \leq \sup_{x \in [0, 1]} g(x).$$

Fourier stability analysis

Solution

Let J be a point at which $\{v_J^{m+1}\}_{j=1}^{n+1}$ attains its maximum. Then

$$v_J^{m+1} = v_J^m - c \frac{\Delta t}{\Delta x} (v_J^{m+1} - v_{J-1}^{m+1}).$$

By assumption, $c > 0$, and by the choice of J we have $v_J^{m+1} - v_{J-1}^{m+1} \geq 0$. Iterating the inequality over all m yields

$$v_j^m \leq \max_{j=0, \dots, n+1} v_j^0 \leq \sup_{x \in [0,1]} g(x).$$

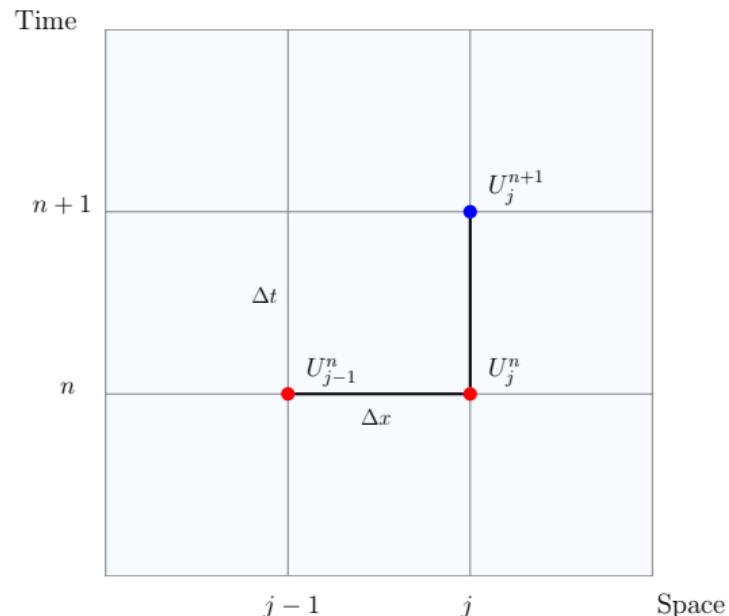
Fourier stability analysis

Example (First-Order Upwind (FOU))

Proposed by Richard Courant, Eugene Isaacson and Mina Rees. Let $c(x, t) = c > 0$.

$$0 = \frac{U_j^{n+1} - U_j^n}{\Delta t} + c \frac{U_j^n - U_{j-1}^n}{\Delta x}.$$

$$U_j^{n+1} = U_j^n - r (U_j^n - U_{j-1}^n), \quad r = c \frac{\Delta t}{\Delta x}.$$



Fourier stability analysis

Theorem (Stability analysis for FOU scheme)

$$r \in (0, 1] \iff |\lambda(k)| \leq 1.$$

Proof.

$$U_j^{n+1} = U_j^n - r(U_j^n - U_{j-1}^n).$$

$$\lambda(k)^{n+1} e^{ik(j\Delta x)} = \lambda(k)^n e^{ik(j\Delta x)} - r\lambda(k)^n (e^{ik(j\Delta x)} - e^{ik(j-1)\Delta x}).$$

$$\lambda(k) = 1 - r(1 - e^{-ik\Delta x}).$$

$$\begin{aligned} |\lambda(k)| &= |1 - r + re^{-ik\Delta x}| \\ &\leq |1 - r| + |re^{-ik\Delta x}| \\ &= |1 - r| + r \\ &\leq 1 \iff r \in (0, 1]. \end{aligned}$$

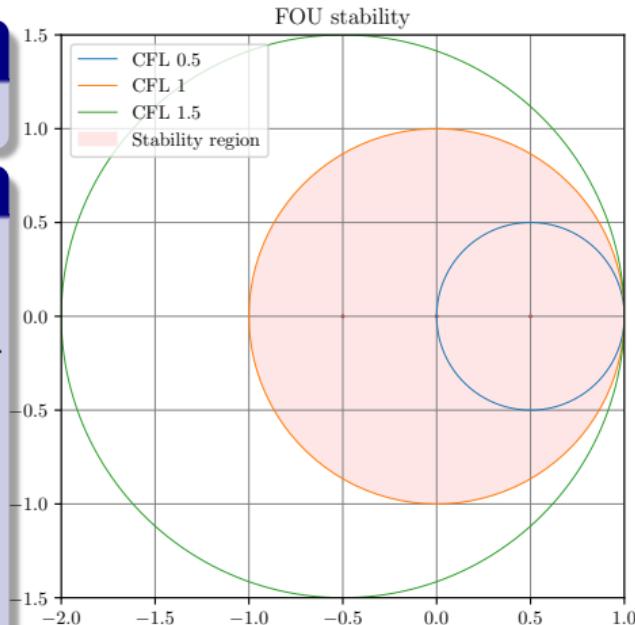


Figure: The FOU scheme is **dissipative**.

```

label = "FOU scheme for 1D linear advection PDE"
space = [0, 10]
spacepoints = [6]
speed = 5
time = [0, 1]
timesteps = 5
piecewise = [2, 4]

```

Program  : Parameters file `fou.toml`.

```

from tomllib import load
import numpy as np
with open(file="fou.toml", mode="rb") as f:
    data = load(f)
c = data["speed"]
x, Δx = np.linspace(
    start=data["space"][0],
    stop=data["space"][1],
    num=data["spacepoints"][0],
    retstep=True,
)
t, Δt = np.linspace(
    start=data["time"][0], stop=data["time"][1], num=data["timesteps"], retstep=True
)
cfl = c * Δt / Δx
u = np.where((data["piecewise"][0] ≤ x) & (data["piecewise"][1] ≤ 4), 1.0, 0)
u = np.insert(u, 0, u[0]) # left hand side ghost node
u = np.append(u, u[-1]) # right hand side ghost node
print(f"CFL number: {cfl}")
print("u0      u1      u2      u3      u4      u5      u6")
for timestep in t:
    print(u)
    u[1:] -= cfl * (u - np.roll(u, 1))[1:]
    u[0] = u[1]
    u[-1] = u[-2]

```

Program  : `fou.py`.

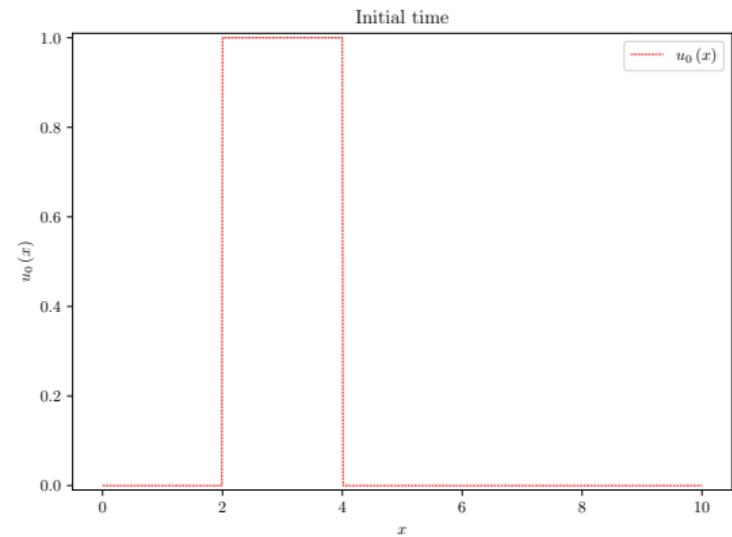
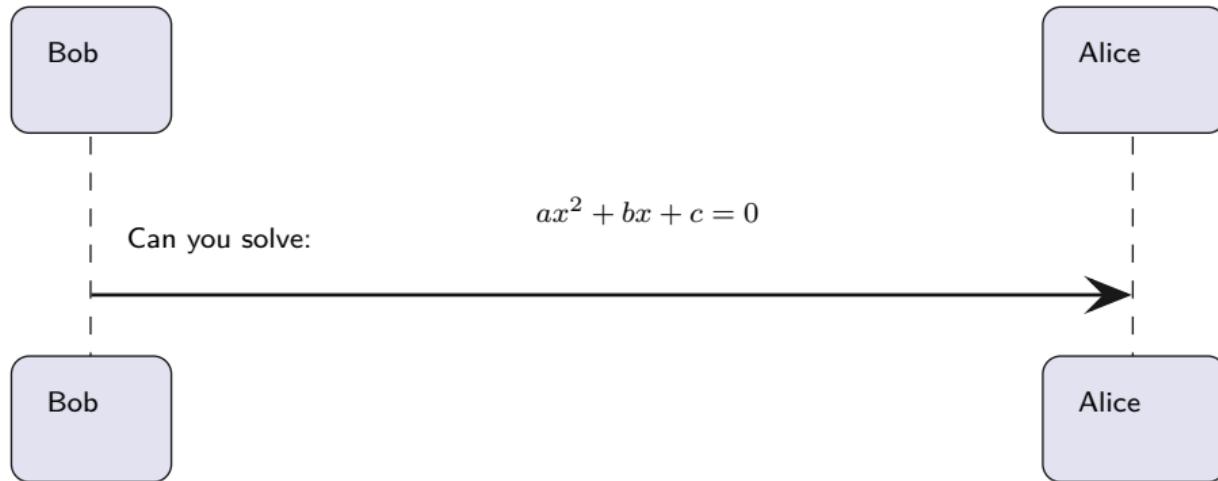


Figure: Initial profile.

CFL number: 0.625

u0	u1	u2	u3	u4	u5	u6
[0. 0. 1. 1. 1. 1. 1.]	[0. 0. 0.375 1. 1. 1. 1.]	[0. 0. 0.140625 0.609375 1. 1. 1.]	[0. 0. 0.052734 0.316406 0.755859 1. 1.]	[0. 0. 0.019775 0.151611 0.481201 0.847412 1.]		

Flow chart

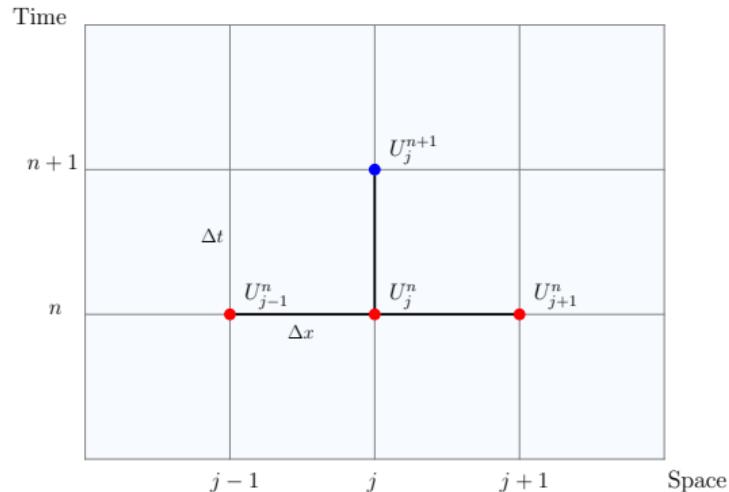


Fourier stability analysis

Example (Forward-Time Central-Space (FTCS))

$$0 = \frac{U_j^{n+1} - U_j^n}{\Delta t} + c \frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x}.$$

$$U_j^{n+1} = U_j^n - \frac{r}{2} (U_{j+1}^n - U_{j-1}^n), \quad r = c \frac{\Delta t}{\Delta x}.$$



Fourier stability analysis

Theorem (Stability analysis for FTCS scheme)

$$\forall r > 0 \iff |\lambda(k)| > 1.$$

Proof.

$$\begin{aligned} U_{j+1}^n &= U_j^n - \frac{r}{2} (U_{j+1}^n - U_{j-1}^n). \\ \lambda(k)^{n+1} e^{ik(j\Delta x)} &= \lambda(k)^n e^{ikj\Delta x} - \frac{r}{2} \lambda(k)^n (e^{ik(j+1)\Delta x} - e^{ik(j-1)\Delta x}). \\ \lambda(k) &= 1 - \frac{r}{2} (e^{ik\Delta x} - e^{-ik\Delta x}). \\ \lambda(k) &= 1 - ir \sin(k\Delta x). \\ |\lambda(k)|^2 &= 1 + r^2 \sin^2(k\Delta x) \\ &> 1 \iff \forall r > 0. \end{aligned}$$

□

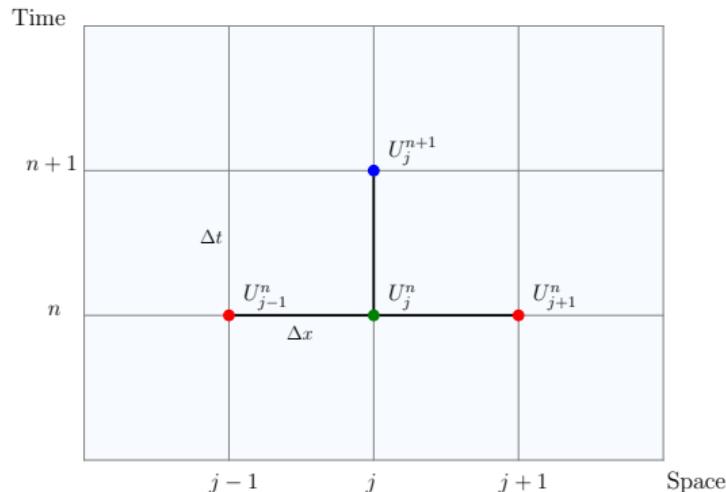
Remark

We say that the FTCS scheme is unconditionally unstable.

Fourier stability analysis

Example (Lax-Friedrichs scheme)

$$0 = \frac{U_j^{n+1} - \frac{U_{j+1}^n + U_{j-1}^n}{2}}{\Delta t} + c \frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x}.$$
$$U_j^{n+1} = \frac{U_{j+1}^n + U_{j-1}^n}{2} - \frac{r}{2} (U_{j+1}^n - U_{j-1}^n), \quad r = c \frac{\Delta t}{\Delta x}.$$



Fourier stability analysis

Theorem (Stability analysis for Lax-Friedrichs)

$$r \in (0, 1] \iff |\lambda(k)| \leq 1.$$

Proof.

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

$$\lambda(k)^{n+1} e^{ik(j\Delta x)} = \frac{\lambda(k)^n}{2} (e^{ik(j+1)\Delta x} + e^{ik(j-1)\Delta x}) - \frac{r\lambda(k)^n}{2} (e^{ik(j+1)\Delta x} - e^{ik(j-1)\Delta x}).$$

$$\lambda(k) = \frac{e^{ik\Delta x} + e^{-ik\Delta x}}{2} - r \frac{e^{ik\Delta x} - e^{-ik\Delta x}}{2}.$$

$$\lambda(k) = \cos(k\Delta x) - ir \sin(k\Delta x).$$

$$|\lambda(k)|^2 = \cos^2(k\Delta x) + r^2 \sin^2(k\Delta x).$$

$$|\lambda(k)|^2 = 1 - \sin^2(k\Delta x) + r^2 \sin^2(k\Delta x).$$

$$|\lambda(k)|^2 = 1 - (1 - r^2) \sin^2(k\Delta x).$$

$$\leq 1 \iff r \in (0, 1].$$

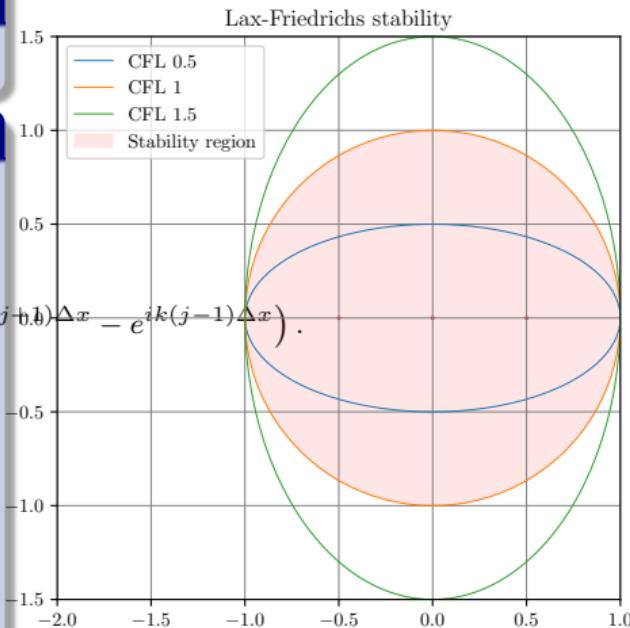


Figure: The Lax-Friedrichs scheme is oscillates.

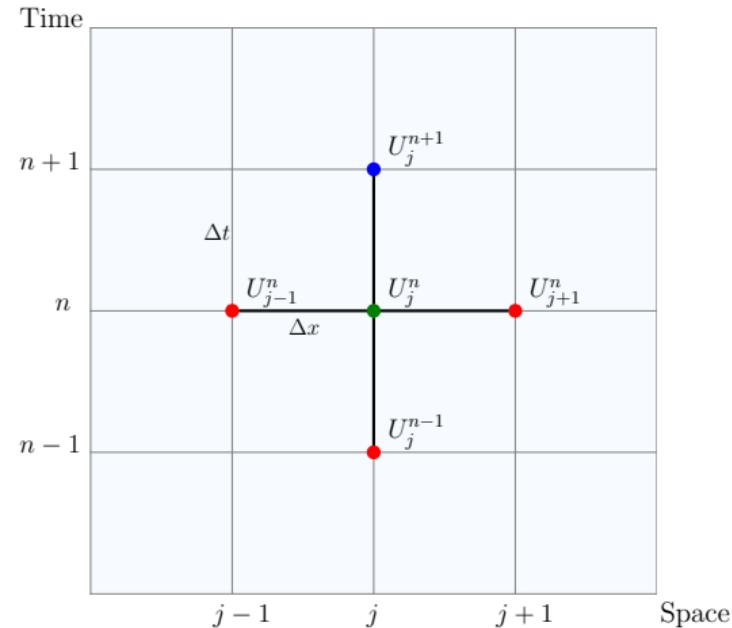


Fourier stability analysis

Example (Leapfrog scheme)

$$0 = \frac{U_j^{n+1} - U_j^{n-1}}{2\Delta t} + c \frac{U_{j+1}^n - U_{j-1}^n}{2\Delta x}.$$

$$U_j^{n+1} = U_j^{n-1} - \frac{c\Delta t}{\Delta x} \frac{U_{j+1}^n - U_{j-1}^n}{2}.$$



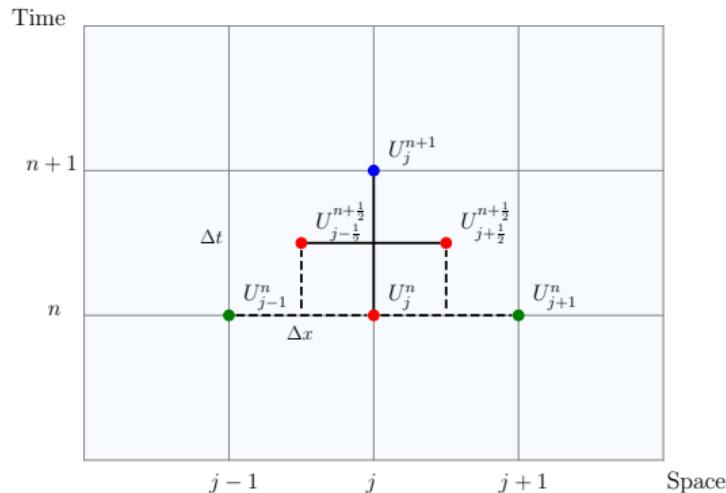
$$|U_j^{n+1}| = \left| U_j^{n-1} - r \frac{U_{j+1}^n - U_{j-1}^n}{2} \right| = \left| 0 - r \frac{e^{ikj\Delta x} e^{ik\Delta x} - e^{ikj\Delta x} e^{-ik\Delta x}}{2} \right|.$$

Fourier stability analysis

Example (Lax-Wendroff scheme)

$$0 = \frac{U_i^{n+1} - U_i^{n-1}}{2\Delta t} + c \frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x}.$$

$$U_j^{n+1} = U_j^n - \frac{c\Delta t}{\Delta x} \left(U_{j+\frac{1}{2}}^{n+\frac{1}{2}} - U_{j-\frac{1}{2}}^{n+\frac{1}{2}} \right).$$

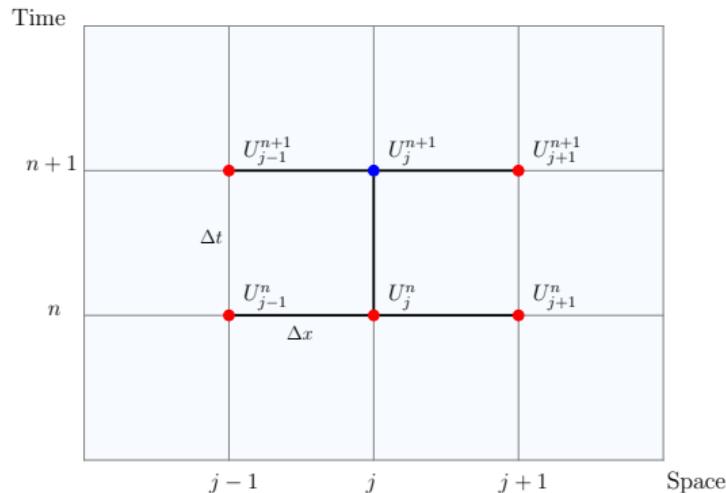


Fourier stability analysis

Example (Crank-Nicolson scheme)

$$0 = \frac{U_i^{n+1} - U_i^{n-1}}{2\Delta t} + c \frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x}.$$

$$U_j^{n+1} = U_j^n - \frac{c\Delta t}{\Delta x} \frac{U_{j+1}^n - U_{j-1}^n + U_{j+1}^{n+1} - U_{j-1}^{n+1}}{4}.$$



Theorem (Lax-Richtmyer equivalence)

A consistent finite-difference scheme for a PDE for which the initial-value problem is well posed is convergent iff it is stable.

Fourier stability analysis

Since it underlies the mass conservation law, the advection equation

$$\partial_t u + \nabla \cdot (cu) = 0$$

is said to be in **conservative form** (also called **conservation** or **flux form**).

From the identity

$$\nabla \cdot (cu) = u \nabla \cdot c + c \cdot \nabla u,$$

$$\partial_t u + a \cdot \nabla u = 0$$

if the vector field c is divergence-free, that is if

$$\nabla \cdot c = 0.$$

If we define the characteristics $(\xi(t), t)$ in the (x, t) space of solutions of the ODE

$$\frac{d\xi(t)}{dt} = c(\xi(t), t),$$

then it follows that

$$\frac{du(\xi(t), t)}{dt} = 0,$$

and hence the solution $u(x, t)$ is constant along the characteristics.

Integral form of the Conservation Law

In general, a **conservation law** in one dimension takes the form

$$\partial_t u + \partial_x f(u) = 0.$$

Let $\Omega_x \subset \mathbb{R}$ be a domain and $u \in C(\Omega_x \times \Omega_t)$. Suppose that $\partial_t u \in C(\Omega_x \times \Omega_t)$ and exists $g(x)$ and $h(x)$ such that

$$\forall t \in \Omega_t : |u(x, t)| \leq g(x)$$

and

$$|\partial_t u| \leq h(x)$$

$$\frac{\partial u(x,t)}{\partial t} + \frac{\partial f(u(x,t))}{\partial x} = 0.$$

Integrate with respect to x

$$\int_a^b \left(\frac{\partial u(x,t)}{\partial t} + \frac{\partial f(u(x,t))}{\partial x} \right) dx = \int_a^b 0 dx.$$

By linearity of integral

$$\int_a^b \frac{\partial u(x,t)}{\partial t} dx + \int_a^b \frac{\partial f(u(x,t))}{\partial x} dx = 0.$$

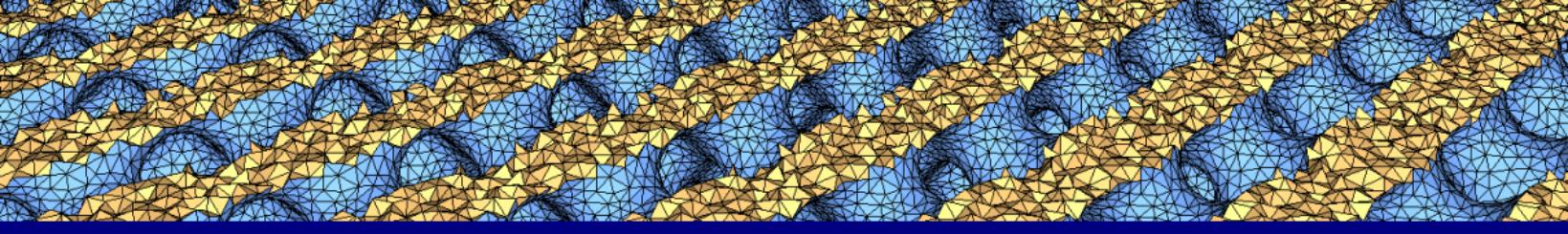
Deriving under the integral sign

$$\frac{\partial}{\partial t} \int_a^b u(x,t) dx + \int_a^b \frac{\partial f(u(x,t))}{\partial x} dx = 0.$$

Applying the Fundamental theorem of calculus

$$\int_a^b u(x,t) dx - \int_a^b u(x,0) dx + \int_0^t \int_a^b f(a,\tau) d\tau dx - \int_0^t \int_a^b f(b,\tau) d\tau dx = 0.$$

$$\int_a^b u(x,0) dx + \int_0^t \int_a^b f(a,\tau) d\tau dx - \int_0^t \int_a^b f(b,\tau) d\tau dx = \int_a^b u(x,t) dx.$$



Basic definitions

Definition (Partial differential equation (PDE))

It is an equation that involves an **unknown function** u and its partial derivatives together with the **independent variables**. It is written as

$$(7) \quad \mathcal{L}(\text{independent variables, } u, \text{ partial derivatives of } u) = 0.$$

Definition (Domain)

A domain Ω is an **open** and **connected** subset of \mathbb{R}^d having a piecewise linear boundary of class C^1 .

Remark

From now on Ω will always be a domain.

Definition (Classical PDE solution)

It is a sufficiently smooth function $u: \Omega \rightarrow \mathbb{R}$ that satisfies (7) for any $x \in \Omega$.

Definition (Auxiliary condition)

An **auxiliary condition** in a general solution is an equality that specifies the value of the unknown function on a subset of Ω .

Definition (Initial Value Problem (IVP))

Let $u: \Omega \times [0, T] \rightarrow \mathbb{R}$ be the solution to (7). An **initial value problem** is a differential equation together with a set of auxiliar conditions that specify the solution and/or its derivatives at $t = 0$.

Example (IVP for the Korteweg-de Vries equation)

$$(8) \quad \begin{cases} \partial_t u + \partial_x^3 u + 6u\partial_x u = 0 & \text{for } (x, t) \in \Omega \times [0, T], \\ u(x, 0) = u_0(x) & \text{for } x \in \Omega. \end{cases}$$

Definition (Boundary Value Problem (BVP))

Let $u: \Omega \rightarrow \mathbb{R}$ be a solution of (7). A **boundary value problem** is a differential equation together with a set of auxiliary conditions that specify the solution and/or its derivatives in $\partial\Omega$.

Example (Laplace's equation with Inhomogeneous Dirichlet Boundary Conditions)

$$\begin{cases} \Delta u = 0 & \text{for } (x, y) \in \Omega, \\ u = f & \text{for } \partial\Omega. \end{cases}$$

Example (Three widely used boundary conditions for a $u(x)$ on $[a, b]$)

- **Dirichlet** boundary conditions are $u(a) = u(b) = 0$.
- **Neumann** boundary conditions are $u'(a) = u'(b) = 0$.
- **Periodic** boundary conditions are $u(a) = u(b)$ and $u'(a) = u'(b)$.

Definition (Well-posed problem)

It is an EDP with auxiliary conditions that the conditions holds

Existence for a given choice of auxiliary conditions, there exists a solution to the PDE that satisfies it.

Uniqueness there is only one solution.

Stability if we perturb slightly the auxiliary condition, then the new solution does not change much with respect to the original solution.

Theorem (Malgrange-Ehrenpreis (1950))

Every constant-coefficients linear PDE on \mathbb{R}^d can be solved.

Example (Lewy operator (1957))

Not every linear PDE with polynomial coefficients has a solution.

$$\partial_x u + i\partial_y u - 2i(x + iy)\partial_t u = f(t).$$

Proof.

See <https://people.maths.ox.ac.uk/trefethen/pdectb/lewy2.pdf>.



Example (Mizohata operator (1962))

$$\partial_x u + ix\partial_y u = g(x, y).$$

Definition (Bulk integrals)

Integrate a function $f(x)$ on some domain $\Omega \subset \mathbb{R}^d$.

$$\int_{\Omega} \cdots \int f(x) dx$$

where $dx = dx_1 \dots dx_d$.

Definition (Flux integrals)

Let S be an orientable surface with its well-defined unit normal vector field n varying continuously across S .

$$\iint_S F \cdot n dS.$$

For a closed surfaces S which are the boundary of a domain $\Omega \subset \mathbb{R}^3$, we will denote the surface by $\partial\Omega$.

$$\iint_{\partial\Omega} F \cdot n dS.$$

Definition (Integrable function)

A function f defined on a domain $\Omega \subset \mathbb{R}^d$ is **integrable** on Ω iff

$$\int_{\Omega} \cdots \int |f(x)| dx$$

exists and is a finite number.

Definition (Locally integrable function)

A function $f: \mathbb{R}^d \rightarrow \mathbb{R}$ is **locally integrable** iff for every $K \subset \mathbb{R}^d$ which is closed and bounded, we have

$$\int_K \cdots \int |f(x)| dx < \infty.$$

Theorem (Divergence)

If F be a smooth vector field on a bounded domain with outer normal, then

$$\iiint_{\Omega} \operatorname{div} F dx dy dz = \iint_{\partial\Omega} F \cdot n dS.$$

Theorem (Green)

If $P(x, y)$ and $Q(x, y)$ be smooth functions on a bounded 2D domain Ω and $C = \partial\Omega$, then

$$\iint_{\Omega} \left(\frac{\partial Q(x, y)}{\partial x} - \frac{\partial P(x, y)}{\partial y} \right) dx dy = \int_C (P(x, y) dx + Q(x, y) dy).$$

Theorem (Componentwise Divergence)

If $f = f(x)$ is smooth function on a bounded domain $\Omega \subset \mathbb{R}^3$, then

$$\forall i = 1, 2, 3 : \iiint_{\Omega} f_{x_i}(x) dx = \iint_{\partial\Omega} f n_i dS$$

where n_i denotes the i -th component of the outer unit normal n .

Theorem (Integration by Parts)

If f and g are smooth functions of one variable x , then

$$\int_a^b f(x) g'(x) dx = [f(x) g(x)] \Big|_a^b - \int_a^b f'(x) g(x) dx.$$

Theorem (Vector Field Integration by Parts Formula)

$$\iiint_{\Omega} u \cdot \nabla u \, dx = - \iiint_{\Omega} (\operatorname{div} u) v \, dx + \iint_{\partial\Omega} (vu) \cdot n \, dS.$$

Here n denotes the outer unit normal to $\partial\Omega$.

Theorem (Green's First Identity)

$$\iiint_{\Omega} v \Delta u \, dx = \iint_{\partial\Omega} u \frac{\partial u}{\partial n} \, dS - \iiint_{\Omega} \nabla u \cdot \nabla v \, dx.$$

Theorem (Green's Second Identity)

$$\iiint_{\Omega} (v \Delta u - u \Delta v) \, dx = \iint_{\partial\Omega} \left(v \frac{\partial u}{\partial n} - u \frac{\partial v}{\partial n} \right) \, dS.$$

Theorem (One-Dimensional Integral to Pointwise Theorem I)

$f \in C(\mathbb{R})$ and $\forall I \subsetneq \mathbb{R}$ bounded interval : $\int\limits_I f(x) dx = 0 \implies f \equiv 0.$

Theorem (One-Dimensional Integral to Pointwise Theorem II)

$f \in C(\mathbb{R})$ and $\forall g \in C_c(\mathbb{R})$ continuous function with compact support : $\int\limits_{\mathbb{R}} f(x) g(x) dx = 0 \implies f \equiv 0.$

Theorem (General Integral to Pointwise Theorem I)

$f \in C(\Omega)$ and $\forall W \subset \Omega$ bounded subdomain : $\int\limits_W \cdots \int f(x) dx = 0 \implies f \equiv 0.$

Theorem (General Integral to Pointwise Theorem II)

$f \in C(\Omega)$ and $\forall g \in C_c(\Omega)$ continuous function with compact support : $\int\limits_{\Omega} \cdots \int f(x) g(x) dx = 0 \implies f \equiv 0.$

Theorem (Lebesgue Dominated Convergence)

Let $\{f_n\}_{n \in \mathbb{N}}$ be a sequence of integrable functions on $I \subset \mathbb{R}$ and let f be an integrable function on I such that $\forall x \in I : \lim_{n \rightarrow \infty} f_n(x) = f(x)$. Suppose that there exists an integrable function $g(x) \geq 0$ on I such that

$$\forall n \in \mathbb{N} : \forall x \in I : |f_n(x)| \leq g(x).$$

Then

$$\lim_{n \rightarrow \infty} \int_I f_n(x) dx = \int_I f(x) dx.$$

Theorem (Monotone Convergence)

Let $\{f_n\}_{n \in \mathbb{N}}$ be a sequence of nonnegative integrable functions on $I \subset \mathbb{R}$ and let f be an integrable function on I such that $\forall x \in I : \lim_{n \rightarrow \infty} f_n(x) = f(x)$. Suppose that $\forall n \in \mathbb{N} : \forall x \in I : 0 \leq f_n(x) \leq f_{n+1}(x)$. Then

$$\lim_{n \rightarrow \infty} \int_I f_n(x) dx = \int_I f(x) dx.$$

Theorem (Differentiation under the Integral Sign)

Let $f \in C(\Omega \times (a, b))$ for some $-\infty \leq a < b \leq \infty$. Suppose the following:

- $\partial_t f(x, t) \in C(\Omega \times (a, b))$.
- There exist integrable functions $g(x)$ and $h(x)$ on Ω such that

$$|f(x, t)| \leq g(x) \text{ and } \forall t \in (a, b) : |\partial_t f(x, t)| \leq h(x).$$

Then the function

$$t \longmapsto \int_{\Omega} \cdots \int f(x, t) \, dx$$

is differentiable (continuous) on $t \in (a, b)$ and

$$\partial_t \left(\int_{\Omega} \cdots \int f(x, t) \, dx \right) = \int_{\Omega} \cdots \int \partial_t f(x, t) \, dx.$$

Proof.



Theorem (Leibnitz Rule)

If $f(t, s)$ be a smooth function in both variables, then

$$\partial_t \left(\int_0^t f(t, s) \, ds \right) = f(t, t) + \int_0^t \partial_t f(t, s) \, ds.$$

Proof.

By the change of variable $\tau = \frac{s}{t}$, we have $\int_0^t f(t, s) \, ds = \int_0^1 tf(t, t\tau) \, d\tau$. Hence by differentiation under the integral sign

$$\partial_t \left(\int_0^1 (tf(t, t\tau)) \, d\tau \right) = \int_0^1 \partial_t (tf(t, t\tau)) \, d\tau = \int_0^1 (f(t, t\tau) + t\partial_t f(t, t\tau) + t\partial_s f(t, t\tau)\tau) \, d\tau.$$

Changing variables back to s , we find $\int_0^1 t\partial_t f(t, t\tau) \, d\tau = \int_0^t \partial_t f(t, s) \, ds$. Finally, by the Fundamental Theorem of Calculus,

$$\int_0^1 f(t, t\tau) + t f_s(t, t\tau) \tau \, d\tau = \int_0^1 \frac{d}{d\tau} (\tau f(t, t\tau)) \, d\tau = f(t, t).$$

□

Theorem (Fubini-Tonelli)

Let $f(x, y)$ be a (possibly) complex-valued function on \mathbb{R}^2 . The

$$\int_{\mathbb{R}} \left(\int_{\mathbb{R}} |f(x, y)| \, dx \right) \, dy = \int_{\mathbb{R}} \left(\int_{\mathbb{R}} |f(x, y)| \, dy \right) \, dx.$$

Furthermore, if either of the above iterated integrals in (A.22) is finite (yielding that f is)

$$\int_{\mathbb{R}} \left(\int_{\mathbb{R}} f(x, y) \, dx \right) \, dy = \int_{\mathbb{R}} \left(\int_{\mathbb{R}} f(x, y) \, dy \right) \, dx.$$

Definition (Order)

We define the **order** of a PDE to be the order of the highest derivative which appears in the equation.

Let's write the PDE in the following form:

All terms containing u and its derivatives = All terms involving only the independent variables.

Thus, any PDE for $u(x)$ can be written as

$$\mathcal{L}(u) = f(x),$$

for some operator \mathcal{L} and some function f .

Definition (Linear)

We say the PDE is **linear** if \mathcal{L} is linear in u . That is,

$$\forall \alpha \in \mathbb{R} : \forall u_1, u_2 : \mathcal{L}(\alpha u_1 + u_2) = \alpha \mathcal{L}(u_1) + \mathcal{L}(u_2).$$

Otherwise, we say the PDE is **nonlinear**.

Example (Advection equation, $\mathcal{L}(u) = \partial_t u + c \partial_x u$)

$$\begin{aligned}\mathcal{L}(\alpha u_1 + u_2) &= \partial_t(\alpha u_1 + u_2) + c \partial_x(\alpha u_1 + u_2) \\ &= \alpha \partial_t u_1 + \alpha c \partial_x u_1 + \partial_t u_2 + c \partial_x u_2 \\ &= \alpha \mathcal{L}(u_1) + \mathcal{L}(u_2).\end{aligned}$$

Definition (Semilinear, Quasilinear and Fully Nonlinear PDE)

A PDE of order k is called:

semilinear iff all occurrences of derivatives of order k appear with a coefficient which only depends on the independent variables,

quasilinear iff all occurrences of derivatives of order k appear with a coefficient which only depends on the independent variables, u , and its derivatives of order strictly less than k ,

fully nonlinear iff it is not quasilinear.

Example

Linear

$$(xy) \partial_x u + e^y \partial_y u + (\sin x) u = x^3 y^4.$$

Semilinear

$$(xy) \partial_x u + e^y \partial_y u + (\sin x) u = u^2.$$

Quasilinear

$$u \partial_x u + \partial_y u = 0.$$

Fully Nonlinear

$$(\partial_x u)^2 + (\partial_y u)^2 = 1.$$

Remark

By definition, we have the strict inclusions

$$\{\text{linear PDEs}\} \subsetneq \{\text{semilinear PDEs}\} \subsetneq \{\text{quasilinear PDEs}\}.$$

Explicit form for first-order PDEs in two independent variables x and y

Linear means the PDE can be written in the form

$$a(x, y) \partial_x u(x, y) + b(x, y) \partial_y u(x, y) = c_1(x, y) u + c_2(x, y)$$

for some given real-valued functions a , b , c_1 and c_2 of x and y .

Semilinear means the PDE can be written in the form

$$a(x, y) \partial_x u(x, y) + b(x, y) \partial_y u(x, y) = c(x, y, u)$$

for some given real-valued functions a and b of x and y , and a real-valued function c of x , y , and u .

Quasilinear means that the PDE can be written in the form

$$a(x, y, u) \partial_x u(x, y) + b(x, y, u) \partial_y u(x, y) = c(x, y, u)$$

for some given real-valued functions a , b , and c of x , y , and u .

Remark

Note that in all cases, the coefficient real-valued functions a , b , and c **need not** be linear in their arguments.

Theorem (Cauchy-Kowalewska)

Let the functions f_i of the system

$$\forall i = 1, \dots, n : \frac{\partial u_i}{\partial t} = f_i(x, t, u, \partial_x u),$$

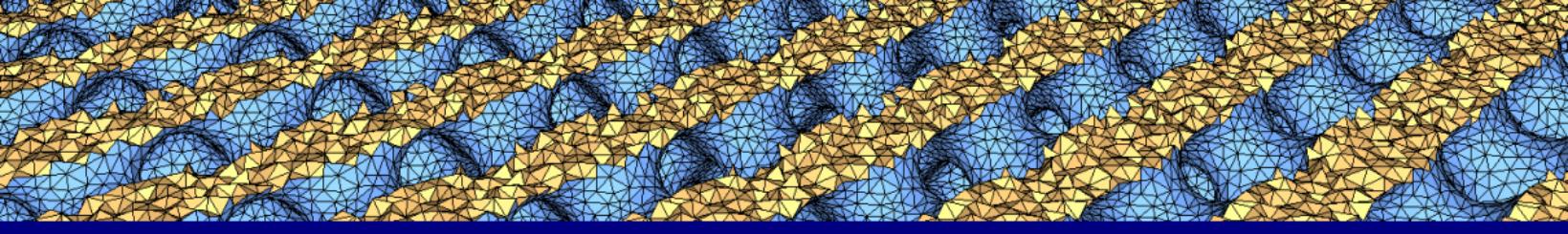
where $u(x, t) = (u_1(x, t), \dots, u_n(x, t))$ with an initial condition

$$u_0(x, 0) = (\phi_1(x), \dots, \phi_n(x))$$

be analytical in some neighborhood of the point

$$t = 0, x = 0, u = 0, \partial_x u = 0.$$

Furthermore, let the initial data (6.1.2) be analytical at $x = 0$. It follows that the Cauchy problem (6.1.1), (6.1.2) admits a unique analytical solution in some neighborhood of the point $x = t = 0$.



Classification of Linear Second Order Partial Differential Equations

Definition (Classification of Linear Second Order PDE)

Let the second order linear EDP with constant coefficients $\mathcal{L}u(x) = f(x)$ and $x \in \Omega$, where the differential operator \mathcal{L} is given by

$$\begin{aligned}\mathcal{L}u(x) &:= \sum_{j=1}^d \sum_{i=1}^d a_{ij} \partial_i \partial_j u(x) + \sum_{i=1}^d b_i \partial_i u(x) + cu(x). \\ &= \langle A, \nabla^2 u(x) \rangle_F + \langle b, \nabla u(x) \rangle + cu(x),\end{aligned}$$

where $A = [a_{ij}] \in \mathbb{R}^{d \times d}$, $b = [b_i]^T \in \mathbb{R}^d$, $\nabla^2 u(x)$ is the Hessian matrix of u at x , $\nabla u(x)$ is the gradient of u at the same point and the Frobenius inner product is defined as

$$\forall A, B \in \mathbb{R}^{d \times d} : \langle A, B \rangle_F := \text{tr}(A^T B).$$

We say that the second-order partial differential operator with constant coefficients \mathcal{D} is

Elliptic iff A has d eigenvalues with the same sign, that is, either $\sigma(A) \subset \mathbb{R}_+$ or $\sigma(A) \subset \mathbb{R}_-$.

Parabolic iff A has exactly $d - 1$ eigenvalues, whether positive or negative, and zero is an eigenvalue of multiplicity one.

Hyperbolic iff A has $d - 1$ positive or negative eigenvalues, and the remaining one is non-zero and of opposite sign.

Ultra-parabolic iff zero is a multiple eigenvalue and all the remaining eigenvalues have the same sign.

Ultra-hyperbolic iff zero is not an eigenvalue and there is more than one positive eigenvalue and more than one negative eigenvalue.

Classification of Linear Second Order Partial Differential Equations

Assume that A is a symmetric matrix. By performing a linear coordinate transformation $\xi = F(x)$. Note that

$$\frac{\partial \xi_i}{\partial x_j} = F_{ij}.$$

$$\frac{\partial u}{\partial x_i} = \sum_{j=1}^d \frac{\partial u}{\partial \xi_j} \frac{\partial \xi_j}{\partial x_i} = \sum_{j=1}^d \frac{\partial u}{\partial \xi_j} F_{ji}.$$

$$\frac{\partial^2 u}{\partial x_i \partial x_j} = \sum_{l=1}^d \sum_{k=1}^d \frac{\partial \xi_k}{\partial x_i} \frac{\partial^2 u}{\partial \xi_k \partial \xi_l} \frac{\partial \xi_l}{\partial x_j} = \sum_{l=1}^d \sum_{k=1}^d F_{ki} \frac{\partial^2 u}{\partial \xi_k \partial \xi_l} F_{lj}.$$

After the coordinate transformation, the differential equation takes the form

$$0 = \sum_{j=1}^d \sum_{i=1}^d a_{ij} \left[\sum_{l=1}^d \sum_{k=1}^d F_{ki} \frac{\partial^2 u}{\partial \xi_k \partial \xi_l} F_{lj} \right] + \sum_{i=1}^d b_i \left[\sum_{j=1}^d \frac{\partial u}{\partial \xi_j} F_{ji} \right] + cu.$$

$$0 = \sum_{l=1}^d \sum_{k=1}^d \left[\sum_{j=1}^d \sum_{i=1}^d F_{ki} a_{ij} F_{lj} \right] \frac{\partial^2 u}{\partial \xi_k \partial \xi_l} + \sum_{j=1}^d \left[\sum_{i=1}^d F_{ji} b_i \right] \frac{\partial u}{\partial \xi_j} + cu.$$

Classification of Linear Second Order Partial Differential Equations

We would like to choose the matrix F so that $D = FAF^T$ is diagonal. Recall that we can diagonalize a symmetric matrix by means of an orthogonal change of variables. In other words, we can choose F to be an **orthogonal matrix**.

Remark

- If D has nonzero diagonal entries all of the same sign, the differential equation is **elliptic**.
- If D has nonzero diagonal entries with one entry of different sign from the others, then the differential equation is **hyperbolic**.
- If D has one zero diagonal entry, the equation may be **parabolic**.

Example (Khokhlov-Zabolotskaya-Kuznetsov (KZK) equation)

$$\partial_{x_1}^2 u + \partial_{x_2}^2 u - \partial_{x_3} \partial_{x_4} u = 0.$$

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & -\frac{1}{2} \\ 0 & 0 & -\frac{1}{2} & 0 \end{bmatrix}, \quad F = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ 0 & 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix}, \quad D = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & -\frac{1}{2} \end{bmatrix}.$$

The KZK equation is **hyperbolic**.

Example (Linear Second Order PDE)

Let $d \in \mathbb{N}$ be the spatial dimension.

Elliptic

$$-\Delta u(x) = \left\langle -I_d, \nabla^2 u(x) \right\rangle_F = -\sum_{i=1}^d \partial_i^2 u(x).$$

Parabolic

$$H(x, t) = \partial_t u(x, t) + \left\langle \begin{bmatrix} -I_d & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}, \nabla^2 u(x, t) \right\rangle_F = \partial_t u(x, t) - \Delta u(x, t).$$

Hyperbolic

$$\square u(x, t) = \left\langle \begin{bmatrix} -I_d & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix}, \nabla^2 u(x, t) \right\rangle_F = \partial_t^2 u(x, t) - \Delta u(x, t).$$

Example (Euler-Tricomi)

Let us consider an equation that changes type.

$$(9) \quad \partial_x^2 u + x \partial_y^2 u = 0.$$

This equation is **elliptic** in the right half-plane $\{(x, y) \in \mathbb{R}^2 \mid x > 0\}$, is **parabolic** in the plane $\{(x, y) \in \mathbb{R}^2 \mid x = 0\}$ and is **hyperbolic** in the left half-plane $\{(x, y) \in \mathbb{R}^2 \mid x < 0\}$.

Remark

(9) is useful in the study of transonic flow (flight at or near the speed of sound).

Classification of Linear Second Order Partial Differential Equations

Theorem (Canonical form of a linear second order PDE)

For any second order linear PDE, there exists a linear change of variables $\xi(x, y)$, $\eta(x, y)$ such that in the new coordinates (ξ, η) , the PDE is transformed as follows

- If PDE is **elliptic**, then

$$\partial_\xi^2 u + \partial_\eta^2 u + F(\partial_\xi u, \partial_\eta u, u) = 0.$$

- If PDE is **parabolic**, then

$$\partial_\eta^2 u + F(\partial_\xi u, \partial_\eta u, u) = 0.$$

- If PDE is **hyperbolic**, then

$$\partial_\xi \partial_\eta u + F(\partial_\xi u, \partial_\eta u, u) = 0.$$

In each case, F is some linear function of three variables.

Proof.

Use the **Sylvester's law of inertia**.



Definition

A system of quasi-linear partial differential equations will be called of

hyperbolic iff its homogeneous part admits wave-like solutions. This implies that a hyperbolic set of equations will be associated to propagating waves and that the behavior and properties of the physical system described by these equations will be dominated by wave-like phenomena.

In other words, a hyperbolic system describes convection phenomena and inversely, convection phenomena are described by hyperbolic equations.

parabolic iff the equations admit solutions corresponding to damped waves.

elliptic if it does not admit wave-like solutions.

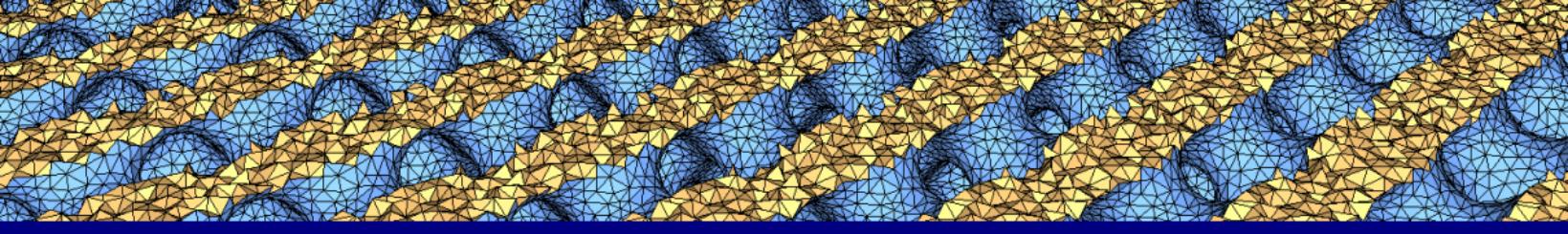
Example

$$a\partial_x u + c\partial_y v = f_1.$$

$$b\partial_x v + c\partial_y u = f_2.$$

Example (Stationary Euler equations)

$$\begin{bmatrix} u & \rho & 0 \\ \frac{c^2}{\rho} & u & 0 \\ 0 & 0 & u \end{bmatrix} \partial_x \begin{bmatrix} \rho \\ u \\ v \end{bmatrix} + \begin{bmatrix} v & 0 & \rho \\ 0 & v & 0 \\ \frac{c^2}{\rho} & 0 & v \end{bmatrix} \partial_y \begin{bmatrix} \rho \\ u \\ v \end{bmatrix} = 0.$$



Method of characteristics

Consider the problem for the explicit form of linear first-order PDEs in two independent variables

$$\begin{cases} a(x, y) \partial_x u + b(x, y) \partial_y u = c_1(x, y) u + c_2(x, y), \\ u(x, y) \text{ given for } (x, y) \in \Gamma. \end{cases}$$

to be solved in some domain $\Omega \subset \mathbb{R}^2$ with data given on some curve $\Gamma \subset \overline{\Omega}$.

Remark

Often the $\Gamma \subset \partial\Omega \subset \mathbb{R}^2$ it will just be one of the coordinate axes.

We find the characteristics, i.e., the curves which follow these directions, by solving

$$\frac{dx}{ds} = a(x(s), y(s)), \quad \frac{dy}{ds} = b(x(s), y(s)).$$

Now suppose u is a solution to the PDE. Let $z(s)$ denote the values of the solution u along a characteristic; i.e.,

$$z(s) := u(x(s), y(s)).$$

Then by the chain rule, we have

$$\frac{dz}{ds} = \partial_x u(x(s), y(s)) \frac{dx}{ds}(x(s), y(s)) + \partial_y u(x(s), y(s)) \frac{dy}{ds}(x(s), y(s)).$$

$$\frac{dz}{ds} = \partial_x u(x(s), y(s)) a(x(s), y(s)) + \partial_y u(x(s), y(s)) b(x(s), y(s)).$$

$$\frac{dz}{ds} = c_1(x(s), y(s)) z(s) + c_2(x(s), y(s)).$$

Definition (Characteristics equations)

There are three **dependent variables** x , y and z and one **independent variable** s .

$$\begin{cases} \frac{dx}{ds}(s) = a(x(s), y(s)) \\ \frac{dy}{ds}(s) = b(x(s), y(s)) \\ \frac{dz}{ds}(s) = c_1(x(s), y(s))z(s) + c_2(x(s), y(s)) \end{cases}$$

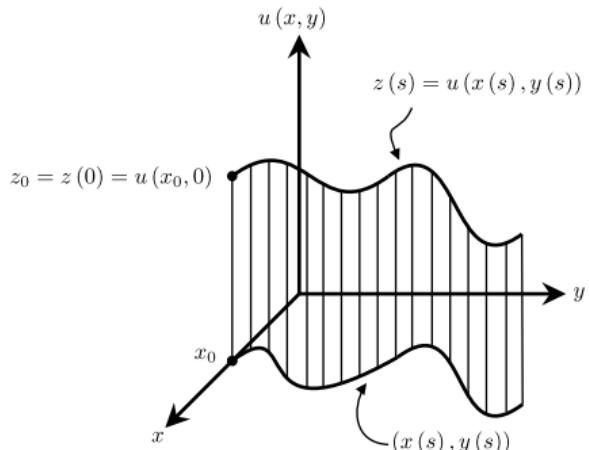


Figure: The solution u is described by the surface defined by $z = u(x, y)$. From any point x_0 on the x -axis, there is a curve $(x(s), y(s))$ in the xy -plane, upon which we can calculate the solution $z = u(x(s), y(s))$. Knowing only the structure of the PDE, x_0 and z_0 we can solve ODEs to find the part of the solution surface which lies above the curve.

Example (Advection)

Linear advection equation

$$\partial_t u + c \partial_x u = 0$$

with an initial condition $u(x, 0) = u_0(x)$ and $c \in \mathbb{R} \setminus \{0\}$.

We can see that $u(x, t) = u_0(x - ct)$ satisfies the PDE. Let $z(x, t) = x - ct$, then from the chain rule we have

$$\begin{aligned}\partial_t u_0(x - ct) + c \partial_t u_0(x - ct) &= \partial_t u_0(z(x, t)) + c \partial_x u_0(z(x, t)). \\ &= u'(z) \partial_t z + cu'_0(z). \\ &= -cu'_0(z) + cu'_0(z). \\ &= 0.\end{aligned}$$

This tell us that the solution transports (or advects) the initial condition with speed c .

The characteristics are paths in the xt -plane, denoted by $(X(t), t)$ on which the solution is constant.

For $\partial_t u + c\partial_x u = 0$ we have $X(t) = X_0 + ct$, since

$$\begin{aligned}\frac{d}{dt}u(X(t), t) &= \partial_t u(X(t), t) + \partial_x u(X(t), t) \frac{dX(t)}{dt} \\ &= \partial_t u(X(t), t) + \partial_x u(X(t), t) c. \\ &= 0.\end{aligned}$$

Hence $u(X(t), t) = u(X(0), 0) = u_0(X_0)$, i.e. the initial condition is transported along characteristics. Characteristics have important implications for the direction of flow of information, and for boundary conditions. More generally, if we have a non-zero right hand side in the PDE, then the situation is a bit more complicated on each characteristic.

Consider $\partial_t u + c\partial_x u = f(x, t, u(x, t))$, and $X(t) = X_0 + ct$.

$$\begin{aligned}
\frac{du(X(t), t)}{dt} &= \partial_t u(X(t), t) + \partial_x u(X(t), t) \frac{dX(t)}{dt} \\
&= \partial_t u(X(t), t) + \partial_x u(X(t), t) \frac{dX(t)}{dt} \\
&= f(X(t), t, u(X(t), t)).
\end{aligned}$$

In this case, the solution is no longer constant on $(X(t), t)$, but we have reduced a PDE to a set of ODEs, so that

$$u(X(t), t) = u_0(X_0) + \int_0^t f(X(t), t, u(X(t), t)) dt.$$

The domain of dependence of the exact solution $u(t_{n+1}, x_j)$ is determined by the characteristics curve passing through (t_{n+1}, x_j) .

Consider the problem for the explicit form of semilinear first-order PDEs in two independent variables

$$\begin{cases} a(x, y) \partial_x u + b(x, y) \partial_y u = c(x, y, u), \\ u(x, y) \text{ given for } (x, y) \in \Gamma. \end{cases}$$

Consider the problem for the explicit form of quasilinear first-order PDEs in two independent variables

$$\begin{cases} \dot{x}(s) = a(x(s), y(s), z(s)), \\ \dot{y}(s) = b(x(s), y(s), z(s)), \\ \dot{z}(s) = c(x(s), y(s), z(s)). \end{cases}$$

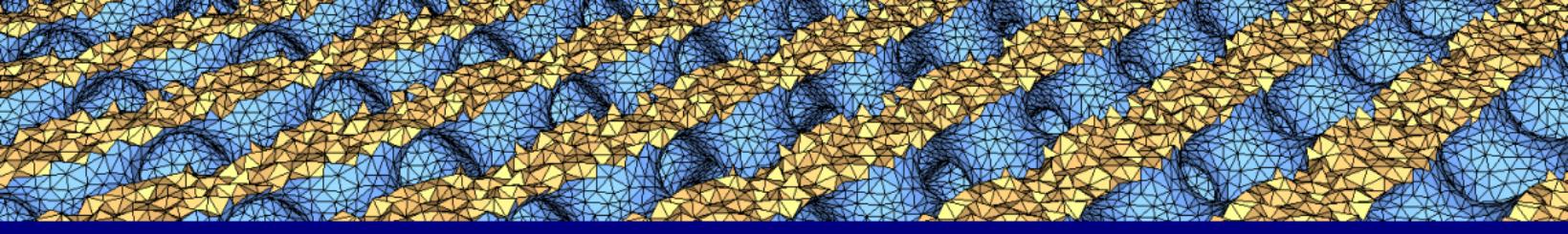
Example

$$\begin{cases} \dot{x}_i(s) = \partial_{p_i} F(p(s), z(s), x(s)), \\ \dot{z}(s) = \sum_{j=1}^N p_j(s) \partial_{p_j} F(p(s), z(s), x(s)), \\ \dot{p}_i(s) = -\partial_{x_i} F(p(s), z(s), x(s)) - \partial_z F(p(s), z(s), x(s)) p_i(s) \\ \quad + \frac{\partial}{\partial s} F(p(s), z(s), x(s)) \end{cases}$$

Example

Thus we consider the problem

$$\begin{cases} \frac{d^2\psi(t)}{dt^2} - \lambda\psi(t) = 0 \\ \frac{d^2\varphi(x)}{dx^2} - \lambda\varphi(x) = 0. \end{cases}$$



Trigonometric Fourier Series

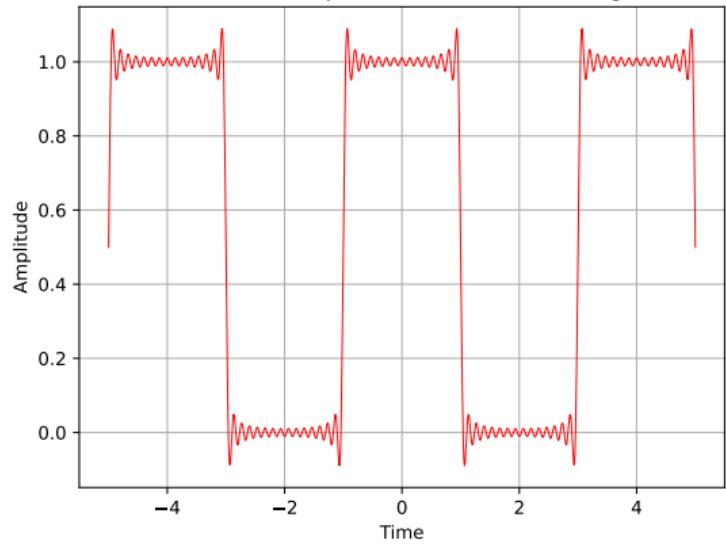
Trigonometric Fourier Series

The subject was founded by **Jean-Baptiste Joseph Fourier**, who discovered what we would recognize as the basics of Fourier Analysis in his studies of heat flow in the 1820s.



Jean-Baptiste Joseph
Fourier (1768–1830).

Fourier Series representation of Periodic Signal



Definition (Fourier series of f relative)

Let $f \in L^2(I)$ and $\{\varphi_k\}_{k \in \mathbb{N}}$ an orthonormal sequence on $I \subset \mathbb{R}$. The Fourier series of f relative of $\{\varphi_k\}_{k \in \mathbb{N}}$ is $\sum_{k \in \mathbb{N}} c_k \varphi_k(\theta)$, where $\forall k \in \mathbb{N} : c_k := \langle f, \varphi_k \rangle = \int_I f(\theta) \overline{\varphi_k(\theta)} d\theta$ are the Fourier coefficients of f relative of $\{\varphi_k\}_{k \in \mathbb{N}}$.

Example

If $I = [0, 2\pi]$ and two orthonormal sequences of trigonometric functions $\{\varphi_k\}_{k \in \mathbb{N}}, \{\phi_k\}_{k \in \mathbb{Z}}$:

$$\text{real } \varphi_0(\theta) = \frac{1}{\sqrt{2\pi}}, \quad \varphi_{2k-1}(\theta) = \frac{\cos(k\theta)}{\sqrt{\pi}}, \quad \varphi_{2k}(\theta) = \frac{\sin(k\theta)}{\sqrt{\pi}}.$$

$$\text{complex } \phi_k(\theta) = \frac{e^{ik\theta}}{\sqrt{2\pi}} = \frac{\cos(k\theta) + i \sin(k\theta)}{\sqrt{2\pi}}.$$

Then, the Fourier series of f relative of $\{\varphi_k\}_{k \in \mathbb{N}}$ and $\{\phi_k\}_{k \in \mathbb{Z}}$ are

$$\text{real } \frac{a_0}{2} + \sum_{k \in \mathbb{N}} a_k \cos(k\theta) + b_k \sin(k\theta).$$

$$\text{complex } \sum_{k \in \mathbb{Z}} \alpha_k e^{ik\theta}, \quad \alpha_k = \frac{1}{2\pi} \int_0^{2\pi} f(\theta) e^{-ik\theta} d\theta.$$

Remark

The set of functions $\left\{ \frac{1}{\sqrt{2\pi}}, \frac{\cos(m\theta)}{\sqrt{\pi}}, \frac{\sin(n\theta)}{\sqrt{\pi}} \right\}_{m,n \in \mathbb{N}} \subset L^2([0, 2\pi])$ is orthonormal.

Indeed, $\forall n, m \in \mathbb{N}$:

$$\bullet \int_0^{2\pi} \left(\frac{1}{\sqrt{2\pi}} \right)^2 d\theta = \int_0^{2\pi} \frac{1}{2\pi} d\theta = \frac{1}{2\pi} \theta \Big|_0^{2\pi} = 1.$$

$$\bullet \int_0^{2\pi} \left(\frac{\cos(m\theta)}{\sqrt{\pi}} \right)^2 d\theta = \int_0^{2\pi} \frac{\cos^2(m\theta)}{\pi} d\theta = \frac{1}{2\pi} \int_0^{2\pi} 1 + \cos(2m\theta) d\theta = \frac{1}{2\pi} \left(\theta + \frac{\sin(4m\theta)}{4m} \right) \Big|_0^{2\pi} = 1.$$

$$\bullet \int_0^{2\pi} \left(\frac{\sin(n\theta)}{\sqrt{\pi}} \right)^2 d\theta = \int_0^{2\pi} \frac{\sin^2(n\theta)}{\pi} d\theta = \frac{1}{2\pi} \int_0^{2\pi} 1 - \cos(2n\theta) d\theta = \frac{1}{2\pi} \left(\theta - \frac{\sin(4n\theta)}{4n} \right) \Big|_0^{2\pi} = 1.$$

$$\bullet \int_0^{2\pi} \frac{1}{\sqrt{2\pi}} \frac{\cos(m\theta)}{\sqrt{\pi}} d\theta = \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} \cos(m\theta) d\theta = 0.$$

$$\bullet \int_0^{2\pi} \frac{1}{\sqrt{2\pi}} \frac{\sin(n\theta)}{\sqrt{\pi}} d\theta = \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} \sin(n\theta) d\theta = 0.$$

$$\bullet \int_0^{2\pi} \frac{\cos(m\theta)}{\sqrt{\pi}} \frac{\sin(n\theta)}{\sqrt{\pi}} d\theta = \frac{1}{\pi} \int_0^{2\pi} \sin(n\theta) \cos(m\theta) d\theta = \frac{1}{\pi} \int_0^{2\pi} \frac{\sin((n+m)\theta) - \sin((n-m)\theta)}{2} d\theta = 0.$$

Definition (Fourier series generated by f)

Let $f \in L^2([0, 2\pi])$. The **Fourier coefficients** of f are given by

$$a_0 = \frac{1}{\pi} \int_0^{2\pi} f(\theta) d\theta, \quad a_k = \frac{1}{\pi} \int_0^{2\pi} f(\theta) \cos(k\theta) d\theta, \quad b_k = \frac{1}{\pi} \int_0^{2\pi} f(\theta) \sin(k\theta) d\theta.$$

and the n -th partial Fourier sum is

$$s_n f(\theta) = \frac{a_0}{2} + \sum_{k=1}^n a_k \cos(k\theta) + b_k \sin(k\theta).$$

Indeed, from the equalities $\forall k \in \mathbb{N}$:

$$\bullet \int_0^{2\pi} \frac{a_0}{2} d\theta = \frac{a_0}{2} \theta \Big|_0^{2\pi} = \pi a_0. \quad \bullet \int_0^{2\pi} \cos(k\theta) d\theta = \frac{\sin(k\theta)}{k} \Big|_0^{2\pi} = 0. \quad \bullet \int_0^{2\pi} \sin(k\theta) d\theta = \frac{-\cos(k\theta)}{k} \Big|_0^{2\pi} = 0.$$

If we integrate the Fourier series term by term

$$\int_0^{2\pi} f(\theta) d\theta = \int_0^{2\pi} \frac{a_0}{2} d\theta + \int_0^{2\pi} \left(\sum_{k=1}^{\infty} a_k \cos(k\theta) + b_k \sin(k\theta) \right) d\theta.$$

Then,

$$\int_0^{2\pi} f(\theta) d\theta = \frac{a_0}{2} \int_0^{2\pi} d\theta + \sum_{k=1}^{\infty} \left(a_k \int_0^{2\pi} \cos(k\theta) d\theta + b_k \int_0^{2\pi} \sin(k\theta) d\theta \right).$$

$$\int_0^{2\pi} f(\theta) d\theta = \pi a_0 + \sum_{k=1}^{\infty} (a_k \cdot 0 + b_k \cdot 0). \quad \Rightarrow \quad a_0 = \frac{1}{\pi} \int_0^{2\pi} f(\theta) d\theta.$$

Multiplying the Fourier series by $\cos(m\theta)$, $m \in \mathbb{N}$ and integrating term by term:

$$\int_0^{2\pi} \cos(m\theta) f(\theta) d\theta = \int_0^{2\pi} \cos(m\theta) \frac{a_0}{2} d\theta + \int_0^{2\pi} \cos(m\theta) \left(\sum_{k=1}^{\infty} a_k \cos(k\theta) + b_k \sin(k\theta) \right) d\theta.$$

$$\int_0^{2\pi} f(\theta) \cos(m\theta) d\theta = 0 + \sum_{k=1}^{\infty} \left(a_k \int_0^{2\pi} \cos(k\theta) \cos(m\theta) d\theta + b_k \int_0^{2\pi} \sin(k\theta) \cos(m\theta) d\theta \right).$$

$$\int_0^{2\pi} f(\theta) \cos(m\theta) d\theta = \sum_{k=1}^{\infty} \left(\frac{a_k}{2} \int_0^{2\pi} \cos((m+k)\theta) + \cos((m-k)\theta) d\theta + \frac{b_k}{2} \int_0^{2\pi} \sin((m+k)\theta) + \sin((m-k)\theta) d\theta \right).$$

When $m \neq k$ both integrals vanish, thus the infinite sum reduces to m -th addend.

$$\int_0^{2\pi} f(\theta) \cos(m\theta) d\theta = a_m \int_0^{2\pi} \cos^2(m\theta) d\theta + b_m \int_0^{2\pi} \sin(m\theta) \cos(m\theta) d\theta.$$

$$\int_0^{2\pi} f(\theta) \cos(m\theta) d\theta = \frac{a_m}{2} \int_0^{2\pi} 1 + \cos(2m\theta) d\theta + b_m \cdot 0.$$

$$\int_0^{2\pi} f(\theta) \cos(m\theta) d\theta = a_m \pi. \quad \Rightarrow \quad a_m = \frac{1}{\pi} \int_0^{2\pi} f(\theta) \cos(m\theta) d\theta.$$

Multiplying the Fourier series by $\sin(m\theta)$, $m \in \mathbb{N}$ and integrating term by term:

$$\int_0^{2\pi} \sin(m\theta) f(\theta) d\theta = \int_0^{2\pi} \sin(m\theta) \frac{a_0}{2} d\theta + \int_0^{2\pi} \sin(m\theta) \left(\sum_{k=1}^{\infty} a_k \cos(k\theta) + b_k \sin(k\theta) \right) d\theta.$$

$$\int_0^{2\pi} f(\theta) \sin(m\theta) d\theta = 0 + \sum_{k=1}^{\infty} \left(a_k \int_0^{2\pi} \cos(k\theta) \sin(m\theta) d\theta + b_k \int_0^{2\pi} \sin(k\theta) \sin(m\theta) d\theta \right).$$

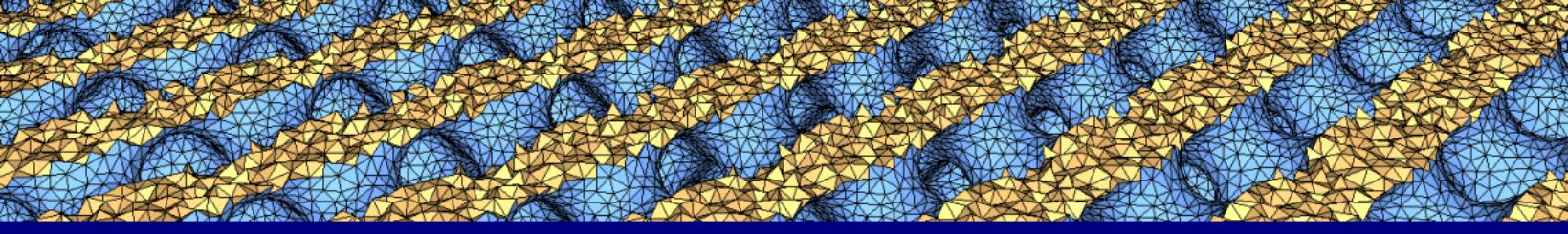
$$\int_0^{2\pi} f(\theta) \sin(m\theta) d\theta = \sum_{k=1}^{\infty} \left(\frac{a_k}{2} \int_0^{2\pi} \sin((m+k)\theta) + \sin((m-k)\theta) d\theta + \frac{b_k}{2} \int_0^{2\pi} \cos((m-k)\theta) - \cos((m+k)\theta) d\theta \right).$$

When $m \neq k$ both integrals vanish, thus the infinite sum reduces to m -th addend.

$$\int_0^{2\pi} f(\theta) \sin(m\theta) d\theta = a_m \int_0^{2\pi} \cos(m\theta) \sin(m\theta) d\theta + b_m \int_0^{2\pi} \sin^2(m\theta) d\theta.$$

$$\int_0^{2\pi} f(\theta) \sin(m\theta) d\theta = a_m \cdot 0 + \frac{b_m}{2} \int_0^{2\pi} 1 - \cos(2m\theta) d\theta.$$

$$\int_0^{2\pi} f(\theta) \sin(m\theta) d\theta = b_m \pi. \quad \Rightarrow \quad b_m = \frac{1}{\pi} \int_0^{2\pi} f(\theta) \sin(m\theta) d\theta.$$



Fourier transform

Fourier transform

$$\frac{d^n f}{dx^n} = i^n k^n$$

$$\widehat{v}(\xi) = \mathcal{F}[v](\xi) = \int_{\mathbb{R}^d} \exp(-ix\xi) v(x) dx$$

$$\left(\partial_t u - \widehat{\alpha^2 \Delta u} \right)(k, t) = \widehat{(\partial_t u)}(k, t) - \widehat{\alpha^2 \Delta u}(k, t) = \widehat{0} = 0$$

$$\widehat{(\partial_t u)}(k, t) = \int_{-\infty}^{\infty} \partial_t u(x, t) \exp(-ikx) dx = \partial_t \left(\int_{-\infty}^{\infty} u(x, t) dx \right) \exp(-ikx) dx = \partial_t \widehat{u}(k, t) = \partial_t \widehat{u} =$$

$$\widehat{\partial_x^2 u}(k, t) = (ik)^2 \widehat{u}(k, t) = -k^2 \widehat{u}(k, t) =$$

Para cada k , $f(t) := \widehat{u}(k, t)$

$$F(x) = \frac{1}{\sqrt{4\pi t}} \exp\left(-\frac{x^2}{4t}\right)$$

$$u(x, t) = \frac{1}{\sqrt{4\pi\alpha t}} \int_{-\infty}^{\infty} \exp\left(-\frac{(x-y)^2}{4\alpha t}\right) g(y) dy$$

Proof.

$$u(x, t) - u_0(x_0) = \frac{1}{(4\pi t)^{d/2}} \int_{\mathbb{R}^d} \exp\left(-\frac{\|x-y\|_2^2}{4t}\right) u_0(y) dy - u_0(x_0)$$

□

One-Parameter Semigroups for Linear Evolution Equations

Find all maps $T: \mathbb{R}_+ \rightarrow \mathbb{C}$ satisfying the functional equation

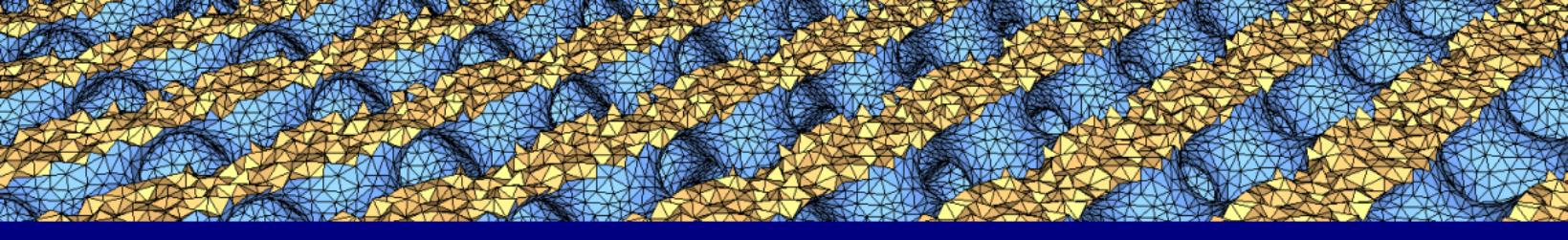
$$\begin{cases} T(t+s) = T(t)T(s), & \forall t, s \geq 0, \\ T(0) = 1. \end{cases}$$

$$\forall a \in \mathbb{C} : t \mapsto \exp(ta).$$

Definition (Semigroup)

A function $U: \mathbb{R}_{>0} \rightarrow B(E)$ is called semigroup iff

- $\forall t, t' \geq 0 : U_{t+t'} = U_t U_{t'}$.
- $U_0 = 1$.



Distribution

Definition (Distribution)

A distribution F is a rule, assigning to each test function $\phi \in C_c^\infty(\mathbb{R})$ a real number, which is **linear** and **continuous**.

$$\begin{aligned} F: C_c^\infty(\mathbb{R}) &\longrightarrow \mathbb{R} \\ \phi &\longmapsto \langle F, \phi \rangle. \end{aligned}$$

Remark

Let $f(x)$ be a locally integrable function on \mathbb{R} . Then $f(x)$ can be interpreted as the distribution F_f where

$$\forall \phi \in C_c^\infty(\mathbb{R}): \langle F_f, \phi \rangle := \int_{-\infty}^{\infty} f(x) \phi(x) \, dx.$$

Think of F_f as the **distribution generated** by the function $f(x)$.

Definition (Derivative of a Distribution)

Let F be any distribution. Then F' is also a distribution defined by $\forall \phi \in C_c^\infty(\mathbb{R}): \langle F', \phi \rangle = -\langle F, \phi' \rangle$.

The **distributional derivative** of the distribution generated by a smooth function is simply the distribution generated by the classical derivative.

Delta “function”

Definition (Delta function)

$$\forall \phi \in C_c^\infty(\mathbb{R}) : \langle \delta_0, \phi \rangle = \phi(0).$$

Definition (Derivative of the Delta Function)

$$\forall \phi \in C_c^\infty(\mathbb{R}) : \langle \delta'_0, \phi \rangle = -\langle \delta_0, \phi' \rangle = -\phi'(0).$$

Definition (Convergence of a Sequence of Distributions)

A sequence of distributions $\{F_n\}_{n \in \mathbb{N}}$ converges to a distribution F iff

$$\forall \phi \in C_c^\infty(\mathbb{R}) : \langle F_n, \phi \rangle \xrightarrow{n \rightarrow \infty} \langle F, \phi \rangle.$$

Definition (Sobolev Space $W^{k,p}(\mathbb{R}^d)$)

Let $k \in \mathbb{N}$, $p \geq 1$, and let $u: \mathbb{R}^d \rightarrow \mathbb{R}$ be a locally integrable function. Suppose the following:

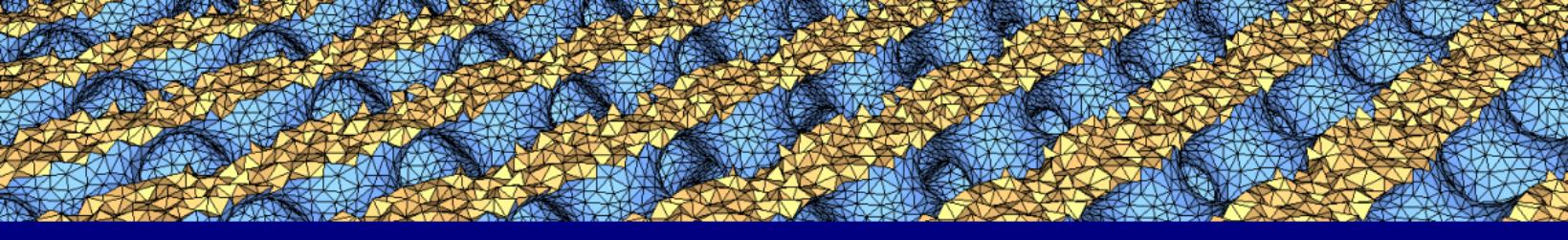
- For any multi-index α of order less than or equal to k , $\partial^\alpha u$ in the sense of distributions is generated by a locally integrable function v_α ; that is,

$$\forall \phi \in C_c^\infty(\mathbb{R}^d) : \langle \partial^\alpha u, \phi \rangle = (-1)^{|\alpha|} \langle u, \partial^\alpha \phi \rangle = (-1)^{|\alpha|} \int_{\mathbb{R}^d} \cdots \int u(x) \partial^\alpha \phi(x) dx = \int_{\mathbb{R}^d} \cdots \int v_\alpha(x) \phi(x) dx.$$

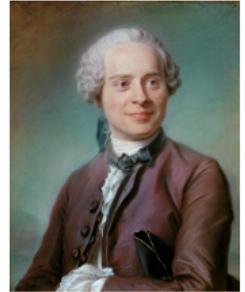
- The function v_α satisfies

$$\int_{\mathbb{R}^d} \cdots \int |v_\alpha(x)|^p dx < \infty.$$

Then we say $u \in W^{k,p}(\mathbb{R}^d)$.



Wave operator



Jean-Baptiste le Rond
D'Alembert
(1717–1783).

Let the **d'Alembertian** operator $\square := \partial_t^2 - c^2 \Delta = \partial_t^2 - c^2 \sum_1^n \partial_j^2$ on $\mathbb{R}^n \times \mathbb{R}$.

The **wave equation**

$$(10) \quad \square u = 0$$

is fulfilled by waves with propagation speed c in a homogeneous isotropic medium.

Invariance

The operator \square is invariant under time-reversal $(x, t) \rightarrow (x, -t)$.

Remark

- If the solution of (10) is $u \in C^2(\mathbb{R} \times \mathbb{R})$, then ∂_x and ∂_t commute and the wave operator is

$$\square = \partial_t^2 - c^2 \partial_x^2 = (\partial_t - c\partial_x)(\partial_t + c\partial_x).$$

Thus, the one-dimensional wave equation becomes $(\partial_t - c\partial_x)[(\partial_t + c\partial_x)u] = 0$. These are the first order operators of the **transport equation**, whose solution is of the form $u(x, t) = g(x - ct)$.

- The wave equation solution is the sum of two solutions of transport equations: one moves **to the left** and the other moves **to the right** both with propagation speed c .

Theorem (General solution of the one-dimensional wave equation)

The solution of (10) is $u(x, t) = f(x + ct) + g(x - ct)$, where $f, g \in C^2(\mathbb{R})$.

Remark

- If $\phi \in C^2(\mathbb{R})$, then $u_{\pm}(x, t) = \phi(x \pm ct)$ fulfills (10) since $\partial_t^2 u_{\pm} = c^2 \partial_x^2 u_{\pm} = \frac{d^2 \phi}{dx^2}(x \pm ct)$.
- If ϕ is locally integrable, then u is a distributional solution.

Proof.

Let $f, g \in C^2(\mathbb{R})$ and the changes of variable $\xi = x + ct$, $\eta = x - ct$. The **chain rule** gives us

$$\partial_x u = (\partial_\xi + \partial_\eta) u, \quad \partial_t u = c(\partial_\xi - \partial_\eta) u.$$

Combining these equations, we find

$$(\partial_t - c\partial_x) u = -2c\partial_\eta u, \quad (\partial_t + c\partial_x) u = 2c\partial_\xi u.$$

Thus, (10) results

$$\square u = (\partial_t - c\partial_x)(\partial_t + c\partial_x) u = (-2c\partial_\eta u)(2c\partial_\xi u) = -4c^2 \partial_{\xi\eta} u = 0.$$

His solution is $u(\xi, \eta) = f(\xi) + g(\eta)$. That is, $u(x, t) = f(x + ct) + g(x - ct)$ with the original variables. □

Global homogeneous Cauchy problem for the one-dimensional wave equation

The boundary value problem for (10) is a Cauchy problem.

$$(11) \quad \begin{cases} \square u = 0 & \text{for } x \in \mathbb{R}, t > 0, \\ u(x, 0) = \phi(x), \quad \partial_t u(x, 0) = \psi(x) & \text{for } x \in \mathbb{R}, \end{cases}$$

where $\phi \in C^2(\mathbb{R})$ and $\psi \in C^1(\mathbb{R})$ are the **initial displacement** and the **initial velocity**, respectively.

Theorem (d'Alembert's formula)

If $u(x, t)$ is solution of (11), then

$$(12) \quad u(x, t) = \frac{1}{2} [\phi(x + ct) + \phi(x - ct)] + \frac{1}{2c} \int_{x-ct}^{x+ct} \psi(s) \, ds.$$

Proof.

By theorem 93, we have that the general solution of (10) is $u(x, t) = f(x + ct) + g(x - ct)$ with $f, g \in C^2(\mathbb{R})$. Let us look for a relation between the functions f y g con ϕ y ψ . Note that

$$u(x, 0) = f(x) + g(x) = \phi(x), \quad \partial_t u(x, 0) = f'(x) + g'(x) = \psi(x).$$

Replace x by α

$$\phi(\alpha) = f(\alpha) + g(\alpha), \quad \psi(\alpha) = f'(\alpha) + g'(\alpha).$$

Proof (Cont.)

Solving the system of equations for f' and g' in terms of ϕ' and ψ' .

$$f'(\alpha) = \frac{1}{2} [\phi'(\alpha) + \psi(\alpha)], \quad g'(\alpha) = \frac{1}{2} [\phi'(\alpha) - \psi(\alpha)].$$

Integrating with respect to α gives us

$$f(\alpha) = \frac{1}{2}\phi(\alpha) + \frac{1}{2} \int_0^\alpha \psi(s) \, ds + C_1, \quad g(\alpha) = \frac{1}{2}\phi(\alpha) - \frac{1}{2} \int_0^\alpha \psi(s) \, ds + C_2.$$

$$\therefore \phi(\alpha) = f(\alpha) + g(\alpha) = \frac{1}{2}\phi(\alpha) + \frac{1}{2} \int_0^\alpha \psi(s) \, ds + C_1 + \frac{1}{2}\phi(\alpha) - \frac{1}{2} \int_0^\alpha \psi(s) \, ds + C_2 = \phi(\alpha) + C_1 + C_2.$$

And we have that $C_1 + C_2 = 0$. Finally,

$$u(x, t) = f(x + ct) + g(x - ct) = \frac{1}{2} [\phi(x + ct) + \phi(x - ct)] + \frac{1}{2} \int_0^{x+ct} \psi(s) \, ds - \frac{1}{2} \int_0^{x-ct} \psi(s) \, ds.$$

$$u(x, t) = \frac{1}{2} [\phi(x + ct) + \phi(x - ct)] + \frac{1}{2} \int_0^{x+ct} \psi(s) \, ds - \left(-\frac{1}{2} \int_{x-ct}^0 \psi(s) \, ds \right).$$

$$u(x, t) = \frac{1}{2} [\phi(x + ct) + \phi(x - ct)] + \frac{1}{2} \int_{x-ct}^{x+ct} \psi(s) \, ds.$$

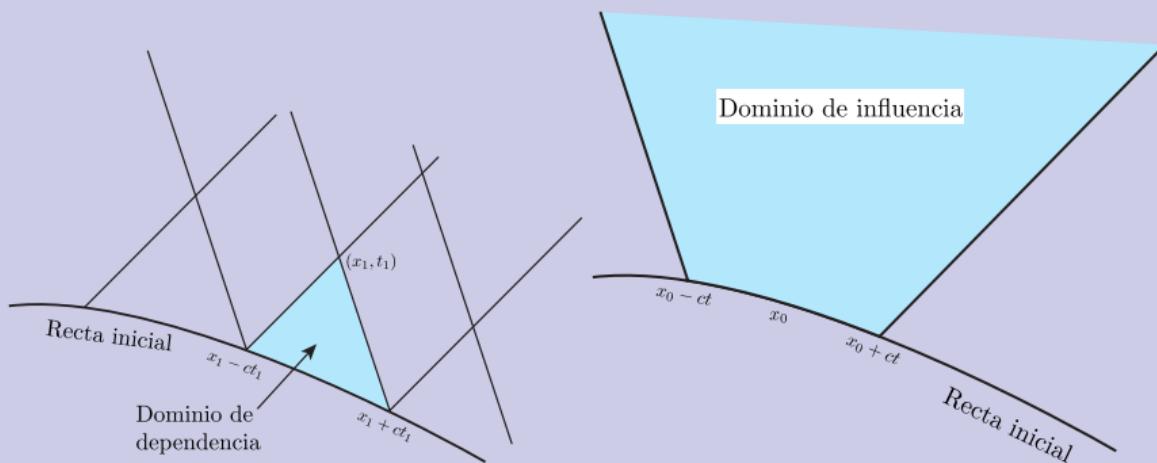
Example

Let $l > 0$ and suppose that $\phi(x) = \sin\left(\frac{x}{l}\right)$ and $\psi(x) = 0$. Then, the **d'Alembert's formula** gives

$$u(x, t) = \frac{1}{2} \left[\sin\left(\frac{x+t}{l}\right) + \sin\left(\frac{x-t}{l}\right) \right] = \sin\left(\frac{x}{l}\right) \cos\left(\frac{ct}{l}\right).$$

Definition (Domain of dependence and influence)

- Fix a point (x_1, t_1) in space-time and obtain an initial data $u(x_1, t_1)$ and look to the past.
- Fix a point x_0 at $t = 0$ and ask yourself after a time $t_1 > 0$, which points on the string are influenced by the displacement/velocity at x_0 for $t = 0$?



The following principle is related to the method of **variation of Parameters** in ODEs.

Global inhomogeneous Cauchy problem for the one-dimensional wave equation

Now suppose that the infinite string is under to a vertical external force at position x and time t .

$$(13) \quad \begin{cases} \square u = f(x, t) & \text{for } x \in \mathbb{R}, t > 0. \\ u(x, 0) = \phi(x), \quad \partial_t u(x, 0) = \psi(x) & \text{for } x \in \mathbb{R}. \end{cases}$$

Theorem

If $u(x, t)$ is a solution of (13), then

$$(14) \quad u(x, t) = \frac{1}{2} [\phi(x + ct) + \phi(x - ct)] + \frac{1}{2c} \int_{x-ct}^{x+ct} \psi(s) \, ds + \frac{1}{2c} \iint_D f(y, \tau) \, dy \, d\tau,$$

where D is the **dependency domain** associated with (x, t) , es decir, the triangle in the plane x versus t and the base points $(x - ct, 0)$ and $(x + ct, 0)$.

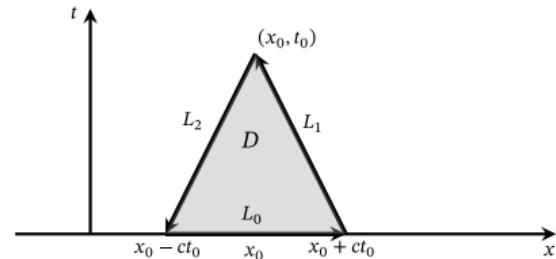
Remark

If $f \equiv 0$ in (13), then we recover d'Alembert's formula in (14).

Proof.

Let $(x_0, t_0) \in D$ fixed and arbitrary. Suppose that (13) has a solution.

$$\begin{aligned}
 f(x, t) &= \square u = \partial_t^2 u - c^2 \partial_x^2 u. \\
 \iint_D f(x, t) \, dx \, dt &= \iint_D (\partial_t^2 u - c^2 \partial_x^2 u) \, dx \, dt. \\
 &= \iint_D (-c^2 \partial_x u)_x - (-\partial_t u)_t \, dx. \\
 &\stackrel{\text{G}}{=} \int_{\partial D} -c^2 \partial_x u - \partial_t u \, dx. \\
 &= \int_{L_0 \cup L_1 \cup L_2} -c^2 \partial_x u - \partial_t u \, dx. \\
 &= \sum_{i=0}^2 \int_{L_i} -c^2 \partial_x u - \partial_t u \, dx.
 \end{aligned}$$



With the help of **Green's theorem** with $P = -\partial_t u$ y $Q = -c^2 \partial_x u$.

Proof (Cont.)

We calculate each line integral

$$\int_{L_0} -c^2 \partial_x u \, dt - \partial_t u \, dx = - \int_{x_0 - ct_0}^{x_0 + ct_0} \psi(x) \, dx.$$
$$\int_{L_1} -c^2 \partial_x u \, dt - \partial_t u \, dx = \int_0^{t_0} (-c^2 \partial_x u + c \partial_t u) \, dt = cu(x_0, t_0) - c\phi(x_0 + t_0).$$
$$\int_{L_2} -c^2 \partial_x u - \partial_t u \, dx = -c\phi(x_0 - ct_0) + cu(x_0, t_0).$$

Then,

$$\iint_D f(x, t) \, dx \, dt = - \int_{x_0 - ct_0}^{x_0 + ct_0} \psi(x) \, dx + cu(x_0, t_0) - c\phi(x_0 + t_0) - c\phi(x_0 - ct_0) + cu(x_0, t_0).$$

Therefore,

$$u(x_0, t_0) = \frac{1}{2} [\phi(x_0 + ct_0) + \phi(x_0 - ct_0)] + \frac{1}{2c} \int_{x_0 - ct_0}^{x_0 + ct_0} + \frac{1}{2c} \iint_D f(x, t) \, dx \, dt.$$



Example

$$\begin{cases} \square u = xt & \text{for } x \in \mathbb{R}, t > 0. \\ u(x, 0) = \sin(x), \quad \partial_t u(x, 0) = \cos(x) & \text{for } x \in \mathbb{R}. \end{cases}$$

Solution

Then, (14) gives

$$u(x, t) = \frac{1}{2} [\sin(x + ct) + \sin(x - ct)] + \frac{1}{2c} \int_{x-ct}^{x+ct} \cos(s) \, ds + \frac{1}{2c} \int_0^t \int_{x-c(t-\tau)}^{x+c(t-\tau)} y\tau \, dy \, d\tau.$$

$$u(x, t) = \frac{1}{2} [2 \sin(x) \cos(ct)] + \frac{1}{2c} [\sin(x + ct) - \sin(x - ct)] + \frac{1}{2} \int_0^t \int_{x-ct+c\tau}^{x+ct-c\tau} y\tau \, dy \, d\tau.$$

$$u(x, t) = \sin(x) \cos(ct) + \frac{1}{2} [2 \cos(x) \sin(ct)] + \frac{1}{2} \int_0^t \left[\frac{(x + ct - c\tau)^2}{2} - \frac{(x - ct + c\tau)^2}{2} \right] \tau \, d\tau.$$

$$u(x, t) = \sin(x) \cos(ct) + \cos(x) \sin(ct) + \frac{1}{4} \int_0^t 4cx(t - \tau) \tau \, d\tau = \sin(x + ct) + cx \int_0^t (t\tau - \tau^2) \, d\tau.$$

$$u(x, t) = \sin(x + ct) + \frac{cxt^3}{6}.$$

Mechanical wave

Recall the physical concepts of the **law of conservation of energy** and the equation of motion of the mechanical wave, and we will deduce its formulas for kinetic and potential energy.

The period T and the frequency f of the wave are given by

$$\frac{\omega}{2\pi} = f = \frac{1}{T}.$$

$$v = \frac{\omega}{k} = \frac{\lambda}{T} = \lambda f.$$

The speed of a wave on a stretched string with tension t and linear density μ is

$$v = \sqrt{\frac{\tau}{\mu}},$$

that is, it depends only on the properties of the string and the medium.

The principle of conservation of mechanical energy

Energy is a scalar quantity associated with the state of one or more bodies.

Definition (Energía cinética)

Es la energía asociada con el estado de movimiento de un cuerpo.

$$K = \frac{1}{2}mv^2.$$

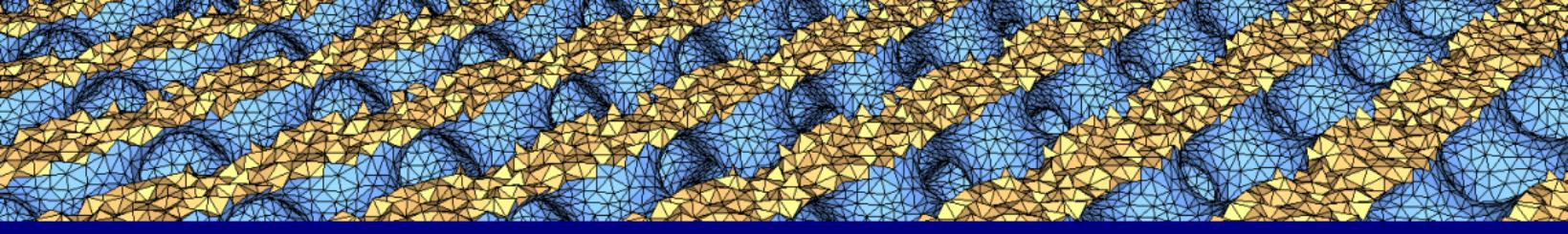
$$K = \frac{1}{2} \int_0^l \rho \partial_t^2 u(x, t) \, dx.$$

$$U = \frac{1}{2} \int_0^l T \partial_x^2 u(x, t) \, dx.$$

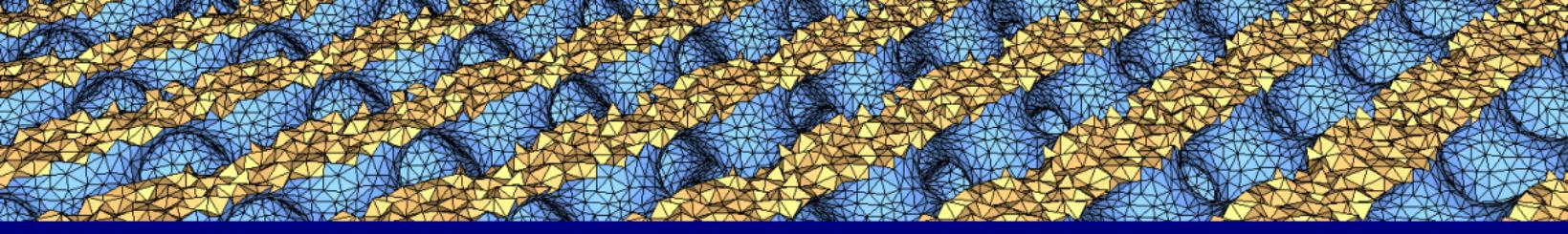
$$E = K + U = \frac{1}{2} \int_0^l (\rho \partial_t^2 u(x, t) + T \partial_x^2 u(x, t)) \, dx.$$

$$\partial_t E = 0.$$

Energy of vibrating string



Wave equation with two spatial dimensions



Diffusion operator

Theorem

Let $u_0 \in C_b(\mathbb{R}^d) \cap L^2(\mathbb{R}^d)$.

$$\forall x_0 \in \mathbb{R}^d : \lim_{(x,t) \rightarrow (x_0,0)} u(x,t) = u_0(x_0).$$

Example

Let

$$u_0(x) = \exp\left(-\frac{x^2}{2}\right).$$

Theorem (Nonexpansiveness)

For any $t > 0$.

$$E(t) : C_b(\mathbb{R}^d) \longrightarrow C_b(\mathbb{R}^d)$$
$$x \longmapsto \frac{1}{(4\pi t)^{\frac{d}{2}}} \int_{\mathbb{R}^d} \exp\left(-\frac{\|x-y\|_2^2}{4t}\right) u(y) \, dy.$$

$$\forall t > 0 : \|u_1(\cdot, t) - u_2(\cdot, t)\|_{L^\infty(\mathbb{R}^d)} \leq \|u_0^1 - u_0^2\|_{L^\infty(\mathbb{R}^d)}$$

Definition (Weak solution)

Let $T \in (0, \infty]$.

$$u \in L^2 \left(0, T; H_0^1(\Omega) \right), \quad \partial_t u \in L^2 \left(0, T; H_0^1(\Omega)' \right)$$

Theorem (Maximum principle)

Let $u: \bar{\mathcal{C}} \rightarrow \mathbb{R}$

$$\partial_t u(x, t) - \Delta u(x, t) \leq 0$$

Example (IVP for the diffusion equation)

$$(15) \quad \begin{cases} \partial_t u - \alpha^2 \Delta u = 0 & \text{for } (x, t) \in \Omega \times [0, T]. \\ u(x, 0) = \phi(x), \quad \partial_t u(x, 0) = \psi(x) & \text{for } x \in \Omega. \end{cases}$$

Hopf-Cole transformation for the one-dimensional quasi-linear advection-diffusion

Is a fundamental PDE that occurs in fluid mechanics, nonlinear acoustics, gas dynamics and traffic flow. We consider the

(Viscous Burger's equation)
$$\begin{cases} \partial_t u + u \partial_x u = \nu \partial_x^2 u & \text{for } a < x < b, t > 0. \\ u(x, 0) = \phi(x) & \text{for } a < x < b. \\ u(a, t) = f(t) \text{ and } u(b, t) = g(t) & \text{for } t > 0. \end{cases}$$

From where

- u is the fluid velocity.
- ν is the kinetic viscosity coefficient.

The **Hopf-Cole transformation** provides an interesting method for solving the above PDE and other higher order PDEs, this describes $u(x, t)$ by a function $w(x, t)$ given as

$$(16) \quad u(x, t) = -2\nu \frac{\partial_x w}{w}.$$

We compute the following derivatives from (16).

$$(17) \quad \begin{cases} \partial_t u &= -2\nu \partial_t \left(\frac{\partial_x w}{w} \right) = -2\nu \frac{w \partial_t \partial_x w - \partial_x w \partial_t w}{w^2} = 2\nu \frac{\partial_x w \partial_t w - w \partial_t \partial_x w}{w^2}. \\ \partial_x u &= -2\nu \partial_x \left(\frac{\partial_x w}{w} \right) = -2\nu \frac{w \partial_x^2 w - (\partial_x w)^2}{w^2} = 2\nu \frac{(\partial_x w)^2 - w \partial_x^2 w}{w^2}. \\ \partial_x^2 u &= -2\nu \partial_x \partial_x \left(\frac{\partial_x w}{w} \right) = 2\nu \frac{-w^2 \partial_x^3 w + 3w \partial_x w \partial_x^2 w - 2w_x^3}{w^3}. \end{cases}$$

Now, replace the equations (16) and (17) into (Viscous Burger's equation).

Left hand side

$$u_t = 2\nu \frac{w_x w_t - w w_{xt}}{w^2}.$$

Right hand side

$$\begin{aligned}\nu u_{xx} - uu_x &= 2\nu^2 \frac{-w^2 w_{xxx} + 3ww_x w_{xx} - 2w_x^3}{w^3} - \left(-2\nu \frac{w_x}{w}\right) 2\nu \frac{w_x^2 - ww_{xx}}{w^2} \\ &= 2\nu^2 \frac{-w^2 w_{xxx} + 3ww_x w_{xx} - 2w_x^3}{w^3} + 4\nu^2 \frac{w_x^3 - ww_x w_{xx}}{w^3} \\ &= -2\nu^2 \frac{ww_{xxx} - w_x w_{xx}}{w^2}.\end{aligned}$$

Finally, we have

$$\text{Left hand side} = \text{Right hand side}$$

$$2\nu \frac{w_x w_t - w w_{xt}}{w^2} = -2\nu^2 \frac{ww_{xxx} - w_x w_{xx}}{w^2}.$$

$$\frac{w_x w_t - w w_{xt}}{w^2} = \nu \frac{w_x w_{xx} - w w_{xxx}}{w^2}.$$

We get

$$\left(\frac{w_t}{w}\right)_x = \nu \left(\frac{w_{xx}}{w}\right)_x.$$

Integrating with respect to x

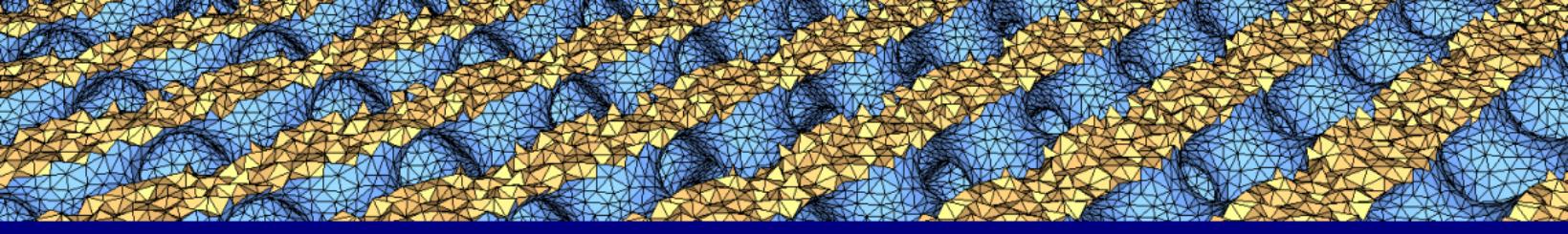
$$\frac{w_t}{w} = \nu \frac{w_{xx}}{w}.$$

(4)

$$w_t = \nu w_{xx}.$$

We see that (Viscous Burger's equation) had been transformed into the heat equation.

Last change: October 28, 2024 at 12:15am.



Laplace operator



Siméon Denis Poisson
(1781–1840).

Theorem (Maximum principle)

Let $u \in C^2(\Omega) \cap C(\overline{\Omega})$.

$$\forall x \in \Omega : \mathcal{L}u(x) \leq 0.$$

- If $c \equiv 0$, then the function u attains its maximum at the boundary

$$\max_{x \in \overline{\Omega}} \leq \max_{x \in \partial\Omega}$$

-

$$\forall x \in \Omega : c(x) \geq 0 \implies \max_{x \in \overline{\Omega}} u(x) \leq \max_{x \in \partial\Omega} \left\{ 0, \max_{x \in \partial\Omega} u(x) \right\}.$$

Laplace's Equation in a Rectangle

Our goal is to find a equilibrium (steady-state) heat distribution in a rectangle $R = [0, a] \times [0, b]$ where we specify the temperature (heat) on the boundary. This means that we are given four functions (of one variable) f_1, f_2, f_3 and f_4 and we wish to solve

$$\begin{cases} u_{xx} + u_{yy} = 0, & 0 < x < a, 0 < y < b, \\ u(x, 0) = f_1(x), & u(x, b) = f_2(x), 0 \leq x \leq a, \\ u(x, 0) = f_1(y), & u(x, b) = f_2(y), 0 \leq x \leq b, \\ u(x, 0) = f_2(y), & u(x, b) = f_2(y), 0 \leq x \leq b, \end{cases}$$

The first thing we note is that it

$$\begin{cases} u(x, 0) = 0, & 0 \leq x < a, 0 < y < b, \\ u(0, y) = 0, & 0 \leq x \leq a, 0 < y \leq b \end{cases}$$

Laplace's Equation in a Disk

$$\begin{cases} \Delta u = 0, & \mathbf{x} \in D = \{(x, y) : x^2 + y^2 < a^2\}, \\ u = h, & \mathbf{x} \in \partial D. \end{cases}$$

Let us solve this problem via separation of variables. Switching to polar coordinates, Rewriting the Laplacian operator in terms of r and θ yields

$$\Delta u = \frac{1}{r} \partial_r u + \frac{1}{r^2} \partial_\theta^2 u.$$

Therefore, we are looking to solve

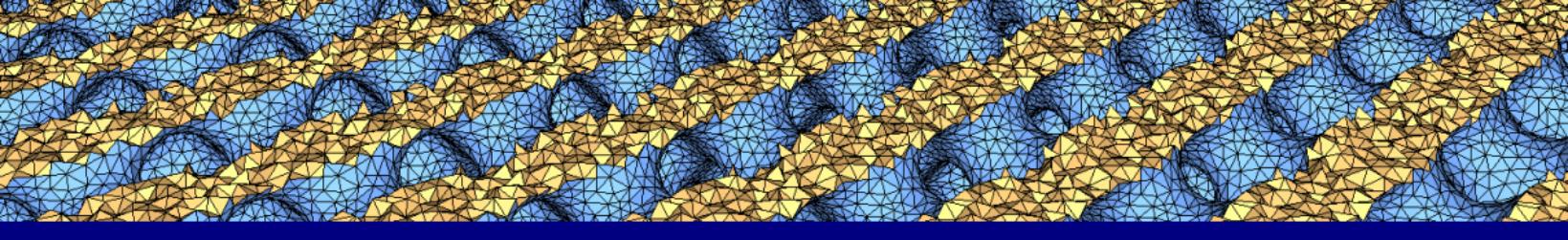
$$\begin{cases} \partial_r^2 u + \frac{1}{r} \partial_r u + \frac{1}{r^2} \partial_\theta^2 u = 0, & r < a, \\ u = h(\theta), & r = a. \end{cases}$$

look for separated solutions of the form $u(r, \theta) = R(r)\Theta(\theta)$ for which the PDE requires that

$$R''\Theta + \frac{1}{r}R'\Theta + \frac{1}{r^2}R\Theta'' = 0.$$

Dividing by $R\Theta$ and multiplying by r^2 , we obtain

$$r^2 \frac{R''}{R} + r \frac{R'}{R} = -\frac{\Theta''}{\Theta} = \lambda.$$



The Separation of Variables Algorithm for Boundary Value Problems

The Separation of Variables Algorithm for Boundary Value Problems

Given a linear PDE with boundary conditions and/or initial conditions, the Separation of Variables Algorithm is based upon the following steps:

- 1 We look for separated solutions to the PDE and the boundary conditions of the form

$$u(x, t) = X(x) T(t).$$

- 2 The boundary conditions carry over to the eigenvalue problem involving $X(x)$. We solve this boundary value / eigenvalue problem to find countably many eigenvalues λ_n for which there exist nontrivial solutions $X_n(x)$.
- 3 We solve the eigenvalue problem of $T(t)$ for each eigenvalue λ_n found in the previous step. We thus arrive at countably many separated solutions

$$u_n(x, t) = X_n(x) T_n(t).$$

to the PDE and the boundary conditions.

- 4 We note that any finite linear combination u_n of these separated solutions will also be a solution to the PDE and the boundary conditions. We boldly consider an infinite linear combination of the form

$$\sum_{n=1}^{\infty} a_n X_n(x) T_n(t).$$

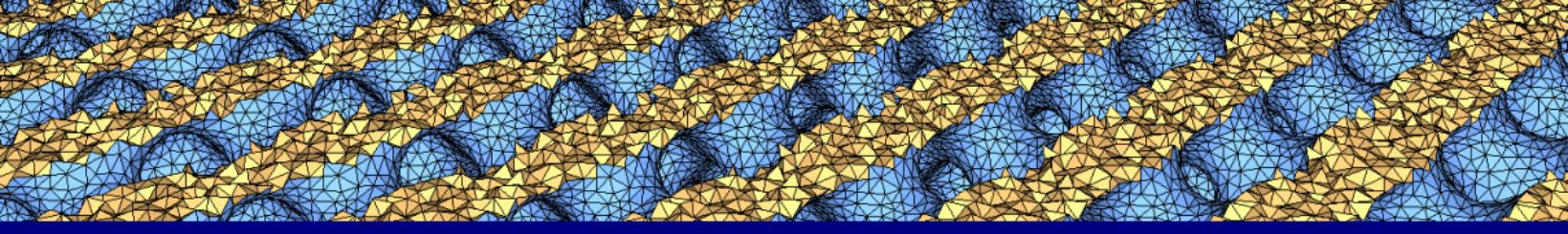
with coefficients a_n .

- 5 We note that achieving the initial conditions amounts to choosing coefficients appropriately. When the eigenvalue problems are for $-\frac{d^2}{dx^2}$ on some interval with a symmetric boundary condition, we arrive at what we previously called a general Fourier series for the data. We find these coefficients by exploiting orthogonality and spanning properties of the eigenfunctions. This effectively means we find each coefficient via projection onto the respective eigenfunction.

Remark

We note that finding these separated solutions reduces to solving eigenvalue problems for each of the components X and T with the same eigenvalue.

Definition (Bessel's functions)



Green's function

Definition (Green's function)

Is a function defined in $\overline{\Omega} \setminus \{x_0\}$ the following hold:

- $\forall x_0 \in \Omega : \exists H_{x_0}(x) \in C(\overline{\Omega})$ such that $\forall x \neq x_0 : G(x, x_0) = \Phi(x - x_0) + H_{x_0}(x)$.
- $\forall x \in \partial\Omega : G(x, x_0) = 0$.

Theorem (Symmetry of the Green's Function)

$$\forall x \neq x_0 \in \Omega : G(x, x_0) = G(x_0, x).$$

Remark

Fundamental solutions and Green's functions (with different boundary conditions) are actually common to all linear PDEs.

Green's Function of the Laplace Equation

0.

Green's Function of the Diffusion Equation

$$\Phi(x, t) := \frac{1}{\sqrt{4\pi\alpha t}} \exp\left(-\frac{x^2}{4\alpha t}\right).$$

Green's Function of the 1D Wave Equation

$$\Phi(x, t) := \frac{1}{2c} H(ct - |x|) = \begin{cases} \frac{1}{2c} & |x| < ct, t > 0 \\ 0 & |x| \geq ct, t > 0 \end{cases}$$

where H is the **Heaviside** function.

Reference list

- Books

-  Daniel Arrigo. *An Introduction to Partial Differential Equations*. Cham: Springer International Publishing, 2023.
ISBN: 978-3-031-22087-6.
-  Rustum Choksi. *Partial Differential Equations: A First Course*. American Mathematical Society, Apr. 2022. ISBN: 978-1-4704-6491-2.
-  Gerald Budge Folland. *Introduction to Partial Differential Equations*. Vol. 102. Princeton University Press, 1995.
-  Abner J. Salgado and Steven M. Wise. *Classical Numerical Analysis: A Comprehensive Course*. Cambridge University Press, Sept. 2022. ISBN: 978-1-108-94260-7.

Reference list

- Articles

-  Igor A. Baratta et al. *DOLFINx: the next generation FEniCS problem solving environment*. preprint. 2023. DOI: [10.5281/zenodo.10447666](https://doi.org/10.5281/zenodo.10447666).
-  Lisandro D. Dalcin et al. "Parallel distributed computing using Python". In: *Advances in Water Resources* 34.9 (2011). New Computational Methods and Software Tools, pp. 1124–1139. ISSN: 0309-1708. DOI: [10.1016/j.advwatres.2011.04.013](https://doi.org/10.1016/j.advwatres.2011.04.013).
-  Alice Harpole et al. "pyro: a framework for hydrodynamics explorations and prototyping". In: *Journal of Open Source Software* 4.34 (2019), p. 1265. DOI: [10.21105/joss.01265](https://doi.org/10.21105/joss.01265). URL: <https://doi.org/10.21105/joss.01265>.
-  S. Kutluay, A.R. Bahadir, and A. Özdeş. "Numerical solution of one-dimensional Burgers equation: explicit and exact-explicit finite difference methods". In: *Journal of Computational and Applied Mathematics* 103.2 (1999), pp. 251–261. ISSN: 0377-0427. DOI: [10.1016/S0377-0427\(98\)00261-1](https://doi.org/10.1016/S0377-0427(98)00261-1).
-  Kyle T Mandli et al. "Clawpack: building an open source ecosystem for solving hyperbolic PDEs". In: *PeerJ Computer Science* 2 (2016), e68. DOI: [10.7717/peerj-cs.68](https://doi.org/10.7717/peerj-cs.68).
-  Aaron Meurer et al. "SymPy: symbolic computing in Python". In: *PeerJ Computer Science* 3 (Jan. 2017), e103. ISSN: 2376-5992. DOI: [10.7717/peerj-cs.103](https://doi.org/10.7717/peerj-cs.103).
-  David Zwicker. "py-pde: A Python package for solving partial differential equations". In: *Journal of Open Source Software* 5.48 (2020), p. 2158. DOI: [10.21105/joss.02158](https://doi.org/10.21105/joss.02158).

Reference list

- Websites

-  Daniel Arrigo. *Math 4315 - PDEs*. URL: <http://www.danielarrigo.com/math-4315---pdes.html> (visited on 09/13/2024).
-  Carlos Aznarán. *The Stone-Weierstraß Theorem*. URL: <https://carlosal1015.github.io/stone-weierstrass/main.beamer.pdf> (visited on 01/16/2023).
-  Michael Eisermann. *Höhere Mathematik 3 für Luft- und Raumfahrttechnik und Materialwissenschaft*. URL: <https://pnp.mathematik.uni-stuttgart.de/igt/eiserm/lehre/HM3/HM3-1x1.pdf> (visited on 01/16/2023).
-  Svetlana Gurevich. *Numerical methods for complex systems I*. URL: https://www.uni-muenster.de/Physik.TP/en/teaching/courses/numerical_methods_for_complex_systems_i_ws2018-2019.html (visited on 09/13/2024).
-  Russell Herman. *A First Course in Partial Differential Equations*. URL: <https://people.uncw.edu/hermanr/pde1/PDE1notes/index.htm> (visited on 09/13/2024).
-  Joachim Krieger. *Ordinary Differential Equations - 2022*. URL: <https://www.epfl.ch/labs/pde/enseignement/equations-differentielles-ordinaires-2022> (visited on 08/27/2024).
-  Mikael Mortensen. *Numeriske metoder for partielle differensialligninger*. URL: <https://www.uio.no/studier/emner/matnat/math/MAT-MEK4270> (visited on 09/13/2024).