Applied Mathematics 205

Unit III: Numerical Calculus

Lecturer: Dr. David Knezevic

Unit III: Numerical Calculus

Chapter III.3: Boundary Value Problems and PDEs

ODE Boundary Value Problems

ODE BVPs

Consider the ODE Boundary Value Problem (BVP):[1] find $u \in C^2[a, b]$ such that

$$-\alpha u''(x) + \beta u'(x) + \gamma u(x) = f(x), \quad x \in [a, b]$$

for $\alpha, \beta, \gamma \in \mathbb{R}$ and $f : \mathbb{R} \to \mathbb{R}$

The terms in this ODE have standard names:

$-\alpha u''(x)$:	diffusion term
$\beta u'(x)$:	convection (or transport) term
$\gamma u(x)$:	reaction term
$f(x)$:	source term

[1] Often called a "Two-point boundary value problem"

ODE BVPs

Also, since this is a BVP u must satisfy some boundary conditions, e.g. $u(a) = c_1$, $u(b) = c_2$

$u(a) = c_1$, $u(b) = c_2$ are called Dirichlet boundary conditions

Can also have:

- A Neumann boundary condition: $u'(b) = c_2$
- A Robin (or "mixed") boundary condition:[2] $u'(b) + c_2 u(b) = c_3$

[2] With $c_2 = 0$, this is a Neumann condition

ODE BVPs

This is an ODE, so we could try to use the ODE solvers from III.3 to solve it!

Question: How would we make sure the solution satisfies $u(b) = c_2$?

ODE BVPs

Answer: Solve the IVP with $u(a) = c_1$ and $u'(a) = s_0$, and then update s_k iteratively for $k = 1, 2, \ldots$ until $u(b) = c_2$ is satisfied

This is called the "shooting method", we picture it as shooting a projectile to hit a target at $x = b$

However, the shooting method does not generalize to PDEs hence it's not broadly useful: we will not cover it in AM205

ODE BVPs

A more general approach is to formulate a coupled system of equations for the BVP based on a finite difference approximation

Suppose we have a grid $x_i = a + (i-1)h$, $i = 1, 2, \ldots, n$, where $h = (b-a)/(n-1)$

Then our approximation to $u \in C^2[a, b]$ is represented by a vector $U \in \mathbb{R}^n$, where $U_i \approx u(x_i)$

ODE BVPs

Recall the ODE:

$$-\alpha u''(x) + \beta u'(x) + \gamma u(x) = f(x), \quad x \in [a, b]$$

Let's develop an approximation for each term in the ODE

For the reaction term γu, we have the pointwise approximation $\gamma U_i \approx \gamma u(x_i)$

ODE BVPs

Similarly, for the derivative terms:

- Let $D_2 \in \mathbb{R}^{n \times n}$ denote diff. matrix for the second derivative
- Let $D_1 \in \mathbb{R}^{n \times n}$ denote diff. matrix for the first derivative

Then $-\alpha(D_2 U)_i \approx -\alpha u''(x_i)$ and $\beta(D_1 U)_i \approx \beta u'(x_i)$

Hence, we obtain $(AU)_i \approx -\alpha u''(x_i) + \beta u'(x_i) + \gamma u(x_i)$, where $A \in \mathbb{R}^{n \times n}$ is:

$$A \equiv -\alpha D_2 + \beta D_1 + \gamma \mathrm{I}$$

Similarly, we represent the right hand side by sampling f at the grid points, hence we introduce $F \in \mathbb{R}^n$, where $F_i = f(x_i)$

ODE BVPs

Therefore, we obtain the linear system for $U \in \mathbb{R}^n$:

$$AU = F$$

Hence, we have converted a linear differential equation into a system of linear algebraic equations

(Similarly we can convert a nonlinear differential equation into a system of nonlinear algebraic equations, see Assignment 4)

However, we are not finished yet, need to account for the boundary conditions!

ODE BVPs

Dirichlet boundary conditions: we need to impose $U_1 = c_1$, $U_n = c_2$

Since we fix U_1 and U_n, they are no longer variables: we should eliminate them from our linear system

However, instead of removing rows and columns from A, it is slightly simpler from the implementational point of view to:

- "zero out" first row of A, then set $A(1,1) = 1$ and $F_1 = c_1$
- "zero out" last row of A, then set $A(n,n) = 1$ and $F_n = c_2$

ODE BVPs

Let's implement this approach for solving ODE BVPs in Matlab

And let's use the following approach[3] to check that our implementation is correct:

1. choose a solution u that satisfies the BCs
2. substitute u into the ODE to get a right-hand side f
3. compute the ODE approximation with f from step 2
4. verify that you get the expected convergence rate for the approximation to u

[3] Sometimes called the "method of manufactured solutions"

ODE BVPs

For example, consider $x \in [0,1]$, $u(0) = u(1) = 0$, and set $u(x) = e^x \sin(2\pi x)$

This then implies that:

$$\begin{aligned} f(x) &\equiv -\alpha u''(x) + \beta u'(x) + \gamma u(x) \\ &= -\alpha e^x \left[4\pi \cos(2\pi x) + (1 - 4\pi^2) \sin(2\pi x)\right] + \\ &\quad \beta e^x \left[\sin(2\pi x) + 2\pi \cos(2\pi x)\right] + \gamma e^x \sin(2\pi x) \end{aligned}$$

ODE BVPs

Matlab example: ODE BVP via finite differences

Convergence results (using infinity norm to compute the error):

h	error
2.0e-2	5.07e-3
1.0e-2	1.26e-3
5.0e-3	3.17e-4
2.5e-3	7.92e-5

$O(h^2)$, as expected due to second order differentiation matrices

ODE BVPs: BCs involving derivatives

Question: How would we impose the Robin boundary condition $u'(b) + c_2 u(b) = c_3$, and preserve the $O(h^2)$ convergence rate?

Option 1: Introduce a "ghost node" at $x_{n+1} = b + h$, this node is involved in both the B.C. and the n^{th} matrix row

Employ central difference approx. to $u'(b)$ to get approx. B.C.:

$$\frac{U_{n+1} - U_{n-1}}{2h} + c_2 U_n = c_3,$$

or equivalently

$$U_{n+1} = U_{n-1} - 2hc_2 U_n + 2hc_3$$

ODE BVPs: BCs involving derivatives

The n^{th} equation is

$$-\alpha \frac{U_{n-1} - 2U_n + U_{n+1}}{h^2} + \beta \frac{U_{n+1} - U_{n-1}}{2h} + \gamma U_n = F_n$$

We can substitute our expression for U_{n+1} into the above equation, and hence eliminate U_{n+1}:

$$\left(-\frac{2\alpha c_3}{h} + \beta c_3\right) - \frac{2\alpha}{h^2} U_{n-1} + \left(\frac{2\alpha}{h^2}(1 + hc_2) - \beta c_2 + \gamma\right) U_n = F_n$$

Set $F_n \leftarrow F_n - \left(-\frac{2\alpha c_3}{h} + \beta c_3\right)$, we get $n \times n$ system $AU = F$

Option 2: Use a second-order one-sided difference formula for $u'(b)$ in the Robin BC

Partial Differential Equations

PDEs

As discussed in III.1, Partial Differential Equations (PDEs) are a natural generalization of ODEs

There are three main classes of PDEs:[4]

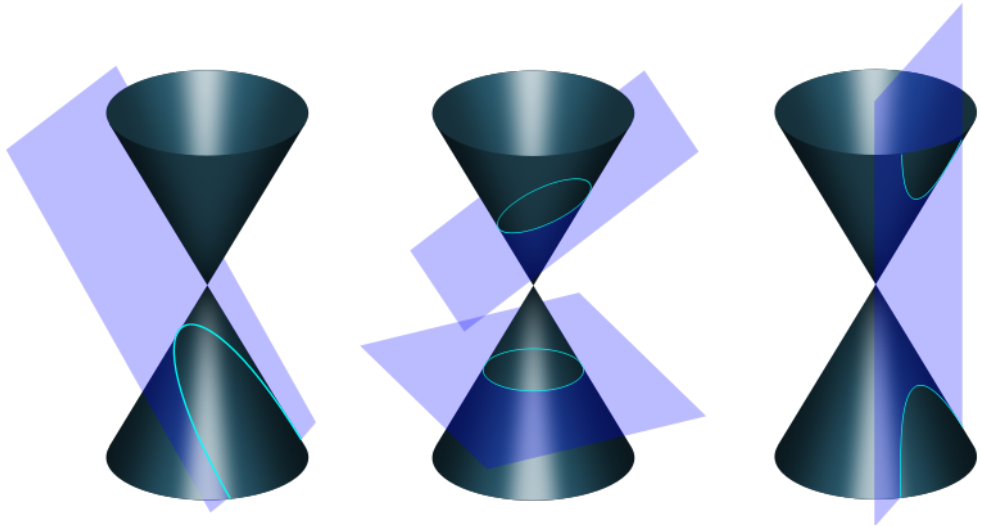
equation type	prototypical example	equation
hyperbolic	wave equation	$u_{tt} - u_{xx} = 0$
parabolic	heat equation	$u_t - u_{xx} = f$
elliptic	Poisson equation	$u_{xx} + u_{yy} = f$

Question: Where do these names come from?

[4] Notation: $u_x \equiv \frac{\partial u}{\partial x}$, $u_{xy} \equiv \frac{\partial}{\partial y}\left(\frac{\partial u}{\partial x}\right)$

PDEs

Answer: The names are related to conic sections



Conic sections are defined algebraically by a quadratic function:

$$q(x, y) = ax^2 + bxy + cy^2 + dx + ey + f$$

This "looks like" the general form of a second-order PDE:

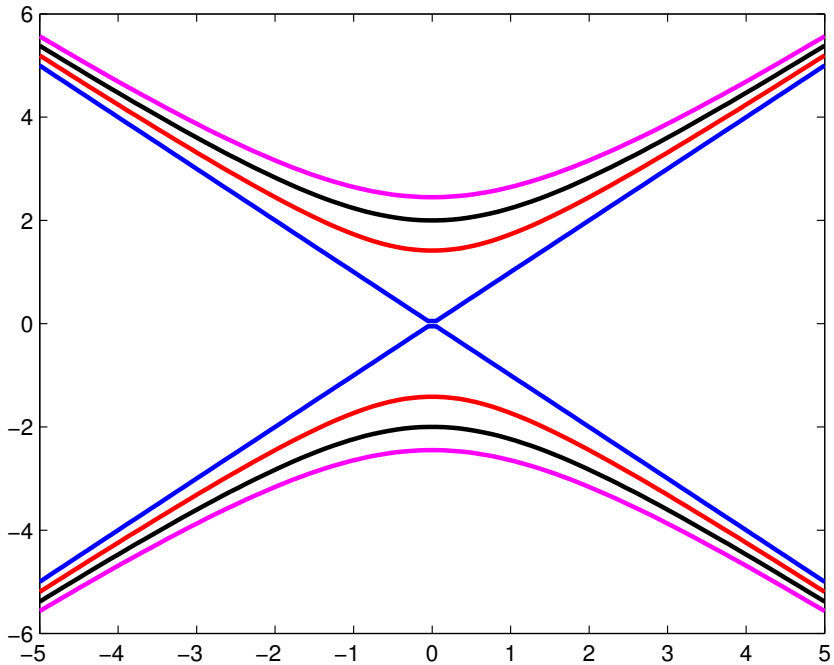
$$au_{xx} + bu_{xy} + cu_{yy} + du_x + eu_y + fu + g = 0$$

PDEs: Hyperbolic

Wave equation: $u_{tt} - u_{xx} = 0$

Corresponding quadratic function is $q(x, t) = t^2 - x^2$

$q(x, t) = c$ gives a hyperbola, e.g. for $c = 0 : 2 : 6$, we have

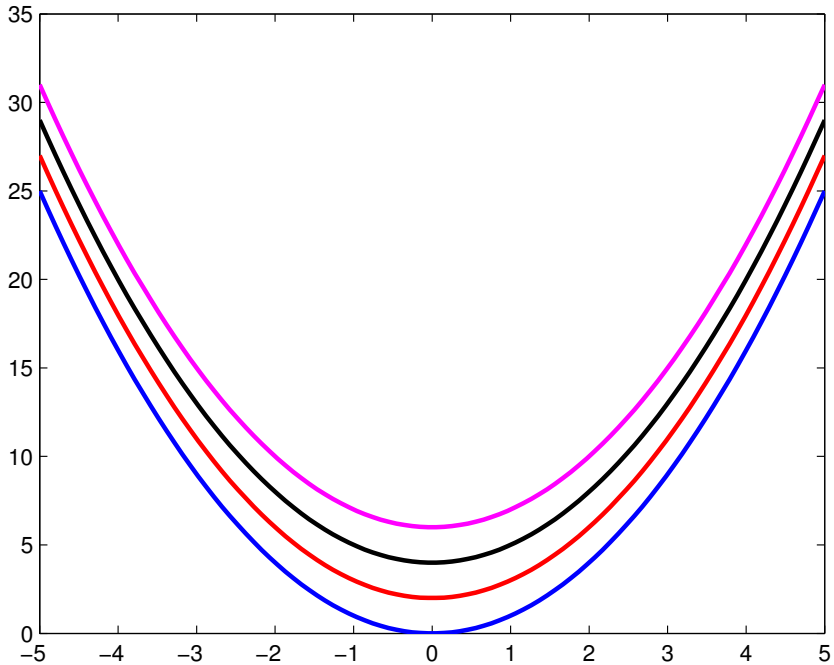


PDEs: Parabolic

Heat equation: $u_t - u_{xx} = 0$

Corresponding quadratic function is $q(x, t) = t - x^2$

$q(x, t) = c$ gives a parabola, e.g. for $c = 0 : 2 : 6$, we have

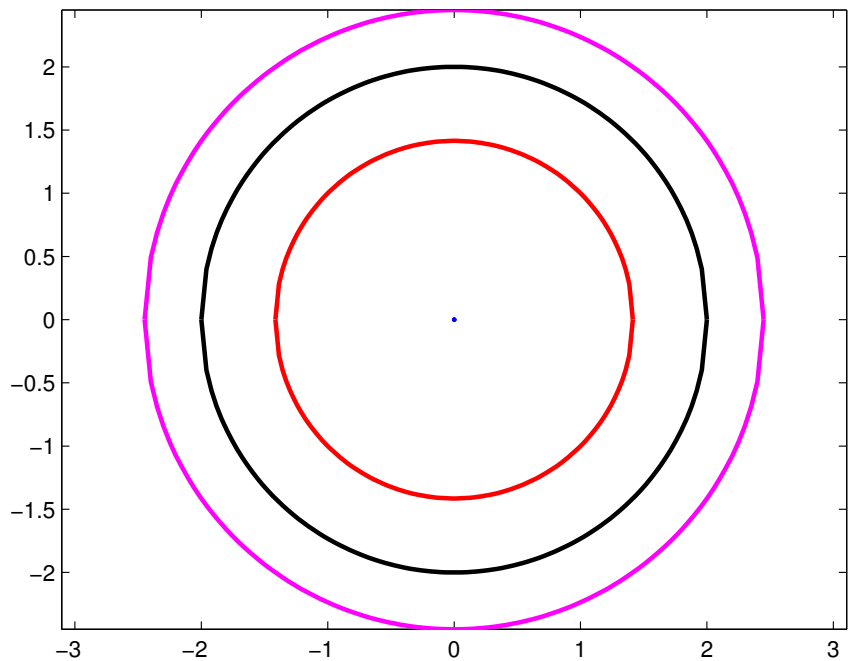


PDEs: Elliptic

Poisson equation: $u_{xx} + u_{yy} = f$

Corresponding quadratic function is $q(x, y) = x^2 + y^2$

$q(x, y) = c$ gives an ellipse, e.g. for $c = 0 : 2 : 6$, we have



PDEs

In general, it is not so easy to classify PDEs using conic section naming

Many problems don't strictly fit into the classification scheme (e.g. nonlinear, or higher order, or variable coefficient equations)

Nevertheless, the names hyperbolic, parabolic, elliptic are the standard ways of describing PDEs, based on the following criteria:

- Hyperbolic: time-dependent, conservative physical process, no steady state
- Parabolic: time-dependent, dissipative physical process, evolves towards steady state
- Elliptic: time-independent, equilibrium/steady-state

Hyperbolic PDEs

Hyperbolic PDEs

We introduced the wave equation $u_{tt} - u_{xx} = 0$ above

Note that the system of first order PDEs

$$\begin{aligned} u_t + v_x &= 0 \\ v_t + u_x &= 0 \end{aligned}$$

is equivalent to the wave equation, since

$$u_{tt} = (u_t)_t = (-v_x)_t = -(v_t)_x = -(-u_x)_x = u_{xx}$$

(This assumes that u, v are smooth enough for us to switch the order of the partial derivatives)

Hyperbolic PDEs

The first-order PDEs from above are linear advection equations

$$u_t + cu_x = 0$$

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

It's a first order PDE, hence doesn't fit our conic section description, but it is:

- time-dependent
- conservative
- not evolving toward steady state

$\Longrightarrow$ hyperbolic!

A second-order hyperbolic PDE can always be formulated as a system of first-order PDEs, hence we focus on the first-order case

Hyperbolic PDEs

Exact solution of the advection equation: $u(x,t) = u_0(x - ct)$

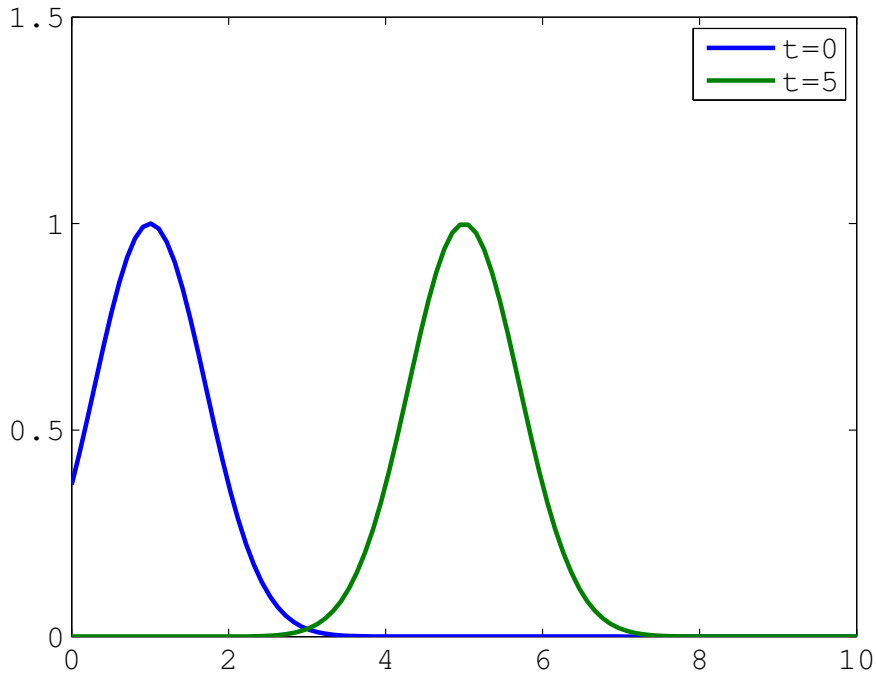
To see this, let $z(x,t) \equiv x - ct$, then from the chain rule we have

$$\begin{aligned}
\frac{\partial}{\partial t} u_0(x-ct) + c\frac{\partial}{\partial x} u_0(x-ct) &= \frac{\partial}{\partial t} u_0(z(x,t)) + c\frac{\partial}{\partial x} u_0(z(x,t)) \\
&= u_0'(z)\frac{\partial z}{\partial t} + cu_0'(z)\frac{\partial z}{\partial x} \\
&= -cu_0'(z) + cu_0'(z) \\
&= 0
\end{aligned}$$

Hyperbolic PDEs

This tells us that the solution transports (or advects) the initial condition with "speed" c

e.g. with $c = 1$ and an initial condition $u_0(x) = e^{-(1-x)^2}$ we have:



Hyperbolic PDEs

We can understand the behavior of hyperbolic PDEs in more detail by considering characteristics

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

For $u_t + cu_x = 0$ we have $X(t) = X_0 + ct$,[5] since

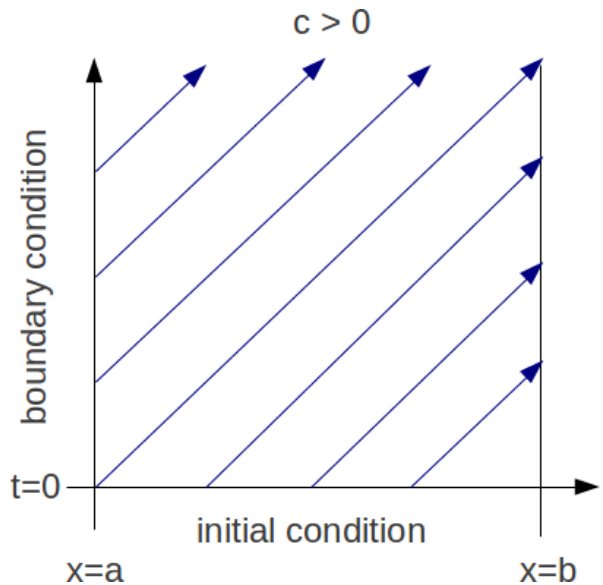
$$\begin{aligned} \frac{\mathrm{d}}{\mathrm{d}t} u(X(t), t) &= u_t(X(t), t) + u_x(X(t), t) \frac{\mathrm{d}X(t)}{\mathrm{d}t} \\ &= u_t(X(t), t) + cu_x(X(t), t) \\ &= 0 \end{aligned}$$

[5] Each different choice of X_0 gives a distinct characteristic curve

Hyperbolic PDEs

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

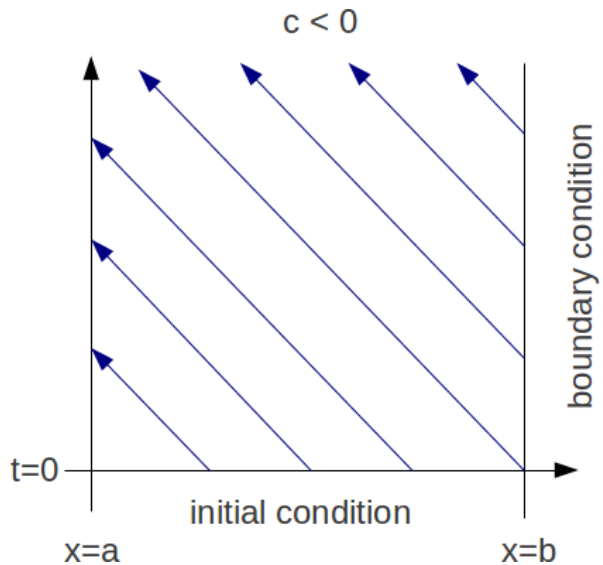


Must impose BC at $x = a$, cannot impose BC at $x = b$

Hyperbolic PDEs

Hence $u(X(t), t) = u(0, X(0)) = u_0(X_0)$, i.e. the initial condition is transported along characteristics

Characteristics have important implications for the direction of flow of information, and for boundary conditions



Must impose BC at $x = b$, cannot impose BC at $x = a$

Hyperbolic PDEs: More Complicated Characteristics

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 $u_t + cu_x = f(t, x, u(t, x))$, and $X(t) = X_0 + ct$

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}u(X(t), t) &= u_t(X(t), t) + u_x(X(t), t)\frac{\mathrm{d}X(t)}{\mathrm{d}t} \\ &= u_t(X(t), t) + cu_x(X(t), t) \\ &= f(t, X(t), u(X(t), t))\end{aligned}$$

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:

$$u(X(t), t) = u_0(X_0) + \int_0^t f(t, X(t), u(X(t), t)\mathrm{d}t$$

Hyperbolic PDEs: More Complicated Characteristics

We can also find characteristics for variable coefficient advection

Exercise: Verify that the characteristic curve for $u_t + c(t,x)u_x = 0$ is given by

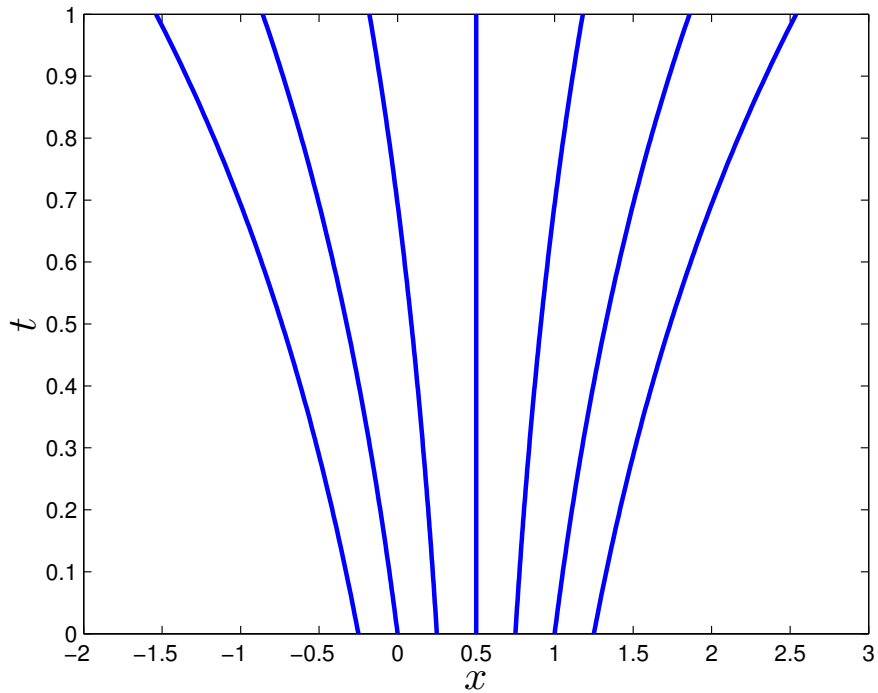
$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} = c(t, X(t))$$

In this case, we have to solve an ODE to obtain the curve $(X(t), t)$ in the xt-plane

e.g. for $c(t,x) = x - 1/2$, we get $X(t) = 1/2 + (X_0 - 1/2)e^t$

Hyperbolic PDEs: More Complicated Characteristics

Hence for for $c(t, x) = x - 1/2$ the characteristics $(X(t), t)$ "bend away" from $x = 1/2$



Characteristics also apply to nonlinear hyperbolic PDEs (e.g. Burger's equation), but this is outside the scope of AM205

Hyperbolic PDEs: Numerical Approximation

We now consider how to solve $u_t + cu_x = 0$ equation using a finite difference method

Question: Why finite differences? Why not just use characteristics?

Answer: Characteristics actually are a viable option for computational methods, and are used in practice

However, characteristic methods can become very complicated in 2D or 3D, or for nonlinear problems

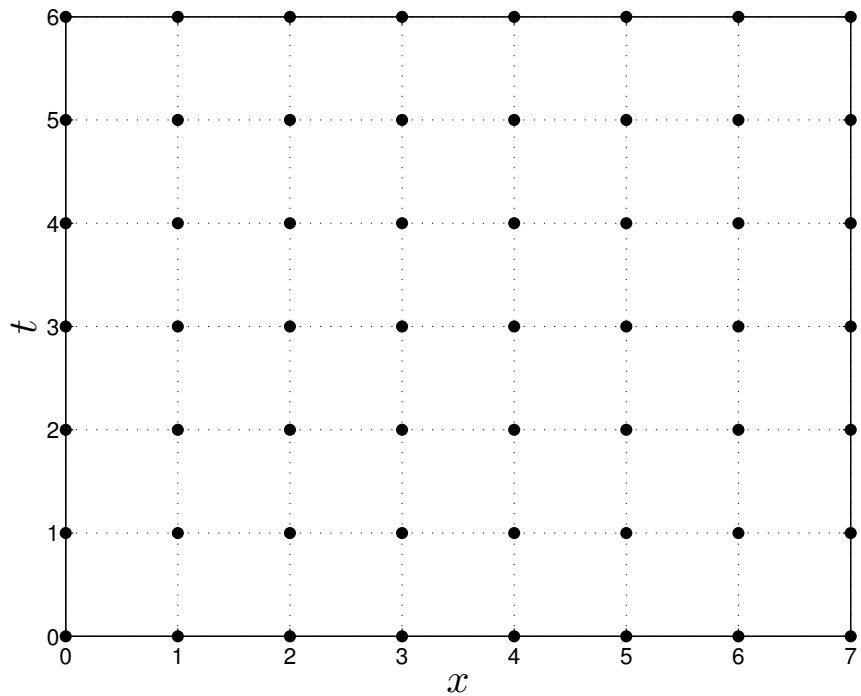
Finite differences are a much more practical choice in most circumstances

Hyperbolic PDEs: Numerical Approximation

Advection equation is an Initial Boundary Value Problem (IBVP)

We impose an initial condition, and a boundary condition (only one BC since first order PDE)

A finite difference approximation leads to a grid in the xt-plane



Hyperbolic PDEs: Numerical Approximation

The first step in developing a finite difference approximation for the advection equation is to consider the CFL condition[6]

The CFL condition is a necessary condition for the convergence of a finite difference approximation of a hyperbolic problem

Suppose we discretize $u_t + cu_x = 0$ in space and time using the explicit (in time) scheme

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

Here $U_j^n \approx u(t_n, x_j)$, where $t_n = n\Delta t$, $x_j = j\Delta x$

[6] Courant-Friedrichs-Lewy condition, published in 1928

Hyperbolic PDEs: Numerical Approximation

This can be rewritten as

$$\begin{aligned} U_j^{n+1} &= U_j^n - \frac{c\Delta t}{\Delta x}(U_j^n - U_{j-1}^n) \\ &= (1-\nu)U_j^n + \nu U_{j-1}^n \end{aligned}$$

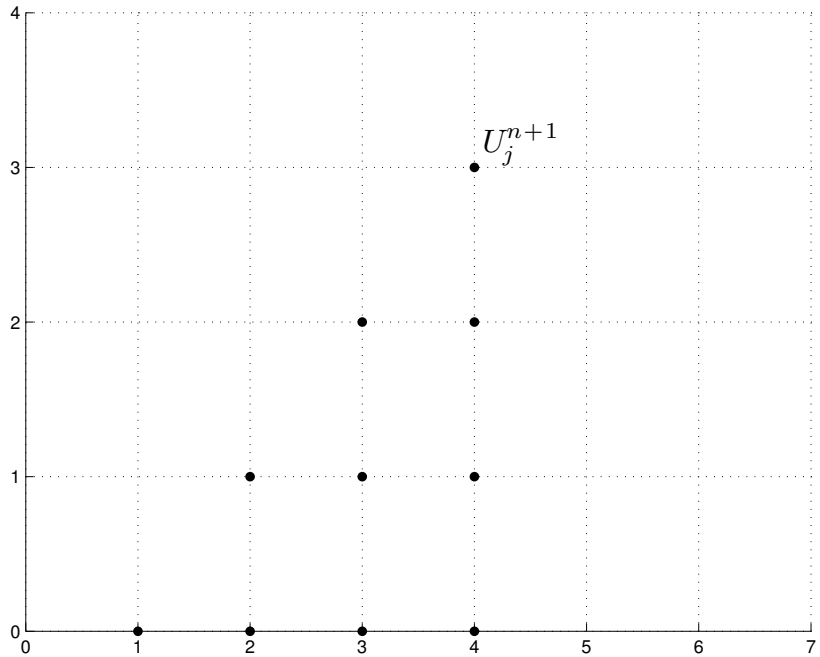
where

$$\nu \equiv \frac{c\Delta t}{\Delta x}$$

We can see that U_j^{n+1} depends only on U_j^n and U_{j-1}^n

Hyperbolic PDEs: Numerical Approximation

Definition: Domain of dependence of U_j^{n+1} is the set of values that U_j^{n+1} depends on



Hyperbolic PDEs: Numerical Approximation

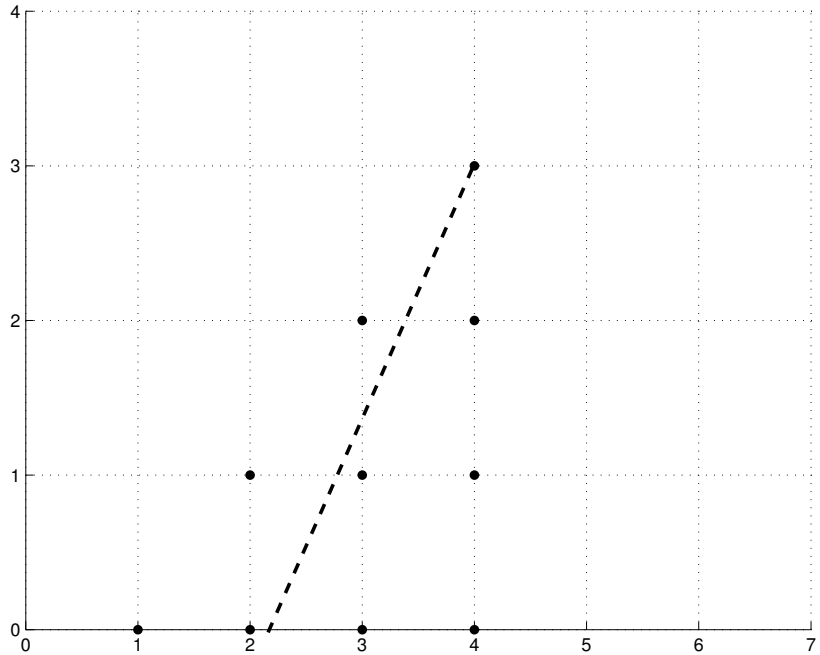
The domain of dependence of the exact solution $u(t_{n+1}, x_j)$ is determined by the characteristic curve passing through (t_{n+1}, x_j)

CFL Condition:

> For a convergent scheme, the domain of dependence of the PDE must lie within the domain of dependence of the numerical method

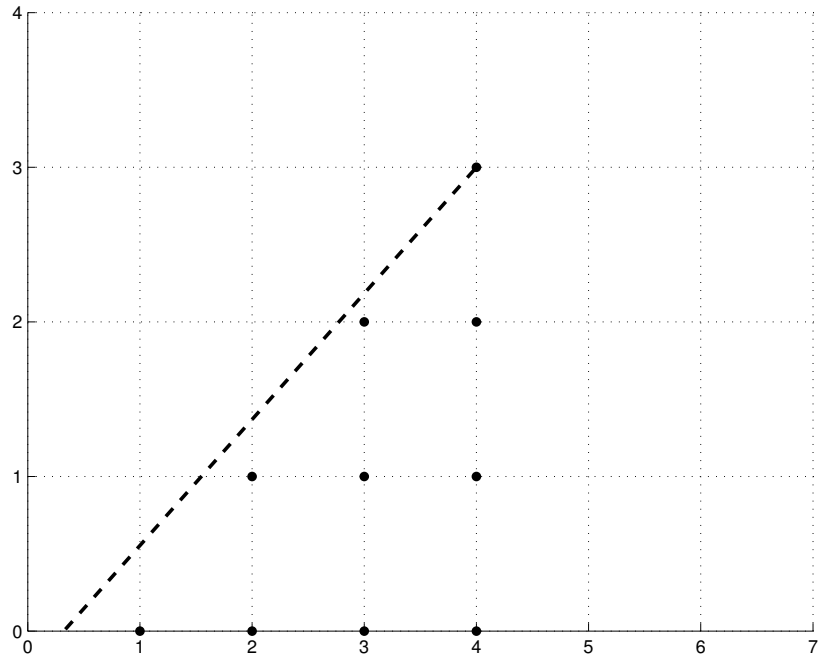
Hyperbolic PDEs: Numerical Approximation

Suppose the dashed line indicates characteristic passing through (t_{n+1}, x_j), then the scheme below satisfies the CFL condition



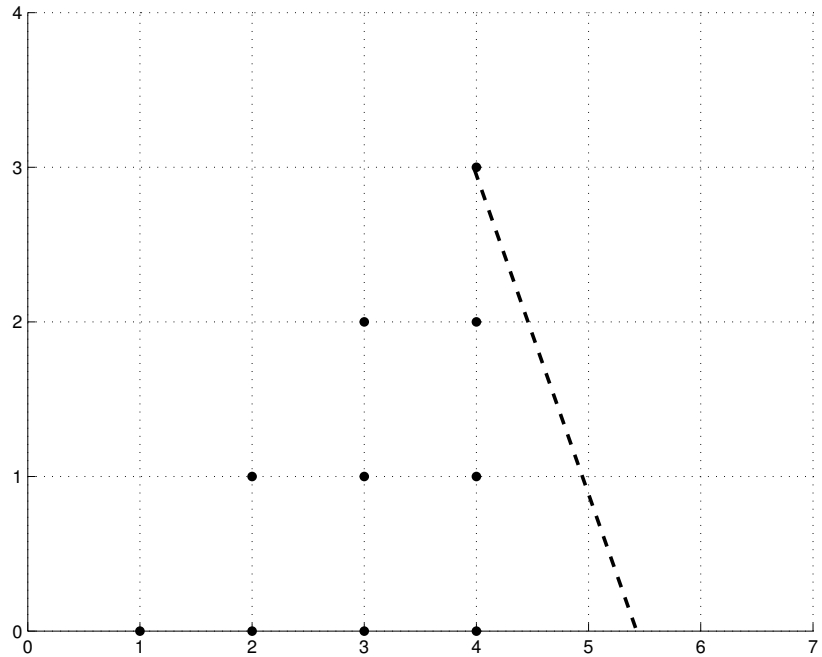
Hyperbolic PDEs: Numerical Approximation

The scheme below does not satisfy the CFL condition



Hyperbolic PDEs: Numerical Approximation

The scheme below does not satisfy the CFL condition (here $c < 0$)



Hyperbolic PDEs: Numerical Approximation

Question: What goes wrong if the CFL condition is violated?

Hyperbolic PDEs: Numerical Approximation

Answer: The exact solution $u(x,t)$ depends on initial value $u_0(x_0)$, which is outside the numerical method's domain of dependence

Therefore, the numerical approx. to $u(x,t)$ is "insensitive" to the value $u_0(x_0)$, which means that the method cannot be convergent

Hyperbolic PDEs: Numerical Approximation

Note that CFL is only a necessary condition for convergence

Its great value is its simplicity: CFL allows us to easily reject F.D. schemes for hyperbolic problems with very little investigation

For example, for $u_t + cu_x = 0$, the scheme

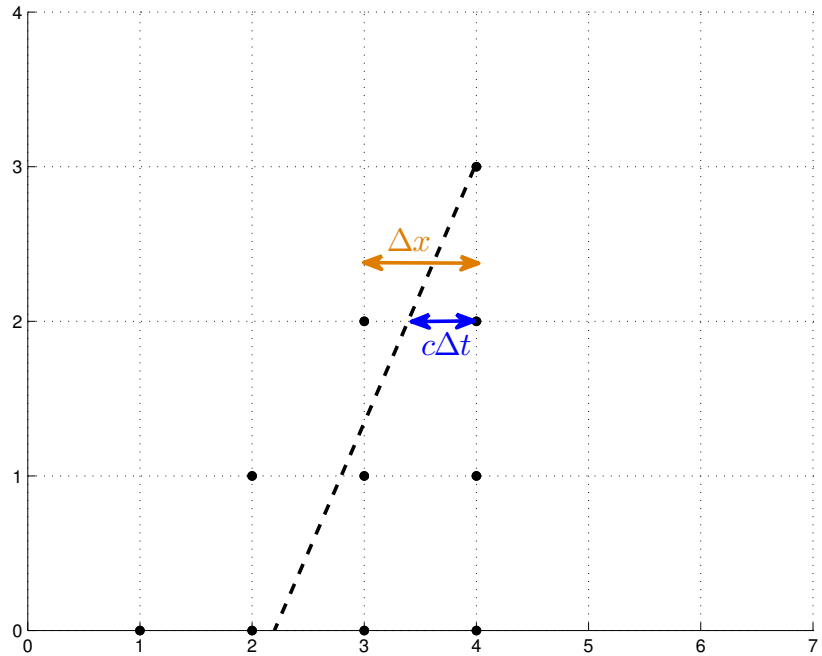
$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + c\frac{U_j^n - U_{j-1}^n}{\Delta x} = 0 \qquad (*)$$

cannot be convergent if $c < 0$

Question: What small change to $(*)$ would give a better method when $c < 0$?

Hyperbolic PDEs: Numerical Approximation

If $c > 0$, then we require $\nu \equiv \frac{c\Delta t}{\Delta x} \leq 1$ in $(*)$ for CFL to be satisfied



Hyperbolic PDEs: Upwind method

As foreshadowed earlier, we should pick our method to reflect the direction of propagation of information

This motivates the upwind scheme for $u_t + cu_x = 0$

$$U_j^{n+1} = \begin{cases} U_j^n - c\frac{\Delta t}{\Delta x}(U_j^n - U_{j-1}^n), & \text{if } c > 0 \\ U_j^n - c\frac{\Delta t}{\Delta x}(U_{j+1}^n - U_j^n), & \text{if } c < 0 \end{cases}$$

The upwind scheme satisfies CFL condition if $|\nu| \equiv |c\Delta t/\Delta x| \leq 1$

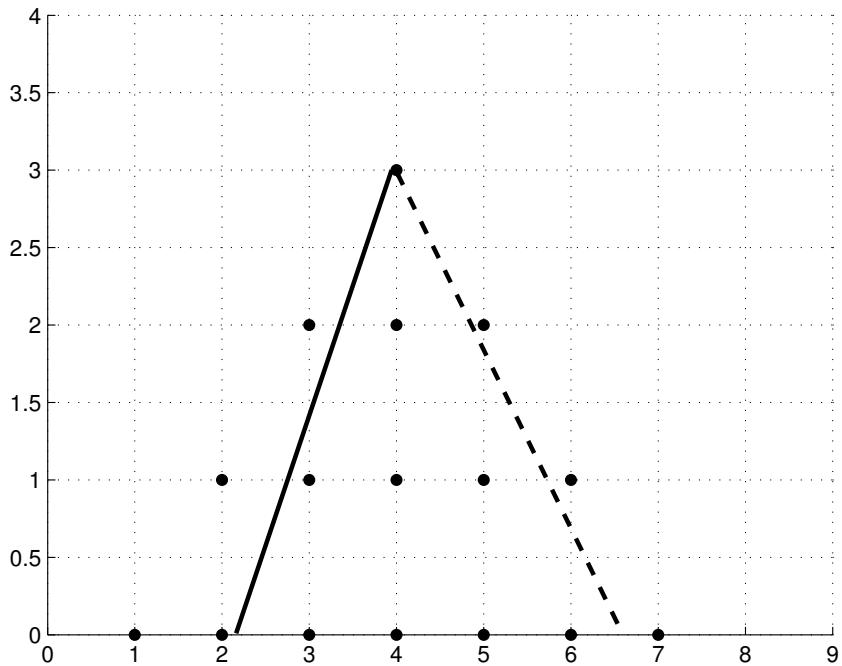
ν is often called the CFL number

Hyperbolic PDEs: Central difference method

Another method that seems appealing is the central difference method:

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

This satisfies CFL for $|\nu| \equiv |c\Delta t/\Delta x| \leq 1$, regardless of sign$(c)$



We shall see shortly, however, that this is a bad method!

Hyperbolic PDEs: Accuracy

Recall from III.3 that truncation error is "what is left over when we substitute exact solution into the numerical approximation"

Truncation error is analogous for PDEs, e.g. for the ($c > 0$) upwind method, truncation error is:

$$T_j^n \equiv \frac{u(t^{n+1}, x_j) - u(t^n, x_j)}{\Delta t} + c\frac{u(t^n, x_j) - u(t^n, x_{j-1})}{\Delta x}$$

The order of accuracy is then the largest p such that

$$T_j^n = O((\Delta x)^p + (\Delta t)^p)$$

Hyperbolic PDEs: Accuracy

See Lecture: For the upwind method, we have

$$T_j^n = \frac{1}{2}\left[\Delta t u_{tt}(t^n, x_j) - c\Delta x u_{xx}(t^n, x_j)\right] + \text{H.O.T.}$$

Hence the upwind scheme is first order accurate

Hyperbolic PDEs: Accuracy

Just like with ODEs, truncation error is related to convergence in the limit $\Delta t, \Delta x \to 0$

Note here we're interested in taking the limit of two values simultaneously, Δt and Δx

Hence when we consider truncation error here, we need to decide on a relationship between Δt and Δx

e.g. to let $\Delta t, \Delta x \to 0$ for the upwind scheme, we would set $\frac{c\Delta t}{\Delta x} = \nu \in (0, 1]$; this ensures CFL is satisfied for all $\Delta x, \Delta t$

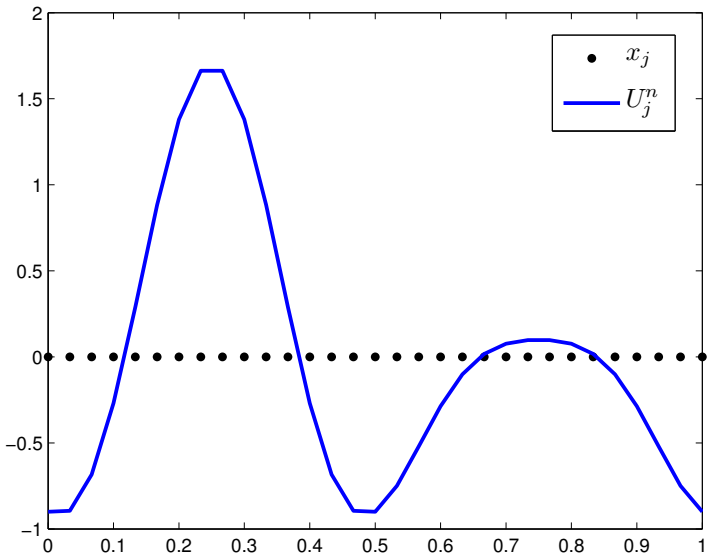
Hyperbolic PDEs: Accuracy

In general, convergence of a finite difference method for a PDE is related to both its truncation error and its stability

We'll discuss this in more detail shortly, but first we consider how to analyze stability via Fourier stability analysis

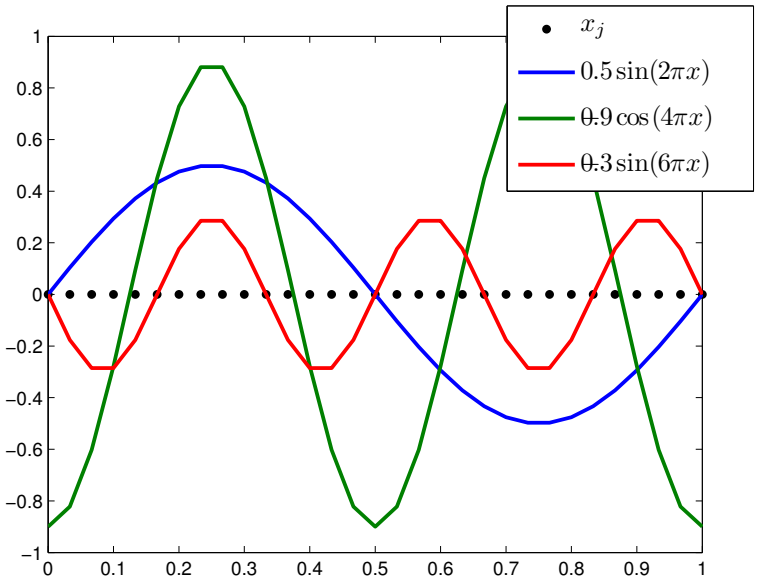
Hyperbolic PDEs: Stability

Let's suppose that U_j^n is periodic on the grid $x_1, x_2, \ldots, x_n$



Hyperbolic PDEs: Stability

Then we can represent U_j^n as a linear combination of sin and cos functions, i.e. Fourier modes



Or, equivalently, as a linear combination of complex exponentials, since $e^{ikx} = \cos(kx) + i\sin(kx)$ so that

$$\sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix}), \qquad \cos(x) = \frac{1}{2}(e^{ix} + e^{-ix})$$

Hyperbolic PDEs: Stability

For simplicity, let's just focus on only one of the Fourier modes

In particular, we consider the ansatz $U_j^n(k) \equiv \lambda(k)^n e^{ikx_j}$, where k is the wave number and $\lambda(k) \in \mathbb{C}$

Key idea: Suppose that $U_j^n(k)$ satisfies our finite difference equation, then this will allow us to solve for $\lambda(k)$

The value of $|\lambda(k)|$ indicates whether the Fourier mode e^{ikx_j} is amplified or damped

If $|\lambda(k)| \leq 1$ for all k then the scheme does not amplify any Fourier modes $\implies$ stable!

Hyperbolic PDEs: Stability

We now perform Fourier stability analysis for the ($c > 0$) upwind scheme (recall that $\nu = \frac{c\Delta t}{\Delta x}$):

$$U_j^{n+1} = U_j^n - \nu(U_j^n - U_{j-1}^n)$$

Substituting in $U_j^n(k) = \lambda(k)^n e^{ik(j\Delta x)}$ gives

$$\begin{aligned} \lambda(k)e^{ik(j\Delta x)} &= e^{ik(j\Delta x)} - \nu(e^{ik(j\Delta x)} - e^{ik((j-1)\Delta x)}) \\ &= e^{ik(j\Delta x)} - \nu e^{ik(j\Delta x)}(1 - e^{-ik\Delta x}) \end{aligned}$$

Hence

$$\lambda(k) = 1 - \nu(1 - e^{-ik\Delta x}) = 1 - \nu(1 - \cos(k\Delta x) + i\sin(k\Delta x))$$

Hyperbolic PDEs: Stability

It follows that

$$\begin{aligned} |\lambda(k)|^2 &= [(1-\nu)+\nu\cos(k\Delta x)]^2 + [\nu\sin(k\Delta x)]^2 \\ &= (1-\nu)^2 + \nu^2 + 2\nu(1-\nu)\cos(k\Delta x) \\ &= 1 - 2\nu(1-\nu)(1-\cos(k\Delta x)) \end{aligned}$$

and from the trig. identity $(1-\cos(\theta)) = 2\sin^2(\frac{\theta}{2})$, we have

$$|\lambda(k)|^2 = 1 - 4\nu(1-\nu)\sin^2\left(\frac{1}{2}k\Delta x\right)$$

Due to the CFL condition, we first suppose that $0 \leq \nu \leq 1$

This implies $0 \leq \nu(1-\nu) \leq 1/4$, so that
$0 \leq 4\nu(1-\nu)\sin^2\left(\frac{1}{2}k\Delta x\right) \leq 1$, and hence $|\lambda(k)| \leq 1$

Hyperbolic PDEs: Stability

In contrast, consider stability of the central difference approx.:

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

Recall that this also satisfies the CFL condition as long as $|\nu| \leq 1$

But Fourier stability analysis yields

$$\lambda(k) = 1 - \nu i \sin(k\Delta x) \implies |\lambda(k)|^2 = 1 + \nu^2 \sin^2(k\Delta x)$$

and hence $|\lambda(k)| > 1$ (unless $\sin(k\Delta x) = 0$), i.e. unstable!

Consistency

We say that a numerical scheme is consistent with a PDE if its truncation error tends to zero as $\Delta x, \Delta t \to 0$

For example, any first (or higher) order scheme is consistent

Lax Equivalence Theorem

Then a fundamental theorem in Scientific Computing is the Lax[7] Equivalence Theorem:

> For a consistent finite difference approx. to a linear evolutionary problem, the stability of the scheme is necessary and sufficient for convergence

This theorem refers to linear evolutionary problems, e.g linear hyperbolic or parabolic PDEs

[7] Peter Lax, Courant Institute, NYU

Lax Equivalence Theorem

We know how to check consistency: Analyze the truncation error

We know how to check stability: Fourier stability analysis

Hence, from Lax, we have a general approach for verifying convergence

Also, as with ODEs, convergence rate is determined by truncation error

Lax Equivalence Theorem

Note that strictly speaking Fourier stability analysis only applies for periodic problems

However, it can be shown that conclusions of Fourier stability analysis hold true more generally

Hence Fourier stability analysis is the standard tool for examining stability of finite difference methods for PDEs

Hyperbolic PDEs: Semi-discretization

So far, we have developed full discretizations (both space and time) of the advection equation, and considered accuracy and stability

However, it can be helpful to consider semi-discretizations, where we discretize only in space, or only in time

For example, discretizing $u_t + c(t,x)u_x = 0$ in space[8] using a backward difference formula gives

$$\frac{\partial U_j(t)}{\partial t} + c_j(t)\frac{U_j(t) - U_{j-1}(t)}{\Delta x} = 0, \qquad j = 1, \ldots, n$$

[8] Here we show an example where c is not constant

Hyperbolic PDEs: Semi-discretization

This gives a system of ODEs, $U_t = f(t, U(t))$, where $U(t) \in \mathbb{R}^n$ and

$$f(t, U(t)) \equiv -c_j(t)\frac{U_j(t) - U_{j-1}(t)}{\Delta x}$$

We could approximate this ODE using forward Euler (to get our Upwind scheme):

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = f(t^n, U^n) = -c_j^n \frac{U_j^n - U_{j-1}^n}{\Delta x}$$

Or backward Euler:

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = f(t^{n+1}, U^{n+1}) = -c_j^{n+1} \frac{U_j^{n+1} - U_{j-1}^{n+1}}{\Delta x}$$

Hyperbolic PDEs: Method of Lines

Or we could use a "black box" ODE solver, such as `ode45`, to solve the system of ODEs

This "black box" approach is called the method of lines

The name "lines" is because we solve each $U_j(t)$ for a fixed x_j, i.e. a line in the xt-plane

With method of lines we let the ODE solver to choose step sizes Δt to obtain a stable and accurate scheme

Hyperbolic PDEs

Matlab example: $u_t + c(t,x)u_x = 0$

We compare:

1. upwind scheme
2. central difference scheme
3. method of lines

The Wave Equation

We now briefly return to the PDE that motivated this section, wave equation:

$$u_{tt} - c^2 u_{xx} = 0$$

In one spatial dimension, this models, say, vibrations in a taut string

We can solve this by solving the coupled system of two first order advection PDEs introduced earlier

(This coupled system also gives us the characteristic curves for the wave equation, though we won't consider this here)

The Wave Equation

Alternatively, we can directly discretize the second order PDE

For example, we could use central difference approximations for both u_{tt} and u_{xx}:

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

Key points:

- Truncation error analysis $\implies$ second-order accurate
- Fourier stability analysis $\implies$ stable for $0 \leq c\Delta t/\Delta x \leq 1$
- Two-step method in time, need a one-step method to "get started"

Parabolic PDEs

The Heat Equation

The canonical parabolic equation is the heat equation

$$u_t - \alpha u_{xx} = f(t, x),$$

where α models thermal diffusivity

In this section, we shall assume for convenience that $\alpha = 1$

Note that this is an Initial Boundary Value Problem (IBVP):

- We impose an initial condition $u(0, x) = u_0(x)$
- We impose boundary conditions on both sides of the domain

The Heat Equation

A natural idea would be to discretize u_{xx} with a central difference, and employ the forward Euler method in time:

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

Or we could use backward Euler in time:

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

The Heat Equation

Or we could do something "halfway in between":

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

This is called the Crank-Nicolson method (from a paper by Crank and Nicolson in 1947)

The Heat Equation

In fact, it is common to consider a 1-parameter "family" of methods that include all of the above: the θ-method

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} - \theta \frac{U_{j-1}^{n+1} - 2U_j^{n+1} + U_{j+1}^{n+1}}{\Delta x^2} - (1-\theta)\frac{U_{j-1}^n - 2U_j^n + U_{j+1}^n}{\Delta x^2} = 0$$

where $\theta \in [0, 1]$

The Heat Equation

With the θ-method:

- $\theta = 0 \implies$ Euler
- $\theta = \frac{1}{2} \implies$ Crank-Nicolson
- $\theta = 1 \implies$ backward Euler

For the θ-method, we can

1. perform Fourier stability analysis
2. calculate the truncation error

The θ-Method: Stability

Fourier stability analysis: Set $U_j^n(k) = \lambda(k)^n e^{ik(j\Delta x)}$ to get

$$\lambda(k) = \frac{1 - 4(1-\theta)\mu \sin^2\left(\frac{1}{2}k\Delta x\right)}{1 + 4\theta\mu \sin^2\left(\frac{1}{2}k\Delta x\right)}$$

where $\mu \equiv \Delta t/(\Delta x)^2$

We can see by inspection that $\lambda(k) \leq 1$

Hence to ensure stability, we just have to ensure that $\lambda(k) \geq -1$, or equivalently:

$$1 - 4(1-\theta)\mu \sin^2\left(\frac{1}{2}k\Delta x\right) \geq -\left[1 + 4\theta\mu \sin^2\left(\frac{1}{2}k\Delta x\right)\right]$$

The θ-Method: Stability

Rearranging this inequality gives:

$$4\mu(1-2\theta)\sin^2\left(\frac{1}{2}k\Delta x\right) \leq 2$$

For $\theta \in [0.5, 1]$ this inequality is always satisfied, hence the θ-method is unconditionally stable

Unconditional stability: Method is stable for any μ, hence no restriction on Δt, Δx for stability

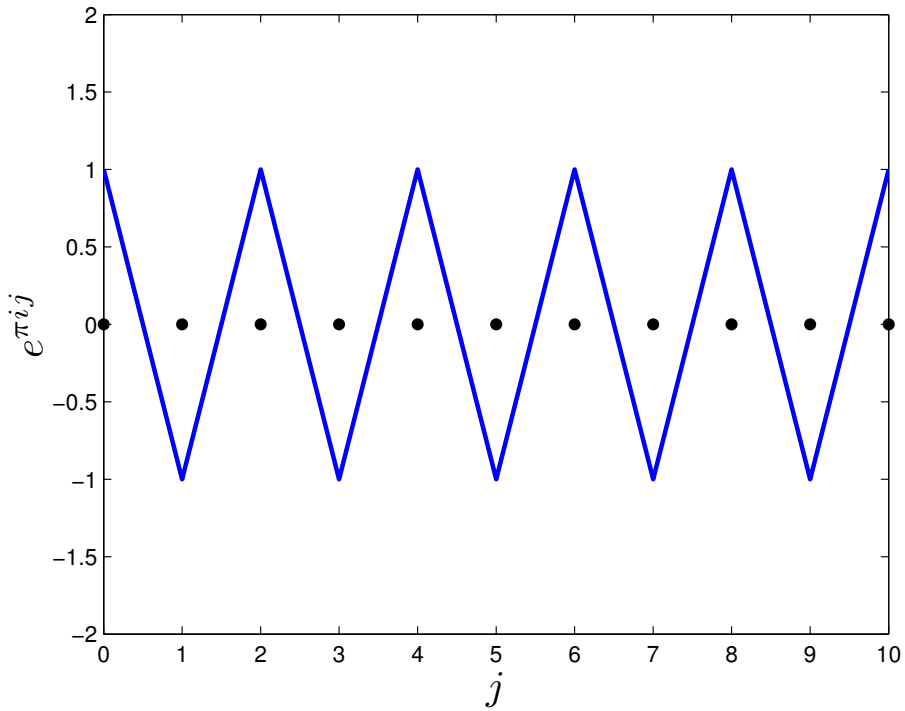
In the $\theta \in [0, 0.5)$ case, the "most unstable" Fourier mode is when $k = \pi/\Delta x$, since this maximizes the factor $\sin^2\left(\frac{1}{2}k\Delta x\right)$

The θ-Method: Stability

Note that this corresponds to the highest frequency mode that can be represented on our grid, since with $k = \pi/\Delta x$ we have

$$e^{ik(j\Delta x)} = e^{\pi ij} = (e^{\pi i})^j = (-1)^j$$

The $k = \pi/\Delta x$ mode:



The θ-Method: Stability

This "sawtooth" mode is stable (and hence all modes are stable) if

$$4\mu(1-2\theta) \leq 2 \iff \mu \leq \frac{1}{2(1-2\theta)},$$

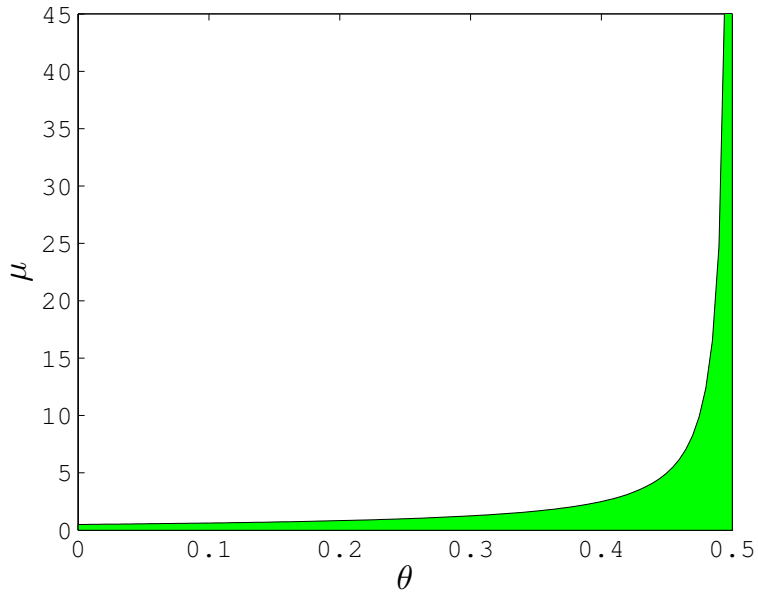
Hence for $\theta \in [0, 0.5)$, the θ-method is only conditionally stable since we require $\Delta t \leq \frac{(\Delta x)^2}{2(1-2\theta)}$ for stability

Note that this is a much tighter time-step restriction than in the hyperbolic case, where we had $\Delta t \leq \frac{\Delta x}{c}$

This is an indication that the system of ODEs that arise from spatially discretizing the heat equation is stiff

The θ-Method: Stability

It's interesting to note that the θ-method "smoothly transitions" to unconditional stability as $\theta \to 1/2$



For $\theta \in [0, 0.5)$, θ-method is stable if μ is in the "green region"

Hence if $\theta = 1/2 - \epsilon$ (e.g. due to rounding error) then we require $\mu \leq 1/(4\epsilon)$, which is unconditional stability for practical purposes

The θ-Method: Accuracy

The truncation error analysis for the θ-method is fairly involved, hence we just give the result:

$$\begin{aligned} T_j^n &\equiv \frac{u_j^{n+1} - u_j^n}{\Delta t} - \theta \frac{u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}}{\Delta x^2} - (1-\theta)\frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2} \\ &= [u_t - u_{xx}] + \left[\left(\frac{1}{2} - \theta\right)\Delta t u_{xxt} - \frac{1}{12}(\Delta x)^2 u_{xxxx}\right] \\ &\quad + \left[\frac{1}{24}(\Delta t)^2 u_{ttt} - \frac{1}{8}(\Delta t)^2 u_{xxtt}\right] \\ &\quad + \left[\frac{1}{12}\left(\frac{1}{2} - \theta\right)\Delta t(\Delta x)^2 u_{xxxxt} - \frac{2}{6!}(\Delta x)^4 u_{xxxxxx}\right] + \cdots \end{aligned}$$

The term $u_t - u_{xx}$ in T_j^n vanishes since u solves the PDE

The θ-Method: Accuracy

Key point: This is a first order method, unless $\theta = 1/2$, in which case we get a second order method!

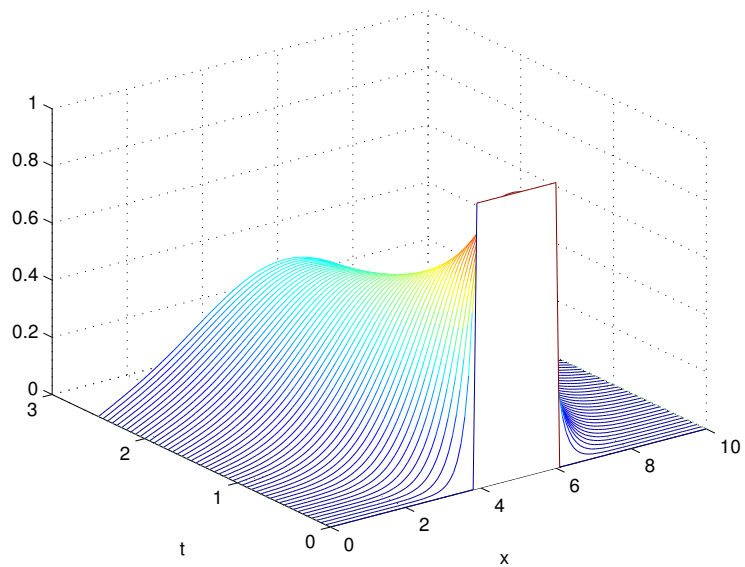
θ-method gives us consistency (since order of accuracy is ≥ 1) and stability (assuming Δt is chosen appropriately when $\theta \in [0, 1/2)$)

Hence, from Lax Equivalence Theorem, the method is convergent

The Heat Equation

Note that the heat equation models a diffusive process, hence it tends to smooth out discontinuities

Matlab demo: Heat equation with discontinous initial condition



This is very different to hyperbolic equations, e.g. the advection equation will just transport a discontinuity in u_0

Elliptic PDEs

Elliptic PDEs

The canonical elliptic PDE is the Poisson equation

In one-dimension, for $x \in [a, b]$, this is $u''(x) = f(x)$ with boundary conditions at $x = a$ and $x = b$

We have seen this problem already: Two-point boundary value problem!

(Recall that Elliptic PDEs model steady-state behavior, there is no time-derivative)

Elliptic PDEs

In order to make this into a PDE, we need to consider more than one spatial dimension

Let $\Omega \subset \mathbb{R}^2$ denote our domain, then the Poisson equation for $(x, y) \in \Omega$ is

$$u_{xx} + u_{yy} = f(x, y)$$

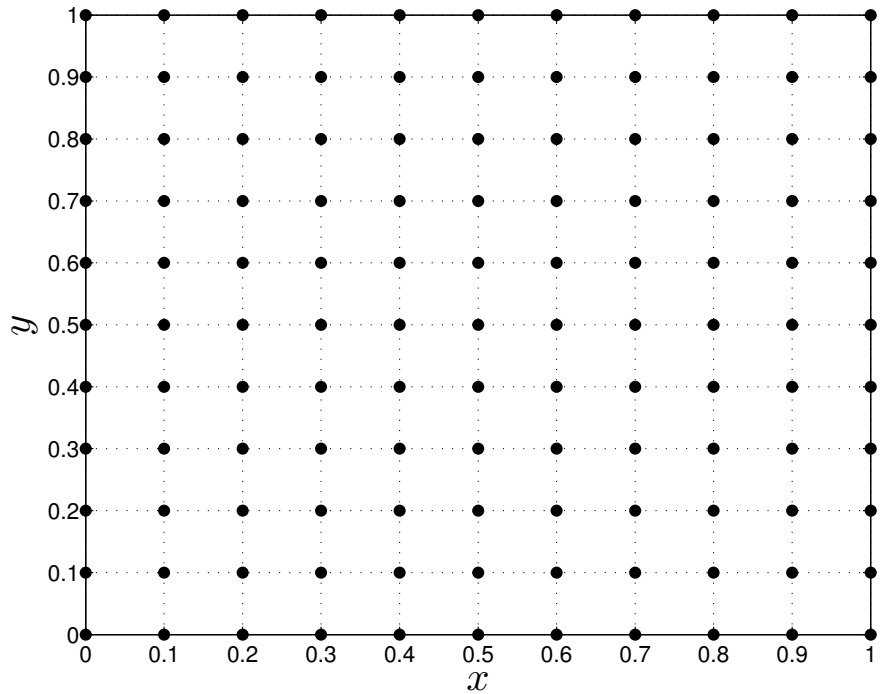
This is generally written more succinctly as $\Delta u = f$

We again need to impose boundary conditions (Dirichlet, Neumann, or Robin) on $\partial\Omega$ (recall $\partial\Omega$ denotes boundary of Ω)

Elliptic PDEs

We will consider how to use a finite difference scheme to approximate this 2D Poisson equation

First, we introduce a uniform grid to discretize Ω



Elliptic PDEs

Let $h = \Delta x = \Delta y$ denote the grid spacing

Then,

- $x_i = (i-1)h$, $i = 1, 2 \ldots, n_x$,
- $y_j = (j-1)h$, $j = 1, 2, \ldots, n_y$,
- $U_{i,j} \approx u(x_i, y_j)$

Then, we need to be able to approximate u_{xx} and u_{yy} on this grid

Natural idea: Use central difference approximation!

Elliptic PDEs

We have

$$u_{xx}(x_i, y_j) = \frac{u(x_{i-1}, y_j) - 2u(x_i, y_j) + u(x_{i+1}, y_j)}{h^2} + O(h^2),$$

and

$$u_{yy}(x_i, y_j) = \frac{u(x_i, y_{j-1}) - 2u(x_i, y_j) + u(x_i, y_{j+1})}{h^2} + O(h^2),$$

so that

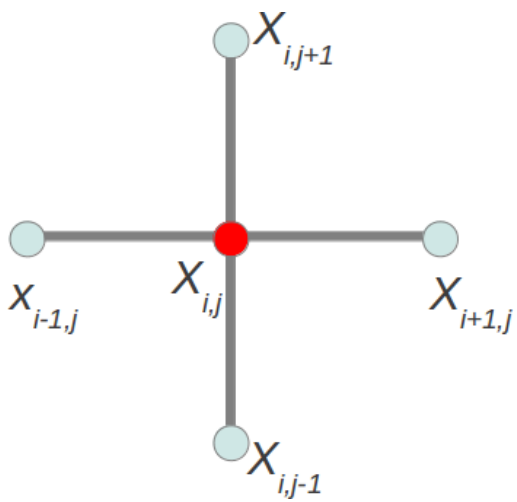
$$u_{xx}(x_i, y_j) + u_{yy}(x_i, y_j) = \frac{u(x_i, y_{j-1}) + u(x_{i-1}, y_j) - 4u(x_i, y_j) + u(x_{i+1}, y_j) + u(x_i, y_{j+1})}{h^2} + O(h^2)$$

Elliptic PDEs

Hence we define our approximation to the Laplacian as

$$\frac{U_{i,j-1} + U_{i-1,j} - 4U_{i,j} + U_{i+1,j} + U_{i,j+1}}{h^2}$$

This corresponds to a "5-point stencil"



Elliptic PDEs

As usual, we represent the numerical solution as a vector $\mathbb{U} \in \mathbb{R}^{n_x n_y}$

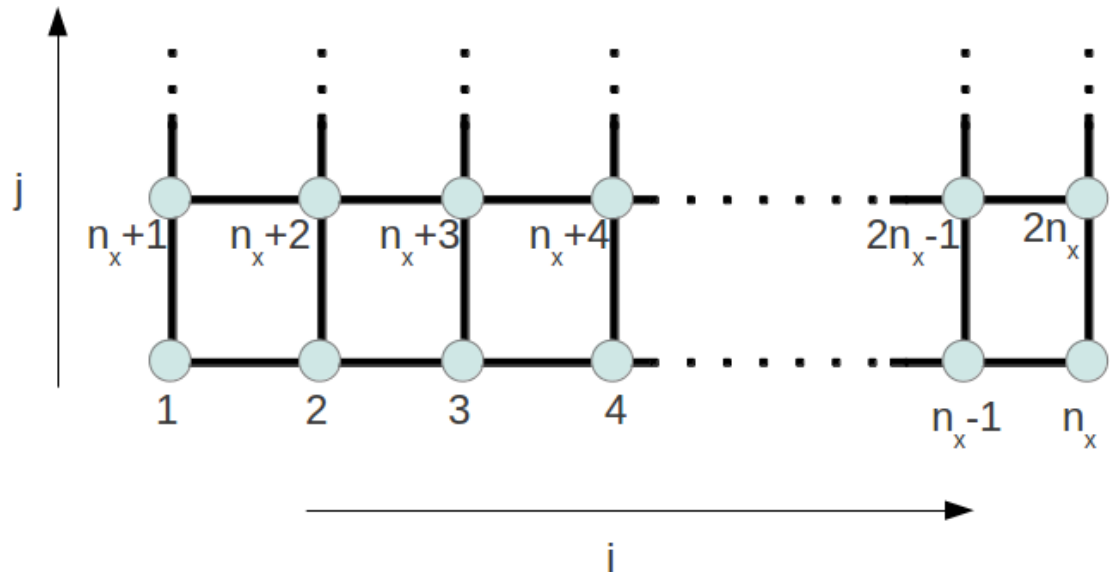
We want to construct a differentiation matrix $D_2 \in \mathbb{R}^{n_x n_y \times n_x n_y}$ that approximates the Laplacian

Question: How many non-zero diagonals will D_2 have?

To construct D_2, we need to be able to relate the entries of the vector $\mathbb{U}$ to the "2D grid-based values" $U_{i,j}$

Elliptic PDEs

Hence we need to number the nodes from 1 to $n_x n_y$ — we number nodes along the "bottom row" first, then second bottom row, etc



Let $\mathcal{G}$ denote the mapping from the "2D indexing" to the "1D indexing," from the above figure we have:

$$\mathcal{G}(i,j;n_x) = (j-1)n_x + i, \quad \text{and hence} \quad \mathbb{U}_{\mathcal{G}(i,j;n_x)} = U_{i,j}$$

Elliptic PDEs

Let us focus on node (i, j) in our F.D. grid, this corresponds to entry $\mathcal{G}(i, j; n_x)$ of $\mathbb{U}$

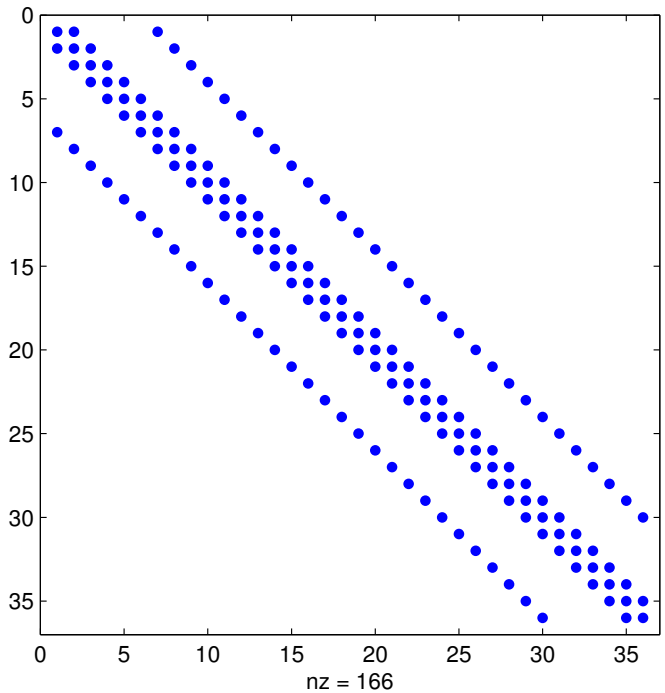
Due to the 5-point stencil, row $\mathcal{G}(i, j; n_x)$ of D_2 will only have non-zeros in columns

$$\mathcal{G}(i, j-1; n_x) \quad [\text{or equivalently } \mathcal{G}(i, j; n_x) - n_x], \tag{1}$$
$$\mathcal{G}(i-1, j; n_x) \quad [\text{or equivalently } \mathcal{G}(i, j; n_x) - 1], \tag{2}$$
$$\mathcal{G}(i, j; n_x), \tag{3}$$
$$\mathcal{G}(i+1, j; n_x) \quad [\text{or equivalently } \mathcal{G}(i, j; n_x) + 1], \tag{4}$$
$$\mathcal{G}(i, j+1; n_x) \quad [\text{or equivalently } \mathcal{G}(i, j; n_x) + n_x] \tag{5}$$

- (2), (3), (4), give the same tridiagonal structure that we're used to from differentiation matrices in 1D domains
- (1), (5) give diagonals shifted by $\pm n_x$

Elliptic PDEs

For example, sparsity pattern of D_2 when $n_x = n_y = 6$



Elliptic PDEs

Matlab demo: Solve the Poisson equation

$$\Delta u = -\exp\left\{-(x-0.25)^2 - (y-0.5)^2\right\},$$

for $(x, y) \in \Omega = [0,1]^2$ with $u = 0$ on $\partial\Omega$