Solving PDE's with FEniCS

Heat, Wave equations and finite differences

Chapters 10–12

Introduction to

Automated Modeling

with FEniCS

by L. Ridgway Scott

Heat transfer

Heat can be exchanged between two different bodies by diffusion, convection or radiation.

The **heat equation** describes diffusion of thermal energy in a medium [7].

The simplest form of the heat equation takes the form

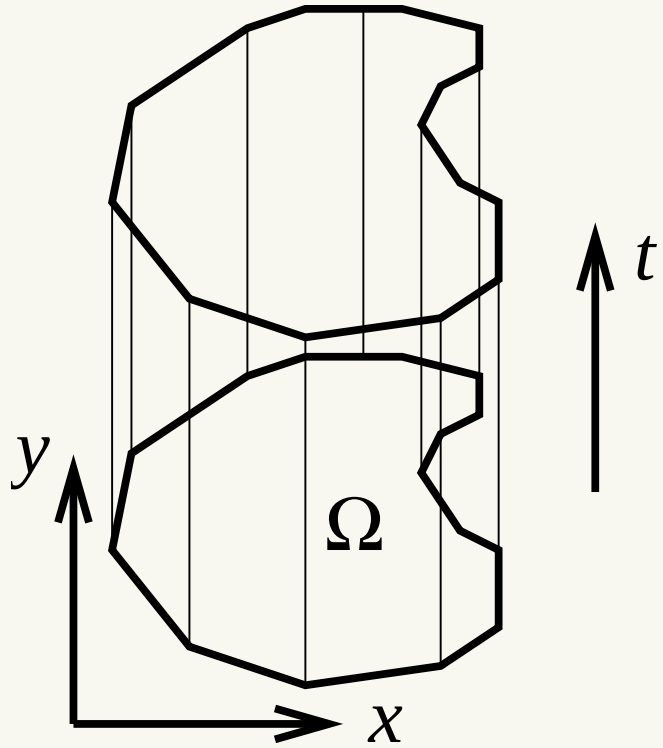
$$\begin{aligned} \frac{\partial u}{\partial \tau}(\xi,\tau) - c\Delta u(\xi,\tau) &= f(\xi,\tau) \quad \forall \xi \in \widetilde{\Omega},\ \tau > 0 \\ u(\xi,0) &= u_0(\xi) \quad \forall \xi \in \widetilde{\Omega}, \end{aligned} \tag{1}$$

together with boundary conditions imposed on $\partial\widetilde{\Omega}$ that will be discussed subsequently.

Cylinder domain

Have distinguished variable τ that we think of as “time” [8] together with the spatial variable ξ.

The natural domain for the problem is a cylinder domain $\Omega \times [0, T]$ as indicated in the figure:



Diffusion coefficient

The diffusion coefficient, c, takes care of the mismatch in units; c has units of length-squared divided by time.

The heat equation is more generally referred to as the **diffusion equation**, and it governs the diffusion of many materials.

Some examples of diffusion coefficients are given in Table 1.

We see that these coefficients can differ by orders of magnitude.

Diffusion coefficients

material	diffusion coefficient	medium	conditions
H_2	1.6 cm^2/sec	self	T = 273 K, p = 0.1 MPa
CO_2	.106 cm^2/sec	self	T = 273 K, p = 0.1 MPa
CO_2	1.92×10^{-5} cm^2/sec	water	T=298 K
sucrose	0.52×10^{-5} cm^2/sec	water	T=25 C
hydrogen	1.66×10^{-9} cm^2/sec	iron	T=10 C
hydrogen	1.24×10^{-7} cm^2/sec	iron	T=100 C
aluminum	1.3×10^{-30} cm^2/sec	copper	T=20 C

Table 1: Diffusion coefficients for various materials in various media. When the medium is “self” the coefficient is the self-diffusion constant.

Nondimensionalization

It is frequently useful to remove the units in a PDE, that is, to **nondimensionalize** the equation.

For the heat/diffusion equation, this allows us to develop some useful intuition that is independent of the domain of application.

We can do this by changing the spatial and time variables:

$$x = a\xi, \quad t = b\tau, \quad a \neq 0, \quad b \neq 0.$$

Change of variables

With this change of variables, the first equation in (1) becomes

$$b\frac{\partial u}{\partial t}(x,t) - ca^2\Delta_x u(x,t) = f(x,t) \quad \forall x \in \Omega,\ t > 0, \quad (2)$$

where $\Omega = \left\{x \ : \ a^{-1}x \in \widetilde{\Omega}\right\}$ and

we added the subscript x to Δ to clarify that it is

$$\Delta_x = \partial^2/\partial x_1^2 + \cdots + \partial^2/\partial x_d^2$$

as opposed to the meaning in the first equation in (1).

New coefficients

Defining $\hat{f}(x,t) = b^{-1} f(x/a, t/b)$, we find

$$\frac{\partial u}{\partial t}(x,t) - \frac{ca^2}{b}\Delta_x u(x,t) = \hat{f}(x,t) \quad \forall x \in \widetilde{\Omega},\ t > 0. \tag{3}$$

We then have wide lattitude to choose the time and/or space coordinates so that

$$ca^2 = b.$$

In such coordinates, the diffusion equation simplifies to

$$\frac{\partial u}{\partial t}(x,t) - \Delta_x u(x,t) = \hat{f}(x,t) \quad \forall x \in \Omega,\ t > 0. \tag{4}$$

We now explore this in detail in one space dimension.

One space dimension

In its simplest, one-dimensional form, it may be written

$$\begin{aligned} \frac{\partial u}{\partial t}(x,t) - \frac{\partial^2 u}{\partial x^2}(x,t) &= f(x,t) \quad \forall x \in [0,1],\ t > 0 \\ u(x,0) &= u_0(x) \quad \forall x \in [0,1] \end{aligned} \tag{5}$$

where $u(x,t)$ denotes temperature of medium at point x and time t.

Simple example: transfer of heat across a window.

The variable x denotes the distance from one face of the window pane to the other, in the direction perpendicular to the plane of the window.

3D effects

Near outer edges of the window, three dimensional effects would be evident.

But in the middle of the window, equation (5) would accurately describe the evolution of the temperature u inside the window.

The function f is included for completeness, but in many cases such a body source of heat would be zero.

These equations must be supplemented by boundary contitions similar to the ones considered previously.

Boundary conditions

They could be of purely Dirichlet (or essential) type, viz.

$$u(0,t) = g_0(t), \quad u(1,t) = g_1(t) \quad \forall t > 0, \tag{6}$$

or of purely Neumann (or natural) type, viz.

$$\frac{\partial u}{\partial x}(0,t) = g_0(t), \quad \frac{\partial u}{\partial x}(1,t) = g_1(t) \quad \forall t > 0, \tag{7}$$

Or a combination of the two:

$$u(0,t) = g_0(t), \quad \frac{\partial u}{\partial x}(1,t) = g_1(t) \quad \forall t > 0. \tag{8}$$

Here g_i, $i = 0, 1$, are given functions of t.

The main characteristic of the heat equation is that it smooths any roughness in the initial data.

For example, in Figure 1 we show the solution at time $t = 0.001$ for the case

$$u_0(x) = \tfrac{1}{2} - |x - \tfrac{1}{2}|. \tag{9}$$

We see that the discontinuity of the derivative of u_0 at $x = \frac{1}{2}$ is instantly smoothed.

This has a corollary for the backwards heat equation that we will explore subsequently.

Basic behavior

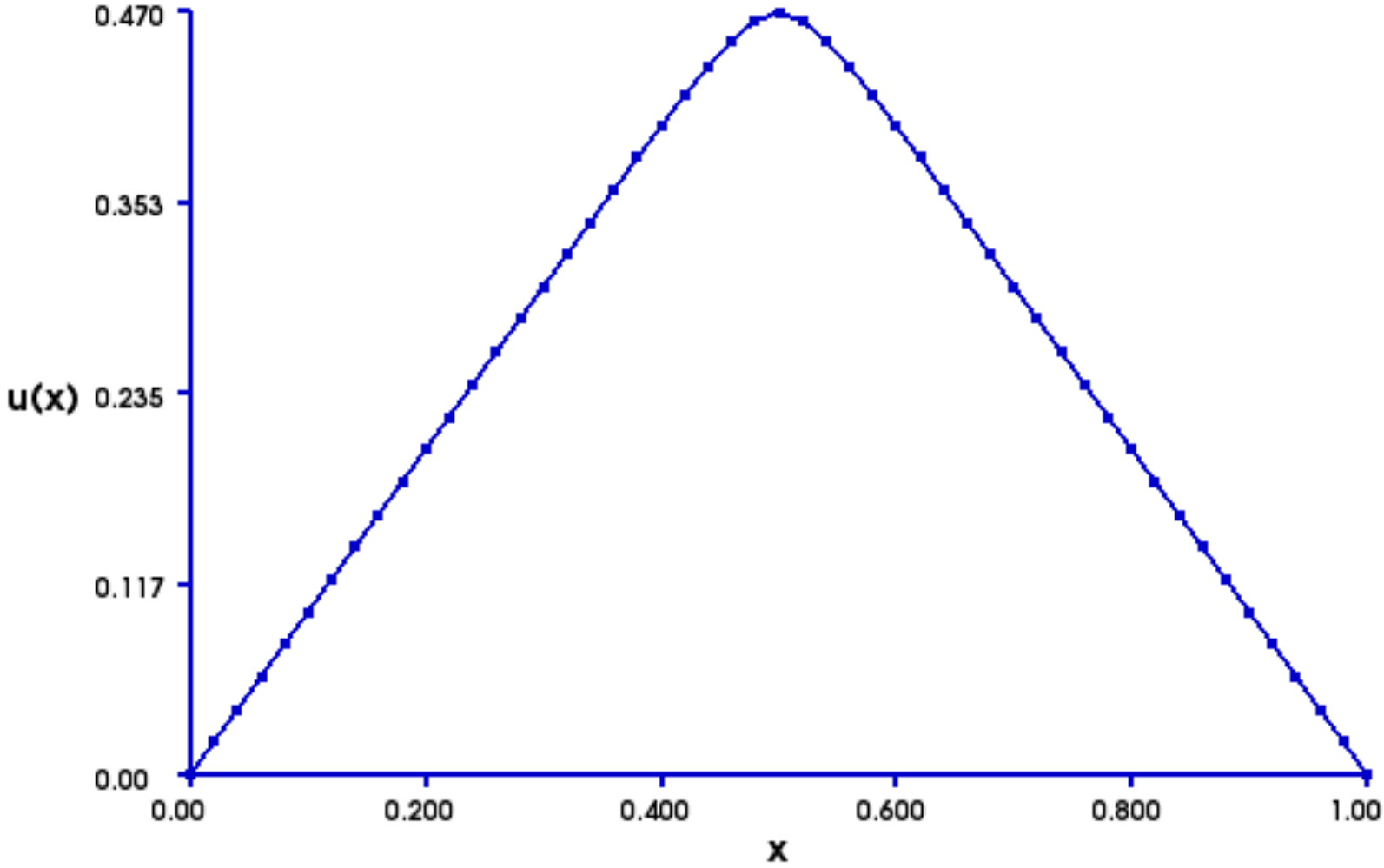


Figure 1: Solution of (5) with initial data (9) at time $t = 0.001$. Computed with piecewise linears with 50 mesh points (uniform mesh).

Compatibility conditions

One type of nonsmooth behavior stems from a mismatch between boundary data and initial data.

This is governed by compatibility conditions.

It is interesting to note that the pure Neumann condition (7) for the heat equation (5) does not suffer the same limitations on the data, or nonuniqueness of solutions, that the steady state counterpart does.

However, there are compatibility conditions required to obtain smooth solutions, linking the boundary and initial data for the heat equation in order to have a smooth solution.

Compatibility conditions

Compatibility conditions derived from observation that the values of u on the spatial boundary have been specified twice at $t = 0$.

Consider the case (8) of combined Dirichlet and Neumann boundary conditions.

The first set of compatibility conditions is

$$u_0(0) = u(0,0) = g_0(0) \text{ and } u_0'(1) = u_x(1,0) = g_1(0). \tag{10}$$

These are obtained by matching the two ways of specifying the solution at the boundary points $(x,t) = (0,0)$ and $(x,t) = (1,0)$.

Neumann conditions

In the case of pure Dirichlet conditions (6) the compatibility conditions become

$$u_0(0) = u(0,0) = g_0(0) \text{ and } u_0(1) = u(1,0) = g_1(0). \quad (11)$$

In the case of pure Neumann conditions (7) the compatibility conditions become

$$u_0'(0) = u_x(0,0) = g_0(0) \text{ and } u_0'(1) = u_x(1,0) = g_1(0). \quad (12)$$

Conditions involving derivatives (Neumann boundary conditions) are higher-order than those for function values (Dirichlet boundary conditions).

They affect bounds of higher-order derivatives.

Incompatible heat

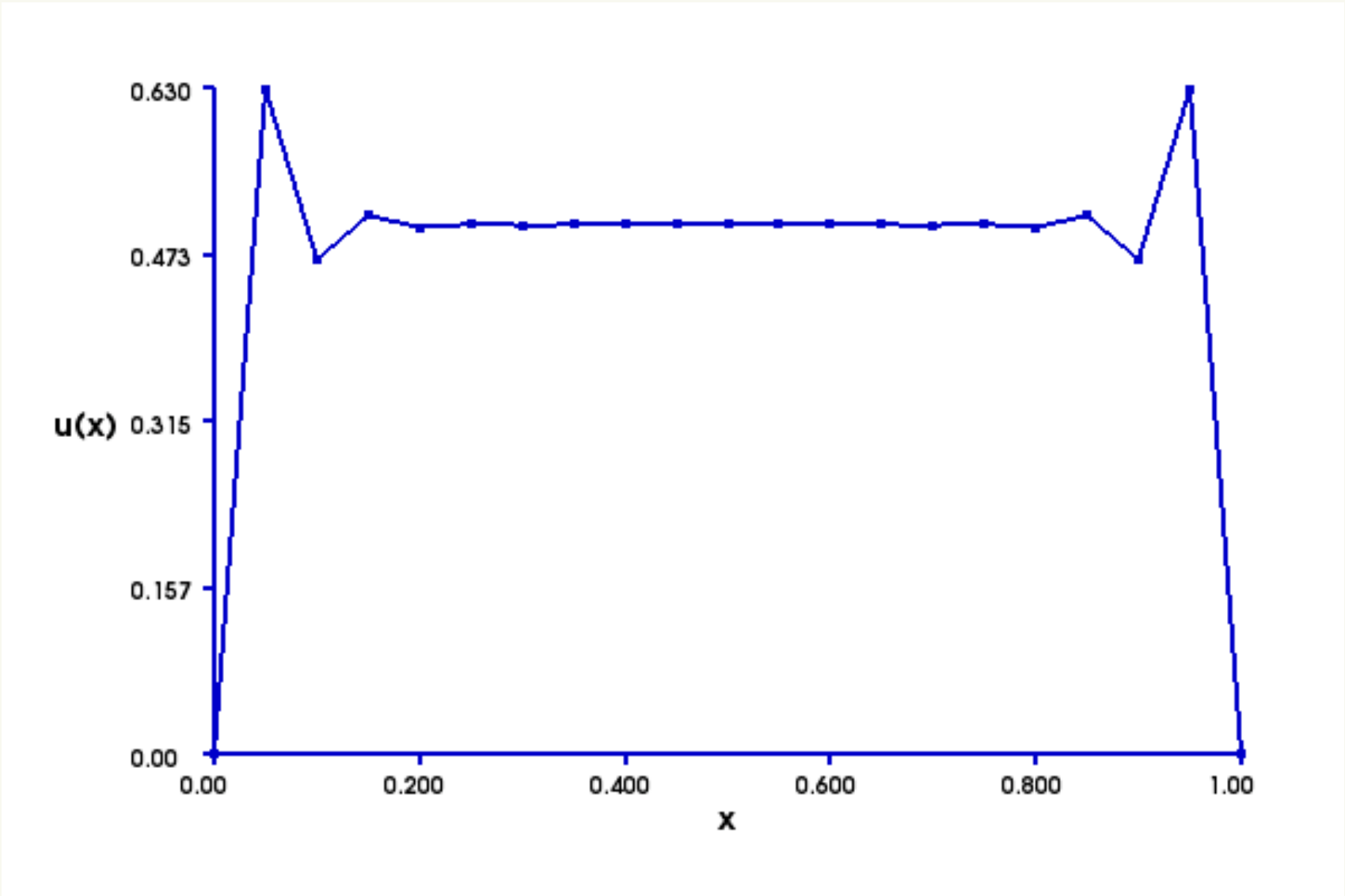


Figure 2: Heat equation with incompatible data after one time step with $\Delta t = 10^{-5}$; degree = 1, 20 mesh intervals. Initial values $u_0 = 0.5$.

Incompatible heat

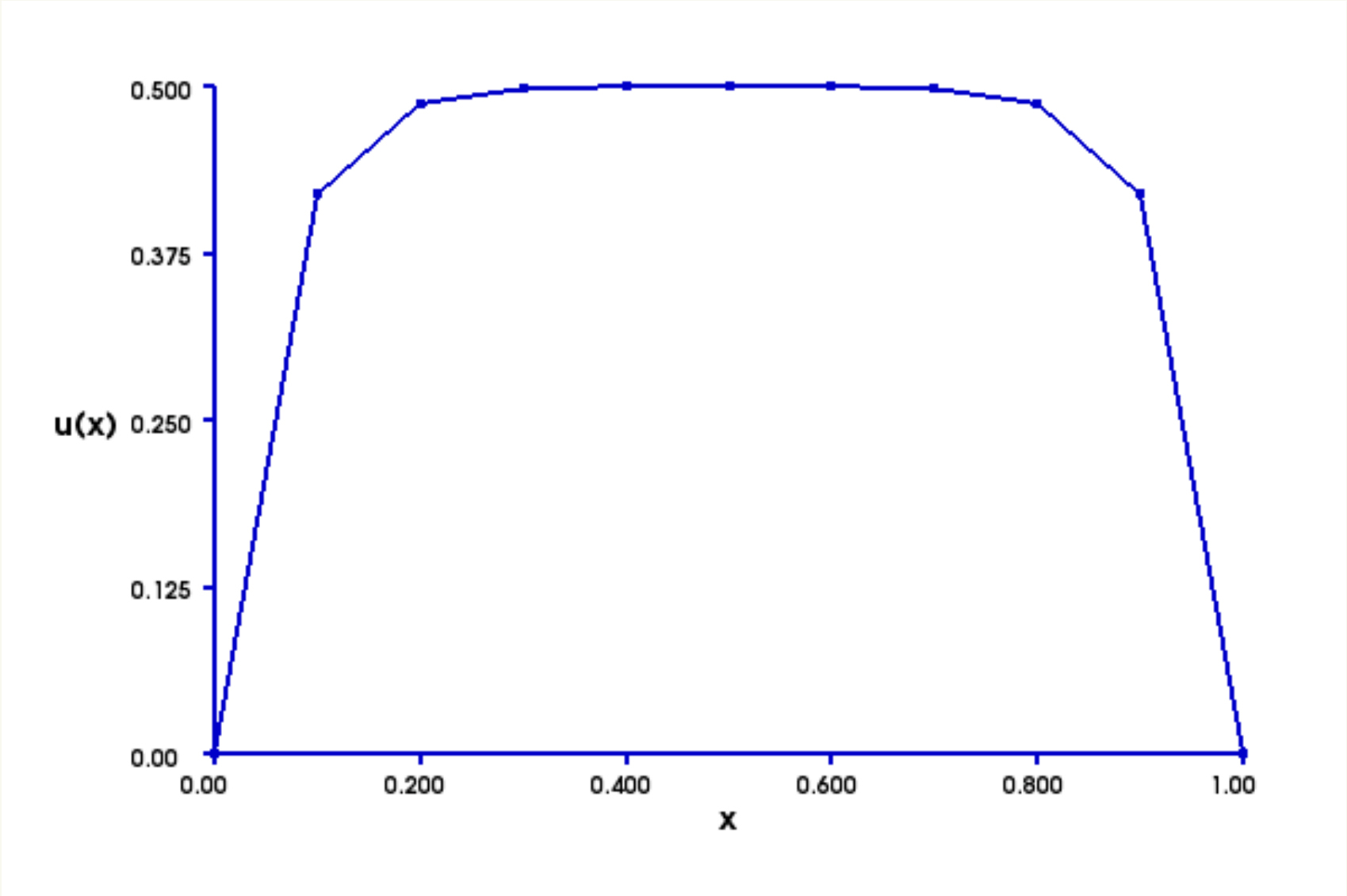


Figure 3: Heat equation with incompatible data after one time step with $\Delta t = 10^{-5}$; degree = 2, 10 mesh intervals. Initial values $u_0 = 0.5$.

Higher order conditions

For an abritrary order of smoothness, there are infinitely many compatibility conditions.

Second set of conditions uses differential equation $u_{xx} = u_t$ to trade spatial derivatives for temporal ones, then applying this at $t = 0$ and $x = 0$:

Dirichlet case:

$$u_0''(0) = u_{xx}(0,0) = u_t(0,0) = g_0'(0).$$

Neumann case:

$$u_0'''(1) = u_{xxx}(1,0) = u_{xt}(1,0) = g_1'(0).$$

Compatibility implications

If compatibilities are not satisfied by data, oscillations will result near $t = 0$ and $x = 0, 1$.

In nonlinear problems, this can cause completely wrong results to occur.

Compatibility conditions do not have to be satisfied for heat equation (5) to be well posed.

There is a unique solution in any case, but the physical model may be incorrect as a result if it is supposed to have a smooth solution.

Compatibility Conditions

Compatibility conditions are a subtle form of constraint on model quality.

In many problems they can be described in terms of local differential-algebraic constraints.

However, such compatibility conditions for the Navier-Stokes equations can lead to global constraints that are hard to verify or satisfy in practice.

Variational form of the heat equation

It is possible to derive a vatiational formulation involving integration over both x and t, but it is more common to use a variational formulation based on x alone.

We seek a function $\tilde{u}(t)$ of time with values in V such that $\tilde{u}(0) = u_0$

$$(\tilde{u}'(t), v)_{L^2(\Omega)} + a(\tilde{u}(t), v) = F(v) \quad \forall v \in V,\ t \geq 0, \quad (13)$$

where $\Omega = [0, 1]$ and $a(w, v) = \int_0^1 w'(x)v'(x)\,dx$.

Since it is a bit awkward to work with a function of one variable (t) which is a function of another (x), we often write (13) in terms of $u(x, t) = \tilde{u}(t)(x)$.

Variational form of the heat equation

Using subscript notation for partial derivatives, it becomes

$$(u_t(\cdot,t),v)_{L^2(\Omega)} + a(u(\cdot,t),v) = F(v) \quad \forall v \in V. \tag{14}$$

for all t. If we remember the dependence on t, we can write this as

$$(u_t,v)_{L^2(\Omega)} + a(u,v) = F(v) \quad \forall v \in V. \tag{15}$$

A stability estimate follows immediately from the variational formulation.

For simplicity, suppose that the right-hand-side form $F \equiv 0$ and that the boundary data vanishes as well (i.e., only the initial data is non-zero).

Variational form of the heat equation

Using $v = u$ (at fixed t, i.e., $v = u(\cdot, t)$) in (15), we find

$$\frac{1}{2}\frac{\partial}{\partial t}\|u\|^2_{L^2(\Omega)} = (u_t, u)_{L^2(\Omega)} = -a(u, u) \leq 0 \quad \forall t \geq 0, \quad (16)$$

where Ω denotes the spatial interval $[0, 1]$. From this, it follows by integrating in time that

$$\|u(\cdot, t)\|_{L^2(\Omega)} \leq \|u(\cdot, 0)\|_{L^2(\Omega)} = \|u_0\|_{L^2(\Omega)} \quad \forall t \geq 0. \quad (17)$$

This result is independent of any compatibility conditions.

However, all it says is that the mean-square of the temperature u remains bounded by its initial value.

If F is nonzero but bounded on V, i.e.,

$$|F(v)| \leq \|F\|_{H^{-1}(\Omega)} \|v\|_{H^1(\Omega)} \quad \forall v \in V, \tag{18}$$

then we retain a bound on $\|u(\cdot, t)\|_{L^2(\Omega)}$:

$$\begin{aligned} \frac{1}{2}\frac{\partial}{\partial t}\|u\|^2_{L^2(\Omega)} = (u_t, u)_{L^2(\Omega)} &= F(u) - a(u,u) \\ &\leq \|F\|_{H^{-1}(\Omega)} \|u\|_{H^1(\Omega)} - a(u,u). \end{aligned} \tag{19}$$

The form $a(\cdot, \cdot)$ always satisfies at least a weak type of coercivity of the form

$$\|v\|^2_{H^1(\Omega)} \leq \gamma_1 a(v,v) + \gamma_2 \|v\|^2_{L^2(\Omega)} \quad \forall v \in V, \tag{20}$$

known as **Gårding's inequality**.

For example, this holds for the pure Neumann problem (7) with $V = H^1(\Omega)$ whereas the stronger form of coercivity we saw earlier does not in this case.

Applying (20) in (19) gives

$$\frac{\partial}{\partial t}\|u\|^2_{L^2(\Omega)} \leq 2\|F\|_{H^{-1}(\Omega)}\|u\|_{H^1(\Omega)} - \frac{2}{\gamma_1}\|u\|^2_{H^1(\Omega)} + \frac{2\gamma_2}{\gamma_1}\|u\|^2_{L^2(\Omega)}. \tag{21}$$

The arithmetic-geometric mean inequality

$$2rs \leq \delta r^2 + \frac{1}{\delta}s^2 \tag{22}$$

holds for any $\delta > 0$ and any real numbers r and s.

Using the arithmetic-geometric mean inequality we find

$$2\|F\|_{H^{-1}(\Omega)}\|u\|_{H^1(\Omega)} \leq \frac{\gamma_1}{2}\|F\|^2_{H^{-1}(\Omega)} + \frac{2}{\gamma_1}\|u\|^2_{H^1(\Omega)}.$$

Thus

$$\frac{\partial}{\partial t}\|u\|^2_{L^2(\Omega)} \leq \frac{\gamma_1}{2}\|F\|^2_{H^{-1}(\Omega)} + \frac{2\gamma_2}{\gamma_1}\|u\|^2_{L^2(\Omega)} \tag{23}$$

Gronwall's Lemma [10] implies

$$\|u(\cdot,t)\|_{L^2(\Omega)} \leq \|u_0\|_{L^2(\Omega)} + e^{t(\gamma_2/\gamma_1)}\|F\|_{H^{-1}(\Omega)} \quad \forall t \geq 0. \tag{24}$$

Stability of gradient

Another stability result can be derived by using $v = u_t$ (assuming $F \equiv 0$ and the boundary data are zero) in (15), to find

$$\|u_t\|^2_{L^2(\Omega)} = -a(u, u_t) = -\frac{1}{2}\frac{\partial}{\partial t}a(u, u). \tag{25}$$

From (25), it follows that

$$\frac{\partial}{\partial t}a(u, u) = -2\|u_t\|^2_{L^2(\Omega)} \leq 0 \quad \forall t \geq 0. \tag{26}$$

Again integrating in time and using (17), we see that

$$\|u(\cdot, t)\|_{H^1(\Omega)} \leq \|u(\cdot, 0)\|_{H^1(\Omega)} = \|u_0\|_{H^1(\Omega)} \quad \forall t \geq 0. \tag{27}$$

Variational form of the heat equation

This result requires first-order compatibility conditions.

Presupposes that $u_0 \in V$, and this may not hold.

Says mean-square of gradient of temperature u also remains bounded by its initial value.

Moreover, if the data F is not zero, this result will not hold.

In particular, if the compatibility condition (10) does not hold, then $u_0 \notin V$ and $\|u(\cdot, t)\|_{H^1(\Omega)}$ will not remain bounded as $t \to 0$.

Discretiztion for heat equation

Simplest discretization uses a finite element method for spatial differential equation and a finite difference method for the temporal part.

Allows us to **reuse existing software** already developed for the spatial problem.

Many time dependent problems can be treated in the same manner. This technique goes by many names:

- (time) **splitting** since the time and space parts are separated and treated by independent methods
- the **method of lines** since the problem is solved on a sequence of lines (copies of the spatial domain), one for each time step.

Explicit Euler Time Discretization

The simplest time discretization method for the heat equation uses the forward (or explicit) Euler difference method. It takes the form

$$\begin{aligned} u^{n+1}(x) &= u^n(x) + \Delta t \frac{\partial^2 u^n}{\partial x^2}(x) \quad \forall x \in [0,1], \\ u^0(x) &= u_0(x) \quad \forall x \in [0,1] \\ u^n(0) = g_0(n\Delta t) \quad &\text{and} \quad u^n(1) = g_1(n\Delta t) \quad \forall n > 0 \end{aligned} \tag{28}$$

where $u^n(x)$ denotes an approximation to $u(x, n\Delta t)$.

Applying the finite difference or finite element approximation to (28) yields a simple algorithm.

For simplicity, we begin with the case that $g_0 = g_1 = 0$.

Implicit Euler Time Discretization

The difficulty with this simple algorithm is that it is *unstable* unless `dt` is sufficiently small.

The simplest implicit time discretization method for the heat equation uses the backward (or implicit) Euler difference method. It takes the form

$$\begin{aligned} u^{n+1}(x) &= u^n(x) + \Delta t \frac{\partial^2 u^{n+1}}{\partial x^2}(x) \quad \forall x \in [0,1], \\ u^0(x) &= u_0(x) \quad \forall x \in [0,1] \\ u^n(0) = g_0(n\Delta t) \quad &\text{and} \quad u^n(1) = g_1(n\Delta t) \quad \forall n > 0 \end{aligned} \tag{29}$$

where $u^n(x)$ again denotes an approximation to $u(x, n\Delta t)$.

Implicit Euler Time Discretization

Applying the finite difference or finite element approximation to (29) yields now a system of equations to be solved at each time step.

For simplicity, we begin with the case that $g_0 = g_1 = 0$.

This algorithm is *stable* for all `dt`, but now we have to solve a system of equations instead of just multiplying by a matrix.

Note however that the system to be solved is just the same as in the ODE boundary value problems studied earlier, so the same family of techniques can be used.

Variational form of the time discretization

Explicit Euler time stepping method is written in variational form as

$$\begin{aligned}(u^{n+1}, v)_{L^2(\Omega)} = (u^n, v)_{L^2(\Omega)} \\ + \Delta t \left(F(v) - a(u^n, v)\right) \quad \forall v \in V.\end{aligned} \tag{30}$$

Solving for u^{n+1} requires inverting the mass matrix.

Implicit Euler time stepping method is written in variational form as

$$\begin{aligned}(u^{n+1}, v)_{L^2(\Omega)} + \Delta t\, a(u^{n+1}, v) = (u^n, v)_{L^2(\Omega)} \\ + \Delta t\, F(v) \quad \forall v \in V.\end{aligned} \tag{31}$$

Variational form of the time discretization

Solving for u^{n+1} requires inverting linear combination of stiffness and the mass matrices.

This is now in the familiar form: find $u^{n+1} \in V$ such that

$$a_{\Delta t}(u^{n+1}, v) = F^n_{\Delta t}(v) \quad \forall v \in V,$$

where

$$\begin{aligned} a_{\Delta t}(v, w) &= \int_\Omega vw + \Delta t \; v'w' \, d\mathbf{x}, \text{ and} \\ F^n_{\Delta t}(v) &= (u^n, v)_{L^2(\Omega)} + \Delta t F(v) \quad \forall v, w \in V. \end{aligned} \tag{32}$$

Backwards differentiation formulæ

A popular way to achieve increased accuracy in time-dependent problems is to use a **backwards differentiation formula** (BDF)

$$\frac{du}{dt}(t_n) \approx \frac{1}{\Delta t} \sum_{i=0}^{k} a_n u_{n-i}, \tag{33}$$

where the coefficients $\{a_i \ : \ i = 0, \ldots k\}$ are given in Table 2.

k	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
1	1	-1						
2	$3/2$	-2	$1/2$					
3	$11/6$	-3	$3/2$	$-1/3$				
4	$25/12$	-4	$6/2$	$-4/3$	$1/4$			
5	$137/60$	-5	$10/2$	$-10/3$	$5/4$	$-1/5$		
6	$49/20$	-6	$15/2$	$-20/3$	$15/4$	$-6/5$	$1/6$	
7	$363/140$	-7	$21/2$	$-35/3$	$35/4$	$-21/5$	$7/6$	$-1/7$

Table 2: Coefficients of the BDF schemes of degree k.

The BDF for $k = 1$ is the same as implicit Euler.

Backwards differentiation formulæ

The BDF formulæ satisfy [9]

$$\sum_{i=0}^{k} a_i u_{n-i} = \sum_{j=1}^{k} \frac{(-1)^j}{j} \Delta^j u_n, \tag{34}$$

where Δu_n is the sequence whose n-th entry is $u_n - u_{n-1}$.

The higher powers are defined by induction: $\Delta^{j+1} u_n = \Delta(\Delta^j u_n)$.

For example, $\Delta^2 u_n = u_n - 2u_{n-1} + u_{n-2}$, and in general

Δ^j has coefficients given from Pascal's triangle.

Backwards differentiation formulæ

We thus see that $a_0 \neq 0$ for all $k \geq 1$; $a_0 = \sum_{i=1}^{k} 1/i$. Similarly, $a_1 = -k$.

For $j \geq 2$, ja_j is an integer conforming to Pascal's triangle.

Given this simple definition of the general case of BDF, it is hard to imagine what could go wrong regarding stability.

Unfortunately, the BDF method of order $k = 7$ is unconditionally unstable and hence cannot be used.

Exercise: explore the use of BDF schemes.

The backwards heat equation

The heat equation is reversible with respect to time, in the sense that if we let time run backwards we get an equation that takes the final values to the initial values.

More precisely, let $u(x,t)$ be the solution to (5) for $0 \leq t \leq T$. Let $v(x,t) := u(x, T-t)$. Then v solves the backwards heat equation

$$\begin{aligned} \frac{\partial v}{\partial t}(x,t) + \frac{\partial^2 v}{\partial x^2}(x,t) = 0 \quad \forall x \in [0,1],\ t > 0 \\ v(x,0) = v_0(x) = u(x,T) \quad \forall x \in [0,1] \\ v(0,t) = g_0(T-t), \quad v(1,t) = g_1(T-t) \quad \forall t > 0 \end{aligned} \tag{35}$$

and $v(x,T)$ will be the same as the initial data u_0 for (5).

With nonsmooth initial data, unreliable results occur.

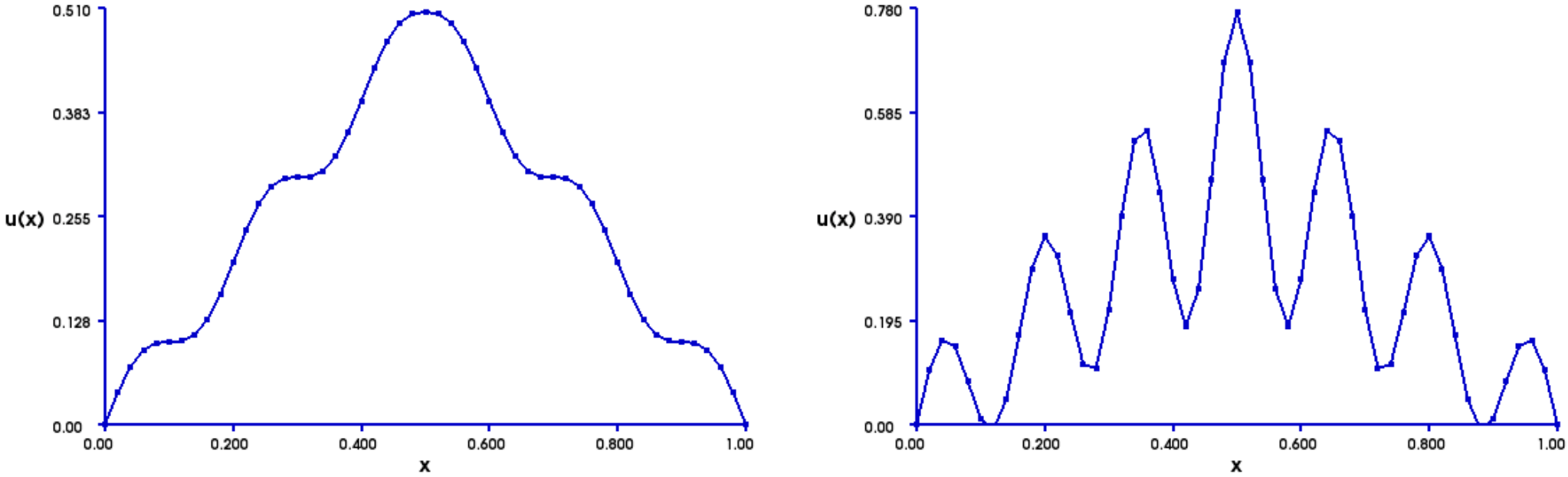


Figure 4: Two solutions of the backwards heat equation with hat-function initial data at time $t = 0.001$, computed with piecewise linears with 50 mesh points (uniform mesh). (left) One time step with $\Delta t = 0.001$. (right) Two time steps with $\Delta t = 0.0005$.

The backwards heat equation

Although (35) has a well-defined solution in many cases, it is not well posed in the usual sense.

It only has a solution starting from solutions of the heat equation.

Moreover, such solutions may exist only for a short time, and then blow up.

Thus great care must be used in attempting to solve the backwards heat equation.

Now we consider briefly

finite difference methods

Allows direct comparison with FEM

Provides some explicit formulas
for heat equation approximation

Consider the two-point boundary-value problem

$$\begin{aligned} -\frac{d^2u}{dx^2} &= f \text{ in } (0,1) \\ u(0) = g_0, \quad u'(1) &= g_1. \end{aligned} \tag{36}$$

FDM: differential operator $\longrightarrow$ difference operator.

In this way we get the approximation for (36)

$$-u(x-h) + 2u(x) - u(x+h) \approx h^2 f(x) \tag{37}$$

where $h > 0$ is the mesh size to be used.

Choosing $x = x_n := nh$ for $n = 0, 1, \ldots, N = 1/h$, we get a system of linear equations

$$-u_{n-1} + 2u_n - u_{n+1} = h^2 f(x_n) \tag{38}$$

where $u_n \approx u(x_n)$.

Same as piecewise linear finite element discretization.

Boundary condition at $x = 0$ translates into $u_0 = g_0$.

Thus equation (38) for $n = 1$ becomes

$$2u_1 - u_2 = h^2 f(x_1) + g_0. \tag{39}$$

But derivative boundary conditions more complex.

Derivative boundary conditions

Derivative boundary condition at $x = 1$ must be approximated by a difference equation.

A natural one to use is

$$u_{N+1} - u_{N-1} = 2hg_1 \tag{40}$$

using a difference over an interval of length $2h$ centered at x_N.

Using (40), equation (38) for $n = N$ becomes

$$-2u_{N-1} + 2u_N = h^2 f(x_N) + 2hg_1. \tag{41}$$

Now we summarize these equations as a matrix equation.

Algebraically, we can express the finite difference method as

$$\mathbf{AU} = \mathbf{F} \tag{42}$$

where

- $\mathbf{U}$ is the vector with entries u_n and
- $\mathbf{F}$ is the vector with entries $h^2 f(x_n)$
- appropriately modified at $n = 1$ and $n = N$ using boundary data.

Finite difference matrix $\mathbf{A}$ has very regular pattern.

- appropriately modified at $n = 1$ and $n = N$

as we now describe.

Finite Difference Matrix

Finite difference matrix $\mathbf{A}$ diagonal entries equal to 2.

First sub- and super-diagonal entries equal to -1

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & \cdots \\ -1 & 2 & -1 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & -1 & 2 & -1 & 0 & \cdots \\ \cdots & 0 & -1 & 2 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & -2 & 2 \end{pmatrix}$$

except the last sub-diagonal entry, which is -2.

Matrix $\mathbf{A}$ created in `octave` by various techniques.

For simplicity, we us consider the case where the differential equation to be solved is

$$\begin{aligned} -u''(x) &= \sin(x) \text{ for } x \in [0, \pi] \\ u(0) &= u(\pi) = 0. \end{aligned} \tag{43}$$

Then $u(x) = \sin(x)$ for $x \in [0, \pi]$.

To create difference operator A in a sparse format, must specify only the non-zero entries of the matrix, that is, you give a list of triples: (i, j, A_{ij}).

Operation `sparse` amalgamates these triples into a sparse matrix.

`octave` code for solving (43)

```
dx=pi/(N+1);
i(1:N)=1:N;
j(1:N)=1:N;
v(1:N)= 2/(dx*dx);
i(N+(1:(N-1)))=(1:(N-1));
j(N+(1:(N-1)))=1+(1:(N-1));
v(N+(1:(N-1)))= -1/(dx*dx);
i((2*N-1)+(1:(N-1)))=1+(1:(N-1));
j((2*N-1)+(1:(N-1)))=(1:(N-1));
v((2*N-1)+(1:(N-1)))= -/(dx*dx);
A=sparse(i,j,v);
F(1:N)=sin(dx*(1:N));
```

`octave` code for solving (43)

In `octave`, the solution to (42) can be achieved simply by writing

```
U=A\F
```

Critical to use vector constructs in `octave` to insure optimal performance.

Code executes much more rapidly, but code is not shorter, more readible, or less prone to error.

For the approximation (u_n) (38) of the solution u of equation (38) it can be shown [5] that

$$\begin{gathered} \max_{1 \le n \le N} |u(x_n) - u_n| \le C_f h^2 \\ \left(h \sum_n (u(x_n) - u_n)^2 \right)^{1/2} \le C_f h^2 \end{gathered} \tag{44}$$

where C_f is a constant depending only on f.

Experiment with a known u can determine if relationship (44) holds as h is decreased.

If $\log e_h$ is plotted as a function of $\log h$, then the resulting plot should be linear, with a slope of two.

Reality Checks

Consider the boundary value problem (43), that is, $-u'' = f$ on $[0, \pi]$ with $f(x) = \sin x$ and $u(0) = u(\pi) = 0$.

Mean-squared error plotted in Figure 5.

The line $e_h = 0.1h^2$ has been added for clarity.

Thus we see for $h \geq 10^{-4}$, the error diminishes quadratically.

However, when mesh size is much less than 10^{-4}, round-off error causes accuracy to diminish, and error even increases as mesh size is further decreased.

Reality Checks

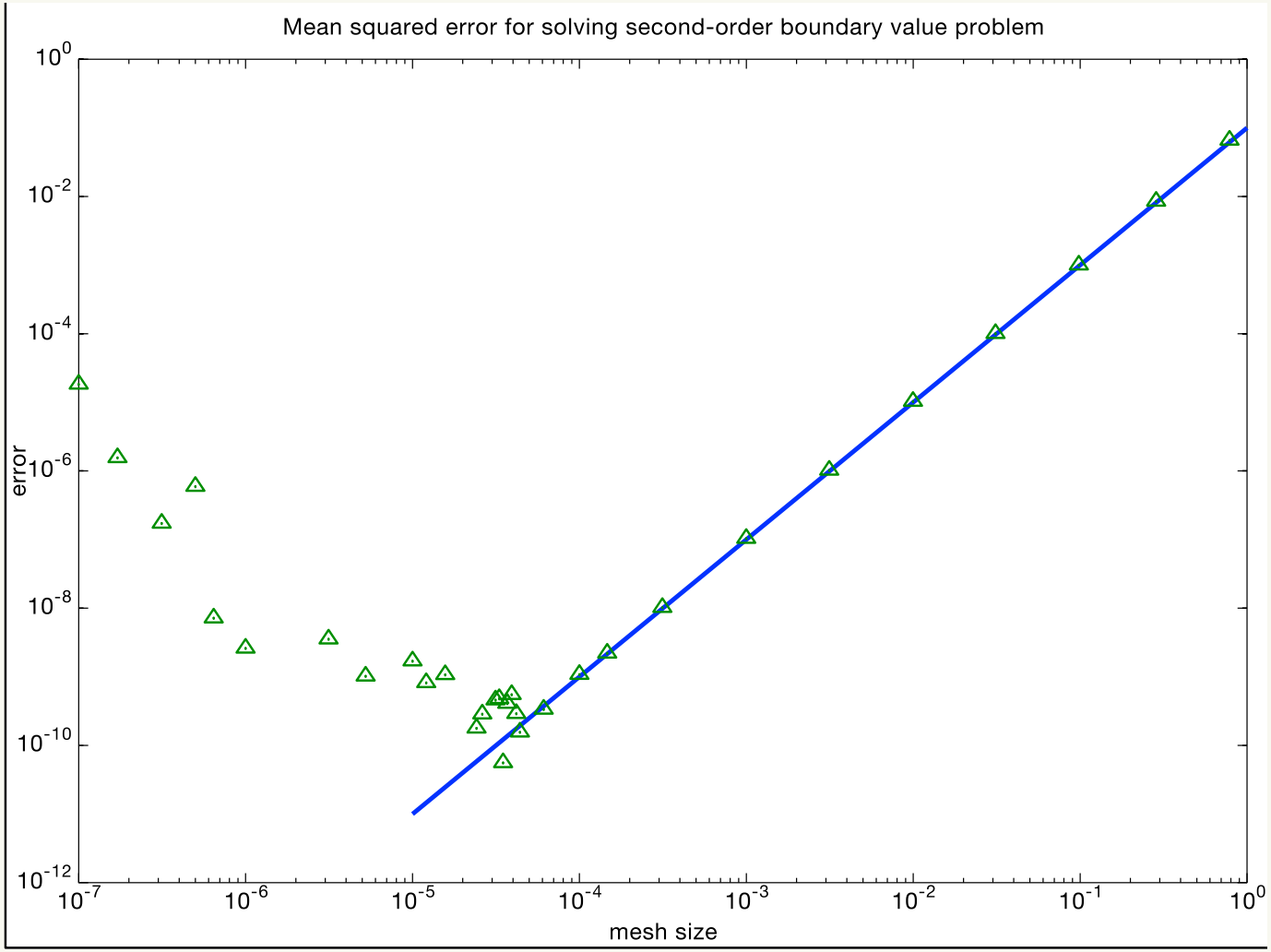


Figure 5: Error in $L^2([0,\pi])$ for the finite difference approximation of the boundary value problem for the differential equation $-u'' = \sin(x)$ on the interval $[0,\pi]$, with boundary conditions $u(0) = u(\pi) = 0$, as a function of the mesh size h. The solid line has a slope of 2 as a reference.

Limits of finite precision arithmetic

All discretization methods can suffer from the effects of finite precision arithmetic.

Typical cause is increasing condition number of linear system $\mathbf{A}$ as h tends to zero.

Condition number grows proportional to $N^2 \approx h^{-2}$.

# of grid points	10	100	1000	10000
mesh size	2.9e-01	3.1e-02	3.1e-03	3.1e-04
condition #	4.8e+01	4.1e+03	4.1e+05	4.1e+07

Table 3: Condition number as a function of N (mesh size $h = \pi/(N+1)$).

Pitfall: Low Accuracy

Error decreases quadratically to a point, then hits wall.

Accuracy then behaves randomly and even decreases as the mesh size is further descreased.

Greatest accuracy achieved is
condition number multiplied by
machine $\epsilon = 2.22 \times 10^{-16}$.

Thus with $N = 10,000$, we cannot expect accuracy better than about 10^{-9}.

The simple way to avoid this difficulty is to avoid very small h values.

The equivalent accuracy can be achieved by using a larger h and a more accurate discretization method.

Indeed, “spectral” methods rely entirely on increasing the degree of approximation on a fixed mesh.

degree	number of grid intervals	error	time
1	10000	1.20e-08	0.137
2	1000	1.14e-09	0.083
4	100	9.57e-11	0.079
8	50	4.08e-10	0.079
16	2	1.02e-09	0.076

Table 4: Error in L^2 norm for the problem $-u'' = \pi^2 \sin(\pi x)$ on $[0, 1]$ with boundary conditions $u(0) = u(1) = 0$ as a function of degree, number of grid points. Time is in seconds.

Same as seen in two-dimensional problem.

degree	mesh number	L^2 error	time (s)
1	1024	2.07e-06	22.5
2	512	2.11e-09	18.4
4	128	4.95e-12	3.0
8	8	3.98e-12	0.13

Table 5: Computational experiments with solving two-dimensional problem. Degree refers to the polynomial degree, mesh number indicates the number of edges along each boundary side, L^2 error is the error measured in the $L^2([0,1]^2)$ norm.

Pitfall: Low Accuracy

Higher-order methods do not solve the problem of round-off error.

Maximum accuracy about same independent of degree of approximation.

Condition number is linked to the accuracy of the approximation.

The condition number of the linear system depends on the largest eigenvalue of the PDE that is resolved.

The better the resolution, the larger the condition number.

Pitfall: Low Accuracy

This is a feature of elliptic systems, like Poisson's equation, whereas in time-stepping methods increasing the accuracy always reduces the effects of round-off errors.

On the other hand, higher-order methods do reduce the run time required to achieve maximal accuracy, as indicated in Table 5.

For this reason, higher-order methods would allow the use of more accurate precision arithmetic at a manageable cost.

Explicit Euler time discretization

Simplest time discretization method for heat equation uses forward (or explicit) Euler difference method:

$$\begin{aligned} u^{n+1}(x) &= u^n(x) + \Delta t \frac{\partial^2 u^n}{\partial x^2}(x) \quad \forall x \in [0,1], \\ u^0(x) &= u_0(x) \quad \forall x \in [0,1] \\ u^n(0) = g_0(n\Delta t) \quad &\text{and} \quad u^n(1) = g_1(n\Delta t) \quad \forall n > 0 \end{aligned} \tag{45}$$

where $u^n(x)$ denotes an approximation to $u(x, n\Delta t)$.

Applying the finite difference or finite element approximation (38) to (45) yields a simple algorithm.

Difficulty: *unstable* unless Δt is sufficiently small.

Using the approximations (45) and (37), we get the following finite difference method for the heat equation:

$$\begin{aligned} u_j^{n+1} &= u_j^n + \frac{\Delta t}{\Delta x^2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right) \\ &= r u_{j-1}^n + (1-2r)u_j^n + r u_{j+1}^n, \end{aligned} \tag{46}$$

where $r = \Delta t/\Delta x^2$.

Here $u(j\Delta x, n\Delta t) \approx u_j^n$.

Very appealing since there are no equations to solve.

However, there is a strict stability limitation.

Stability limitation

Start with the initial condition $u_j^0 = (-1)^j$.

Although this is a rough function, we know that the heat equation should smooth it out.

For the moment, let us ignore boundary conditions and assume that (46) holds for all j and n.

This is equivalent to assuming that $\Omega = \mathbb{R}$.

Then we will prove by induction that

$$u_j^n = (1 - 4r)^n (-1)^j. \tag{47}$$

First, by assumption this holds for $n = 0$.

Using (46) and the induction hypothesis, we have

$$\begin{aligned} u_j^{n+1} &= r u_{j-1}^n + (1-2r) u_j^n + r u_{j+1}^n \\ &= r(1-4r)^n (-1)^{j-1} + (1-2r)(1-4r)^n (-1)^j \\ &\quad + r(1-4r)^n (-1)^{j+1} \\ &= (1-4r)^n (-1)^j (-r + (1-2r) - r) \\ &= (1-4r)^{n+1} (-1)^j, \end{aligned} \tag{48}$$

completing the induction step. So (47) holds $\forall n, j$.

$r = \Delta t / \Delta x^2 > 0$ so $|1-4r| \leq 1$ only for $r \leq 1/2$.

For $r < 1/2$, then u_j^n decreases rapidly to zero.

Approximation blow up

If $r > 1/2$, then $|1 - 4r| > 1$, and (47) blows up exponentially. Most insideously, we have

$$u(x,t) = u(j\Delta x, n\Delta t) \approx u_j^n = (1-4r)^{t/\Delta t}(-1)^j,$$

so the blow up accelerates as $\Delta t \to 0$.

Thus the explicit method (46) for the heat equation is restricted by the stringent requirement

$$\Delta t \leq \frac{\Delta x^2}{2}.$$

Therefore the implicit methods are more efficient in many cases, allowing larger time steps.

The solution of the Jeffrey-Hamel problem can be effected using a difference method for the differential operator as before:

$$-u_{n-1} + 2u_n - u_{n+1} + 4h^2 u_n + 6h^2 u_n^2 = h^2 C \qquad (49)$$

where $u_n \approx u(x_n)$.

Since this system of equations is nonlinear, we cannot solve it directly.

A standard algorithm to use is Newton's method, which can be written as follows.

First, we write the system of equations as $F((u_i)) = 0$ where

$$f_n := -u_{n-1} + 2u_n - u_{n+1} + 4h^2 u_n + 6h^2 u_n^2 - h^2 C \quad (50)$$

Newton's iteration takes the form

$$u \leftarrow u - J_f(u)^{-1} f(u) \quad (51)$$

where $J_f(u)$ denotes the Jacobian of the mapping f evaluated at u.

This can be written in `octave` as follows.

Pitfall: Low Accuracy

Suppose that `A` is defined as before.

Then `f` can be written

```
delta=((n+1)/alf)*((n+1)/alf);
f=delta*A*uhf-4*uhf-6*ujh.*ujh+cvec;
```

where

```
cvec=C*ones(n,1);
```

The Jacobian `J` of `f` is

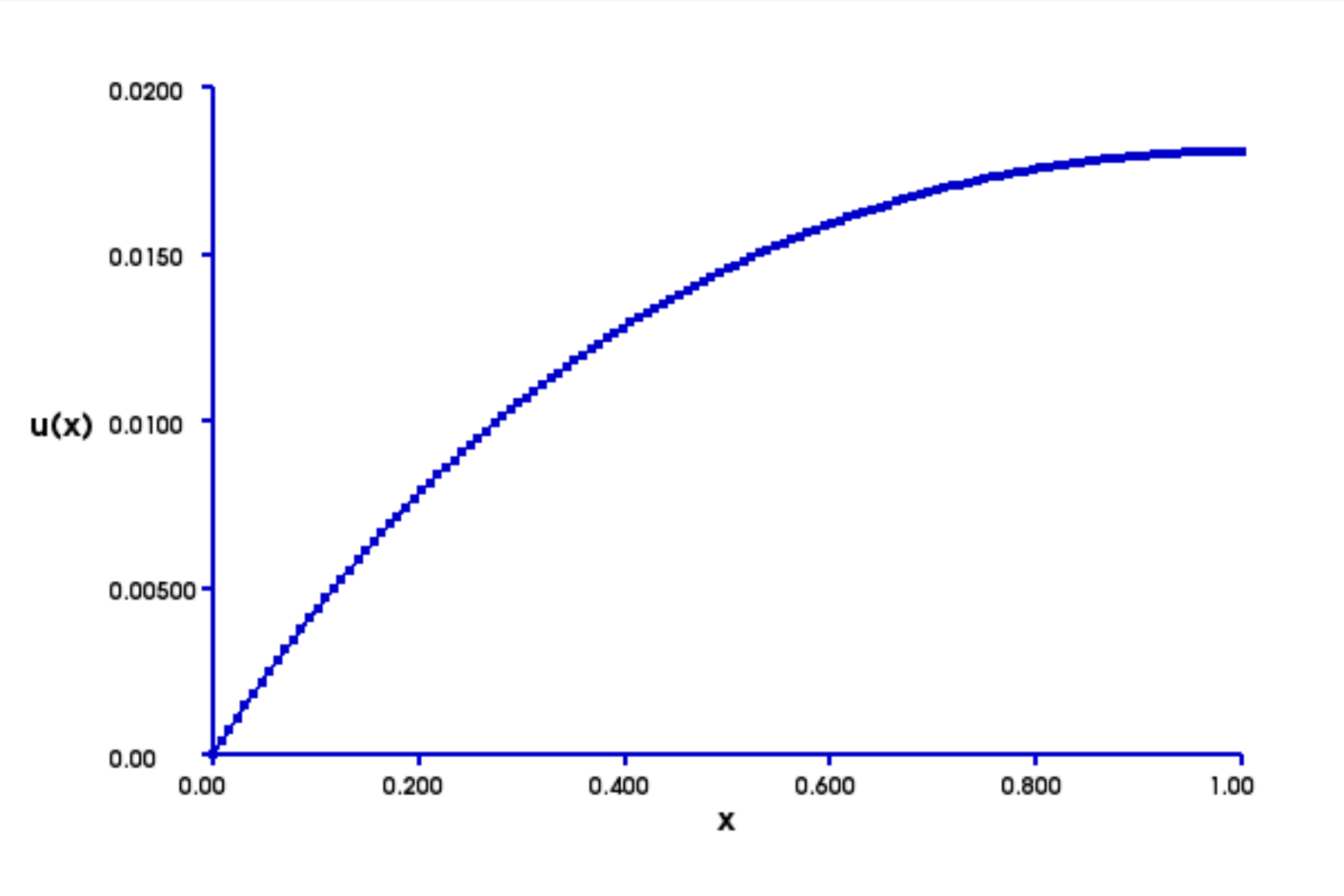
```
J = delta*A - 4*eye(n) - 12*diag(ujh,0);
```

Newton's method

Newton's method takes the following form in `octave`.

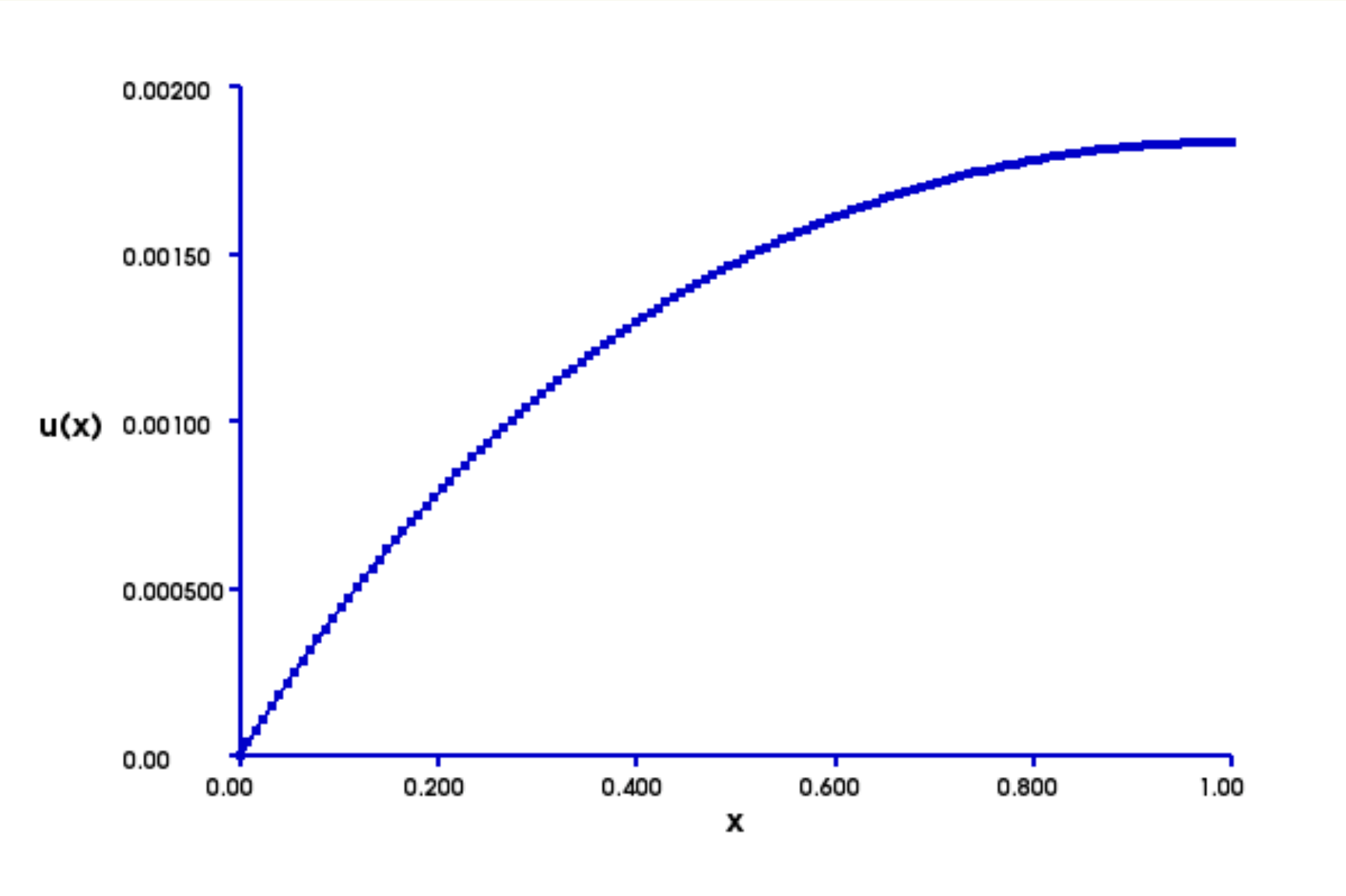
```
JA = delta*A - 4*eye(n);
ujh =- JA\cvec;
enorm = 1;
while (enorm >> .000000000001)
f = JA*ujh - 6*ujh.*ujh + cvec;
J = JA - 12*diag(ujh,0);
x = ujh - J\f;
enorm = norm(ujh-x)/(norm(ujh)+norm(x));
ujh=x;
end
```

Jeffrey-Hamel solution



Solution of the Jeffrey-Hamel equation with $C = 0.1$.

Jeffrey-Hamel solution



Solution of the Jeffrey-Hamel equation with $C = 0.01$.

Classification of PDEs

There are two major classifications of PDEs, one for nonlinearities and one for linear PDEs.

The latter is based on an algebraic trichotomy for second-order differential operators D in two dimensions: elliptic, parabolic, and hyperbolic.

This arises from the analogy with conic sections and their algebraic equations, as indicated in Table 6.

formula	classification	example	equation	variable substitution
$x^2 + y^2 = 1$	elliptic	Laplace	$u_{,xx} + u_{,yy}$	none
$x = y^2$	parabolic	diffusion/heat	$u_{,t} - u_{,yy}$	$x \to t$
$x^2 - y^2 = 1$	hyperbolic	wave	$u_{,tt} - u_{,yy}$	$x \to t$

Table 6: Classification of linear PDEs. “formula” is name for a conic section.

Classification of PDEs: hyperbolic

Examples of elliptic equations are the Laplace/Poisson equations and the variants considered so far.

Heat equation is the prototypical parabolic equation.

In the classification in Table 6, this leaves hyperbolic equations.

Switching variables from y to x, such an equation takes the form

$$u_{tt} - u_{xx} = 0, \tag{52}$$

in one space dimension.

The wave equation

More generally, what is called **The wave equation** is

$$u_{tt} - c^2 \Delta u = 0, \tag{53}$$

in multiple space dimensions, where c is the wave speed and Δ is the Laplacean.

The speed c may always be taken to be 1 in suitable coordinates.

For example, the speed of light in a vacuum is approximately 299792458 metres per second.

Choice of units

A meter is about 3.28084 feet.

So the speed of light in a vacuum is approximately 9.8357e+08 feet per second.

Define **big foot** to be 1.0167 feet (about 12.2 inches), then speed of light is one big foot per nanosecond.

Approximately one giga-big-feet per second.

In either of these units, $c = 1$.

In one of them, time unit is small, and in other length unit is big.

Understanding units

Assume now that units are chosen so that $c = 1$.

Grace Hopper[1] famously displayed in her talks a piece of copper wire whose length corresponded to the distance an electrical signal traveled in a nanosecond.

Her point was to emphasize the need for careful programming, but it also showed that $c = 1$ makes sense in a computer.

[1] Grace Brewster Murray Hopper (1906–1992) was an early advocate for automating programming via the use of compilers to translate human-readable descriptions into machine code.

Other wave equations

There are other equations of higher order that do not fit the classification in Table 6, such as the dispersive Airy equation $u_t + u_{xxx} = 0$.

Using a more sophisticated classification [1], the Stokes equations are an elliptic system.

So we must consider Table 6 as just a starting point for understanding the differences between different PDEs.

But it does suggest a new equation to consider which is intimately related to wave motion.

The one-dimensional wave operator factors, so that (52) can be written

$$\begin{aligned} 0 = u_{tt} - u_{xx} &= \Big(\frac{\partial}{\partial t} - \frac{\partial}{\partial x}\Big)\Big(\frac{\partial}{\partial t} + \frac{\partial}{\partial x}\Big)u \\ &= \Big(\frac{\partial}{\partial t} - \frac{\partial}{\partial x}\Big)\big(u_t + u_x\big). \end{aligned} \tag{54}$$

In particular, $u(x,t) = f(t-x)$ solves (54) for any function f of one variable, since $u_t(x,t) = f'(t-x)$ and $u_x(x,t) = -f'(t-x)$, and thus $u_t + u_x = 0$.

Thus solutions are constant on lines $x(t) = t$.

Consist of just translating f to right at constant speed.

Waves go in both directions

Similarly, by re-ordering the factorization (54), we conclude that another family of solutions consist of translations to the left, without change of shape.

Such solutions are easy to visualize, and are markedly different from the behavior for the heat equation, or elliptic equations.

In particular, it appears at least formally that smoothness of f does not matter, and that even discontinuous solutions would be allowed.

d’Alembert’s formula for solutions of wave equation:

$$u(x,t) = \tfrac{1}{2} f(t-x) + \tfrac{1}{2} f(t+x) + \frac{1}{2} \int_{x-t}^{x+t} g(s)\, ds, \quad (55)$$

where $g(x) = u_t(x,0)$ and $f(x) = u(x,0)$.

The contributions of d’Alembert are memorialized in the name the wave operator

$$\Box u = u_{tt} - \Delta u,$$

known as the **d’Alembertian operator**.

d'Alembert draws information from the **light cone**:

$$u(x,t) = \tfrac{1}{2}f(t-x) + \tfrac{1}{2}f(t+x) + \frac{1}{2}\int_{x-t}^{x+t} g(s)\,ds,$$

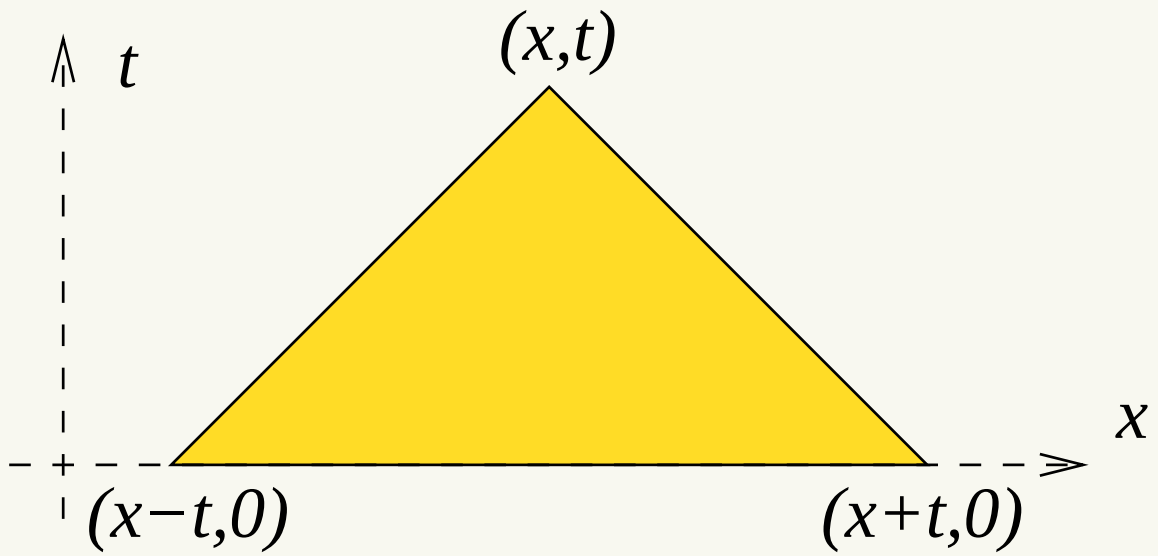


Figure 6: The light cone for the wave equation.

Using the light cone

Although corresponding relationship between data and solution is more complex in higher spatial dimensions, concept of light cone generalizes naturally.

Since waves tend to propagate without change of shape, the spatial domain Ω is often infinite.

In such cases, we truncate Ω for computational purposes.

The fact that information comes only from the light cone allows us to estimate the required size of the computational domain accurately.

One-way propagation

d'Alembert formula seems to imply waves always propagate in both directions.

But our splitting of the wave operator implied that there are solutions of the form $u_{\pm}(x,t) = f(x \pm t)$.

Suppose that $v(x,t) = f(x+t)$.

Then $v_t(x,0) = f'(x)$.

So now consider the case where f is given and $g(x) = f'(x)$ for all x.

Either-way propagation

For $g(x) = f'(x)$ for all x, (55) gives

$$2u(x,t) = f(x-t) + f(x+t) + \int_{x-t}^{x+t} f'(s)\,ds$$
$$= f(x-t) + f(x+t) + (f(x+t) - f(x-t)) = 2f(x+t).$$

Thus $u = v$ as expected. Similarly, if $g(x) = f'(-x)$, then

$$2u(x,t) = f(x-t) + f(x+t) + \int_{x-t}^{x+t} f'(-s)\,ds$$
$$= f(x-t) + f(x+t) + (f(x-t) - f(x+t)) = 2f(x-t).$$

So we get the expected one-way solutions by adjusting the initial impulse g.

Variational form of wave equation

Since waves tend to propagate without change of shape, the spatial domain Ω is often infinite.

We truncate Ω for computational purposes.

Using standard algorithm, following heat equation, we obtain the variational expression

$$(u_{tt}, v)_{L^2(\Omega)} + a(u, v) = 0 \quad \forall v \in H^1(\Omega), \tag{56}$$

where

$$a(v, w) = \int_\Omega v'(x) w'(x)\, dx. \tag{57}$$

What is different is second-order time derivative.

We can approximate this, for example, via

$$u_{tt} \approx \frac{u^{n+1} - 2u^n + u^{n-1}}{\tau^2},$$

where $\tau > 0$ is the time step, and u^j denotes an approximation to $u(j\tau)$. We can use this to define the vatiational formulation

$$(u^{n+1}, v)_{L^2(\Omega)} - 2(u^n, v)_{L^2(\Omega)} + (u^{n-1}, v)_{L^2(\Omega)} + \tau^2 a(u^{n+1}, v) = 0,$$

where τ is the time step. Thus we solve

$$\begin{aligned}(u^{n+1}, v)_{L^2(\Omega)} + \tau^2 a(u^{n+1}, v) &= 2(u^n, v)_{L^2(\Omega)} \\ &\quad - (u^{n-1}, v)_{L^2(\Omega)} \quad \forall v \in H^1(\Omega).\end{aligned} \tag{58}$$

Time discretization of wave equation

We immediately see that we can not solve the wave equation knowing just the initial conditions $u(x, 0)$.

In addition, we need to know $u_t(x, 0)$ and we can use this to get a good approximation to u^{-1}.

Unfurtunately, the time-stepping scheme (58) is only first-order accurate in time.

Another time-stepping scheme, advocated in [6], is

$$(u^{n+1}, v)_{L^2(\Omega)} - 2(u^n, v)_{L^2(\Omega)} + (u^{n-1}, v)_{L^2(\Omega)} = -\tau^2\big(\theta a(u^{n+1}, v) + (1-2\theta)a(u^n, v) + \theta a(u^{n-1}, v)\big), \tag{59}$$

where $\theta \in [0, 1]$. Thus we solve

$$(u^{n+1}, v)_{L^2(\Omega)} + \theta\tau^2 a(u^{n+1}, v) = 2(u^n, v)_{L^2(\Omega)} - (u^{n-1}, v)_{L^2(\Omega)} - \tau^2\big((1-2\theta)a(u^n, v) + \theta a(u^{n-1}, v)\big), \quad \forall v \in H^1(\Omega). \tag{60}$$

We can see the rationale for this approach by doing a Taylor expansion.

Taylor expansion

$$u((n \pm 1)\tau) = u(n\tau) \pm \tau u_t(n\tau) + \tfrac{1}{2}\tau^2 u_{tt}(n\tau) \pm \frac{\tau^3}{6} u_{ttt}(n\tau) + \mathcal{O}\left(\tau^4\right). \tag{61}$$

Adding terms and dividing by τ^2, and again making the correspondence $u^j \approx u(j\tau)$, we find

$$\frac{u^{n+1} - 2u^n + u^{n-1}}{\tau^2} = u_{tt}(n\tau) + \mathcal{O}\left(\tau^2\right).$$

Similarly, multiplying (61) by θ and adding gives

$$\theta u((n+1)\tau) + (1-2\theta)u(n\tau) + \theta u((n-1)\tau) = u(n\tau) + \theta\tau^2 u_{tt}(n\tau) + \mathcal{O}\left(\tau^4\right). \tag{62}$$

Thus for any value of θ we get a second-order approximation:

$$\begin{aligned}\frac{u^{n+1}-2u^n+u^{n-1}}{\tau^2}+\theta u^{n+1}+(1-2\theta)u^n+\theta u^{n-1}\\ =u_{tt}(n\tau)+u(n\tau)+\mathcal{O}\left(\tau^2\right),\end{aligned} \tag{63}$$

and truncation error is decreasing in θ.

If we take $\theta=0$ we get an explicit scheme with time-step restrictions.

In [6], the choice $\theta=1/4$ was advocated for stability reasons.

Higher accuracy

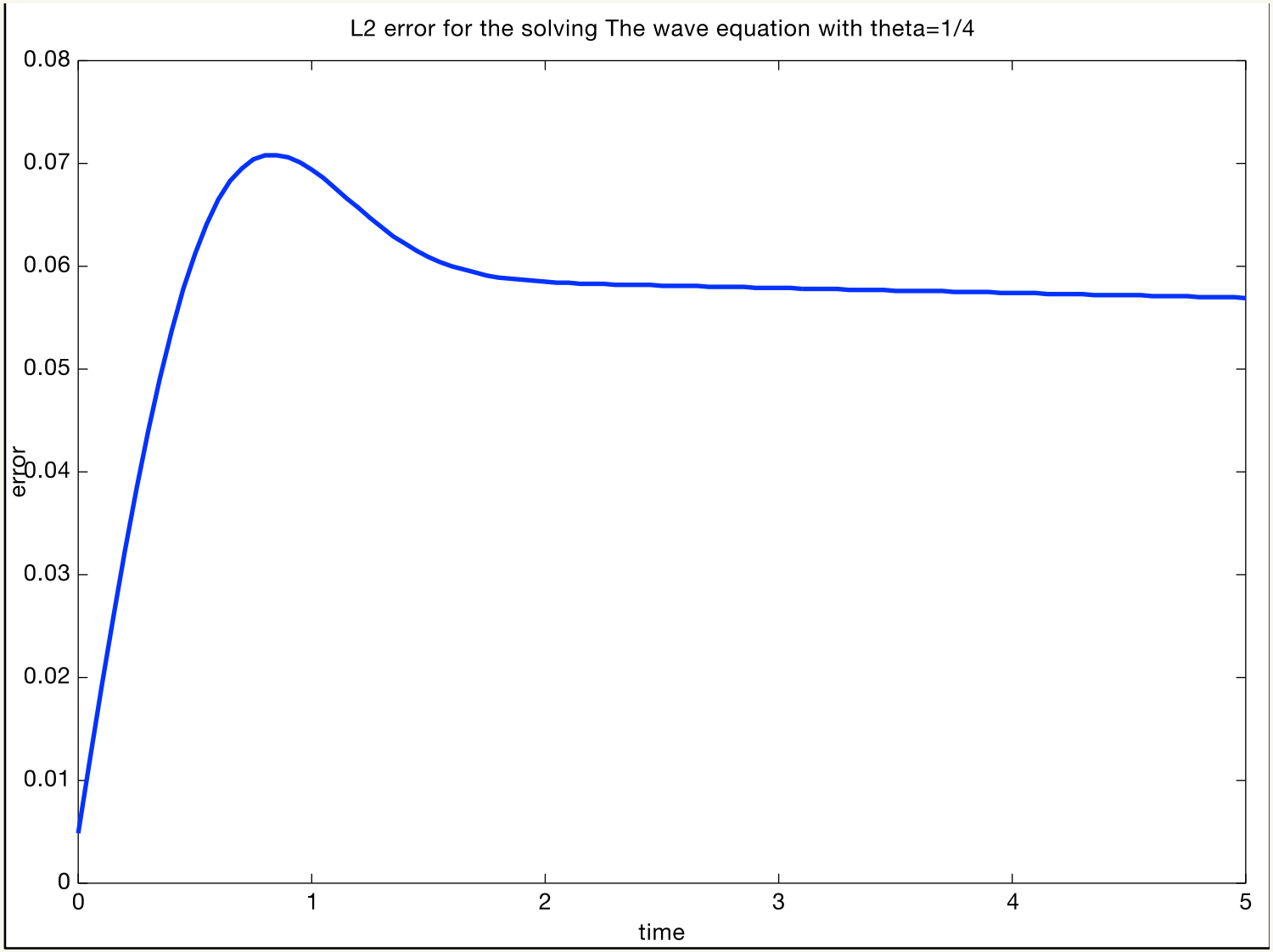


Figure 7: Error in $L^2(\Omega)$ as function of time for θ scheme with $\theta = 1/4$, $\Delta t = 0.05$, $L = 10$, $T = 5$, $M = 2000$ mesh points, using piecewise linears.

Computational example

Start the simulation (in one space dimension) with

$$u_0(x) = e^{-x^2}$$

on domain $\Omega = [-L, L]$. Take L as large as necessary.

Take $u_t(x, 0) = 0$. Then the exact solution follows from d'Alembert's formula:

$$u(x, t) = \tfrac{1}{2}e^{-(t-x)^2} + \tfrac{1}{2}e^{-(t+x)^2}. \tag{64}$$

Error as a function of time is indicated in Figure 7.

Error increases rapidly initially but then decreases.

Higher dimensions

The conversion to higher space dimensions is notationally trivial.

We have written our bilinear form

$$a(u, v) = \int_{\Omega} \nabla u \cdot \nabla v \, d\mathbf{x}$$

which makes sense in any number of dimensions.

Thus it is just a matter of defining the spatial domain to be multi-dimensional and waiting a bit longer for the answers to appear.

Consider solutions U of wave equation (53) given by

$$U(\mathbf{x}, t) = u(\mathbf{x})e^{\iota kt},$$

where $k \in \mathbb{R}$ and $\iota = \sqrt{-1}$. Then

$$\Delta U(\mathbf{x}, t) = (\Delta u(\mathbf{x}))e^{\iota kt} \quad \text{and} \quad U_{tt}(\mathbf{x}, t) = -k^2 u(\mathbf{x})e^{\iota kt}.$$

Thus the wave equation

$$U_{tt}(\mathbf{x}, t) - \Delta U(\mathbf{x}, t) = f(\mathbf{x})e^{\iota kt}$$

is satisfied if

$$-k^2 u(\mathbf{x}) - \Delta u(\mathbf{x}) = f(\mathbf{x}). \tag{65}$$

Helmholtz equation

Equation (65) is known as **Helmholtz equation**.

Solving this (with boundary conditions) can be touchy since corresponding variational form not coercive.

If bothered by complex varaible ansatz for U, replace it with real counterparts, e.g.,

$$U^1(\mathbf{x},t) = u(\mathbf{x})\cos(kt), \qquad U^2(\mathbf{x},t) = u(\mathbf{x})\sin(kt),$$

but the conclusion is the same:

U^i's solve the wave equation if (65) holds.

Dispersive waves

Many types of wave behavior can be characterized as **dispersive**.

One example is water waves.

These are the waves that a child first sees.

The main characteristic of these waves is the way that they spread as time evolves.

Unlike the behavior we saw for The wave equation where waveforms just translate linearly in time as indicated in (55).

Dispersive waves

When the wave length λ is long and the amplitude a is small, compared to the water depth, approximate equations can be derived for surface wave behavior.

More precisely, define the Stokes parameter $S = a\lambda^2/d^3$, where d is the (assumed uniform) depth of the water.

When $S = \mathcal{O}(1)$, then models for one-way propagation include the KdV equation [11, 3]

$$u_t + u_x + 2uu_x + u_{xxx} = 0,$$

where u is the wave height.

The waves are assumed to be propagating in the x-direction, and constant in the y-direction.

We see that this equation is a nonlinear perturbation of the one-way wave equation $u_t + u_x = 0$.

Another, equivalent [3], model is

$$u_t + u_x + 2uu_x - u_{xxt} = 0. \tag{66}$$

The two equations related as perturbations of the basic wave equation $u_t + u_x = 0$.

Suggests the substitution $u_t \approx -u_x$, leads to $u_{xxt} \approx -u_{xxx}$.

Simulations using equation (66) have been compared with laboratory experiments [2].

Solutions of (66) can be approximated numerically [4] by writing it in the form

$$\left(I - \frac{\partial^2}{\partial x^2}\right) u_t = -u_x - 2uu_x = -\left(u + u^2\right)_x, \tag{67}$$

where I denotes the identity operator.

The left-hand side is a familiar elliptic operator, so we can write a variational equation

$$\begin{aligned} a(u_t, v) &= -(u_x + 2uu_x, v)_{L^2(\Omega)} \\ &= -((1 + 2u)u_x, v)_{L^2(\Omega)}, \end{aligned} \tag{68}$$

where

$$a(v, w) = \int_\Omega v(x)w(x) + v'(x)w'(x)\, dx. \tag{69}$$

A variety of time-stepping schemes can be used to discretize (68).

One family of schemes that work well [4] is the **Runge-Kutta** schemes.

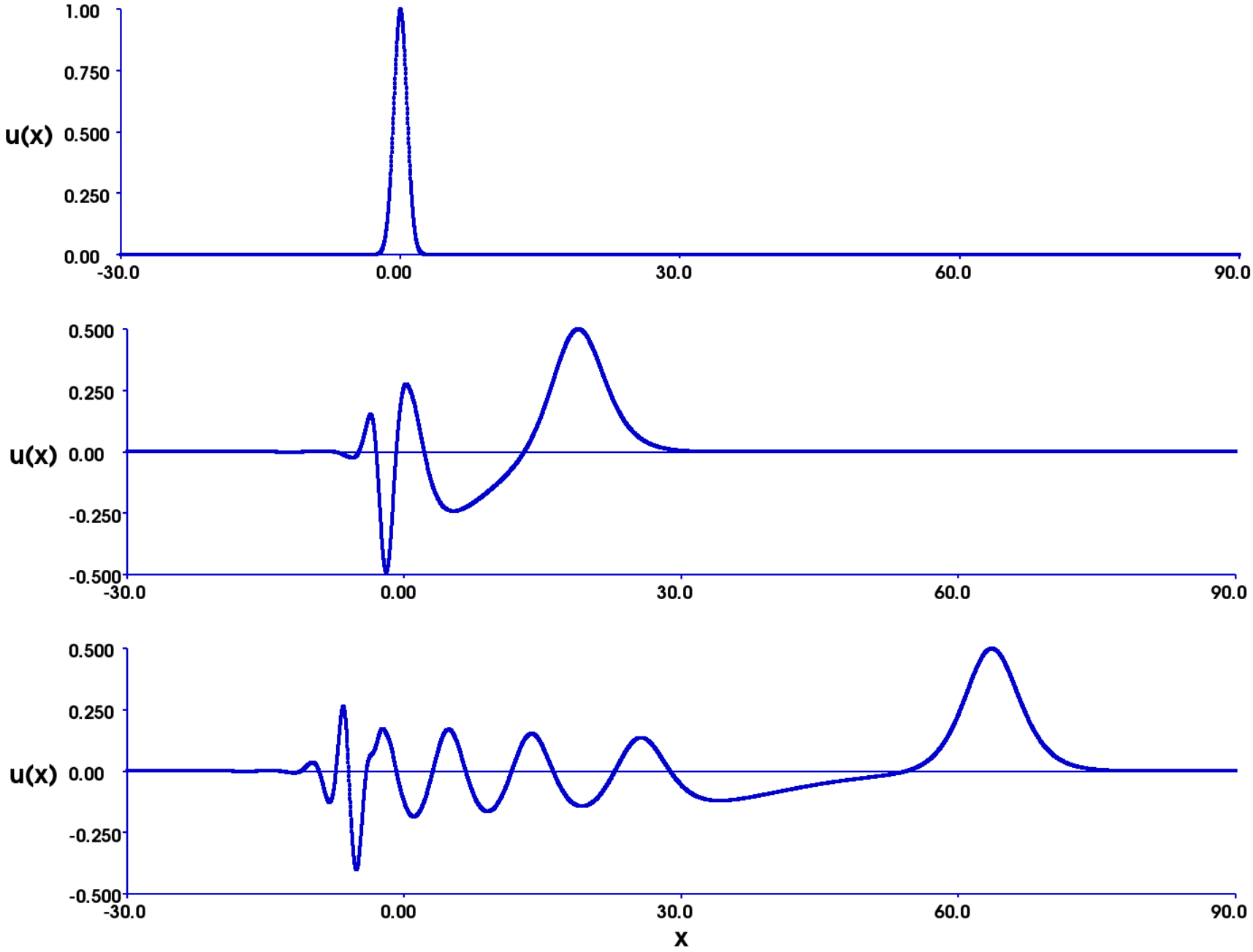
These schemes can be high order yet not require previous time values of the solution.

Often used as start-up schemes for other schemes that do utilize previous values, as do the θ schemes (59).

Simplest is **modified Euler** scheme: $\forall v \in V$

$$\begin{aligned} a(\hat{u}^{n+1}, v) &= a(u^n, v) - \Delta t \left((1 + 2u^n)u_x^n, v\right)_{L^2(\Omega)} \\ a(u^{n+1}, v) &= a(u^n, v) - \tfrac{1}{2}\Delta t \Big(\left((1 + 2u^n)u_x^n, v\right)_{L^2(\Omega)} \\ &\qquad + \left((1 + 2\hat{u}^{n+1})\hat{u}_x^{n+1}, v\right)_{L^2(\Omega)}\Big). \end{aligned} \tag{70}$$

Time-stepping schemes



An example

The first step in (70) is called the **predictor step** and the second step in (70) is called the **corrector step**.

Figure presents example of this scheme with piecewise linear approximation in space.

The initial data $u(x, 0) = e^{-x^2}$ evolves into a complex wave form that spreads out as it travels to the right, and even has a part that moves slowly to the left.

The wave speed is approximately $1 + 2u$, and in 25 time units the leading wave moves over 60 units to the right.

A linear wave (that is, if u were small) would have moved only 25 units to the right.

An example

A common measure of wave length for a complex wave shape is its width at half height.

For the function $f(x) = e^{-x^2}$, this is the point x where $e^{-x^2} = 1/2$.

Thus $-x^2 = \log(1/2)$, so $x = \sqrt{\log 2} = 0.83\ldots$.

Thus the initial wave length is less than 2, whereas the leading wave at $T = 25$ has a width that is about 7 units, over 4 times larger.

The dispersion causes the initial narrow wave to spread.

Solitary waves

Despite the fact that there is both dispersion and nonlinearity in (67), it is possible to find special wave forms that propagate without change of shape.

These are called **solitary waves**.

For the equation (67), they take the form

$$u(x,t) = \frac{3}{2} A \,\mathrm{sech}^2\left(\frac{1}{2}\sqrt{\frac{A}{A+1}}(x-(1+A)t)\right), \qquad (71)$$

where $A > 0$ can be any positive value. Recall that

$$\mathrm{sech}(z) = \frac{2}{e^z + e^{-z}} = \frac{1}{\cosh(z)}.$$

References

[1] Shmuel Agmon, Avron Douglis, and Louis Nirenberg. Estimates near the boundary for solutions of elliptic partial differential equations satisfying general boundary conditions II. *Communications on Pure and Applied Mathematics*, 17(1):35–92, 1964.

[2] J. L. Bona, W. G. Pritchard, and L. R. Scott. An evaluation of a model equation for water waves. *Philos. Trans. Roy. Soc. London Ser. A 302*, pages 457–510, 1981.

[3] J. L. Bona, W. G. Pritchard, and L. R. Scott. A comparison of solutions of two model equations for long waves. In *Fluid Dynamics in Astrophysics and Geophysics, N. R. Lebovitz, ed.,*, volume 20, pages 235–267. Providence, Rhode Island: American Mathematical Society, 1983.

[4] J. L. Bona, W. G. Pritchard, and L. R. Scott. Numerical schemes for a model for nonlinear dispersive waves. *J. Comp Phys.*, 60:167–186, 1985.

[5] Susanne C. Brenner and L. Ridgway Scott. *The Mathematical Theory of Finite Element Methods*. Springer-Verlag, third edition, 2008.

[6] Todd Dupont. L^2-estimates for Galerkin methods for second order hyperbolic equations. *SIAM Journal on Numerical Analysis*, 10(5):880–889, 1973.

[7] Jean Baptiste Joseph Fourier. *Théorie analytique de la chaleur*. Firmin Didot, Paris, 1822.

[8] Stephen W. Hawking and Michael Jackson. *A brief history of time*. Bantam New York, NY, 2008.

[9] L. Ridgway Scott. *Numerical Analysis*. Princeton Univ. Press, 2011.

[10] V. Thomee. *Galerkin Finite Element Methods for Parabolic Problems*. Springer Verlag, 1997.

[11] J. Yan and C.W. Shu. A local discontinuous galerkin method for kdv type equations. *SIAM Journal on Numerical Analysis*, pages 769–791, 2003.